Image Formation from a Large Sequence of RAW Images : performance and accuracy

Thibaud Briand

École Doctorale Mathématiques & Sciences et Technologies de l'Information et de la Communication

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PARIS-EST
Domaine : Traitement du Signal et des Images

présentée par **Thibaud BRIAND**
pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ PARIS-EST

Image Formation from a Large Sequence of RAW Images: Performance and Accuracy

---

Soutenue publiquement le 13 novembre 2018 devant le jury composé de :

Isabelle BLOCH	Télécom ParisTech	Examinatrice
Pascal MONASSE	École des Ponts ParisTech	Directeur de Thèse
Jean-Michel MOREL	Université Paris-Saclay	Directeur de Thèse
Pablo MUSÉ	Universidad de la Repùblica	Rapporteur
Philippe THÉVENAZ	École Polytechnique Fédérale de Lausanne	Rapporteur
Patrick VANDEWALLE	KU Leuven	Rapporteur

École des Ponts ParisTech
LIGM-IMAGINE
6, Av Blaise Pascal - Cité Descartes
Champs-sur-Marne
77455 Marne-la-Vallée cedex 2
France

Université Paris-Est Marne-la-Vallée
École Doctorale Paris-Est MSTIC
Département Études Doctorales
6, Av Blaise Pascal - Cité Descartes
Champs-sur-Marne
77454 Marne-la-Vallée cedex 2
France

Remerciements

Je remercie mes rapporteurs, Pablo Musé, Philippe Thévenaz et Patrick Vandewalle, pour le temps consacré à la relecture de ce manuscrit. Merci également à Isabelle Bloch d'avoir accepté d'évaluer cette thèse.

Je tiens évidemment à remercier du plus pronfond du coeur mes deux directeurs de thèse, Pascal Monasse et Jean-Michel Morel, pour m'avoir accompagné, guidé et tant appris durant ces quatre années. Mon travail aurait été bien différent sans vous et certainement qu'une grande partie n'aurait même pas vu le jour. Merci pour votre disponibilité, votre patience et votre humanité.

Je remercie Gabriele, Javier, Loïc, Mauricio et Sandra pour leur contribution ou bien tout simplement l'aide qu'ils ont pu apporter à mon/notre travail lors de la rédaction des articles.

Je tiens à remercier Guillermo Sapiro et son équipe pour nous avoir accueillis, nous la Duke Team, dans les locaux de l'université de Duke. Ces deux séjours d'un mois ont grandement contribué à l'avancement de mon travail de thèse.

Ce sont dans les locaux du CMLA que j'étais principalement lors de cette thèse. Les captivants et nourrissants GTTI, avec entre autres Arthur, Carlo, Enric (merci pour ton aide et tes codes !), Frédéric et Rafael, ont rythmé mes semaines. Le CMLA c'est aussi de nombreux doctorants, en général situés dans une seule et même pièce (celle du fond où il y a toujours quelqu'un qui parle). Merci à vous les Axel, Charles, Claire, Jérémy (un thé ?), Mariano, Marie, Pierre, Sébastien, Tina, Tristan (espèce de truffe), Valentin et Xavier. En quatre ans j'ai colonisé plusieurs bureaux et je les ai partagés avec grand plaisir avec Hector, Matthias, Miguel, Nelson, Pablo, Thibaud (pas moi l'autre) et Zhijin. Un grand merci à eux d'avoir supporté/subi mes siestes digestives.

Le peu de temps que j'ai pu passer au LIGM a toujours été fort agréable. Cela ne l'aurait pas été sans les merveilleux repas au MRS (ça change du CROUS de Cachan...) avec Benjamin, Bertrand, Francisco, Guillaume, Laura, Loïc, Marina, Matthieu, Renaud, Sergeï, Shell, Spyros, Thibault et Yohann (merci pour les templates !).

Je remercie profondément Virginie, Véronique et Alina pour toute l'aide qu'elles ont pu m'apporter. Vous avez toujours été très efficaces pour gérer mes soucis administratifs. Du côté des Ponts, je remercie de la même manière Sylvie Cach, Fatima De Freitas, Cyril Nicaud, Corinne Palescandolo et Julie Szymkowicz.

Durant ma thèse, j'ai effectué mon monitorat à l'ENS Cachan. Ce fut un plaisir de travailler avec les Alains (Trouvé et Durmus), Frédéric, Laure et Sandrine. J'ai beaucoup appris avec vous sur mon futur métier d'enseignant. Je tiens globalement à remercier tous les professeurs que j'ai eu durant ma scolarité et, en particulier, Nicolas Neveu et Nicolas Tosel qui ont fortement contribués à mon orientation vers les mathématiques.

D'un côté plus personnel, je remercie chaleureusement tous ceux qui ont été à mes côtés durant cette thèse. Je commence par les personnes de mon club de volley VB14 (ex-Femina a.k.a le club de Babak). Merci à la Team BBK d'avoir su me changer les idées et me détendre (sur et en dehors des terrains). Côté volley (mais pas que !) il y a aussi la team ENS Cachou

avec Benoit, Chloé (et son Mickayelle) et Paul. Que de bons moments et de souvenirs en tournoi ou bien en vacances avec vous.

Je tiens également à remercier la fine équipe des doctorants cachanais: Chris (pas matheux mais il compte comme), Keurcien, Matthieu, Maxime, Pierre, Romain et William. Chacun sa thèse (plus ou moins vite achevée), chacun ses soucis administratifs mais tous dans le même bateau. Un gros bisou à vous tous !

Parmi ceux que j'ai déjà cités il y en a deux qui sortent un peu du lot: mes colocs. D'abord Maxime, qui pendant trois ans, a refusé qu'on ait un chat mais qui pour compenser faisait des bons petits plats. Ensuite Benoit qui, depuis un an, me nourrit également à merveille (coïncidence ?) et surtout veille à ce que j'ingurgite mon quota hebdomadaire de gingembre. Un gros bisou à vous deux et vivement que tu reviennes en France, Maxime.

Pour finir je vais évidemment remercier toute ma famille. Je pense que je n'ai jamais su vous expliquer ce que je faisais. Vous m'avez souvent demandé "Comment ça va la thèse ?" et je répondais de manière enthousiaste "Hmm, ça va...". Mes derniers remerciements seront pour ceux que je ne vois pas forcément le plus fréquemment mais qui sont les plus importants à mes yeux: mes parents.

Résumé

Le but de cette thèse est de créer une image couleur de haute qualité, contenant un faible niveau de bruit et d'aliasing, à partir d'une grande séquence (par exemple des centaines ou des milliers) d'images RAW prises avec une caméra grand public. Il s'agit d'un problème complexe nécessitant d'effectuer à la volée du dématriçage, du débruitage et de la super-résolution. Les algorithmes existants produisent des images de haute qualité, mais le nombre d'images d'entrée est limité par des coûts de calcul et de mémoire importants.

Dans cette thèse, nous proposons un algorithme de fusion d'images qui traite les images séquentiellement de sorte que le coût mémoire ne dépend que de la taille de l'image de sortie. Après un pré-traitement, les images mosaïquées (ou CFA) sont alignées dans un système commun de coordonnées en utilisant une méthode de recalage en deux étapes que nous introduisons. Ensuite, une image couleur est calculée par accumulation des données irrégulièrement échantillonnées en utilisant une régression à noyau classique. Enfin, le flou introduit pendant la régression est supprimé en appliquant l'inverse du filtre équivalent asymptotique correspondant (que nous introduisons).

Au cours de la thèse, chaque étape de la méthode est introduite et analysée séparément. Les performances et la précision sont évaluées sur des données synthétiques et réelles. Cette procédure permet de contrôler l'erreur à chaque étape, mais également d'analyser correctement les améliorations proposées des méthodes existantes.

La première partie de la thèse est consacrée aux méthodes d'interpolation. En effet, la génération de données synthétiques nécessite une méthode d'interpolation et le contrôle de l'erreur d'interpolation est crucial pour analyser les performances de notre méthode. Nous étudions en détail les méthodes d'interpolation par polynôme trigonométrique et par B-spline. Nous tirons de cette étude de nouvelles méthodes d'interpolation affinées que nous utilisons pour générer des données synthétiques dans les deux parties suivantes.

La deuxième partie de la thèse traite des méthodes de recalage. Nous commençons par améliorer une méthode existante, à savoir l'algorithme *inverse compositional*. Ensuite, nous considérons le recalage d'images mosaïquées. Nous proposons une méthode en deux étapes qui convertit les images en images non mosaïquées par filtrage passe-bas avant d'effectuer le recalage avec une méthode préexistante (généralement l'algorithme *inverse compositional* modifié).

La troisième partie de la thèse est consacrée aux méthodes de fusion d'images. Premièrement, nous étudions l'ajustement des données irrégulièrement échantillonnées par des méthodes de régression à noyau. Nous montrons que le système linéaire impliqué peut être obtenu par accumulation des données. Pour la régression à noyau classique, nous introduisons le filtre équivalent asymptotique, une approximation du filtre équivalent réel qui explique le flou introduit. À partir de cette étude, nous développons un algorithme de fusion d'images, rapide et à faible coût mémoire, qui est conçu pour un grand nombre d'images. Les paramètres optimaux et la performance de notre méthode sont méticuleusement évalués expérimentalement. Enfin, cette méthode est adaptée aux images RAW puisque nous proposons l'algorithme de formation d'image décrit précédemment.

Nous trouvons que pour une grande séquence d'images RAW, notre méthode de formation d'images améliore avec succès la résolution tout en ayant un bruit résiduel décroissant comme prévu. Nous avons obtenu des résultats similaires à ceux de méthodes plus lentes et plus gourmandes en mémoire.

Cette méthode ouvre la voie à la formation d'images en temps réel à partir d'images RAW. Les images sont traitées séquentiellement pendant les étapes de pré-traitement, de recalage et d'accumulation, de sorte que des calculs à la volée peuvent être effectués.

Abstract

The aim of this thesis is to build a high-quality color image, containing a low level of noise and aliasing, from a large sequence (e.g. hundreds or thousands) of RAW images taken with a consumer camera. This is a challenging issue requiring to perform on the fly demosaicking, denoising and super-resolution. Existing algorithms produce high-quality images but the number of input images is limited by severe computational and memory costs.

In this thesis we propose an image fusion algorithm that processes the images sequentially so that the memory cost only depends on the size of the output image. After a preprocessing step, the mosaicked (or CFA) images are aligned in a common system of coordinates using a two-step registration method that we introduce. Then, a color image is computed by accumulation of the irregularly sampled data using classical kernel regression. Finally, the blur introduced by the underlying kernel regression is removed by applying the inverse of the corresponding asymptotic equivalent filter (that we introduce).

During the dissertation each step of the method is introduced and analyzed separately. The performance and the accuracy are evaluated on synthetic and real data. This procedure allows for a control of the error at each step but also for a proper analysis of the proposed improvements of existing methods.

The first part of the dissertation is devoted to interpolation methods. Indeed, generating synthetic data requires an interpolation method and controlling the interpolation error is crucial for analyzing the performance of our method. We study in detail the trigonometric polynomial and B-spline interpolation methods. We derive from this study new fine-tuned interpolation methods that we use to generate synthetic data in the following two parts.

The second part of the dissertation deals with registration methods. We begin by improving an existing intensity-based method, namely the inverse compositional algorithm. Then, we consider the registration of mosaicked images. We propose a two-step method that converts the images into non-mosaicked images by lowpass filtering before performing the registration with a pre-existing method (typically the modified inverse compositional algorithm).

The third part of the dissertation is dedicated to image fusion methods. First, we study the irregularly sampled data fitting by kernel regression. We show that the linear system involved can be obtained by a data accumulation. For classical kernel regression, we introduce the asymptotic equivalent filter, an approximation of the actual equivalent filter that explains the blur introduced by the method. From this study, we derive a fast and low memory image fusion algorithm that is designed for a large number of images. The optimal parameters and the performance of our method are meticulously evaluated experimentally. Finally, this method is adapted to RAW images as we propose the image formation algorithm described previously.

We find that for a large sequence of RAW images, our image formation method from RAW images successfully performs super-resolution with a residual noise decreasing as expected. We obtained results similar to those obtained by slower and memory greedy methods.

This efficient method opens the way to real time image formation from RAW images. The images are processed sequentially during the preprocessing, registration and accumulation steps so that on the fly computations can be performed.

On peut fusionner mille images une fois.
On peut fusionner mille images mille fois.
Mais on ne peut pas fusionner une image une fois.

Pause Thé n°21228239, 07/2018, Anonyme.

Contents

List of Algorithms

Chapter 1

Introduction (en français)

L'objectif général de cette thèse est d'obtenir des images de haute qualité par traitement numérique. La qualité d'une image peut être améliorée par des méthodes mono-image (par exemple [18, 25] pour le débruitage, [47, 67, 135] pour la super-résolution et [3] pour l'augmentation de la dynamique). Ces méthodes sont généralement basées sur des transformées (changement de domaine) ou sur des exemples (patches similaires, apprentissage). Au contraire, les méthodes multi-images produisent une image de haute qualité à partir d'un ensemble d'images de qualité inférieure et sont plus proches de l'essence de la photographie, qui accumule les photons. De tels procédés compensent les inconvénients du système d'imagerie par un traitement numérique adéquat des données accumulées. Cette idée a été exploitée avec succès pour augmenter la résolution [35, 87, 125], pour le débruitage [19, 71], pour le défloutage [26] ou pour améliorer la dynamique d'image [2, 53, 81]. La plupart des méthodes prennent en entrée des images prétraitées, ce qui peut entraîner des artefacts. Par conséquent, les bonnes méthodes devraient plutôt gérer les images RAW [35, 36, 37, 48, 61, 131]. La contrepartie de l'utilisation d'images RAW est que ces images peuvent être plus difficiles à gérer. Notamment, la plupart des images RAW sont des images mosaïquées (ou CFA) où une seule valeur d'intensité de couleur est disponible par pixel.

Les performances des algorithmes de traitement d'image dépendent fortement de la qualité des images d'entrée. En particulier, le bruit et l'aliasing sont deux des principales sources d'erreur. Le bruit introduit une incertitude dans les échantillons mesurés, qui diffèrent des valeurs réelles du signal sous-jacent. Par exemple, l'influence du bruit sur la précision de la stéréovision est discutée dans [101]. Le bruit peut généralement être réduit en combinant les données bruitées disponibles. L'aliasing se produit lorsqu'un signal est sous-échantillonné [59]. Cela introduit une ambiguïté dans la reconstruction du signal continu sous-jacent. Les images aliasées ne peuvent pas être correctement interpolées.

L'objectif de cette thèse est de construire les algorithmes produisant une image couleur de haute qualité, contenant un faible niveau de bruit et aliasing, à partir d'une grande séquence (par exemple des centaines ou des milliers) d'images RAW prises avec un appareil photo grand public. Nous supposons que les réglages de la caméra sont fixes et que les images, représentant une scène statique, sont prises quasi instantanément. Les images diffèrent en raison des petits mouvements de la caméra, du bruit et des petites variations d'éclairage. Il s'agit d'un problème difficile nécessitant d'effectuer à la volée du dématriçage [68], du débruitage et de la super-résolution.

Les méthodes multi-images (à partir d'images RAW ou non) peuvent généralement être décomposées en deux étapes principales: l'étape de recalage, dans laquelle les images sont exprimées dans un système commun de coordonnées, et l'étape de combinaison, où les données sont combinées pour construire une image. Même si certaines méthodes ne nécessitent pas une précision de recalage sous-pixellique [119], il s'agit en général d'une condition préalable à

l'obtention de bonnes performances. Cependant, il n'y a pas de méthode standard et satisfaisante pour recaler des images mosaïquées [48, 131]. En raison du contenu particulier de ces images, les méthodes de recalage existantes, conçues pour les images classiques, ne peuvent pas être utilisées directement.

En supposant que les images sont correctement recalées, la plupart des méthodes de combinaison (spécifiques aux images mosaïquées ou non) sont conçues pour une faible quantité d'images d'entrée [37, 38, 48, 86, 87, 91, 117, 118, 131]. Elles utilisent un schéma itératif et exigent la disponibilité de toutes les données en mémoire en même temps. Elles peuvent produire des images de haute qualité, mais la quantité de données accumulées, c'est-à-dire le nombre d'images d'entrée, est nécessairement limitée par la capacité mémoire de l'ordinateur.

Une méthode multi-image rapide et à faible coût mémoire nécessite une méthode de combinaison plus simple. Pour un grand nombre d'images, elle devrait pouvoir atteindre des performances similaires à celles des méthodes plus gourmandes en mémoire. Une image peut être calculée pixel par pixel sans schéma itératif en utilisant une régression à noyau classique [117]. C'est la méthode de régression à noyau la plus simple et la plus rapide car elle ne prend en compte que la distance spatiale mais pas la distance photométrique. Cependant, cela donne des résultats de mauvaise qualité dans les zones contenant des textures ou des discontinuités. En fait, l'image finale est floue. Comme montré dans [41], chaque valeur de pixel calculée est une moyenne locale pondérée des échantillons et est donc obtenue par un filtrage linéaire local. Mais le filtre équivalent dépend de la répartition spatiale des données, qui varie avec la position du pixel.

Dans cette thèse, nous brisons les deux limites (évoquées précédemment) des méthodes existantes. Nous proposons un algorithme de formation d'image à partir d'images RAW qui traite les images séquentiellement de sorte que le coût mémoire ne dépend que de la taille de l'image de sortie. Après une étape de pré-traitement, les images mosaïquées (ou CFA) sont alignées dans un système commun de coordonnées en utilisant une méthode de recalage en deux étapes que nous introduisons. Ensuite, une image couleur est calculée en utilisant une régression à noyau classique et en accumulant les données irrégulièrement échantillonnées. Enfin, le flou introduit pendant la régression est supprimé en appliquant l'inverse du filtre équivalent asymptotique correspondant (que nous introduisons).

Dans cette thèse, chaque étape de la méthode est introduite et analysée séparément. Ses performances et sa précision sont évaluées sur des données synthétiques et réelles. De cette manière, nous parvenons à contrôler l'erreur à chaque étape et à fournir une analyse appropriée des améliorations proposées des méthodes existantes. La génération de données synthétiques nécessite une méthode d'interpolation et le contrôle de l'erreur d'interpolation est crucial pour analyser les performances de notre méthode. Par conséquent, nous étudions également en détail les méthodes d'interpolation. Nous tirons de cette étude de nouvelles méthodes d'interpolation affinées que nous utilisons pour générer des données synthétiques.

La partie I (du chapitre 3 à 6) est consacré aux méthodes d'interpolation. La partie II (chapitre 7 et 8) traite des méthodes de recalage. Finalement, la partie III (du chapitre 9 à 11) est dédiée aux méthodes de fusion d'images.

1.1 Introduction à la partie I: Interpolation

L'interpolation consiste à construire de nouvelles valeurs à partir de la donnée d'un ensemble discret de valeurs connues. Cette notion est étroitement liée aux concepts d'approximation [21], d'ajustement [29] et d'extrapolation. En traitement du signal, elle est couramment exprimée comme le problème de la récupération du signal continu sous-jacent à partir duquel les valeurs connues sont échantillonnées. En supposant que le signal appartient à une classe de fonctions donnée, le principe commun à tous les schémas d'interpolation est de déterminer les paramètres de la représentation continue du signal. Les méthodes d'interpolation les plus courantes sont présentées dans [46, 123].

Une représentation continue du signal est nécessaire lorsque l'on souhaite implémenter numériquement un opérateur défini initialement dans le domaine continu. En particulier, cette représentation est requise lors de l'application d'une transformation géométrique à une image. Soit $\sigma(\mathbb{R}^2)$ l'ensemble des fonctions bijectives de $\mathbb{R}^2$ dans lui-même. Une fonction $\varphi \in \sigma(\mathbb{R}^2)$ est appelée une transformation géométrique. Soit $\underline{u}$ une image de taille $M \times N$. Appliquer la transformation géométrique φ à $\underline{u}$ consiste à ré-échantillonner $\underline{u}$ aux emplacements $\varphi^{-1}(k,l)$. Comme représenté dans la figure 1.1, en général $\varphi^{-1}(k,l) \in \mathbb{R}^2$ n'appartient pas à la grille entière et une représentation continue $u : \mathbb{R}^2 \to \mathbb{R}$ de $\underline{u}$ est requise. L'image transformée $\underline{u}_\varphi$ est alors définie comme l'image de taille $M \times N$ vérifiant

$$(\underline{u}_\varphi)_{k,l} = u\left(\varphi^{-1}(k,l)\right). \tag{1.1}$$

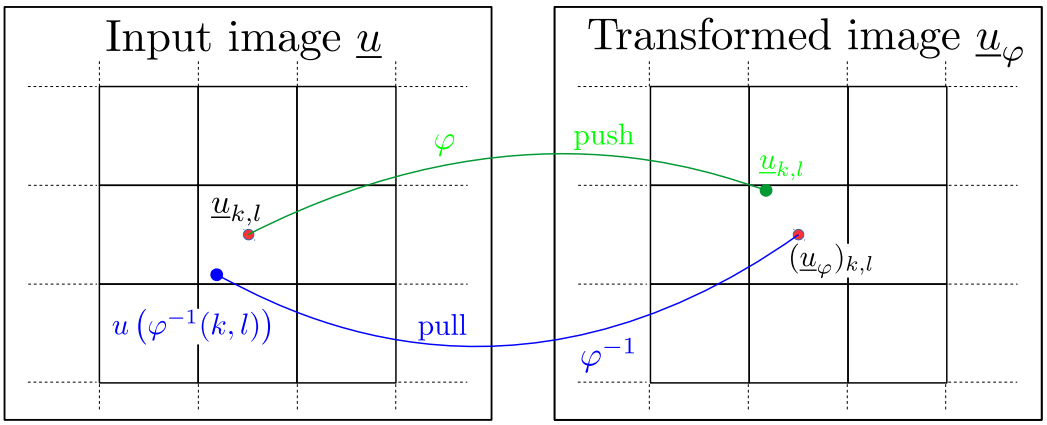


Figure 1.1: Transformation géométrique d'une image. Pour appliquer une transformation $\varphi \in \sigma(\mathbb{R}^2)$ à une image $\underline{u}$, les valeurs de pixels aux emplacements $\varphi^{-1}(k,l)$ doivent être calculées. Comme les échantillons de $\underline{u}$ sont uniquement connus aux positions entières, une représentation continue $u : \mathbb{R}^2 \to \mathbb{R}$ de $\underline{u}$ est requise. Cette représentation s'obtient par une méthode d'interpolaton. Les échantillons $(\underline{u}_\varphi)_{k,l}$ sont dits "tirés" (*pulled*). Dans le système de coordonnées transformé, les seules valeurs de pixel qui peuvent être obtenues sans interpolation correspondent aux échantillons "poussés" (*pushed*) $\underline{u}_{k,l}$ se trouvant aux positions $\varphi(k,l)$.

Dans la partie II et la partie III, nous évaluerons les performances des méthodes proposées sur des données synthétiques qui sont composées d'images transformées. La génération des données requiert une méthode d'interpolation dont le choix influe sur les résultats finaux. L'erreur d'interpolation doit être prise en compte. C'est pourquoi dans la partie I nous étudions en détail les méthodes d'interpolation par polynôme trigonométrique et par B-spline. Nous tirons de cette étude de nouvelles méthodes affinées d'interpolation.

1.1.1 Chapitre 3: Interpolation d'image par polynôme trigonométrique

Pour des signaux 1D ou 2D, l'interpolation de Shannon-Whittaker avec extension périodique peut être formulée comme une interpolation par polynôme trigonométrique [1]. Dans le chapitre 3, nous introduisons la théorie de l'interpolation par polynôme trigonométrique ainsi que quelques unes de ses applications. Premièrement, les polynômes trigonométriques interpolateurs d'une image sont caractérisés et il est prouvé qu'il y en a une infinité dès lors que l'une des tailles de l'image est paire. Trois choix classiques d'interpolateur pour des images à valeur réelles sont présentés et les cas où ils coïncident sont explicités. Ensuite, l'interpolation par polynôme trigonométrique est appliquée à la transformation géométrique des images, au sur-échantillonage et au sous-échantillonnage. Les résultats généraux sont exprimés pour n'importe quel choix d'interpolateur mais davantage de détails sont donnés pour les trois cas particuliers considérés. On montre que les calculs bien-connus basés sur la transformée de Fourier discrète (DFT) doivent être légèrement adaptés.

L'application au filtrage est détaillée séparément dans le chapitre 4. Les performances et les limites de l'interpolation par polynôme trigonométrique sont discutés dans le chapitre 6.

1.1.2 Chapitre 4: Filtrage utilisant l'interpolation par polynôme trigonométrique

Le chapitre 4 est extrait de [15] et est une application du chapitre 3. Nous proposons des algorithmes permettant de filtrer des images à valeurs réelles lorsque le filtre est fourni par une fonction continue $\phi : [-\pi,\pi]^2 \to \mathbb{C}$. Ce problème est ambigu puisque les images sont des entités discrètes et qu'il n'y a pas de moyen unique de définir le filtrage. Nous fournissons un cadre théorique permettant d'examiner les implémentations classiques et efficaces de filtrage qui sont basées sur les DFTs.

Dans ce cadre, le filtrage est interprété comme la convolution $f_\phi * P$ d'une distribution f_ϕ, représentant le filtre, avec un polynôme trigonométrique interpolateur P de l'image. Les différentes interpolations plausibles et les choix de la distribution conduisent à trois algorithmes qui sont tous les trois également licites et qui peuvent être vus comme des variantes du même algorithme standard de filtrage, décrit dans l'algorithme 1.1.

Algorithme 1.1: Algorithme standard de filtrage

Entrée: Une image à valeurs réelles $\underline{u}$ de taille $M \times N$, un filtre $\phi : [-\pi,\pi]^2 \to \mathbb{C}$ (fonction continue) et l'indice $j \in \{1,2,3\}$ de la variante.

Sortie : L'image filtrée $\underline{v}$ de taille $M \times N$ correspondant à la variante j.

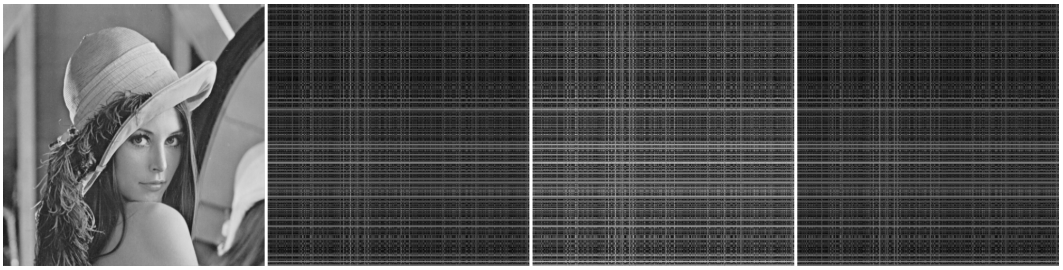
1 Calculer $\tilde{u} = \mathcal{F}_{M,N}(\underline{u})$ la DFT de $\underline{u}$.

2 Calculer $S = (S_{m,n})_{(m,n)\in\hat{\Omega}_{M,N}}$ les échantillons spectraux correspondant à la variante j (voir chapitre 4 pour les détails).

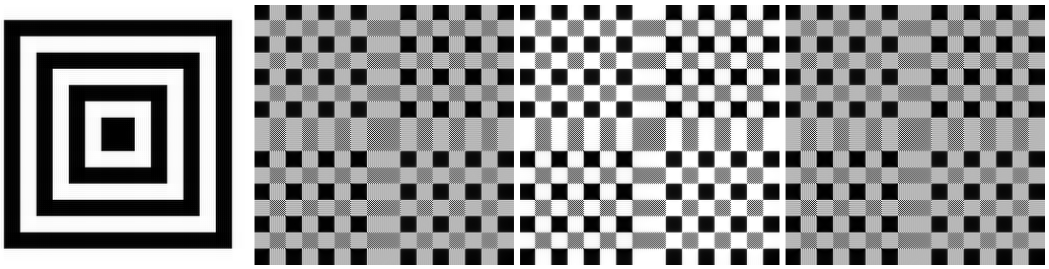
3 Calculer $\tilde{v} = (\tilde{u}_{m,n} S_{m,n})_{(m,n)\in\hat{\Omega}_{M,N}}$ la mutiplication point à point de $\tilde{u}$ et S.

4 Calculer $\underline{v} = \mathcal{F}^{-1}_{M,N}(\tilde{v})$ la DFT inverse de $\tilde{v}$.

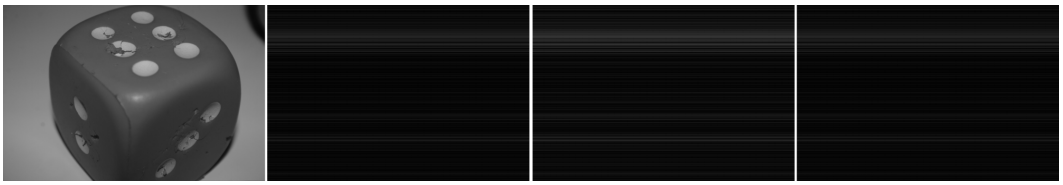
Aucune méthode ne devrait *a priori* être préférée aux autres; le choix dépend de l'application. Les différences entre les résultats sont localisées à la frontière de la DFT et ne sont en pratique pas visibles à l'oeil nu. Nous montrons cela sur plusieurs configurations expérimentales en faisant varier l'image d'entrée et le filtre considéré. Par exemple, les résultats du filtre de translation de paramètre $(1/4, 1/4)$ sont comparés dans la figure 1.2 et dans la table 1.1. La différence maximale globale se situe autour de deux niveaux de gris. Dans certains cas, cependant, nous discutons de la manière dont le choix de la variante peut affecter les propriétés fondamentales du filtrage. Nous fournissons une implémentation des algorithmes à l'adresse http://www.ipol.im/pub/art/2016/116/.



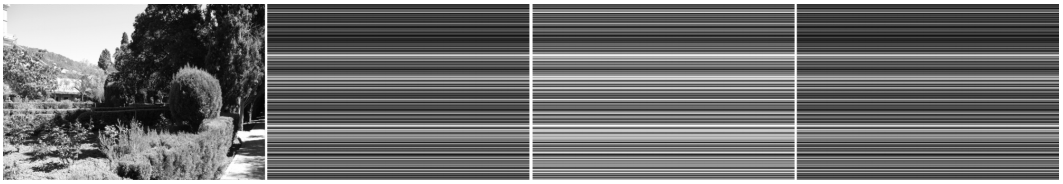
(a) Lenna



(b) Square



(c) Dice



(d) Garden

Figure 1.2: Comparaison des trois méthodes de filtrage pour le filtre de translation de paramètre $(1/4, 1/4)$. De gauche à droite nous montrons pour chaque image d'entrée: la partie réelle de la première méthode et les différences relatives $\Delta_{\text{rel}}^{1,2}$, $\Delta_{\text{rel}}^{1,3}$ et $\Delta_{\text{rel}}^{2,3}$. Les images de différence sont multipliées par 10^5 pour des soucis de visualisation. Les différences entre les résultats sont localisées à la frontière de la DFT et ne sont pas visibles à l'oeil nu.

		Lenna	Square	Dice	Garden
1 vs 2	d	0.77 (0.34%)	1.6 (0.41%)	0.13 (0.067%)	1.4 (0.43%)
	m_d	0.12 (0.052%)	0.55 (0.14%)	0.020 (0.010%)	0.30 (0.094%)
1 vs 3	d	1.1 (0.48%)	2.2 (0.58%)	0.18 (0.093%)	2.0 (0.61%)
	m_d	0.17 (0.075%)	0.77 (0.20%)	0.029 (0.015%)	0.42 (0.13%)
2 vs 3	d	0.77 (0.34%)	1.6 (0.41%)	0.13 (0.067%)	1.4 (0.43%)
	m_d	0.12 (0.053%)	0.55 (0.14%)	0.020 (0.010%)	0.30 (0.094%)

Table 1.1: Différence maximale d et différence moyenne m_d entre les trois méthodes de filtrage pour le filtre de translation de paramètre $(1/4, 1/4)$. Les valeurs relatives sont entre parenthèses. La différence maximale globale est d'environ deux niveaux de gris de sorte qu'aucune différence ne peut être vue à l'oeil nu.

1.1.3 Chapitre 5: Interpolation par B-spline

Le chapitre 5 est extrait de [14]. L'interpolation par B-spline est une alternative non bande limitée et largement utilisée à l'interpolation de Shannon-Whittaker. Pour des signaux 1D elle est définie de la manière suivante.

Definition 1.1. *La fonction de B-spline normalisée d'ordre n, notée* $\beta^{(n)}$*, est définie récursivement par*

$$\beta^{(0)}(x) = \begin{cases} 1, & -\frac{1}{2} < x < \frac{1}{2} \\ \frac{1}{2}, & x = \pm\frac{1}{2} \\ 0, & \text{sinon} \end{cases} \qquad \text{et pour } n \geq 0, \quad \beta^{(n+1)} = \beta^{(n)} * \beta^{(0)} \tag{1.2}$$

où le symbole $*$ *désigne l'opérateur de convolution.*

Definition 1.2 (Interpolation par B-spline)**.** *L'interpolateur par B-spline d'ordre n d'un signal discret* $f \in \mathbb{R}^{\mathbb{Z}}$ *est la spline* $\varphi^{(n)}$ *de degré n définie pour* $x \in \mathbb{R}$ *par*

$$\varphi^{(n)}(x) = \sum_{i \in \mathbb{Z}} c_i \beta^{(n)}(x - i) \tag{1.3}$$

où les coefficients de B-spline $c = (c_i)_{i \in \mathbb{Z}}$ *sont caractérisés par la condition d'interpolation*

$$\varphi^{(n)}(k) = f_k, \quad \forall k \in \mathbb{Z}. \tag{1.4}$$

Dans le chapitre 5, nous expliquons comment les signaux et les images peuvent être efficacement interpolés par B-spline en utilisant du filtrage linéaire. Dans la méthode phare en deux étapes proposée par Unser et al. en 1991 [129], les coefficients de B-spline c sont d'abord calculés lors d'une étape de préfiltrage. Ensuite, les valeurs interpolées sont calculées à l'aide de (1.3). Nous proposons deux algorithmes de préfiltrage légèrement différents dont les précisions sont contrôlées grâce à un traitement rigoureux des conditions au bord. Le premier algorithme est général et fonctionne pour toute extension au bord tandis que le second est applicable sous des hypothèses spécifiques.

Le chapitre 5 contient toutes les informations, théoriques et pratiques, nécessaires pour effectuer efficacement une interpolation par B-spline pour tout ordre et toute extension au bord. Nous décrivons précisément comment évaluer le noyau et comment calculer les paramètres de l'interpolateur B-spline.

Dans une partie expérimentale, nous montrons que l'augmentation de l'ordre améliore la qualité de l'interpolation. Par exemple, l'interpolation B-spline se rapproche de l'interpolation de Shannon-Whittaker lorsque l'ordre augmente [5]. La figure 1.3 illustre la décroissance de la différence avec l'ordre.

En tant qu'application fondamentale, nous fournissons également une implémentation de la transformation homographique des images en utilisant une interpolation par B-spline à l'adresse `http://www.ipol.im/pub/art/2018/221/`. Un exemple de transformation homographique est présenté dans la figure 1.4.

1.1.4 Chapitre 6: Cohérence des méthodes d'interpolation

Il n'y a pas de procédure universelle pour évaluer la qualité et la performance d'une méthode d'interpolation. Dans le chapitre 6, nous introduisons une nouvelle quantité: la mesure de cohérence. Pour une image donnée, la mesure de cohérence $\mathcal{CM}(\varphi)$ pour la transformation $\varphi \in \sigma(\mathbb{R}^2)$ mesure l'erreur après avoir appliqué successivement φ, une extraction de sous-image

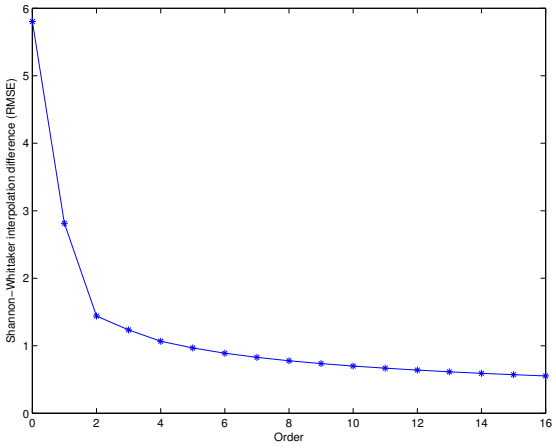


Figure 1.3: Décroissance de la différence entre l'interpolation de Shannon-Whittaker et l'interpolation par B-spline pour $n \in \{0, \ldots, 16\}$. Des transformations géométriques de l'image *Lenna* sont comparées. La RMSE est prise au centre des images de sorte que l'extension au bord a une influence négligeable. L'extension périodique est utilisée pour les deux méthodes d'interpolation.

Figure 1.4: Exemple de transformation homographique utilisant l'interpolation par B-spline d'ordre 11 (avec condition au bord symétrique).

(supprimant les artefacts au bord) et l'inverse φ^{-1}. Une moyenne sur des transformations aléatoires $(\varphi_i)_{1 \leq i \leq N_{\text{transf}}}$ est faite pour supprimer la dépendance en la transformation. Nous définissons la mesure de cohérence $\mathcal{CM}$ par

$$\mathcal{CM} = \frac{1}{N_{\text{transf}}} \sum_{i=1}^{N_{\text{transf}}} \mathcal{CM}(\varphi_i). \tag{1.5}$$

Une mesure plus précise, qui élimine les artefacts à très haute fréquence, est obtenue en coupant le spectre de la différence. La variante est appelée la mesure de consistance coupée et est notée $\mathcal{CM}^c$.

Nous proposons également de nouvelles méthodes d'interpolation affinées qui sont basées sur le zoom avant par DFT et sur des méthodes d'interpolation préexistantes. La version zoomée

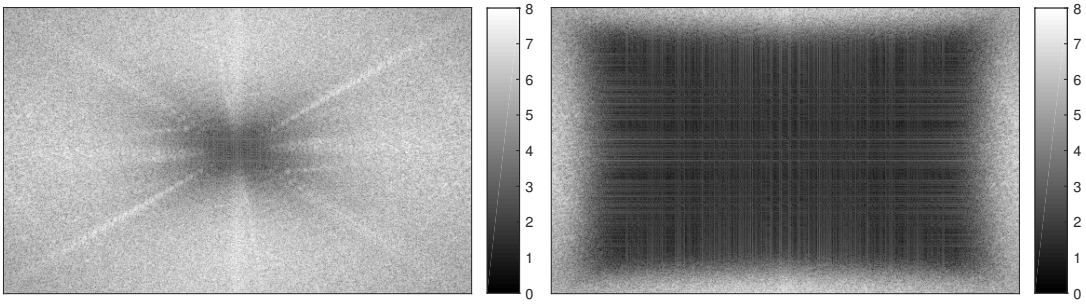
d'une méthode d'interpolation est obtenue en l'appliquant au zoom avant par DFT de l'image. Dans la version *periodic plus smooth* de deux méthodes d'interpolation, la non-périodicité est gérée en appliquant la version zoomée de la première méthode à la composante périodique et la seconde méthode à la composante lisse (*smooth*).

Dans une partie expérimentale, nous montrons que les méthodes affinées proposées ont de meilleures mesures de cohérence que leurs méthodes d'interpolation de base et que l'erreur est principalement localisée dans une petite bande haute fréquence. Cela peut être vu dans les résultats présentés dans le tableau 1.2. Le spectre de l'image de différence (à partir de laquelle les mesures de cohérence sont calculées) est montré dans la figure 1.5 pour diverses méthodes d'interpolation.

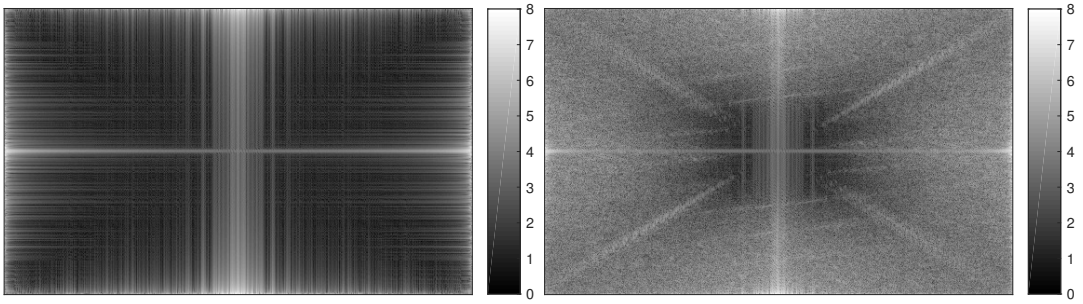
Sur la base de cette analyse, il est recommandé d'utiliser les versions *periodic plus smooth* de l'interpolation par B-spline d'ordre élevé. Cette méthode est plus rapide et donne de meilleurs résultats que l'interpolation par polynôme trigonométrique. Dans le chapitre 10, une telle méthode sera utilisée pour générer des données synthétiques et pour la méthode de *burst denoising*.

	$\mathcal{CM}$	$\mathcal{CM}^c$	Temps (s)
spline1	2.53480	2.50077	225
bic	1.18612	1.13573	336
spline3	0.75702	0.67967	283
spline11	0.35355	0.19626	796
tpi	0.20564	0.06345	4767
spline1-z2	0.86002	0.81810	1308
bic-z2	0.25140	0.15072	1425
spline3-z2	0.20585	0.06816	1407
spline11-z2	0.20564	0.06345	2053
p+s-spline1	0.84660	0.81640	2039
p+s-spline1-bic	0.84660	0.81640	2159
p+s-bic	0.18008	0.13811	2355
p+s-spline3	0.09417	0.03007	2199
p+s-spline3-bic	0.09417	0.03007	2259
p+s-spline11	0.08785	0.01506	3360
p+s-spline11-bic	0.08785	0.01506	2907
p+s-tpi-spline1	0.08785	0.01506	5865
p+s-tpi-bic	0.08785	0.01506	5960
p+s-tpi-spline3	0.08785	0.01506	5926
p+s-tpi-spline11	0.08785	0.01506	6443

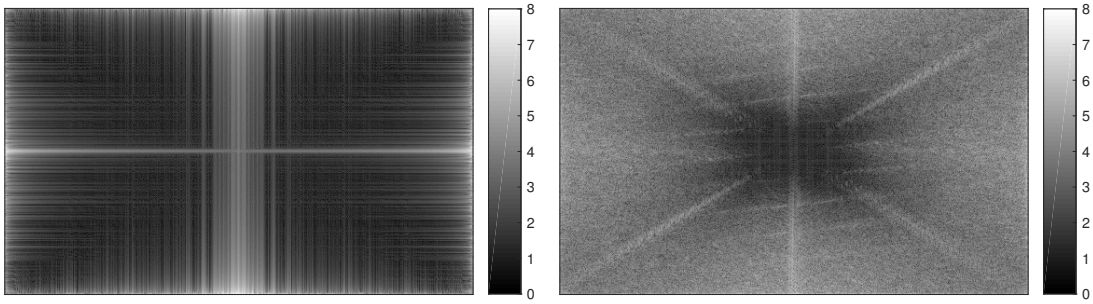
Table 1.2: Évaluation des mesures de cohérence pour $N_{\text{transf}} = 1000$ homographies aléatoires. L'évaluation est faite en excluant au bord des images une bande de largeur $\delta = 20$ pour rendre les résultats indépendants du choix de l'extension. Le temps affiché correspond au temps de calcul, en secondes, de toutes les transformations. Cela peut également être interprété comme le temps moyen, en millisecondes, nécessaire pour appliquer une transformation et son inverse. Les méthodes *tpi* et *bic* représentent respectivement l'interpolation par polynôme trigonométrique et l'interpolation bicubique. Le suffixe *z2* correspond aux versions zoomées. Pour la version *periodic plus smooth*, nous utilisons le préfixe *p+s*. La première méthode est celle utilisée sur la composante périodique. Il est clair que les méthodes affinées ont une meilleure cohérence. Les erreurs sur les versions coupées (colonne 2) étant significativement plus faibles, nous concluons que l'erreur est principalement localisée dans une petite bande haute fréquence. Les méthodes *periodic plus smooth* fournissent de meilleurs résultats car la non-périodicité de l'image est correctement gérée.



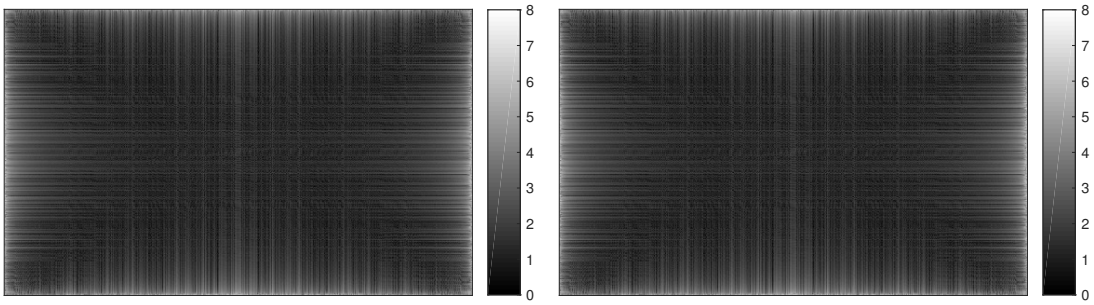
(a) *bic* (1.37, 1.31) (b) *spline11* (0.423, 0.236)



(c) *tpi* (0.211, 0.0584) (d) *bic-z2* (0.253, 0.145)



(e) *spline11-z2* (0.211, 0.0584) (f) *p+s-bic* (0.191, 0.135)



(g) *p+s-spline11* (0.117, 0.0187) (h) *p+s-tpi-spline11* (0.117, 0.0187)

Figure 1.5: Spectre de l'image de différence (à partir de laquelle les mesures de cohérence sont calculées) pour différentes méthodes d'interpolation. Les spectres correspondent à la transformée de Fourier discrète non normalisée en échelle logarithmique $u \mapsto \log(1+u)$. La transformation géométrique φ utilisée est une homographie. Les valeurs entre parenthèses correspondent aux mesures de cohérence $(\mathcal{CM}(\varphi), \mathcal{CM}^c(\varphi))$, qui sont respectivement les énergies L2 sans et avec suppression de la bande des fréquences les 5% plus élevées. Pour les méthodes *bic*, *bic-z2* et *p+s-bic*, i.e., les méthodes où les images éventuellement zoomées sont interpolées par interpolation bicubique, le spectre de la différence est élevé partout sauf dans une petite région de basse fréquence. Pour la méthode *spline11*, la région correspondante est plus grande. Pour les méthodes *tpi* et *spline11-z2*, le spectre a des valeurs non-négligeables dans une bande autour de la fréquence de Nyquist et dans une structure en croix verticale et horizontale. La structure en croix est également visible pour la méthode *bic-z2* et est due à la gestion incorrecte de la non-périodicité. En effet, elle disparaît dans les méthodes *p+s-spline11* et *p+s-tpi-spline11*.

1.2 Introduction à la partie II: Recalage

Le recalage d'images est l'une des techniques les plus utilisées en vision par ordinateur. L'objectif est de trouver la transformation optimale qui met en correspondance les pixels de deux (ou plus) images. Le recalage d'images est requis par exemple lorsque les images sont acquises à des moments différents, à partir de points de vue distincts ou par différents capteurs. Une estimation précise et rapide est importante dans des problèmes tels que l'estimation du flux optique [10, 56], le suivi des objets [134], la stabilisation vidéo [78], l'assemblage [115] ou la reconstruction 3D [52]. La tâche est difficile car elle traite des problèmes tels que les occlusions, le bruit, les changements de luminosité locaux ou les mouvements parasites.

Des revues des méthodes existantes sont fournies par [75, 88, 137]. Les méthodes sont en général fondées sur l'information géométrique [27] ou sur l'intensité. Les méthodes basées sur l'intensité sont généralement plus rapides mais plus sensibles aux changements de luminosité et aux valeurs aberrantes. Les méthodes basées sur la géométrie sont généralement plus sensibles au bruit et au flou de mouvement, mais permettent d'estimer des déformations plus fortes [114].

Dans la partie III, nous considérerons des méthodes multi-images qui nécessitent une étape de recalage efficace. Les images d'entrée ne différeront que par de petits mouvements de la caméra, du bruit, de la quantification et éventuellement de petites variations d'éclairage. C'est pourquoi dans la partie II nous considérons des méthodes de recalage basées sur l'intensité. Tout d'abord, nous améliorons l'algorithme *inverse compositional*. Ensuite, nous développons une méthode en deux étapes pour les images mosaïquées (ou CFA).

1.2.1 Chapitre 7: Algorithme *inverse compositional* modifié

Introduit pour la première fois dans [8, 122], l'algorithme *inverse compositional* est une amélioration de la méthode classique de Lucas-Kanade [74] pour l'estimation de mouvement paramétrique. À chaque étape de son schéma itératif, il résout un problème de minimisation équivalent à Lucas-Kanade mais plus efficacement. Plus précisément, à une étape donnée $j \geq 1$, l'idée est d'affiner la transformation estimée actuelle $\Psi(\cdot;\mathbf{p}_{j-1})$ avec une transformation incrémentale inversée (d'où le nom de l'algorithme) $\Psi(\cdot;\Delta\mathbf{p}_j)^{-1}$, i.e.,

$$\Psi(\cdot;\mathbf{p}_j) = \Psi(\cdot;\mathbf{p}_{j-1}) \circ \Psi(\cdot;\Delta\mathbf{p}_j)^{-1}. \tag{1.6}$$

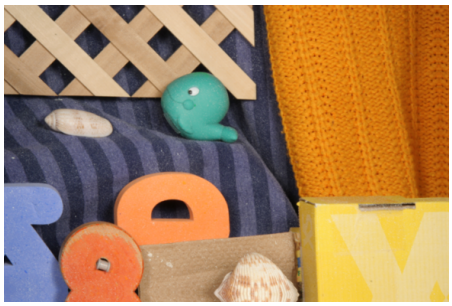
L'incrément $\Delta\mathbf{p}_j$ approche le minimiseur de l'énergie incrémentale

$$\Delta\mathbf{p} \in \mathbb{R}^n \mapsto E_1(\Delta\mathbf{p};\mathbf{p}_{j-1}) = \sum_{\mathbf{x}\in\Omega} \rho\left(\|I_2(\Psi(\mathbf{x};\mathbf{p}_{j-1})) - I_1(\Psi(\mathbf{x};\Delta\mathbf{p}))\|^2\right) \tag{1.7}$$

où $\rho : \mathbb{R}^+ \to \mathbb{R}^+$ est appelée fonction d'erreur. Une meilleure précision pour les larges déplacements est obtenue avec des approches multi-échelles comme dans [104].

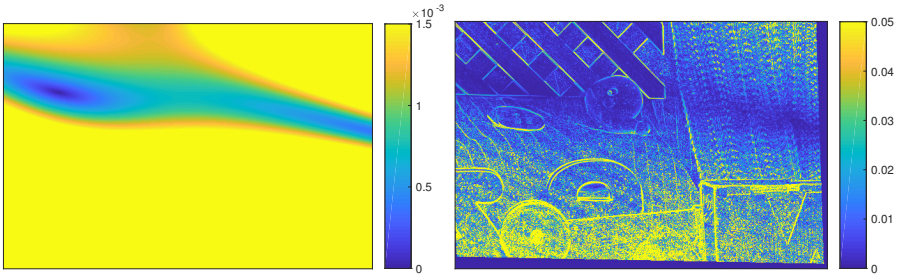
Dans le chapitre 7, nous introduisons plusieurs améliorations de l'algorithme *inverse compositional*. Nous proposons une gestion améliorée des pixels du bord, une gestion des couleurs et une estimation différente du gradient, ainsi que la possibilité de sauter des échelles dans le schéma multi-échelle. Dans une partie expérimentale, nous analysons l'influence des modifications. Nous trouvons que la précision d'estimation est améliorée d'un facteur supérieur à 1,3 alors que le temps de calcul est réduit d'un facteur supérieur à 2,2 pour les images en couleur. Un exemple d'estimation de mouvement sur des données synthétiques, avec et sans modifications, est présenté dans la figure 1.6. Les modifications conduisent à des résultats nettement meilleurs.

L'algorithme *inverse compositional* modifié sera utilisé dans le chapitre 8 dans la méthode en deux étapes proposée pour les images mosaïquées.

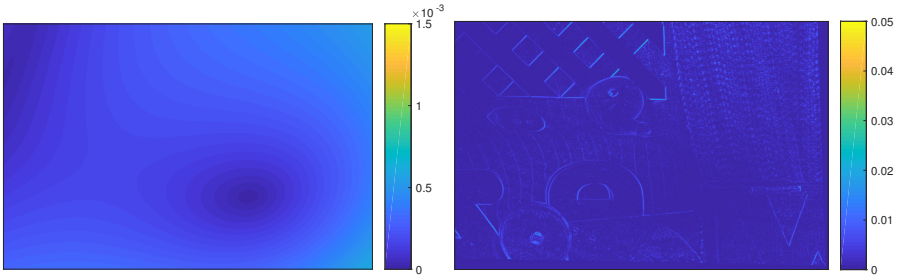


(a) Image de référence $I_1 = I_2(\Psi(\cdot;\mathbf{p}^\star))$

(b) Image déformée I_2



(c) Champ d'erreur $\mathrm{EPE}(\mathbf{p},\mathbf{p}^\star)(\mathbf{x})$ (gauche) et résidu (droite) pour l'algorithme IC



(d) Champ d'erreur $\mathrm{EPE}(\mathbf{p},\mathbf{p}^\star)(\mathbf{x})$ (gauche) et résidu (droite) pour l'algorithme IC modifié

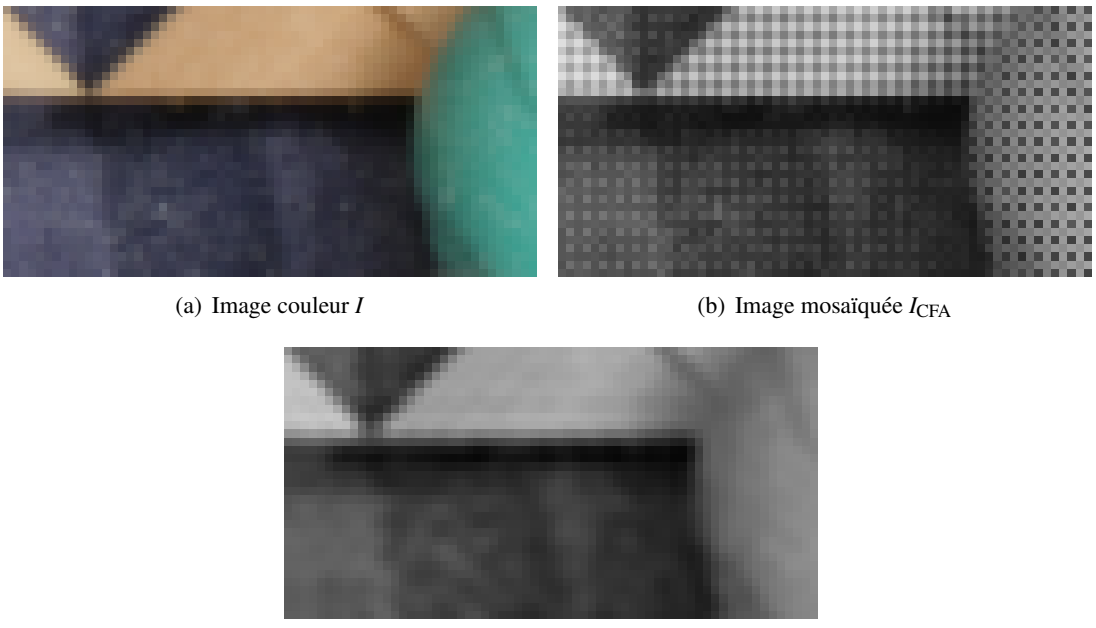
Figure 1.6: Exemple d'estimation de mouvement sur des données synthétiques. L'image de référence $I_1 = I_2(\Psi(\cdot;\mathbf{p}^\star))$ est un exemple d'image synthétique relié à I_2 par une homographie. À la deuxième et troisième ligne, le mouvement estimé $\mathbf{p}$ est obtenu soit par l'algorithme *inverse compositional* soit par sa version modifiée. Pour les deux méthodes, l'image de droite est en réalité la RMS sur les canaux du résidu $I_1 - I_2(\Psi(\cdot;\mathbf{p}))$, qui est obtenue par interpolation bicubique. Sans modification, on a EPE=0.00460 et $\mathrm{RMSE}(I_1(\mathbf{x}), I_2(\Psi(\mathbf{x};\mathbf{p}))) = 0.042838$. Avec modification, on a EPE=0.00022 et $\mathrm{RMSE}(I_1(\mathbf{x}), I_2(\Psi(\mathbf{x};\mathbf{p}))) = 0.001790$.

1.2.2 Chapitre 8: Recalage d'images mosaïquées (ou CFA)

Une image mosaïquée (ou CFA) est une image numérique dans laquelle un seul des trois canaux de couleur a été capturé à chaque emplacement de pixel. L'image en niveaux de gris correspondante a une structure en mosaïque provoquée par le filtre-réseau coloré (CFA). L'image RAW acquise est un exemple typique d'image mosaïquée. Le modèle le plus commun pour le CFA, et celui qui est considéré ici, est le motif de Bayer [12]. Dans de nombreuses applications, la transformation géométrique entre deux images mosaïquées doit être estimée sans connaissance des images couleur sous-jacentes. Malheureusement, il n'existe pas de méthode de recalage standard et satisfaisante pour les images mosaïquées. Les méthodes de recalage existantes conçues pour les images classiques ne peuvent pas être utilisées directement et une étape de pré-traitement est requise.

Dans le chapitre 8, nous introduisons des méthodes en deux étapes pour le recalage des images mosaïquées. Tout d'abord, les deux images mosaïquées sont converties en images non-mosaïquées (niveaux de gris) par filtrage passe-bas. Selon [6], ces images filtrées estiment l'information de luminance contenue dans les images mosaïquées. Un exemple de conversion est présenté dans la figure 1.7. Ensuite, la transformation est estimée en appliquant une méthode de recalage préexistante conçue pour les images classiques.

Les performances des méthodes proposées sont évaluées expérimentalement pour plusieurs filtres passe-bas et plusieurs méthodes de recalage préexistantes. Nous concluons qu'un filtre passe-bas parfait doit être appliqué et que l'algorithme *inverse compositional* modifié avec fonction d'erreur robuste (voir le chapitre 7) doit être utilisé. Une comparaison des performances pour différentes méthodes de recalage de base est présentée dans le tableau 1.3. La méthode recommandée est à la fois précise et rapide, et sera utilisée dans notre algorithme de formation d'image à partir d'images RAW (voir le chapitre 11).



(a) Image couleur I (b) Image mosaïquée I_{CFA}

(c) Filtrage passe-bas parfait de I_{CFA}

Figure 1.7: Conversion d'image mosaïquée à non-mosaïquée. L'image mosaïquée I_{CFA} (en (b)) est obtenue en appliquant le filtre de Bayer à l'image couleur I (en (a)). L'image en (c) est obtenue après filtrage passe-bas parfait de (b).

		SIFT + RANSAC	IC	M-IC
$\sigma = 0$	EPE	0.01932	0.01173	**0.00360**
	Temps (s)	1298	458	**149**
$\sigma = 3$	EPE	0.03002	0.01309	**0.00559**
	Temps (s)	1334	453	**151**
$\sigma = 5$	EPE	0.04929	0.01678	**0.00979**
	Temps (s)	1393	451	**153**
$\sigma = 10$	EPE	0.15314	0.02391	**0.01550**
	Time	1671	447	**165**
$\sigma = 20$	EPE	0.49535	0.04903	**0.03352**
	Temps (s)	2575	424	**204**
$\sigma = 30$	EPE	0.83937	0.07455	**0.05327**
	Temps (s)	3029	463	**344**
$\sigma = 50$	EPE	1.84189	0.13512	**0.10812**
	Temps (s)	3003	579	**531**

Table 1.3: Influence de la méthode de recalage de base pour la méthode de recalage en deux étapes d'images mosaïquées. La conversion en images non-mosaïquées est faite par filtrage passe-bas parfait. Pour les algorithmes *inverse compositional* (IC) et *inverse compositional* modifié (M-IC), la fonction d'erreur de Lorentz est utilisée. Le temps de calcul affiché correspond au temps CPU utilisé pour les estimations de mouvement de $N_{\text{images}} = 1000$ images et est exprimé en secondes. Notez que cela correspond également au temps de calcul moyen par image en millisecondes. La précision est évaluée en termes d'*end-point-error* (EPE). L'algorithme M-IC donne les meilleurs résultats pour tous les niveaux de bruit, à la fois en termes d'efficacité et de précision. Grâce à la manipulation correcte des pixels au bord et au pré-filtrage avant l'estimation du gradient, il est possible d'obtenir une estimation de mouvement précise en quelques itérations, même si les images non-mosaïquées contiennent des artefacts.

1.3 Introduction à la partie III: Fusion/Formation d'images

Les étapes principales des méthodes multi-images sont les étapes de recalage et de combinaison. Après le recalage, les valeurs de pixels peuvent être exprimées dans le système de coordonnées de référence soit en tirant soit en poussant les échantillons (voir Figure 1.1). Les valeurs d'intensité tirées sont calculées en effectuant une transformation géométrique des images et donc en utilisant une méthode d'interpolation. Par exemple, dans les méthodes de *burst denoising* [19, 71], l'image fusionnée est calculée par une moyenne des images transformées. Cela fournit des résultats médiocres lorsque les images sont aliasées et ne peut pas être utilisé dans un contexte de super-résolution. Au contraire, la combinaison des échantillons poussés est ouverte à la super-résolution. Les échantillons poussés définissent un ensemble de données irrégulièrement échantillonnées (voir l'exemple de la figure 1.8). Le calcul d'une valeur d'intensité à un emplacement arbitraire est appelé ajustement (ou approximation) de données irrégulièrement échantillonnées. Une image est obtenue en calculant des intensités dans une grille régulière.

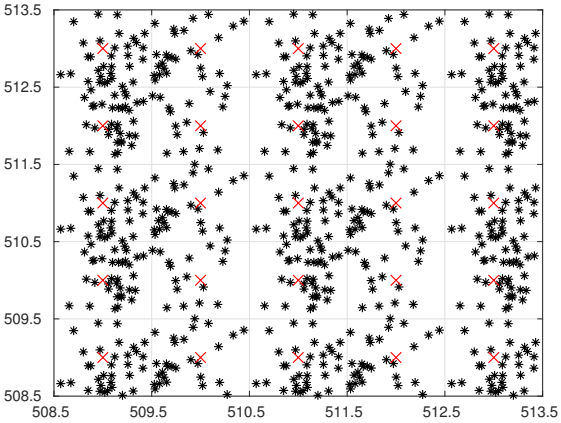


Figure 1.8: Exemple de répartition spatiale de données irrégulièrement échantillonnées. Pour calculer les intensités aux positions entières (croix rouges), une méthode d'ajustement des données irrégulièrement échantillonnées doit être utilisée.

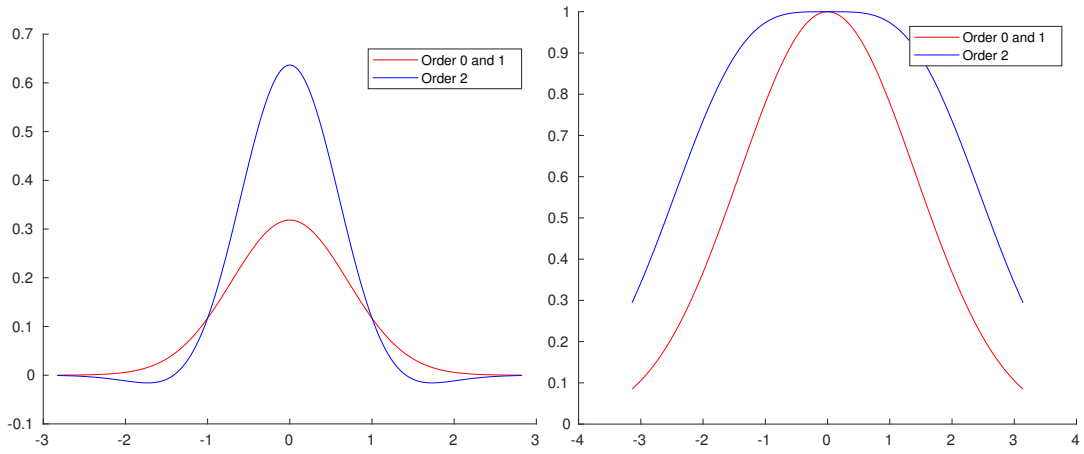
1.3.1 Chapitre 9: Fusion d'images rapide et à faible coût mémoire utilisant une régression à noyau classique

Dans le chapitre 9, nous considérons l'ajustement de données irrégulièrement échantillonnées par régression à noyau (KR) [117]. Nous proposons une méthode de fusion d'images rapide et à faible coût mémoire qui est conçue pour un grand nombre d'images.

Dans les méthodes KR, la valeur d'intensité des données est localement approchée par une expansion polynomiale (d'ordre au plus 2 dans la suite), dont les coefficients sont obtenus en résolvant une régression linéaire pondérée (avec des poids issus d'un noyau). Nous montrons que les systèmes linéaires impliqués peuvent être obtenus par accumulation de données.

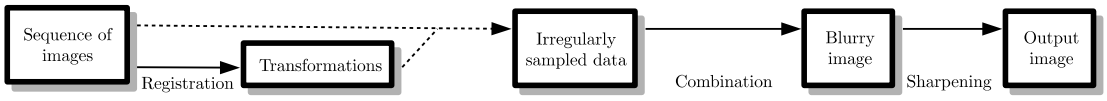
La régression à noyau classique (CKR) est le cas le plus simple et le plus efficace où les poids ne dépendent que de la répartition spatiale des données. Étant équivalent à un filtrage linéaire local [41], la CKR introduit du flou. Nous introduisons le filtre équivalent asymptotique (AEF), une approximation du filtre équivalent réel. Un exemple d'AEF est montré dans la figure 1.9.

Dans l'algorithme de fusion d'images proposé l'étape de combinaison, utilisant la CKR avec noyau gaussien, est divisée en une étape d'accumulation, où les images sont traitées séquentiellement, et une étape de calcul de l'image. Le flou, introduit par la CKR, est inversé en appliquant l'inverse du AEF. Les étapes principales de notre méthode sont résumées à la figure 1.10.

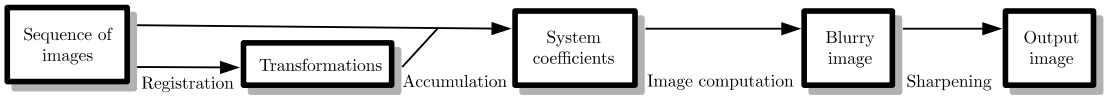


(a) $x \mapsto \tilde{w}_{\sigma_s,\infty}(x,0)$ (b) $x \mapsto \mathcal{F}(\tilde{w}_{\sigma_s,\infty})(x,0)$

Figure 1.9: Filtre équivalent asymptotique $\tilde{w}_{\sigma_s,\infty}$ pour un noyau gaussien d'écart-type $\sigma_s = 0.7$. Dans le domaine spatial (a) et de Fourier (b), les fonctions sont radiales de sorte que seule la coupe $y = 0$ est montrée. Les fonctions décroissent quand $\|\mathbf{x}\|$ augmente. En particulier, le filtrage par AEF atténue le contenu haute fréquence.



(a) Algorithme gourmand en mémoire



(b) Notre algorithme

Figure 1.10: Étapes principales de notre méthode de fusion d'images. (a) et (b) sont théoriquement équivalents. La combinaison des données irrégulièrement échantillonnées est remplacée par une étape d'accumulation, où les images sont traitées séquentiellement pour obtenir les coefficients des systèmes linéaires, et une étape de calcul de l'image, où les systèmes sont résolus.

La méthode de recalage utilisée est celle introduite dans le chapitre 7. Le choix des paramètres optimaux et l'évaluation de la performance de notre algorithme sont les sujets du chapitre 10. L'étape de combinaison est adaptée aux images mosaïquées dans le chapitre 11.

1.3.2 Chapitre 10: Évaluation expérimentale de notre algorithme de fusion d'images

Dans le chapitre 10, nous évaluons expérimentalement les performances de notre algorithme de fusion d'images sur des données synthétiques et réelles. Notre analyse du cas synthétique montre que l'étape d'affûtage est cruciale pour obtenir une image débruitée et sans flou. Nous déterminons également la meilleure configuration pour l'ordre et l'échelle, en fonction du facteur de sous-échantillonnage. En supposant une répartition uniforme des échantillons, nous concluons qu'il est préférable d'utiliser l'ordre 0 lorsqu'aucune super-résolution n'est requise, et l'ordre 2 pour la super-résolution. Nous trouvons qu'une échelle d'environ 0,7 pour le noyau gaussien est le meilleur choix.

Le résultat de notre méthode pour 200 images réelles de taille 512×512 est comparé à ceux d'une méthode de *burst denoising* et de ACT dans la figure 1.12 pour un zoom de $\lambda \in$

$\{1,2\}$. Comme ils ne peuvent pas être distingués à l'oeil nu, la différence avec notre méthode est montrée. Les temps de calcul et les utilisations mémoire maximales des méthodes sont les suivants:

		Burst denoising	ACT	Notre méthode (avec KR)
$\lambda = 1$	Temps (s)	442	1784	**135**
	Mémoire maximale (kB)	175468	5562344	**82192**
$\lambda = 2$	Temps (s)	ND	1942	**444**
	Mémoire maximale (kB)	ND	6224528	**1200492**

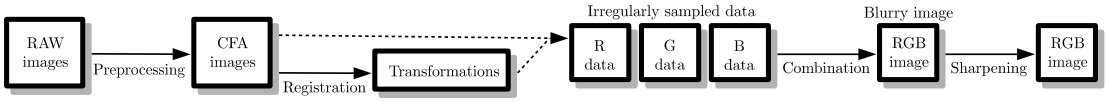
Nous montrons que pour un grand nombre d'images (synthétiques ou réelles), notre algorithme de fusion d'images fournit des résultats similaires aux méthodes plus lentes et plus gourmandes en mémoire. Le bruit résiduel sur les exemples synthétiques et réels diminue comme prévu et notre algorithme est capable d'effectuer de la super-résolution.

À partir des expériences sur les données réelles, nous constatons que les performances des méthodes de fusion d'images sont limitées par des processus incontrôlés (dématriçage, compression JPEG, quantification sur 8 bits). C'est pourquoi nous proposons dans le chapitre 11 un algorithme de formation d'image prenant en entrée des images RAW.

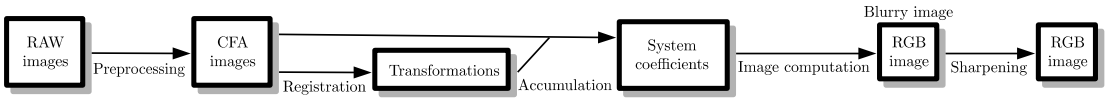
1.3.3 Chapitre 11: Formation d'image à partir d'une grande séquence d'images RAW

Dans le chapitre 11, nous proposons une méthode de formation d'image à partir d'une grande séquence d'images RAW. Il s'agit une version adaptée de notre méthode de fusion d'images, décrite dans le chapitre 9, pour les images RAW. Ce chapitre fait suite au travail publié dans [13].

Les images RAW sont d'abord pré-traitées pour transformer la courbe de bruit et ajuster les histogrammes. Ensuite, la méthode de recalage en deux étapes pour les images mosaïquées, introduite dans le chapitre 8, est utilisée pour aligner les images. Comme dans le chapitre 9, les images sont combinées par une régression à noyau classique du second ordre avec noyau gaussien. Le processus comporte deux étapes. La première est un processus d'accumulation dans lequel les images sont traitées séquentiellement. Dans la seconde étape, le flou introduit par la régression à noyau classique est inversé par un filtre d'affûtage. Les principales étapes de notre méthode de formation d'image à partir d'images RAW sont présentées dans la figure 1.11.



(a) Algorithme gourmand en mémoire



(b) Notre algorithme

Figure 1.11: Étapes principales de notre méthode de formation d'image à partir d'images RAW. (a) et (b) sont théoriquement équivalents. La combinaison des données irrégulièrement échantillonnées est remplacée par une étape d'accumulation, où les images sont traitées séquentiellement, et une étape de calcul de l'image.

Nous montrons expérimentalement que, pour un grand nombre d'images RAW, notre méthode de formation d'image fournit, de manière efficace et avec une faible utilisation de la mémoire, un résultat de haute qualité. Contrairement aux images fusionnées à partir d'images traitées (comme dans le chapitre 10), les images formées à partir d'images RAW ne contiennent

pas d'artefact provenant du processus inconnu de traitement des images. Notre méthode effectue avec succès de la super-résolution et le bruit résiduel diminue comme prévu. Nous avons obtenu des résultats similaires à ceux obtenus par des méthodes plus lentes et plus gourmandes en mémoire. Le résultat de notre méthode appliquée à 200 images RAW est montré dans la figure 1.13.

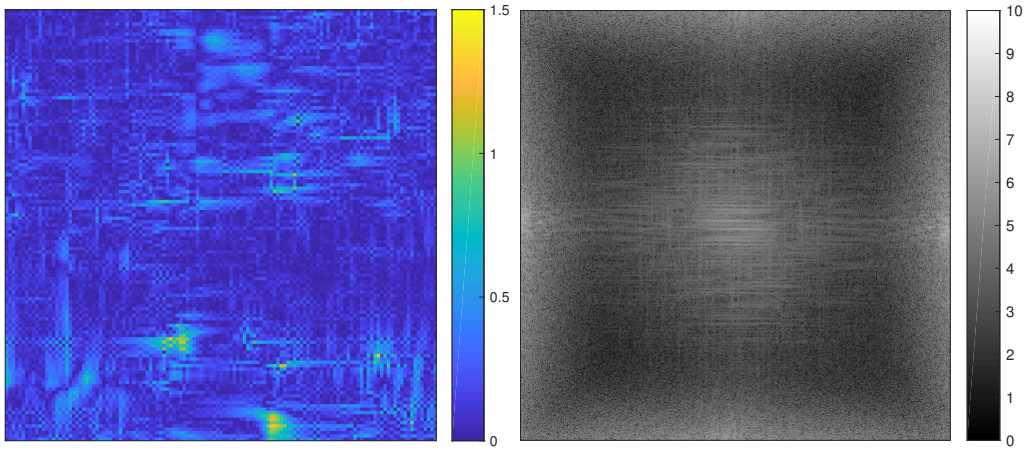
1.4 Contributions et publications

Liste des contributions. Les principales contributions de cette thèse, chapitre par chapitre, sont les suivantes:

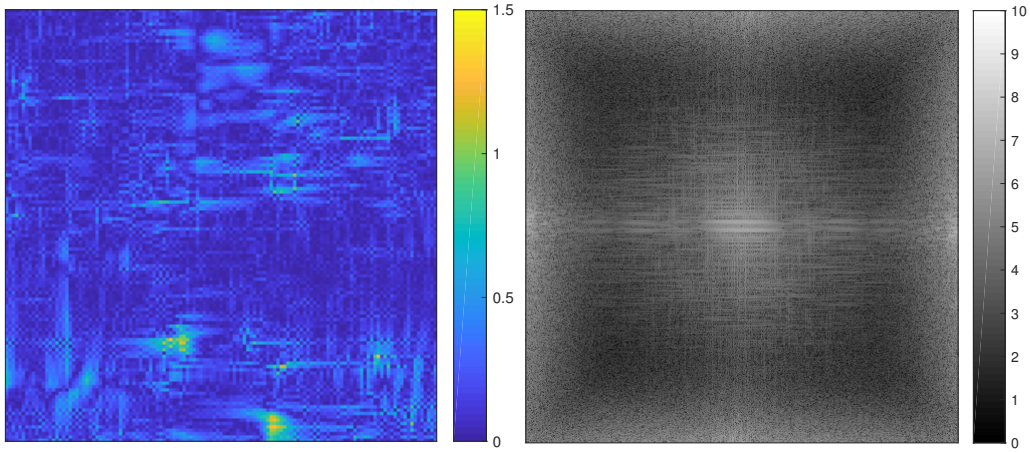
- **Chapitre 3**: Caractérisation des polynômes trigonométriques interpolateurs. Application à la transformation géométrique des images, au sur-échantillonnage et au sous-échantillonnage avec des calculs basés sur la DFT.
- **Chapitre 4:** Extrait de [15]. Des algorithmes basés sur la DFT pour le filtrage d'images à valeurs réelles. Justification théorique basée sur l'interpolation par polynôme trigonométrique. Théorie et implémentation fournie à l'adresse `http://www.ipol.im/pub/art/2016/116/`.
- **Chapitre 5:** Extrait de [14]. Description complète de l'interpolation par B-spline (théorie et pratique). Deux algorithmes de préfiltrage dont la précision est contrôlée. Implémentation fournie à l'adresse `http://www.ipol.im/pub/art/2018/221/`.
- **Chapitre 6:** Introduction des mesures de cohérences. Méthodes d'interpolation affinées ayant des meilleures mesures de cohérence.
- **Chapitre 7:** Extrait de [16]. Améliorations de l'algorithme *inverse compositional* en termes de précision et de rapidité. Implémentation et prépublication fournies à l'adresse `https://www.ipol.im/pub/pre/222/`.
- **Chapitre 8:** Méthode de recalage en deux étapes des images mosaïquées.
- **Chapitre 9:** Présentation de la régression à noyau. Introduction du filtre équivalent asymptotique dans le cas de la régression à noyau classique. Une méthode de fusion d'images rapide et à faible coût mémoire qui est conçue pour un grand nombre d'images.
- **Chapitre 10:** Justification expérimentale de notre méthode de fusion d'images. Détermination des meilleurs ordre et échelle sur des données synthétiques.
- **Chapitre 11:** Une méthode de formation d'image rapide et à faible coût mémoire qui est conçue pour un large nombre d'images RAW.

Liste des publications. Le travail durant cette thèse a donné lieu aux publications suivantes:

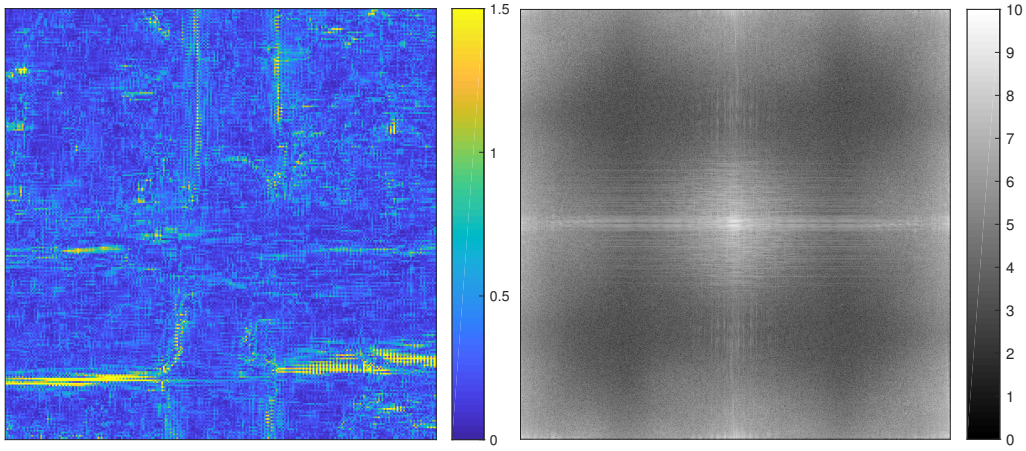
- T. Briand et J. Vacher. "How to Apply a Filter Defined in the Frequency Domain by a Continuous Function". Dans Image Processing On Line (IPOL), vol. 6, pp. 183–211, 2016.
- T. Briand et P. Monasse. "Theory and Practice of Image B-Spline Interpolation". Dans Image Processing On Line (IPOL), vol. 8, pp. 99–141, 2018.
- T. Briand. "Low Memory Image Reconstruction Algorithm from RAW Images". Dans IEEE Image, Video, and Multidimensional Signal Processing (IVMSP), 2018.



(a) Burst denoising (RMSD = 0.17)

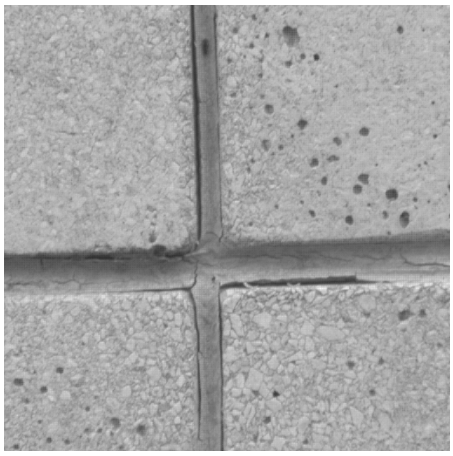


(b) ACT pour $\lambda = 1$ (RMSD = 0.19)

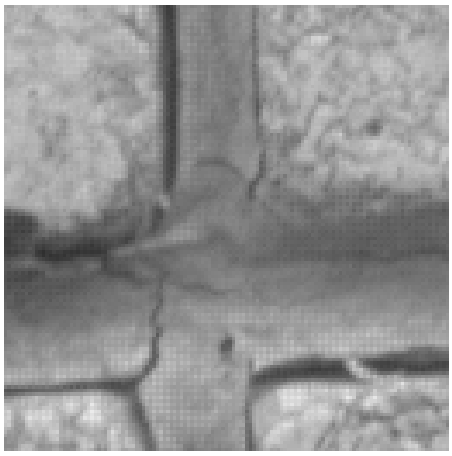


(c) ACT pour $\lambda = 2$ (RMSD = 0.48)

Figure 1.12: Différence entre notre méthode de fusion d'images et les méthodes ACT et *burst denoising* pour 200 images réelles. Les résultats sont très proches et ne peuvent être distingués à l'oeil nu. Par conséquent, les images de différences sont montrées. Elles ne sont pas composées de bruit. Pour $\lambda = 2$, la différence est plus importante en particulier près des discontinuités (où les structures de type *zipper* peuvent être vues).

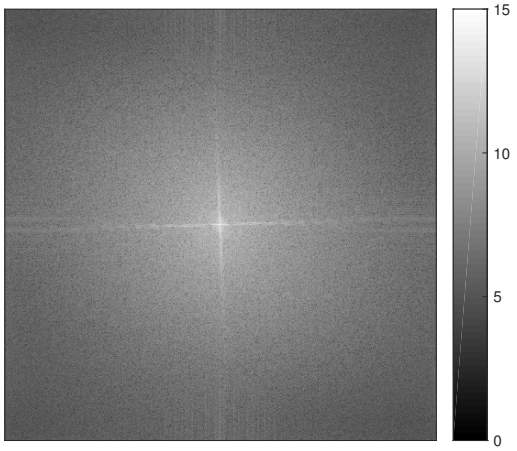


(a) Image mosaïquée de référence



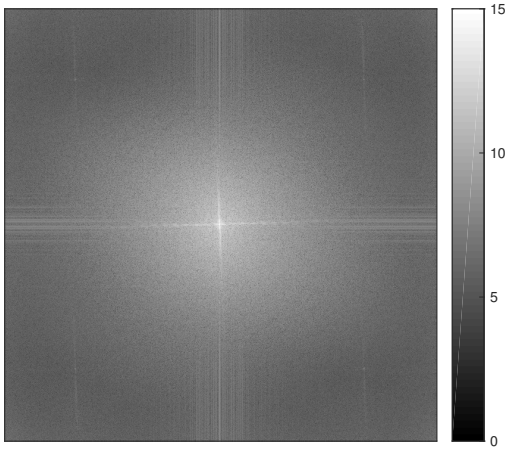
(b) Détails de (a)

(c) Image formée ($\lambda = 1$)



(d) Spectre de (a)

(e) Image formée ($\lambda = 1.5$)



(f) Spectre de (c)

Figure 1.13: Résultat de notre méthode de formation d'images appliquée à 200 images RAW. L'image mosaïquée de référence (en (a)) est de taille 512×512 et correspond à l'image après pré-traitement de l'image RAW de référence. Des détails dans une zone de taille 128×128 sont montrés en (b). La structure en mosaïque est clairement visible en (b), en particulier près des discontinuités. Les images formées, en (c) et (e), sont nettes, sans artefact et sans bruit apparent.

- T. Briand, G. Facciolo, et J. Sánchez. "Improvements of the Inverse Compositional Algorithm for Parametric Motion Estimation". Dans Image Processing On Line (IPOL), soumis.

Chapter 2

Introduction

The general objective of this thesis is to obtain high-quality images by digital processing. The quality of an image can be enhanced in single-image methods (e.g. [18, 25] for denoising, [47, 67, 135] for super-resolution and [3] for increasing the dynamic range). These methods are generally transform-based or exemplar-based. On the contrary multi-image methods produce a high-quality image from a set of lower quality images and are closer to the essence of photography, which proceed to an accumulation of photons. Such methods compensate for the shortcomings of the imaging system by an adequate digital processing of the accumulated data. This idea has been successfully exploited for increasing resolution [35, 87, 125], denoising [19, 71], reducing blur [26] or improving the image dynamic [2, 53, 81]. Most methods take as input preprocessed images, which may lead to artifacts. Therefore, good methods should rather handle RAW images [35, 36, 37, 48, 61, 131]. The counterpart of using RAW images is that such images may be more difficult to handle. Notably, most RAW images are mosaicked (or CFA) images where only one color intensity value is available per pixel.

The performances of image processing algorithms highly depend on the quality of the input images. In particular, noise and aliasing are two of the main sources of error. Noise introduces uncertainty in the measured samples, that differ from the actual values of the underlying signal. For instance, the influence of noise in stereovision accuracy is discussed in [101]. Noise may typically be reduced by combining the available noisy data. Aliasing arises when a signal is undersampled [59]. This introduces ambiguity in the reconstruction of the underlying continuous signal. Aliased images cannot be properly interpolated.

The aim of this thesis is to build the algorithms producing a high-quality color image, containing a low level of noise and aliasing, from a large sequence (e.g. hundreds or thousands) of RAW images taken with a consumer camera. We assume that the camera settings are fixed and that the images, representing a static scene, are taken quasi instantaneously. The images differ because of small motions of the camera, noise and small illumination variations. This is a challenging issue requiring to perform on the fly demosaicking [68], denoising and super-resolution.

Multi-image methods (from RAW images or not) can generally be decomposed into two main steps: the registration step, where the images are expressed in a common system of coordinates, and the combination step, where the data are combined to build an image. Even though some methods do not require a sub-pixel accuracy for the registration [119], it is in general a pre-requisite for achieving good performances. However, there is no standard and satisfactory method registering mosaicked images [48, 131]. Because of the particular content of these images, existing registration methods designed for classical grayscale or color images cannot be used directly.

Assuming the images are correctly registered, most combination methods (specific to mosaicked images or not) are designed for a small amount of input images [37, 38, 48, 86, 87, 91, 117, 118, 131]. They use an iterative scheme and require the availability of all the data in mem-

ory at once. They might be able to achieve high performance but the amount of accumulated data, i.e. the number of input images, is necessarily limited by the computer memory capacity.

A fast and low memory multi-image method requires a simpler combination method. For a large number of images it should be able to achieve similar performance as more advanced memory greedy methods. An image can be computed pixel-by-pixel without any iterative scheme using classical kernel regression [117]. This is the simplest and most efficient kernel regression method as it only takes into account the spatial distance but not the photometric distance. However, this provides low quality results in areas containing textures or edges. Actually, the final image is blurry. As shown in [41], each output pixel value is a local weighted average of the samples, i.e. obtained by an underlying local linear filtering. But the equivalent filter depends on the data spatial repartition, which varies with the pixel position.

In this thesis we break the two evoked limitations of existing methods. We propose an image formation algorithm from RAW images that processes the images sequentially so that the memory cost only depends on the size of the output image. After a preprocessing step, the mosaicked (or CFA) images are aligned in a common system of coordinates using a two-step registration method that we introduce. Then, a color image is computed by accumulation of the irregularly sampled data using classical kernel regression. Finally, the method blur is removed by applying the inverse of the corresponding asymptotic equivalent filter (that we introduce).

In this dissertation, each step of the method is introduced and analyzed separately. Its performance and accuracy are evaluated on synthetic and real data. In that way we manage to control the error at each step, and provide a proper analysis of the proposed improvements of existing methods. Generating synthetic data requires an interpolation method and controlling the interpolation error is crucial for analyzing the performance of our method. Therefore, we also study in detail interpolation methods. We derive from this study new fine-tuned interpolation methods that we use to generate synthetic data.

Part I (from Chapter 3 to Chapter 6) is devoted to interpolation methods. Part II (from Chapter 7 to Chapter 8) deals with registration methods. Finally, Part III (from Chapter 9 to Chapter 11) is dedicated to the image fusion methods.

2.1 Introduction to Part I: Interpolation

Interpolation consists in constructing new data points within the range of a discrete set of known data points. It is closely related to the concept of approximation [21], fitting [29] and extrapolation. In signal processing it is commonly expressed as the problem of recovering the underlying continuous signal from which the known data points are sampled. Under the assumption that the signal belongs to a given class of functions, the common principle of all interpolation schemes is to determine the parameters of the continuous signal representation. The most common interpolation methods are presented in [46, 123].

A continuous signal representation is required when one wishes to implement numerically an operator that is initially defined in the continuous domain. In particular, this representation is required when applying a geometric transformation to an image. Denote by $\sigma(\mathbb{R}^2)$ the set of bijective functions of $\mathbb{R}^2$ to itself. A function $\varphi \in \sigma(\mathbb{R}^2)$ is called a geometric transformation. Let $\underline{u}$ be an image of size $M \times N$. Applying the geometric transformation φ to $\underline{u}$ consists in resampling $\underline{u}$ at locations $\varphi^{-1}(k,l)$. As represented in Figure 2.1, in general $\varphi^{-1}(k,l) \in \mathbb{R}^2$ does not belong to the integer grid and a continuous representation $u : \mathbb{R}^2 \to \mathbb{R}$ of $\underline{u}$ is required. The transformed image $\underline{u}_\varphi$ is then defined as the image of size $M \times N$ verifying

$$(\underline{u}_\varphi)_{k,l} = u\left(\varphi^{-1}(k,l)\right). \tag{2.1}$$

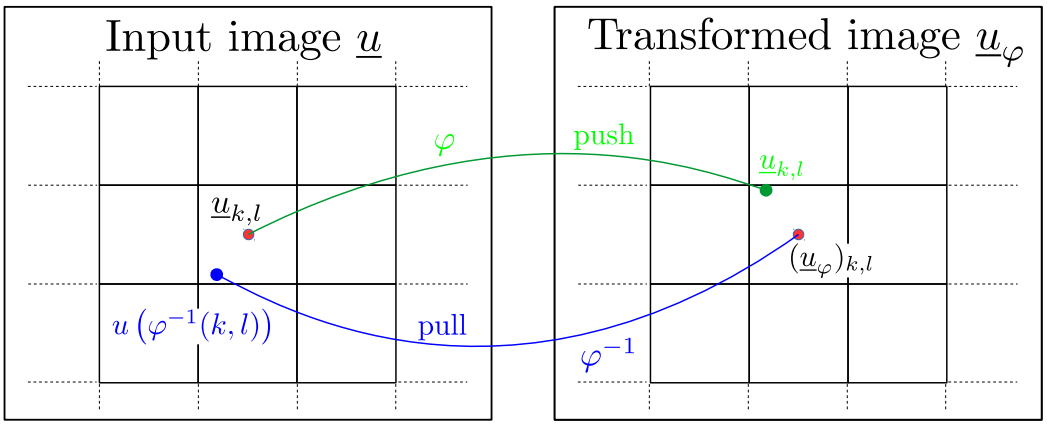


Figure 2.1: Geometric transformation of an image. To apply a transformation $\varphi \in \sigma(\mathbb{R}^2)$ to an image $\underline{u}$, the pixel values at locations $\varphi^{-1}(k,l)$ must be computed. As the samples of $\underline{u}$ are only known at integer locations, a continuous representation $u : \mathbb{R}^2 \to \mathbb{R}$ of $\underline{u}$ is required. This representation is obtained by an interpolation method. The samples $(\underline{u}_\varphi)_{k,l}$ are said to be pulled. In the transformed system of coordinates, the only pixel values that can be obtained without interpolation correspond to the pushed samples $\underline{u}_{k,l}$ located at $\varphi(k,l)$.

In Part II and Part III, we will evaluate the performance of the proposed methods on synthetic dataset composed of transformed images. This generation of synthetic data involves an interpolation method whose choice impacts the results. The interpolation error has to be taken into account. This is why in Part I we study in detail the trigonometric polynomial and B-spline interpolation methods. We derive from this study new fine-tuned interpolation methods.

2.1.1 Chapter 3: Trigonometric Polynomial Interpolation of Images

For 1D and 2D signals, the Shannon-Whittaker interpolation with periodic extension can be formulated as a trigonometric polynomial interpolation (TPI) [1]. In Chapter 3, we introduce the theory of TPI of images and some of its applications. First, the trigonometric polynomial interpolators of an image are characterized and it is shown that there is an ambiguity as soon as one size of the image is even. Three classical choices of interpolator for real-valued images are presented and cases where they coincide are pointed out. Then, TPI is applied to the geometric transformation of images, to up-sampling and to down-sampling. General results are expressed for any choice of interpolator but more details are given for the three proposed ones. It is proven that the well-known computations based on the discrete Fourier transform (DFT) have to be slightly adapted.

The application of TPI to filtering is detailed separately in Chapter 4. The performances and the limits of TPI are discussed in Chapter 6.

2.1.2 Chapter 4: Filtering using Trigonometric Polynomial Interpolation

Chapter 4 is taken from [15] and is an application of Chapter 3. We propose algorithms for filtering real-valued images, when the filter is provided as a continuous function $\phi : [-\pi,\pi]^2 \to \mathbb{C}$ defined in the Nyquist frequency domain. This problem is ambiguous because images are discrete entities and there is no unique way to define the filtering. We provide a theoretical framework designed to analyse the classical and computationally efficient filtering implementations based on DFTs.

In this framework, the filtering is interpreted as the convolution $f_\phi * P$ of a distribution f_ϕ, standing for the filter, with a trigonometric polynomial interpolator P of the image. The various plausible interpolations and choices of the distribution lead to three equally licit algorithms which can be seen as method variants of the same standard filtering algorithm, which is described in Algorithm 2.1.

Algorithm 2.1: Standard filtering algorithm

Input : A real-valued image $\underline{u}$ of size $M \times N$, a filter $\phi : [-\pi,\pi]^2 \to \mathbb{C}$ (continuous function) and the number $j \in \{1,2,3\}$ of the method variant.

Output: The filtered image $\underline{v}$ of size $M \times N$ corresponding to the method j.

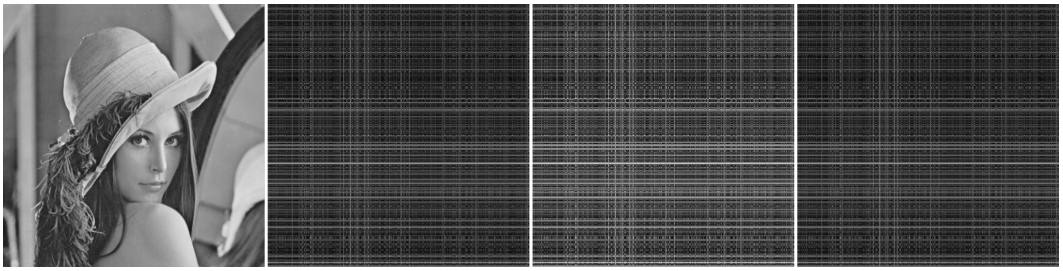
1 Compute $\tilde{u} = \mathcal{F}_{M,N}(\underline{u})$ the DFT of $\underline{u}$.

2 Compute $S = (S_{m,n})_{(m,n)\in\hat{\Omega}_{M,N}}$ the spectral samples corresponding to method j (see Chapter 4 for the details).

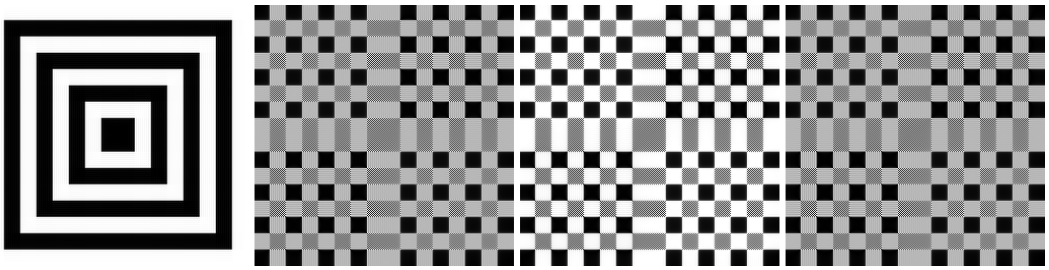
3 Compute $\tilde{v} = (\tilde{u}_{m,n} S_{m,n})_{(m,n)\in\hat{\Omega}_{M,N}}$ the element-wise multiplication of $\tilde{u}$ and S.

4 Compute $\underline{v} = \mathcal{F}^{-1}_{M,N}(\tilde{v})$ the inverse DFT of $\tilde{v}$.

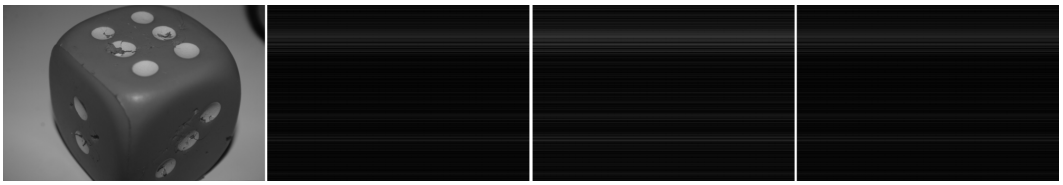
No method should be preferred *a priori* to the others; the choice depends on the application. In practice, the image differences arising for different boundary DFT coefficients are not visible to the naked eye. We demonstrate this on several experimental configurations by varying the input image and the considered filter. For instance, the results for the shift filter of parameter $(1/4, 1/4)$ are compared in Figure 2.2 and Table 2.1. The overall maximal difference is around two grey levels. In some cases however, we discuss how the choice of the variant may affect fundamental properties of the filtering. We provide an implementation of the algorithms at http://www.ipol.im/pub/art/2016/116/.



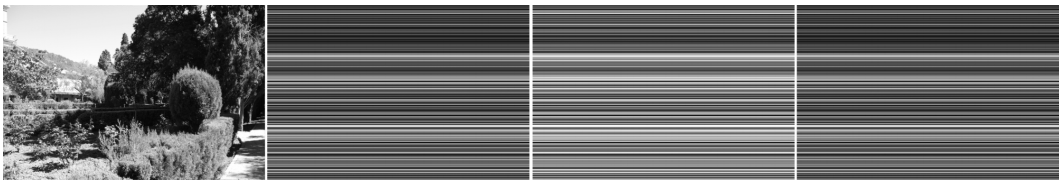
(a) Lenna



(b) Square



(c) Dice



(d) Garden

Figure 2.2: Comparison of the three filtering methods for the shift filter of parameter $(1/4, 1/4)$. From left to right we display for each input image: the real part of the first method and the relative difference images $\Delta_{\text{rel}}^{1,2}$, $\Delta_{\text{rel}}^{1,3}$ and $\Delta_{\text{rel}}^{2,3}$. The difference images are multiplied by 10^5 for visualization purposes. The method differences, which come from the boundary DFT coefficients, are not visible to the naked eye.

		Lenna	Square	Dice	Garden
1 vs 2	d	0.77 (0.34%)	1.6 (0.41%)	0.13 (0.067%)	1.4 (0.43%)
	m_d	0.12 (0.052%)	0.55 (0.14%)	0.020 (0.010%)	0.30 (0.094%)
1 vs 3	d	1.1 (0.48%)	2.2 (0.58%)	0.18 (0.093%)	2.0 (0.61%)
	m_d	0.17 (0.075%)	0.77 (0.20%)	0.029 (0.015%)	0.42 (0.13%)
2 vs 3	d	0.77 (0.34%)	1.6 (0.41%)	0.13 (0.067%)	1.4 (0.43%)
	m_d	0.12 (0.053%)	0.55 (0.14%)	0.020 (0.010%)	0.30 (0.094%)

Table 2.1: Maximum difference d and mean difference m_d between the three filtering methods for the shift filter of parameter $(1/4, 1/4)$. Relative values are in brackets. The overall maximal difference is around two grey levels so that no difference can be seen to the naked eye.

2.1.3 Chapter 5: B-spline Interpolation

Chapter 5 is taken from [14]. B-spline interpolation is a widely used non band-limited alternative to Shannon-Whittaker interpolation. For 1D signals it is defined as follows.

Definition 2.1. *The normalized B-spline function of order n, noted* $\beta^{(n)}$*, is defined recursively by*

$$\beta^{(0)}(x) = \begin{cases} 1, & -\frac{1}{2} < x < \frac{1}{2} \\ \frac{1}{2}, & x = \pm\frac{1}{2} \\ 0, & \text{otherwise} \end{cases} \qquad \text{and for } n \geq 0, \quad \beta^{(n+1)} = \beta^{(n)} * \beta^{(0)} \tag{2.2}$$

where the symbol $*$ *denotes the convolution operator.*

Definition 2.2 (B-spline interpolation)**.** *The B-spline interpolate of order n of a discrete signal* $f \in \mathbb{R}^{\mathbb{Z}}$ *is the spline* $\varphi^{(n)}$ *of degree n defined for* $x \in \mathbb{R}$ *by*

$$\varphi^{(n)}(x) = \sum_{i \in \mathbb{Z}} c_i \beta^{(n)}(x - i) \tag{2.3}$$

where the B-spline coefficients $c = (c_i)_{i \in \mathbb{Z}}$ *are uniquely characterized by the interpolating condition*

$$\varphi^{(n)}(k) = f_k, \quad \forall k \in \mathbb{Z}. \tag{2.4}$$

In Chapter 5 we explain how the B-spline interpolation of signals and images can be efficiently performed by linear filtering. In the seminal two-step method proposed by Unser et al. in 1991 [129], the B-spline coefficients c are first computed in a prefiltering step. Then, interpolated values are computed using (2.3). We propose two slightly different prefiltering algorithms whose precisions are proven to be controlled thanks to a rigorous boundary handling. The first algorithm is general and works for any boundary extension while the second is applicable under specific assumptions.

Chapter 5 contains all the information, theoretical and practical, required to perform efficiently B-spline interpolation for any order and any boundary extension. We describe precisely how to evaluate the kernel and to compute the B-spline interpolator parameters.

In an experimental part, we show that increasing the order improves the interpolation quality. For instance, B-spline interpolation approaches the Shannon-Whittaker interpolation as the order increases [5]. An illustration of the decay of the difference with the order is shown in Figure 2.3.

As a fundamental application we also provide an implementation of homographic transformation of images using B-spline interpolation at `http://www.ipol.im/pub/art/2018/221/`. An example of homographic transformation is shown in Figure 2.4.

2.1.4 Chapter 6: Consistency of Interpolation Methods

There is no universal procedure for evaluating the quality and performance of an interpolation method. In Chapter 6 we introduce a new quantity: the consistency measurement. For a given image, the consistency measurement $\mathcal{CM}(\varphi)$ for the transformation $\varphi \in \sigma(\mathbb{R}^2)$ measures the error after applying successively φ, a crop (removing boundary artifacts) and the inverse φ^{-1}. An average over random transformations $(\varphi_i)_{1 \leq i \leq N_{\text{transf}}}$ is made to remove the dependency on the transformation. We define the consistency measurement $\mathcal{CM}$ as

$$\mathcal{CM} = \frac{1}{N_{\text{transf}}} \sum_{i=1}^{N_{\text{transf}}} \mathcal{CM}(\varphi_i). \tag{2.5}$$

A more precise measurement discarding very high-frequency artefacts is obtained by clipping the spectrum of the difference. The variant is called the clipped consistency measurement and is noted $\mathcal{CM}^c$.

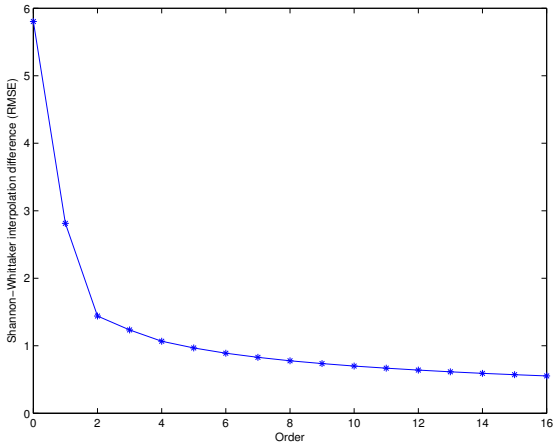


Figure 2.3: Decay of the difference between the Shannon-Whittaker interpolation and the B-spline interpolation for $n \in \{0, \ldots, 16\}$. It is obtained by comparing transformations of the *Lenna* image. The RMSE is taken on the central part of the images so that the boundary extension has a negligible influence. The periodic extension is used for both interpolation methods.

Figure 2.4: Example of homographic transformation using B-spline interpolation of order 11 (with half-symmetric boundary condition).

We also propose new fine-tuned interpolation methods that are based on the DFT zoom-in and pre-existing interpolation methods. The zoomed version of an interpolation method is obtained by applying it to the DFT zoom-in of the image. In the periodic plus smooth version of interpolation methods, the non-periodicity is handled by applying the zoomed version to the periodic component and a base interpolation method to the smooth component.

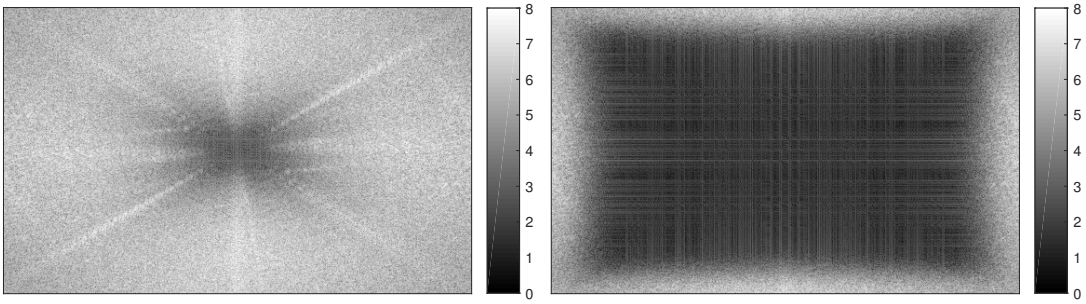
In an experimental part, we show that the proposed fine-tuned methods have better consistency measurements than their base interpolation methods and that the error is mainly localized in a small high-frequency band. This is can be seen in the results presented in Table 2.2. The spectrum of the difference image (from which the consistency measurements are computed) is shown in Figure 2.5 for various interpolation methods.

Based this analysis, we recommend to use the periodic plus smooth versions of high order B-spline interpolation. It is more efficient and provides better results than trigonometric polyno-

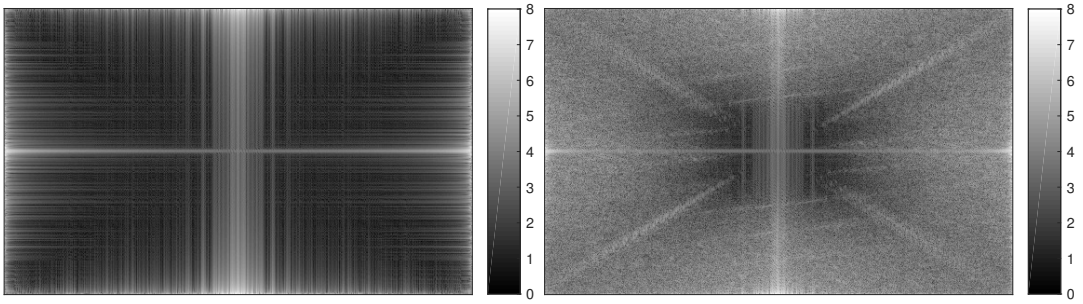
mial interpolation. In Chapter 10, such a method will be used for generating synthetic data and for the burst denoising method.

	$\mathcal{CM}$	$\mathcal{CM}^c$	Time (s)
spline1	2.53480	2.50077	225
bic	1.18612	1.13573	336
spline3	0.75702	0.67967	283
spline11	0.35355	0.19626	796
tpi	0.20564	0.06345	4767
spline1-z2	0.86002	0.81810	1308
bic-z2	0.25140	0.15072	1425
spline3-z2	0.20585	0.06816	1407
spline11-z2	0.20564	0.06345	2053
p+s-spline1	0.84660	0.81640	2039
p+s-spline1-bic	0.84660	0.81640	2159
p+s-bic	0.18008	0.13811	2355
p+s-spline3	0.09417	0.03007	2199
p+s-spline3-bic	0.09417	0.03007	2259
p+s-spline11	0.08785	0.01506	3360
p+s-spline11-bic	0.08785	0.01506	2907
p+s-tpi-spline1	0.08785	0.01506	5865
p+s-tpi-bic	0.08785	0.01506	5960
p+s-tpi-spline3	0.08785	0.01506	5926
p+s-tpi-spline11	0.08785	0.01506	6443

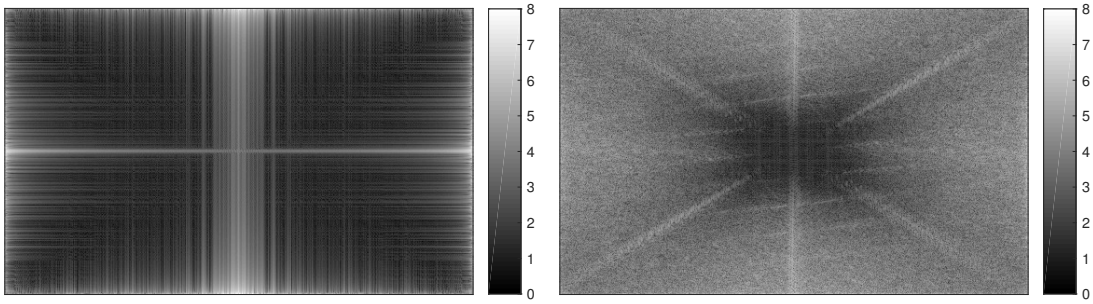
Table 2.2: Evaluation of the consistency measurements for $N_{\text{transf}} = 1000$ random homographies. The evaluation is made excluding a boundary band of width $\delta = 20$ to render results independent of the boundary extension choice. The displayed time corresponds to the computation time, in seconds, for all the transformations. This can also be interpreted as the average time, in milliseconds, required to apply a transformation and its inverse. The *tpi* and *bic* methods represent respectively the trigonometric polynomial and the bicubic interpolations. The suffix *z2* corresponds to zoomed versions. For the periodic plus smooth version we use the prefix *p+s*. The first method is the one used on the periodic component. It is clear that fine-tuned methods have a better consistency measurement. As the errors on the clipped versions (column 2) are significantly smaller, we conclude that the error is mainly localized in a small high-frequency band. The periodic plus smooth methods provide better results because the non-periodicity of the image is correctly handled.



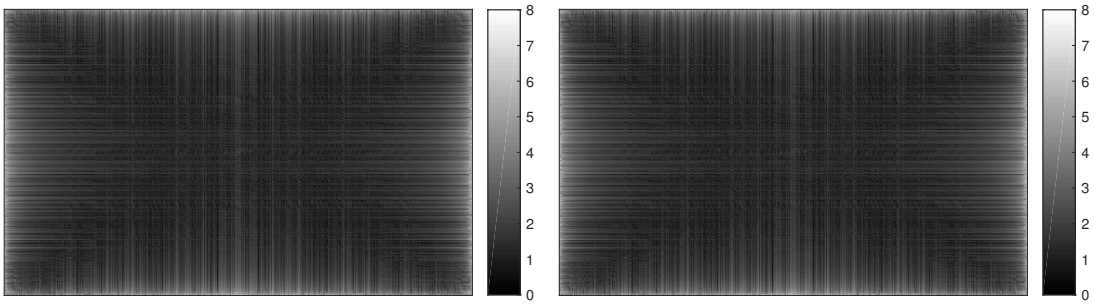
(a) *bic* (1.37, 1.31) (b) *spline11* (0.423, 0.236)



(c) *tpi* (0.211, 0.0584) (d) *bic-z2* (0.253, 0.145)



(e) *spline11-z2* (0.211, 0.0584) (f) *p+s-bic* (0.191, 0.135)



(g) *p+s-spline11* (0.117, 0.0187) (h) *p+s-tpi-spline11* (0.117, 0.0187)

Figure 2.5: Spectrum of the difference image (from which the consistency measurements are computed) for various interpolation methods. The spectrums correspond to the unnormalized discrete Fourier transform in logarithmic scale $u \mapsto \log(1+u)$. The geometric transformation φ used is an homography. The values between parentheses correspond to the consistency measurements $(\mathcal{CM}(\varphi), \mathcal{CM}^c(\varphi))$, which are the L2 energies respectively without and with removing the band of 5% highest frequencies. For the *bic*, *bic-z2* and *p+s-bic* methods i.e. the methods where the possibly zoomed images are interpolated by bicubic interpolation, the spectrum of the difference is high everywhere except in a small low-frequency region. For the *spline11* method the corresponding region is larger. For the *tpi* and *spline11-z2* methods, the spectrum has non-negligible values in a band around the Nyquist frequency and in a vertical and horizontal cross structure. The cross structure is also visible in the *bic-z2* method and is due to the incorrect handling of the non-periodicity. Indeed it disappears in the *p+s-spline11* and *p+s-tpi-spline11* methods.

2.2 Introduction to Part II: Registration

Image registration is one of the most widely used techniques in computer vision. The objective of image registration methods is to find the optimal transformation that puts in correspondence the pixels of two (or more) images. Image registration is required for instance when images are acquired at different times, from distinct viewpoints or by different sensors. An accurate and efficient estimation is important in problems such as optical flow estimation [10, 56], object tracking [134], video stabilization [78], image stitching [115] or 3D reconstruction [52]. The task is difficult because it deals with problems like occlusions, noise, local brightness changes or spurious motions.

Surveys of existing methods are provided in [75, 88, 137]. The methods can be classified into intensity-based and feature-based [27]. Intensity-based methods are usually faster but more sensitive to brightness changes and outliers. Feature-based methods are typically more sensitive to noise and motion blur, but allow to estimate stronger deformations [114].

In Part III, we will consider multi-image methods that require an efficient registration step. The input images will only differ because of small motions of the camera, noise, quantization, and possibly small illumination variations. This is why in Part II we consider intensity-based methods. First, we improve the inverse compositional algorithm. Then, we develop a two-step methods for mosaicked (or CFA) images.

2.2.1 Chapter 7: Modified Inverse Compositional Algorithm

First introduced in [8, 122], the inverse compositional algorithm for parametric motion estimation is an improvement of the classical intensity-based method of Lucas-Kanade [74]. At each step of its iterative scheme it solves a minimization problem equivalent to Lucas-Kanade but more efficiently. More precisely, at a given step $j \geq 1$, the idea is to refine the current estimated transformation $\Psi(\cdot;\mathbf{p}_{j-1})$ with an inverted incremental transformation (hence the name of the algorithm) $\Psi(\cdot;\Delta\mathbf{p}_j)^{-1}$, i.e.,

$$\Psi(\cdot;\mathbf{p}_j) = \Psi(\cdot;\mathbf{p}_{j-1}) \circ \Psi(\cdot;\Delta\mathbf{p}_j)^{-1}. \tag{2.6}$$

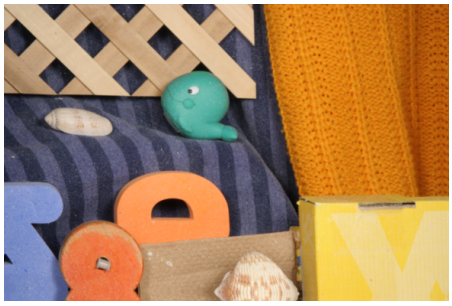
The increment $\Delta\mathbf{p}_j$ approximates the minimizer of the incremental energy

$$\Delta\mathbf{p} \in \mathbb{R}^n \mapsto E_1(\Delta\mathbf{p};\mathbf{p}_{j-1}) = \sum_{\mathbf{x}\in\Omega} \rho\left(\|I_2(\Psi(\mathbf{x};\mathbf{p}_{j-1})) - I_1(\Psi(\mathbf{x};\Delta\mathbf{p}))\|^2\right) \tag{2.7}$$

where $\rho : \mathbb{R}^+ \to \mathbb{R}^+$ is called the error function. Better precision for large motions can be obtained with a coarse-to-fine multiscale approach as in [104].

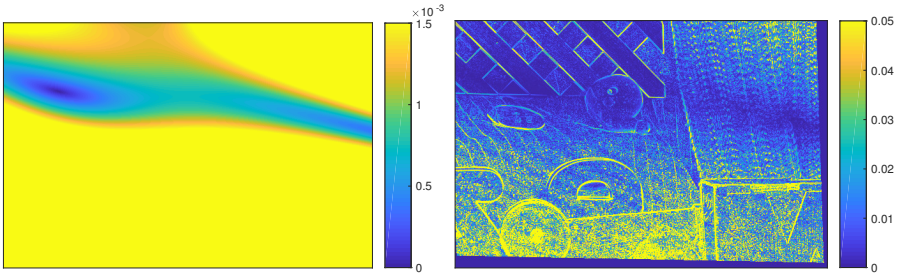
In Chapter 7, we introduce several improvements of the inverse compositional algorithm. We propose an improved handling of boundary pixels, a different color handling and gradient estimation, and the possibility to skip scales in the multiscale coarse-to-fine scheme. In an experimental part, we analyze the influence of the modifications. We find that our estimation accuracy is improved by a factor larger than 1.3 while the computation time reduced by a factor larger than 2.2 for color images. An example of motion estimation on synthetic data, with and without modifications, is presented in Figure 2.6. The modifications lead to significantly better results.

The modified inverse compositional algorithm will be used in Chapter 8 in the proposed two-step method for mosaicked images.

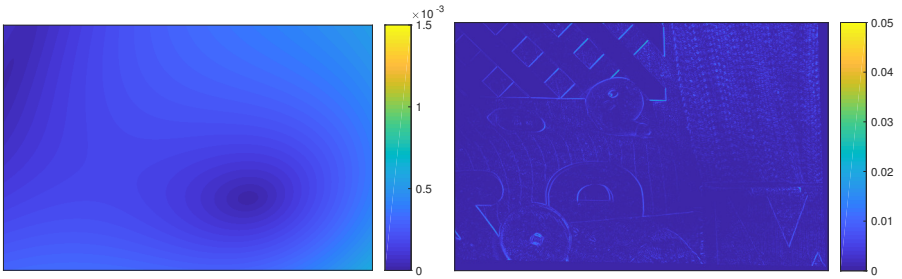


(a) Reference image $I_1 = I_2(\Psi(\cdot;\mathbf{p}^\star))$

(b) Warped image I_2



(c) End-point error field $\mathrm{EPE}(\mathbf{p},\mathbf{p}^\star)(\mathbf{x})$ (left) and residual (right) using the IC algorithm



(d) End-point error field $\mathrm{EPE}(\mathbf{p},\mathbf{p}^\star)(\mathbf{x})$ (left) and residual (right) using the modified IC algorithm

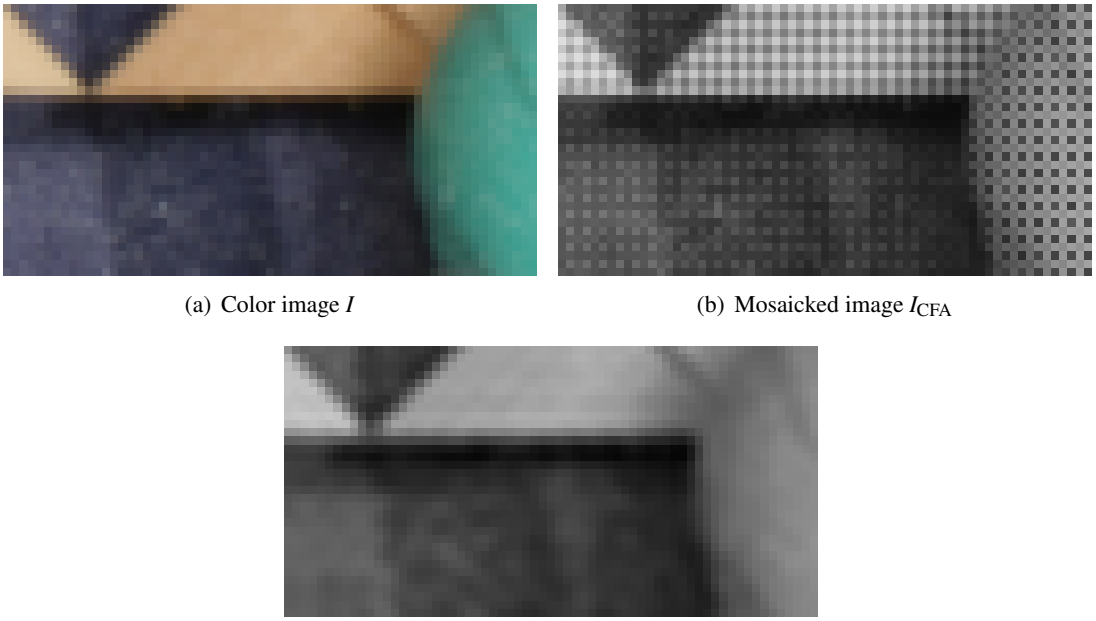
Figure 2.6: Example of motion estimation on synthetic data. The reference image $I_1 = I_2(\Psi(\cdot;\mathbf{p}^\star))$ is an example of synthetic image related to I_2 by an homography. On the second and third line, the estimated motion $\mathbf{p}$ is obtained using either the inverse compositional algorithm or our modified inverse compositional algorithm. For both methods, the right image is actually the root mean square over the channels of the residual $I_1 - I_2(\Psi(\cdot;\mathbf{p}))$, which is obtained by bicubic interpolation. Without modification, we have EPE=0.00460 and $\mathrm{RMSE}(I_1(\mathbf{x}), I_2(\Psi(\mathbf{x};\mathbf{p}))) = 0.042838$. With modification, we have EPE=0.00022 and $\mathrm{RMSE}(I_1(\mathbf{x}), I_2(\Psi(\mathbf{x};\mathbf{p}))) = 0.001790$.

2.2.2 Chapter 8: Registration of Mosaicked (or CFA) Images

A mosaicked (or CFA) image is a digital image where only one of the three color channels has been captured at each pixel location. The corresponding grayscale image has a mosaic structure caused by the color filter array (CFA). The acquired RAW image is a typical example of mosaicked image. The most common pattern for the CFA, and the one that is considered here, is the Bayer pattern [12]. In many applications, the geometric transformation between two mosaicked images has to be estimated without knowledge of the underlying color images. Unfortunately there is no standard and satisfactory registration method for mosaicked images. Existing registration methods designed for classical images cannot be used directly, and a preprocessing step is required.

In Chapter 8 we introduce two-step methods for the registration of mosaicked images. First, the two mosaicked images are converted into non-mosaicked (grayscale) images by lowpass filtering. According to [6], these filtered images contain an estimate of the luminance information in the mosaicked images. An example of conversion is shown in Figure 2.7. Then, the transformation is estimated by applying a pre-existing registration method designed for classical images.

The performances of the proposed methods are evaluated experimentally for several lowpass filters and pre-existing registration methods. We conclude that a perfect lowpass filter should be applied and that the modified inverse compositional algorithm with robust error function (see Chapter 7) should be used. A comparison of the performance using different base registration methods is shown in Table 2.3. The recommended method is both accurate and efficient, and will be used in our image formation algorithm from RAW images (see Chapter 11).



(a) Color image I

(b) Mosaicked image I_{CFA}

(c) Perfect lowpass filtering of I_{CFA}

Figure 2.7: Conversions from mosaicked to non-mosaicked images. The mosaicked image I_{CFA} in (b) is obtained by applying the Bayer filter to the color image I in (a). The image in (c) is obtained after perfect lowpass filtering of (b).

		SIFT + RANSAC	IC	M-IC
$\sigma = 0$	EPE	0.01932	0.01173	**0.00360**
	Time	1298	458	**149**
$\sigma = 3$	EPE	0.03002	0.01309	**0.00559**
	Time	1334	453	**151**
$\sigma = 5$	EPE	0.04929	0.01678	**0.00979**
	Time	1393	451	**153**
$\sigma = 10$	EPE	0.15314	0.02391	**0.01550**
	Time	1671	447	**165**
$\sigma = 20$	EPE	0.49535	0.04903	**0.03352**
	Time	2575	424	**204**
$\sigma = 30$	EPE	0.83937	0.07455	**0.05327**
	Time	3029	463	**344**
$\sigma = 50$	EPE	1.84189	0.13512	**0.10812**
	Time	3003	579	**531**

Table 2.3: Influence of the base registration method for the two-step registration method of mosaicked images. The conversion to non-mosaicked images is done by perfect lowpass filtering. For the inverse compositional (IC) and the modified inverse compositional (M-IC) algorithms, the Lorentzian error function is used. The displayed computation time corresponds to the CPU time used for the $N_{\text{images}} = 1000$ motion estimations and is expressed in seconds. Note that it also corresponds to the average computation time per image in milliseconds. The accuracy is evaluated in terms of end-point error (EPE). The M-IC algorithm gives the best results for all noise levels σ both in terms of efficiency and accuracy. Thanks to the correct handling of the boundary pixels and to the prefiltering before the gradient estimation, it is able to achieve a precise motion estimation in only a few iterations even if the non-mosaicked images contain artifacts.

2.3 Introduction to Part III: Image Fusion/Formation

The two main steps of multi-image methods are the registration and the combination steps. After the registration, the pixel values can be expressed in the reference system of coordinates either by pulling or by pushing (see Figure 2.1). The pulled intensity values are computed by performing a geometric transformation of the images and thus by using an interpolation method. For instance, in burst denoising methods [19, 71] the fused image is computed by an averaging of the transformed images. This provides poor results when the images are aliased and cannot be used in a super-resolution context. On the contrary, the combination of pushed samples is open to super-resolution. The pushed samples define a set of irregularly sampled data (see example of Figure 2.8). Computing an intensity value at an arbitrary location is called irregularly sampled data fitting (or approximation). An image is obtained by computing intensities in a regular grid.

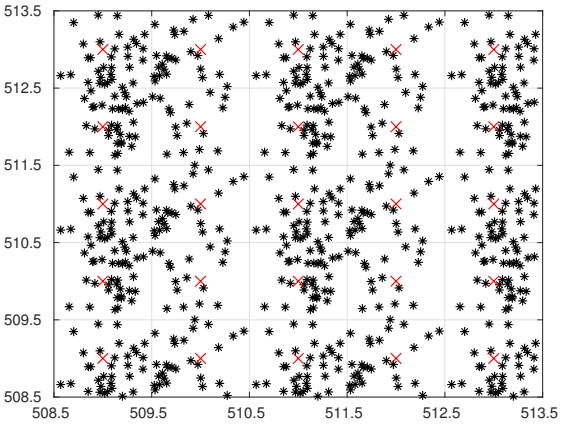


Figure 2.8: Example of spatial repartition of irregularly sampled data. To compute the intensity values at integer locations (red crosses), an irregularly sampled data fitting method has to be used.

2.3.1 Chapter 9: Fast and Low Memory Image Fusion using Classical Kernel Regression

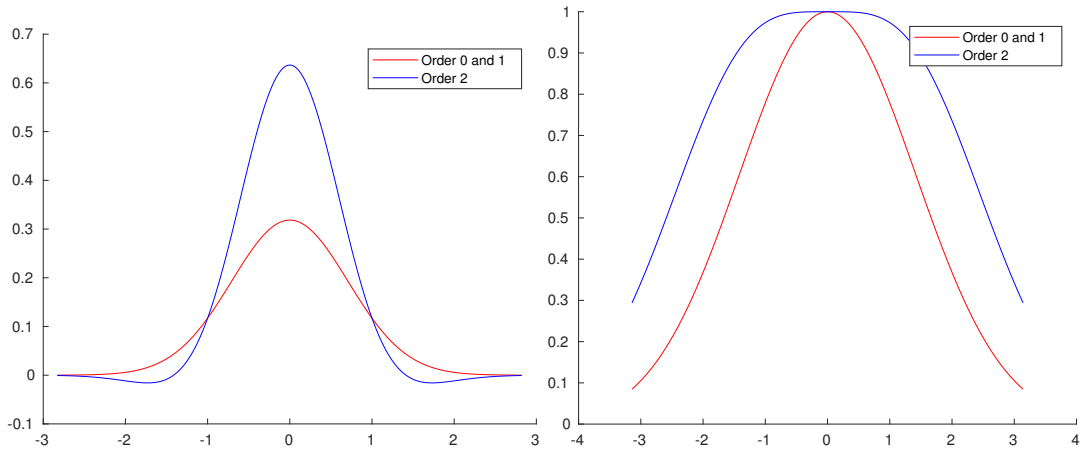
In Chapter 9 we consider irregularly sampled data fitting by kernel regression (KR) [117] and propose a fast and low memory image fusion method that is designed for a large number of images.

Using KR, the intensity value of the data is locally approximated by a polynomial expansion (of order at most 2 in the following), whose coefficients are obtained by solving a weighted linear regression (with weights built from a kernel function). We show that the linear systems involved can be obtained by a data accumulation.

Classical kernel regression (CKR) is the most simple and efficient case where the weights only depend on the data spatial repartition. As it is equivalent to a local linear filtering [41], CKR introduces blur. We introduce the asymptotic equivalent filter (AEF), an approximation of the actual equivalent filter. An example of AEF is shown in Figure 2.9.

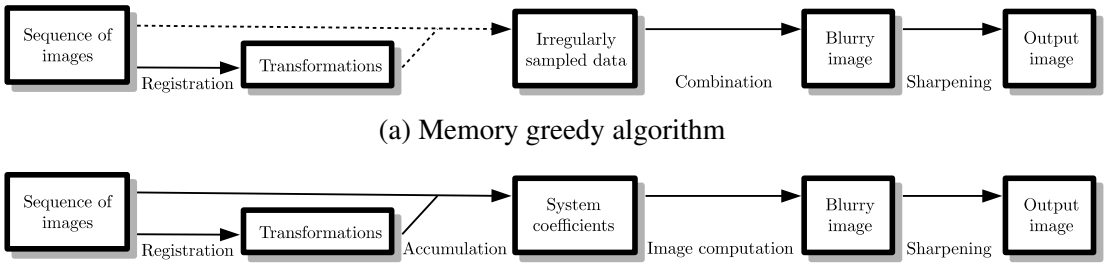
In the proposed image fusion algorithm the combination part, using CKR with a Gaussian kernel, is split into an accumulation part, where the images are processed sequentially, and an image computation part. The blur, introduced by CKR, is inverted by applying the inverse of the AEF. The main steps of our method are summarized in Figure 2.10.

The registration method used is the one introduced in Chapter 7. The choice of the optimal parameters and the evaluation of the performance of our algorithm are the subjects of Chapter 10. The combination part of our image fusion method is adapted to mosaicked images in Chapter 11.



(a) $x \mapsto \tilde{w}_{\sigma_s,\infty}(x,0)$ (b) $x \mapsto \mathcal{F}(\tilde{w}_{\sigma_s,\infty})(x,0)$

Figure 2.9: Asymptotic equivalent filter $\tilde{w}_{\sigma_s,\infty}$ for the Gaussian kernel of standard deviation $\sigma_s = 0.7$. In both spatial (a) and Fourier domains (b), the functions are radial so that only the slice $y = 0$ is shown. They decrease as $\|\mathbf{x}\|$ increases. In particular, the filtering using the AEF attenuates the high-frequency content.



(a) Memory greedy algorithm

(b) Our algorithm

Figure 2.10: Main steps of our image fusion method. (a) and (b) are theoretically equivalent. The combination from the irregularly sampled data is replaced by the accumulation step, where the images are sequentially processed to get the system coefficients, and the image computation step, where the systems are solved.

2.3.2 Chapter 10: Experimental Evaluation of our Image Fusion Algorithm

In Chapter 10 we evaluate experimentally the performance of our proposed image fusion algorithm on synthetic and real data. Our analysis of the synthetic case shows that the sharpening step is crucial to obtain a blur-free denoised image. We also figure out the best configuration for the order and the scale, depending on the sub-sampling factor. Assuming a uniform repartition of the samples, we conclude that it is better to use the order 0 when no super-resolution is required, and the order 2 for super-resolution. We find that a Gaussian kernel scale of about 0.7 is the best choice.

The result using our method for 200 real images of size 512×512 is compared to the ones of a burst denoising method and the ACT in Figure 2.12 for a zoom of $\lambda \in \{1,2\}$. As they cannot be distinguished to the naked eye, the difference with our method is shown. The computation times and memory requirements of the methods are following:

		Burst denoising	ACT	Our method (KR)
$\lambda = 1$	Time (s)	442	1784	**135**
	Max memory (kB)	175468	5562344	**82192**
$\lambda = 2$	Time (s)	ND	1942	**444**
	Max memory (kB)	ND	6224528	**1200492**

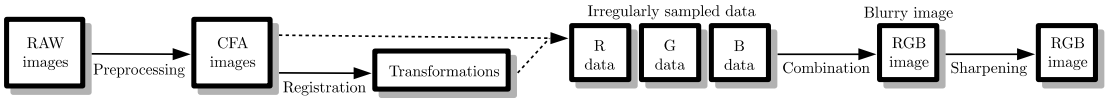
We show that for a large amount of data (synthetic or real) our image fusion algorithm provides similar results as slower and memory greedy methods. The residual noise on both synthetic and real examples decreases as expected and our algorithm can perform super-resolution.

From the experiments on real data we see that the performance of image fusion methods is limited by uncontrolled processes (demosaicking, JPEG compression, 8-bit quantization). This is why we propose in Chapter 11 an image formation algorithm taking RAW images as input.

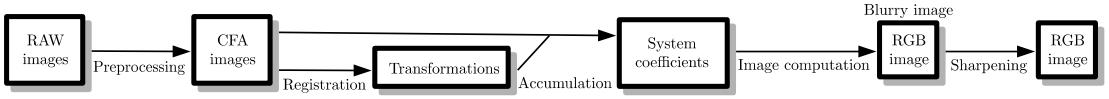
2.3.3 Chapter 11: Image Formation from a Large Sequence of RAW images

In Chapter 11 we propose an image formation method from a large sequence of RAW images. This is an adapted version of our image fusion method, described in Chapter 9, for RAW images and it follows the work of [13].

The RAW images are first preprocessed to transform the noise curve and to adjust the histograms. Then the two-step registration method for mosaicked images, introduced in Chapter 8, is used to align the images. As in Chapter 9, the images are combined by classical kernel regression of second order with Gaussian kernel. The process has two stages. The first one is an accumulation process where the images are processed sequentially. In the second stage, the blur introduced by the classical kernel regression is inverted by a sharpening filter. The main steps of our image formation method from RAW images are shown in Figure 2.11.



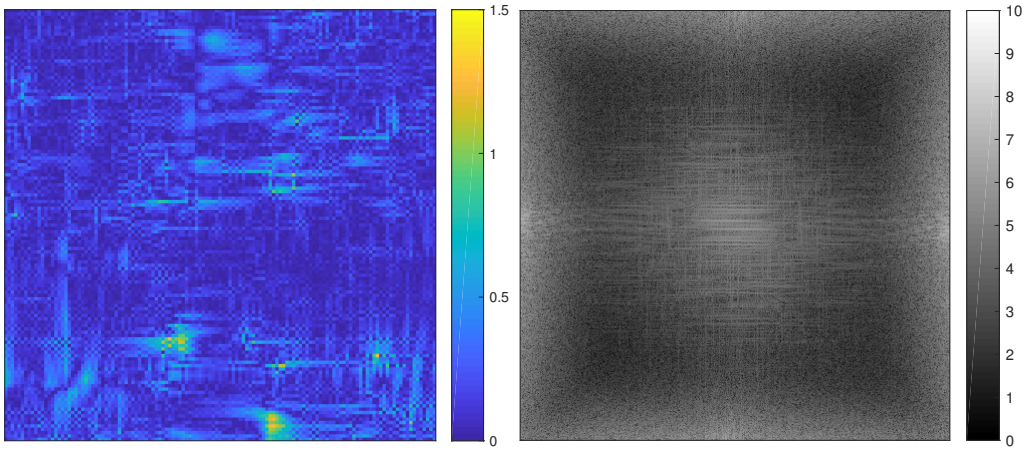
(a) Memory greedy algorithm



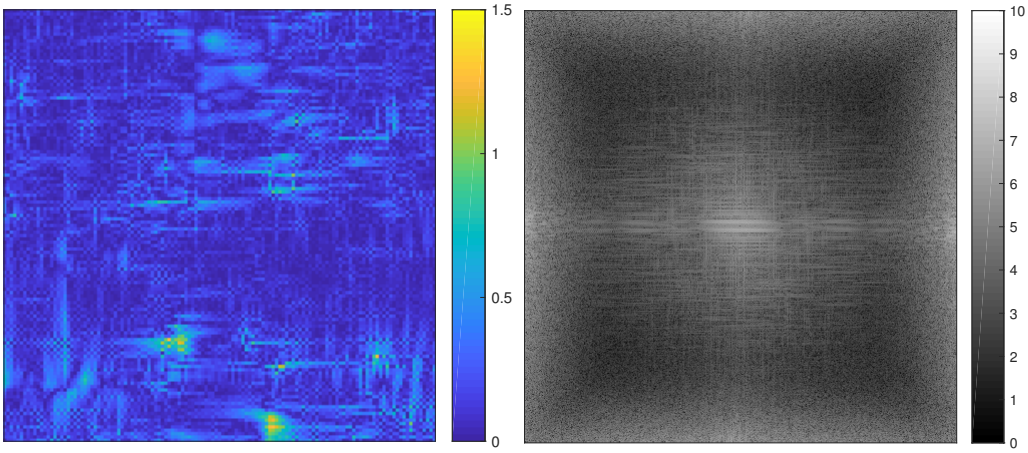
(b) Our algorithm

Figure 2.11: Main steps of our image formation method from RAW images. (a) and (b) are theoretically equivalent. The combination from the irregularly sampled data is replaced by the accumulation step, where the images are sequentially processed, and the image computation step.

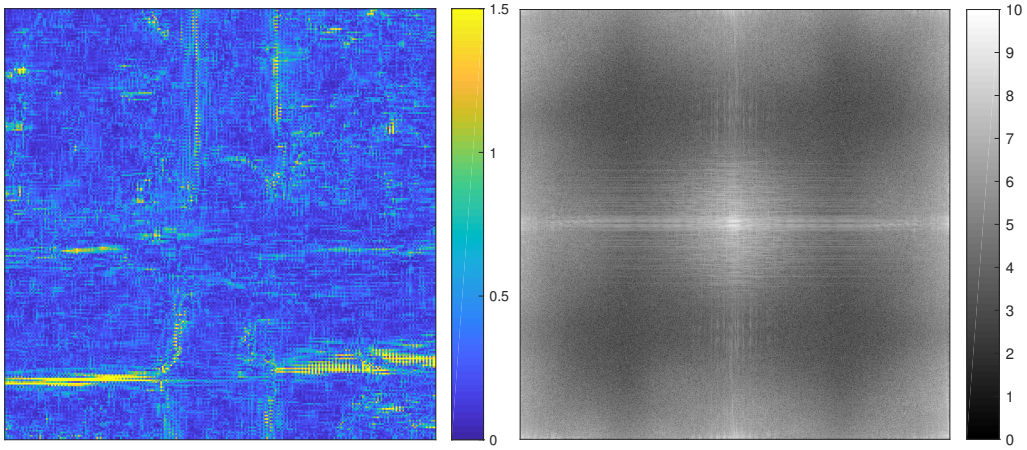
We show experimentally that, for a large number of RAW images, our image formation method provides, efficiently and with a low memory usage, a high-quality result. Contrarily to images fused from processed images (as in Chapter 10), the images formed from RAW images do not contain artifacts coming from the unknown image processing pipeline. Our method successfully performs super-resolution and the residual noise decreases as expected. We obtained results similar to those obtained by slower and memory greedy methods. The result of our method applied to 200 RAW images is shown in Figure 2.13.



(a) Burst denoising (RMSD = 0.17)

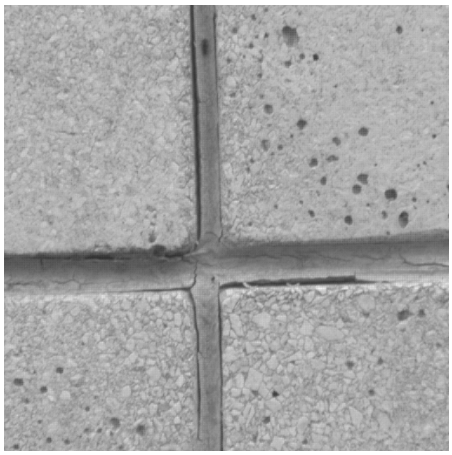


(b) ACT for $\lambda = 1$ (RMSD = 0.19)

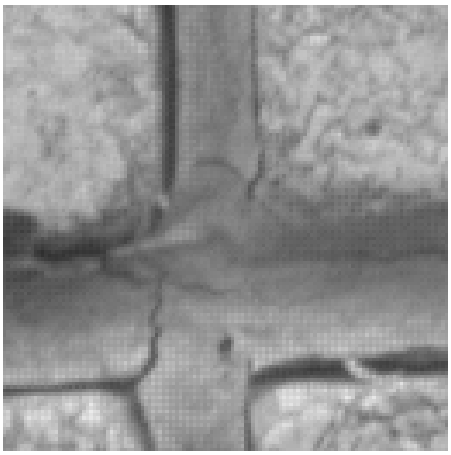


(c) ACT for $\lambda = 2$ (RMSD = 0.48)

Figure 2.12: Difference of our image fusion method with the ACT and the burst denoising methods for 200 real images. The results are really close and cannot be distinguished to the naked eye. Therefore the difference images are considered. They are not composed of noise. For $\lambda = 2$ the difference is more important in particular around the edges (where zipper structures can be seen).

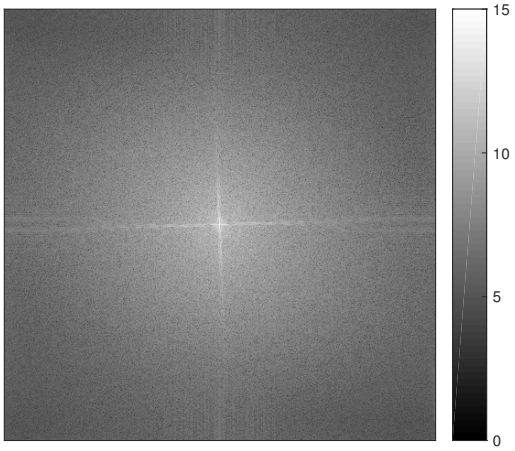


(a) Reference CFA image



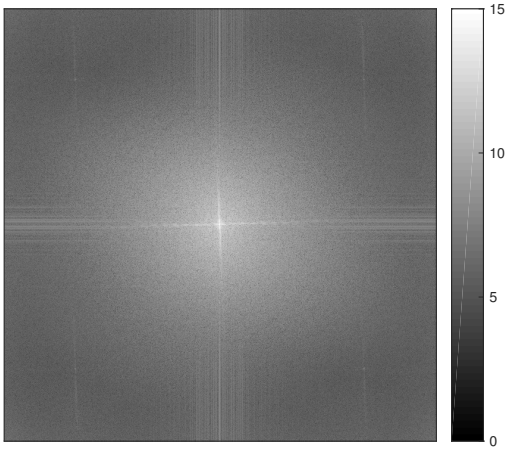
(b) Details of (a)

(c) Output image ($\lambda = 1$)



(d) Spectrum of (a)

(e) Output image ($\lambda = 1.5$)



(f) Spectrum of (c)

Figure 2.13: Result of our image formation method applied to 200 RAW images. The reference CFA image in (a) is of size 512×512 and corresponds to the preprocessed version of the reference RAW image. Details in a zone of size 128×128 are shown in (b). The mosaic structure is clearly visible in (b), in particular near the edges. The output color images in (c) and (e) are sharp, without artifacts and there is no apparent noise.

2.4 Contributions and Publications

List of contributions. The main contributions of this thesis, chapter-by-chapter, are the following:

- **Chapter 3**: Characterization of the trigonometric polynomial interpolators. Application to the geometric transformation of images, to up-sampling and down-sampling with DFT-based computations.
- **Chapter 4:** Taken from [15]. DFT-based algorithms for the filtering of real-valued images. Theoretical justification based on trigonometric polynomial interpolation. Theory and implementation provided at http://www.ipol.im/pub/art/2016/116/.
- **Chapter 5:** Taken from [14]. Complete description of B-spline interpolation (theory and practice). Two prefiltering algorithms with controlled precision. Implementation provided at http://www.ipol.im/pub/art/2018/221/.
- **Chapter 6:** Introduction of the consistency measurements. Fine-tuned interpolation methods with better consistency measurements.
- **Chapter 7:** Taken from [16]. Improvements of the inverse compositional algorithm in terms of accuracy and efficiency. Implementation and preprint provided at https://www.ipol.im/pub/pre/222/.
- **Chapter 8:** Two-step registration methods for mosaicked images.
- **Chapter 9:** Presentation of kernel regression. Introduction of the asymptotic equivalent filter for classical kernel regression. A fast and low memory image fusion method that is designed for a large sequence of images.
- **Chapter 10:** Experimental justification of our image fusion method. Determination of the best order and scale on synthetic data.
- **Chapter 11:** Fast and low memory image formation method that is designed for a large sequence of RAW images.

List of publications. The work in this thesis has led to the following publications:

- T. Briand and J. Vacher. "How to Apply a Filter Defined in the Frequency Domain by a Continuous Function". In Image Processing On Line (IPOL), vol. 6, pp. 183–211, 2016.
- T. Briand and P. Monasse. "Theory and Practice of Image B-Spline Interpolation". In Image Processing On Line (IPOL), vol. 8, pp. 99–141, 2018.
- T. Briand. "Low Memory Image Reconstruction Algorithm from RAW Images". In IEEE Image, Video, and Multidimensional Signal Processing (IVMSP), 2018.
- T. Briand, G. Facciolo, and J. Sánchez. "Improvements of the Inverse Compositional Algorithm for Parametric Motion Estimation". In Image Processing On Line (IPOL), submitted.

Part I

Interpolation

Chapter 3

Trigonometric Polynomial Interpolation of Images

Abstract

For 1D and 2D signals, the Shannon-Whittaker interpolation with periodic extension can be formulated as a trigonometric polynomial interpolation (TPI). In this chapter, we introduce the theory of TPI of images and some of its applications. First, the trigonometric polynomial interpolators of an image are characterized and it is shown that there is an ambiguity as soon as one size of the image is even. Three classical choices of interpolator for real-valued images are presented and cases where they coincide are pointed out. Then, TPI is applied to the geometric transformation of images, to up-sampling and to down-sampling. General results are expressed for any choice of interpolator but more details are given for the three proposed ones. It is proven that the well-known DFT-based computations have to be slightly adapted. The filtering using TPI is detailed separately in Chapter 4. The performances and the limits of TPI are discussed in Chapter 6.

Contents

3.1 Introduction

A fundamental example of interpolation formula is given by Shannon-Whittaker's sampling theory [108, 133]. Let $d \in \mathbb{N}$ (typically 1 or 2).

Definition 3.1 (Fourier transform)**.** *The continuous Fourier transform applied to $L^1(\mathbb{R}^d)$ and its inverse, denoted $\mathcal{F}$ and $\mathcal{F}^{-1}$, are defined by*

$$\forall u \in L^1(\mathbb{R}^d), \quad \forall \mathbf{y} \in \mathbb{R}^d, \quad \hat{u}(\mathbf{y}) = \mathcal{F}(u)(\mathbf{y}) = \int_{\mathbb{R}^d} u(\mathbf{x}) e^{-i\mathbf{x}.\mathbf{y}} d\mathbf{x} \tag{3.1}$$

$$\text{and} \quad \forall \mathbf{x} \in \mathbb{R}^d, \quad \mathcal{F}^{-1}(u)(\mathbf{x}) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} u(\mathbf{y}) e^{i\mathbf{x}.\mathbf{y}} d\mathbf{y}. \tag{3.2}$$

The Fourier transform extended to tempered distributions [107, 113] is still denoted $\mathcal{F}$.

Definition 3.2 (Band-limited distribution)**.** *A tempered distribution u is said to be band-limited if the support of its Fourier transform is bounded. It is said to be Nyquist band-limited if the support is contained in the Nyquist domain $[-\pi, \pi]^d$.*

Theorem 3.1 (Shannon-Whittaker sampling theorem [108, 133])**.** *Let $u \in L^1(\mathbb{R}^d, \mathbb{C})$ be a Nyquist band-limited function. Then, u is continuous and uniquely determined by its values at integer locations $\{u(\mathbf{k})\}_{\mathbf{k} \in \mathbb{Z}^d}$ since for $\mathbf{x} \in \mathbb{R}^d$,*

$$u(\mathbf{x}) = \sum_{\mathbf{k} \in \mathbb{Z}^d} f(\mathbf{k}) \text{sinc}(\mathbf{x} - \mathbf{k}) \tag{3.3}$$

where the cardinal sine function sinc *is the continuous function defined by*

$$\forall (x_1, \ldots, x_d) \in (\mathbb{R}^*)^d, \quad \text{sinc}(x_1, \ldots, x_d) = \prod_{i=1}^{d} \frac{\sin(\pi x_i)}{\pi x_i}. \tag{3.4}$$

Theorem 3.1, namely the Shannon-Whittaker sampling theorem, provides an equivalence between a band-limited function and its equidistant samples taken at a frequency that is superior or equal to the Nyquist rate. According to the Shannon-Whittaker interpolation formula (3.3), a band-limited signal can be written as the convolution between a Dirac comb weighted by its samples and the cardinal sine (or sinc) function. However this result cannot be used directly because, among others, it requires an infinite number of samples [109]. The finite signal first needs to be arbitrarily extended. Among all the possible extensions a classical solution is the periodic extension. For 1D and 2D signals, the Shannon-Whittaker interpolation with periodic extension can be formulated as a trigonometric polynomial interpolation (TPI), namely the Discrete Shannon interpolation [1].

More generally, TPI is theoretically interesting since the band-limited periodic distributions are exactly the trigonometric polynomials. But the main advantage is practical. It is well known that the discrete Fourier transform (DFT) of a signal, which can be computed efficiently using the fast Fourier transform (FFT) algorithm [24], is deeply linked to TPI (or sampling). Efficient image processing algorithms that rely on DFT-based computations can be used, for instance, to perform linear filtering, up-sampling and down-sampling or to shift signals. More generally, any geometric transformation can be applied efficiently to images using the nonequispaced Fast Fourier transform (NFFT) algorithm [94], which is based on oversampled FFT.

However it has to be noted that as soon as one of the dimensions is even there is an ambiguity in the definition of trigonometric polynomial interpolators (whose degree corresponds to the size of the image). The various interpolators differ from each other at the Nyquist frequency. Three particular interpolators are commonly used for real-valued images. The most common is

obtained directly from the DFT coefficients so that it is compatible with DFT-based computations but it may be complex-valued. Its real part is also a trigonometric polynomial interpolator and is usually used implicitly by taking the real part afterwards. The third interpolator is also real-valued and is given by the discrete Shannon interpolation. The choice of the interpolator depends on the context of application and may influence the results of algorithms relying on TPI (e.g. see [15]). Also, the compatibility with DFT-based computations has to be proven.

In this chapter we present the TPI of images and some of its applications. The trigonometric polynomial interpolators are characterized in terms of the DFT of the image. Three classical choices of interpolator for real-valued images are presented and cases where they coincide are pointed out. The theory is applied to the geometric transformations of images, and to up-sampling and down-sampling by integer factor. General results are expressed for any choice of interpolator but more details are given for the three proposed interpolators. This study proves that the well-known DFT-based computations have to be slightly adapted. The filtering using TPI is detailed separately in Chapter 4. The performances and the limits of TPI are discussed in Chapter 6.

This chapter is organized as follows: Section 3.2 presents the theory of TPI of images. This theory is applied to the geometric transformation of images in Section 3.3, and to the up-sampling and down-sampling in Section 3.4.

3.2 Trigonometric Polynomial Interpolation of Images

In this section we present the theory of TPI for images. First, useful definitions and notations are introduced in Section 3.2.1. Then, the trigonometric polynomial interpolators of an image are characterized in Section 3.2.2 and it is shown that there is an ambiguity as soon as one of the dimensions is even. In Section 3.2.3 three classical choices of interpolator for real-valued images are proposed and cases where they coincide are pointed out.

3.2.1 Definitions and Notations

In the following M and N denote two positive integers. First, discrete domains are defined as in

Definition 3.3 (Discrete domains)**.** *The discrete spatial domain $\Omega_{M,N}$ is defined by*

$$\Omega_{M,N} = \{0,\dots,M-1\} \times \{0,\dots,N-1\}. \tag{3.5}$$

The discrete Fourier domain $\hat{\Omega}_{M,N}$, associated to $\Omega_{M,N}$, is defined by $\hat{\Omega}_{M,N} = \hat{\Omega}_M \times \hat{\Omega}_N$ where for a positive integer L

$$\hat{\Omega}_L = \begin{cases} \{-\frac{L-1}{2},\dots,\frac{L-1}{2}\} & \text{if } L \text{ is odd,} \\ \{-\frac{L}{2},\dots,\frac{L}{2}-1\} & \text{if } L \text{ is even.} \end{cases} \tag{3.6}$$

The boundary $\Gamma_{M,N}$ of $\hat{\Omega}_{M,N}$ is defined by $\Gamma_{M,N} = \left(\Gamma_M \times \hat{\Omega}_N\right) \cup \left(\hat{\Omega}_M \times \Gamma_N\right)$ where for a positive integer L

$$\Gamma_L = \begin{cases} \emptyset & \text{if } L \text{ is odd,} \\ \{-\frac{L}{2}\} & \text{if } L \text{ is even.} \end{cases} \tag{3.7}$$

The symmetrized discrete Fourier domain $\hat{\Omega}^s_{M,N}$, associated to $\Omega_{M,N}$, is defined by $\hat{\Omega}^s_{M,N} = \hat{\Omega}^s_M \times \hat{\Omega}^s_N$ where for a positive integer L

$$\hat{\Omega}^s_L = \begin{cases} \hat{\Omega}_L & \text{if } L \text{ is odd} \\ \hat{\Omega}_L \cup \{\frac{L}{2}\} & \text{if } L \text{ is even.} \end{cases} \tag{3.8}$$

The boundary $\Gamma^s_{M,N}$ of $\hat{\Omega}^s_{M,N}$ is defined by $\Gamma^s_{M,N} = \hat{\Omega}^s_{M,N} \setminus \left(\hat{\Omega}_{M,N} \setminus \Gamma_{M,N}\right)$.

As an example, the discrete domains $\hat{\Omega}_{M,N}$, $\hat{\Omega}^s_{M,N}$, $\Gamma_{M,N}$ and $\Gamma^s_{M,N}$ for $M = N = 4$ are displayed in Figure 3.1. Note that assuming that M and N are odd numbers, $\hat{\Omega}_{M,N} = \hat{\Omega}^s_{M,N}$ and $\Gamma_{M,N} = \Gamma^s_{M,N} = \emptyset$.

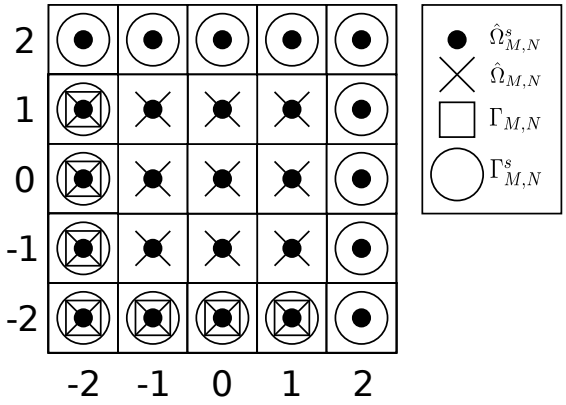


Figure 3.1: Discrete domains $\hat{\Omega}_{M,N}$, $\hat{\Omega}^s_{M,N}$, $\Gamma_{M,N}$ and $\Gamma^s_{M,N}$ for $M = N = 4$.

Definition 3.4 (Image). *An image (or digital image) $\underline{u}$ of size $M \times N$ is defined as a two-dimensional finite matrix of complex numbers $(\underline{u}_{k,l})_{(k,l)\in\Omega_{M,N}}$. The image is said to be real-valued when every number is real.*

In the following $\underline{u}$ denotes an image of size $M \times N$. Unless it is specified, $\underline{u}$ is not assumed to be real-valued. Trigonometric polynomial functions of the plane are defined as follows.

Definition 3.5 (Trigonometric polynomial). *A function $P : \mathbb{R}^2 \to \mathbb{C}$ is said to be a trigonometric polynomial of degree $\leq M \times N$ if there exists $c \in \mathbb{C}^{\hat{\Omega}^s_{M,N}}$, called coefficients of P, s.t.*

$$\forall (x,y) \in \mathbb{R}^2, \quad P(x,y) = \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} c_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})}. \tag{3.9}$$

The set of trigonometric polynomials of degree $\leq M \times N$, denoted $\mathcal{P}_{M,N}$, is a subset of the set of $(M,N)-$periodic functions. Denote $\mathcal{R}$ the real part operator and let $P \in \mathcal{P}_{M,N}$ then $\mathcal{R}(P) \in \mathcal{P}_{M,N}$.

Note that another possible convention [100, p. 88] for trigonometric polynomials of degree $M \times N$ is to consider $(2\pi, 2\pi)-$periodic functions of the form

$$Q(x,y) = \sum_{(m,n)\in\hat{\Omega}_{2M+1,2N+1}} c_{m,n} e^{i(mx+ny)} = \sum_{m=-M}^{M} \sum_{n=-N}^{N} c_{m,n} e^{i(mx+ny)}. \tag{3.10}$$

It is directly linked to our convention (see Definition 3.5) since $(x,y) \in \mathbb{R}^2 \mapsto Q(\frac{2\pi}{M}x, \frac{2\pi}{N}y)$ is a trigonometric polynomial of degree $\leq (2M+1) \times (2N+1)$. Note that the corresponding degree necessarily involves odd numbers. Our convention has the advantage of being compatible with the size of images and with the Discrete Fourier transform (DFT), which is defined as in

Definition 3.6 (Discrete Fourier transform). *The discrete Fourier transform (DFT) of $\underline{u} \in \mathbb{C}^{\Omega_{M,N}}$ is denoted $\mathcal{F}_{M,N}(\underline{u}) \in \mathbb{C}^{\hat{\Omega}_{M,N}}$ and is defined by*

$$\forall (m,n) \in \hat{\Omega}_{M,N}, \quad \mathcal{F}_{M,N}(\underline{u})_{m,n} = \frac{1}{MN} \sum_{(k,l)\in\Omega_{M,N}} \underline{u}_{k,l} e^{-2\pi i(k\frac{m}{M}+l\frac{n}{N})}. \tag{3.11}$$

The inverse discrete Fourier transform (iDFT) of $\underline{v} \in \mathbb{C}^{\hat{\Omega}_{M,N}}$ *is denoted* $\mathcal{F}_{M,N}^{-1}(\underline{v}) \in \mathbb{C}^{\Omega_{M,N}}$ *and is defined by*

$$\forall (k,l) \in \Omega_{M,N}, \quad \mathcal{F}_{M,N}^{-1}(\underline{v})_{k,l} = \sum_{(m,n)\in\hat{\Omega}_{M,N}} \underline{v}_{m,n} e^{2\pi i (m\frac{k}{M} + n\frac{l}{N})}. \tag{3.12}$$

Note that another classical convention is obtained by moving the normalization factor $\frac{1}{MN}$ from the DFT to the iDFT. The resulting unnormalized DFT is used in this document to show the spectrum of images (for visualization purposes). The DFT and iDFT of size $M \times N$ can be computed efficiently in $O(MN\log(MN))$ floating point operations thanks to the fast Fourier transform (FFT) algorithm [24].

3.2.2 Trigonometric Polynomial Interpolators

The trigonometric polynomial interpolators of an image $\underline{u}$ are defined and characterized as follows.

Definition 3.7 (Trigonometric polynomial interpolator)**.** *A trigonometric polynomial interpolator of* $\underline{u}$ *is a trigonometric polynomial* $P \in \mathcal{P}_{M,N}$ *verifying the interpolation condition*

$$\forall (k,l) \in \Omega_{M,N}, \quad P(k,l) = \underline{u}_{k,l}. \tag{3.13}$$

By definition a trigonometric polynomial interpolator has a degree smaller than the size of the interpolated image. Note that if the degree is not controlled, the interpolation condition in (3.13) is verified by infinitely many trigonometric polynomials.

Proposition 3.1. *Let* $P \in \mathcal{P}_{M,N}$*. P is a trigonometric polynomial interpolator of* $\underline{u}$ *if and only if its coefficients* $c \in \mathbb{C}^{\hat{\Omega}^s_{M,N}}$ *verify*

$$\begin{cases} c_{m,n} = \mathcal{F}_{M,N}(\underline{u})_{m,n} & \text{for } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ c_{\frac{M}{2},n} + c_{-\frac{M}{2},n} = \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} & \text{for } |n| < \frac{N}{2} \\ c_{m,\frac{N}{2}} + c_{m,-\frac{N}{2}} = \mathcal{F}_{M,N}(\underline{u})_{m,-\frac{N}{2}} & \text{for } |m| < \frac{M}{2} \\ c_{\frac{M}{2},\frac{N}{2}} + c_{\frac{M}{2},-\frac{N}{2}} + c_{-\frac{M}{2},\frac{N}{2}} + c_{-\frac{M}{2},-\frac{N}{2}} = \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}}. \end{cases} \tag{3.14}$$

Proof. Let $P \in \mathcal{P}_{M,N}$ be a trigonometric polynomial interpolator of $\underline{u}$. Let $(k,l) \in \Omega_{M,N}$. We have

$$\underline{u}_{k,l} = P(k,l) = \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} c_{m,n} e^{2i\pi(k\frac{m}{M} + l\frac{n}{N})}. \tag{3.15}$$

Noting that for any $(L,j) \in \mathbb{N}^* \times \mathbb{N}$ we have $e^{2i\pi j\frac{L}{2}\frac{1}{L}} = e^{2i\pi j\frac{-L}{2}\frac{1}{L}} = (-1)^j$, we can write

$$\underline{u}_{k,l} = \sum_{(m,n)\in\hat{\Omega}_{M,N}} d_{m,n} e^{2i\pi(k\frac{m}{M} + l\frac{n}{N})} \tag{3.16}$$

where $d = (d_{m,n}) \in \mathbb{C}^{\hat{\Omega}_{M,N}}$ verify for $(m,n) \in \hat{\Omega}_{M,N}$,

$$d_{m,n} = \begin{cases} c_{m,n} & \text{if } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ c_{\frac{M}{2},n} + c_{-\frac{M}{2},n} & \text{if } m = -\frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ c_{m,\frac{N}{2}} + c_{m,-\frac{N}{2}} & \text{if } |m| < \frac{M}{2} \text{ and } n = -\frac{N}{2} \\ c_{\frac{M}{2},\frac{N}{2}} + c_{\frac{M}{2},-\frac{N}{2}} + c_{-\frac{M}{2},\frac{N}{2}} + c_{-\frac{M}{2},-\frac{N}{2}} & \text{if } m = -\frac{M}{2} \text{ and } n = -\frac{N}{2}. \end{cases} \tag{3.17}$$

Finally by definition of the DFT and iDFT we have $\mathcal{F}_{M,N}^{-1}(d) = \underline{u} = \mathcal{F}_{M,N}^{-1}(\mathcal{F}_{M,N}(\underline{u}))$ and thus $d = \mathcal{F}_{M,N}(\underline{u})$.

Conversely, if $P \in \mathcal{P}_{M,N}$ is defined by the coefficients c verifying (3.14) then it is an interpolator of $\underline{u}$. □

The existence of trigonometric polynomial interpolators and their characterization is given by Proposition 3.1. They can easily be obtained from the DFT coefficients of the image so that the TPI is commonly called DFT interpolation.

Corollary 3.1. *One of the two following cases occur.*

- *If M and N are odd, then there is a unique trigonometric polynomial interpolator noted $P_{\underline{u}}$ whose coefficients are $\mathcal{F}_{M,N}(\underline{u})$.*
- *If M or N is even, then there are infinitely many trigonometric polynomial interpolators. Their coefficients only differ from each other on the boundary $\Gamma^s_{M,N}$.*

Proof. It is a direct consequence of Proposition 3.1. □

When for instance M is even, one can easily check that the interpolation property is kept when adding any multiple of the trigonometric polynomial of degree $M \times N$ defined by $(x,y) \in \mathbb{R}^2 \mapsto \sin(\pi x)$. As pointed out in [1] and [15], the lack of symmetry for even sizes introduces an ambiguity in the choice of the trigonometric polynomial interpolator.

This can also be seen with a dimensional approach. The space of trigonometric polynomials of degree $\leq M \times N$ has for dimension $\#\hat{\Omega}^s_{M,N}$, which is greater or equal to $MN = \#\Omega_{M,N}$. The linear map $P \in \mathcal{P}_{M,N} \mapsto (P(k,l))_{(k,l)\in\Omega_{M,N}}$ is one-to-one (and bijective) if and only if the kernel dimension $\#\left(\Gamma^s_{M,N} \setminus \Gamma_{M,N}\right)$ is 0. Finally, it is one-to-one if and only if M and N are odd numbers. Actually $\#\hat{\Omega}^s_{M,N}$ is the number of coefficients $c_{m,n}$ used to represent trigonometric polynomials of degree $M \times N$ and is a product of odd numbers. If M is even and N is odd then the kernel dimension is N. If both M and N are even then the kernel dimension is $M+N+1$.

The dimensional approach also shows that $M \times N$ is the smallest degree (along both dimensions) ensuring the existence of a trigonometric polynomial verifying the interpolation condition (3.13).

3.2.3 Trigonometric Polynomial Interpolators of a Real-valued Image

Assume that $\underline{u}$ is real-valued. Three particular trigonometric polynomial interpolators of $\underline{u}$ are proposed and the cases where they coincide are pointed out.

Definition 3.8 (Trigonometric polynomial interpolator in complex convention)**.** *The trigonometric polynomial interpolator of $\underline{u}$ in complex convention is the trigonometric polynomial $P^{(c)}_{\underline{u}} \in \mathcal{P}_{M,N}$ defined by*

$$\forall (x,y) \in \mathbb{R}^2, \quad P^{(c)}_{\underline{u}}(x,y) = \sum_{(m,n)\in\hat{\Omega}_{M,N}} \mathcal{F}_{M,N}(\underline{u})_{m,n} e^{2i\pi\left(x\frac{m}{M}+y\frac{n}{N}\right)}. \tag{3.18}$$

The trigonometric polynomial interpolator in complex convention is the natural and simplest way to define a trigonometric polynomial interpolator since the coefficients are directly expressed in terms of the DFT of the image. Using Proposition 3.1 it is proven to be an interpolator (even for a complex-valued image) since it corresponds to the particular case where the coefficients are null in $\Gamma^s_{M,N} \setminus \Gamma_{M,N}$. However because of the DFT asymmetry it may be complex-valued. Therefore two other interpolators, which are guaranteed to be real-valued, are built.

Proposition 3.2. *Assume $\underline{u}$ is real-valued. Then, $\mathcal{R}(P_{\underline{u}}^{(c)}) \in \mathcal{P}_{M,N}$ is a trigonometric polynomial interpolator of $\underline{u}$ whose coefficients $c \in \hat{\Omega}_{M,N}^s$ verify for $(m,n) \in \hat{\Omega}_{M,N}^s$,*

$$c_{m,n} = \begin{cases} \mathcal{F}_{M,N}(\underline{u})_{m,n} & \text{if } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} & \text{if } |m| = \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{m,-\frac{N}{2}} & \text{if } |m| < \frac{M}{2} \text{ and } |n| = \frac{N}{2} \\ \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}} & \text{if } (m,n) \in \left\{\pm(\frac{M}{2},\frac{N}{2})\right\} \\ 0 & \text{if } (m,n) \in \left\{\pm(-\frac{M}{2},\frac{N}{2})\right\}. \end{cases} \tag{3.19}$$

Proof. This is a consequence of the Hermitian symmetry of the DFT of real-valued images. Indeed, extending the DFT by $(M,N)-$periodicity we easily get $\mathcal{F}_{M,N}(\underline{u})_{m,n} = \overline{\mathcal{F}_{M,N}(\underline{u})_{-m,-n}}$ for all $(m,n) \in \mathbb{Z}^2$. Let $(x,y) \in \mathbb{R}^2$. By grouping the terms of the sum in Equation (3.18) we can deduce that the complex contribution of $P_{\underline{u}}^{(c)}$ comes from the terms with indices in $\Gamma_{M,N}$,

$$\begin{aligned} P_{\underline{u}}^{(c)}(x,y) &= \underbrace{\mathcal{F}_{M,N}(\underline{u})_{0,0} + \sum_{\substack{0\le m<\frac{M}{2},\\ 0\le n<\frac{N}{2}\\ (m,n)\neq(0,0)}} 2\mathcal{R}\left(\mathcal{F}_{M,N}(\underline{u})_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})}\right)}_{\in\mathbb{R}} \\ &\quad + \sum_{(m,n)\in\Gamma_{M,N}} \mathcal{F}_{M,N}(\underline{u})_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})}. \end{aligned} \tag{3.20}$$

Case 1: Assume M and N are odd. Then $\Gamma_{M,N} = \emptyset$. This implies that $P_{\underline{u}}^{(c)}$ is real-valued and we have the result.

Case 2: Assume that M is even and N is odd. Then $\Gamma_{M,N} = \left\{-\frac{M}{2}\right\} \times \left\{-\frac{N-1}{2},\ldots,\frac{N-1}{2}\right\}$. Using the relation $\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} = \overline{\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-n}}$ we have

$$\sum_{(m,n)\in\Gamma_{M,N}} \mathcal{F}_{M,N}(\underline{u})_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})} = e^{-i\pi x} \underbrace{\left(\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},0} + \sum_{n=1}^{\frac{N-1}{2}} 2\mathcal{R}\left(\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} e^{2i\pi\frac{ny}{N}}\right)\right)}_{\in\mathbb{R}}. \tag{3.21}$$

Finally as $\mathcal{R}(e^{-i\pi x}) = \frac{1}{2}(e^{i\pi x} + e^{-i\pi x}) = \frac{1}{2}(e^{2i\pi\frac{M}{2}\frac{x}{M}} + e^{2i\pi(-\frac{M}{2})\frac{x}{M}})$, we have

$$\mathcal{R}\left(\sum_{(m,n)\in\Gamma_{M,N}} \mathcal{F}_{M,N}(\underline{u})_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})}\right) = \sum_{(m,n)\in\Gamma_{M,N}^s} \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})} \tag{3.22}$$

and the result is obtained by identification. Similarly, we deal with the case M odd and N even by switching the coordinates.

Case 3: Assume that M and M are even. Then we have the partition

$$\Gamma_{M,N} = \left\{-\frac{M}{2}\right\} \times \left\{-\frac{N}{2}+1,\ldots,\frac{N}{2}-1\right\} \bigsqcup \left\{-\frac{M}{2}+1,\ldots,\frac{M}{2}-1\right\} \times \left\{-\frac{N}{2}\right\} \bigsqcup \left\{-\frac{M}{2}\right\} \times \left\{-\frac{N}{2}\right\} \tag{3.23}$$

so that the sum $\sum_{(m,n)\in\Gamma_{M,N}} \mathcal{F}_{M,N}(\underline{u})_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})}$ can be decomposed into three sums. The two first components can be handled as in case 2 (even though the sum indices are slighlty different).

The third component corresponds to the index $\left(-\frac{M}{2},-\frac{N}{2}\right)$ for which we have

$$\mathcal{R}\left(\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}}e^{-i\pi(x+y)}\right) = \underbrace{\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}}}_{\in\mathbb{R}}\mathcal{R}(e^{-i\pi(x+y)}) \tag{3.24}$$

$$= \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}}(e^{i\pi(x+y)}+e^{-i\pi(x+y)}). \tag{3.25}$$

Finally the result is obtained by identification. □

As stated in Proposition 3.2, $\mathcal{R}(P_{\underline{u}}^{(c)})$ is a real-valued trigonometric polynomial interpolator of $\underline{u}$ which can be easily obtained from the complex convention. However when M and N are even its coefficients show an asymmetry in the highest frequencies. Therefore the following alternative may be prefered.

Definition 3.9 (Trigonometric polynomial interpolator in real convention)**.** *Assume $\underline{u}$ is real-valued. The trigonometric polynomial interpolator of $\underline{u}$ in real convention is defined as the trigonometric polynomial $P_{\underline{u}}^{(r)} \in \mathcal{P}_{M,N}$ whose coefficients $c \in \hat{\Omega}_{M,N}^{s}$ verify for $(m,n) \in \hat{\Omega}_{M,N}^{s}$,*

$$c_{m,n} = \begin{cases} \mathcal{F}_{M,N}(\underline{u})_{m,n} & \text{if } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} & \text{if } |m| = \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{m,-\frac{N}{2}} & \text{if } |m| < \frac{M}{2} \text{ and } |n| = \frac{N}{2} \\ \frac{1}{4}\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}} & \text{if } |m| = \frac{M}{2} \text{ and } |n| = \frac{N}{2}. \end{cases} \tag{3.26}$$

Proposition 3.3. *Assume $\underline{u}$ is real-valued. Then $P_{\underline{u}}^{(r)}$ is a real-valued interpolating function of $\underline{u}$ since for $(x,y) \in \mathbb{R}^2$,*

$$P_{\underline{u}}^{(r)}(x,y) = \begin{cases} \mathcal{R}(P_{\underline{u}}^{(c)})(x,y) + \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}}\sin(\pi x)\sin(\pi y) & \text{if } M \text{ and } N \text{ are even} \\ \mathcal{R}(P_{\underline{u}}^{(c)})(x,y) & \text{otherwise.} \end{cases} \tag{3.27}$$

Proof. Noticing that $\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}} \in \mathbb{R}$ and using the relation

$$\mathcal{R}(e^{-i\pi(x+y)}) + \sin(\pi x)\sin(\pi y) = \frac{1}{4}\left(e^{i\pi(x+y)}+e^{-i\pi(x+y)}+e^{i\pi(x-y)}+e^{i\pi(-x+y)}\right), \tag{3.28}$$

it is directly obtained from the definition of $P_{\underline{u}}^{(r)}$ and Proposition 3.2. □

Proposition 3.3 states that $P_{\underline{u}}^{(r)}$ is another real-valued trigonometric polynomial interpolator and makes the link with $\mathcal{R}(P_{\underline{u}}^{(c)})$. When M and N are even $P_{\underline{u}}^{(r)}$ is usually preferred to $\mathcal{R}(P_{\underline{u}}^{(c)})$ because it has the same DFT coefficients at the four corners of $\hat{\Omega}_{M,N}^{s}$. In [1] the trigonometric polynomial interpolator in real convention is called discrete Shannon interpolator because it corresponds to the Shannon-Whittaker interpolator with periodic boundary extension.

The particular conditions under which the proposed trigonometric polynomial interpolators are equal are presented in Proposition 3.4.

Proposition 3.4. *Assume $\underline{u}$ is real-valued. The cases of equality of the three proposed trigonometric polynomial interpolators are:*

1. $P_{\underline{u}}^{(c)} = \mathcal{R}(P_{\underline{u}}^{(c)}) = P_{\underline{u}}^{(r)}$ *if and only if*

$$\{(m,n) \in \Gamma_{M,N} \,|\, \mathcal{F}_{M,N}(\underline{u})_{m,n} \neq 0\} = \emptyset. \tag{3.29}$$

In particular, it is the case when M and N are odd since there is a unique trigonometric polynomial interpolator and $\Gamma_{M,N} = \emptyset$.

2. *Assume that M or N is odd. Then, $\mathcal{R}(P_{\underline{u}}^{(c)}) = P_{\underline{u}}^{(r)}$.*

3. *Assume that M and N are even. Then $\mathcal{R}(P_{\underline{u}}^{(c)}) = P_{\underline{u}}$ if and only if $\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}} = 0$.*

Proof. It is a direct consequence of Proposition 3.2 and Proposition 3.3. □

3.3 Application to Geometric Transformation of Images

In this section we apply TPI to geometric transformations of images. Let $P \in \mathcal{P}_{M,N}$ be a trigonometric polynomial interpolator of $\underline{u}$ with coefficients $c \in \hat{\Omega}^s_{M,N}$ and $\varphi \in \sigma(\mathbb{R}^2)$ be a geometric transformation. The transformation of $\underline{u}$ by φ using P is noted $\underline{u}_{P,\varphi}$ and is defined by

$$\forall (k,l) \in \Omega_{M,N}, (\underline{u}_{P,\varphi})_{k,l} = P(\varphi^{-1}(k,l)). \tag{3.30}$$

First the case of translations is considered in Section 3.3.1. It is shown that a translated image can be computed efficiently using DFT-based computations. In particular the results for the three classical trigonometric interpolators introduced in Section 3.2.3 are detailed. The invertibility of the translation operation is also studied. Then, an efficient algorithm for computing any transformation of an image is proposed in Section 3.3.2.

3.3.1 Translation

Let $(\alpha,\beta) \in \mathbb{R}^2$ be a shift parameter. The translation by (α,β) is defined by $\varphi : (x,y) \in \mathbb{R}^2 \mapsto (x+\alpha, y+\beta)$. For simplicity we use slightly different notation for $\underline{u}_{P,\varphi}$ as proposed in

Definition 3.10. *The translated image of $\underline{u}$ with shift (α,β) using the interpolator P is noted $\underline{u}_{(P,\alpha,\beta)} \in \mathbb{C}^{\Omega_{M,N}}$. It is defined by*

$$\forall (k,l) \in \Omega_{M,N}, \quad \left(\underline{u}_{(P,\alpha,\beta)}\right)_{k,l} = P(k-\alpha, l-\beta). \tag{3.31}$$

Proposition 3.5. *The DFT coefficients of $\underline{u}_{(P,\alpha,\beta)}$ verify*

$$\begin{cases} \mathcal{F}_{M,N}(\underline{u}_{(P,\alpha,\beta)})_{m,n} = c_{m,n} e^{-2i\pi(\alpha\frac{m}{M}+\beta\frac{n}{N})} & \text{for } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \mathcal{F}_{M,N}(\underline{u}_{(P,\alpha,\beta)})_{-\frac{M}{2},n} = \left(c_{-\frac{M}{2},n} e^{i\pi\alpha} + c_{\frac{M}{2},n} e^{-i\pi\alpha}\right) e^{-2i\pi\beta\frac{n}{N}} & \text{for } |n| < \frac{N}{2} \\ \mathcal{F}_{M,N}(\underline{u}_{(P,\alpha,\beta)})_{m,-\frac{N}{2}} = \left(c_{m,-\frac{N}{2}} e^{i\pi\beta} + c_{m,\frac{N}{2}} e^{-i\pi\beta}\right) e^{-2i\pi\alpha\frac{m}{M}} & \text{for } |m| < \frac{M}{2} \\ \mathcal{F}_{M,N}(\underline{u}_{(P,\alpha,\beta)})_{-\frac{M}{2},-\frac{N}{2}} = c_{-\frac{M}{2},-\frac{N}{2}} e^{i\pi(\alpha+\beta)} + c_{\frac{M}{2},\frac{N}{2}} e^{-i\pi(\alpha+\beta)} \\ \qquad + c_{-\frac{M}{2},\frac{N}{2}} e^{i\pi(\alpha-\beta)} + c_{\frac{M}{2},-\frac{N}{2}} e^{-i\pi(\alpha-\beta)}. \end{cases} \tag{3.32}$$

Proof. For $(k,l) \in \Omega_{M,N}$ we have

$$\left(\underline{u}_{(P,\alpha,\beta)}\right)_{k,l} = P(k-\alpha, l-\beta) \tag{3.33}$$

$$= \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} c_{m,n} e^{-2i\pi(\alpha\frac{m}{M}+\beta\frac{n}{N})} e^{2i\pi(\frac{mk}{M}+\frac{nl}{N})}. \tag{3.34}$$

The result is obtained using the same reasoning as in the proof of Proposition 3.1 except that the $c_{m,n}$ are multiplied by $e^{-2i\pi(\alpha\frac{m}{M}+\beta\frac{n}{N})}$. □

As stated in Proposition 3.5 the DFT coefficients of the translated image $\underline{u}_{(P,\alpha,\beta)}$ can be easily computed from the trigonometric polynomial coefficients c by a phase shift. Therefore the translation using TPI is commonly called the DFT translation. The coefficients in $\Gamma_{M,N}$ have a slightly different expression that depends on the choice of the interpolator. In particular, for the three classical trigonometric polynomial interpolators the DFT coefficients of the translated images are given by

Proposition 3.6. *For all* $(m,n) \in \hat{\Omega}_{M,N}$,

$$\mathcal{F}_{M,N}(\underline{u}_{(P_{\underline{u}}^{(c)},\alpha,\beta)})_{m,n} = \mathcal{F}_{M,N}(\underline{u})_{m,n} e^{-2i\pi(\alpha\frac{m}{M}+\beta\frac{n}{N})}. \tag{3.35}$$

If in addition $\underline{u}$ *is real-valued, then*

$$\begin{cases} \mathcal{F}_{M,N}(\underline{u}_{(\mathcal{R}(P_{\underline{u}}^{(c)}),\alpha,\beta)})_{m,n} = \mathcal{F}_{M,N}(\underline{u})_{m,n} e^{-2i\pi(\alpha\frac{m}{M}+\beta\frac{n}{N})} & \text{for } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \mathcal{F}_{M,N}(\underline{u}_{(\mathcal{R}(P_{\underline{u}}^{(c)}),\alpha,\beta)})_{-\frac{M}{2},n} = \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} \cos(\pi\alpha) e^{-2i\pi\beta\frac{n}{N}} & \text{for } |n| < \frac{N}{2} \\ \mathcal{F}_{M,N}(\underline{u}_{(\mathcal{R}(P_{\underline{u}}^{(c)}),\alpha,\beta)})_{m,-\frac{N}{2}} = \mathcal{F}_{M,N}(\underline{u})_{m,-\frac{N}{2}} \cos(\pi\beta) e^{-2i\pi\alpha\frac{m}{M}} & \text{for } |m| < \frac{M}{2} \\ \mathcal{F}_{M,N}(\underline{u}_{(\mathcal{R}(P_{\underline{u}}^{(c)}),\alpha,\beta)})_{-\frac{M}{2},-\frac{N}{2}} = \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}} \cos(\pi(\alpha+\beta)) & \end{cases} \tag{3.36}$$

and

$$\begin{cases} \mathcal{F}_{M,N}(\underline{u}_{(P_{\underline{u}}^{(r)},\alpha,\beta)})_{m,n} = \mathcal{F}_{M,N}(\underline{u})_{m,n} e^{-2i\pi(\alpha\frac{m}{M}+\beta\frac{n}{N})} & \text{for } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \mathcal{F}_{M,N}(\underline{u}_{(P_{\underline{u}}^{(r)},\alpha,\beta)})_{-\frac{M}{2},n} = \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} \cos(\pi\alpha) e^{-2i\pi\beta\frac{n}{N}} & \text{for } |n| < \frac{N}{2} \\ \mathcal{F}_{M,N}(\underline{u}_{(P_{\underline{u}}^{(r)},\alpha,\beta)})_{m,-\frac{N}{2}} = \mathcal{F}_{M,N}(\underline{u})_{m,-\frac{N}{2}} \cos(\pi\beta) e^{-2i\pi\alpha\frac{m}{M}} & \text{for } |m| < \frac{M}{2} \\ \mathcal{F}_{M,N}(\underline{u}_{(P_{\underline{u}}^{(r)},\alpha,\beta)})_{-\frac{M}{2},-\frac{N}{2}} = & \\ \quad \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}} \frac{1}{2}(\cos(\pi(\alpha+\beta)) + \cos(\pi(\alpha-\beta))). & \end{cases} \tag{3.37}$$

Proof. It is a direct consequence of Proposition 3.5. The coefficients of the trigonometric polynomial interpolators are given in Definition 3.8, Proposition 3.2 and Definition 3.9. □

Using Proposition 3.6 it is possible to determine if the DFT translation can be inverted by applying the DFT translation with opposite shift. It is the case for the trigonometric polynomial interpolator in complex convention as stated in

Proposition 3.7. *Set* $\underline{v} = \underline{u}_{(P_{\underline{u}}^{(c)},\alpha,\beta)}$. *Then,*

$$\underline{v}_{(P_{\underline{v}}^{(c)},-\alpha,-\beta)} = \underline{u}. \tag{3.38}$$

Proof. It is obtained using (3.35) successively for (α,β) and $-(\alpha,\beta)$. □

On the contrary for the two classical real-valued trigonometric polynomial interpolators it is not automatically the case. The DFT coefficients of the image obtained after the two opposite translations may differ from the original ones on the Fourier boundary. More precisely, the DFT coefficients are given by

Proposition 3.8. *Assume* $\underline{u}$ *is real-valued. Set* $\underline{v} = \underline{u}_{(\mathcal{R}(P_{\underline{u}}^{(c)}),\alpha,\beta)}$. *Then,* $\underline{v}_{(\mathcal{R}(P_{\underline{v}}^{(c)}),-\alpha,-\beta)}$ *verifies*

$$\begin{cases} \mathcal{F}_{M,N}(\underline{v}_{(\mathcal{R}(P_{\underline{v}}^{(c)}),-\alpha,-\beta)})_{m,n} = \mathcal{F}_{M,N}(\underline{u})_{m,n} & \text{for } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \mathcal{F}_{M,N}(\underline{v}_{(\mathcal{R}(P_{\underline{v}}^{(c)}),-\alpha,-\beta)})_{-\frac{M}{2},n} = \cos(\pi\alpha)^2 \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} & \text{for } |n| < \frac{N}{2} \\ \mathcal{F}_{M,N}(\underline{v}_{(\mathcal{R}(P_{\underline{v}}^{(c)}),-\alpha,-\beta)})_{m,-\frac{N}{2}} = \cos(\pi\beta)^2 \mathcal{F}_{M,N}(\underline{u})_{m,-\frac{N}{2}} & \text{for } |m| < \frac{M}{2} \\ \mathcal{F}_{M,N}(\underline{v}_{(\mathcal{R}(P_{\underline{v}}^{(c)}),-\alpha,-\beta)})_{-\frac{M}{2},-\frac{N}{2}} = \cos(\pi(\alpha+\beta))^2 \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}}. & \end{cases} \tag{3.39}$$

Set $\underline{w} = \underline{u}_{(P_{\underline{u}}^{(r)},\alpha,\beta)}$. *Then,* $\underline{w}_{(P_{\underline{w}}^{(r)},-\alpha,-\beta)}$ *verifies*

$$\begin{cases} \mathcal{F}_{M,N}(\underline{w}_{((P_{\underline{w}}^{(r)},-\alpha,-\beta)})_{m,n} = \mathcal{F}_{M,N}(\underline{u})_{m,n} & \text{for } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \mathcal{F}_{M,N}(\underline{w}_{((P_{\underline{w}}^{(r)},-\alpha,-\beta)})_{-\frac{M}{2},n} = \cos(\pi\alpha)^2 \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} & \text{for } |n| < \frac{N}{2} \\ \mathcal{F}_{M,N}(\underline{w}_{((P_{\underline{w}}^{(r)},-\alpha,-\beta)})_{m,-\frac{N}{2}} = \cos(\pi\beta)^2 \mathcal{F}_{M,N}(\underline{u})_{m,-\frac{N}{2}} & \text{for } |m| < \frac{M}{2} \\ \mathcal{F}_{M,N}(\underline{w}_{((P_{\underline{w}}^{(r)},-\alpha,-\beta)})_{-\frac{M}{2},-\frac{N}{2}} = \\ \quad \frac{1}{4}\left(\cos\left(\pi(\alpha+\beta)\right) + \cos\left(\pi(\alpha-\beta)\right)\right)^2 \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}}. \end{cases} \tag{3.40}$$

Proof. It is a direct consequence of (3.36) and (3.37) applied successively to (α,β) and $-(\alpha,\beta)$. □

The particular cases where a DFT translation is inverted by the opposite DFT translation can be summarized as in

Proposition 3.9. *Assume* $\underline{u}$ *is real-valued. Let* $P \in \{P_{\underline{u}}^{(c)}, \mathcal{R}(P_{\underline{u}}^{(c)}), P_{\underline{u}}^{(r)}\}$. *Then, the following propositions are equivalent:*

(1) *The translation of* $\underline{u}$ *with shift* (α,β) *using the interpolator* P *can be inverted by the opposite translation.*

(2) *For all* $(m,n) \in \hat{\Omega}_{M,N}$,

$$\left|\mathcal{F}_{M,N}(\underline{u}_{(P,\alpha,\beta)})\right|_{m,n} = \left|\mathcal{F}_{M,N}(\underline{u})\right|_{m,n}. \tag{3.41}$$

(3) *For all* $(m,n) \in \Gamma_{M,N}$,

$$\left|\mathcal{F}_{M,N}(\underline{u}_{(P,\alpha,\beta)})\right|_{m,n} = \left|\mathcal{F}_{M,N}(\underline{u})\right|_{m,n}. \tag{3.42}$$

In particular, (1) holds as soon as:

1. $P = P_{\underline{u}}^{(c)}$.
2. $(\alpha,\beta) \in \mathbb{Z}^2$.
3. $P_{\underline{u}}^{(c)} = \mathcal{R}(P_{\underline{u}}^{(c)}) = P_{\underline{u}}^{(r)}$ *i.e.*

$$\{(m,n) \in \Gamma_{M,N} \,|\, \mathcal{F}_{M,N}(\underline{u})_{m,n} \neq 0\} = \emptyset. \tag{3.43}$$

In particular it is the case when M *and* N *are odd.*

4. $\beta \in \mathbb{Z}$ *and*

$$\left\{n \in \Gamma_N \,|\, \mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} \neq 0\right\} = \emptyset. \tag{3.44}$$

In particular it is the case when N *is odd.*

5. $\alpha \in \mathbb{Z}$ *and*

$$\left\{m \in \Gamma_M \,|\, \mathcal{F}_{M,N}(\underline{u})_{m,-\frac{N}{2}} \neq 0\right\} = \emptyset. \tag{3.45}$$

In particular it is the case when M *is odd.*

Proof. Set $\underline{v} = \underline{u}_{(P,\alpha,\beta)}$ and $\underline{w} = \underline{v}_{(P',-\alpha,-\beta)}$ where P' is the corresponding trigonometric polynomial interpolator of $\underline{v}$. Using Proposition 3.6 we can write

$$\mathcal{F}_{M,N}(\underline{v}) = h(\alpha,\beta)\mathcal{F}_{M,N}(\underline{u}). \tag{3.46}$$

Set $h'(\alpha,\beta) = |h(\alpha,\beta)|^2$. Using Proposition 3.8 we have

$$\mathcal{F}_{M,N}(\underline{w}) = h'(\alpha,\beta)\mathcal{F}_{M,N}(\underline{u}). \tag{3.47}$$

Thus, (1) holds if and only if $\underline{w} = \underline{u}$ if and only if for all $(m,n) \in \Omega_{M,N}$, $h'(\alpha,\beta)_{m,n} = 1$ or $\mathcal{F}_{M,N}(\underline{u}) = 0$. As $h'(\alpha,\beta) = |h(\alpha,\beta)|^2$, (1) holds if and only if for all $(m,n) \in \Omega_{M,N}$, $|h(\alpha,\beta)|_{m,n} = 1$ or $\mathcal{F}_{M,N}(\underline{u}) = 0$. Using (3.46) we obtain the equivalence of (1) and (2). For all $(m,n) \in \Omega_{M,N} \setminus \Gamma_{M,N}$, $h(\alpha,\beta)_{m,n} = e^{-2i\pi(\alpha\frac{m}{M}+\beta\frac{n}{N})}$ so that $|h(\alpha,\beta)|_{m,n} = 1$. This shows that (2) and (3) are equivalent.

The verification of the particular cases is straightforward using Proposition 3.8. □

Note that in the proof of Proposition 3.9 the DFT translation is implicitly expressed as a filtering through (3.46). In Chapter 4 we consider another approach where the DFT translation is defined as a filtering (see Section 4.4.1).

How to deal with the non-invertibility of the DFT translation. The non-invertibility may be avoided by working with images with odd sizes, which is not always possible, or by killing the DFT coefficients on the boundary $\Gamma_{M,N}$, which modifies the image content. Alternatively it is possible to take into account the effect of the non-invertibility on the output result. For instance in Chapter 6 we propose a measure of the consistency measurement where high frequency components are discarded before the comparison.

3.3.2 Efficient Image Transformation Algorithm

The transformed image $\underline{u}_{P,\varphi}$ of $\underline{u}$ by φ using P is given by (3.30). The interpolated values correspond to the evaluation of the trigonometric polynomial P at locations $\left\{\varphi^{-1}(k,l)\right\}_{(k,l)\in\Omega_{M,N}}$, which are a priori nonequispaced. As stated in Corollary 3.1 the coefficients c of P are expressed in terms of the DFT of $\underline{u}$. When φ is a translation it was shown in Section 3.3.1 that $\underline{u}_{P,\varphi}$ is computed by phase shift and an inverse DFT. Using the FFT algorithm [24] it can be obtained in $O(MN\log(MN))$ floating point operations. As described below, in general the computation of the transformed image is more costly but can be approximated with an efficient algorithm.

Trigonometric polynomial evaluation. Let us consider the general problem of trigonometric polynomial evaluation at arbitrary locations. Let N_s be the number of output values and $\{(x_j,y_j)\}_{1\le j\le N_s}$ be the locations. Then, P can be evaluated at (x_j,y_j) directly from the coefficients using the formula

$$P(x_j,y_j) = \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} c_{m,n} e^{2i\pi\left(\frac{mx_j}{M}+\frac{ny_j}{N}\right)}. \tag{3.48}$$

The total cost is of $O(MNN_s)$ floating point operations so that this operation cannot be done in practice.

Let us introduce the nonequispaced discrete Fourier transform (NDFT) algorithm. Let M' and N' be two even numbers and $(\hat{f}_{m,n})_{(m,n)\in\hat{\Omega}_{M',N'}}$. Let $\left\{(x'_j,y'_j)\right\}_{1\le j\le N_s}$ be pixel positions in $[-\frac{1}{2},\frac{1}{2})^2$. The NDFT evaluates the sums

$$f(x'_j,y'_j) = \sum_{(m,n)\in\hat{\Omega}_{M',N'}} \hat{f}_{m,n} e^{-2i\pi(x'_j m+y'_j n)}. \tag{3.49}$$

The evaluation is done using the straightforward matrix form and requires $O(M'N'N_s)$ floating point operations.

It is clear that the expressions in (3.48) and (3.49) are closely related. Actually P can be evaluated using the NDFT algorithm. The correspondences between positions are given by

$$\begin{cases} x'_j = -\frac{x_j}{M} + \alpha_j, \\ y'_j = -\frac{y_j}{N} + \beta_j \end{cases} \tag{3.50}$$

where α_j and β_j are integers insuring $(x'_j, y'_j) \in [-\frac{1}{2}, \frac{1}{2})^2$. The even bandwidth M' and N' sizes are taken as

$$M' = \begin{cases} M+1 & \text{if M is odd,} \\ M+2 & \text{if M is even} \end{cases} \quad \text{and} \quad N' = \begin{cases} N+1 & \text{if N is odd,} \\ N+2 & \text{if N is even.} \end{cases} \tag{3.51}$$

The Fourier coefficients $(\hat{f}_{m,n})_{(m,n)\in\hat{\Omega}_{M',N'}}$ are obtained from $(\hat{c}_{m,n})_{(m,n)\in\hat{\Omega}^s_{M,N}}$ by zero-padding on the boundary $\Gamma_{M',N'}$ i.e.

$$\hat{f}_{m,n} = \begin{cases} c_{m,n} & (m,n) \in \hat{\Omega}^s_{M,N}, \\ 0 & (m,n) \in \Gamma_{M',N'}. \end{cases} \tag{3.52}$$

The interest of considering the NDFT formulation is that it can be efficiently approximated by the nonequispaced fast Fourier transform [94] (NFFT) algorithm. The NFFT approximation is based on the usage of an oversampled FFT and a window function which is simultaneously localised in space and frequency. It only requires $O(M'N'\log(M'N') + |\log(\varepsilon)|^2 N_s)$ operations where ε denotes the desired (relative) output precision. Details concerning the NFFT performances are provided in [65].

How to transform an image. Algorithm 3.1 details how the image $\underline{u}$ is transformed by φ using the trigonometric polynomial interpolator P. First the locations $\{\varphi^{-1}(k,l)\}_{(k,l)\in\Omega_{M,N}}$ are computed. Then, the DFT coefficients of $\underline{u}$ are computed (thanks to the FFT algorithm [24] in $O(MN\log(MN))$ floating point operations) and linked to the coefficients of P. Finally the values $(\underline{u}_{P,\varphi})_{k,l} = P(\varphi^{-1}(k,l))$ are approximated on $\Omega_{M,N}$ using the NFFT algorithm using the correspondances provided in (3.50), (3.51) and (3.52). An implementation of this algorithm can be done using the FFTW library [42] and the NFFT3 library [63].

Algorithm 3.1: Transformation of an image using trigonometric polynomial interpolation

Input : An image $\underline{u}$ of size $M \times N$, the geometric transformation φ and the trigonometric polynomial interpolator P

Output: The transformed image $\underline{u}_{P,\varphi}$

1 Compute the locations $\{\varphi^{-1}(k,l)\}_{(k,l)\in\Omega_{M,N}}$

2 Compute the DFT coefficients of $\underline{u}$ with the FFT algorithm

3 Deduce the coefficients of P from the DFT coefficients

4 Compute $(\underline{u}_{P,\varphi})_{k,l} = P(\varphi^{-1}(k,l))$ on $\Omega_{M,N}$ from the coefficients of P using the NFFT algorithm. See (3.50), (3.51) and (3.52) for the correspondences.

Assume that the trigonometric polynomial interpolator P is one of the three classical interpolators i.e. $P \in \{P^{(c)}_{\underline{u}}, \mathcal{R}(P^{(c)}_{\underline{u}}), P^{(r)}_{\underline{u}}\}$. When $P = P^{(c)}_{\underline{u}}$ it is possible to avoid unnecessary computations during the NFFT by keeping even bandwidths. M' and N', given in (3.51), are replaced by

$$M' = \begin{cases} M+1 & \text{if M is odd,} \\ M & \text{if M is even} \end{cases} \quad \text{and} \quad N' = \begin{cases} N+1 & \text{if N is odd,} \\ N & \text{if N is even.} \end{cases} \tag{3.53}$$

The result for $P \in \{\mathcal{R}(P_{\underline{u}}^{(c)}), P_{\underline{u}}^{(r)}\}$ can be obtained by taking the real part and by possibly using (3.27).

3.4 Up-sampling and Down-sampling

Up-sampling and down-sampling are common operations in image processing. For instance in multi-scale approaches the down-sampling factor corresponds to the pyramid scale. These operations cannot be interpreted as geometric transformations as described in Section 3.3 since the image sizes change. However they involve a spatial scaling and are closely related to zooming. That is why it is common to (improperly) refer to up-sampling as a zoom-in and to down-sampling as a zoom-out.

Let z be a positive integer representing the resampling factor. In this section we present the up-sampling and down-sampling by factor z using TPI. It is shown that DFT based computations can be performed and the link between the two operations is established. Note that this study may be extended to rational resampling factor.

3.4.1 Up-sampling

Here up-sampling refers to the process of increasing the sampling rate of a signal/image. As no additional information is provided, up-sampling can be seen as resampling on a finer grid using interpolation. Let $P \in \mathcal{P}_{M,N}$ be a trigonometric polynomial interpolator of $\underline{u}$ with coefficients $c \in \hat{\Omega}_{M,N}$. First the up-sampling of $\underline{u}$ by factor z using P is defined as in

Definition 3.11 (Up-sampling of an image)**.** *The up-sampled image of $\underline{u}$ by factor z using the interpolator $P \in \mathcal{P}_{M,N}$ is noted $\underline{u}_{(P,z)} \in \mathbb{C}^{\Omega_{zM,zN}}$ and is defined by*

$$\forall (k,l) \in \Omega_{zM,zN}, \quad \left(\underline{u}_{(P,z)}\right)_{k,l} = P\left(\frac{k}{z}, \frac{l}{z}\right). \tag{3.54}$$

Note that the scaling of factor z is involved in (3.54). The DFT coefficients of the up-sampled image $\underline{u}_{(P,z)}$ can be easily computed from the trigonometric polynomial coefficients by padding with zeros as stated in

Proposition 3.10. *For $(m,n) \in \hat{\Omega}_{zM,zN}$,*

$$\mathcal{F}_{zM,zN}(\underline{u}_{(P,z)})_{m,n} = \begin{cases} c_{m,n} & \text{if } (m,n) \in \hat{\Omega}^s_{M,N} \\ 0 & \text{otherwise.} \end{cases} \tag{3.55}$$

Proof. The result is obvious for $z = 1$. Now assume $z > 1$. For $(k,l) \in \Omega_{zM,zN}$ we have

$$\begin{aligned} \left(\underline{u}_{(P,z)}\right)_{k,l} &= P\left(\frac{k}{z}, \frac{l}{z}\right) && (3.56) \\ &= \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} c_{m,n} e^{2i\pi\left(\frac{mk}{zM} + \frac{nl}{zN}\right)} && (3.57) \\ &= \sum_{(m,n)\in\hat{\Omega}_{zM,zN}} d_{m,n} e^{2i\pi\left(\frac{mk}{zM} + \frac{nl}{zN}\right)} && (3.58) \end{aligned}$$

where

$$d_{m,n} = \begin{cases} c_{m,n} & \text{if } (m,n) \in \hat{\Omega}^s_{M,N} \\ 0 & \text{otherwise.} \end{cases} \tag{3.59}$$

By uniqueness of the iDFT (of size $zM \times zN$) we obtain the result. □

Therefore the up-sampling using TPI is commonly referred to as the DFT zero-padding. We recall that the trigonometric polynomial coefficients are expressed in terms of the DFT of $\underline{u}$ (see Corollary 3.1). In particular, for the three classical trigonometric polynomial interpolators the DFT coefficients of the up-sampled images are given by

Proposition 3.11. *For* $(m,n) \in \hat{\Omega}_{zM,zN}$,

$$\mathcal{F}_{zM,zN}(\underline{u}_{(P_{\underline{u}}^{(c)},z)})_{m,n} = \begin{cases} \mathcal{F}_{M,N}(\underline{u})_{m,n} & \text{if } (m,n) \in \hat{\Omega}_{M,N} \\ 0 & \text{otherwise.} \end{cases} \tag{3.60}$$

If in addition $\underline{u}$ *is real-valued, then for* $(m,n) \in \Omega_{zM,zN}$

$$\mathcal{F}_{zM,zN}(\underline{u}_{(\mathcal{R}(P_{\underline{u}}^{(c)}),z)})_{m,n} = \begin{cases} \mathcal{F}_{M,N}(\underline{u})_{m,n} & \text{if } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} & \text{if } |m| = \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{m,-\frac{N}{2}} & \text{if } |m| < \frac{M}{2} \text{ and } |n| = \frac{N}{2} \\ \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}} & \text{if } (m,n) \in \{\pm(\frac{M}{2},\frac{N}{2})\} \\ 0 & \text{otherwise} \end{cases} \tag{3.61}$$

and

$$\mathcal{F}_{zM,zN}(\underline{u}_{(P_{\underline{u}}^{(r)},z)})_{m,n} = \begin{cases} \mathcal{F}_{M,N}(\underline{u})_{m,n} & \text{if } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},n} & \text{if } |m| = \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \frac{1}{2}\mathcal{F}_{M,N}(\underline{u})_{m,-\frac{N}{2}} & \text{if } |m| < \frac{M}{2} \text{ and } |n| = \frac{N}{2} \\ \frac{1}{4}\mathcal{F}_{M,N}(\underline{u})_{-\frac{M}{2},-\frac{N}{2}} & \text{if } |m| = \frac{M}{2} \text{ and } |n| = \frac{N}{2} \\ 0 & \text{otherwise.} \end{cases} \tag{3.62}$$

Proof. It is a direct consequence of Proposition 3.10. The coefficients of the different trigonometric polynomial interpolators are given in Definition 3.8, Proposition 3.2 and Definition 3.9. □

Note that the up-sampled image using the trigonometric polynomial interpolator in complex convention $P_{\underline{u}}^{(c)}$ may be complex-valued for $\underline{u}$ real-valued. Actually it is real-valued if and only if $\{(m,n) \in \Gamma_{M,N} \,|\, \mathcal{F}_{M,N}(\underline{u})_{m,n} \neq 0\} = \emptyset$.

3.4.2 Down-sampling

Here down-sampling refers to the process of reducing the sampling rate of a signal/image. It is also called decimation and is usually expressed as a two-step process. First the high-frequency component is reduced by a low-pass filtering. Then, the down-sampled image is obtained by keeping only one sample over z^2 of the filtered image. The aim of the low-pass filtering is to avoid the introduction of a strong aliasing.

Let $\underline{v}$ be an image of size $zM \times zN$. A natural definition for the down-sampling by factor z of $\underline{v}$ is $(\underline{v}_{zk,zl})_{(k,l)\in\Omega_{M,N}}$. It does not require low-pass filtering but the output image may be aliased. Discrete spatial filters may be considered for the low-pass filtering. A classical example is the Gaussian filter (e.g. with standard deviation $\sigma = 0.6$). Low-pass filters may also be defined in the Fourier domain since it covers all the possible image sizes in a single formula. As described in Chapter 4 the filters are then applied using DFT-based computations that rely on TPI. For instance it is the case for the low-pass filter used to build the steerable pyramid of E. Simoncelli et al. [111].

The down-sampling of $\underline{v}$ by factor z using TPI is defined as in

Definition 3.12 (Down-sampling of an image)**.** *Let $z \geq 2$ and $\underline{v}$ be an image of size $zM \times zN$. The down-sampled image of $\underline{v}$ with factor z is noted $\underline{v}_{(\frac{1}{z})} \in \mathbb{C}^{\Omega_{M,N}}$. It is defined by*

$$\forall (k,l) \in \Omega_{M,N}, \quad \left(\underline{v}_{(\frac{1}{z})}\right)_{k,l} = P_{\underline{v},\frac{1}{z}}(k,l) \tag{3.63}$$

where $P_{\underline{v},\frac{1}{z}} \in \mathcal{P}_{M,N}$ is given by

$$\forall (x,y) \in \mathbb{Z}^2, \quad P_{\underline{v},\frac{1}{z}}(x,y) = \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} \mathcal{F}_{zM,zN}(\underline{u})_{m,n} e^{2i\pi\left(\frac{m}{M}x+\frac{n}{N}y\right)}. \tag{3.64}$$

The definition is naturally extended to $z = 1$ by setting $\underline{v}_{(1)} = \underline{v}$.

The DFT coefficients of the zoomed image $\underline{v}_{(\frac{1}{z})}$ can be easily computed from DFT coefficients of $\underline{v}$ as stated in

Proposition 3.12. *Assume that $z \geq 2$ and let $\underline{v}$ be an image of size $zM \times zN$. Then for $(m,n) \in \hat{\Omega}_{M,N}$,*

$$\mathcal{F}_{M,N}(\underline{v}_{(\frac{1}{z})})_{m,n} = \begin{cases} \mathcal{F}_{zM,zN}(\underline{v})_{m,n} & \text{if } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \mathcal{F}_{zM,zN}(\underline{v})_{\frac{M}{2},n} + \mathcal{F}_{zM,zN}(\underline{v})_{-\frac{M}{2},n} & \text{if } m = -\frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ \mathcal{F}_{zM,zN}(\underline{v})_{m,-\frac{N}{2}} + \mathcal{F}_{zM,zN}(\underline{v})_{m,-\frac{N}{2}} & \text{if } |m| < \frac{M}{2} \text{ and } n = -\frac{N}{2} \\ \mathcal{F}_{zM,zN}(\underline{v})_{\frac{M}{2},\frac{N}{2}} + \mathcal{F}_{zM,zN}(\underline{v})_{\frac{M}{2},-\frac{N}{2}} & \\ + \mathcal{F}_{zM,zN}(\underline{v})_{-\frac{M}{2},\frac{N}{2}} + \mathcal{F}_{zM,zN}(\underline{v})_{-\frac{M}{2},-\frac{N}{2}} & \text{if } m = -\frac{M}{2} \text{ and } n = -\frac{N}{2}. \end{cases} \tag{3.65}$$

Proof. It is obtained by using similar computations as in the proof of Proposition 3.1. □

The down-sampling using TPI is commonly called DFT zoom-out. For $z \geq 2$ it relies on TPI through $P_{\underline{v},\frac{1}{z}}$, which is defined in (3.64) without ambiguity since it is built from the DFT coefficients of $\underline{v}$ on $\hat{\Omega}^s_{M,N}$. Intuitively $P_{\underline{v},\frac{1}{z}}$ is obtained by taking any trigonometric polynomial interpolator of $\underline{v}$, "killing" the coefficients in $\hat{\Omega}_{zM,zN} \setminus \hat{\Omega}^s_{M,N}$ and applying a scaling of factor z. The underlying continuous low-pass filter is the perfect low-pass defined by the indicator function of $[-\frac{\pi}{z},\frac{\pi}{z}]^2$.

3.4.3 Link between Up-sampling and Down-sampling

The left invertibility of the up-sampling using TPI is guaranteed by

Proposition 3.13. *Let $P \in \mathcal{P}_{M,N}$ be a trigonometric polynomial interpolator of $\underline{u}$. Then,*

$$\left(\underline{u}_{(P,z)}\right)_{(\frac{1}{z})} = \underline{u}. \tag{3.66}$$

Proof. It is obtained using Proposition 3.10 and Proposition 3.12. □

More precisely, the left inverse of the up-sampling with any trigonometric polynomial interpolator is the down-sampling with the same factor. Obviously for $z \geq 2$ the up-sampling does not admit a right inverse since up-sampled images have imposed null DFT coefficients. Similarly the down-sampling does not admit a left inverse since DFT coefficients are "killed" i.e. some information is lost.

3.5 Conclusion

In this chapter we presented a trigonometric polynomial interpolation theory for images and applied it to the geometric transformation of images, to the up-sampling and to the down-sampling by an integer factor. The trigonometric polynomial interpolators of an image were characterized and it was shown that there are infinitely many candidates as soon as one of the image dimensions is even. The interpolator choice has an influence as shown in the two discussed applications. Three classical choices of interpolator for real-valued images were presented and the cases where they coincide were pointed out.

For image translation, the classical DFT-based computations by phase shift were described. In the general case an efficient but approximate algorithm, based on the NFFT algorithm, was proposed. DFT-based computations were also presented for the up-sampling and down-sampling by an integer factor. All of the algorithms described are efficient and can be used in practice. Trigonometric polynomial interpolation is also applied to linear filtering in Chapter 4.

The performances and the limits of trigonometric polynomial interpolation are discussed in Chapter 6.

Chapter 4

Filtering using Trigonometric Polynomial Interpolation

Abstract

This chapter is taken from [15] and is an application of Chapter 3. We propose algorithms for filtering real-valued images, when the filter is provided as a continuous function defined in the Nyquist frequency domain. This problem is ambiguous because images are discrete entities and there is no unique way to define the filtering. We provide a theoretical framework designed to analyse the classical and computationally efficient filtering implementations based on discrete Fourier transforms (DFT). In this framework, the filtering is interpreted as the convolution of a distribution, standing for the filter, with a trigonometric polynomial interpolator of the image. The various plausible interpolations and choices of the distribution lead to three equally licit algorithms which can be seen as method variants of the same standard filtering algorithm. In general none should be preferred to the others and the choice depends on the application. In practice, the method differences, which come from the boundary DFT coefficients, are not visible to the naked eye. We demonstrate that claim on several experimental configurations by varying the input image and the considered filter. In some cases however, we discuss how the choice of the variant may affect fundamental properties of the filtering. We provide an implementation[1] of the algorithms.

Contents

[1] http://www.ipol.im/pub/art/2016/116/

4.1 Introduction

The application to images of a filter defined by a continuous function in the frequency domain, more precisely in the Nyquist domain $[-\pi,\pi]^2$, is a well-known problem. Images are discrete entities so there is no unique way to define the filtering whether it be as a continuous or a discrete convolution.

The definition in the Fourier domain by a continuous function may be handy to impose a given property to a filter (e.g. low-pass, high-pass or steered filter). Moreover, it covers all the possible image sizes in a single formula. For instance this is a classical approach used to build multi-scale structures (e.g. the steerable pyramid of E. Simoncelli et al. [111]).

In practice, the filtering is usually applied by performing discrete Fourier transform (DFT) computations [17, 45, 89] but, to the best of our knowledge, no clear theoretical justification can be found in the literature. In particular, the interpretations generally involve a trigonometric polynomial interpolation. As shown in Chapter 3, it is ambiguous for even-sized images and relies on the complex convention whose influence on the boundary DFT coefficients should be taken into account.

The aim of this chapter is to give a clear definition, as a continuous convolution, of the filtering of a discrete real-valued image by a filter specified by a continuous function in the Nyquist domain $[-\pi,\pi]^2$. Additionally, for computational purposes it should be compatible with DFT computations.

With this goal in mind, we interpret the filtering as the convolution of a distribution, standing for the filter, with a trigonometric polynomial interpolator of the image. The various plausible interpolations and choices of the distribution lead to three equally licit algorithms which can be seen as variants of the same standard filtering algorithm. In general none should be preferred to the others and the choice depends on the application. The method differences, which come from the boundary DFT coefficients, are not visible to the naked eye in practice. This analysis is illustrated by an application to several fundamental filters. When necessary, we also discuss the impact of the chosen method on some desired properties of the filtering (e.g. semi-group property, exact reconstruction).

The remainder of this chapter is organized as follows: We present in Section 4.2 the theory intended to give a clear interpretation of the filtering of a discrete real-valued image by a filter defined by a continuous function in the Nyquist domain. It underlies the three proposed algorithms which are detailed in Section 4.3. Finally an experimental study with fundamental filter examples is presented in Section 4.4.

4.2 Theoretical Results

Let $\phi : [-\pi,\pi]^2 \to \mathbb{C}$ be a continuous function. Let M and N be two positive integers and $\underline{u}$ be a discrete real-valued image of size $M \times N$.

In this section we introduce the theory intended to give a clear interpretation of the filtering of $\underline{u}$ by a filter defined by ϕ. It is formalized as the convolution between a trigonometric polynomial interpolator $P_{\underline{u}}$, associated with the image $\underline{u}$, and a tempered distribution f_ϕ associated with the filter ϕ, whose Fourier transform is continuous on $[-\pi,\pi]^2$. This theory is aimed at being consistent with the DFT implementations that are typical in the literature.

Section 4.2.1 justifies that the convolution between the tempered distribution f_ϕ and the trigonometric polynomial $P_{\underline{u}}$ is mathematically well sounded (see Theorem 4.2). In Section 4.2.2, according to the choice of $P_{\underline{u}}$ and f_ϕ, we consider three different but equally licit ways of defining the filtering of discrete images (see Definition 4.9). The resulting filtered images can be efficiently obtained by DFT based computations thanks to Proposition 4.2.

4.2.1 Convolution of Trigonometric Polynomials

In this part we define the convolution between a tempered distribution whose Fourier transform is a continuous function on $[-\pi,\pi]^2$ and a trigonometric polynomial. This convolution will be used in Section 4.2.2 to define the filtering of an image. The distribution will represent the filter and the trigonometric polynomial the image. To do this we extend known results on the set of integrable functions.

Definition 4.1 (Fourier transform)**.** *The continuous Fourier transform applied to $L^1(\mathbb{R}^2)$ and its inverse, denoted $\mathcal{F}$ and $\mathcal{F}^{-1}$, are defined by*

$$\forall u \in L^1(\mathbb{R}^2),\quad \forall(\xi,\nu)\in\mathbb{R}^2,\quad \hat{u}(\xi,\nu)=\mathcal{F}(u)(\xi,\nu)=\int_{\mathbb{R}^2} u(x,y)e^{-i(x\xi+y\nu)}dxdy \tag{4.1}$$

$$\text{and}\quad \forall(x,y)\in\mathbb{R}^2,\quad \mathcal{F}^{-1}(u)(x,y)=\frac{1}{(2\pi)^2}\int_{\mathbb{R}^2} u(\xi,\nu)e^{i(x\xi+y\nu)}d\xi d\nu. \tag{4.2}$$

The Fourier transform extended to tempered distributions [107, 113] is still denoted $\mathcal{F}$.

Definition 4.2 (Convolution)**.** *Let $u\in L^1(\mathbb{R}^2)$ and $v\in L^\infty(\mathbb{R}^2)$. The convolution of u and v, denoted $u*v$, is defined by*

$$\forall(x,y)\in\mathbb{R}^2,\quad u*v(x,y)=\int_{\mathbb{R}^2} u(x-s,y-t)v(s,t)dsdt. \tag{4.3}$$

We recall that the notations and definitions relative to trigonometric polynomials were introduced in Section 3.2.1. As a trigonometric polynomial is bounded, its convolution with an integrable function is well-defined and the following result holds.

Theorem 4.1 (Convolution theorem 1 [62])**.** *Let $f\in L^1(\mathbb{R}^2)$ and $P\in\mathcal{P}_{M,N}$ with coefficients c. Then, $f*P\in\mathcal{P}_{M,N}$ and verifies*

$$\forall(x,y)\in\mathbb{R}^2,\quad (f*P)(x,y)=\sum_{(m,n)\in\hat{\Omega}^s_{M,N}} \hat{f}\left(\frac{2\pi m}{M},\frac{2\pi n}{N}\right)c_{m,n}e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})}. \tag{4.4}$$

Theorem 4.1 states that this convolution is also a trigonometric polynomial whose coefficients depend on the values of the Fourier transform of the integrable function on $[-\pi,\pi]^2$ and the trigonometric polynomial coefficients. We now want to extend this result to the wider set of distributions defined as follows.

Definition 4.3. *A distribution T of the plane is said to be continuous on $[-\pi,\pi]^2$ if its restriction[2] $T_{]-\pi,\pi[^2}$ is a function of $]-\pi,\pi[^2$ admitting a continuous extension to $[-\pi,\pi]^2$. By construction, this extension is unique.*

Definition 4.4. *We define CF_π as the subspace of tempered distributions of the plane whose Fourier transforms are continuous on $[-\pi,\pi]^2$ (in the sense of Definition 4.3).*

Note that we have $L^1(\mathbb{R}^2)\subset CF_\pi$ since the Fourier transform of an integrable function is continuous on $\mathbb{R}^2$. As the extension of the convolution to CF_π will only depend on the Fourier transforms of the distributions on $[-\pi,\pi]^2$ we consider the following equivalence relation and quotient space.

[2] Note that we can define distribution restrictions only on open sets.

Definition 4.5. *We define the equivalence relation $\mathcal{R}$ on CF_π by*

$$f\mathcal{R}g \Leftrightarrow \hat{f} = \hat{g} \text{ on } [-\pi,\pi]^2. \tag{4.5}$$

The set of equivalence classes is denoted $\langle CF_\pi\rangle$ and the equivalence class of $f \in CF_\pi$ is denoted $\langle f\rangle$. We define on the vectorial space $\langle CF_\pi\rangle$ the norm $N : \langle CF_\pi\rangle \to \mathbb{R}^+$ by

$$\forall \langle f\rangle \in \langle CF_\pi\rangle, \quad N(\langle f\rangle) = \sup_{[-\pi,\pi]^2} |\hat{f}|. \tag{4.6}$$

The equivalence classes which admit an integrable representative play a special role since we have the following density result.

Proposition 4.1. *Let $\langle L^1(\mathbb{R}^2)\rangle = \left\{\langle f\rangle \in \langle CF_\pi\rangle \quad | \quad f \in L^1(\mathbb{R}^2)\right\}$. Then $\langle L^1(\mathbb{R}^2)\rangle$ is dense in $(\langle CF_\pi\rangle, N)$.*

Proof. Let $\langle f\rangle \in \langle \mathrm{CF}_\pi\rangle$. Starting from the definitions of CF_π and $\langle \mathrm{CF}_\pi\rangle$ we can easily build $g \in \mathrm{CF}_\pi$ such that $\langle g\rangle = \langle f\rangle$ and $\hat{g}$ is a continuous function on $\mathbb{R}^2$. Now let $(h_n)_{n\in\mathbb{N}}$ be a sequence of Schwartz functions (on $\mathbb{R}^2$) that converges uniformly to $\hat{g}$ on $[-\pi,\pi]^2$. Then for each $n \in \mathbb{N}$, $g_n = \mathcal{F}^{-1}(h_n)$ is also a Schwartz function and thus is in $L^1(\mathbb{R}^2)$. By construction, $(\langle g_n\rangle)_{n\in\mathbb{N}} \in \langle L^1(\mathbb{R}^2)\rangle^{\mathbb{N}}$ and converges in the norm N to $\langle g\rangle = \langle f\rangle$. □

Starting from Theorem 4.1 we can define the convolution between an element of $\langle L^1(\mathbb{R}^2)\rangle$ and a trigonometric polynomial.

Definition 4.6. *Let $P \in \mathcal{P}_{M,N}$ with coefficients c and $\langle f\rangle \in \langle L^1(\mathbb{R}^2)\rangle$. We define the convolution of $\langle f\rangle$ and P, still denoted $\langle f\rangle * P$, as the trigonometric polynomial of degree $M \times N$ verifying*

$$\forall (x,y) \in \mathbb{R}^2, \quad (\langle f\rangle * P)(x,y) = \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} \hat{f}\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right) c_{m,n} e^{2i\pi(x\frac{m}{M} + y\frac{n}{N})}. \tag{4.7}$$

This definition is compatible with the classical definition of the convolution given in Definition 4.2. It is extended to $\langle \mathrm{CF}_\pi\rangle$ in the following theorem.

Theorem 4.2 (Convolution theorem 2)**.** *Let $P \in \mathcal{P}_{M,N}$ with coefficients c. The bounded linear transformation $T_P : \langle f\rangle \in (\langle L^1(\mathbb{R}^2)\rangle, N) \mapsto \langle f\rangle * P \in (\mathcal{P}_{M,N}, \|.\|_\infty)$ can be uniquely extended to a bounded linear transformation $\tilde{T}_P : (\langle CF_\pi\rangle, N) \to (\mathcal{P}_{M,N}, \|.\|_\infty)$.*

Let $\langle f\rangle \in \langle CF_\pi\rangle$ then $\tilde{T}_P(\langle f\rangle)$ is called the convolution of $\langle f\rangle$ and P. It is denoted $\langle f\rangle * P$ and verifies for all $(x,y) \in \mathbb{R}^2$,

$$(\langle f\rangle * P)(x,y) = \tilde{T}_P(\langle f\rangle)(x,y) = \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} \hat{f}\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right) c_{m,n} e^{2i\pi(x\frac{m}{M} + y\frac{n}{N})}. \tag{4.8}$$

The convolution of $f \in CF_\pi$ with a trigonometric polynomial P, still denoted $f * P$, is then defined by $f * P = \langle f\rangle * P$.

Proof. The transformation T_P is clearly linear and is bounded since for all $\langle f\rangle \in \langle \mathrm{CF}_\pi\rangle$,

$$\|T_P(\langle f\rangle)\|_\infty \leq \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} \left|\hat{f}\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right) c_{m,n}\right| \leq \left(\sum_{(m,n)\in\hat{\Omega}^s_{M,N}} c_{m,n}\right) N(\langle f\rangle). \tag{4.9}$$

As $(\mathcal{P}_{M,N}, \|.\|_\infty)$ is a complete normed linear space and $\langle L^1(\mathbb{R}^2)\rangle$ is dense in $(\langle \mathrm{CF}_\pi\rangle, N)$ (see Proposition 4.1) then the bounded linear transformation (B.L.T) theorem [96, p. 9] states that T_P can be uniquely extended to a bounded linear transformation $\tilde{T}_P : (\langle \mathrm{CF}_\pi\rangle, N) \to (\mathcal{P}_{M,N}, \|.\|_\infty)$.

Let $\langle f \rangle \in \langle \mathrm{CF}_\pi \rangle$ and consider a sequence $(\langle f_k \rangle)_{k\in\mathbb{N}} \in \langle L^1(\mathbb{R}^2) \rangle^N$ that converges to $\langle f \rangle$ in norm N. Define

$$g_{\langle f\rangle,P} : (x,y) \in \mathbb{R}^2 \mapsto \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} \hat{f}\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right) c_{m,n} e^{2i\pi\left(x\frac{m}{M}+y\frac{n}{N}\right)}. \tag{4.10}$$

Then,

$$\begin{aligned} \left\| \langle f_k \rangle * P - g_{\langle f\rangle,P} \right\|_\infty &\leq \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} \left| (\hat{f} - \hat{f}_k)\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right) c_{m,n} \right| && (4.11)\\ &\leq \left(\sum_{(m,n)\in\hat{\Omega}^s_{M,N}} |c_{m,n}| \right) N(f - f_k) && (4.12)\\ &\underset{k\to\infty}{\longrightarrow} 0. && (4.13) \end{aligned}$$

Thus by continuity we have $\tilde{T}_P = g_{\langle f\rangle,P}$. □

Therefore, Theorem 4.2 provides a sound definition for the convolution of a tempered distribution whose Fourier transform is continuous on $[-\pi,\pi]^2$ with a trigonometric polynomial. The resulting distribution is itself also a trigonometric polynomial. It is also interesting to point out that this convolution is entirely independent from the distribution of $\hat{f}$ outside $[-\pi,\pi]^2$.

4.2.2 Discrete Image Filtering

Thanks to Theorem 4.2 we can define the filtering of discrete real-valued images. A continuous function on $[-\pi,\pi]^2$ is sufficient to define a filter. In the rest of the chapter we will indistinctly refer to ϕ as a function that defines a filter and the filter itself.

Definition 4.7. *We denote by f_ϕ any element of the equivalence class*

$$\left\{ f \in CF_\pi \quad | \quad \hat{f} = \phi \text{ on } [-\pi,\pi]^2 \right\}. \tag{4.14}$$

In Definition 4.7 we introduce f_ϕ as any representative of the equivalence class of tempered distributions whose Fourier transforms are equal to the filter ϕ on $[-\pi,\pi]^2$. The filtering of real-valued images by ϕ can be defined as the convolution[3] of f_ϕ with trigonometric polynomial interpolators of the images. The choice between the complex and real conventions (see Section 3.2.3) provides two equally licit definitions of the filtering. A third one is obtained by considering a slight modification of the filter that removes the ambiguity due to the Fourier coefficients located at the boundary of the Nyquist domain.

Definition 4.8. *We define the continuous function $\phi_{M,N} : [-\pi,\pi]^2 \to \mathbb{C}$ by*

$$\forall (\mu,\nu) \in \mathbb{R}^2, \quad \phi_{M,N}(\mu,\nu) = \phi(\mu,\nu) l_M(|\mu|) l_N(|\nu|) \tag{4.15}$$

where, for a positive integer L, we note

$$l_L : x \in [0,\pi] \mapsto \begin{cases} 1 & \text{if } 0 \leq x \leq \pi - \frac{1}{L} \\ L(\pi - x) & \text{if } \pi - \frac{1}{L} < x \leq \pi. \end{cases} \tag{4.16}$$

As an example, we display in Figure 4.1 the continuous piece-wise linear function l_L near π for $L = 32$.

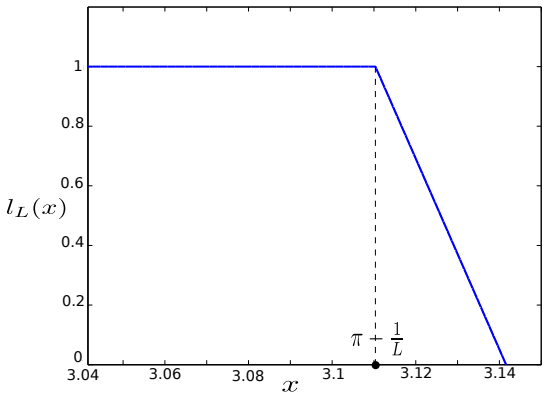


Figure 4.1: The continuous piece-wise linear function l_L near π for $L = 32$.

The function $\phi_{M,N}$ introduced in Definition 4.8 is constructed so that for $(m,n) \in \hat{\Omega}^s_{M,N}$,

$$\phi_{M,N}\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right) = \begin{cases} \phi\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right) & \text{if } |m| < \frac{M}{2} \text{ and } |n| < \frac{N}{2} \\ 0 & \text{if } (m,n) \in \Gamma^s_{M,N}. \end{cases} \tag{4.17}$$

This entails that $f_{\phi_{M,N}} * P^{(c)}_{\underline{u}} = f_{\phi_{M,N}} * P^{(r)}_{\underline{u}}$. It allows us to define the filtering of real-valued images as below.

Definition 4.9 (Filtering real-valued images)**.** *1. The filtering $\phi_1(\underline{u})$ of $\underline{u}$ by ϕ in complex convention is defined as the convolution*

$$\phi_1(\underline{u}) = f_\phi * P^{(c)}_{\underline{u}}. \tag{4.18}$$

2. The filtering $\phi_2(\underline{u})$ of $\underline{u}$ by ϕ in real convention is defined as the convolution

$$\phi_2(\underline{u}) = f_\phi * P^{(r)}_{\underline{u}}. \tag{4.19}$$

3. The windowed filtering $\phi_3(\underline{u})$ of $\underline{u}$ by ϕ is defined as the convolution

$$\phi_3(\underline{u}) = f_{\phi_{M,N}} * P_{\underline{u}} \tag{4.20}$$

where $P_{\underline{u}}$ can be taken indifferently as $P^{(c)}_{\underline{u}}$ or $P^{(r)}_{\underline{u}}$.

Let $j \in \{1,2,3\}$. The filtered image $\underline{\phi_j(\underline{u})}$ is defined as the image canonically associated to the filtering $\phi_j(\underline{u})$ i.e

$$\underline{\phi_j(\underline{u})} = (\phi_j(\underline{u})(k,l))_{(k,l)\in\Omega_{M,N}}. \tag{4.21}$$

In Definition 4.9 the three proposed results for the filtering of an image $\underline{u}$ by the filter ϕ are trigonometric polynomials whose coefficients depend on samples of ϕ and on the DFT of $\underline{u}$. Applying Theorem 4.2 we see that their coefficients only differ at the boundary $\Gamma^s_{M,N}$ of $\hat{\Omega}^s_{M,N}$. In particular when M and N are odd numbers, the three definitions are equivalent.

Now let us present how the resulting filtered images can be obtained by using DFT computations.

[3] We recall that this convolution does not depend on the choice of the representative f_ϕ.

Definition 4.10 (Spectral samples)**.** *1. The spectral samples* $(S^{1,M,N}_{m,n})_{(m,n)\in\hat{\Omega}_{M,N}}$ *of size* $M\times N$ *in the complex convention are defined for all* $(m,n)\in\hat{\Omega}_{M,N}$ *by*

$$S^{1,M,N}_{m,n} = \phi\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right). \tag{4.22}$$

2. The spectral samples $(S^{2,M,N}_{m,n})_{(m,n)\in\hat{\Omega}_{M,N}}$ *of size* $M\times N$ *in the real convention are defined for all* $(m,n)\in\hat{\Omega}_{M,N}$ *by*

$$S^{2,M,N}_{m,n} = \begin{cases} \phi(\frac{2\pi m}{M}, \frac{2\pi n}{N}) & \text{if } (m,n)\notin\Gamma_{M,N} \\ \frac{1}{2}\left(\phi(-\pi, \frac{2\pi n}{N}) + \phi(\pi, \frac{2\pi n}{N})\right) & \text{if } m = -\frac{M}{2}, n\neq -\frac{N}{2} \\ \frac{1}{2}\left(\phi(\frac{2\pi m}{M}, -\pi) + \phi(\frac{2\pi m}{M}, \pi)\right) & \text{if } m\neq -\frac{M}{2}, n = -\frac{N}{2} \\ \frac{1}{4}\left(\phi(-\pi,-\pi) + \phi(\pi,-\pi) + \phi(-\pi,\pi) + \phi(\pi,\pi)\right) & \text{if } m = -\frac{M}{2}, n = -\frac{N}{2}. \end{cases} \tag{4.23}$$

3. The spectral windowed samples $(S^{3,M,N}_{m,n})_{(m,n)\in\hat{\Omega}_{M,N}}$ *of size* $M\times N$ *are defined for all* $(m,n)\in\hat{\Omega}_{M,N}$ *by*

$$S^{3,M,N}_{m,n} = \begin{cases} \phi(\frac{2\pi m}{M}, \frac{2\pi n}{N}) & \text{if } (m,n)\notin\Gamma_{M,N} \\ 0 & \text{if } (m,n)\in\Gamma_{M,N}. \end{cases} \tag{4.24}$$

Proposition 4.2. *Let* $j\in\{1,2,3\}$*. Then, the DFT of the filtered image* $\underline{\phi_j(u)}$ *is given for all* $(m,n)\in\hat{\Omega}_{M,N}$ *by*

$$\mathcal{F}_{M,N}\left(\underline{\phi_j(u)}\right)_{m,n} = S^{j,M,N}_{m,n}\,\mathcal{F}_{M,N}(\underline{u})_{m,n}. \tag{4.25}$$

Proof. We carry the proof for all three cases in turn.

Complex convention : With Theorem 4.2 and Definition 4.7 we have for $(x,y)\in\mathbb{R}^2$,

$$\phi_1(\underline{u})(x,y) = (f_\phi * P^{(c)}_{\underline{u}})(x,y) = \sum_{(m,n)\in\hat{\Omega}_{M,N}} \phi\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right)\mathcal{F}_{M,N}(\underline{u})_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})} \tag{4.26}$$

$$= \sum_{(m,n)\in\hat{\Omega}_{M,N}} S^{1,M,N}_{m,n}\,\mathcal{F}_{M,N}(\underline{u})_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})}. \tag{4.27}$$

We conclude by evaluating at $(x,y) = (k,l)\in\Omega_{M,N}$.

Real convention : Similarly we have for $(x,y)\in\mathbb{R}^2$,

$$\phi_2(\underline{u})(x,y) = (f_\phi * P^{(r)}_{\underline{u}})(x,y) = \sum_{(m,n)\in\hat{\Omega}^s_{M,N}} \phi\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right) c_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})} \tag{4.28}$$

where the coefficients c are given in Definition 3.9. Using the partition $\hat{\Omega}^s_{M,N} = \hat{\Omega}_{M,N}\setminus\Gamma_{M,N}\sqcup\Gamma^s_{M,N}$ we get

$$\phi_2(\underline{u})(x,y) = \left(\sum_{(m,n)\in\hat{\Omega}_{M,N}\setminus\Gamma_{M,N}} + \sum_{(m,n)\in\Gamma^s_{M,N}}\right)\phi\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right) c_{m,n} e^{2i\pi(x\frac{m}{M}+y\frac{n}{N})}. \tag{4.29}$$

Now let $(x,y) = (k,l)\in\Omega_{M,N}$. By using the simple identity $e^{2i\pi\frac{j}{L}\frac{L}{2}} = (-1)^j = e^{2i\pi\frac{j}{L}\frac{-L}{2}}$, we can perform a straightforward identification of the DFT coefficients of the boundary $\Gamma_{M,N}$.

Windowed filtering : The proof is similar to the complex convention one by replacing $\phi\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right)$ by 0 for $(m,n) \in \Gamma_{M,N}$.

□

As stated in Proposition 4.2, the filtered image (see Definition 4.9) can be characterized by its DFT which is the element-wise product of the DFT of the image $\underline{u}$ and the spectral samples (introduced in Definition 4.10). This observation underlies the efficient FFT based algorithm presented in Section 4.3.

4.3 Algorithms

In this section we propose three filtering algorithms taking as inputs a discrete image $\underline{u}$ and a filter specified by a continuous function ϕ defined on the Nyquist domain $[-\pi,\pi]^2$. They are presented as three variants of Algorithm 4.1, namely the common standard filtering algorithm, as follows:

- Method 1: Filtering in complex convention.
  The spectral samples are the $S^{1,M,N}$ given by Equation (4.22).
- Method 2: Filtering in real convention.
  The spectral samples are the $S^{2,M,N}$ given by Equation (4.23).
- Method 3: Windowed filtering.
  The spectral samples are the $S^{3,M,N}$ given by Equation (4.24).

We summarize in Table 4.1 how the spectral samples are computed according to the method variant. Our implementation is available at the web page[4] of [15].

Filtering method	1	2	3
Spectral samples	$S^{1,M,N}$ given by Equation (4.22)	$S^{2,M,N}$ given by Equation (4.23)	$S^{3,M,N}$ given by Equation (4.24)

Table 4.1: How to compute the spectral samples according to the filtering method.

Algorithm 4.1: Standard filtering algorithm

Input : A real-valued image $\underline{u}$ of size $M \times N$, a filter $\phi : [-\pi,\pi]^2 \to \mathbb{C}$ (continuous function) and the number $j \in \{1,2,3\}$ of the method variant.

Output: The filtered image $\underline{v}$ of size $M \times N$ corresponding to the method j.

1 Compute $\tilde{u} = \mathcal{F}_{M,N}(\underline{u})$ the DFT of $\underline{u}$.

2 Compute $S = (S_{m,n})_{(m,n)\in\hat{\Omega}_{M,N}}$ the spectral samples corresponding to method j (see Table 4.1).

3 Compute $\tilde{v} = (\tilde{u}_{m,n} S_{m,n})_{(m,n)\in\hat{\Omega}_{M,N}}$ the element-wise multiplication of $\tilde{u}$ and S.

4 Compute $\underline{v} = \mathcal{F}^{-1}_{M,N}(\tilde{v})$ the inverse DFT of $\tilde{v}$.

These algorithms rely on DFT computations whose interpretations are given in Section 4.2. Let M and N be two positive integers. For an image $\underline{u}$ of size $M \times N$ the filtered image is computed in $O(MN\log(MN))$ operations thanks to Fast Fourier transform (FFT) algorithms [42]. Color images can be filtered by applying the algorithms independently to each color channel.

[4] http://www.ipol.im/pub/art/2016/116/

The filtering in complex convention (method 1) corresponds to the classical DFT based computations that can be found in the literature [17]. Note that the three methods can be naturally extended to complex-valued images.

The three methods only differ by the computation of the spectral samples which may vary at the boundary $\Gamma_{M,N}$ of $\hat{\Omega}_{M,N}$. In particular suppose the sizes M and N are odd (i.e. $\Gamma_{M,N} = \emptyset$) then the three methods are equivalent. Similarly, suppose the continuous function ϕ that defines the filter is null at the boundary of the Nyquist domain then the three methods are equivalent for any input image of any size.

These algorithms are applicable for any image size and any filter that can be defined by a continuous function in the Nyquist domain. The current general form can be modified in particular cases. For instance, suppose we want to apply a filter to a set of images of the same sizes. Then the filtering only depends (with respect to the filter) on the spectral samples of ϕ which can be precomputed. This discretization step entails a loss of information that may lead to the following degenerated situation:

- Two different filters may lead to the same filtered images for a fixed size.
- It is easy to construct a filter that behaves in very different ways for different sizes.

4.4 Experiments

In this experimental part we apply the three methods introduced in the previous section to several fundamental filters. We start by presenting the filters before comparing the results of the different methods applied to an experimental set of four images.

4.4.1 Filters

We propose to consider the following fundamental filters.

Cardinal sine (*sinc*) filter. The first filter proposed is the simplest one since it is defined by a constant function on the Nyquist domain.

Definition 4.11 (Cardinal sine (*sinc*) filter)**.** *The cardinal sine (or sinc) filter*[5] *is defined by the constant function*

$$\phi_{sinc} : (\xi, \nu) \in [-\pi, \pi]^2 \mapsto 1. \tag{4.30}$$

Since ϕ_{sinc} is symmetric at the boundary of $[-\pi, \pi]^2$ the spectral samples in both conventions coincide i.e. method 1 and method 2 are equivalent. The filtered image obtained by applying method 1 (or 2) is exactly the input image. This property is in general false for method 3.

Shift Filter. Shifting an image is common in image processing. It can be done by using the following filter.

Definition 4.12 (Shift filter)**.** *Let $a = (a_1, a_2) \in \mathbb{R}^2$. The shift filter of parameter a is defined by the function*

$$\phi_a : (\xi, \nu) \in [-\pi, \pi]^2 \mapsto e^{i(a_1\xi + a_2\nu)} \in \mathbb{C}. \tag{4.31}$$

Derivative Filter. The derivative filters are used to highlight the variations along an axis.

[5] The name of the filter comes from the separable cardinal sine function, which Fourier transform is the indicator function of $[-\pi, \pi]^2$.

Definition 4.13 (Derivative filter)**.** *The partial derivative filter with respect to the variable x is defined by the function*

$$\varphi_x : (\xi, \nu) \in [-\pi, \pi]^2 \mapsto i\xi \in \mathbb{C}. \tag{4.32}$$

Similarly the partial derivative filter with respect to the variable y is defined by the function

$$\varphi_y : (\xi, \nu) \in [-\pi, \pi]^2 \mapsto i\nu \in \mathbb{C}. \tag{4.33}$$

In the following we choose to concentrate on the filter φ_x (the results for φ_y are similar). Let M and N be the size of the image to be filtered. We notice that when M is even and N is odd the spectral samples of method 2 are null at the boundary $\Gamma_{M,N}$ i.e. method 2 and method 3 are equivalent. When M is odd and N is even the spectral samples of method 1 and method 2 are the same i.e. method 1 and method 2 are equivalent. Of course, if both M and N are odd, all three methods coincide (since this is true for any filter).

Laplacian Filter. The Laplacian filter is used to highlight the high frequencies of images. In particular, it is often used for edge detection.

Definition 4.14 (Laplacian filter)**.** *The Laplacian filter is defined by the function*

$$L : (\xi, \nu) \in [-\pi, \pi]^2 \mapsto -\xi^2 - \nu^2 \in \mathbb{C}. \tag{4.34}$$

Since L is a radial function the spectral samples in both conventions coincide i.e. method 1 and method 2 are equivalent.

Gaussian Filter. It is common to use a Gaussian filter (e.g. to blur images).

Definition 4.15 (Gaussian filter)**.** *Let $\sigma > 0$. The Gaussian filter of standard deviation σ is defined by the function*

$$g_\sigma : (\xi, \nu) \in [-\pi, \pi]^2 \mapsto e^{-\sigma^2 \frac{\xi^2+\nu^2}{2}} \in \mathbb{R}. \tag{4.35}$$

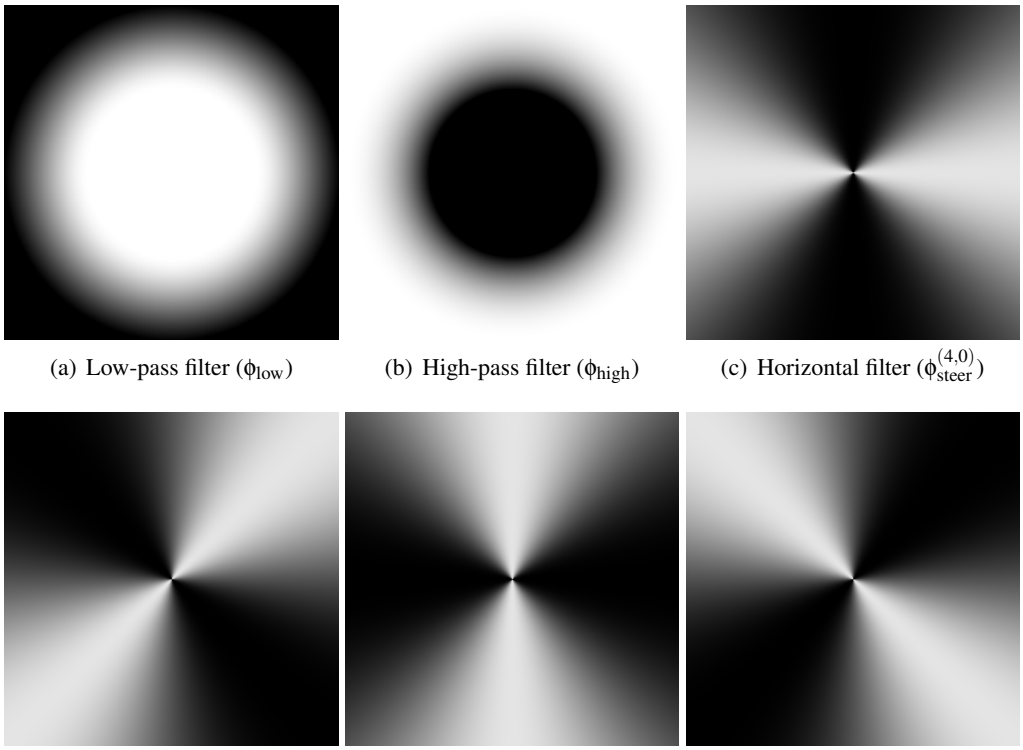
Since g_σ is a radial function the spectral samples in both conventions coincide i.e. method 1 and method 2 are equivalent.

Let us justify the definition of the filter. Let $\sigma > 0$ and define $f_\sigma : (x, y) \in \mathbb{R}^2 \mapsto \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$. Then we have,

$$\forall (\xi, \nu) \in [-\pi, \pi]^2, \quad \hat{f}_\sigma(\xi, \nu) = g_\sigma(\xi, \nu).$$

The convolution with f_σ, namely Gaussian convolution [45, 98], is a common operation and building block for algorithms in image processing. Note that for all the three methods, the sequential application of g_σ and $g_{\sigma'}$ is equivalent to the direct application of $g_{\sqrt{\sigma^2+\sigma'^2}}$ (semi-group property).

Steerable Pyramid Filters. The steerable pyramid is a linear multiscale and multi-orientation image decomposition that has been developed in the 90s by E. Simoncelli and his co-authors [93, 111]. It is designed after the receptive fields found by Hubel and Wiesel [58]. Given an input image, it is obtained by first splitting the image in a high-frequency part and a low-frequency part and then by sequentially applying bandpass oriented filters and downsampling to the low-frequency image. This process results in a sequence of images having different sizes, known as a pyramid. With the exception of two special images in the pyramid, each image corresponds to a certain scale and orientation. The two remaining images are referred to as respectively the high-frequency residual and the low-frequency residual. A fundamental property of the steerable pyramid decomposition is that it can be inverted i.e. the input image can be recovered from its decomposition. For instance, this property is essential in the Heeger & Bergen texture synthesis algorithm [17, 54].



(a) Low-pass filter (ϕ_{low}) (b) High-pass filter (ϕ_{high}) (c) Horizontal filter ($\phi_{\text{steer}}^{(4,0)}$)

(d) 1st diagonal filter ($\phi_{\text{steer}}^{(4,1)}$) (e) Vertical filter ($\phi_{\text{steer}}^{(4,2)}$) (f) 2nd diagonal filter ($\phi_{\text{steer}}^{(4,3)}$)

Figure 4.2: Functions defining the low-pass, high-pass and steered filters of the steerable pyramid in the Nyquist domain (with a number of orientations $Q = 4$). These images are actually the spectral samples corresponding to method 1 for $M = N = 512$ (multiplied by a factor 255 for visualization purposes).

The three building block types (low-pass, high-pass and steered filters) are defined by continuous functions on the Nyquist domain. All of them rely on the transformation between polar and cartesian coordinates given by the following formula:

$$\rho^{-1} : (r,\theta) \in \mathbb{R}^{+}\times]-\pi,\pi] \mapsto (r\cos\theta, r\sin\theta) \in \mathbb{R}^2. \tag{4.36}$$

Definition 4.16 (Low-pass filter)**.** *The low-pass filter (of the steerable pyramid) is defined by the function* $\phi_{low} = l \circ \rho : [-\pi,\pi]^2 \to \mathbb{R}$ *where*

$$l(r,\theta) = l(r) = \begin{cases} 1 & \text{if } r \leqslant \frac{\pi}{4}, \\ \cos(\frac{\pi}{2}\log_2(\frac{4r}{\pi})) & \text{if } \frac{\pi}{4} \leqslant r \leqslant \frac{\pi}{2}, \\ 0 & \text{if } r \geqslant \frac{\pi}{2}. \end{cases} \tag{4.37}$$

Since ϕ_{low} is null at the boundary of $[-\pi,\pi]^2$ the three methods are equivalent for this filter.

Definition 4.17 (High-pass filter)**.** *The high-pass filter (of the steerable pyramid) is defined by the function* $\phi_{high} = h \circ \rho : [-\pi,\pi]^2 \to \mathbb{R}$ *where*

$$h(r,\theta) = h(r) = \begin{cases} 0 & \text{if } r \leqslant \frac{\pi}{4}, \\ \cos(\frac{\pi}{2}\log_2(\frac{2r}{\pi})) & \text{if } \frac{\pi}{4} \leqslant r \leqslant \frac{\pi}{2}, \\ 1 & \text{if } r \geqslant \frac{\pi}{2}. \end{cases} \tag{4.38}$$

Since ϕ_{high} is symmetric at the boundary of $[-\pi,\pi]^2$, method 1 and method 2 are equivalent for this filter.

Definition 4.18 (Steered filter)**.** *Let Q be the number of orientations of the steerable pyramid and $q \in \{0,\dots,Q-1\}$. The steered filter of orientation q is defined by the function $\phi_{steer}^{(Q,q)} = s_{Q,q} \circ \rho : [-\pi,\pi]^2 \to \mathbb{R}$ where*

$$s_{Q,q}(r,\theta) = s_{Q,q}(\theta) = \alpha_Q(\cos(\theta - \frac{\pi q}{Q})^{Q-1}\mathbb{1}_{|\theta-\frac{\pi q}{Q}|\leqslant\frac{\pi}{2}} + \cos(\theta - \frac{\pi(q-Q)}{Q})^{Q-1}\mathbb{1}_{|\theta-\frac{\pi(q-Q)}{Q}|\leqslant\frac{\pi}{2}}), \tag{4.39}$$

where $\alpha_Q = 2^{Q-1}\frac{(Q-1)!}{\sqrt{Q(2(Q-1))!}}$ is a normalization constant and $p!$ denotes the factorial of the non-negative integer p.

Let Q be the number of orientations of the steerable pyramid. We notice that for the horizontal filter $\phi_{\text{steer}}^{(Q,0)}$ the spectral samples in both conventions are the same i.e. method 1 and method 2 are equivalent.

The reconstruction process (detailed in [17]) is made possible by the relation

$$\phi_{\text{low}}^2 + \phi_{\text{high}}^2 = 1 \tag{4.40}$$

and the normalization constant α_Q which guarantees that $\sum(\phi_{\text{steer}}^{(Q,q)})^2 = 1$. The continuous functions that define the low-pass, high-pass and steered filters in the Nyquist domain for $Q = 4$ are represented in Figure 4.2.

4.4.2 Application of the Algorithms

We now apply the three filtering methods to an experimental set of four images and compare the results.

Experimental Set of Images. In order to explain how the set of experimental images is chosen we lead a simple study on the difference between filtered images obtained with two different methods.

Let $\underline{u}$ be a real-valued image and $\phi : [-\pi,\pi]^2 \to \mathbb{C}$ be a filter. Denote by $d(\underline{u},\phi)$ the maximum absolute difference between filtered images obtained with two different methods and define $\phi_{\max} = \sup_B |\phi|$ where B denotes the boundary of $[-\pi,\pi]^2$. The following definition allows us to obtain a simple upper-bound of $d(\underline{u},\phi)$.

Definition 4.19 (Boundary value)**.** *Let M and N be two positive integers and $\underline{u}$ be an image of size $M \times N$. The boundary value of $\underline{u}$ is denoted $Bv(\underline{u})$ and is defined by*

$$Bv(\underline{u}) = \sum_{(m,n)\in\Gamma_{M,N}} |\mathcal{F}_{M,N}(\underline{u})_{m,n}|\,. \tag{4.41}$$

The relative boundary value of $\underline{u}$ is denoted $Bv_{rel}(\underline{u})$ and is defined by

$$Bv_{rel}(\underline{u}) = \frac{Bv(\underline{u})}{\sum_{(m,n)\in\hat{\Omega}_{M,N}} |\mathcal{F}_{M,N}(\underline{u})_{m,n}|} \tag{4.42}$$

with the convention "$0/0 = 0$". It is the ratio between the boundary value and the total value $\frac{1}{MN}\sum_{(m,n)\in\hat{\Omega}_{M,N}} |\mathcal{F}_{M,N}(\underline{u})_{m,n}|$.

A straightforward computation shows that

$$d(\underline{u},\phi) \leq \phi_{\max} Bv(\underline{u}). \tag{4.43}$$

The difference between two filtered images obtained by different methods depends on the considered filter and on the frequency content of the input image and more precisely, as stated in

Equation (4.43), on its boundary value. This is why we propose to work with images with diverse frequency contents (from oversampled to textured). The four images used in the experiments are presented in Figure 4.3. The modulus of the DFT in logarithmic scale of an image $\underline{u}$ of size $M \times N$ denoted $M(\underline{u})$ is defined by

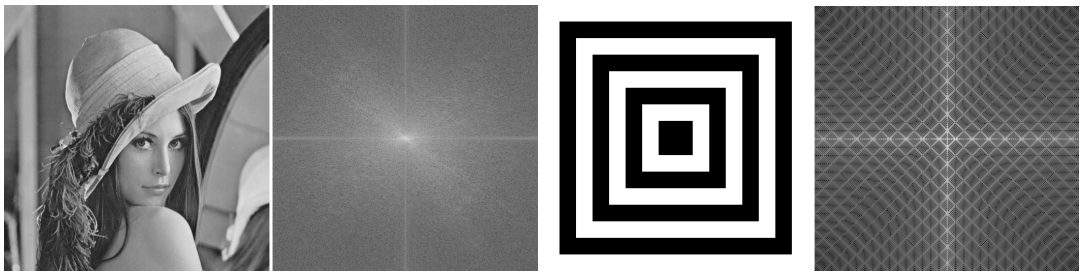
$$\forall (m,n) \in \hat{\Omega}_{M,N}, \quad M(\underline{u})_{m,n} = \log\left(1 + MN|\mathcal{F}_{M,N}(\underline{u})_{m,n}|\right). \tag{4.44}$$

For the visualization (in Figure 4.3), it is multiplied by the constant factor 15. We notice that the Dice image is almost oversampled (it has a little high-frequency content) whereas the Garden image is textured (it has a large amount of high-frequencies). The Garden and Dice images only have one even size. The Square image is a binary image that has two strong orientations and an important boundary value. The *Lenna* image is an intermediate example of natural image whose relative boundary value is higher than the Garden image one because it is a smaller image with two even sizes.

Comparison Procedure of Two Methods. Let us explain in detail how the comparison between the methods is made.

Let ϕ be a filter and $\underline{u}$ be an image of size $M \times N$. Let $(j, j') \in \{1,2,3\}^2$ and denote by $\underline{v}_j$ the filtered image obtained by method j. Let R be the range (maximum value minus minimum value) of $\mathcal{R}(\underline{v}_1)$[6]. Then a comparison of $\underline{v}_j$ and $\underline{v}_{j'}$ is done by considering the difference image $\Delta^{j,j'} = |\underline{v}_j - \underline{v}_{j'}|$ and the relative difference image[7] $\Delta^{j,j'}_{\text{rel}} = \Delta^{j,j'}/R$. Before visualization the

[6] It's an arbitrary choice.
[7] We use the convention "$0/0 = 0$".



(a) Lenna image (512×512): Bv $= 5.5$ (0.10%)

(b) Square image (256×256): Bv $= 7.4$ (0.21%)

(c) Dice image (704×469): Bv $= 0.68$ (0.028%)

(d) Garden image (704×469): Bv $= 10$ (0.068%)

Figure 4.3: Set of four images used in the experiments. For each image we plot the image (left) and the modulus of its DFT (right) in logarithmic scale (multiplied by a factor 15). The boundary value is noted *Bv* and the relative boundary value is written in percentage. We notice that the Dice image is almost oversampled (it has a few high frequencies amount) whereas the Garden image is textured (it has a large amount of high frequencies). Note that the Garden and Dice images only have one even size. The Square image is a binary image that has two strong orientations and an important boundary value. The *Lenna* image is an intermediate example of natural image whose relative boundary value is higher than the Garden image one because it is a smaller image with two even sizes.

relative difference image is multiplied by the constant factor 10^5 (which is the same whatever are the filter and the input image). We also compute the following values:

1. the maximum difference (maximum value of $\Delta^{j,j'}$ over $\Omega_{M,N}$) noted $d = d^{j,j'}(\underline{u},\phi)$ and the relative maximum difference (maximum value of $\Delta^{j,j'}_{\text{rel}}$ over $\Omega_{M,N}$) expressed in percentage,

2. the mean difference (mean value of $\Delta^{j,j'}$ over $\Omega_{M,N}$) noted $m_d = m_d^{j,j'}(\underline{u},\phi)$ and the relative mean difference (mean value of $\Delta^{j,j'}_{\text{rel}}$ over $\Omega_{M,N}$) expressed in percentage.

Note that the differences highly depend on the range of the filtered images, thus the relative values provide more significant information while comparing differences (over the experimental set of images for a given filter or over the filters for a given image).

Results.

We apply the comparison procedure to our experimental set of four images and the nine following filters:

- the *sinc* filter (Figure 4.4),
- the shift filter of parameter $(1/4, 1/4)$ (Figure 4.5 and Table 4.2),
- the derivative filter φ_x (Figure 4.6 and Table 4.3),
- the Laplacian filter L (Figure 4.7),
- the Gaussian filter of parameter $\sigma_0 = \frac{\sqrt{\log(2)}}{\pi}$ (Figure 4.8)[8],
- the low-pass filter (Figure 4.9),
- the high-pass filter (Figure 4.10),
- the horizontal filter $\phi^{(4,0)}_{\text{steer}}$ with $Q = 4$ orientations[9] (Figure 4.11),
- the first diagonal filter $\phi^{(4,1)}_{\text{steer}}$ with $Q = 4$ orientations (Figure 4.12 and Table 4.4).

When the three methods provide different filtered images we decompose the presentation of the results into a figure containing filtered images and relative difference images and a table with the maximum difference and mean difference values (and the corresponding relative values).

To display images we use the most common method. The brightness of a pixel is represented by its pixel value which is an integer from 0 to 255 (8-bits images). Let $\underline{u}$ be a real-valued image. The 8-bit image corresponding to $\underline{u}$ is the integral part of $\max(0, \min(255, \underline{u}))$. An affine transformation may have to be applied to filtered images (or their real parts) before obtaining 8-bit images in order to avoid saturation[10] and badly constrasted images. The affine transformation choice (which is arbitrary) depends on the filter and is detailed in the corresponding figure.

The difference is more important when the input image has a strong high-frequency content (Garden and Square images). For 8-bit images the typical range value is $R = 255$, therefore a maximum difference of one grey level corresponds approximately to a relative difference of $1/255 \simeq 0.39\%$. For the proposed set of nine filters and four images, the relative maximum difference values are lower than $5/255 \simeq 1.9\%$ and the relative mean difference values are lower than $3/255 \simeq 1.2\%$. It is hardly possible to distinguish with the naked eye a difference between the results of two methods. That is why we only display the real part of the filtered image in

[8] The value σ_0 is chosen so that $g_{\sigma_0}(\pi,\pi) = 1/2$.

[9] It is the default value used in the Heeger & Bergen texture synthesis algorithm.

[10] For instance saturation occurs for zero-mean images.

the complex convention (method 1). The higher difference values are obtained for the derivative filter and the Laplacian filter. Indeed, it comes from the fact that $|D_x|$ and $|L|$ reach their maxima on the boundary of $[-\pi,\pi]^2$.

4.4.3 Which Method should I use?

The three methods are equally licit and in general none should be preferred to the others. The results are numerically different but indistinguishable in practice. The maximum difference between the methods of at most 5 grey levels[11] (in a scale from 0 to 255) leads to no visible distinction but determining if it represents a meaningful difference depends on the application. For instance if the goal is to blur an image the user may attach less importance to the differences than when trying to denoise it.

It is left to the user to choose the method. The following remarks may be helpful for making the decision:

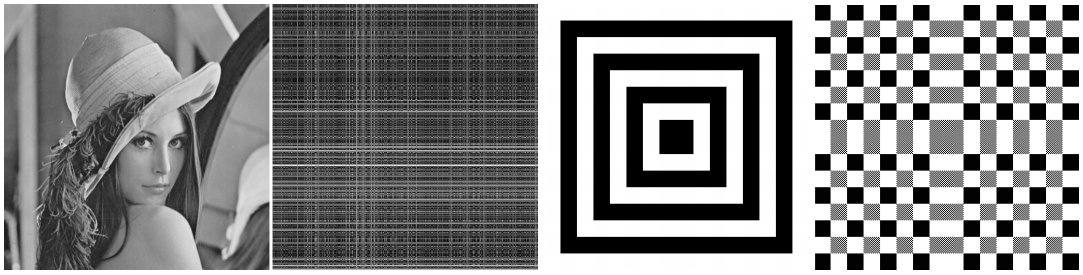
- Method 1 is the easiest one to implement. For simple applications it can be used in first place.
- Suppose the filter ϕ has Hermitian symmetry. Then the filtered images obtained by methods 2 and 3 are real-valued.
- Method 3 destroys high-frequency information but there is no ambiguity in the interpolation. It should not be used in analysis/synthesis scheme based on a reconstruction property.

We recall that when the sizes of the image are odd the three methods are equivalent. If the user has the possibility to modify the size of its input images odd sizes should be considered. However generally the images sizes cannot be changed and sometimes are powers of two (e.g. in a pyramidal decomposition).

		Lenna	Square	Dice	Garden
1 vs 2	d	0.77 (0.34%)	1.6 (0.41%)	0.13 (0.067%)	1.4 (0.43%)
	m_d	0.12 (0.052%)	0.55 (0.14%)	0.020 (0.010%)	0.30 (0.094%)
1 vs 3	d	1.1 (0.48%)	2.2 (0.58%)	0.18 (0.093%)	2.0 (0.61%)
	m_d	0.17 (0.075%)	0.77 (0.20%)	0.029 (0.015%)	0.42 (0.13%)
2 vs 3	d	0.77 (0.34%)	1.6 (0.41%)	0.13 (0.067%)	1.4 (0.43%)
	m_d	0.12 (0.053%)	0.55 (0.14%)	0.020 (0.010%)	0.30 (0.094%)

Table 4.2: Maximum difference d and mean difference m_d for the shift filter of parameter $(1/4,1/4)$. Relative values are in brackets.

[11] Actually for most of the examples it is less than 1 grey level.



(a) Lenna: 1.0 (0.46%) and 0.17 (0.078%)

(b) Square: 2.0 (0.78%) and 0.75 (0.30%)

(c) Dice: 0.17 (0.092%) and 0.029 (0.016%)

(d) Garden: 2.3 (0.92%) and 0.43 (0.17%)

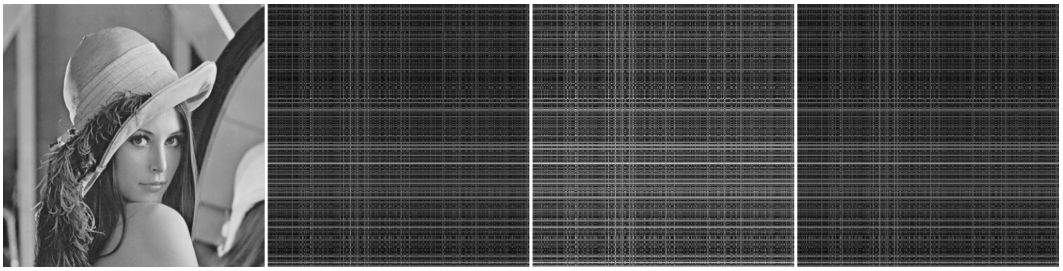
Figure 4.4: Comparison of the filtering methods for the *sinc* filter. The filtered image $\underline{v}_1$ is the input. For each image we display $\underline{v}_3$ and the relative difference image $\Delta_{\text{rel}}^{1,3}$ (right). The numbers below the images refer respectively to the maximum difference d and the mean difference m_d (with the relative values in brackets).

		Lenna	Square	Dice	Garden
1 vs 2	d	1.6 (0.67%)	3.1 (0.84%)	0.53 (0.32%)	7.2 (1.4%)
	m_d	0.44 (0.19%)	1.6 (0.42%)	0.091 (0.055%)	1.3 (0.26%)
1 vs 3	d	1.8 (0.75%)	3.2 (0.86%)	0.53 (0.32%)	7.2 (1.4%)
	m_d	0.50 (0.21%)	1.7 (0.45%)	0.091 (0.055%)	1.3 (0.26%)
2 vs 3	d	0.80 (0.33%)	0.72 (0.19%)	0	0
	m_d	0.16 (0.067%)	0.20 (0.053%)	0	0

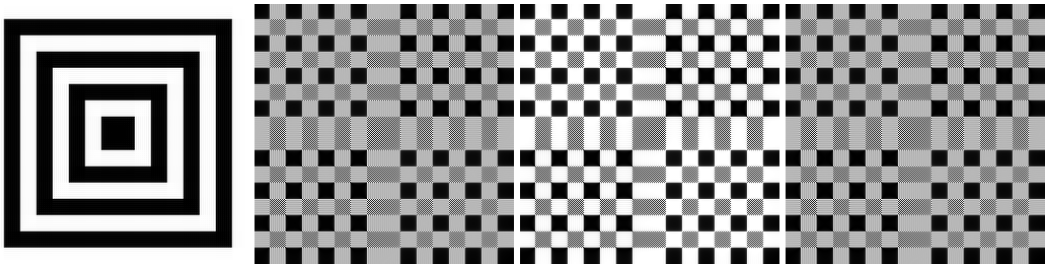
Table 4.3: Maximum difference d and mean difference m_d for the derivative filter φ_x. Relative values are in brackets.

		Lenna	Square	Dice	Garden
1 vs 2	d	0.29 (0.14%)	0.36 (0.10%)	0.058 (0.054%)	0.64 (0.19%)
	m_d	0.046 (0.022%)	0.056 (0.016%)	0.0080 (0.0074%)	0.12 (0.036%)
1 vs 3	d	0.38 (0.18%)	0.79 (0.23%)	0.076 (0.071%)	0.83 (0.25%)
	m_d	0.086 (0.040%)	0.27 (0.077%)	0.015 (0.014%)	0.22 (0.064%)
2 vs 3	d	0.37 (0.17%)	0.70 (0.20%)	0.060 (0.056%)	0.81 (0.24%)
	m_d	0.062 (0.029%)	0.24 (0.070%)	0.011 (0.0097%)	0.16 (0.047%)

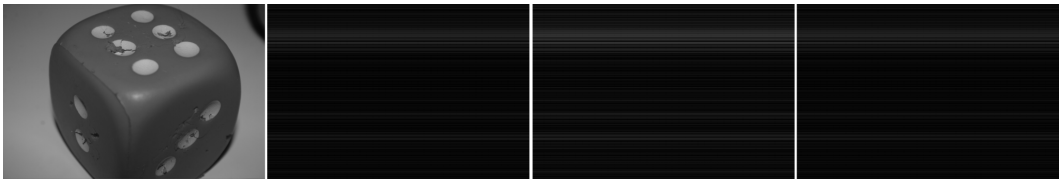
Table 4.4: Maximum difference d and mean difference m_d for the first diagonal filter $L_{\text{steer}}^{(4,1)}$ with $Q = 4$ orientations. Relative values are in brackets.



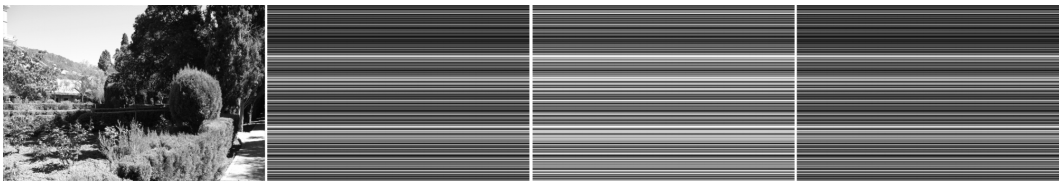
(a) Lenna



(b) Square



(c) Dice

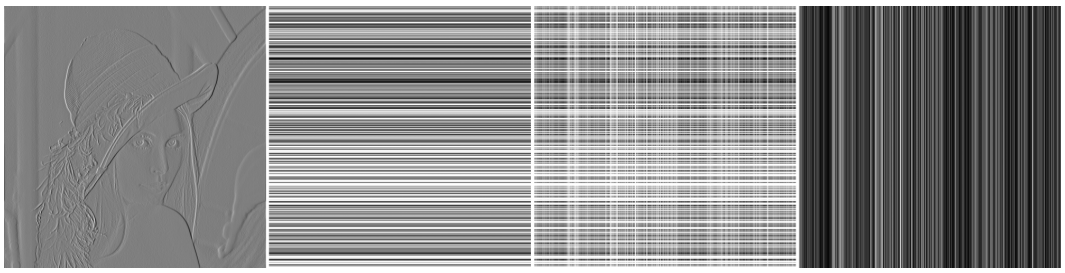


(d) Garden

Figure 4.5: Comparison of the filtering methods for the shift filter of parameter $(1/4, 1/4)$. From left to right we display for each input image: the real part of $\underline{v}_1$ and the relative difference images $\Delta_{\text{rel}}^{1,2}$, $\Delta_{\text{rel}}^{1,3}$ and $\Delta_{\text{rel}}^{2,3}$.

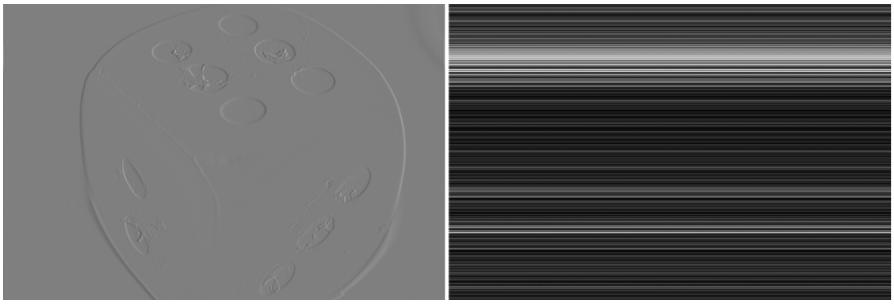
		Lenna	Square	Dice	Garden
Sinc	d	1.013115	1.996120	0.169090	2.299789
	m_d	0.170532	0.752396	0.028972	0.425675
Highpass	d	1.013123	1.996094	0.169037	2.299735
	m_d	0.170532	0.752396	0.028971	0.425676

Table 4.5: Maximal difference d and mean difference m_d between method 1 and method 3 for the *sinc* and high-pass filters. In theory the differences should be the same. In practice it is not the case because of numerical error while computing the DFT and inverse DFT. 10^{-7}

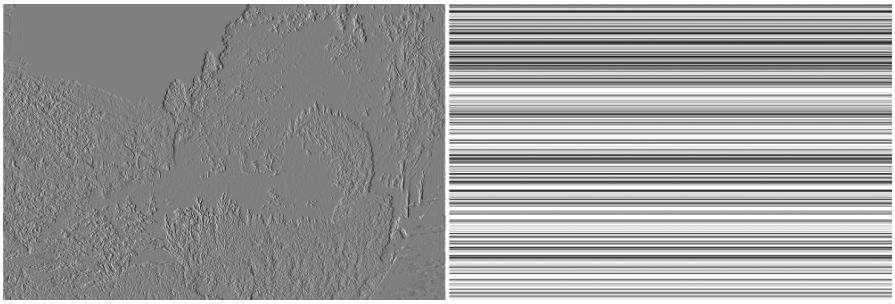


(a) Lenna

(b) Square

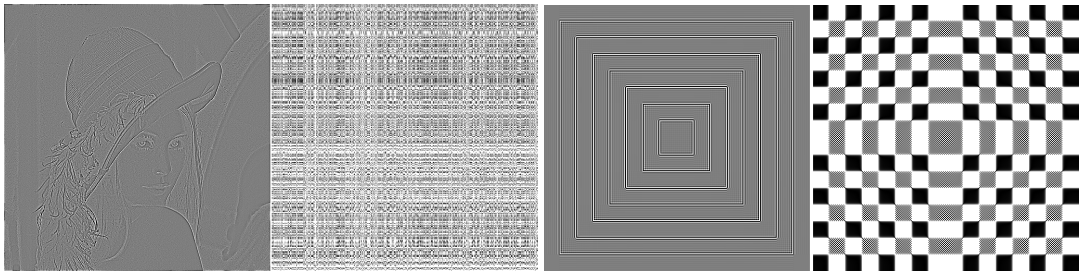


(c) Dice

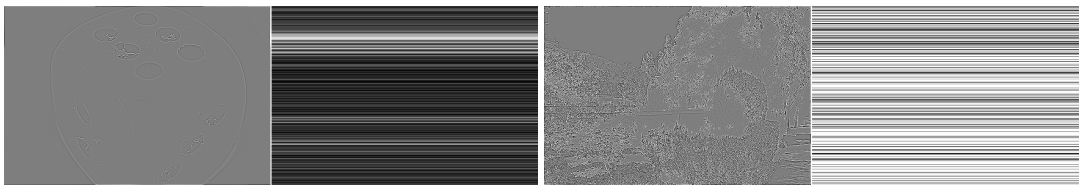


(d) Garden

Figure 4.6: Comparison of the filtering methods for the derivative filter φ_x. From left to right we display for the Lenna and Square images: the real part of $\underline{v}_1$ and the relative difference images $\Delta_{\text{rel}}^{1,2}$, $\Delta_{\text{rel}}^{1,3}$ and $\Delta_{\text{rel}}^{2,3}$. As method 2 and 3 are equivalent for images of the size of the Dice and Garden image, we only display the real part of $\underline{v}_1$ and $\Delta_{\text{rel}}^{1,2}$ for these images. For visualization purposes we change the mean of $\underline{v}_1$ to 127 (instead of 0). The derivative filter highlights the strong horizontal variations of the input image. We notice a lot of ringing effect (particularly for the Square image).

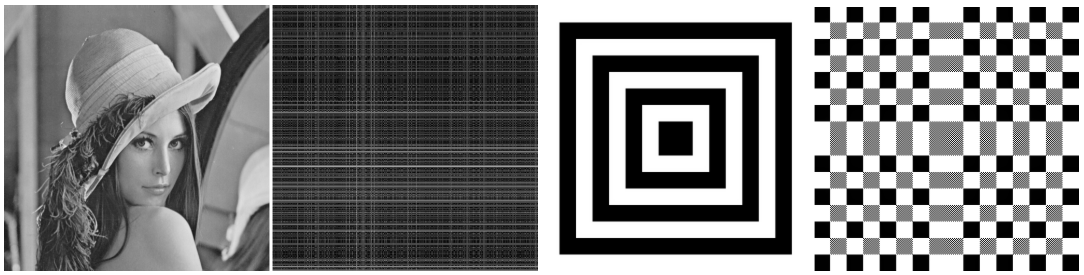


(a) Lenna: 12 (1.6%) and 2.0 (0.29%) (b) Square: 23 (1.4%) and 7.7 (0.46%)



(c) Dice: 2.1 (0.38%) and 0.35 (0.064%) (d) Garden: 26 (1.6%) and 5.2 (0.32%)

Figure 4.7: Comparison of the filtering methods for the Laplacian filter. For each image we display the real part of $\underline{v}_1$ and the relative difference image $\Delta_{\text{rel}}^{1,3}$ (right). The numbers below the images refer respectively to the maximum difference d and the mean difference m_d (with the relative values in brackets). For visualization purposes we change the mean of $\underline{v}_1$ to 127 (instead of 0). The Laplacian filter highlights the high-frequencies of the input image. We notice a lot of ringing effect (particularly for the Square image).



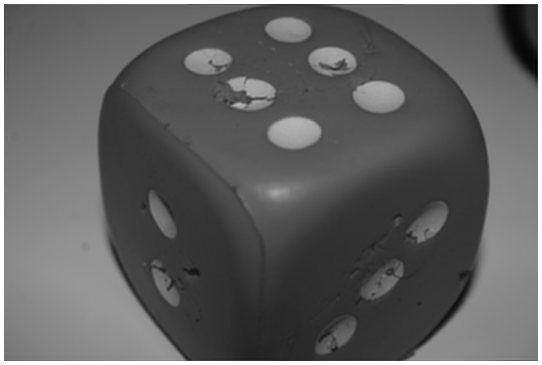
(a) Lenna: 0.68 (0.31%) and 0.11 (0.052%) (b) Square: 1.4 (0.54%) and 0.53 (0.20%)

(c) Dice: 0.11 (0.063%) and 0.019 (0.011%) (d) Garden: 1.6 (0.61%) and 0.28 (0.11%)

Figure 4.8: Comparison of the filtering methods for the Gaussian filter of standard deviation $\sigma_0 = \frac{\sqrt{\log(2)}}{\pi}$. For each image we display the real part of $\underline{v}_1$ and the relative difference image $\Delta_{\text{rel}}^{1,3}$ (right). The numbers below the images refer respectively to the maximum difference d and the mean difference m_d (with the relative values in brackets). With a higher value of standard deviation the filtered images are more blurry and the differences between the methods decrease.

(a) Lenna

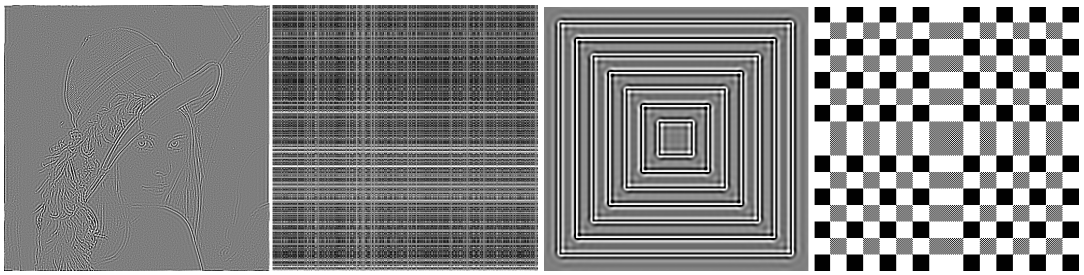
(b) Square



(c) Dice

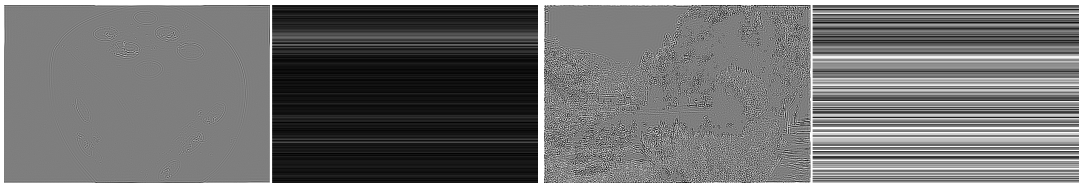
(d) Garden

Figure 4.9: Filtering of the images by the low-pass filter. The three methods are equivalent. We notice that as expected the filtered images are blurred versions of the input.



(a) Lenna: 1.0 (0.61%) and 0.17 (0.10%)

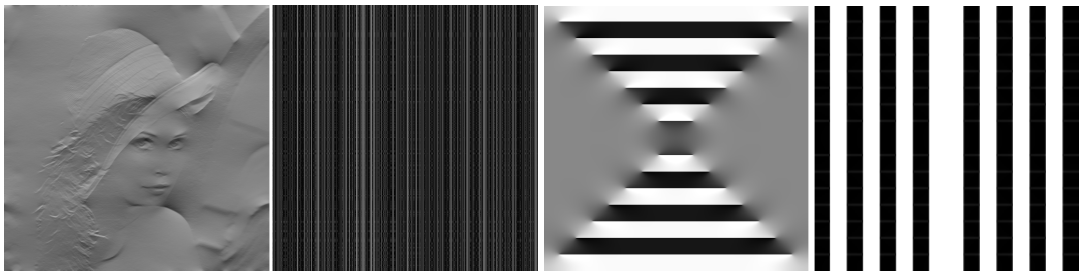
(b) Square: 2.0 (0.69%) and 0.75 (0.26%)



(c) Dice: 0.17 (0.16%) and 0.030 (0.027%)

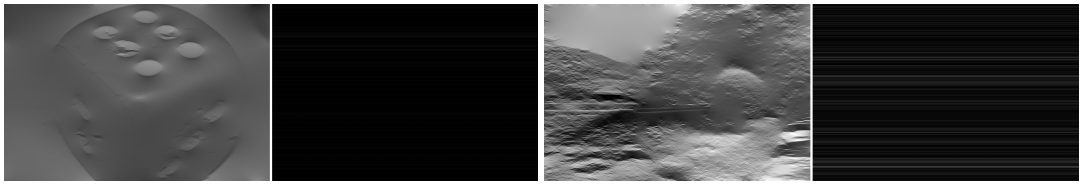
(d) Garden: 2.3 (0.84%) and 0.43 (0.16%)

Figure 4.10: Comparison of the filtering methods for the high-pass filter. For each image we display the real part of $\underline{v}_1$ and the relative difference image $\Delta_{\text{rel}}^{1,3}$ (right). The numbers below the images refer respectively to the maximum difference d and the mean difference m_d (with the relative values in brackets). For visualization purposes we multiply by a factor 5 the image $\underline{v}_1$ and then set the mean to 127 (instead of 0). The high-pass filter highlights the strong variations of the input image.



(a) Lenna: 0.37 (0.22%) and 0.071 (0.042%)

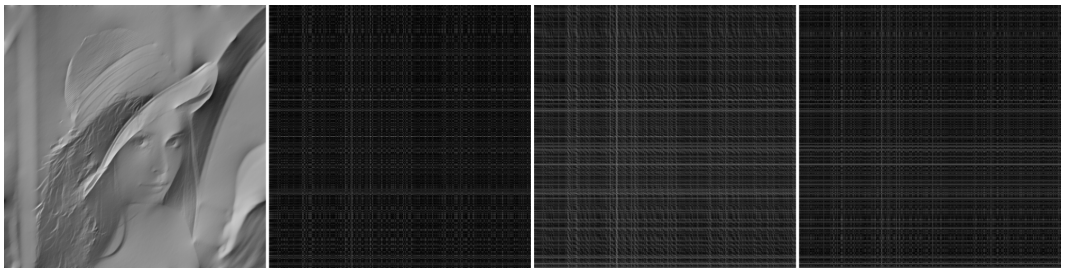
(b) Square: 0.95 (0.35%) and 0.45 (0.16%)



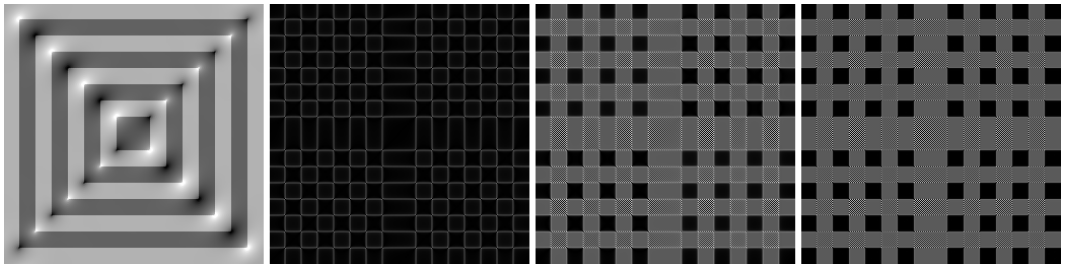
(c) Dice: 0.022 (0.016%) and 0.0031 (0.0023%)

(d) Garden: 0.22 (0.070%) and 0.047 (0.015%)

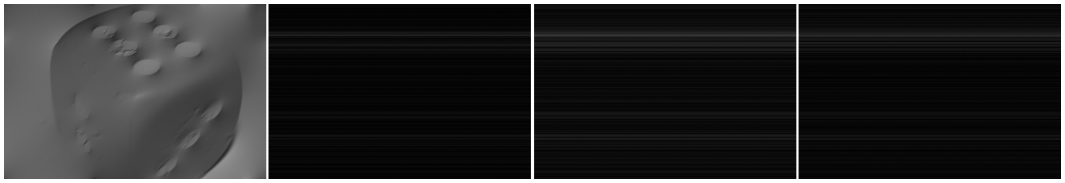
Figure 4.11: Comparison of the filtering methods for the horizontal filter $\phi_{\text{steer}}^{(4,0)}$ with $Q = 4$ orientations. For each image we display the real part of $\underline{v}_1$ and the relative difference image $\Delta_{\text{rel}}^{1,3}$ (right). The numbers below the images refer respectively to the maximum d and the mean difference m_d (with the relative values in brackets). The horizontal filter highlights the strong horizontal variations of the input image.



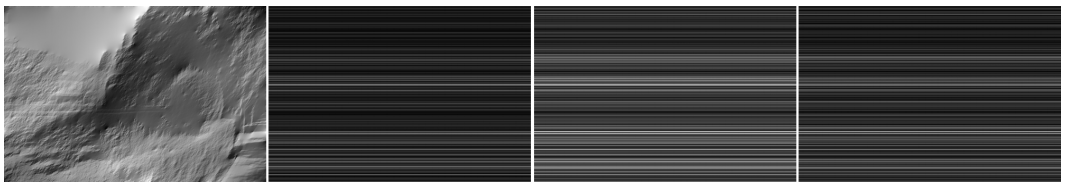
(a) Lenna



(b) Square



(c) Dice



(d) Garden

Figure 4.12: Comparison of the filtering methods for the first diagonal filter $\phi_{\text{steer}}^{(4,1)}$ with $Q = 4$ orientations. From left to right we display for each input image: the real part of $\underline{v}_1$ and the relative difference images $\Delta_{\text{rel}}^{1,2}$, $\Delta_{\text{rel}}^{1,3}$ and $\Delta_{\text{rel}}^{2,3}$. As expected the first diagonal filter highlights the strong variations in the first diagonal orientation.

4.5 Conclusion

In this chapter we gave a clear interpretation, as a continuous convolution, of the filtering of a discrete real-valued image by a filter specified by a continuous function in the Nyquist domain $[-\pi,\pi]^2$. The filtering is interpreted as the convolution of a distribution, standing for the filter, with a trigonometric polynomial interpolator of the image. Two plausible interpolations and two choices of distribution are considered. They lead to three equally licit algorithms. All of them can be seen as method variants of the same standard filtering algorithm in the DFT domain. By an application to several fundamental filters we have shown that the method differences, which come from the boundary DFT coefficients, is not visible to the naked eye in practice. For 8-bit images, the difference order of magnitude is in general of one grey level but may be higher for filters with large values on the boundary of the Nyquist domain. In general none method should be preferred to the others and the choice depends on the desired properties of the filtering. For instance, in the implementation of the Heeger-Bergen pyramid-based texture synthesis algorithm provided in [17], method 1 is used in order to guarantee the exact reconstruction property.

Chapter 5

B-spline Interpolation: Theory and Practice

Abstract

This chapter is taken from [14]. B-spline interpolation is a widely used non band-limited alternative to Shannon-Whittaker interpolation. In this chapter, we explain how the B-spline interpolation of signals and images can be efficiently performed by linear filtering. Based on the seminal two-step method proposed by Unser et al. in 1991, we propose two slightly different prefiltering algorithms whose precisions are proven to be controlled thanks to a rigorous boundary handling. The first algorithm is general and works for any boundary extension while the second is applicable under specific assumptions. This chapter contains all the information, theoretical and practical, required to perform efficiently B-spline interpolation for any order and any boundary extension. We describe precisely how to evaluate the kernel and to compute the B-spline interpolator parameters. We show experimentally that increasing the order improves the interpolation quality. As a fundamental application we also provide an implementation[1] of homographic transformation of images using B-spline interpolation.

Contents

[1] http://www.ipol.im/pub/art/2018/221/

5.1 Introduction

In this chapter, n denotes a non-negative integer. A non band-limited signal representation that has been widely used since the 1990s is the spline representation. A spline of degree n is a continuous piece-wise polynomial function of degree n of a real variable with derivatives up to order $n-1$. This representation has the advantage of being equally justifiable on a theoretical and practical basis [126]. It can model the physical process of drawing a smooth curve and it is well adapted to signal processing thanks to its optimality properties. It is handy to consider the B-spline representation [106] where the continuous underlying signal is expressed as the convolution between the B-spline kernel, that is compactly supported, and the parameters of the representation, namely the B-spline coefficients. One of the strongest arguments in favor of the B-spline interpolation is that it approaches the Shannon-Whittaker interpolation as the order increases [5].

The determination of the B-spline coefficients is done in a prefiltering step which, in general, can be done by solving a band diagonal system of linear equations [57, 70]. For uniformly spaced data points, Unser et al. proposed in [129] an efficient and stable algorithm based on linear filtering that works for any order. More details, in particular regarding the determination of the interpolator parameters, were provided by the authors in later publications [127, 128].

As for Shannon's sampling theory, the spline representation is designed for infinite signals. Finite signals need to be extended in an arbitrary way in order to apply the interpolation scheme. This issue is commonly referred to as the boundary handling. The influence of the unknown exterior data decays with the distance to the boundary of the known data points [109] and is therefore often neglected. However in some applications this decay may be too slow or it can be relevant to consider a particular extension. Unser's algorithm is described with a symmetrical boundary handling which is a classical but restrictive choice. Additional boundary extensions are available in the implementation provided by [46]. The problem that arises in this prefiltering algorithm is that the boundary handling involves infinite sums that are approximated by truncation. Because it uses a recursive structure, the B-spline coefficients are computed *a priori* with an uncontrolled error.

In this chapter we provide all the information, theoretical and practical, required to perform B-spline interpolation for any order and any boundary extension. We propose two slightly different prefiltering algorithms based on Unser's algorithm but with additional computations that take into account the boundary extension. The computational errors are proven, theoretically and experimentally, to be controlled (up to dimension two) thanks to a correct boundary handling. The computational cost increases slowly with the desired precision (which can be set to the single precision in most of the applications). The first algorithm is general and works for any boundary extension while the second is applicable under specific assumptions. In addition, we describe precisely how to evaluate the B-spline kernel and to compute the B-spline interpolator parameters. We show experimentally that increasing the order improves the interpolation quality. As a fundamental application we also provide an implementation of homographic transformation of images using B-spline interpolation.

This chapter is organized as follows: Section 5.2 presents the B-spline interpolation theory of a discrete infinite unidimensional signal as a two-step method. Section 5.3 describes two prefiltering algorithms for a finite unidimensional signal whose computational errors are controlled thanks to a proper boundary handling. The extension in higher dimension, with a particular focus to dimension two, is done in Section 5.4. Section 5.5 proves that the prefiltering error is controlled in dimension one and two. Details concerning practical computations and the provided numerical implementation are given in Section 5.6. Section 5.7 presents some experiments.

5.2 B-spline Interpolation of a Discrete Signal

In this section we present the B-spline interpolation of an *infinite* discrete unidimensional signal as a two-step interpolation method. We detail how the first step, i.e., the prefiltering step, can be decomposed into a cascade of exponential filters, themselves separated into two complementary causal and anti-causal components. We also explain how to evalute by closed form formulas the B-spline values at arbitrary points.

5.2.1 B-spline Interpolation Theory

A spline of degree $n \geq 1$ is a continuous piece-wise polynomial function of degree n of a real variable with continuous derivatives up to order $n-1$. The junction abscissas between successive polynomials are called the *knots*.

Definition 5.1. *The normalized B-spline function of order n, noted* $\beta^{(n)}$*, is defined recursively by*

$$\beta^{(0)}(x) = \begin{cases} 1, & -\frac{1}{2} < x < \frac{1}{2} \\ \frac{1}{2}, & x = \pm\frac{1}{2} \\ 0, & \textit{otherwise} \end{cases} \qquad \textit{and for } n \geq 0, \quad \beta^{(n+1)} = \beta^{(n)} * \beta^{(0)} \tag{5.1}$$

where the symbol $*$ *denotes the convolution operator.*

The normalized B-spline function of order n is even, compactly supported in $\mathrm{supp}(\beta^{(n)}) = [-\frac{n+1}{2}, \frac{n+1}{2}]$ and non-negative. Moreover, it has $n+2$ equally spaced knots $k \in \mathbb{Z} \cap \mathrm{supp}(\beta^{(n)})$ when n is odd and $n+2$ equally spaced knots $k \in \left(\mathbb{Z} + \frac{1}{2}\right) \cap \mathrm{supp}(\beta^{(n)})$ when n is even. An explicit formula for $\beta^{(n)}$ can be derived from its recursive definition [126] and will be used in Section 5.6.2. Figure 5.1 displays $\beta^{(n)}$ for $n = 0, \ldots, 3$.

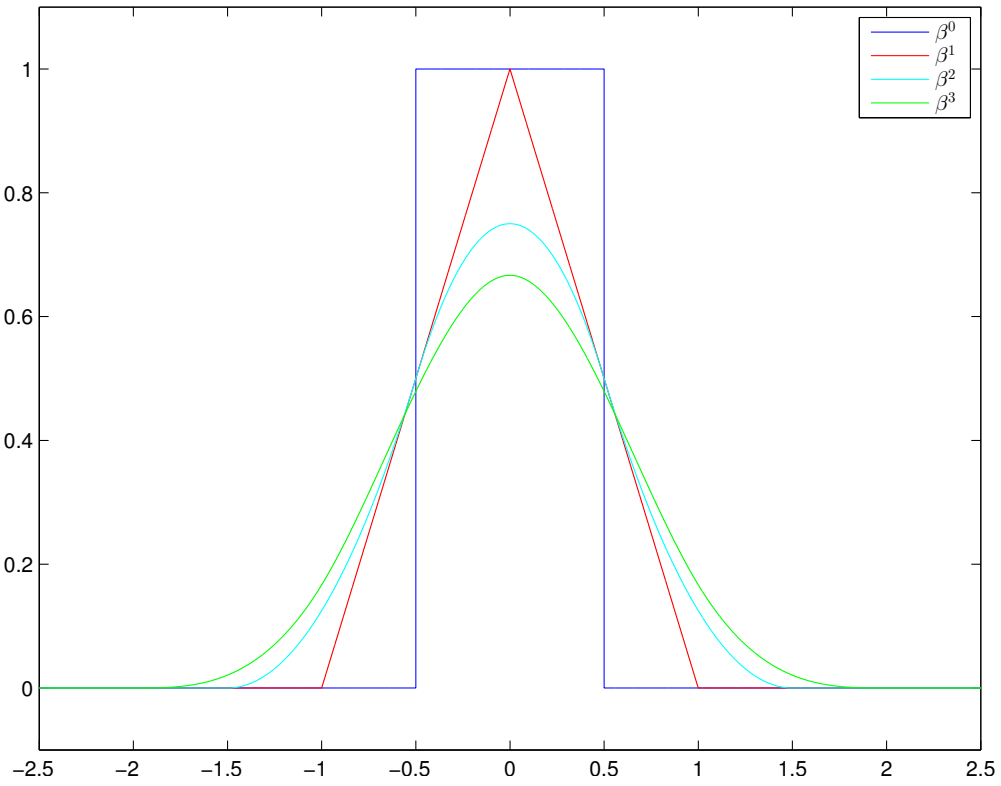


Figure 5.1: Normalized B-spline functions $\beta^{(n)}$ for $0 \leq n \leq 3$.

The normalized B-spline functions are the basic atoms for constructing splines in the case of equally spaced knots. Let φ be a spline of degree n with equally spaced knots belonging to $\mathbb{Z}$ for n odd and to $\mathbb{Z}+\frac{1}{2}$ for n even. Then, as proved by Schoenberg [106], φ can be uniquely represented as the weighted sum of shifted normalized B-splines of order n i.e.

$$\varphi(x) = \sum_{i\in\mathbb{Z}} c_i \beta^{(n)}(x-i) \tag{5.2}$$

where the weights $c = (c_i)_{i\in\mathbb{Z}} \in \mathbb{R}^{\mathbb{Z}}$ are called the B-spline coefficients. The spline φ is uniquely characterized by its B-spline coefficients or equivalently by its samples $(\varphi(k))_{k\in\mathbb{Z}} \in \mathbb{R}^{\mathbb{Z}}$ at integer locations. Thus, the interpolation of a discrete signal $f \in \mathbb{R}^{\mathbb{Z}}$ by a spline of degree n, namely the B-spline interpolation of order n, can be defined as follows.

Definition 5.2 (B-spline interpolation)**.** *The B-spline interpolate of order n of a discrete signal $f \in \mathbb{R}^{\mathbb{Z}}$ is the spline $\varphi^{(n)}$ of degree n defined for $x \in \mathbb{R}$ by*

$$\varphi^{(n)}(x) = \sum_{i\in\mathbb{Z}} c_i \beta^{(n)}(x-i) \tag{5.3}$$

where the B-spline coefficients $c = (c_i)_{i\in\mathbb{Z}}$ are uniquely characterized by the interpolating condition

$$\varphi^{(n)}(k) = f_k, \quad \forall k \in \mathbb{Z}. \tag{5.4}$$

Given a signal f and a real x, computing the right hand side of (5.3) requires two evaluations: the signal $c = (c_i)_{i\in\mathbb{Z}}$ and $\beta^{(n)}(x-i)$. This explains the two steps involved in the computation:

- Step 1 (prefiltering or direct B-spline transform) provides a B-spline representation of the signal. The computation of the B-spline coefficients c is done in the prefiltering step detailed in Section 5.2.2. Except in the simplest cases, $n = 0$ or $n = 1$, in which case $\beta^{(n)}(k-i) = \delta_i(k)$ and so $c_i = f_i$, the determination of the coefficients c_i from f so as to satisfy (5.4) is not straightforward.

- Step 2 (indirect B-spline transform [129]) reconstructs the signal values from the B-spline representation. Given the Dirac comb of B-spline coefficients $\mathbf{c} = \sum_{i\in\mathbb{Z}} c_i \delta_i$ the value of $\varphi^{(n)}(x)$ in (5.3) is computed at any location $x \in \mathbb{R}$ as a convolution of $\mathbf{c}$ with the finite signal $\beta^{(n)}(x-.)$, whose computation is explained in Section 5.6.2.

The B-spline interpolation can be expressed also as a direct interpolation

$$\varphi^{(n)}(x) = \sum_{i\in\mathbb{Z}} f_i \eta^{(n)}(x-i) \quad (x \in \mathbb{R}) \tag{5.5}$$

where $\eta^{(n)}$ is called the cardinal spline function of order n [126, 127]. In [5] it is proven that the cardinal spline Fourier transform approaches the ideal filter (i.e., the Fourier transform of the cardinal sine) when n goes to infinity. This result makes the link between the Shannon's sampling theory and the B-spline interpolation. For $n = 0$ and $n = 1$, we have $c_i = f_i$ and $\eta^{(n)} = \beta^{(n)}$ so that the direct and indirect methods coincide. These interpolations correspond respectively to the nearest neighbor and linear interpolation methods [46]. For $n \geq 2$, $\eta^{(n)}$ is no longer compactly supported so that the two-step representation becomes more efficient. See [123] for more information about two-step interpolation methods.

5.2.2 Prefiltering Step

The prefiltering step (or direct B-spline transform) consists in computing the B-spline coefficients c introduced in (5.3).

Expression as a Discrete Convolution. The interpolating condition (5.4) can be written as

$$\varphi^{(n)}(k) = \sum_{i\in\mathbb{Z}} c_i \beta^{(n)}(k-i) = f_k, \quad \forall k \in \mathbb{Z}. \tag{5.6}$$

Let us define $b^{(n)} \in \mathbb{R}^{\mathbb{Z}}$ by $b_i^{(n)} = \beta^{(n)}(i)$ for $i \in \mathbb{Z}$. Then, we recognize in (5.6) the convolution equation

$$c * b^{(n)} = f. \tag{5.7}$$

Solving for c in the latest equation is efficiently done thanks to transfer functions:

Definition 5.3 (Z-transform (or transfer function) [60])**.** *The Z-transform of a discrete signal $y \in \mathbb{R}^{\mathbb{Z}}$ is the formal power series $\mathcal{Z}[y]$ defined by*

$$\mathcal{Z}[y](z) = \sum_{i\in\mathbb{Z}} y_i z^{-i}. \tag{5.8}$$

The set of complex points for which the Z-transform summation converges is called the region of convergence (ROC). In particular, if the ROC contains the unit circle then the Z-transform can be inverted and characterizes the signal.

The advantage of introducing this transform is its property of transforming a convolution into a simple multiplication, so that inverting a convolution amounts to a division in the Z-domain:

Proposition 5.1 (Convolution property [60])**.** *Let $v \in \mathbb{R}^{\mathbb{Z}}$ and $w \in \mathbb{R}^{\mathbb{Z}}$ be two discrete signals. Then,*

$$\mathcal{Z}[v * w] = \mathcal{Z}[v]\mathcal{Z}[w] \tag{5.9}$$

on the intersection of their regions of convergence.

The finite discrete convolution kernel $b^{(n)}$ is entirely characterized by its Z-transform $B^{(n)} = \mathcal{Z}[b^{(n)}]$ (whose ROC is $\mathbb{C}\setminus\{0\}$ since $\beta^{(n)}$ is compactly supported). As $B^{(n)}$ has no zeros on the unit circle [5], the inverse operator $(b^{(n)})^{-1}$ exists and is uniquely defined by its Z-transform, noted $(B^{(n)})^{-1}$, which verifies (formally)

$$(B^{(n)})^{-1} = \mathcal{Z}[(b^{(n)})^{-1}] = \frac{1}{\mathcal{Z}[b^{(n)}]} = \frac{1}{B^{(n)}} \tag{5.10}$$

and whose ROC contains the full unit circle. Finally, the prefiltering step boils down to the discrete convolution

$$c = (b^{(n)})^{-1} * f. \tag{5.11}$$

Decomposition into Elementary Filters. The filter $(b^{(n)})^{-1}$ can be decomposed into elementary causal and anti-causal filters. Using the fact that $\beta^{(n)}$ is even and supported in $\left[-\frac{n+1}{2}, \frac{n+1}{2}\right]$, denoting $\tilde{n} = \lfloor \frac{n}{2} \rfloor$, we can write

$$B^{(n)}(z) = b_0^{(n)} + \sum_{i=1}^{\tilde{n}} b_i^{(n)}(z^i + z^{-i}). \tag{5.12}$$

Schoenberg proved that $B^{(n)}$ has only negative (simple) zeros [105, lemma 8]. By the symmetry $B^{(n)}(z) = B^{(n)}(z^{-1})$ for $z \neq 0$, these zeros can be grouped in reciprocal pairs (α, α^{-1}). Denoting by $R^{(n)}$ the set of zeros of $B^{(n)}$ yields

$$R^{(n)} = \bigcup_{i=1}^{\tilde{n}} \{z_i, z_i^{-1}\}, \quad \text{with } -1 < z_1 < \cdots < z_{\tilde{n}} < 0. \tag{5.13}$$

The z_i's are called the poles of the B-spline interpolation. Their practical computation is dealt with in Section 5.6.1. As $z^{\tilde{n}}B^{(n)}(z)$ is a polynomial whose roots are the elements of $R^{(n)}$, $B^{(n)}$ can be rewritten for $z \neq 0$ as

$$B^{(n)}(z) = b_{\tilde{n}}^{(n)} z^{-\tilde{n}} \prod_{i=1}^{\tilde{n}} (z - z_i)(z - z_i^{-1}), \tag{5.14}$$

which gives for $z \notin R^{(n)} \cup \{0\}$,

$$(B^{(n)})^{-1}(z) = \gamma^{(n)} \prod_{i=1}^{\tilde{n}} H(z; z_i) \tag{5.15}$$

where

$$\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}} \tag{5.16}$$

and

$$H(z; z_i) = \frac{-z_i}{(1 - z_i z^{-1})(1 - z_i z)}. \tag{5.17}$$

Let $-1 < \alpha < 0$, α playing the role of one z_i. Denote by $k^{(\alpha)} \in \mathbb{R}^{\mathbb{Z}}$ the causal filter and by $l^{(\alpha)} \in \mathbb{R}^{\mathbb{Z}}$ the anti-causal filter defined for $i \in \mathbb{Z}$ by

$$k_i^{(\alpha)} = \begin{cases} 0 & i < 0 \\ \alpha^i & i \geq 0 \end{cases} \quad \text{and} \quad l_i^{(\alpha)} = \begin{cases} 0 & i > 0 \\ \alpha^{-i} & i \leq 0. \end{cases} \tag{5.18}$$

In terms of Z-transform we have for $|z| > |\alpha|$,

$$\mathcal{Z}[k^{(\alpha)}](z) = \sum_{i=0}^{\infty} \alpha^i z^{-i} = \frac{1}{1 - \alpha z^{-1}} \tag{5.19}$$

and for $|z| < |\alpha^{-1}|$,

$$\mathcal{Z}[l^{(\alpha)}](z) = \sum_{i=-\infty}^{0} \alpha^{-i} z^{-i} = \frac{1}{1 - \alpha z}. \tag{5.20}$$

Set $h^{(\alpha)} = -\alpha l^{(\alpha)} * k^{(\alpha)}$. The filter $h^{(\alpha)}$ is called exponential filter because it is shown, using for instance (5.30), that for $j \in \mathbb{Z}$,

$$h_j^{(\alpha)} = \frac{\alpha}{\alpha^2 - 1} \alpha^{|j|}. \tag{5.21}$$

Define the domain $D_\alpha = \{z \in \mathbb{C}, |\alpha| < |z| < |\alpha^{-1}|\}$. Applying Proposition 5.1 to $h^{(\alpha)}$ and combining (5.19), (5.20) and (5.17), we get the following equality, valid on D_α:

$$\mathcal{Z}[h^{(\alpha)}] = -\alpha \mathcal{Z}[k^{(\alpha)}] \mathcal{Z}[l^{(\alpha)}] = H(.; \alpha). \tag{5.22}$$

Since $z_1 < \cdots < z_{\tilde{n}} < 0$ we have $D_{z_1} \subset \cdots \subset D_{z_{\tilde{n}}}$. Thus with Proposition 5.1 and (5.22) we get for $z \in D_{z_1}$,

$$\mathcal{Z}\left[\gamma^{(n)} h^{(z_{\tilde{n}})} * \cdots * h^{(z_1)}\right](z) = \gamma^{(n)} \prod_{i=1}^{\tilde{n}} H(z; z_i) = (B^{(n)})^{-1}(z). \tag{5.23}$$

Since D_{z_1} contains the unit circle, this yields

$$(b^{(n)})^{-1} = \gamma^{(n)} h^{(z_{\tilde{n}})} * \cdots * h^{(z_1)} \tag{5.24}$$

and provides a new expression for c,

$$c = \gamma^{(n)} h^{(z_{\tilde{n}})} * \cdots * h^{(z_1)} * f. \tag{5.25}$$

To simplify, define recursively the signal $c^{(i)} \in \mathbb{R}^{\mathbb{Z}}$ for $i \in \{0,\dots,\tilde{n}\}$ by

$$c^{(0)} = f \quad \text{and for } i \geq 1, \quad c^{(i)} = h^{(z_i)} * c^{(i-1)}. \tag{5.26}$$

We have $c = \gamma^{(n)} c^{(\tilde{n})}$. Thus the computation of the prefiltering step can be decomposed[2] into $\tilde{n}$ successive filtering steps with exponential filters that can themselves be separated into two complementary causal and anti-causal components. The corresponding algorithm for prefiltering an infinite discrete signal is presented in Algorithm 5.1. This is a theoretical algorithm that cannot be used in practice because it requires an infinite input signal. Turning this algorithm into a practical one, that is, applicable to a finite signal, is the subject of Section 5.3.

Algorithm 5.1: Theoretical prefiltering of an infinite signal

Input : A discrete infinite signal $f = (f_i)_{i\in\mathbb{Z}}$ and the B-spline interpolation order n
Output: The B-spline coefficients $c = (c_i)_{i\in\mathbb{Z}}$ of f

1 Compute the poles $(z_i, z_i^{-1})_{1\leq i\leq \tilde{n}}$ and the normalization coefficient $\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}$ (Section 5.6.1)
2 Define $c^{(0)} = f$
3 **for** $i = 1$ **to** $\tilde{n}$ **do**
4 Compute $c^{(i)} = h^{(z_i)} * c^{(i-1)}$ where $h^{(z_i)}$ is given by (5.21)
5 Normalization: $c = \gamma^{(n)} c^{(\tilde{n})}$

5.3 B-spline Interpolation of a Finite Signal

In practice the signal $\underline{f}$ to be interpolated is finite and discrete, i.e., $\underline{f} = (\underline{f}_i)_{0\leq i\leq K-1}$ for a given positive integer K. There exists an infinite number of coefficients $(c_i)_{i\in\mathbb{Z}}$ satisfying the interpolating condition (5.6) for $0 \leq k \leq K-1$. To insure uniqueness, an arbitrary extension of $\underline{f}$ outside $\{0,\dots,K-1\}$ is necessary. B-spline interpolation theory can then be applied to the extended signal $f \in \mathbb{R}^{\mathbb{Z}}$. To simplify the notations in the following, no distinction will be made between the signal $\underline{f}$ and its extension f when there is no ambiguity.

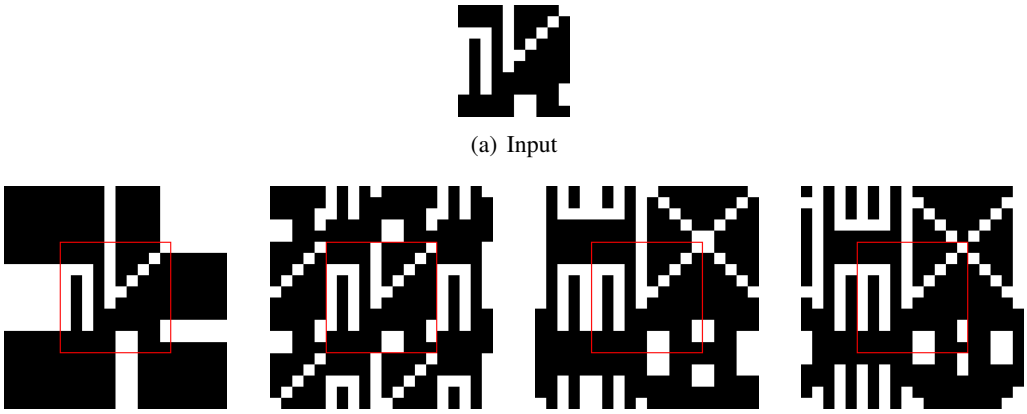
Extension on a finite domain. Let $x \in [0, K-1]$. To compute the interpolated value $\varphi^{(n)}(x)$, the indirect B-spline transform in (5.3) only requires the c_i's for $i \in \left(x + \left]-\frac{n+1}{2}, \frac{n+1}{2}\right[\right) \cap \mathbb{Z} \subset \{-\tilde{n},\dots,K-1+\tilde{n}\}$. In addition, even though the value f_k for $k \in \mathbb{Z}$ contributes to every B-spline coefficient c_i, this contribution vanishes when $|k-i|$ tends to infinity. As presented in the following, to compute $\varphi^{(n)}(x)$ with a relative precision ε it is sufficient to extend the signal to a finite domain $\{-L^{(n,\varepsilon)},\dots,K-1+L^{(n,\varepsilon)}\}$ where $L^{(n,\varepsilon)}$ is a positive integer that only depends on the B-spline order n and the desired precision ε. The precision is relative to the values of the signal. A relative precision of ε means that the error committed is less than $\varepsilon \sup_{k\in\mathbb{Z}} |f_k| = \varepsilon \|f\|_\infty$. In practice the images are large enough so that $L^{(n,\varepsilon)} < K$ and it is possible to express the extension as a boundary condition around 0 and $K-1$. Otherwise the extension is obtained by iterating the boundary condition. The most classical boundary condition choices are summarized in Table 5.1 and represented in Figure 5.2.

Prefiltering computation using the extension. As presented in Algorithm 5.1, for computing the B-spline coefficients using the prefiltering decomposition given in (5.25), only the first exponential filter $h^{(z_1)}$ is applied directly to $f = c^{(0)}$. Therefore, for $i \geq 1$ the intermediate filtered signals $c^{(i)}$ are known *a priori* only where they are computed. Considering this, the two following approaches are proposed in order to perform the prefiltering.

[2] In general the same decomposition principle is applicable to linear interpolation methods whose kernel is symmetrical and compactly supported [123].

Extension	Signal *abcde*
Constant	*aaa*|*abcde*|*eee*
Half-symmetric	*cba*|*abcde*|*edc*
Whole-symmetric	*dcb*|*abcde*|*dcb*
Periodic	*cde*|*abcde*|*abc*

Table 5.1: Classical boundary extensions of the signal *abcde* by $L = 3$ values.



(a) Input

(b) Constant (c) Periodic (d) Half-symmetric (e) Whole-symmetric

Figure 5.2: Classical extensions by $L = 5$ values of a binary image of size 10×10. The boundary of the original image is colored in red.

- **Approach 1:** The intermediate filtered signals $c^{(i)}$ are computed in a larger domain than $\{-\tilde{n}, \dots, K-1+\tilde{n}\}$. This works with any extension.
- **Approach 2:** The extension, expressed as a boundary condition, is chosen so that it is transmitted after the application of each exponential filter $h^{(z_i)}$. The intermediate filtered signals $c^{(i)}$ (and the B-spline coefficients c) verify the same boundary condition and only need to be computed in $\{0, \dots, K-1\}$.

In the rest of this section, we first detail the general method for computing an exponential filter application on a finite domain. Then we propose two algorithms, corresponding to both above-mentioned approaches, for computing the B-spline coefficients of a finite signal with a given precision. Finally we present the simple algorithm for performing the indirect B-spline transform, i.e., for evaluating the interpolated values.

5.3.1 Application of the Exponential Filters

Let $s \in \mathbb{R}^{\mathbb{Z}}$ be an infinite discrete signal. Let $-1 < \alpha < 0$ and $L_{\text{ini}} < L_{\text{end}}$. The application of the exponential filter $h^{(\alpha)} = -\alpha l^{(\alpha)} * k^{(\alpha)}$ to the signal s is computed in the domain of interest $\{L_{\text{ini}}, \dots, L_{\text{end}}\}$ as follows.

Causal filtering. To simplify the notation we set $s^{(\alpha)} = k^{(\alpha)} * s$ so that $h^{(\alpha)} * s = -\alpha l^{(\alpha)} * s^{(\alpha)}$. Given the initialization

$$s^{(\alpha)}_{L_{\text{ini}}} = \left(k^{(\alpha)} * s\right)_{L_{\text{ini}}} = \sum_{i=0}^{+\infty} \alpha^i s_{L_{\text{ini}}-i}, \tag{5.27}$$

the application of the causal filter $k^{(\alpha)}$ to s can be computed recursively from $i = L_{\text{ini}} + 1$ to $i = L_{\text{end}}$ according to the recursion formula

$$s^{(\alpha)}_i = s_i + \alpha s^{(\alpha)}_{i-1}. \tag{5.28}$$

Anti-causal filtering initialization. With a simple partial fraction decomposition we can rewrite $H(z;\alpha)$, the Z-transform of $h^{(\alpha)}$ introduced in (5.17), as

$$H(z;\alpha) = \frac{\alpha}{\alpha^2-1}\left(\frac{1}{1-\alpha z^{-1}} + \frac{1}{1-\alpha z} - 1\right). \tag{5.29}$$

Thus $h^{(\alpha)}$ can also be written as

$$h^{(\alpha)} = \frac{\alpha}{\alpha^2-1}\left(k^{(\alpha)} + l^{(\alpha)} - \delta_0\right) \tag{5.30}$$

and we have

$$h^{(\alpha)} * s = \frac{\alpha}{\alpha^2-1}\left(k^{(\alpha)} * s + l^{(\alpha)} * s - s\right). \tag{5.31}$$

This last formula provides an expression for the initialization of the (renormalized) anti-causal filtering,

$$\left(h^{(\alpha)} * s\right)_{L_{\text{end}}} = \frac{\alpha}{\alpha^2-1}\left(s^{(\alpha)}_{L_{\text{end}}} + \left(l^{(\alpha)} * s\right)_{L_{\text{end}}} - s_{L_{\text{end}}}\right) \tag{5.32}$$

where

$$\left(l^{(\alpha)} * s\right)_{L_{\text{end}}} = \sum_{i=0}^{+\infty} \alpha^i s_{L_{\text{end}}+i}. \tag{5.33}$$

Anti-causal filtering. The renormalized anti-causal filtering is computed recursively from $i = L_{\text{end}} - 1$ to $i = L_{\text{ini}}$ according to the following formula,

$$\begin{aligned}
\left(h^{(\alpha)} * s\right)_i &= \left(-\alpha l^{(\alpha)} * s^{(\alpha)}\right)_i && (5.34)\\
&= -\alpha\left(s^{(\alpha)}_i + \alpha\left(l^{(\alpha)} * s^{(\alpha)}\right)_{i+1}\right) && (5.35)\\
&= \alpha\left(\left(h^{(\alpha)} * s\right)_{i+1} - s^{(\alpha)}_i\right). && (5.36)
\end{aligned}$$

Approximation of the initialization values. The two infinite sums in (5.27) and (5.33) cannot be computed numerically. Let N be a non-negative integer. The initialization values are approximated by truncating the sums at index N so that

$$\left(k^{(\alpha)} * s\right)_{L_{\text{ini}}} \simeq \sum_{i=0}^{N} \alpha^i s_{L_{\text{ini}}-i} \tag{5.37}$$

and

$$\left(l^{(\alpha)} * s\right)_{L_{\text{end}}} \simeq \sum_{i=0}^{N} \alpha^i s_{L_{\text{end}}+i}. \tag{5.38}$$

Algorithm. The general method for computing the application of the exponential filter $h^{(\alpha)}$ to a discrete signal in a finite domain is summarized in Algorithm 5.2. Note that is is sufficient to know the signal in the domain $\{L_{\text{ini}} - N, \ldots, L_{\text{end}} + N\}$. It consists in $6(N+1) + 4(L_{\text{end}} - L_{\text{ini}})$ operations.

Algorithm 5.2: Application of the exponential filter $h^{(\alpha)}$ to a discrete signal

Input : A pole $-1 < \alpha < 0$, a range of indices $L_{\text{ini}} < L_{\text{end}}$, a truncation index N and a discrete signal s (whose values are known in $\{L_{\text{ini}} - N, \dots, L_{\text{end}} + N\}$)

Output: The filtered signal $h^{(\alpha)} * s$ at indices $\{L_{\text{ini}}, \dots, L_{\text{end}}\}$

Compute $s^{(\alpha)}_{L_{\text{ini}}}$ using (5.37)
for $i = L_{ini} + 1$ **to** L_{end} **do**
    Compute $s^{(\alpha)}_i$ using (5.28)
Compute $\left(l^{(\alpha)} * s\right)_{L_{\text{end}}}$ using (5.38)
Compute $\left(h^{(\alpha)} * s\right)_{L_{\text{end}}}$ using (5.32)
for $i = L_{end} - 1$ **to** L_{ini} **do**
    Compute $\left(h^{(\alpha)} * s\right)_i$ using (5.36)

5.3.2 Prefiltering of a Finite Signal

Define $(\mu_j)_{1 \le j \le \tilde{n}}$ by

$$\begin{cases} \mu_1 = 0 \\ \mu_k = \left(1 + \frac{1}{\log|z_k| \sum_{i=1}^{k-1} \frac{1}{\log|z_i|}}\right)^{-1}, \quad 2 \le k \le \tilde{n}. \end{cases} \tag{5.39}$$

Let $\varepsilon > 0$. Define for $1 \le i \le \tilde{n}$,

$$N^{(i,\varepsilon)} = \left\lfloor \frac{\log\left(\varepsilon \rho^{(n)} (1 - z_i)(1 - \mu_i) \prod_{j=i+1}^{\tilde{n}} \mu_j\right)}{\log|z_i|} \right\rfloor + 1, \tag{5.40}$$

where

$$\rho^{(n)} = \left(\prod_{j=1}^{\tilde{n}} \frac{1 + z_j}{1 - z_j}\right)^2. \tag{5.41}$$

We propose two algorithms for computing the B-spline coefficients of a finite signal with precision ε. In both cases they are computed by successively applying the exponential filters to the intermediate filtered signals using Algorithm 5.2 with the truncation indices $(N^{(i,\varepsilon)})_{1 \le i \le \tilde{n}}$. The difference between both algorithms lies in the computation domains. The choice for the truncation indices guarantees a precision ε for $n \le 16$, as stated in the next theorem.

Theorem 5.1. *Assume $n \le 16$. Let $\varepsilon > 0$ and f be a finite signal of length at least 4 (arbitrarily extended to $\mathbb{Z}$). The computation of the B-spline coefficients of f using Algorithm 5.3 or Algorithm 5.4 with the truncation indices $(N^{(i,\varepsilon)})_{1 \le i \le \tilde{n}}$ has a precision of ε, i.e., the error committed is less than $\varepsilon \|f\|_\infty$.*

Proof. See Section 5.5.1. □

Notice that $N^{(i,\varepsilon)} = O(\log \varepsilon)$, which shows that the truncation indices remain moderate even for high precision specification ε. However, a problematic factor is $\log|z_i|$ in the denominator, which increases the indices when z_i is close to -1. This is not unexpected, since a root with modulus close to 1 involves a slowly decaying exponential filter.

Approach 1: Extended Domain. The first approach for computing the prefiltering coefficients with precision ε consists in computing the intermediate filtered signals $c^{(i)}$ in a larger domain than $\{-\tilde{n}, \dots, K - 1 + \tilde{n}\}$. For $0 \le j \le \tilde{n}$ define

$$L_j^{(n,\varepsilon)} = \tilde{n} + \sum_{i=j+1}^{\tilde{n}} N^{(i,\varepsilon)} \tag{5.42}$$

which can be computed recursively using

$$\begin{cases} L_{\tilde{n}}^{(n,\varepsilon)} & = \tilde{n}, \\ L_{j}^{(n,\varepsilon)} & = L_{j+1}^{(n,\varepsilon)} + N^{(j+1,\varepsilon)}, \quad j = \tilde{n}-1 \text{ to } 0. \end{cases} \tag{5.43}$$

The input signal $f = c^{(0)}$ is first extended to $\{-L_0^{(n,\varepsilon)}, \dots, K-1+L_0^{(n,\varepsilon)}\}$. Then, for $i = 1$ to $\tilde{n}$, $c^{(i)}$ is computed in $\{-L_i^{(n,\varepsilon)}, \dots, K-1+L_i^{(n,\varepsilon)}\}$ from the values of $c^{(i-1)}$ in $\{-L_{i-1}^{(n,\varepsilon)}, \dots, K-1+L_{i-1}^{(n,\varepsilon)}\}$ using Algorithm 5.2 with truncation index $N^{(i,\varepsilon)}$. The prefiltering algorithm on a larger domain is presented in Algorithm 5.3.

Approach 2: Transmitted Boundary Condition. The second approach for computing the prefiltering coefficients with precision ε consists in extending the input signal with a boundary condition that is transmitted after the application of the exponential filters. Assume that the $c^{(i)}$ for $0 \le i \le \tilde{n}$ share the same boundary condition. Then, $c^{(i)}$ can be computed at any index from the values of $c^{(i-1)}$ in $\{0, \dots, K-1\}$ using Algorithm 5.2 (with truncation index $N^{(i,\varepsilon)}$). Indeed, we notice that the initialization values in (5.37) and (5.38) only depend on the values of $c^{(i-1)}$ in $\{0, \dots, K-1\}$. The prefiltering algorithm with a transmitted boundary condition is described in Algorithm 5.4.

Note that the initialization values in (5.27) and (5.33) can be expressed as weighted sums of the $c_k^{(i-1)}$ for $k \in \{0, \dots, K-1\}$ where the weights depend on k, the poles z_i and the boundary condition. For specific boundary conditions the weights, and therefore the initialization values, can be exactly computed. However these explicit expressions are not used in the following because they generally involve sums of K terms, i.e., more computation.

Particular cases. Among the four classical boundary extensions presented in Table 5.1, the periodic, half-symmetric and whole-symmetric boundary conditions are transmitted after the application of an exponential filter. For the periodic extension the filtered signal by any filter always remains periodic. For the half-symmetric and whole-symmetric extensions it is a consequence of the symmetry of the exponential filters. However, this property is not satisfied for the constant extension, so that Algorithm 5.4 is not applicable in this case.

For the three boundary conditions that are transmitted, the anti-causal initialization value in (5.32) can be computed without using (5.38) as follows. Let s be a finite signal of length K and $-1 < \alpha < 0$. We recall that for the two symmetrical extensions the signal is extended to $\mathbb{Z}$ by periodization.

- **Periodic extension:** As s is a K-periodic signal then $s^{(\alpha)} = k^{(\alpha)} * s$ is also K-periodic. Noting N the truncation index, we can write

$$\left(l^{(\alpha)} * s^{(\alpha)}\right)_{K-1} \simeq \sum_{i=0}^{N} s_{K-1+i}^{(\alpha)} \alpha^i \tag{5.44}$$

$$= s_{K-1}^{(\alpha)} + \alpha \sum_{i=0}^{N-1} s_{K+i}^{(\alpha)} \alpha^i \tag{5.45}$$

$$= s_{K-1}^{(\alpha)} + \alpha \sum_{i=0}^{N-1} s_{i}^{(\alpha)} \alpha^i. \tag{5.46}$$

It yields,

$$\left(h^{(\alpha)} * s\right)_{K-1} \simeq -\alpha \left(s_{K-1}^{(\alpha)} + \alpha \sum_{i=0}^{N-1} s_{i}^{(\alpha)} \alpha^i \right). \tag{5.47}$$

Algorithm 5.3: Prefiltering algorithm on a larger domain

Input : A finite discrete signal f of length K, a boundary extension, a B-spline interpolation order $n \leq 16$ and a precision ε

Output: The B-spline coefficients c of order n at indices $\{-\tilde{n}, \dots, K-1+\tilde{n}\}$ with precision ε

Precomputations

Compute the poles $(z_i, z_i^{-1})_{1 \leq i \leq \tilde{n}}$ and the normalization coefficient $\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}$ (Section 5.6.1).

for $1 \leq i \leq \tilde{n}$ **do**

  Compute the truncation index $N^{(i,\varepsilon)}$ using (5.40)

Define $L_{\tilde{n}}^{(n,\varepsilon)} = \tilde{n}$

for $j = \tilde{n} - 1$ **to** 0 **do**

  Compute $L_j^{(n,\varepsilon)} = L_{j+1}^{(n,\varepsilon)} + N^{(j+1,\varepsilon)}$

Prefiltering the signal:

Set $c_k^{(0)} = f_k$ for $k \in \{-L_0^{(n,\varepsilon)}, \dots, K-1+L_0^{(n,\varepsilon)}\}$ using the boundary extension

for $i = 1$ **to** $\tilde{n}$ **do**

  Compute $c_k^{(i)} = \left(h^{(z_i)} * c^{(i-1)}\right)_k$ for $k \in \{-L_i^{(n,\varepsilon)}, \dots, K-1+L_i^{(n,\varepsilon)}\}$ using Algorithm 5.2 with truncation index $N^{(i,\varepsilon)}$

Renormalize $c = \gamma^{(n)} c^{(\tilde{n})}$

Algorithm 5.4: Prefiltering algorithm with a transmitted boundary condition

Input : A finite discrete signal f of length K, a boundary condition that is transmitted, a B-spline interpolation order $n \leq 16$ and a precision ε

Output: The B-spline coefficients c of order n at indices $\{0, \dots, K-1\}$ with precision ε

Precomputations:

Compute the poles $(z_i, z_i^{-1})_{1 \leq i \leq \tilde{n}}$ and the normalization coefficient $\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}$ (Section 5.6.1).

for $1 \leq i \leq \tilde{n}$ **do**

  Compute the truncation index $N^{(i,\varepsilon)}$ using (5.40)

Prefiltering of the signal:

Set $c_k^{(0)} = f_k$ for $k \in \{0, \dots, K-1\}$

for $i = 1$ **to** $\tilde{n}$ **do**

  Compute $c_k^{(i-1)}$ for $k \in \{-N^{(i,\varepsilon)}, \dots, -1\} \cup \{K, \dots, K-1+N^{(i,\varepsilon)}\}$ using the boundary condition

  Compute $c_k^{(i)} = \left(h^{(z_i)} * c^{(i-1)}\right)_k$ for $k \in \{0, \dots, K-1\}$ using Algorithm 5.2 with truncation index $N^{(i,\varepsilon)}$

Renormalize $c = \gamma^{(n)} c^{(\tilde{n})}$

• **Half-symmetric extension:** For $i \geq 0$ we have $s_{K+i} = s_{K-1-i}$ so that we can write

$$\left(l^{(\alpha)} * s\right)_{K-1} = s_{K-1} + \alpha \sum_{i=0}^{+\infty} s_{K+i}\alpha^i \tag{5.48}$$

$$= s_{K-1} + \alpha \sum_{i=0}^{+\infty} s_{K-1-i}\alpha^i \tag{5.49}$$

$$= s_{K-1} + \alpha s_{K-1}^{(\alpha)}. \tag{5.50}$$

It yields with (5.32),

$$\left(h^{(\alpha)} * s\right)_{K-1} = \frac{\alpha}{\alpha - 1} s_{K-1}^{(\alpha)} = \frac{\alpha}{\alpha - 1}\left(k^{(\alpha)} * s\right)_{K-1}. \tag{5.51}$$

• **Whole-symmetric extension:** For $i \geq 0$ we have $s_{K+i} = s_{K-2-i}$ so that we can write

$$\left(l^{(\alpha)} * s\right)_{K-1} = s_{K-1} + \alpha \sum_{i=0}^{+\infty} s_{K+i}\alpha^i \tag{5.52}$$

$$= s_{K-1} + \alpha \sum_{i=0}^{+\infty} s_{K-2-i}\alpha^i \tag{5.53}$$

$$= s_{K-1} + \alpha s_{K-2}^{(\alpha)}. \tag{5.54}$$

It yields with (5.32),

$$\left(h^{(\alpha)} * s\right)_{K-1} = \frac{\alpha}{\alpha^2 - 1}\left(s_{K-1}^{(\alpha)} + \alpha s_{K-2}^{(\alpha)}\right) = \frac{\alpha}{\alpha^2 - 1}\left(\left(k^{(\alpha)} * s\right)_{K-1} + \alpha\left(k^{(\alpha)} * s\right)_{K-2}\right). \tag{5.55}$$

In practice, when one of these three boundary conditions is used in Algorithm 5.4, a slightly different version of Algorithm 5.2 is called in Line 8. The boundary extension is added to the input list and the computation of $\left(h^{(\alpha)} * s\right)_{K-1}$ (see Line 4 and Line 5) is done using (5.47), (5.51) or (5.55) according to the extension case. Note that the anti-causal initialization value admits an exact expression for the two symmetric extensions.

5.3.3 Indirect B-spline Transform: Computation of the Interpolated Value

The indirect B-spline transform reconstructs the signal values from the B-spline representation. Given the B-spline coefficients c in $\{-\tilde{n}, \ldots, K-1+\tilde{n}\}$, the interpolated value $\varphi^{(n)}(x)$ can be computed with precision ε, using (5.3), as the convolution of $\sum_{i=-\tilde{n}}^{K-1+\tilde{n}} c_i \delta_i$ with the compactly supported function $\beta^{(n)}$. The computation of the indirect B-spline transform at location $x \in [0, K-1]$ is presented in Algorithm 5.5. Assume the B-spline coefficients are computed with precision ε using either Algorithm 5.3 or Algorithm 5.4 then $\varphi^{(n)}(x)$ is also computed with precision ε because $\sum_{k\in\mathbb{Z}} \beta^{(n)}(x-k) = 1$. We recall that in Algorithm 5.4 the B-spline coefficients are known in $\{-\tilde{n}, \ldots, -1\} \cup \{K, \ldots, K-1+\tilde{n}\}$ thanks to the boundary condition.

The computation of (5.3) at a point x involves only a finite sum. Noting $r = (n+1)/2$ the radius of the support of $\beta^{(n)}$, we have for $n \geq 1$

$$\begin{aligned} \beta(x-k) > 0 &\Leftrightarrow -r < x-k < r \\ &\Rightarrow \lceil x-r \rceil \leq k \leq \lfloor x+r \rfloor. \end{aligned}$$

In general, we have $\lfloor x+r \rfloor - \lceil x-r \rceil + 1 = 2r = n+1$ except when $x \pm r$ is an integer. In the latter case, we have only $2r-2$ terms since $\beta^{(n)}(x-.)$ vanishes at the bounds $k = x \pm r$. For $n = 0$, $\beta^{(n)}$ is special because it does not vanish at the bounds of its support, and in that case up to 2 terms are involved. In any case, we have at most $\max(1, n) + 1$ terms.

Algorithm 5.5: Indirect B-spline transform

Input : A finite discrete signal f of length K, a B-spline interpolation order n, the corresponding B-spline coefficients c in $\{-\tilde{n},\dots,K-1+\tilde{n}\}$ and $x \in [0,K-1]$

Output: The interpolated value $\varphi^{(n)}(x)$

$x_0 \leftarrow \lceil x-(n+1)/2 \rceil$

Initialize $\varphi^{(n)}(x) \leftarrow 0$

for $k=0$ **to** $\max(n,1)$ **do**

  Update $\varphi^{(n)}(x) \leftarrow \varphi^{(n)}(x) + c_k \beta^{(n)}(x-(x_0+k))$

5.4 Extension to Higher Dimensions

The one-dimensional B-spline interpolation theory (Section 5.2) and practical algorithms (Section 5.3) are easily extended to higher dimensions by using tensor-product basis functions. The multi-dimensional prefiltering is performed by applying the unidimensional prefiltering successively along each dimension. The indirect B-spline transform is efficiently computed as the convolution with a separable and compactly supported function.

Note d the dimension. The Normalized B-spline function of order n is defined in dimension d as follows.

Definition 5.4. *The Normalized B-spline function of order n and dimension d, noted* $\beta^{(n,d)}$*, is defined for* $x=(x_1,\dots,x_d) \in \mathbb{R}^d$ *by*

$$\beta^{(n,d)}(x) = \prod_{j=1}^{d} \beta^{(n)}(x_j). \tag{5.56}$$

Then, the B-spline interpolate of order n of a discrete signal $f \in \mathbb{R}^{\mathbb{Z}^d}$ can be naturally defined as follows.

Definition 5.5. *The B-spline interpolate of order n of a discrete signal* $f \in \mathbb{R}^{\mathbb{Z}^d}$ *is the function* $\varphi^{(n)} : \mathbb{R}^d \mapsto \mathbb{R}$ *defined for* $x \in \mathbb{R}^d$ *by*

$$\varphi^{(n)}(x) = \sum_{i \in \mathbb{Z}^d} c_i \beta^{(n,d)}(x-i) \tag{5.57}$$

where the B-spline coefficients $c=(c_i)_{i\in\mathbb{Z}^d}$ *are uniquely defined by the interpolation condition*

$$\varphi^{(n)}(k) = f_k, \quad \forall k \in \mathbb{Z}^d. \tag{5.58}$$

Prefiltering Decomposition. The B-spline interpolation in dimension d is also a two-step interpolation method. The prefiltering step is decomposed as follows. Define $b^{(n,d)} = \prod_{j=1}^{d} b^{(n,d,j)}$ where for $1 \le j \le d$ and $k=(k_1,\dots,k_d) \in \mathbb{Z}^d$, $b_k^{(n,d,j)} = b_{k_j}$. Then, the interpolating condition is rewritten as

$$f = c * b^{(n,d)}. \tag{5.59}$$

Using the separability of $b^{(n,d)}$ and Z-transform based arguments as in Section 5.2.2, the prefiltering can be expressed as the filtering

$$c = \left(b^{(n,d)}\right)^{-1} * f \tag{5.60}$$

where for $k=(k_1,\dots,k_d) \in \mathbb{Z}^d$

$$\left(\left(b^{(n,d)}\right)^{-1}\right)_k = \prod_{j=1}^{d} \left(\left(b^{(n)}\right)^{-1}\right)_{k_j}. \tag{5.61}$$

The prefiltering filter $\left(b^{(n,d)}\right)^{-1}$ being separable, the B-spline coefficients c can be computed by filtering f successively along each dimension by $\left(b^{(n)}\right)^{-1}$. In other words, the multi-dimensional prefiltering is decomposed in successive unidimensional prefilterings along each dimension.

Algorithms in 2D. A particular and interesting case is given by $d = 2$ where the finite discrete signals to be interpolated are images. The B-spline coefficients are obtained by applying successively the unidimensional prefiltering on the columns and on the rows.

More precisely, let $g \in \mathbb{R}^{\mathbb{Z}^2}$ and $(i, j) \in \mathbb{Z}^2$. We denote $C^j(g) \in \mathbb{R}^{\mathbb{Z}}$ the j-th column of g so that $C^j(g)_i = g_{i,j}$. Similarly, we denote $R^i(g) \in \mathbb{R}^{\mathbb{Z}}$ the i-th row of g so that $R^i(g)_j = g_{i,j}$. Define $c_{\text{col}}(f) \in \mathbb{R}^{\mathbb{Z}^2}$, the unidimensional prefiltering of the columns of f, by their columns

$$C^j(c_{\text{col}}(f)) = C^j(f) * (b^{(n)})^{-1}. \tag{5.62}$$

Then, the B-spline coefficients c are given by the unidimensional prefiltering of the lines of c_{col} i.e.

$$R^i(c) = R^i(c_{\text{col}}(f)) * (b^{(n)})^{-1}. \tag{5.63}$$

In practice the images are finite and an arbitrary extension has to be chosen. Let f be an image of size $K \times L$. In order to compute interpolated values in $[0, K-1] \times [0, L-1]$ the B-spline coefficients c of f has to be computed in $\{-\tilde{n}, \dots, K-1+\tilde{n}\} \times \{-\tilde{n}, \dots, L-1+\tilde{n}\}$. According to (5.62) and (5.63) it is done by applying Algorithm 5.3 or Algorithm 5.4 successively on the columns and on the rows.

Theorem 5.2. *Assume $n \leq 16$. Let $\varepsilon > 0$ and f be a finite image of size at least 4 along each dimension (arbitrarily extended to $\mathbb{Z}^2$). Denote $\varepsilon' = \frac{\varepsilon \rho^{(n)}}{2}$. The computation of the B-spline coefficients of f by applying Algorithm 5.3 or Algorithm 5.4 successively on the columns and on the rows with truncation indices $(N^{(i,\varepsilon')})_{1 \leq i \leq \tilde{n}}$ (as in Algorithm 5.7) have a precision of ε.*

Proof. See Section 5.5.2 □

Theorem 5.2 provides a control of the error that is committed during the two-dimensional prefiltering. Note that to insure a precision ε using Algorithm 5.3, $c_{\text{col}}(f)$ has to be computed in $\{-\tilde{n}, \dots, K-1+\tilde{n}\} \times \{-L_0^{(n,\varepsilon')}, \dots, L-1+L_0^{(n,\varepsilon')}\}$ i.e. for the columns indexed by $j \in \{-L_0^{(n,\varepsilon')}, \dots, L-1+L_0^{(n,\varepsilon')}\}$. In Figure 5.3 is displayed, for the four classical boundary condition, the extended image used during the prefiltering (using Algorithm 5.3) for a precision of $\varepsilon = 10^{-8}$ i.e. 8 digits and $n = 11$. The image is extended of $L_0^{(n,\varepsilon')} = 125$ pixels from its boundary.

The two-dimensional indirect B-spline transform is efficiently performed, as described in Algorithm 5.6, as a convolution with the separable and compactly supported function $\beta^{(n,2)}$. Finally, the two-dimensional B-spline interpolation of an image is presented in Algorithm 5.7.

(a) Constant (b) Periodic

(c) Half-symmetric (d) Whole-symmetric

Figure 5.3: Extended image used during the prefiltering using Algorithm 5.3 for $\varepsilon = 10^{-8}$ i.e 8 digits and $n = 11$. The image is extended of $L_0^{(n,\varepsilon')} = 125$ pixels from its boundary.

Algorithm 5.6: Two-dimensional indirect B-spline transform

Input : An image f of size $K \times L$, a B-spline interpolation order n, the corresponding B-spline coefficients c in $\{-\tilde{n}, \dots, K-1+\tilde{n}\} \times \{-\tilde{n}, \dots, L-1+\tilde{n}\}$ and a location $(x,y) \in [0, K-1] \times [0, L-1]$

Output: The interpolated value $\varphi^{(n)}(x,y)$

$x0 \leftarrow \lceil x - (n+1)/2 \rceil$
for $k = 0$ **to** $\max(1,n)$ **do**
  Tabulate xBuf$[k] \leftarrow \beta^{(n)}(x - (x0+k))$
Initialize $\varphi^{(n)}(x,y) \leftarrow 0$
for $l = 0$ **to** $\max(1,n)$ **do**
  $y0 \leftarrow \lceil y - (n+1)/2 \rceil$
  Initialize $s \leftarrow 0$
  for $k = 0$ **to** $\max(1,n)$ **do**
    Update $s \leftarrow s + c_{x0+k,y0+l}\,$xBuf$[k]$
  Update $\varphi^{(n)}(x,y) \leftarrow \varphi^{(n)}(x,y) + s\,\beta^{(n)}(y - (y0+l))$

Algorithm 5.7: Two-dimensional B-spline interpolation

Input : An image f of size $K \times L$, an extension method, a B-spline interpolation order $n \leq 16$, a precision ε and a list of pixel locations $(x_j, y_j)_{1 \leq j \leq J} \in ([0, K-1] \times [0, L-1])^J$

Output: The interpolated values $(\varphi^{(n)}(x_j, y_j))_{j \in J}$ (with precision ε)

Compute $\varepsilon' = \frac{\rho^{(n)} e}{2}$.
Compute with precision ε' the unidimensional prefiltering of the columns of f, noted c_{col}, using Algorithm 5.3 or Algorithm 5.4
Compute with precision ε' the unidimensional prefiltering of the rows of c_{col}, noted c, using Algorithm 5.3 or Algorithm 5.4
for $j = 1$ **to** J **do**
  Compute, from the B-spline coefficient c, the interpolated value $\varphi^{(n)}(x_j, y_j)$ using Algorithm 5.6

5.5 Control of the Prefiltering Error

Theorem 5.1 and Theorem 5.2 state that the error committed during the prefiltering can be controlled with an adequate choice of the truncation indices. They are proven to be true in the following.

5.5.1 Proof of Theorem 5.1

In Algorithm 5.3 and Algorithm 5.4, the output error comes from the truncation of the initialization sums during the application of the exponential filters. Each truncation introduces an error that propagates to the following computations. To prove Theorem 5.1 we state and prove a more general theorem which provides a control of the error incurring after each application of an exponential filter.

Given N, a non-negative integer, $L_{\text{ini}} < L_{\text{end}}$ two integers, and $s \in \mathbb{R}^{\mathbb{Z}}$, we define the truncated signal $T_{N,L_{\text{ini}},L_{\text{end}}}(s)$ by

$$\forall k \in \mathbb{Z}, \quad T_{N,L_{\text{ini}},L_{\text{end}}}(s)_k = \begin{cases} s_k & L_{\text{ini}} - N \leq k \leq L_{\text{end}} + N, \\ 0 & \text{otherwise.} \end{cases} \tag{5.64}$$

We recall that f denotes the input finite signal of length K and $\varepsilon > 0$ the output precision. Instead of computing the intermediate filtered signals $(c^{(i)})_{0 \leq i \leq \tilde{n}}$, in Algorithm 5.3 and Algorithm 5.4 we compute the $(p^{(i)})_{0 \leq i \leq \tilde{n}}$ defined by

$$\begin{cases} p^{(0)} = f, \\ p^{(i)} = h^{(z_i)} * T_{N^{(i,\varepsilon)},L_{\text{ini}}^{(i,\varepsilon)},L_{\text{end}}^{(i,\varepsilon)}}(p^{(i-1)}) \quad \text{for } 1 \leq i \leq \tilde{n}, \end{cases} \tag{5.65}$$

where

$$L_{\text{ini}}^{(i,\varepsilon)} = \begin{cases} -L_i^{(n,\varepsilon)} & \text{in Algorithm 5.3,} \\ 0 & \text{in Algorithm 5.4,} \end{cases} \tag{5.66}$$

and

$$L_{\text{end}}^{(i,\varepsilon)} = \begin{cases} K - 1 + L_i^{(n,\varepsilon)} & \text{in Algorithm 5.3,} \\ K - 1 & \text{in Algorithm 5.4.} \end{cases} \tag{5.67}$$

For $0 \leq i \leq \tilde{n}$, we can write $p^{(i)} = c^{(i)} + e^{(i)}$ where $e^{(i)}$ is the error committed at step i. Finally the computed coefficients are $p = \gamma^{(n)} p^{(\tilde{n})} = c^{(\tilde{n})} + \gamma^{(n)} e^{(\tilde{n})}$. Note that the filtered signals are implicitly extended outside their computational domains (by zeros in Algorithm 5.3 and by the boundary condition in Algorithm 5.4).

Lemma 5.1. *Let $s \in \mathbb{R}^{\mathbb{Z}}$ and $-1 < \alpha < 0$. Let $g \in \mathbb{R}^{\mathbb{Z}}$ be a perturbation. Let N be a non-negative integer and $L_{ini} < L_{end}$ be two integers. Define $s' = T_{N,L_{ini},L_{end}}(s+g)$ and the error $e = h^{(\alpha)} * (s' - s)$. Then,*

$$\|e\|_\infty \leq C_\alpha \left((1-\alpha) \|g\|_\infty + |\alpha|^{N+1} \left(1 + |\alpha|^{L_{end} - L_{ini}} \right) \|s\|_\infty \right), \tag{5.68}$$

where

$$C_\alpha = \frac{-\alpha}{(1-\alpha^2)(1+\alpha)}. \tag{5.69}$$

Proof of Lemma 5.1. With (5.30) we can write

$$e = \frac{\alpha}{\alpha^2 - 1} \left(\underbrace{k^{(\alpha)} * (s' - s)}_{\text{causal part}} + \underbrace{l^{(\alpha)} * (s' - s)}_{\text{anticausal part}} - (s' - s) \right) \tag{5.70}$$

Now, let us evaluate e_i for $L_{\text{ini}} \leq i \leq L_{\text{end}}$. First, by definition of s', $(s'-s)_i = g_i$. Then we deal with the causal and anti-causal part of (5.70).

Causal part. With the adequate change of indices we have

$$\left(k^{(\alpha)} * (s'-s)\right)_i = \sum_{j=0}^{\infty} \alpha^j (s'-s)_{i-j} \tag{5.71}$$

$$= \sum_{j=-\infty}^{i} \alpha^{i-j} (s'-s)_j \tag{5.72}$$

$$= \alpha^i \left(\sum_{j=L_{\text{ini}}-N}^{i} \alpha^{-j} g_j - \sum_{j=-\infty}^{L_{\text{ini}}-N-1} \alpha^{-j} s_j \right). \tag{5.73}$$

Anticausal part. Similarly we have

$$\left(l^{(\alpha)} * (s'-s)\right)_i = \sum_{j=0}^{\infty} \alpha^j (s'-s)_{i+j} \tag{5.74}$$

$$= \sum_{j=i}^{\infty} \alpha^{j-i} (s'-s)_j \tag{5.75}$$

$$= \alpha^{-i} \left(\sum_{j=i}^{L_{\text{end}}+N} \alpha^j g_j - \sum_{j=L_{\text{end}}+N+1}^{\infty} \alpha^j s_j \right). \tag{5.76}$$

Finally,

$$e_i = \frac{\alpha}{\alpha^2-1} \left(\underbrace{\alpha^i \sum_{j=L_{\text{ini}}-N}^{i-1} \alpha^{-j} g_j + \alpha^{-i} \sum_{j=i}^{L_{\text{end}}+N} \alpha^j g_j}_{\text{previous error}} \underbrace{- \alpha^i \sum_{j=-\infty}^{L_{\text{ini}}-N-1} \alpha^{-j} s_j - \alpha^{-i} \sum_{j=L_{\text{end}}+N+1}^{\infty} \alpha^j s_j}_{\text{initialization error}} \right). \tag{5.77}$$

By the triangular inequality we obtain the four upper-bounds

$$\left| \sum_{j=L_{\text{ini}}-N}^{i-1} \alpha^{-j} g_j \right| \leq |\alpha|^{N-L_{\text{ini}}} \frac{1-|\alpha|^{-(i-L_{\text{ini}}+N)}}{1-|\alpha|^{-1}} \|g\|_\infty = -\alpha \frac{|\alpha|^{-i} - |\alpha|^{N-L_{\text{ini}}}}{1+\alpha} \|g\|_\infty, \tag{5.78}$$

$$\left| \sum_{j=i}^{L_{\text{end}}+N} \alpha^j g_j \right| \leq |\alpha|^i \frac{1-|\alpha|^{L_{\text{end}}+N-i+1}}{1-|\alpha|} \|g\|_\infty = \frac{|\alpha|^i - |\alpha|^{L_{\text{end}}+N+1}}{1+\alpha} \|g\|_\infty, \tag{5.79}$$

$$\left| \sum_{j=-\infty}^{L_{\text{ini}}-N-1} \alpha^{-j} s_j \right| \leq \frac{|\alpha|^{N+1-L_{\text{ini}}}}{1+\alpha} \|s\|_\infty, \tag{5.80}$$

$$\left| \sum_{j=L_{\text{end}}+N+1}^{\infty} \alpha^j s_j \right| \leq \frac{|\alpha|^{N+1+L_{\text{end}}}}{1+\alpha} \|s\|_\infty. \tag{5.81}$$

Thus,

$$|e_i| \leq C_\alpha \left(\|g\|_\infty \left[1 - \alpha \left(1 - |\alpha|^{N-L_{\text{ini}}+i} - |\alpha|^{N+L_{\text{end}}-i} \right) \right] + \|s\|_\infty |\alpha|^{N+1} \left(|\alpha|^{i-L_{\text{ini}}} + |\alpha|^{L_{\text{end}}-i} \right) \right) \tag{5.82}$$

Using the relation $|\alpha|^{i-L_{\text{ini}}} + |\alpha|^{L_{\text{end}}-i} \leq 1 + |\alpha|^{L_{\text{end}}-L_{\text{ini}}}$ and by removing the negative terms we get the following upper-bound that is independent of i,

$$\|e\|_\infty \leq C_\alpha \left((1-\alpha) \|g\|_\infty + |\alpha|^{N+1} \left(1 + |\alpha|^{L_{\text{end}}-L_{\text{ini}}} \right) \|s\|_\infty \right). \tag{5.83}$$

□

For $0 \leq i \leq \tilde{n}$, define $D^{(i)}$ and $\varepsilon^{(i)}$ by

$$D^{(i)} = \prod_{j=1}^{i} \frac{-z_j}{(1+z_j)^2} \tag{5.84}$$

and

$$\varepsilon^{(i)} = \varepsilon \rho^{(n)} D^{(i)} \prod_{j=i+1}^{\tilde{n}} \mu_j. \tag{5.85}$$

We recall that the $(\mu_j)_{1\leq j\leq \tilde{n}}$ are defined in (5.39). Lemma 5.1 provides a control of the error when an exponential filter is applied to a signal whose values may be perturbed by a small error and where the initialization sums are truncated. We deduce from it the following theorem.

Theorem 5.3. *Assume $n \leq 16$ and $K \geq 4$. For $0 \leq i \leq \tilde{n}$, we have*

$$\|e^{(i)}\|_\infty \leq \varepsilon^{(i)} \|f\|_\infty. \tag{5.86}$$

To prove this theorem we need the two following lemmas.

Lemma 5.2. *For $0 \leq l \leq \tilde{n}$, we have*

$$\|c^{(l)}\|_\infty \leq D^{(l)} \|f\|_\infty. \tag{5.87}$$

Proof of Lemma 5.2. Let $s \in \mathbb{R}^{\mathbb{Z}}$ and $-1 < \alpha < 0$. Applying the triangular inequality in (5.31) we get

$$\|h^{(\alpha)} * s\|_\infty \leq \frac{\alpha}{\alpha^2 - 1} \frac{1-\alpha}{1+\alpha} |s|_\infty = \frac{-\alpha}{(1+\alpha)^2} \|s\|_\infty. \tag{5.88}$$

The result is proven by successively applying this inequality to $c^{(j)}$ and z_j for $1 \leq j \leq l$. □

Define $\theta : (x,m) \in]-1,0[\times\mathbb{N} \mapsto -\frac{\log(1+|x|^m)}{\log|x|}$.

Lemma 5.3. *For $x > -0.75$ and $m \geq 4$ we have $\theta(x,m) < 1$.*

Proof of Lemma 5.3. θ is a decreasing function with respect to its variables and $\theta(-0.75,4) < 1$. □

Proof of Theorem 5.3. The result is proved by induction.

- **Base case:** For $i = 0$ we have $e^{(0)} = 0$ and because $\mu_1 = 0$, $\varepsilon^{(0)} = 0$. Thus, $\|e^{(0)}\|_\infty = \varepsilon^{(0)} \|f\|_\infty$.
- **Inductive step:** For $1 \leq i \leq \tilde{n}$, assume $\|e^{(i-1)}\|_\infty \leq \varepsilon^{(i-1)} \|f\|_\infty$. Then we prove that $\|e^{(i)}\|_\infty \leq \varepsilon^{(i)} \|f\|_\infty$ as follows.

  Applying Lemma 5.1 to $s = c^{(i-1)}$, $\alpha = z_i$, $g = e^{(i-1)}$, $N = N^{(i,\varepsilon)}$, $L_{\text{ini}} = L_{\text{ini}}^{(i,\varepsilon)}$ and $L_{\text{end}} = L_{\text{end}}^{(i,\varepsilon)}$ we get

$$\|e^{(i)}\|_\infty \leq C_{z_i} \left((1-z_i) \|e^{(i-1)}\|_\infty + |z_i|^{N^{(i,\varepsilon)}+1} \left(1 + |z_i|^{L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)}}\right) \|c^{(i-1)}\|_\infty \right). \tag{5.89}$$

  Using the induction hypothesis and Lemma 5.2 with $l = i-1$ we have

$$\|e^{(i)}\|_\infty \leq \eta^{(i,\varepsilon)} \|f\|_\infty \tag{5.90}$$

  where

$$\eta^{(i,\varepsilon)} = C_{z_i} \left((1-z_i)\varepsilon^{(i-1)} + |z_i|^{N^{(i,\varepsilon)}+1} \left(1 + |z_i|^{L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)}}\right) D^{(i-1)} \right). \tag{5.91}$$

Noting that for $n \leq 16$ the poles are greater than -0.75, we have by definition of $N^{(i,\varepsilon)}$ and with Lemma 5.3 applied to $x = z_i > -0.75$ and $m = L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)} \geq 4$,

$$N^{(i,\varepsilon)} + 1 \geq \frac{\log\left(\varepsilon\rho^{(n)}(1-z_i)(1-\mu_i)\prod_{j=i+1}^{\tilde{n}}\mu_j\right)}{\log|z_i|} + 1 \tag{5.92}$$

$$\geq \frac{\log\left(\varepsilon\rho^{(n)}(1-z_i)(1-\mu_i)\prod_{j=i+1}^{\tilde{n}}\mu_j\right)}{\log|z_i|} + \theta\left(z_i, L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)}\right) \tag{5.93}$$

$$\geq \frac{\log\left(\varepsilon\rho^{(n)}(1-z_i)(1-\mu_i)\prod_{j=i+1}^{\tilde{n}}\mu_j\right) - \log\left(1+|z_i|^{L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)}}\right)}{\log|z_i|}. \tag{5.94}$$

The previous relation is equivalent to

$$|z_i|^{N^{(i,\varepsilon)}+1}\left(1+|z_i|^{L_{\text{end}}^{(i,\varepsilon)} - L_{\text{ini}}^{(i,\varepsilon)}}\right) \leq \varepsilon\rho^{(n)}(1-z_i)(1-\mu_i)\prod_{j=i+1}^{\tilde{n}}\mu_j \tag{5.95}$$

$$= \frac{(1-z_i)\varepsilon^{(i)}}{D^{(i)}} - \frac{(1-z_i)\varepsilon^{(i-1)}}{D^{(i-1)}}. \tag{5.96}$$

Finally,

$$\eta^{(i,\varepsilon)} \leq C_{z_i}\frac{D^{(i-1)}}{D^{(i)}}(1-z_i)\varepsilon^{(i)} = \frac{-z_i(1+z_i)^2(1-z_i)}{-z_i(1-z_i^2)(1+z_i)}\varepsilon^{(i)} = \varepsilon^{(i)} \tag{5.97}$$

and using (5.90) we obtain $\|e^{(i)}\|_\infty \leq \varepsilon^{(i)}\|f\|_\infty$.

□

Theorem 5.3 is more general than Theorem 5.1 because it provides a control of the error at each step.

Lemma 5.4. *We have*

$$\rho^{(n)}D^{(\tilde{n})} = \frac{1}{\gamma^{(n)}}. \tag{5.98}$$

Proof of Lemma 5.4. With the recursive definition of $\beta^{(n)}$ in (5.1) we have for $n \geq 1$,

$$\sum_{k\in\mathbb{Z}}\beta^{(n)}(k) = \sum_{k\in\mathbb{Z}}\int_{\mathbb{R}}\beta^{(n-1)}(x)\beta^{(0)}(k-x)dx = \int_{\mathbb{R}}\beta^{(n-1)}(x)dx = 1. \tag{5.99}$$

Noticing that $\sum_{k\in\mathbb{Z}}\beta^{(n)}(k) = B^{(n)}(1)$ and with (5.14) it can be rewritten has

$$\gamma^{(n)} = \prod_{j=1}^{\tilde{n}}\frac{(1-z_j)^2}{-z_j}. \tag{5.100}$$

By definition of $D^{(\tilde{n})}$ and $\rho^{(n)}$ we have the result. □

From Lemma 5.4 we deduce that $\varepsilon^{(\tilde{n})} = \frac{\varepsilon}{\gamma^{(n)}}$. In particular with $i = \tilde{n}$ in Theorem 5.3, we have $\|e^{(\tilde{n})}\| \leq \varepsilon^{(\tilde{n})}\|f\|_\infty = \frac{\varepsilon}{\gamma^{(n)}}\|f\|_\infty$ which proves Theorem 5.1.

Remarks:

- In practice the error decreases exponentially with the distance to the boundaries. Thus, the upper-bounds in the proof of Theorem 5.3 are not tight. The theoretical precision overestimates the experimental error. In addition, in Algorithm 5.3 we do not initialize at the same places so that the propagation error between steps could be neglected.
- For really small values of ε (e.g. $\varepsilon < 10^{-12}$), the machine precision should be considered. Thus the experimental error may be higher in this case.
- In our implementation (see Section 5.6.3) the order is limited to 16 because of the poles computation. For higher order Theorem 5.3 may remain true provided the signals are large enough.

5.5.2 Proof of Theorem 5.2

Theorem 5.2 can be proven as a direct consequence of Theorem 5.1. We recall that f denotes the input finite image of size at least 4 along each dimension and $\varepsilon > 0$ the output precision. Instead of computing $c_{\text{col}}(f)$ by applying Algorithm 5.3 or Algorithm 5.4 on the columns of f with truncation indices $N^{(n,\varepsilon')}$, we actually compute $p_{\text{col}}(f) = c_{\text{col}}(f) + e_{\text{col}}(f)$ where $e_{\text{col}}(f) \in \mathbb{R}^{\mathbb{Z}^2}$ is an unidimensional prefiltering error. Then, by applying Algorithm 5.3 or Algorithm 5.4 on the rows of $p_{\text{col}}(f)$ we obtain

$$p = c + e_{\text{row}} + e_{\text{row+col}} \tag{5.101}$$

where e_{row} is the error that comes from the prefiltering along the rows of $p_{\text{col}}(f)$ and $e_{\text{row+col}}$ is the prefiltering along the rows of $e_{\text{col}}(f)$. The output error $e = e_{\text{row}} + e_{\text{row+col}}$ is proven to be smaller than ε as follows. On the one hand,

$$\|e_{\text{row+col}}\|_\infty \underset{\textit{Lemma 5.2}}{\leq} \gamma^{(n)} D^{(\tilde{n})} \|e_{\text{col}}\|_\infty \underset{\textit{Theorem 5.1}}{\leq} \varepsilon' \gamma^{(n)} D^{(\tilde{n})} \|f\|_\infty. \tag{5.102}$$

On the other hand,

$$\|e_{\text{row}}\|_\infty \underset{\textit{Theorem 5.1}}{\leq} e' \|c_{\text{col}}(f)\|_\infty \underset{\textit{Lemma 5.2}}{\leq} \varepsilon' \gamma^{(n)} D^{(\tilde{n})} \|f\|_\infty \tag{5.103}$$

Finally,

$$\|e\|_\infty \leq 2\varepsilon' \gamma^{(n)} D^{(\tilde{n})} \underset{\textit{Lemma 5.4}}{=} \varepsilon \tag{5.104}$$

which proves Theorem 5.2.

5.5.3 Choice of the μ_i

To prove Theorem 5.3 we did not use the value of μ_i for $2 \leq i \leq \tilde{n}$. Any $(\mu_j)_{2\leq j\leq \tilde{n}} \in]0,1[^{\tilde{n}-1}$ is acceptable to guarantee the output precision ε but the choice has an influence on the truncation indices $N^{(i,\varepsilon)}$ and thus on the complexity of the algorithms. Indeed, a small value for μ_i leads to less computations for the step i but more computations for steps $j < i$. We set $\mu_1 = 0$ because $e^{(0)} = 0$.

We select $\mu = (\mu_j)_{2\leq j\leq \tilde{n}} \in]0,1[^{\tilde{n}-1}$ that minimizes the function

$$\Psi : \nu = (\nu_j)_{2\leq j\leq \tilde{n}} \in]0,1[^{\tilde{n}-1} \mapsto \sum_{i=1}^{\tilde{n}} \frac{\log\left((1-\nu_i)\prod_{j=i+1}^{\tilde{n}} \nu_j\right)}{\log|z_i|} \tag{5.105}$$

with the notation $\nu_1 = 0$. In other words it is selected so that $\sum_{i=1}^{\tilde{n}} N^{(i,\varepsilon)}$ is small[3]. Ψ is a derivable function that admits a unique minimizer μ that is given by the Euler condition $\nabla\Psi(\mu) = 0$. For $2 \leq k \leq \tilde{n}$ we have

$$\frac{\partial\Psi}{\partial\nu_k}(\mu) = \frac{-1}{1-\mu_k}\frac{1}{\log|z_k|} + \frac{1}{\mu_k}\sum_{i=1}^{k-1}\frac{1}{\log|z_i|} = 0 \tag{5.106}$$

i.e.

$$\frac{1}{\mu_k} = 1 + \frac{1}{\log|z_k|\sum_{i=1}^{k-1}\frac{1}{\log|z_i|}} \tag{5.107}$$

which corresponds to the definition given in (5.39). The values of $(\mu_i)_{2\leq i\leq\tilde{n}}$ are displayed in Table 5.2 for $4 \leq n \leq 9$.

n	$(\mu_i)_{2\leq i\leq\tilde{n}}$
4	$\mu_2 = 0.8081702588338142$
5	$\mu_2 = 0.7886523126940346$
6	$\mu_2 = 0.7775037872839968$
	$\mu_3 = 0.9217057449487258$
7	$\mu_2 = 0.7705847640302491$
	$\mu_3 = 0.9069526580525736$
8	$\mu_2 = 0.7660491039752506$
	$\mu_3 = 0.8982276825918423$
	$\mu_4 = 0.9583935084163903$
9	$\mu_2 = 0.7628638545450653$
	$\mu_3 = 0.8921921530329509$
	$\mu_4 = 0.9478524258426756$

Table 5.2: Values of $(\mu_i)_{2\leq i\leq\tilde{n}}$ for $4 \leq n \leq 9$.

5.6 Practical Computations and Numerical Implementation Details

In this section we explain how computations are performed in practice. For any order n, algorithms for computing the poles along with the normalization constant and evaluating the B-spline function are detailed.

5.6.1 Practical Computation of the Poles and the Normalization Constant

As expressed in (5.25), the prefiltering step requires the knowledge of the normalization constant $\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}$ and of the poles of order n introduced in (5.13). The poles correspond to the roots in $]-1,0[$ of the (palindromic) polynomial $\tilde{B}^{(n)}$ of degree $2\tilde{n}$ defined for $z \in \mathbb{C}$ by

$$\tilde{B}^{(n)}(z) = z^{\tilde{n}}B^{(n)}(z) = b_0^{(n)}z^{\tilde{n}} + \sum_{i=1}^{\tilde{n}} b_i^{(n)}(z^{\tilde{n}+i} + z^{\tilde{n}-i}). \tag{5.108}$$

The coefficients $(b_k^{(n)})_{0\leq k\leq\tilde{n}} = \left(\beta^{(n)}(k)\right)_{0\leq k\leq\tilde{n}}$ being given, the poles can be approximated numerically using a polynomial equation solver. The coefficients themselves can be computed directly thanks to the explicit formula given in Proposition 5.4 but a more efficient computation based on a recursive formula, which only involves simple additions and multiplications, is preferred.

[3]The minimality is not guaranteed because of the integral part in the definition of $N^{(i,\varepsilon)}$.

Proposition 5.2. *Let $m \geq 1$ and $x \in \mathbb{R}$. Then,*

$$m\beta^{(m)}(x) = \left(\frac{m+1}{2}+x\right)\beta^{(m-1)}\left(x+\frac{1}{2}\right)+\left(\frac{m+1}{2}-x\right)\beta^{(m-1)}\left(x-\frac{1}{2}\right). \tag{5.109}$$

Proof. It is shown by induction on m. For $m = 1$, it is straightforward using the explicit expression

$$\beta^{(1)}(x) = \begin{cases} 1-|x|, & |x| \leq 1 \\ 0, & |x| > 1. \end{cases} \tag{5.110}$$

Assume the identity is verified up to $m \geq 1$. Then we can write

$$\beta^{(m+1)}(x) = \int_{-1/2}^{1/2} \beta^{(0)}(y)\beta^{(m)}(x-y)\,dy, \tag{5.111}$$

and integrating by parts we get

$$\begin{aligned} \beta^{(m+1)}(x) &= \left[y\beta^{(m)}(x-y)\right]_{-1/2}^{1/2} + \int_{-1/2}^{1/2} y\beta^{(m)'}(x-y)\,dy \\ &= \frac{1}{2}\left(\beta^{(m)}(x+\frac{1}{2})+\beta^{(m)}(x-\frac{1}{2})\right) + \int_{\mathbb{R}} \beta^{(0)}(y)y\beta^{(m)'}(x-y)\,dy. \end{aligned} \tag{5.112}$$

For $m = 1$, $\beta^{(1)'}$ is defined everywhere except at $x \in \{-1,0,1\}$, but from the point of view of integration we can ignore this defect and write

$$\begin{aligned} \beta^{(1)'}(x) &= \begin{cases} 1, & -1 < x < 0 \\ -1, & 0 < x < 1 \\ 0, & |x| > 1, \end{cases} \\ &= \beta^{(0)}(x+\frac{1}{2}) - \beta^{(0)}(x-\frac{1}{2}). \end{aligned} \tag{5.113}$$

Now for $m \geq 2$, we have

$$\begin{aligned} \beta^{(m)'}(x) &= \int \beta^{(m-2)}(y)\beta^{(1)'}(x-y)\,dy \\ &= \int \beta^{(m-2)}(y)\beta^{(0)}(x+\frac{1}{2}-y)\,dy - \int \beta^{(m-2)}(y)\beta^{(0)}(x-\frac{1}{2}-y)\,dy \\ &= \beta^{(m-1)}(x+\frac{1}{2}) - \beta^{(m-1)}(x-\frac{1}{2}), \end{aligned} \tag{5.114}$$

so that according to (5.113), this equation is also valid for $m = 1$. Note that this could also be shown more concisely using distributions by the observation that $\beta^{(0)'} = \delta_{-1/2} - \delta_{1/2}$. Therefore

$$\int \beta^{(0)}(y)y\beta^{(m)'}(x-y)\,dy = \int \beta^{(0)}(y)y\left(\beta^{(m-1)}(x+\frac{1}{2}-y) - \beta^{(m-1)}(x-\frac{1}{2}-y)\right)dy. \tag{5.115}$$

But the recursivity assumption at location $x-y$ can be rewritten

$$\begin{aligned} y\left(\beta^{(m-1)}(x+\frac{1}{2}-y) - \beta^{(m-1)}(x-\frac{1}{2}-y)\right) &= x\left(\beta^{(m-1)}(x+\frac{1}{2}-y) - \beta^{(m-1)}(x-\frac{1}{2}-y)\right) \\ &\quad + \frac{m+1}{2}\left(\beta^{(m-1)}(x+\frac{1}{2}-y) + \beta^{(m-1)}(x-\frac{1}{2}-y)\right) \\ &\quad - m\beta^{(m)}(x-y) \end{aligned} \tag{5.116}$$

Combining (5.115) and (5.116), and then coming back to (5.112), we can write

$$\beta^{(m+1)}(x) = \left(\frac{m+1}{2}+\frac{1}{2}\right)\left(\beta^{(m)}(x+\frac{1}{2})+\beta^{(m)}(x-\frac{1}{2})\right) + x\beta^{(m)}(x+\frac{1}{2}) - x\beta^{(m)}(x-\frac{1}{2}) - m\beta^{(m+1)}(x), \tag{5.117}$$

which yields after rearrangement of the terms

$$(m+1)\beta^{(m+1)}(x) = \left(\frac{(m+1)+1}{2}+x\right)\beta^{(m)}(x+\frac{1}{2}) + \left(\frac{(m+1)+1}{2}-x\right)\beta^{(m)}(x-\frac{1}{2}), \tag{5.118}$$

exactly (5.109) at $m+1$. □

Let $m \geq 1$. Setting for simplicity $d_k^{(m)} = \beta^{(m)}\left(k+\frac{1}{2}\right)$ for $k \in \mathbb{Z}$, we have with Proposition 5.2

$$m b_k^{(m)} = \left(\frac{m+1}{2}+k\right) d_k^{(m-1)} + \left(\frac{m+1}{2}-k\right) d_{k-1}^{(m-1)} \tag{5.119}$$

$$m d_k^{(m)} = \left(\frac{m+2}{2}+k\right) b_{k+1}^{(m-1)} + \left(\frac{m}{2}-k\right) b_k^{(m-1)}. \tag{5.120}$$

Define $\tilde{m} = \left[\frac{m}{2}\right]$. As $b_{\tilde{m}+1}^{(m)} = 0$ and $d_{-1}^{(m)} = d_0^{(m)}$ we can compute the coefficients $(b_k^{(n)})_{0\leq k\leq \tilde{n}}$ recursively using Algorithm 5.8.

Algorithm 5.8: Polynomial coefficients computation

Input : The B-spline order n
Output: The coefficients $(b_k^{(n)})_{0\leq k\leq \tilde{n}}$ of $\tilde{B}^{(n)}$

Initialize with $b_0^{(0)} = 1$ and $d_0^{(0)} = \frac{1}{2}$
for $m = 1$ **to** $n-1$ **do**
  Define $\tilde{m} = \left[\frac{m}{2}\right]$
  for $k = 0$ **to** $\tilde{m}$ **do**
    Compute $b_k^{(m)}$ using (5.119)
    Compute $d_k^{(m)}$ using (5.120)
for $k = 0$ **to** $\tilde{n}$ **do**
  Compute $b_k^{(n)}$ using (5.119)

In particular, it is possible to obtain an explicit expression of $\gamma^{(n)} = \frac{1}{b_{\tilde{n}}^{(n)}}$ as a function of n.

Proposition 5.3. *We have*

$$\gamma^{(n)} = \begin{cases} 2^n n! & n \text{ even} \\ n! & n \text{ odd.} \end{cases} \tag{5.121}$$

Proof. Assuming $n \geq 2$ and applying (5.119) to $k = \tilde{n}$ and $m = n$ we get

$$b_{\tilde{n}}^{(n)} = \frac{1}{n}\left(\frac{n+1}{2}-\tilde{n}\right) d_{\tilde{n}-1}^{(n-1)}. \tag{5.122}$$

Similarly, applying (5.120) to $k = \tilde{n}-1$ and $m = n-1$ we get

$$d_{\tilde{n}-1}^{(n-1)} = \frac{1}{n-1}\left(\frac{n+1}{2}-\tilde{n}\right) b_{\tilde{n}-1}^{(n-2)}. \tag{5.123}$$

As $\tilde{n}-1=\widetilde{n-2}$ we obtain the recursive equation

$$b_{\tilde{n}}^{(n)}=\frac{1}{n(n-1)}\left(\frac{n+1}{2}-\tilde{n}\right)^2 b_{\widetilde{n-2}}^{(n-2)}. \tag{5.124}$$

Now let $n\geq 0$. Noting that $b_{\tilde{0}}^{(0)}=b_{\tilde{1}}^{(1)}=1$ and

$$\frac{m+1}{2}-\tilde{m}=\begin{cases}\frac{1}{2} & m \text{ even}\\ 1 & m \text{ odd},\end{cases} \tag{5.125}$$

we finally have the following explicit expression for $b_{\tilde{n}}^{(n)}$,

$$b_{\tilde{n}}^{(n)}=\begin{cases}\frac{1}{2^n n!} & n \text{ even}\\ \frac{1}{n!} & n \text{ odd}.\end{cases} \tag{5.126}$$

□

It is a direct consequence of Proposition 5.3 that $\gamma^{(n)}\tilde{B}^{(n)}$ is a polynomial with integer coefficients. Indeed, let $j\in\mathbb{Z}$. Combining (5.128) at location $x=j$ with (5.121), we obtain $b_j^{(n)}=b_{\tilde{n}}^{(n)}a_j^{(n)}$ where

$$a_j^{(n)}=\begin{cases}\sum_{i=0}^{n+1}\binom{n+1}{i}(-1)^i(2(j-i)+n+1)_+^n, & n \text{ even}\\ \sum_{i=0}^{n+1}\binom{n+1}{i}(-1)^i\left(j-i+\frac{n+1}{2}\right)_+^n, & n \text{ odd}.\end{cases} \tag{5.127}$$

Thus $a_j^{(n)}=\frac{b_j^{(n)}}{b_{\tilde{n}}^{(n)}}\in\mathbb{Z}$. Actually it is a non-negative integer since $\beta^{(n)}$ is non-negative. This property justifies why the expression of $B^{(n)}$ in Table 5.3 only involves integers. However it is not used in practice because the renormalization by $\gamma^{(n)}$ may introduce numerical errors for n large.

5.6.2 Normalized B-spline Function Evaluation

The evaluation of the normalized B-spline function $\beta^{(n)}$ at any location $x\in\mathbb{R}$ is necessary in order to perform the indirect B-spline transform (see Algorithm 5.5 and Algorithm 5.6). As $\beta^{(n)}$ is continuous, even and compactly supported in $[-\frac{n+1}{2},\frac{n+1}{2}]$, it is sufficient to evaluate $\beta^{(n)}(x)$ for $0\leq x<\frac{n+1}{2}$. Moreover, $\beta^{(n)}$ is a piece-wise polynomial function so this evaluation can be efficiently performed after the precomputation of the polynomial coefficients between each pair of successive knots.

Proposition 5.4 (Explicit expression of $\beta^{(n)}$)**.** *For $n\geq 1$ and $x\in\mathbb{R}$,*

$$\beta^{(n)}(x)=\frac{1}{n!}\sum_{i=0}^{n+1}(-1)^i\binom{n+1}{i}\left(x-i+\frac{n+1}{2}\right)_+^n \tag{5.128}$$

where for all $y\in\mathbb{R}$, $y_+=\max(y,0)$ denotes the positive part of y.

Proof. It is shown by induction on n. For $n=1$, we have the four cases for the right hand side of (5.128):

$$\begin{cases}0+0+0=0 & \text{if } x<-1 \text{ since } x-i+1<0 \text{ for } i=0,1,2.\\ (x+1)_+ +0+0=x+1 & \text{if } -1\leq x\leq 0.\\ (x+1)-2x+0=1-x & \text{if } 0\leq x\leq 1.\\ (x+1)-2x+(x-1)=0 & \text{if } 1\leq x.\end{cases}$$

We recognize the function $\max(0,1-|x|)$. On the other hand, we have

$$\beta^{(1)}(x) = \int \beta^{(0)}(y)\beta^{(0)}(x-y)\,dy = \left(\int_{\max(-1/2,x-1/2)}^{\min(1/2,x+1/2)} dy\right)_+ = (1+\min(0,x)-\max(0,x))_+ = (1-|x|)_+,$$

which justifies (5.128) for $n=1$. To proceed with the induction, let us first consider the function $p_n(x) = (x)_+^n$ and compute

$$p_n * \beta^{(0)}(x) = \int_0^{+\infty} y^n \beta^{(0)}(x-y)\,dy = \int_{(x-1/2)_+}^{(x+1/2)_+} y^n\,dy = \frac{1}{n+1}\left((x+\frac{1}{2})_+^{n+1} - (x-\frac{1}{2})_+^{n+1}\right).$$

Now, assuming (5.128) holds for some $n \geq 1$, we get

$$\begin{aligned}
\beta^{(n+1)}(x) =& \frac{1}{n!}\sum_{i=0}^{n+1}(-1)^i\binom{n+1}{i} p_n * \beta^{(0)}\left(x-i+\frac{n+1}{2}\right)\\
=& \frac{1}{(n+1)!}\sum_{i=0}^{n+1}(-1)^i\binom{n+1}{i}\left(\left(x-i+\frac{(n+1)+1}{2}\right)_+^{n+1} - \left(x-(i+1)+\frac{(n+1)+1}{2}\right)_+^{n+1}\right)\\
=& \frac{1}{(n+1)!}\sum_{i=0}^{n+1}(-1)^i\binom{n+1}{i}\left(x-i+\frac{(n+1)+1}{2}\right)_+^{n+1} +\\
& \frac{1}{(n+1)!}\sum_{i=1}^{n+2}(-1)^i\binom{n+1}{i-1}\left(x-i+\frac{(n+1)+1}{2}\right)_+^{n+1}\\
=& \frac{1}{(n+1)!}(-1)^0\binom{n+2}{0}\left(x-0+\frac{(n+1)+1}{2}\right)_+^{n+1} +\\
& \frac{1}{(n+1)!}\sum_{i=0}^{n+1}(-1)^i\left(\binom{n+1}{i}+\binom{n+1}{i-1}\right)\left(x-i+\frac{(n+1)+1}{2}\right)_+^{n+1} +\\
& \frac{1}{(n+1)!}(-1)^{n+2}\binom{n+2}{n+2}\left(x-(n+2)+\frac{(n+1)+1}{2}\right)_+^{n+1}\\
=& \frac{1}{(n+1)!}\sum_{i=0}^{n+2}(-1)^i\binom{n+2}{i}\left(x-i+\frac{(n+1)+1}{2}\right)_+^{n+1}.
\end{aligned}$$

The third equality is obtained by changing index i to $i+1$, and the last one using the identity $\binom{n+1}{i}+\binom{n+1}{i-1} = \binom{n+2}{i}$. Thus, we get the explicit formula (5.128) at index $n+1$. □

Let $0 \leq x < \frac{n+1}{2}$ and $n \geq 1$. Using Proposition 5.4 and by symmetry we get,

$$\beta^{(n)}(x) = \beta^{(n)}(-x) = \frac{1}{n!}\sum_{i=0}^{n+1}(-1)^i\binom{n+1}{i}\left(\frac{n+1}{2}-i-x\right)_+^n \tag{5.129}$$

To get rid of the $+$ subscript in this equation, we observe that

$$\frac{n+1}{2}-i-x > 0 \Leftrightarrow x-\frac{n+1}{2}+i < 0 \Leftrightarrow \left\lfloor x-\frac{n+1}{2}+i\right\rfloor \leq -1 \Leftrightarrow i \leq k$$

with

$$k = \left\lceil \frac{n+1}{2}-x\right\rceil - 1.$$

Therefore (5.129) can be rewritten as the restricted sum

$$\beta^{(n)}(x) = \frac{1}{n!}\sum_{i=0}^{k}(-1)^i\binom{n+1}{i}\left(\frac{n+1}{2}-i-x\right)^n. \tag{5.130}$$

We then expand the powers to write $\beta^{(n)}(x)$ as a sum of monomials. We observe that $0 \leq k \leq \tilde{n}$.

Polynomial expression. Define $y = \frac{n+1}{2} - x - k$, so that $0 < y \leq 1$. Using the relation $\frac{n+1}{2} - i - x = y + (k-i)$ and the binomial expansion, (5.130) becomes

$$\beta^{(n)}(x) = \frac{1}{n!}\left((-1)^k\binom{n+1}{k}y^n + \sum_{i=0}^{k-1}(-1)^i\binom{n+1}{i}\sum_{j=0}^{n}\binom{n}{j}y^j(k-i)^{n-j}\right) \tag{5.131}$$

$$= \frac{1}{n!}\left((-1)^k\binom{n+1}{k}y^n + \sum_{j=0}^{n}\left(\binom{n}{j}\sum_{i=0}^{k-1}(-1)^i\binom{n+1}{i}(k-i)^{n-j}\right)y^j\right) \tag{5.132}$$

For $0 \leq j \leq n$ define the polynomial coefficients

$$C_{k,j}^{(n)} = \begin{cases} \binom{n}{j}\sum_{i=0}^{k-1}(-1)^i\binom{n+1}{i}(k-i)^{n-j} & 0 \leq j \leq n-1 \\ \sum_{i=0}^{k}(-1)^i\binom{n+1}{i} & j = n \end{cases} \tag{5.133}$$

so that

$$\beta^{(n)}(x) = \frac{1}{n!}\sum_{j=0}^{n} C_{k,j}^{(n)} y^j. \tag{5.134}$$

Polynomial expression near 0. Assume $k = \tilde{n}$, i.e., $0 \leq x < \frac{n+1}{2} - \tilde{n}$. This upper bound is 0.5 for even n and 1 for odd n. Using the binomial expansion, we have

$$\beta^{(n)}(x) = \frac{1}{n!}\sum_{i=0}^{\tilde{n}}(-1)^i\binom{n+1}{i}\sum_{j=0}^{n}(-1)^j\binom{n}{j}x^j\left(\frac{n+1}{2}-i\right)^{n-j} \tag{5.135}$$

$$= \frac{1}{n!}\sum_{j=0}^{n}\left((-1)^j\binom{n}{j}\sum_{i=0}^{\tilde{n}}(-1)^i\binom{n+1}{i}\left(\frac{n+1}{2}-i\right)^{n-j}\right)x^j \tag{5.136}$$

For $0 \leq j \leq n$, define the polynomial coefficients

$$D_{\tilde{n},j}^{(n)} = (-1)^j\binom{n}{j}\sum_{i=0}^{\tilde{n}}(-1)^i\binom{n+1}{i}\left(\frac{n+1}{2}-i\right)^{n-j}$$

$$= \frac{1}{2^{n-j}}\binom{n}{j}\sum_{i=0}^{\tilde{n}}(-1)^{i+j}\binom{n+1}{i}(n+1-2i)^{n-j} \tag{5.137}$$

so that

$$\beta^{(n)}(x) = \frac{1}{n!}\sum_{j=0}^{n} D_{\tilde{n},j}^{(n)} x^j. \tag{5.138}$$

The function $\beta^{(n)}$ is even and $(n-1)$-times differentiable, so that $\frac{d^j\beta^{(n)}}{dx^j}(0) = 0$ for any odd j such that $0 \leq j \leq n-1$. In other words, $D_{\tilde{n},j} = 0$ for $0 \leq j \leq n-1$, j odd. Applying (5.138) results in only $n+1-\tilde{n}$ terms (i.e., $\tilde{n}+1$ if n is even and $\tilde{n}+2$ if n is odd) in the sum instead of $n+1$ with (5.134).

In practice, the polynomial coefficients $(C_{l,j}^{(n)})_{0\leq l\leq\tilde{n}-1, 0\leq j\leq n}$ and $(D_{\tilde{n},j}^{(n)})_{0\leq j\leq n}$ are precomputed[4] before the indirect B-spline transform, as detailed in Algorithm 5.9. Then, the evaluation of $\beta^{(n)}(x)$ is done by Horner's method [30].

[4]Note that [83] provides a recursive algorithm, that is not used in our implementation, for computing these coefficients.

Algorithm 5.9: Coefficients of the piecewise polynomial expression of $\beta^{(n)}$

Input : Order n of the B-spline

Output: The coefficients $C_{k,j}^{(n)}$ and $D_{\tilde{n},j}^{(n)}$ for $j = 0 \dots n$, $k = 0 \dots \tilde{n} - 1$

1 Compute $\binom{n}{j}$ for $j = 0 \dots n$ using recursive formulas $\binom{n}{0} = 1$, $\binom{n}{j} = \frac{n-j+1}{j}\binom{n}{j-1}$ for $j = 1 \dots \tilde{n}$ and $\binom{n}{j} = \binom{n}{n-j}$ for $j > \tilde{n}$.

2 Compute $\binom{n+1}{i} = \frac{n+1}{i}\binom{n}{i-1}$ for $i = 1 \dots \tilde{n}$

3 Compute matrix P with $P_{ij} = j^i$ for $i = 0 \dots n$, $j = 1 \dots n+1$ using $P_{0,\cdot} = 1$ and $P_{i,j} = jP_{i-1,j}$ for $i \geq 1$

4 Compute $C_{k,j}^{(n)}$ for $j = 0 \dots n$, $k = 0 \dots \tilde{n} - 1$ applying (5.133)

5 Compute $D_{\tilde{n},j}^{(n)}$ applying (5.137).

5.6.3 Provided implementation

In the provided implementation[5] of Algorithm 5.7, the 2D B-spline interpolation can be performed for $0 \leq n \leq 16$. The order limitation is due to numerical errors in the computation of the poles and the kernel evaluation. We replace $\beta^{(n)}$ by $n!\beta^{(n)}$ and $\gamma^{(n)}$ by

$$\gamma'^{(n)} = \frac{\gamma^{(n)}}{n!} = \begin{cases} 2^n & n \text{ even} \\ 1 & n \text{ odd.} \end{cases} \tag{5.139}$$

This prevents numerical errors that could occur when n is large because of the useless renormalization by $n!$. The following entities are precomputed:

- the poles $(z_i)_{1 \leq i \leq \tilde{n}}$ as described in Section 5.6.1,
- the normalized B-spline coefficients $(C_{l,j})_{0 \leq l \leq \tilde{n}, 0 \leq j \leq n}$ as described in Section 5.6.2,
- the normalization constant $\gamma'^{(n)}$ defined in (5.139),
- the truncation indices $(N^{(i,\varepsilon')})_{1 \leq i \leq \tilde{n}}$ defined in (5.40) with $\varepsilon' = \frac{\rho^{(n)}\varepsilon}{2}$.

The first three items are tabulated for $n \leq 11$. For information purposes, $B^{(n)}$, $\gamma^{(n)}$, $\gamma'^{(n)}$ and the corresponding poles are displayed in Table 5.3 for $2 \leq n \leq 7$. Note that the computation are performed in double-precision floating-point format to prevent from round-off error. Multi-channel images, and in particular color images, are handled by applying the prefiltering algorithm on each channel independently, which gives the multi-channel B-spline coefficients. Then, the interpolated values are computed by applying the indirect B-spline transform to the multi-channel B-spline coefficients.

Homographic transformation. In the field of computer vision homographies [52, 85] are widely used to relate images of scene assimilable to planar surfaces (or when the camera motion is a rotation around the optical center). As a fundamental application of Algorithm 5.7 we provide an implementation of homographic (or projective) transformation of images. Given a 2D homography h and an image f of size $K \times L$, the homographic transformation of f by h is the image f_h of size $K' \times L'$ verifying

$$\forall (i,j) \in \{0, \dots, K'-1\} \times \{0, \dots, L'-1\}, \quad (f_h)_{i,j} = f\left(h^{-1}(i,j)\right). \tag{5.140}$$

It is done by applying Algorithm 5.7 at locations $\left(h^{-1}(i,j)\right)_{(i,j) \in \{0,\dots,K'-1\} \times \{0,\dots,L'-1\}}$. In the provided implementation the output image has the same size as the input, i.e., $K' = K$ and $L' = L$.

[5] http://www.ipol.im/pub/art/2018/221/

n	$B^{(n)}$	poles
2	$(z^{-1}+6+z)/8$	-0.1715728752538099
3	$(z^{-1}+4+z)/6$	-0.26794919243112281
4	$(z^{-2}+76z^{-1}+230z+76z+1)/384$	-0.36134122590021989
		-0.013725429297339109
5	$(z^{-2}+26z^{-1}+66z+26z+1)/120$	-0.4305753470999743
		-0.043096288203264443
6	$(z^{-3}+722z^{-2}+10543z^{-1}+23548+10543z+722z^2+z^3)/46080$	-0.48829458930303893
		-0.081679271076238694
		-0.0014141518083257976
7	$(z^{-3}+120z^{-2}+1191z^{-1}+2416+1191z+120z^2+z^3)/5040$	-0.53528043079643672
		-0.12255461519232777
		-0.0091486948096082266

Table 5.3: $B^{(n)}$ and the (approximate values of the) poles for $2 \leq n \leq 7$.

If the location $h^{-1}(i,j)$ does not belong to $[0,K-1]\times[0,L-1]$ the value $(f_h)_{i,j}$ is arbitrarily set to 0 to avoid extrapolation. The implementation inputs are:

- an image f,
- a homography h,
- a B-spline order $n \in \{0,\dots,16\}$,
- one of the four classical extensions of Table 5.1,
- a desired precision ε,
- a choice between the two proposed prefiltering algorithms (larger or exact domain).

Online Demo. The implementation is accompanied by an online demo where the user can upload an image and apply to it an homographic transformation. The choice of the homography is made by selecting the images of the four corners of the image. In Figure 5.4 we display an example of the online demo use which corresponds to the homographic transformation of the 512×512 *Lenna* image by the homography h defined by

$$\begin{cases} h(0,0) & = (25,13) \\ h(0,511) & = (11,500) \\ h(511,0) & = (480,12) \\ h(511,511) & = (468,482). \end{cases} \tag{5.141}$$

We recall that the pixels outside $[0,511]^2$ are arbitrarily set to 0.

Figure 5.4: Example of the online demo use. The *Lenna* image is transformed by the homography h defined in (5.141). We use order 11, the half-symmetric boundary condition, $\varepsilon = 10^{-6}$ and the exact domain.

5.7 Experiments

In this section we make several experiments using Algorithm 5.7. The input image used is a standard test image, *Lenna*, a gray-level image of size 512×512.

5.7.1 Computational Cost

The computational cost of the B-spline interpolation depends on the order n, the size of the input signal (and its dimension), the desired precision ε and the number of interpolated values. The total length of the extension has a dependency with respect to n and ε that is not straightforward so that it is difficult to express the complexity of the prefiltering step in general. It is given by $2L_0^{(n,\varepsilon)} = 2\left(\tilde{n} + \sum_{i=1}^{\tilde{n}} N^{(i,\varepsilon)}\right)$ in dimension one and $2L_0^{(n,\varepsilon')}$ in dimension two. We display in Table 5.4 and Table 5.5 the values of $2L_0^{(n,\varepsilon)}$ and $2L_0^{(n,\varepsilon')}$ for different order and precision values. We notice that the values are slightly greater in dimension two. Assuming that the length of the extension has the same order of magnitude as the input length we obtain in dimension one and two the complexities, independent of ε, presented in Table 5.6. The complexity of the indirect B-spline transform corresponds to the computation of a single interpolated value.

$n \setminus \varepsilon$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}
2	8	12	14	16	20	22	24	28	30	32	34
3	12	14	18	22	26	28	32	36	40	42	46
4	18	24	30	36	40	46	52	58	62	68	74
5	22	28	34	42	48	56	62	70	76	84	90
6	30	38	46	56	66	74	82	90	100	110	118
7	32	42	54	64	76	84	96	106	116	126	138
8	40	54	66	78	90	104	116	130	140	154	166
9	46	58	72	90	102	116	130	144	162	172	188
10	54	68	88	104	118	134	152	168	186	202	218
11	58	76	94	110	132	148	168	188	204	222	242

Table 5.4: Total length of extension $2L_0^{(n,\varepsilon)}$ for unidimensional signal in function of the order n and the precision ε.

$n \setminus \varepsilon$	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}
2	10	14	16	18	20	24	26	28	32	34	36
3	14	18	22	24	28	32	36	38	42	46	48
4	20	26	30	38	42	48	52	60	64	70	76
5	24	32	38	44	52	58	64	72	80	86	92
6	32	40	48	58	68	76	86	94	106	112	120
7	36	48	56	68	78	88	98	110	120	132	142
8	44	56	70	80	96	110	120	132	144	158	172
9	50	64	78	92	106	122	134	150	164	180	192
10	56	76	90	108	126	140	158	172	190	206	222
11	64	82	100	118	138	154	172	190	210	228	246

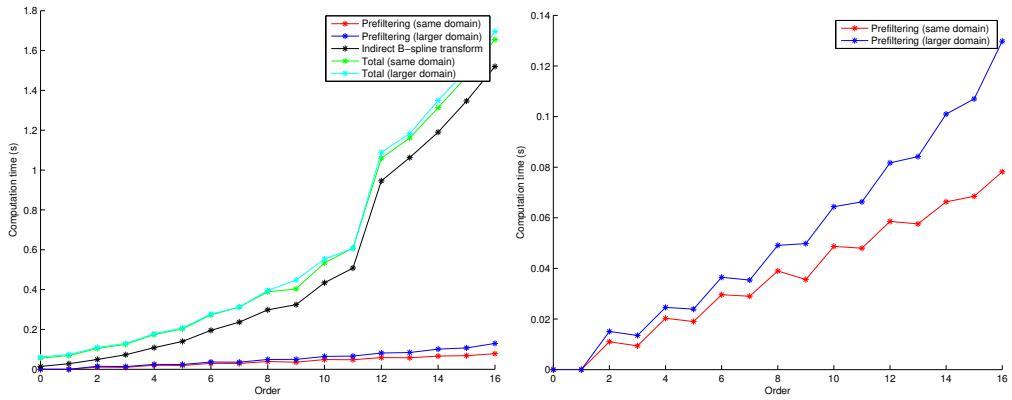
Table 5.5: Total length of extension $2L_0^{(n,\varepsilon')}$ for two-dimensional signals in function of the order n and the precision ε.

We verified empirically how the computation time of each step depends on the B-spline order n for a homographic transformation. As it depends neither on the extension choice nor

on the homography, we took the half-symmetric extension and the identity. In Figure 5.5 we display the computation times for $\varepsilon = 10^{-6}$. The prefiltering in the larger domain is more costly than the prefiltering in the same domain and the difference increases with the order. However in any case the prefiltering cost remains negligible with respect to the indirect B-spline transform cost. The strong increase between $n = 11$ and $n = 12$ in the indirect B-spline transform cost comes from the non-tabulation of the kernel.

	1D	2D
Prefiltering	$O(nK)$	$O(2nKL)$
Indirect B-spline transform	$O(n^2)$	$O(n^3)$

Table 5.6: Complexity of the B-spline interpolation algorithms for a signal of length K or an image of size $K \times L$ using order n. The complexity of the indirect B-spline transform corresponds to the computation of a single interpolated value.



(a) Global (b) Prefiltering

Figure 5.5: Computation time of the different B-spline interpolation steps for a homographic transformation of the 512×512 gray-level image *Lenna* with $\varepsilon = 10^{-6}$. The jump between $n = 11$ and $n = 12$ is due to the kernel not being tabulated in the code above order 11.

5.7.2 Computation Error

The computation error committed during the prefiltering step is estimated by checking if the interpolation condition (5.58) is verified. In practice it is done by comparing the initial image f with its homographic transformation by the identity f_{Id}.

We lead the computations for:

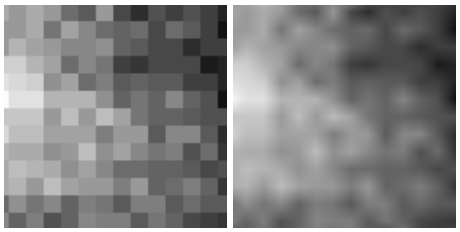
- the four boundary extensions of Table 5.1,
- any order $n \in \{2, \ldots, 16\}$,
- any precision $\varepsilon \in \{10^{-2}, \ldots, 10^{-12}\}$,
- the two prefiltering algorithms (only on the larger domain for the constant extension).

In all cases the computation error is less than ε i.e. $\|f - f_{\mathrm{Id}}\|_\infty \leq \varepsilon$. It empirically shows that the result of Theorem 5.2 is verified.

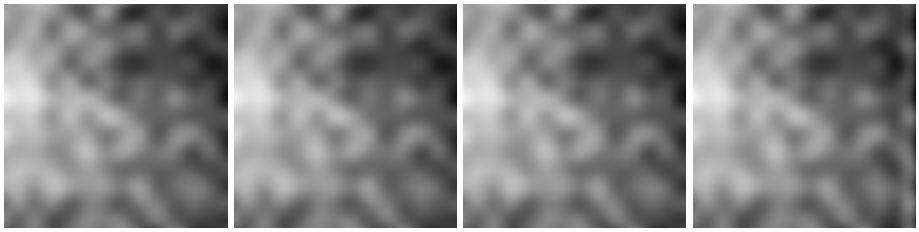
Which precision ε should be used? There is no recommended choice for ε because it depends on the context of application. There is clearly a trade-off between precision and computational cost. As the prefiltering step cost is negligible with respect to the indirect B-spline transform cost (see previous section) when a large amount of pixel values are interpolated, the computational cost aspect should not be taken into account by the user. For instance if the interpolated values are stored in single-precision floating-point format, the value $\varepsilon = 10^{-6}$ should be considered.

5.7.3 Zoom at the Boundary

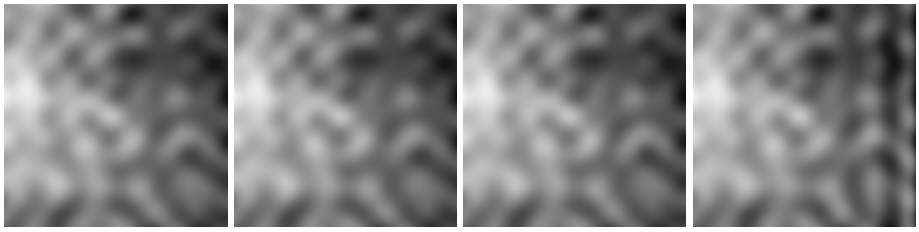
By zooming at one of the boundaries of an image using the B-spline interpolation we are able to highlight the influence of the boundary extension. We performed a zoom by a factor 20 using various orders and boundary conditions. It is computed using $\varepsilon = 10^{-6}$ and the prefiltering is done on a larger domain. In Figure 5.6 we display a small part of size 256×256 of the zoomed images that corresponds to the center of the right boundary. We see more and more details as the order increases. The boundary condition influence can be seen by comparing the right part of images. As an example we display in Figure 5.7 the comparison between the small images corresponding to order 16.



(a) Order 0 and order 1

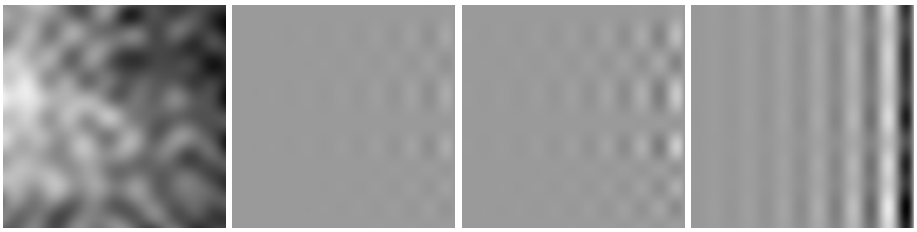


(b) Order 3



(c) Order 16

Figure 5.6: Zoom by a factor 20 (crop of size 256×256 centered in the middle of the right boundary) using B-spline interpolation for various orders and boundary conditions. It is computed using $\varepsilon = 10^{-6}$ and the prefiltering on a larger domain. From the left to the right we use the constant, half-symmetric, whole-symmetric and periodic extension. For orders 0 and 1 the result are the same with the four extensions. We see more and more details as the order increases. The boundary condition influence can be seen by comparing the right part of images. The affine transformation $y = 9x - 1080$ is applied to these images before visualization.



(a) Half-symmetric (b) Difference with constant (c) Difference with whole-symmetric (d) Difference with periodic

Figure 5.7: Comparison between the results of Figure 5.6 for order 16. The difference is made with the half-symmetric extension result. The affine transformation $y = 22x + 154$ is applied to the difference images before visualization.

5.7.4 Evolution of the Results with the Order of Interpolation

In this part, we analyze experimentally the evolution of the B-spline interpolation results with the order n. It is commonly admitted that the results of the cubic ($n = 3$) B-spline interpolation are sufficient and that increasing the order does not lead to significant improvements. We show the contrary and recommend, except when efficiency is primordial, to use a higher order B-spline interpolation. In the following, the computations are done using $\varepsilon = 10^{-6}$ and the prefiltering on a larger domain.

Comparison to the Shannon-Whittaker Interpolation. The B-spline interpolation approaches the Shannon-Whittaker interpolation when the order goes to infinity [5]. We empirically highlight this result by comparing the homographic transformations of the *Lenna* image by h (defined by (5.141)) obtained using respectively the B-spline interpolation and the Shannon-Whittaker interpolation. The comparison is done by computing the root mean square error (RMSE) between the central parts of the two resampled images. Since the boundary extension choice has no influence we choose the periodic extension. As explained in Chapter 3, the Shannon-Whittaker interpolation with periodic extension corresponds to the trigonometric polynomial interpolation in real convention (also called discrete Shannon interpolation in [1]). Algorithm 3.1 is used to resample the image. It is based on the nonequispaced fast Fourier transform (NFFT) algorithm [63], which is a much slower algorithm (see Chapter 6) than the B-spline interpolation. The decay of the error difference with the order n is visible in Figure 5.8.

Consistency of the B-spline Interpolation. In order to study the consistency of the B-spline interpolation we applied ten shifts of 0.1 pixels (in the horizontal direction) and then one shift of -1 pixel. The consistency measurement is then given as the RMSE between the central parts of the initial and output images. Note that in Chapter 6 we propose another, but similar, definition for the consistency measurement. As in practice the boundary condition influence on this measurement is negligible, we arbitrarily chose to use the half-symmetric boundary condition. The decay of the consistency measurement with the order n is displayed in Figure 5.9. It justifies the choice of a high order B-spline interpolation ($n = 11$ for instance) while it is commonly admitted that the cubic B-spline interpolation is sufficient. Note that the computational error is negligible with respect to the model error. In Figure 5.10 we display for $n \in \{0, 1, 3, 16\}$ the central parts of the difference images and the corresponding discrete Fourier transform modulus. The differences are localized in the high frequencies were the model error is higher. For $n = 0$ it's exactly the gradient of the image.

Comparison between Different Orders. We also compared the B-spline interpolation results for different orders. As in Section 5.7.4, we computed the homographic transformations of the *Lenna* image by h (defined by (5.141)) for different orders. The resampled images are compared

to the one corresponding to the maximal order available, i.e., $n = 16$. The comparison is done by computing the RMSE between the central parts of the images. As the boundary extension choice has no influence we chose the half-symmetric extension. The decay of the difference with the order n is visible in Figure 5.11. The average difference for the cubic B-spline is around one grey level and is three times smaller for order 11. In Figure 5.12 we display for $n \in \{1, 3, 11, 15\}$ the central parts of the resampled images and of the difference images (with the corresponding discrete Fourier transform modulus). As the order increases the resampled image becomes sharper. The interpolation kernel becomes closer to the cardinal sine so that less high-frequency content is attenuated.

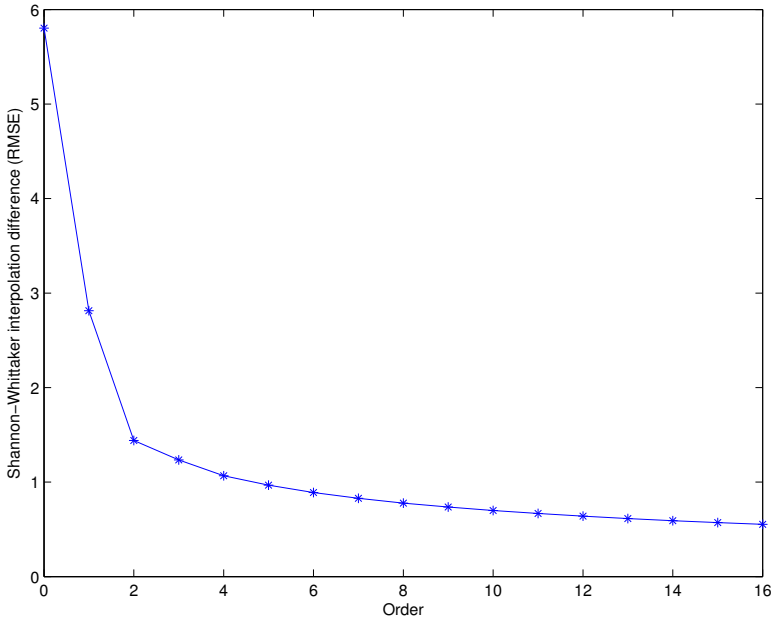


Figure 5.8: Decay of the difference between the Shannon-Whittaker interpolation and the B-spline interpolation for $n \in \{0,\dots,16\}$. The RMSE is taken on the central part of the images so that the boundary extension has a negligible influence. The periodic extension is used for both interpolation methods. The Shannon-Whittaker interpolation corresponds to the trigonometric polynomial interpolation with real convention introduced in Chapter 3.

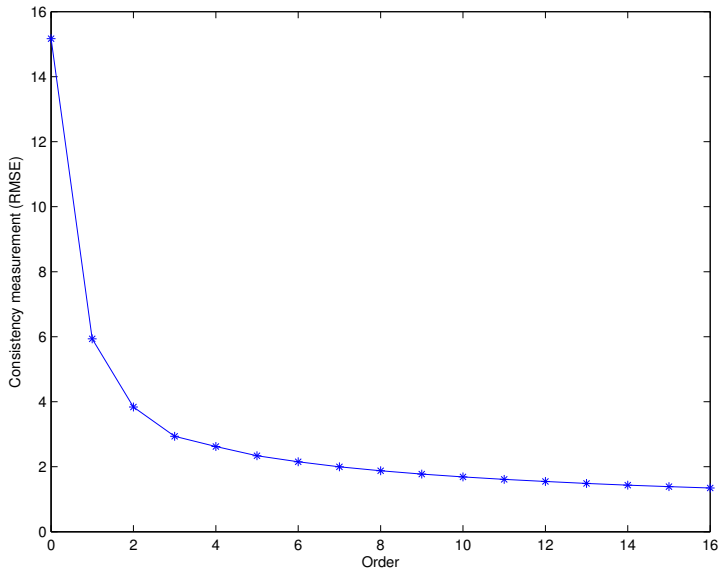
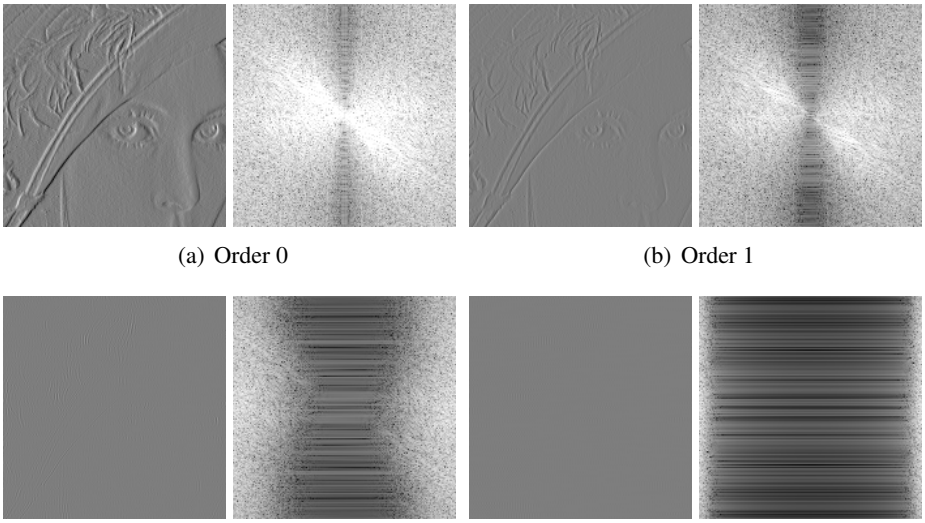


Figure 5.9: Decay of the error for the interpolation order $n \in \{0,\dots,16\}$ when performing ten successive translations by 0.1 pixel of an image and finally compensating with a -1 pixel translation. It justifies the choice of a high order B-spline interpolation ($n = 11$ for instance) while it is commonly admitted that the cubic B-spline interpolation is sufficient. Note that the computational error is negligible with respect to the model error.



(a) Order 0 (b) Order 1

(c) Order 3 (d) Order 16

Figure 5.10: Central parts of the difference images for the consistency tests and the corresponding unnormalized discrete Fourier transform modulus (in logarithmic scale $u \mapsto \log(1+u)$) for $n \in \{0,1,3,16\}$. The differences are localized in the high frequencies were the model error is higher. For $n = 0$ it is exactly the gradient of the image. We added 126 to the difference images and multiplied the spectrum modulus by 30 before visualization.

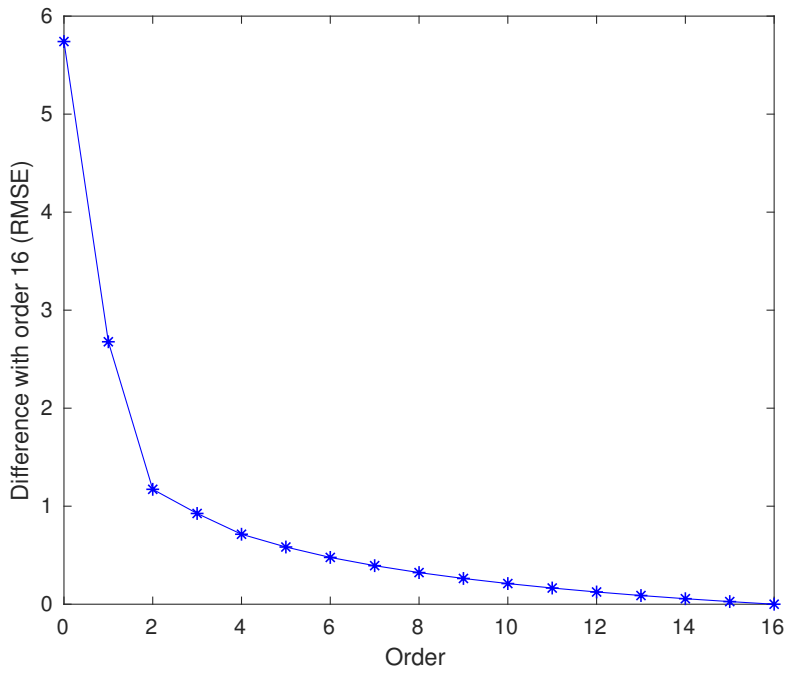
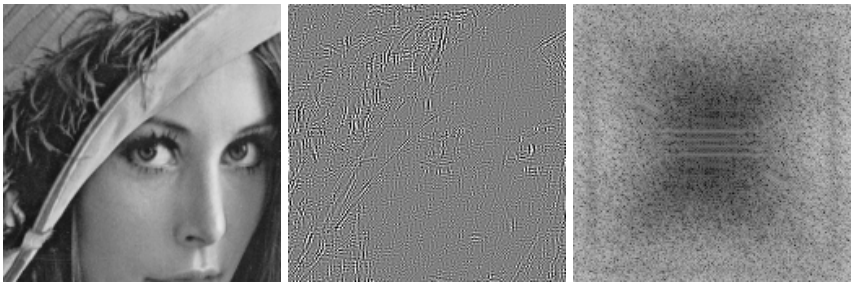
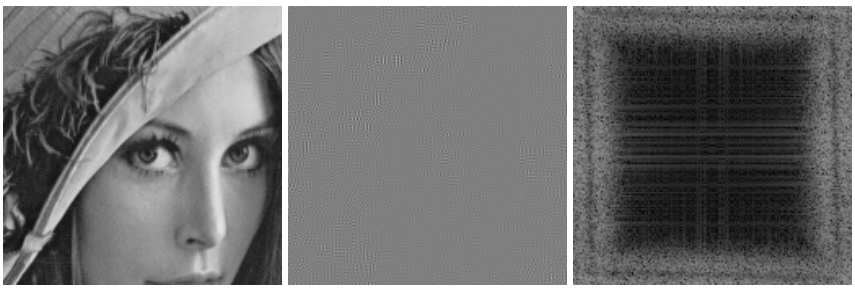


Figure 5.11: Decay of the difference between the B-spline interpolation of order $n \in \{0,\dots,16\}$ and the one of order 16. The comparison is done by computing the RMSE between the central parts of the images. As the boundary extension choice has no influence we chose the half-symmetric extension. The average difference for the cubic B-spline is around one grey level and is three times smaller for order 11.

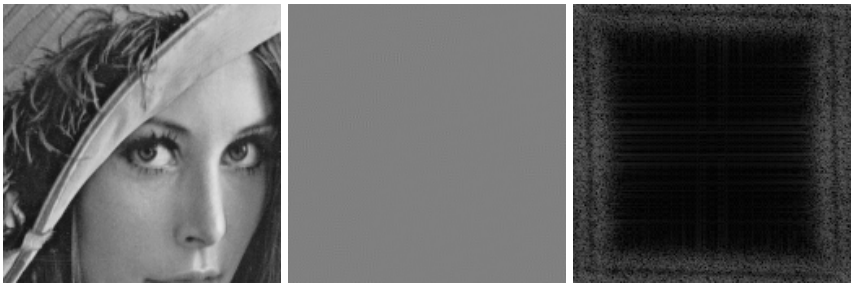
(a) Order 1



(b) Order 3



(c) Order 11



(d) Order 15

Figure 5.12: Comparison between resampled images obtained using $n \in \{0,1,3,15\}$ and the one using $n = 16$. Only the central parts of the resampled image (left) and of the difference image (center) are shown. The right image is 30 times the (unnormalized) discrete Fourier transform modulus (in logarithmic scale $u \mapsto \log(1+u)$) of the difference. The affine transformation $y = 50x + 128$ is applied to the difference images before visualization. As the order increases the resampled image becomes sharper. The interpolation kernel becomes closer to the cardinal sine so that less high-frequency content is attenuated.

5.8 Conclusion

In this chapter we presented the theory and practice to perform B-spline interpolation for any order, in particular in the case of images. It is based on the seminal two-step method proposed by Unser et al. in 1991 that uses linear filtering and for which the computational error is not controlled and the boundary extension is fixed.

The two proposed prefiltering algorithms require additional computations to handle correctly any boundary extension. We prove theoretically and experimentally that the computational errors are controlled (up to dimension two). The first algorithm is general and works for any boundary extension while the second is applicable under specific assumptions. The global interpolation algorithm remains efficient because the computational cost increases slowly with the precision (which can be set to the single precision in most of the applications).

In an experimental part we showed that increasing the order of the B-spline interpolation improves the interpolation quality. When efficiency is not primordial, a high order B-spline interpolation must be preferred to cubic B-spline interpolation.

In addition, we provide a detailed description, and the corresponding implementation, of how to evaluate the B-spline kernel and to compute the B-spline interpolator parameters. As a fundamental application we also provide an implementation of homographic transformation of images using B-spline interpolation.

Chapter 6

Consistency of Interpolation Methods: Definition and Improvements

Abstract

There is no universal procedure for evaluating the quality and performance of an interpolation method. In this chapter, we introduce a new quantity: the consistency measurement. For a given image, it measures the error after applying successively a transformation, a crop (removing boundary artefacts) and the inverse transformation. An average over random transformations is made to remove the dependency on the transformation. A more precise measurement discarding very high-frequency artefacts is obtained by clipping the spectrum of the difference. We also propose new fine-tuned interpolation methods that are based on the DFT zoom-in and pre-existing interpolation methods. The zoomed version of an interpolation method is obtained by applying it to the DFT zoom-in of the image. In the periodic plus smooth version of interpolation methods, the non-periodicity is handled by applying the zoomed version to the periodic component and a base interpolation method to the smooth component. In an experimental part, we show that the proposed fine-tuned methods have better consistency measurements than their base interpolation methods and that the error is mainly localized in a small high-frequency band. We recommend to use the periodic plus smooth versions of high order B-spline. It is more efficient and provides better results than trigonometric polynomial interpolation. In Chapter 10, such a method will be used for generating synthetic data and for the burst denoising method.

Contents

6.1 Introduction

Interpolation is one of the most useful tools in image processing since the evaluation of an image at subpixel locations is required in many algorithms. The overall performances are directly impacted and it is not always possible to neglect the error introduced. For instance, assume the generation of a dataset involves an interpolation step. Then, the interpolation error has to be taken into account during the performance evaluation of an algorithm on these synthetic data. In particular, this will influence the choice of the interpolation method used for generating synthetic data and for the burst denoising method in Chapter 10.

There is no universal procedure for evaluating the quality and performance of an interpolation method. A non-rigourous first possibility is to have a qualitative approach with a visual analysis of the results. A second standard procedure consists in interpolating samples for which an underlying continuous signal is known. However there is no satisfactory and practical model for real-world images. A third procedure, which does not require a ground truth model, is to measure the error after applying successively several transformations whose composition is the identity (for instance a transformation and its inverse). The accuracy of the interpolation method is not directly measured but this gives an information about the consistency (or stability) of the interpolation method. For instance, in [123, Section VIII] they apply 15 rotations of angle $\frac{2\pi}{15}$. In [69], they perform forward and backward image transformations using optical flow.

Even though the Shannon-Whittaker formula cannot be used in practice, images are still commonly assumed to be bandlimited (at least theoretically). The interpolation error is interpreted as the consequence of the cardinal sine approximation by the interpolation kernel. The Fourier transform of the kernel is, in general, a smooth approximation of the Nyquist domain indicator function [46, 123]. In practice, for finite images, the Shannon-Whittaker interpolate becomes a trigonometric polynomial when the bandlimited assumption is made along with a periodic extension [1]. As shown in Chapter 3, trigonometric polynomial interpolation is compatible with DFT-based computations and can be used in practice. It provides high quality results because it does not introduce spectral attenuations or aliasing in the high-frequencies. As pointed out in [1], it is possible to improve the performance of interpolation methods using a DFT zoom-in i.e. an upsampling by trigonometric polynomial interpolation. However, the periodic assumption is not compatible in general with the image content and this may introduce undesirable ringing artefacts. In [84], the periodic plus smooth decomposition is applied for handling the non-periodicity during the up-sampling of images. The periodic component is upsampled using a DFT zoom-in while the smooth component is interpolated by bilinear interpolation.

In this chapter, we introduce a new quantity for evaluating the interpolation method quality: the consistency measurement. For a given image and a geometric transformation, the error after applying successively the transformation, a crop (removing boundary artefacts) and the inverse transformation is measured. The consistency measurement is obtained by averaging the error over random transformations. A more precise measurement discarding very high-frequency artefacts is obtained by clipping the spectrum of the difference. We also propose new fine-tuned interpolation methods that are based on the DFT zoom-in and pre-existing interpolation methods. The zoomed version of an interpolation method is obtained by applying it to the DFT zoom-in of the image. In the periodic plus smooth version of interpolation methods, the non-periodicity is handled by applying the zoomed version to the periodic component and a base interpolation method to the smooth component. In an experimental part, we show that the proposed fine-tuned methods have better consistency measurements than their base interpolation methods and that the error is mainly localized in a small high-frequency band.

This chapter is organized as follows. First, the consistency measurements are defined in Section 6.2. Then, the new fine-tuned methods are introduced in Section 6.3. Finally, the per-

formances of these methods are experimentally evaluated in Section 6.4.

6.2 Consistency Measurements of Interpolation Methods

In the rest of this chapter, $\underline{u}$ denotes a real-valued image of size $M \times N$ and $\varphi \in \sigma(\mathbb{R}^2)$ is a geometric transformation. The quantity $u(x,y)$ denotes the interpolated value at location $(x,y) \in \mathbb{R}^2$ of $\underline{u}$ obtained with an interpolation method and a boundary extension (specified in the context). The transformation of $\underline{u}$ by φ is noted $\underline{u}_\varphi$ and is defined by

$$\forall (k,l) \in \Omega_{M,N}, \quad (\underline{u}_\varphi)_{k,l} = u(\varphi^{-1}(k,l)). \tag{6.1}$$

Two variants of the consistency measurements of an interpolation method are introduced in Section 6.2.2. But first, spectrum clipping is presented in Section 6.2.1 since one of the consistency measurements relies on it.

6.2.1 Spectrum Clipping

Let $r \in [0,1]$. The spectrum of the image $\underline{v}$ of size $M \times N$ is clipped with ratio r as follows. For L a positive integer, denote by $c_{L,r}$ the piece-wise constant function defined by

$$c_{L,r}(x) = \begin{cases} 1 & |x| \leq (1-r)L/2, \\ 0 & |x| > (1-r)L/2. \end{cases} \tag{6.2}$$

The image $\underline{v}^c$, whose spectrum corresponds to the spectrum of $\underline{v}$ clipped with ratio r, is the image of size $M \times N$ defined by

$$\mathcal{F}_{M,N}(\underline{v}^c)_{m,n} = \mathcal{F}_{M,N}(\underline{v})_{m,n} c_{M,r}(m) c_{N,r}(n). \tag{6.3}$$

Spectrum clipping can be seen as a (perfect) lowpass filtering operation. Along each dimension only the frequencies lower or equal to $(1-r)\pi$ are kept. The coefficient $\mathcal{F}_{M,N}(\underline{v})_{m,n}$ is killed if and only if $\frac{2\pi m}{M} > (1-r)\pi$ or $\frac{2\pi n}{N} > (1-r)\pi$ if and only if $c_{M,r}(m)c_{N,r}(n) = 0$. Note that because of the Parseval Theorem, spectrum clipping reduces the energy (i.e. the root mean square) of the images.

To simplify, the dependency of $\underline{v}^c$ on r is skipped in the notation. For $r = 0$ spectrum clipping has no impact i.e. $\underline{v}^c = \underline{v}$. For $r = 1$, $\underline{v}^c$ is the constant image corresponding to the mean of $\underline{v}$. An example of spectrum clipping with $r = 5\%$ is shown in Figure 6.1.

6.2.2 Definition and Evaluation

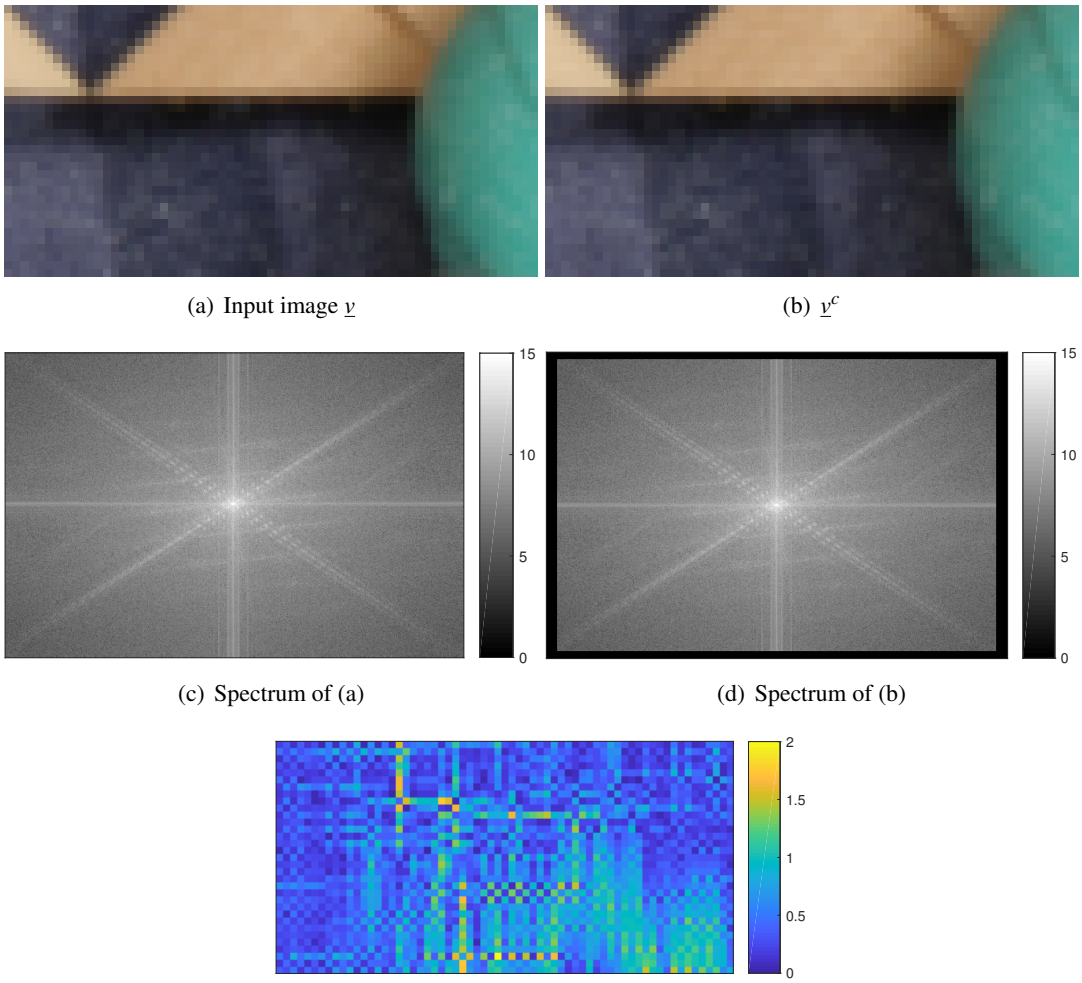
Let δ be a non-negative integer that is assumed to be small with respect to the size of the image (e.g. $\delta = 20$). Denote by C_δ the crop operator of size δ so that $C_\delta(\underline{u})$ is the image of size $(M-2\delta) \times (N-2\delta)$ verifying

$$(C_\delta(\underline{u}))_{k,l} = \underline{u}_{k+\delta,l+\delta}. \tag{6.4}$$

The image $\tilde{\underline{u}}(\varphi) = \tilde{\underline{u}}(\varphi,\delta)$ is defined as the image of size $(M-2\delta) \times (N-2\delta)$ verifying

$$\tilde{\underline{u}}(\varphi) = \left(C_\delta(\underline{u}_\varphi)\right)_{\varphi^{-1}}. \tag{6.5}$$

The image $\tilde{\underline{u}}(\varphi)$ represents the reconstruction of the image $\underline{u}$ after successively applying the transformation φ and its inverse φ^{-1}. For $\delta = 0$, $\tilde{\underline{u}}(\varphi) = \tilde{\underline{u}}(\varphi,0) = (\underline{u}_\varphi)_{\varphi^{-1}}$. For $\delta > 0$, the transformation φ^{-1} is applied to the cropped transformed image $C_\delta(\underline{u}_\varphi)$. The aim of the crop is to get rid of boundary artefacts introduced during the first transformation.



(a) Input image $\underline{v}$

(b) $\underline{v}^c$

(c) Spectrum of (a)

(d) Spectrum of (b)

(e) Difference between (a) and (b)

Figure 6.1: Example of spectrum clipping using $r = 5\%$ on the *RubberWhale* image. Only details of the images are shown in (a), (b) and (e). The difference (e) between (a) and (b) is not visible to the naked eye. The spectrums in (c) and (d) correspond to the unnormalized discrete Fourier transform in logarithmic scale $u \mapsto \log(1+u)$. The root mean square over the channels is shown in (c)-(d)-(e). The RMSE between $\underline{v}$ and $\underline{v}^c$ is approximately 0.80. The energy (i.e the root mean square) of $\underline{v}$ is around 143 so that the ratio of energy removed is approximately of 0.6%.

The difference image $DI(\varphi) = DI(\varphi, \underline{u}, \delta)$ is defined as the image of size $(M-4\delta) \times (N-4\delta)$ verifying

$$DI(\varphi) = C_\delta(\tilde{\underline{u}}(\varphi)) - C_{2\delta}(\underline{u}). \tag{6.6}$$

$DI(\varphi)$ is nothing but the cropped difference image between the reconstructed image $\tilde{\underline{u}}(\varphi)$ and the input image $\underline{u}$. Here, the aim of the crop is to prevent from considering the boundary artefacts introduced during the second transformation.

Consistency measurement for a transformation. The consistency measurement for the transformation φ, noted $\mathcal{CM}(\varphi)$, is defined as the ℓ^2 energy of the difference image $DI(\varphi)$. It is given by

$$\mathcal{CM}(\varphi)^2 = \frac{1}{(M-4\delta)(N-4\delta)} \sum_{k=0}^{(M-1-4\delta)} \sum_{l=0}^{(N-1-4\delta)} |DI(\varphi)_{k,l}|^2. \tag{6.7}$$

For $\delta = 0$, it corresponds to the RMSE between the reconstructed image $\tilde{\underline{u}}(\varphi)$ and $\underline{u}$. Note that to simplify the dependencies on the image $\underline{u}$, the interpolation method and δ are not specified.

Clipped consistency measurement for a transformation. A variant is obtained by replacing $DI(\varphi)$ by its clipped version $DI(\varphi)^c$. The spectrum clipping of $DI(\varphi)$ is done with ratio $r = 5\%$ as described in Section 6.2.1. This variant is called the clipped consistency measurement for the transformation φ and is denoted by $\mathcal{CM}^c(\varphi)$. Both measurements are linked by the inequality $\mathcal{CM}^c(\varphi) \leq \mathcal{CM}(\varphi)$ by Parseval's theorem.

Consistency measurements. The quantity $\mathcal{CM}(\varphi)$ highly depends on the transformation φ. First, the dependency is reduced by considering transformations of the same type (e.g. translations, affinities or homographies) that are moderate i.e. close to the identity. Secondly, it is reduced by averaging over several random transformations as follows. Consider a random list $(\varphi_i)_{1\leq i\leq N_{\text{transf}}}$ (e.g. $N_{\text{transf}} = 1000$) of moderate transformations of the same type. The consistency measurement, noted $\mathcal{CM}$, is defined as the average of the consistency measurements over the transformations φ_i,

$$\mathcal{CM} = \frac{1}{N_{\text{transf}}} \sum_{i=1}^{N_{\text{transf}}} \mathcal{CM}(\varphi_i). \tag{6.8}$$

It is clear that $\mathcal{CM}$ is a random variable whose value depends on the random transformations $(\varphi_i)_{1\leq i\leq N_{\text{transf}}}$ (and the way they are generated) but for N_{transf} large enough it should be close enough to the expected value. The clipped variant $\mathcal{CM}^c$ is naturally defined by using the clipped measure in (6.8). The evaluation of the consistency measurements is summarized in Algorithm 6.1.

Algorithm 6.1: Evaluation of the consistency measurements

Input : An image $\underline{u}$, a sequence of transformations $(\varphi_i)_{1\leq i\leq N_{\text{transf}}}$, a non-negative integer δ, an interpolation method and a boundary condition

Output: The consistency measures $\mathcal{CM}$ and $\mathcal{CM}^c$

1 **for** $i \in \{1, \ldots, N_{transf}\}$ **do**
2 Compute $C_\delta(\underline{u}_{\varphi_i})$ using the interpolation method and the boundary condition
3 Compute $\tilde{\underline{u}}(\varphi_i) = \left(C_\delta(\underline{u}_{\varphi_i})\right)_{\varphi_i^{-1}}$ using the interpolation method and the boundary condition
4 Compute the difference image $DI(\varphi_i) = C_\delta(\tilde{\underline{u}}(\varphi_i)) - C_{2\delta}(\underline{u})$
5 Compute the clipped difference image $DI^c(\varphi_i)$ with $r = 5\%$ as described in Section 6.2.1
6 Compute the consistency measurements $\mathcal{CM}(\varphi_i)$ and $\mathcal{CM}^c(\varphi_i)$ for the transformation φ_i using (6.7) and its clipped variant
7 Compute the consistency measurements $\mathcal{CM}$ and $\mathcal{CM}^c$ by averaging as in (6.8)

6.3 Fine-tuned Interpolation Methods

In this section fine-tuned interpolation methods are built from pre-existing methods, which are called "base" methods. They rely on the DFT zoom-in i.e. the upsampling by trigonometric polynomial interpolation. In the following we use the trigonometric polynomial interpolator in real convention. See Chapter 3 for more details.

6.3.1 Zoomed Version of Interpolation Methods

As pointed out in [1], it is possible to improve the performance of interpolation methods using a DFT zoom-in. Consider a base interpolation method and let us describe the corresponding zoomed version. Let $\lambda \geq 2$ be an integer representing a zoom factor and denote by $\underline{u}^\lambda$ the DFT zoom-in of $\underline{u}$ by factor λ. The interpolate of $\underline{u}$ using the zoomed method is defined as the rescaling by factor $\frac{1}{\lambda}$ of the interpolate of $\underline{u}^\lambda$ using the base method. In other words, for $(x,y) \in \mathbb{R}^2$, the interpolated value $u(x,y)$ is given by

$$u(x,y) = u^\lambda(\lambda(x,y)) \tag{6.9}$$

where $\underline{u}^\lambda$ is interpolated at location $\lambda(x,y)$ using the base interpolation method. The geometric transformation of an image using a zoomed interpolation method is described in Algorithm 6.2.

Algorithm 6.2: Transformation of an image using a zoomed interpolation method

Input : An image $\underline{u}$ of size $M \times N$, a zoom factor λ, a base interpolation method, a boundary extension and a transformation φ

Output: A transformed image $\underline{u}_\varphi$

1 Compute the locations $\left\{\varphi^{-1}(k,l)\right\}_{(k,l)\in\Omega_{M,N}}$

2 Compute $\underline{u}^\lambda$ the DFT zoom-in of $\underline{u}$ by factor λ

3 Compute $(\underline{u}_\varphi)_{k,l} = u^\lambda(\lambda\varphi^{-1}(k,l))$ on $\Omega_{M,N}$ using the base interpolation method with the boundary extension

The zoomed version is computationally less efficient than its base interpolation method since it requires to zoom the image and to compute the representation of the zoomed image for the base interpolation method. As shown in Section 6.4, the zoomed versions are more consistent than their base methods. Intuitively, they are closer to the discrete Shannon interpolation method and the base interpolation method is applied to a well-sampled image. However the DFT zoom-in relies on trigonometric polynomial interpolation and assumes a periodic extension, which may not be adapted to the image content and may introduce ringing artefacts.

6.3.2 Periodic plus Smooth Decomposition Version of Interpolation Methods

The *periodic plus smooth* decomposition of [84] is useful for handling the non-periodicity of images [1, 17, 43]. The image $\underline{u}$ is decomposed into a periodic component $\underline{p}$, which is almost periodic, and a smooth component $\underline{s}$, which varies slowly. The images $\underline{p}$ and $\underline{s}$ (of size $M \times N$) are linked by the relation $\underline{u} = \underline{p} + \underline{s}$ and $\underline{p}$ has the same mean as $\underline{u}$. More precisely, the decomposition is characterized and computed as follows. Denote by $\underline{v}_1$ and $\underline{v}_2$ the two images of size $M \times N$ defined by

$$(\underline{v}_1)_{k,l} = \begin{cases} \underline{u}(M-1-k,l) - \underline{u}(k,l) & \text{if } k = 0 \text{ or } k = M-1, \\ 0 & \text{otherwise,} \end{cases} \tag{6.10}$$

and

$$(\underline{v}_2)_{k,l} = \begin{cases} \underline{u}(k,N-1-l) - \underline{u}(k,l) & \text{if } l = 0 \text{ or } l = N-1, \\ 0 & \text{otherwise.} \end{cases} \tag{6.11}$$

Set $\underline{v} = \underline{v}_1 + \underline{v}_2$. Denote by Δ the discrete Laplacian operator that associates to an image $\underline{w}$ of size $M \times N$ the image $\Delta\underline{w}$, of the same size, defined by

$$\Delta\underline{w} = K_{\text{lap}} \star \underline{w} \tag{6.12}$$

where $\star$ represents the (M,N)-periodic convolution operator and

$$K_{\text{lap}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{6.13}$$

Then, the periodic component $\underline{p}$ of $\underline{u}$ is defined as the unique solution of the discrete Poisson equation

$$\begin{cases} \Delta \underline{p} = \Delta \underline{u} - \underline{v}, \\ \text{mean}(\underline{p}) = \text{mean}(\underline{u}). \end{cases} \tag{6.14}$$

In practice the decomposition is obtained using DFT-based computations. Indeed, the DFT of the smooth component $\underline{s}$ is given by $\mathcal{F}_{M,N}(\underline{s})_{0,0} = 0$ (zero mean) and

$$\forall (m,n) \in \hat{\Omega}_{M,N} \setminus (0,0), \quad \mathcal{F}_{M,N}(\underline{s})_{m,n} = \frac{\mathcal{F}_{M,N}(\underline{v})_{m,n}}{2\cos\left(\frac{2\pi m}{M}\right) + 2\cos\left(\frac{2\pi n}{N}\right) - 4}. \tag{6.15}$$

An example of decomposition is shown in Figure 6.2. The cross structure in the spectrum of $\underline{u}$, which is caused by its non-periodicity, is not visible in the spectrum of $\underline{p}$.

In [84], this periodic plus smooth decomposition is used to up-sample images. More generally, the decomposition can be used to improve interpolation methods as follows. The periodic component $\underline{p}$ is almost periodic so that the periodic extension of its DFT zoom-in does not introduce indesirable ringing artefacts. It is interpolated using a zoomed version of an interpolation method with periodic extension. The smooth component $\underline{s}$ varies slowly and is easily interpolated by a base interpolation method with any extension. Each component is interpolated independently, possibly using different base interpolation methods, and the results are added to get the interpolated values. The transformation of an image using the periodic plus smooth version of interpolation methods is described in Algorithm 6.3. As the periodic plus smooth decomposition is computed using DFT-based computations (see (6.15)), in practice the DFT zoom-in of $\underline{p}$ is obtained directly from $\mathcal{F}_{M,N}(\underline{p})$, which saves the computations of a DFT and an iDFT.

Algorithm 6.3: Transformation of an image using the periodic plus smooth version of interpolation methods

Input : An image $\underline{u}$, a zoom factor λ, two base interpolation methods, a boundary condition and a transformation φ

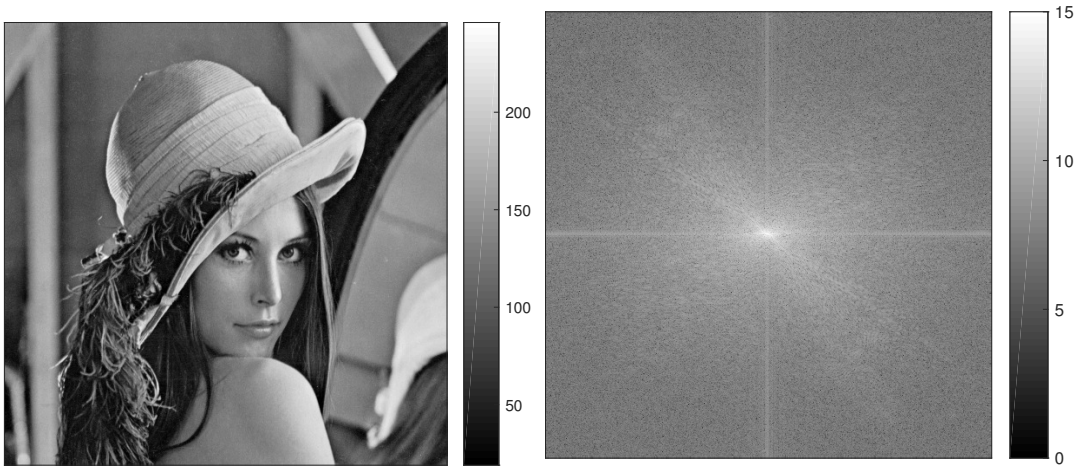
Output: A transformed image $\underline{u}_\varphi$

1 Compute the periodic plus smooth decomposition $\underline{p} + \underline{s}$ of $\underline{u}$ using (6.15)

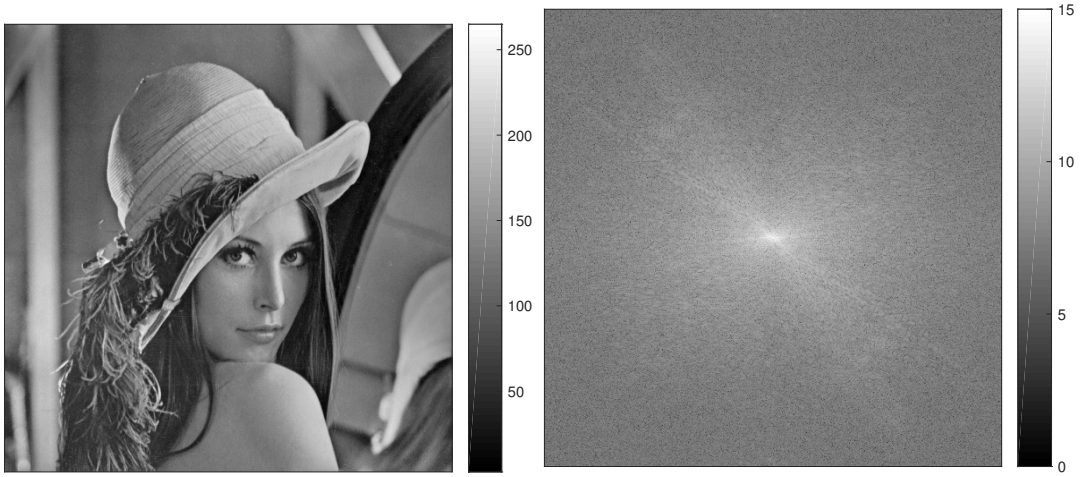
2 Compute the transformed periodic component $\underline{p}_\varphi$ using Algorithm 6.2 with factor λ and the first base interpolation method with periodic extension

3 Compute the transformed smooth component $\underline{s}_\varphi$ using the second base interpolation method and the boundary condition

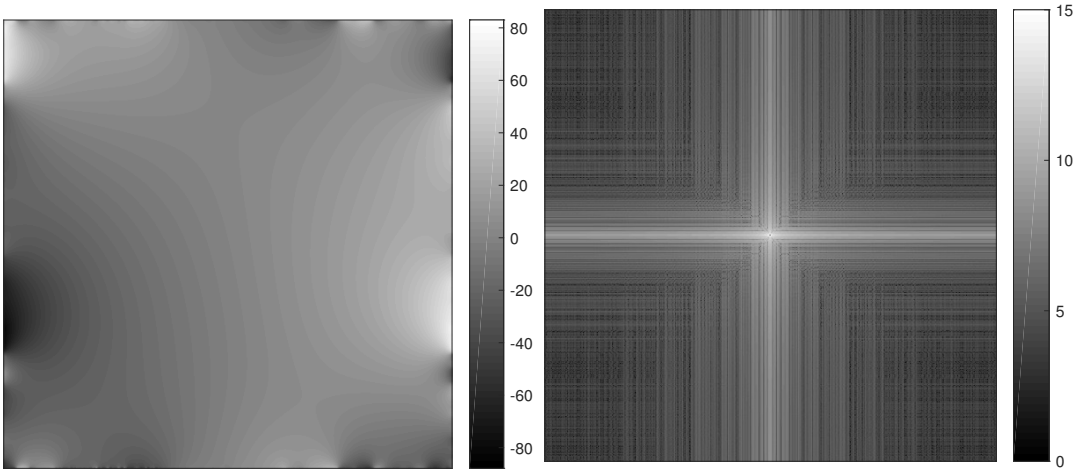
4 Compute $\underline{u}_\varphi = \underline{p}_\varphi + \underline{s}_\varphi$



(a) Input image $\underline{u}$ (b) Spectrum of $\underline{u}$



(c) Periodic component $\underline{p}$ (d) Spectrum of $\underline{p}$



(e) Smooth component $\underline{s}$ (f) Spectrum of $\underline{s}$

Figure 6.2: Periodic plus smooth decomposition $\underline{u} = \underline{p} + \underline{s}$ of the Lena image. The spectrums in (b), (d) and (f) correspond to the unnormalized discrete Fourier transform in logarithmic scale $u \mapsto \log(1+u)$. The content of the periodic component p is almost periodic and is similar to the input except near the image boundaries. The smooth component $\underline{s}$ varies slowly and is close to 0 in the center of the image. The cross structure in the spectrum of $\underline{u}$, which is caused by the non-periodicity, is not visible in the spectrum of $\underline{p}$.

6.4 Experiments

In this experimental part, interpolations methods are compared in terms of consistency and efficiency. The interpolation methods used are the following:

- Five base interpolation methods: trigonometric polynomial interpolation (in real convention), bicubic interpolation and B-spline interpolations of order 1, 3 and 11. They are noted respectively *tpi*, *bic*, *spline1*, *spline3* and *spline11*. Note that *spline1* corresponds to the so-called bilinear interpolation.
- The zoomed versions, with zoom factor $\lambda = 2$, of the base interpolation methods (except for *tpi* because by construction the DFT zoom-in is useless). The suffix *z2* is added to the corresponding base method.
- The periodic plus smooth versions, with zoom factor $\lambda = 2$, of several pairs of interpolation methods. The notation of the method starts with *p+s*, then the second term refers to the interpolation method used for the periodic component and an eventual third term corresponds to the interpolation method used for the smooth component (if different from the method used for the periodic component). Note that the DFT zoom-in is not applied for the *p+s-tpi-**** methods.

For example the periodic plus smooth version of the B-spline interpolation of order 11 and the bicubic interpolation is noted *p+s-spline11-bic*. The half-symmetric extension is used for all the methods except the *tpi* for which the use of the periodic extension is required. The performance evaluation was made using the crop size $\delta = 20$, so that the four classical boundary extensions proposed in Table 5.1 provide the same results.

The conclusions on the order of the methods are clear-cut and do not depend on the choice of the image. The image $\underline{u}$ considered is the *RubberWhale* image, which is a generic color image of size 584×388 taken from the Middlebury database [10]. Only details on a small part of the images are shown. For the difference and spectrum images, the displayed images correspond to the root mean square over the channels. The computations were made using an Intel(R) Xeon(R) CPU E5-2650 @ 2.60GHz.

6.4.1 Evaluation of the Consistency Measurements

In this section, the consistency measurements are evaluated for the different interpolation methods using Algorithm 6.1 with $N_{\text{transf}} = 1000$ moderate random homographies. The homographies were generated by moving the four corners of the image. More precisely we used Algorithm 6.4 with $L = 1$.

Algorithm 6.4: Generation of a random homography

Input : A size $M \times N$ and an integer $L > 0$.
Output: A random homography h.

1 **for** $i \in \{1,2,3,4\}$ **do**
2 Generate a couple of real numbers D_i according to the uniform law in $[-L,L]^2$
3 Define $C_1 = (0,0)$, $C_2 = (M-1,0)$, $C_3 = (0,N-1)$ and $C_4 = (M-1,N-1)$ the four corners and $C'_i = C_i + D_i$ the four transformed corners
4 Compute the unique homography h corresponding to the four pairs (C_i, C'_i)

The results are presented in Table 6.1. It contains the two consistency measurements for the considered interpolations methods and the computation time, in seconds, for all the transformations. Therefore the displayed time can also be interpreted as the average time, in milliseconds,

required to apply a transformation and its inverse. The B-spline interpolation of order 3 was more efficiently implemented than the bicubic one, which explains why the computation time is lower while the complexity is higher.

	$\mathcal{CM}$	$\mathcal{CM}^c$	Time (s)
spline1	2.53480	2.50077	225
bic	1.18612	1.13573	336
spline3	0.75702	0.67967	283
spline11	0.35355	0.19626	796
tpi	0.20564	0.06345	4767
spline1-z2	0.86002	0.81810	1308
bic-z2	0.25140	0.15072	1425
spline3-z2	0.20585	0.06816	1407
spline11-z2	0.20564	0.06345	2053
p+s-spline1	0.84660	0.81640	2039
p+s-spline1-bic	0.84660	0.81640	2159
p+s-bic	0.18008	0.13811	2355
p+s-spline3	0.09417	0.03007	2199
p+s-spline3-bic	0.09417	0.03007	2259
p+s-spline11	0.08785	0.01506	3360
p+s-spline11-bic	0.08785	0.01506	2907
p+s-tpi-spline1	0.08785	0.01506	5865
p+s-tpi-bic	0.08785	0.01506	5960
p+s-tpi-spline3	0.08785	0.01506	5926
p+s-tpi-spline11	0.08785	0.01506	6443

Table 6.1: Evaluation of consistency measurements for the *RubberWhale* image and $N_{\text{transf}} = 1000$ random homographies. A crop of size $\delta = 20$ was used so that the results do not depend on the boundary extension choice. The displayed time corresponds to the computation time, in seconds, for all the transformations. This can also be interpreted as the average time, in milliseconds, required to apply a transformation and its inverse.

Base interpolation methods. Except for the *tpi*, the consistency measurements for the base interpolation methods are the worst and spectrum clipping did not reduce significantly the value. The measurements for the *tpi* are better and the errors are mainly localized in the high frequencies since the clipped consistency measure is way lower. The resulting ranking is *tpi, spline11, spline3, bic, spline1*. In particular increasing the B-spline order leads to better results as already seen in Chapter 5.

Zoomed versions. The zoomed versions have better consistency measurements than their corresponding base methods but are slower (by a factor between 3 and 8). As the clipped consistency measurement is significantly lower (except for the *spline1-z2*), we deduce that the error is mainly localized in the high frequencies. The zoomed methods are closer to the *tpi*. In particular, using *spline11-z2* provides almost the same results as the *tpi* but is two times faster.

Periodic plus smooth versions. The periodic plus smooth versions have the best consistency measurements. This is due to the proper handling of the non-periodicity during the DFT zoom-in. The base method used for interpolating the zoomed periodic component has to be of high order (here *tpi* or *spline11*). The method used for the smooth component can be any base method (except the *tpi*) since it varies smoothly and is easily interpolable. No distinction can be made in the results. Indeed, the best results are obtained using *p+s-spline11-*** or *p+s-tpi-*** methods.

The *p+s-spline11-**** methods are way more efficient (three to four times here) and should be preferred.

Clipped consistency measurement. As already remarked above, for the fined-tuned methods the clipped consistency measurement $\mathcal{CM}^c$ is significantly lower than the non-clipped consistency measurement $\mathcal{CM}$. The amount of energy removed from $\underline{u}$ only represents 0.6% of its total energy. This means that the error is mainly localized in the high frequencies for the fined-tuned methods but not for the base methods (except the *tpi*).

6.4.2 Transformation by a Homography

In this section, the analysis of Section 6.4.1 will be confirmed visually by considering the transformation by an arbitrary homography. We define the homography φ by moving respectively the top-left corner, the top-right corner, the bottom-left and the bottom-right corner by $(1,1)$, $(-1,-1)$, $(0,0)$ and $(1,1)$. It is represented by the matrix

$$H_\varphi = \begin{pmatrix} 0.990261 & -0.001957 & 1 \\ -0.0039025 & 0.9922065 & 1 \\ -1.14219.10^{-5} & -1.14219.10^{-5} & 1 \end{pmatrix}. \tag{6.16}$$

The transformed images $\underline{u}_\varphi$ and $\tilde{\underline{u}}(\varphi)$ were computed using an interpolation method with half-symmetric extension. The methods considered were *bic*, *spline11*, *tpi*, *bic-z2*, *spline11-z2*, *p+s-bic*, *p+s-spline11* and *p+s-tpi-spline11*.

The difference images $DI(\varphi)$ are shown in Figure 6.3. For the base methods *bic* and *spline11*, we recognize the structure of the image. It is also the case for the *bic-z2* and *p+s-bic* methods i.e. the fine-tuned methods using bicubic interpolation on the zoomed image. For the *tpi*, *spline11-z2*, *p+s-spline11* and *p+s-tpi-spline11* methods the difference is mainly composed of typical high frequency structures.

The spectra of the difference images are shown in Figure 6.4. For the *bic*, *bic-z2* and *p+s-bic* methods i.e. the methods where the possibly zoomed images are interpolated by bicubic interpolation, the spectrum of the difference is high everywhere except in a small low-frequency region. For the *spline11* method the corresponding region is larger. This explains why the structure of the image is visible in the difference image for these methods in Figure 6.3. For the *tpi* and *spline11-z2* methods, the spectrum has non-negligible values in a band around the Nyquist frequency and in a vertical and horizontal cross structure. The cross structure is also visible in the *bic-z2* method and is due to the incorrect handling of the non-periodicity. Indeed it disappears in the *p+s-spline11* and *p+s-tpi-spline11* methods.

The difference images $DI(\varphi)^c$ with spectrum clipping of ratio $r = 5\%$ are shown in Figure 6.5. Only the results for the *tpi*, *spline11-z2*, *p+s-spline11* and *p+s-tpi-spline11* methods are presented because in the other cases the clipping has no significative impact on the difference images. For the considered methods, the structure of the input image is visible while it was not before clipping (see Figure 6.3). The error is clearly lower when the periodic plus smooth decomposition is used.

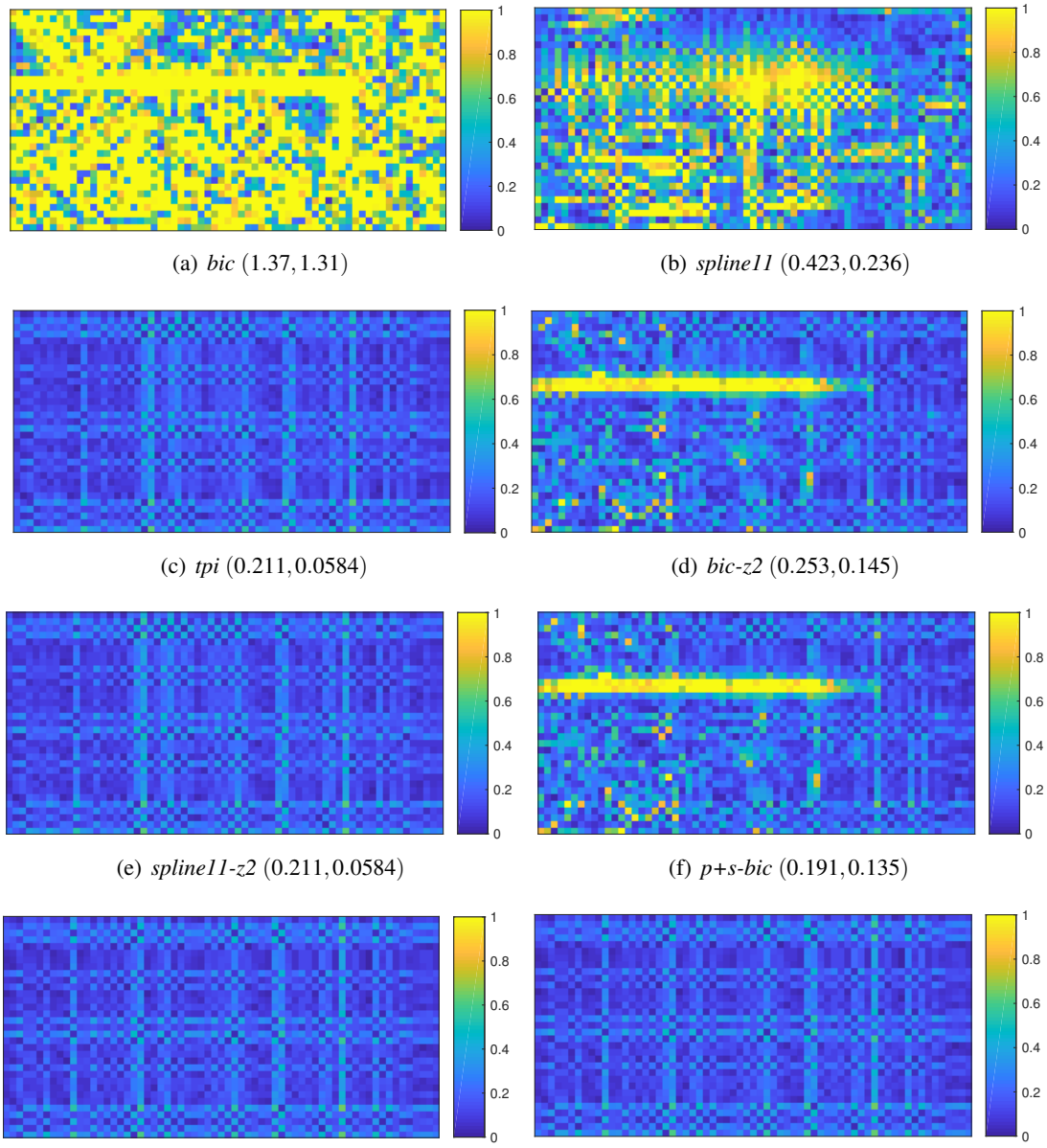
6.4.3 Propagation of the Error

In this section, we evaluate the propagation of the error after successive direct and inverse transformations. Let $(\varphi_i)_{1\leq i\leq N_{\text{transf}}}$ be random geometric transformations. Define the images $\underline{u}_i$ for $0 \leq i \leq N_{\text{transf}}$ by

$$\begin{cases} \underline{u}_0 = \underline{u} \\ \underline{u}_{i+1} = \left((\underline{u}_i)_{\varphi_{i+1}}\right)_{\varphi_{i+1}^{-1}} \quad 0 \leq i \leq N_{\text{transf}} - 1. \end{cases} \tag{6.17}$$

The RMSE, taken at a distance $\delta = 20$ of the boundary, between the $\underline{u_i}$ and $\underline{u}$ represents the error after i successive direct and inverse transformations. The fundamental difference with the consistency measurement is that no crop is done between the transformations. The clipped variant of this error is obtained by clipping the spectrum of the difference in a band of $b = 10$ pixels.

The interpolation methods considered are the same as in Section 6.4.2 (with half-symmetric extension). The evolutions of the errors using up to $N_{\text{transf}} = 50$ random homographies are shown in Figure 6.6. Globally the errors tend to increase with the number of transformations. To prevent an important increase of the error *p+s-spline11-**** or *p+s-tpi-**** methods must be used. The curves of *p+s-tpi-spline11* and *p+s-spline11* are indistinguishable since the results are almost the same. Otherwise a large part of the spectrum is not well reconstructed: either because the interpolation method is not able to reconstruct the high frequencies or either because the non-periodicity is not well handled (this is apparent by the presence of a cross structure in the difference spectrum). The clipped versions of the error for the *tpi*, *spline11-z2*, *p+s-spline11* and *p+s-tpi-spline11* methods are significantly smaller than the original error. We deduce that the error is mainly localized in a small high-frequency band.



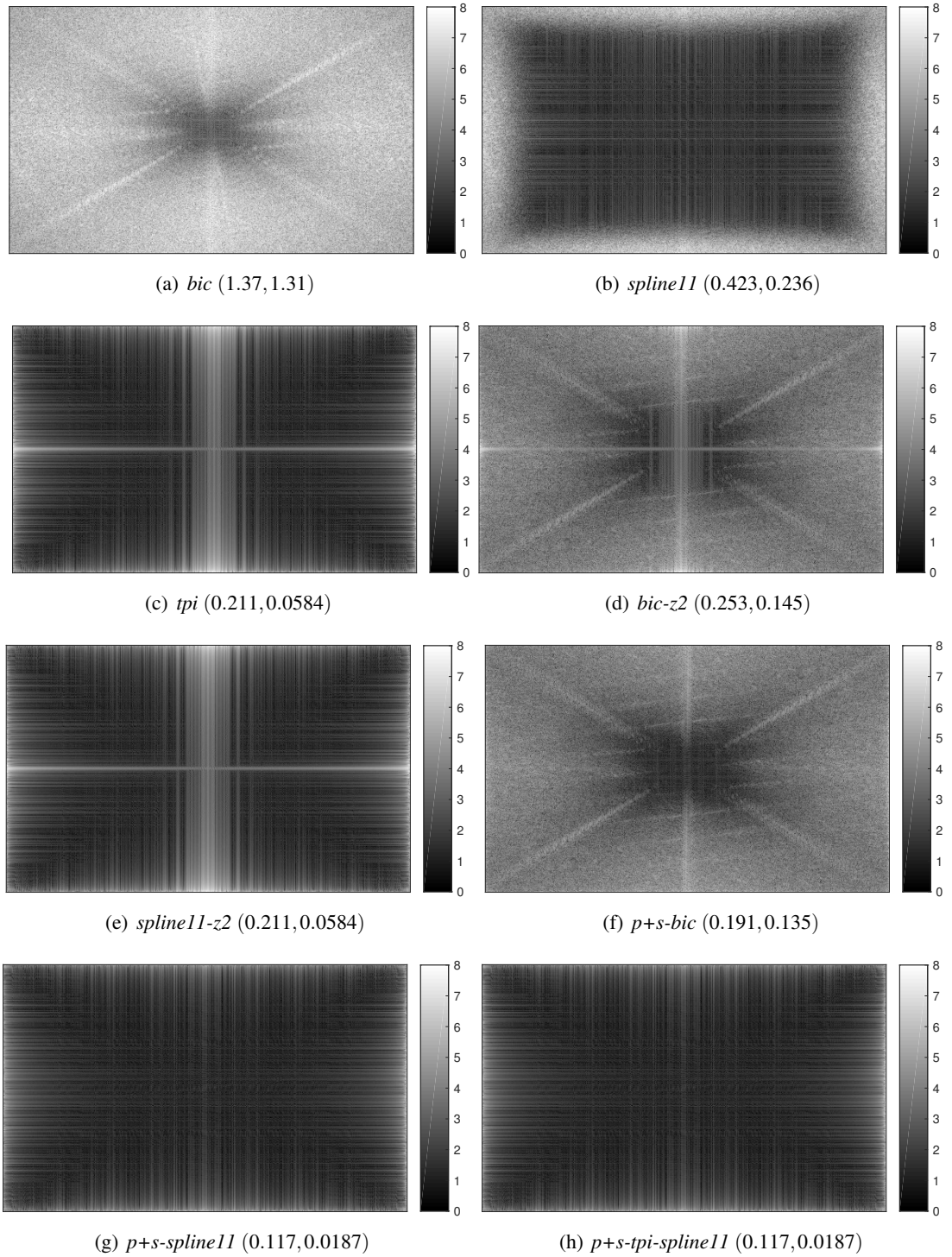
(a) *bic* (1.37, 1.31) (b) *spline11* (0.423, 0.236)

(c) *tpi* (0.211, 0.0584) (d) *bic-z2* (0.253, 0.145)

(e) *spline11-z2* (0.211, 0.0584) (f) *p+s-bic* (0.191, 0.135)

(g) *p+s-spline11* (0.117, 0.0187) (h) *p+s-tpi-spline11* (0.117, 0.0187)

Figure 6.3: Difference image $DI(\varphi)$ for various interpolation methods. The geometric transformation φ used is the homography defined by the matrix H_φ in (6.16). The values between parentheses correspond to $(\mathcal{CM}(\varphi), \mathcal{CM}^c(\varphi))$. For *bic* and *spline11*, we recognize the structure of the image. It is also the case for the *bic-z2* and *p+s-bic* methods i.e. the fine-tuned methods using bicubic interpolation on the zoomed image. For the *tpi*, *spline11-z2*, *p+s-spline11* and *p+s-tpi-spline11* methods the difference is mainly composed of high frequency structures.



(a) *bic* (1.37, 1.31) (b) *spline11* (0.423, 0.236)

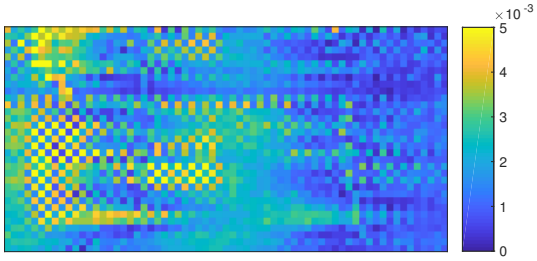
(c) *tpi* (0.211, 0.0584) (d) *bic-z2* (0.253, 0.145)

(e) *spline11-z2* (0.211, 0.0584) (f) *p+s-bic* (0.191, 0.135)

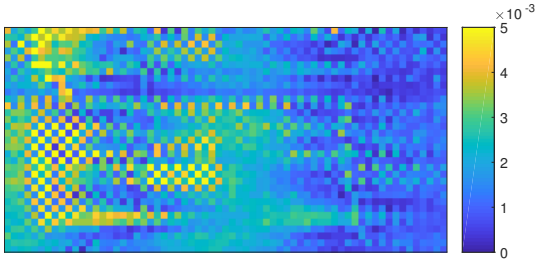
(g) *p+s-spline11* (0.117, 0.0187) (h) *p+s-tpi-spline11* (0.117, 0.0187)

Figure 6.4: Spectrum of the difference image $DI(\varphi)$ for various interpolation methods. The spectrums correspond to the unnormalized discrete Fourier transform in logarithmic scale $u \mapsto \log(1+u)$. The geometric transformation used is the homography φ defined by the matrix H_φ in (6.16). The values between parentheses correspond to $(\mathcal{CM}(\varphi), \mathcal{CM}^c(\varphi))$. For the *bic*, *bic-z2* and *p+s-bic* methods i.e. the methods where the possibly zoomed images are interpolated by bicubic interpolation, the spectrum of the difference is high everywhere except in a small low-frequency region. For the *spline11* method the corresponding region is larger. This explains why the structure of the image is visible in the difference image for these methods in Figure 6.3. For the *tpi* and *spline11-z2* methods, the spectrum has non-negligible values in a band around the Nyquist frequency and in a vertical and horizontal cross structure. The cross structure is also visible in the *bic-z2* method and is due to the incorrect handling of the non-periodicity. Indeed it disappears in the *p+s-spline11* and *p+s-tpi-spline11* methods.

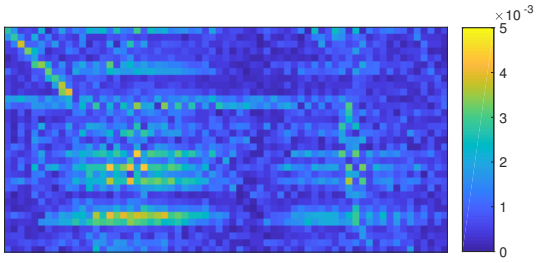
(a) Input



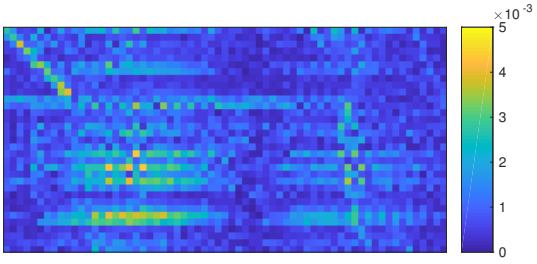
(b) *tpi* ($\mathcal{CM}^c(\varphi) = 0.0584$)



(c) *spline11-z2* ($\mathcal{CM}^c(\varphi) = 0.0584$)



(d) *p+s-spline11* ($\mathcal{CM}^c(\varphi) = 0.0187$)



(e) *p+s-tpi-spline11* ($\mathcal{CM}^c(\varphi) = 0.0187$)

Figure 6.5: Difference image $DI(\varphi)^c$ with spectrum clipping of ratio $r = 5\%$ for various interpolation methods. The geometric transformation φ used is the homography defined by the matrix H_φ in (6.16). The structure of the input image is visible while it was not before clipping (see Figure 6.3). The error is clearly lower when the periodic plus smooth decomposition is used.

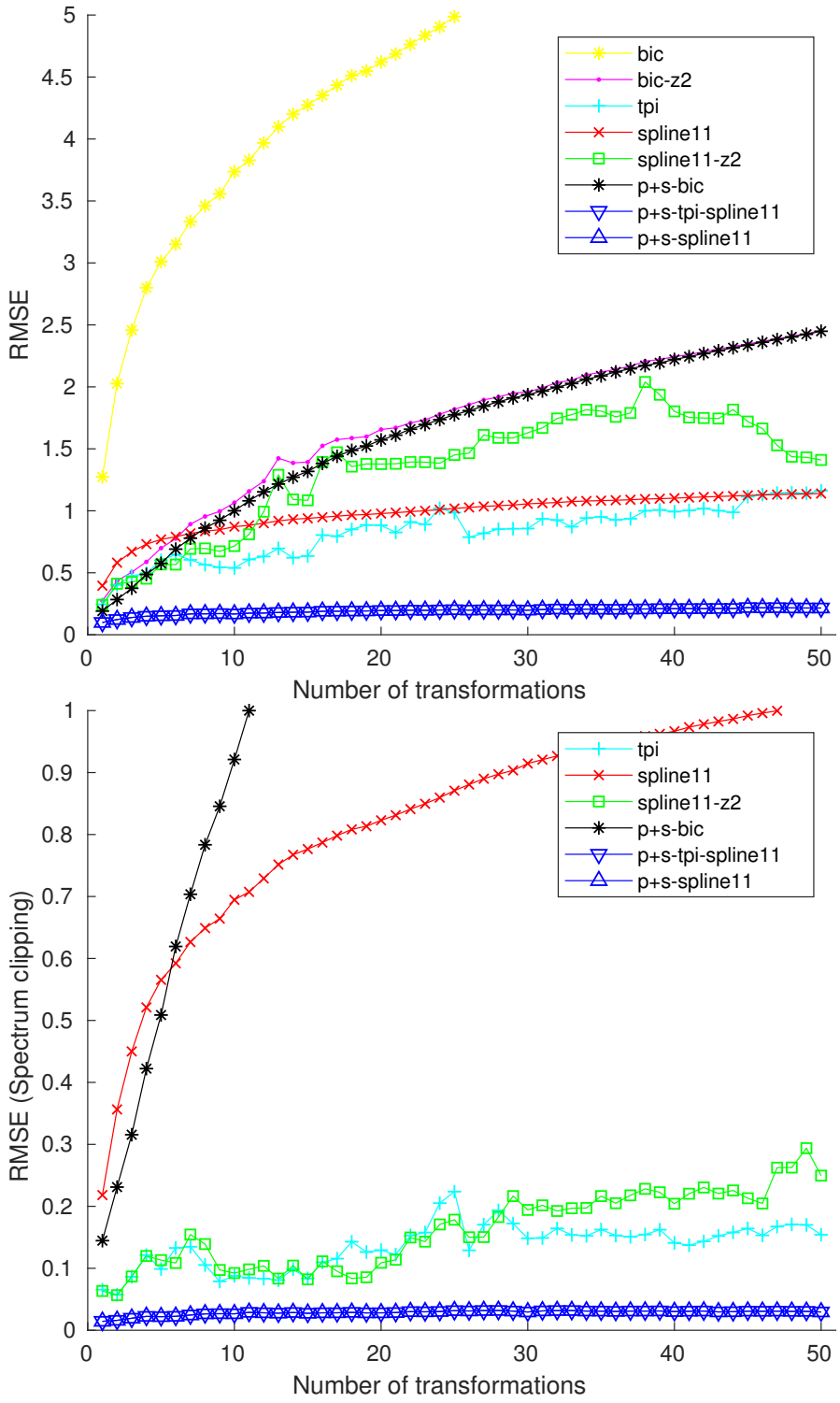


Figure 6.6: Evolution of the error after successive direct and inverse transformations for various interpolation methods (with half-symmetric extension). Up to $N_{\text{transf}} = 50$ random homographies are used. For the error with spectrum clipping (bottom), the *bic* and *bic-z2* methods are not shown since not in the range. The curves of *p+s-tpi-spline11* and *p+s-spline11* are indistinguishable since the results are almost the same. Globally the errors tend to increase with the number of transformations. The clipped versions of the error for the *tpi*, *spline11-z2*, *p+s-spline11* and *p+s-tpi-spline11* methods are significantly smaller than the original error. We deduce that the error is mainly localized in a small high-frequency band. To prevent an important increase of the error *p+s-spline11-**** or *p+s-tpi-**** methods must be used.

6.5 Conclusion

In this chapter, we introduced the consistency measurement of interpolation methods. For a given image, it measures the error after applying successively a transformation, a crop (removing boundary artefacts) and the inverse transformation. An average over random transformations is made to remove the dependency on the transformation. The clipped consistency measurement is obtained by clipping the spectrum of the difference.

We also proposed new fine-tuned interpolation methods that are based on pre-existing interpolation methods. The zoomed version of an interpolation method is obtained by applying it to the DFT zoom-in of the image. It is closer to the trigonometric polynomial interpolation and has a better performance but it suffers from artefacts due to the non-periodicity of the image. In the periodic plus smooth version of interpolation methods, this non-periodicity is handled by applying the zoomed version to the periodic component and a base interpolation method, possibly of low order, to the smooth component.

In an experimental part, we have shown that the proposed fine-tuned methods have better consistency measurements than pre-existing interpolation methods and that the error is mainly localized in a small high-frequency band. The periodic plus smooth versions should be used to interpolate images. We recommend to use the periodic plus smooth versions of high order B-spline (e.g. *p+s-spline11-****). It is more efficient and provides better results than trigonometric polynomial interpolation. In Chapter 10, the *p+s-spline11* method will be used for generating synthetic data and for the burst denoising method.

Part II

Registration

Chapter 7

Modified Inverse Compositional Algorithm

Abstract

This chapter is taken from [16]. First introduced in [8, 122], the inverse compositional algorithm for parametric motion estimation is an improvement of the classical intensity-based method of Lucas-Kanade [74]. At each step of its iterative scheme it solves a minimization problem equivalent to Lucas-Kanade but more efficiently. In this chapter, we introduce several improvements of the inverse compositional algorithm. We propose an improved handling of boundary pixels, a different color handling and gradient estimation, and the possibility to skip scales in the multiscale coarse-to-fine scheme. In an experimental part, we analyze the influence of the modifications. The estimation accuracy is at least improved by a factor 1.3 while the computation time is at least reduced by a factor 2.2 for color images. The modified inverse compositional algorithm will be used in Chapter 8 in the proposed two-step method for mosaicked images.

Contents

7.1 Introduction

First introduced in [8, 122], the inverse compositional algorithm for parametric motion estimation is an improvement of the classical intensity-based method of Lucas-Kanade [74]. At each step of its iterative scheme it solves a minimization problem equivalent to Lucas-Kanade but more efficiently. It allows for precomputations and the gradient of the reference image is not interpolated. The inverse compositional algorithm was studied in more detail in [9]. With the extension to robust error functions in [11], it becomes less sensitive to outliers. Better precision for large motions can be obtained with a coarse-to-fine multiscale approach as in [104]. Using the implementation provided by [104], it can be observed that, for moderate deformations, this method is faster and more accurate than classical feature-based methods that rely on SIFT keypoints [72, 97] and the RANSAC algorithm [39].

We claim that the inverse compositional algorithm can be further improved and accelerated with a correct handling of the boundary values, a different color handling and gradient estimation, and by skipping scales in the multiscale coarse-to-fine scheme. The estimation accuracy is at least improved by a factor 1.3 while the computation time is at least reduced by a factor 2.2 for color images. In this chapter, we detail the inverse compositional algorithm and analyze the influence of the proposed modifications. An implementation of the modified algorithm based on the one of [104] is also provided[1].

The chapter is organized as follows. First, we detail the inverse compositional algorithm in Section 7.2 as it is presented in [104], i.e., with the use of robust error functions and a multiscale approach. In Section 7.3, we present the proposed modifications that lead to the modified inverse compositional algorithm. In the experimental part (Section 7.4) we study and discuss the influence of the modifications, and we finish with conclusions in Section 7.5.

7.2 The Inverse Compositional Algorithm for Parametric Registration

Let M, N, C be three positive integers. Define the spatial domain $\Omega = \Omega_{M,N} = \{0, \dots, M-1\} \times \{0, \dots, N-1\}$. Let I_1 and I_2 be two images of size $M \times N$ with C channels (e.g. $C = 1$ for grayscale images and $C = 3$ for color images). The channels are handled so that, for $\mathbf{x} \in \Omega$, $I_1(\mathbf{x}) = (I_1^{(1)}(\mathbf{x}), \dots, I_1^{(C)}(\mathbf{x}))^T \in \mathbb{R}^C$ is a vector of length C.

The motion between the two images is assumed to be representable by a parametric motion model. Denote by $\Psi(\cdot;\mathbf{p}) : \mathbb{R}^2 \to \mathbb{R}^2$ the transformation parametrized by $\mathbf{p} \in \mathbb{R}^n$. The parametric motion estimation problem is to find a motion parameter $\mathbf{p}^\star$ such that

$$\forall \mathbf{x} \in \Omega, \quad I_1(\mathbf{x}) \simeq I_2\left(\Psi(\mathbf{x};\mathbf{p}^\star)\right). \tag{7.1}$$

There is in practice no equality in (7.1) because, for instance, images contain noise and occlusions may occur. In addition, the motion model in general only approximates the real motion.

Additional hypotheses. In order to apply the inverse compositional algorithm, additional hypotheses are made on the set of transformations $\{\Psi(\cdot;\mathbf{p}), \mathbf{p} \in \mathbb{R}^n\}$ and its parametrization.

1. The motion parameter $\mathbf{p} = 0$ corresponds to the identity transformation, i.e. $\Psi(\mathbf{x};0) = \mathbf{x}$ for all $\mathbf{x} \in \mathbb{R}^2$.
2. The set of transformations has a group structure under composition of functions.
3. For all $\mathbf{x} \in \mathbb{R}^2$, the function $\mathbf{p} \in \mathbb{R}^n \mapsto \Psi(\mathbf{x};\mathbf{p})$ is differentiable at $\mathbf{p} = 0$.

[1] http://dev.ipol.im/~briand/mInverseCompositional_1.00.zip

Table 7.1 lists the typical planar transformations and provides examples of parametrization. An example of images related by an homographic transformation is proposed in Figure 7.1.

Transform	n	Parameters - $\mathbf{p}$	Matrix - $H(\mathbf{p})$	Jacobian - $J(x,y)$
Translation	2	(t_x,t_y)	$\begin{pmatrix}1&0&t_x\\0&1&t_y\\0&0&1\end{pmatrix}$	$\begin{pmatrix}1&0\\0&1\end{pmatrix}$
Euclidean	3	(t_x,t_y,θ)	$\begin{pmatrix}\cos\theta&-\sin\theta&t_x\\\sin\theta&\cos\theta&t_y\\0&0&1\end{pmatrix}$	$\begin{pmatrix}1&0&-y\\0&1&x\end{pmatrix}$
Similarity	4	(t_x,t_y,a,b)	$\begin{pmatrix}1+a&-b&t_x\\b&1+a&t_y\\0&0&1\end{pmatrix}$	$\begin{pmatrix}1&0&x&-y\\0&1&y&x\end{pmatrix}$
Affinity	6	$(t_x,t_y,a_{11},a_{12},a_{21},a_{22})$	$\begin{pmatrix}1+a_{11}&a_{12}&t_x\\a_{21}&1+a_{22}&t_y\\0&0&1\end{pmatrix}$	$\begin{pmatrix}1&0&x&y&0&0\\0&1&0&0&x&y\end{pmatrix}$
Homography	8	$(h_{11},h_{12},h_{13},\ldots,h_{32})$	$\begin{pmatrix}1+h_{11}&h_{12}&h_{13}\\h_{21}&1+h_{22}&h_{23}\\h_{31}&h_{32}&1\end{pmatrix}$	$\begin{pmatrix}x&y&1&0&0&0&-x^2&-xy\\0&0&0&x&y&1&-xy&-y^2\end{pmatrix}$

Table 7.1: Planar transformations in their homogeneous coordinates and their Jacobians.

Figure 7.1: Example of reference image I_1 (left) and warped image I_2 (right). These are color images of size 512×512 (i.e. $M = N = 512$ and $L = 3$) linked by a homography. The reference image is generated from the warped image by bicubic interpolation where outside pixel values are set to 0 (black region).

The inverse compositional algorithm tries to solve the parametric motion problem with an iterative scheme. Firstly, we start by introducing the mathematical construction on which the algorithm relies on. Secondly, we detail the inverse compositional algorithm along with all its parameters. Then, we discuss the influence of the error function. Finally, we present the coarse-to-fine multiscale approach.

7.2.1 Mathematical Construction

A good candidate for the parameter $\mathbf{p}^*$ in (7.1) is a minimizer of the energy

$$\mathbf{p} \in \mathbb{R}^n \mapsto E_0(\mathbf{p}) = \sum_{\mathbf{x}\in\Omega} \rho\left(\|I_2(\Psi(\mathbf{x};\mathbf{p})) - I_1(\mathbf{x})\|^2\right) \tag{7.2}$$

where $\rho : \mathbb{R}^+ \to \mathbb{R}^+$ is an increasing and derivable function called the *error function* whose influence is discussed in Section 7.2.3. As there is in general no explicit expression for computing a minimizer, it is approximated using an iterative scheme.

At a given step $j \geq 1$, the idea of the inverse compositional algorithm is to refine the current estimated transformation $\Psi(\cdot;\mathbf{p}_{j-1})$ with an inverted incremental transformation (hence the name of the algorithm) $\Psi(\cdot;\Delta\mathbf{p}_j)^{-1}$, i.e.,

$$\Psi(\cdot;\mathbf{p}_j) = \Psi(\cdot;\mathbf{p}_{j-1}) \circ \Psi(\cdot;\Delta\mathbf{p}_j)^{-1}. \tag{7.3}$$

The ideal choice for the increment $\Delta\mathbf{p}_j$ would be a minimizer of the incremental energy

$$\Delta\mathbf{p} \in \mathbb{R}^n \mapsto E_1(\Delta\mathbf{p};\mathbf{p}_{j-1}) = \sum_{\mathbf{x}\in\Omega} \rho\left(\|I_2(\Psi(\mathbf{x};\mathbf{p}_{j-1})) - I_1(\Psi(\mathbf{x};\Delta\mathbf{p}))\|^2\right) \tag{7.4}$$

as it would give $\mathbf{p}_j = \mathbf{p}^\star$ with (7.3) under the assumption that (7.1) holds. But, as for the original energy E_0, such a minimizer cannot be computed and can only be approximated. Therefore the energy E_1 is approximated as follows using two successive first order Taylor expansions. Let $\mathbf{x} \in \Omega$.

First approximation. A first order Taylor expansion of the function $\Delta\mathbf{p} \in \mathbb{R}^n \mapsto I_1(\Psi(\mathbf{x};\Delta\mathbf{p}))$ around 0 gives

$$I_1(\Psi(\mathbf{x};\Delta\mathbf{p})) \simeq I_1(\mathbf{x}) + \nabla I_1^T(\mathbf{x}) J(\mathbf{x}) \Delta\mathbf{p} \tag{7.5}$$

where

$$J(\mathbf{x}) = \frac{\partial\Psi}{\partial\mathbf{p}}(\mathbf{x};0) \in \mathcal{M}_{2,n} \tag{7.6}$$

is the Jacobian matrix of the model at $\mathbf{x}$ and $\nabla I_1(\mathbf{x}) = \left(\frac{\partial I_1}{\partial x}(\mathbf{x}), \frac{\partial I_1}{\partial y}(\mathbf{x})\right)^T \in \mathcal{M}_{2,C}$ is the gradient of I_1 at $\mathbf{x}$. The Jacobian matrices J of some typical planar transformations can be found in Table 7.1. For $C > 1$, ∇I_1 actually corresponds to the transposed of the Jacobian matrix of I_1 but to simplify we keep the gradient notation. Let us set

$$G(\mathbf{x}) = \nabla I_1^T(\mathbf{x}) J(\mathbf{x}) \in \mathcal{M}_{L,n}, \tag{7.7}$$

and denote DI the difference image defined by

$$DI(\mathbf{x}) = DI(\mathbf{x};\mathbf{p}_{j-1}) = I_2(\Psi(\mathbf{x};\mathbf{p}_{j-1})) - I_1(\mathbf{x}) \in \mathbb{R}^C. \tag{7.8}$$

Using (7.5) in (7.4), we define the approximated incremental energy

$$\Delta\mathbf{p} \in \mathbb{R}^n \mapsto E_2(\Delta\mathbf{p};\mathbf{p}_{j-1}) = \sum_{\mathbf{x}\in\Omega} \rho\left(\|DI(\mathbf{x}) - G(\mathbf{x})\Delta\mathbf{p})\|^2\right). \tag{7.9}$$

Second approximation. A first order Taylor expansion of the function $t \in \mathbb{R} \mapsto \rho(\|DI(\mathbf{x})\|^2 + t)$ around 0 gives

$$\rho(\|DI(\mathbf{x})\|^2 + t) \simeq \rho(\|DI(\mathbf{x})\|^2) + \rho'(\|DI(\mathbf{x})\|^2)t. \tag{7.10}$$

Using the expansion

$$\|DI(\mathbf{x}) - G(\mathbf{x})\Delta\mathbf{p})\|^2 = \|DI(\mathbf{x})\|^2 - 2\Delta\mathbf{p}^T G(\mathbf{x})^T DI(\mathbf{x}) + \Delta\mathbf{p}^T G(\mathbf{x})^T G(\mathbf{x})\Delta\mathbf{p} \tag{7.11}$$

and $t = -2\Delta\mathbf{p}^T G(\mathbf{x})^T DI(\mathbf{x}) + \Delta\mathbf{p}^T G(\mathbf{x})^T G(\mathbf{x})\Delta\mathbf{p}$ in (7.10), we obtain the approximation

$$\rho(\|DI(\mathbf{x}) - G(\mathbf{x})\Delta\mathbf{p})\|^2) \simeq \rho(\|DI(\mathbf{x})\|^2) + \rho'(\|DI(\mathbf{x})\|^2)\left(-2\Delta\mathbf{p}^T G(\mathbf{x})^T DI(\mathbf{x}) + \Delta\mathbf{p}^T G(\mathbf{x})^T G(\mathbf{x})\Delta\mathbf{p}\right). \tag{7.12}$$

To simplify we denote $\tilde{\rho}'(\mathbf{x}) = \tilde{\rho}'(\mathbf{x};\mathbf{p}_{j-1}) = \rho'(\|DI(\mathbf{x})\|^2) \in \mathbb{R}^+$. We define the Hessian matrix $H \in \mathcal{M}_n$ and the vector $\mathbf{b} \in \mathbb{R}^n$ by

$$H = H(\mathbf{p}_{j-1}) = \sum_{\mathbf{x}\in\Omega} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T G(\mathbf{x}), \tag{7.13}$$

$$\mathbf{b} = \mathbf{b}(\mathbf{p}_{j-1}) = \sum_{\mathbf{x}\in\Omega} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T DI(\mathbf{x}). \tag{7.14}$$

Using (7.12), we define a second approximated incremental energy

$$\Delta\mathbf{p} \in \mathbb{R}^n \mapsto E_3(\Delta\mathbf{p};\mathbf{p}_{j-1}) = \sum_{\mathbf{x}\in\Omega} \rho(\|DI(\mathbf{x})\|^2) - 2\Delta\mathbf{p}^T\mathbf{b} + \Delta\mathbf{p}^T H \Delta\mathbf{p}. \tag{7.15}$$

Assuming that this quadratic form is non-degenerate (i.e. that the symmetric matrix H is positive definite), we define the increment $\Delta\mathbf{p}_j$ as its unique minimizer that is given by

$$\Delta\mathbf{p}_j = H^{-1}\mathbf{b}. \tag{7.16}$$

7.2.2 Algorithm

In practice, the gradient of I_1 is estimated using the central differences scheme, i.e. for $\mathbf{x} = (x,y) \in \Omega$,

$$\frac{\partial I_1}{\partial x}(\mathbf{x}) \simeq \frac{1}{2}\left(I_1(x+1,y) - I_1(x-1,y)\right), \tag{7.17}$$

$$\frac{\partial I_1}{\partial y}(\mathbf{x}) \simeq \frac{1}{2}\left(I_1(x,y+1) - I_1(x,y-1)\right). \tag{7.18}$$

The difference image DI defined in (7.8) requires the values $I_2(\Psi(\mathbf{x};\mathbf{p}_{j-1}))$, which are computed by bicubic interpolation [46]. Neumann boundary conditions are adopted for both, the gradient estimation, and the interpolation. As in [104], when $I_2(\Psi(\mathbf{x};\mathbf{p}_{j-1}))$ has to be evaluated outside of the domain $[0,M-1]\times[0,N-1]$ its values are arbitrarily set to 0. This may be useful for simulated images as the one shown in Figure 7.1 but it is not adapted to real data. In Section 7.3.2 we discuss the impact of this choice and propose a strategy to avoid introducing bias.

Given an initialization $\mathbf{p}_0 \in \mathbb{R}^n$, the $\mathbf{p}_j$'s can be computed using (7.3) and (7.16), and are expected to be close to a minimizer of the original energy E_0 after enough iterations. The iterations are stopped as soon as the increment $\Delta\mathbf{p}_j$ is small enough. More precisely, for a given threshold $\varepsilon > 0$, the stopping criterion is

$$\|\Delta\mathbf{p}_j\| \leq \varepsilon. \tag{7.19}$$

Because the sequence $(\mathbf{p}_j)_{j\in\mathbb{N}}$ is not guaranteed to converge, a maximum number of iterations $j_{\max}$ is also set. When the error function ρ depends on a threshold parameter we adjust it during the iterations as explained in Section 7.2.3. The one-scale inverse compositional algorithm is shown in Algorithm 7.1.

This algorithm is an improvement of the Lucas-Kanade method [74] since at each step of the incremental refinement it solves an equivalent minimization problem but more efficiently [8]. As ∇I_1, G, and $G^T G$ do not depend on $\mathbf{p}_{j-1}$ they are precomputed before the incremental refinement. The Hessian H can also be precomputed for the L2 error function (see Section 7.2.3). Note that precomputing is memory greedy and may be replaced by in-place computations. For instance the precomputation of $G^T G$ requires to store n^2MN values. In addition, the gradient of the reference image is not interpolated during the incremental refinement.

Note that contrast change may deteriorate the algorithm performance since it is not taken into account in (7.1). It can be handled by equalizing the input image contrasts, for instance using the Midway Image Equalization algorithm [28, 50] or a simpler mean equalization method.

Algorithm 7.1: One-scale inverse compositional algorithm

input : I_1, I_2, $\mathbf{p}$, ε, $j_{\max}$, ρ
output: Motion parameter $\mathbf{p} \in \mathbb{R}^n$
// **Precomputations**
Estimate the gradient ∇I_1 as in (7.17) and (7.18) using central differences with constant boundary condition.
Compute the Jacobian matrix J as in (7.6) (see Table 7.1 for typical planar transformations).
Compute $G = \nabla I_1^T J$.
Compute $G^T G$.
// **Incremental refinement**
Initialize $\mathbf{p}_0 = \mathbf{p}$ and $j = 1$.
repeat
  for $\mathbf{x} \in \Omega$ **do**
    if $\Psi(\mathbf{x}; \mathbf{p}_{j-1}) \in [0, M-1] \times [0, N-1]$ **then**
      Compute $I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1}))$ by bicubic interpolation with constant boundary condition.
    else
      Set $I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1})) = 0$
    Compute $DI(\mathbf{x}) = I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1})) - I_1(\mathbf{x})$.
  Update the threshold parameter of the error function ρ as explained in Section 7.2.3.
  Compute $\tilde{\rho}' = \rho'(\|DI\|^2)$.
  Compute the vector $\mathbf{b} = \sum_{\mathbf{x}\in\Omega} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T DI(\mathbf{x})$.
  Compute the Hessian $H = \sum_{\mathbf{x}\in\Omega} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T G(\mathbf{x})$ and invert it.
  Compute $\Delta\mathbf{p}_j = H^{-1}\mathbf{b}$.
  $\Psi(\cdot; \mathbf{p}_j) \leftarrow \Psi(\cdot; \mathbf{p}_{j-1}) \circ \Psi(\cdot; \Delta\mathbf{p}_j)^{-1}$.
  $j \leftarrow j + 1$.
until $j > j_{max}$ ***or*** $\|\Delta\mathbf{p}_{j-1}\| \leq \varepsilon$;
Return $\mathbf{p} = \mathbf{p}_j$.

7.2.3 Error Function

The influence of the error function ρ on the model estimation is only determined by its variations, i.e., its derivative ρ' through the weighting by $\rho'(\|DI(\mathbf{x})\|^2)$ in (7.13) and (7.14). In theory, any increasing and derivable function can be chosen as the error function but in the following we only consider the L2 error function, for which the computations are simplified, and the robust error functions, for which the model estimation is robust to outliers. Note that the method can be extended to error functions that are derivable almost everywhere by arbitrarily choosing a representative of the derivative. In particular, it allows to use the truncated L2 error function.

L2 error function. The error function originally considered in [8] was the identity function $\rho(s) = s$, which results in an L2 error in (7.2). It has the advantage of having a constant derivative $\rho' = 1$ so that the Hessian H, defined in (7.13), and its inverse can be precomputed before the incremental refinement. Another theoretical advantage is that in Section 7.2.1 only one first order Taylor expansion is necessary and $E_2 = E_3$. However, as it gives the same weight to every pixel, the model estimation is not robust to outliers.

Robust error function. We call robust error function [11, 104] an error function that reduces the influence of high errors in the model estimation. Typically, it is the case when ρ' is bounded

and $\rho'(s) \underset{s\to+\infty}{\longrightarrow} 0$. The model estimation using a robust error function is more robust to outliers because they have less influence. In particular, it allows to deal with problems like occlusions, noise, local brightness changes or spurious motions. Typical robust error functions considered in [104] are given in Table 7.2. Note that they all depend on a threshold parameter $\lambda > 0$, which controls the variation of the error function, i.e., the influence of outliers. A small value limits the influence of pixels with a high error while a large value means that all the pixels tend to have the same weighting.

Selection of the threshold parameter. In Algorithm 7.1, when the error function depends on a threshold parameter that is not specified by the user, it is initialized with a large value λ_0 and then geometrically reduced during the incremental refinement. This strategy successively reduces the influence of outliers, which are gradually eliminated. In practice, at step j of the incremental refinement we use the parameter $\lambda_j = \max(0.9^j\lambda_0, 5)$, where $\lambda_0 = 80$.

7.2.4 Coarse-to-fine Multiscale Approach

The two approximations in Section 7.2.1 are only valid under the assumption that $\Delta\mathbf{p}$ is small. Therefore, in order to estimate large displacements, a coarse-to-fine multiscale approach is used.

Gaussian pyramid of an image. Let I be an image, N_{scales} be the number of scales in the pyramid and $\eta \in (0,1)$ be the downsampling factor. For $s \in \{0,\dots,N_{\text{scales}}-1\}$ we note I^s the image of the pyramid at scale s. The Gaussian pyramid is recursively computed for $s = 1$ to $N_{\text{scales}} - 1$ from $I_0 = I$ by

$$I^s(\mathbf{x}) = \left(G_\sigma * I^{s-1}\right)\left(\frac{1}{\eta}\mathbf{x}\right). \tag{7.20}$$

In order to avoid aliasing and to reduce the noise, the images are smoothed with a Gaussian kernel of standard deviation

$$\sigma = \sigma(\eta) = \sigma_0\sqrt{\frac{1}{\eta^2} - 1}, \tag{7.21}$$

where $\sigma_0 = 0.6$ is found empirically in [79]. After the convolution (which is computed by applying a discrete kernel with finite support and using the whole-symmetric boundary condition [46]), the images are resampled using bicubic interpolation with a step $\frac{1}{\eta}$. In practice, the number of scales N_{scales} is adjusted so that the coarsest scale image is greater than 32 pixels in the shortest dimension. Thus, the maximal number of scales is

$$N_{\text{scales}}^{\max} = 1 + \left\lfloor \frac{\log\left(\frac{\min(M,N)}{32}\right)}{-\log(\eta)} \right\rfloor. \tag{7.22}$$

Coarse-to-fine approach. The estimation of the motion between I_1 and I_2 using the multiscale approach is done as follows. First, the two Gaussian pyramids $(I_1^s)_{0\le s\le N_{\text{scales}}-1}$ and $(I_2^s)_{0\le s\le N_{\text{scales}}-1}$ are computed as explained in the previous paragraph. Assuming that the images share a large common part, the motion is initialized at the coarsest scale $s = N_{\text{scales}} - 1$ as $\mathbf{p}^{N_{\text{scales}}-1} = 0$ (i.e., the identity transformation) and is estimated using the one-scale inverse compositional algorithm (Algorithm 7.1). Then the estimation is refined in the following finer scales. To transfer the motion parameter $\mathbf{p}^s$ at scale s to the motion parameter $\mathbf{p}^{s-1}$ at scale $s-1$, the transformation is updated according to the parametrization. Update rules for the parametrizations of planar transformations proposed in Table 7.1 are presented in Table 7.3. The multiscale inverse compositional algorithm is shown in Algorithm 7.2.

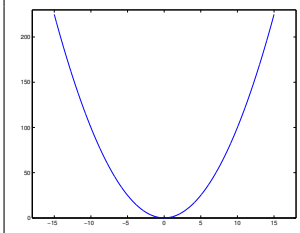
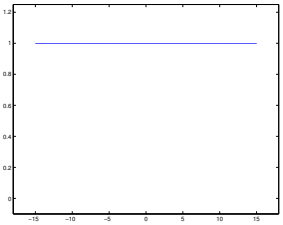
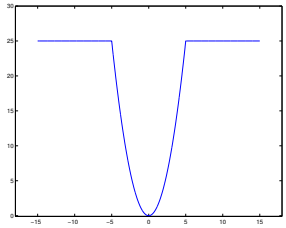
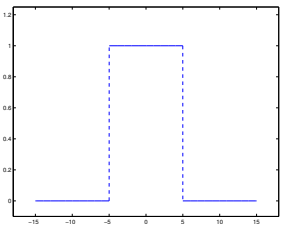
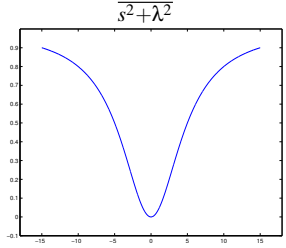
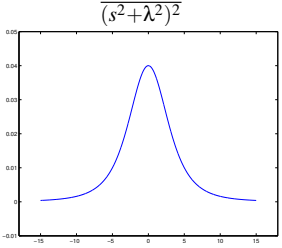
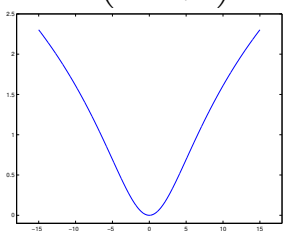
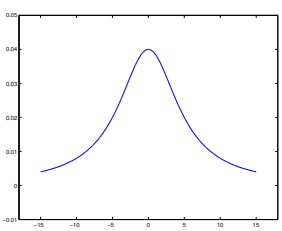
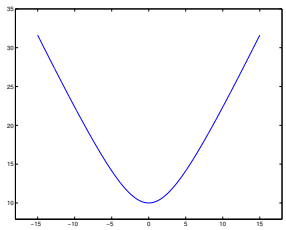
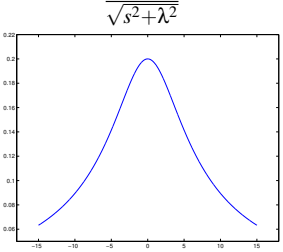
Type	$s \mapsto \rho_\lambda(s^2)$	$s \mapsto \rho'_\lambda(s^2)$
L2	s^2 	1 
Truncated L2	$\begin{cases} s^2 & \text{if } s < \lambda \\ \lambda^2 & \text{otherwise} \end{cases}$ 	$\begin{cases} 1 & \text{if } s < \lambda \\ 0 & \text{otherwise} \end{cases}$ 
Geman & McClure	$\frac{s^2}{s^2+\lambda^2}$ 	$\frac{\lambda^2}{(s^2+\lambda^2)^2}$ 
Lorentzian	$\log\left(1+\left(\frac{s}{\lambda}\right)^2\right)$ 	$\frac{1}{s^2+\lambda^2}$ 
Charbonnier	$2\sqrt{s^2+\lambda^2}$ 	$\frac{s}{\sqrt{s^2+\lambda^2}}$ 

Table 7.2: Example of error functions. Except for the L2 error function, they are all robust error functions depending on a threshold parameter λ. The curves correspond to $\lambda = 5$.

Transform	$\mathbf{p}^{s-1}$	$\mathbf{p}^{s}$
Translation	(t_x, t_y)	$\frac{1}{\eta}(t_x, t_y)$
Euclidean	(t_x, t_y, θ)	$(\frac{1}{\eta}t_x, \frac{1}{\eta}t_y, \theta)$
Similarity	(t_x, t_y, a, b)	$(\frac{1}{\eta}t_x, \frac{1}{\eta}t_y, a, b)$
Affinity	$(t_x, t_y, a_{11}, a_{12}, a_{21}, a_{22})$	$(\frac{1}{\eta}t_x, \frac{1}{\eta}t_y, a_{11}, a_{12}, a_{21}, a_{22})$
Homography	$(h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32})$	$(h_{11}, h_{12}, \frac{1}{\eta}h_{13}, h_{21}, h_{22}, \frac{1}{\eta}h_{23}, \eta h_{31}, \eta h_{32})$

Table 7.3: Update rule in the coarse-to-fine scheme for the parametrizations of planar transformations proposed in Table 7.1.

Algorithm 7.2: Multiscale inverse compositional algorithm

input : I_1, I_2, ε, $j_{\max}$, ρ, N_{scales}, η
output: Transformation $\mathbf{p} \in \mathbb{R}^n$

Create a Gaussian pyramid of images I_1^s, I_2^s for $s = 0, \dots, N_{\text{scales}} - 1$
Initialize with $\mathbf{p}^{N_{\text{scales}}-1} = 0$
for $s = N_{scales} - 1$ **to** 0 **do**
    Compute $\mathbf{p}^s$ with Algorithm 7.1 applied to I_1^s, I_2^s, $\mathbf{p}^s$, ε, $j_{\max}$, ρ.
    if $s > 0$ **then**
        Compute $\mathbf{p}^{s-1}$ from $\mathbf{p}^s$ by zoom of factor η (see Table 7.3 for typical planar transformations).
Return $\mathbf{p} = \mathbf{p}^0$

7.3 Modifications of the Inverse Compositional Algorithm

In this section we propose simple modifications to the inverse compositional algorithm that allow to improve its performance and precision. These improvements are experimentally verified in Section 7.4.

7.3.1 Grayscale Conversion

Assume that I_1 and I_2 are color images (i.e. $C = 3$). We consider a classical alternative for color handling, which consists in averaging the channels to obtain a grayscale image. This is valid since the motion is the same for all the channels. This divides by 3 the number of input pixels and by $\sqrt{3} \simeq 1.7$ the noise level. As we will see in Section 7.4.2, there is no clear advantage of using color over grayscale images. Since operating on grayscale images implies less computations, we use it.

7.3.2 Boundary Handling by Discarding Boundary Pixels

Even though it concerns a relatively small amount of pixels, the handling of boundary pixels has a significant impact on the performance of the algorithm. To estimate $\nabla I_1(\mathbf{x})$ at a pixel $\mathbf{x}$ close to the boundary of Ω, an arbitrary extension of the domain is needed (constant extension in Algorithm 7.1 and whole-symmetric extension in Section 7.3.3). Also the evaluation of $I_2(\Psi(\mathbf{x}; \mathbf{p}_{j-1}))$ by bicubic interpolation requires an arbitrary extension for position $\Psi(\mathbf{x}; \mathbf{p}_{j-1})$ that fall close to the boundary of $[0, M-1] \times [0, N-1]$. Therefore, during the motion estimation the boundary pixels are more likely to have incorrect gradient estimates, which deteriorates the performance of the algorithm. When a robust error function is used, the influence of these boundary effects is lessened but is still noticeable.

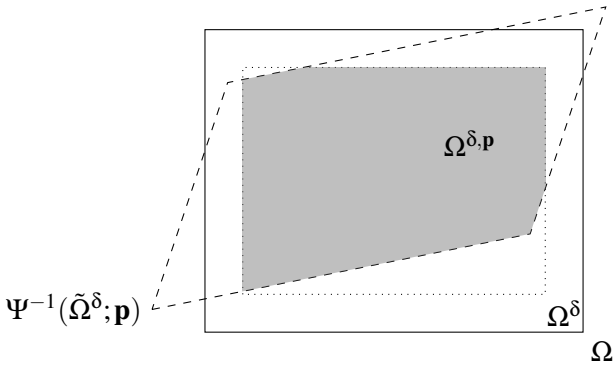


Figure 7.2: Example of domain $\Omega^{\delta,\mathbf{p}}$. Ω is the spatial domain where I_1 and I_2 are defined. Ω^{δ} represents the pixels at distance at least δ of the boundary of Ω. Pixels outside of Ω^{δ} are discarded to avoid incorrect gradient estimations. $\tilde{\Omega}^{\delta}$ is the continuous domain corresponding to the convex hull of Ω^{δ}. The continuous domain $\Psi^{-1}(\tilde{\Omega}^{\delta};\mathbf{p})$ represents the points whose images by $\Psi(\cdot;\mathbf{p})$ belongs to $\tilde{\Omega}^{\delta}$. Pixels outside of $\Omega \cap \Psi^{-1}(\tilde{\Omega}^{\delta};\mathbf{p})$ are discarded to avoid incorrect interpolations. Finally, pixels outside of the gray area are discarded and the computation are made on the pixels of $\Omega^{\delta,\mathbf{p}} = \Omega^{\delta} \cap \Psi^{-1}(\tilde{\Omega}^{\delta};\mathbf{p})$.

The handling proposed in [104], and used in Algorithm 7.1, only avoids the interpolation of values outside of the image domain. Let $\mathbf{x} \in \Omega$ such that $\Psi(\mathbf{x};\mathbf{p}_{j-1})$ falls outside of the domain $[0,M-1] \times [0,N-1]$. Then, the value $I_2(\Psi(\mathbf{x};\mathbf{p}_{j-1}))$ is set to 0. This strategy is only adapted to synthetic images where pixels outside the domain are set to 0 during the resampling (see example in Figure 7.1). Otherwise, as in general $I_1(\mathbf{x})$ has no reason to be close to 0, the model error at $\mathbf{x}$ is likely to be high.

Discarding boundary pixels. We propose an alternative strategy that handles all boundary pixels without introducing outliers. It consists in discarding boundary pixels from the sums in the energies introduced in Section 7.2.1. Note that it has the advantage of reducing the complexity of the algorithm.

More precisely, let δ be a non-negative integer and define for $\mathbf{p} \in \mathbb{R}^n$

$$\Omega^{\delta} = \Omega^{\delta}_{M,N} = \{\delta,\dots,M-1-\delta\} \times \{\delta,\dots,N-1-\delta\}, \tag{7.23}$$

$$\tilde{\Omega}^{\delta} = \tilde{\Omega}^{\delta}_{M,N} = [\delta,M-1-\delta] \times [\delta,N-1-\delta], \tag{7.24}$$

$$\Omega^{\delta,\mathbf{p}} = \{\mathbf{x} \in \Omega^{\delta} \mid \Psi(\mathbf{x};\mathbf{p}) \in \tilde{\Omega}^{\delta}\} = \Omega^{\delta} \cap \Psi^{-1}(\tilde{\Omega}^{\delta};\mathbf{p}). \tag{7.25}$$

Ω^{δ} represents the pixels at distance at least δ of the boundary of Ω. Pixels outside of Ω^{δ} are discarded to avoid incorrect gradient estimations. $\tilde{\Omega}^{\delta}$ is the continuous domain corresponding to the convex hull of Ω^{δ}. The continuous domain $\Psi^{-1}(\tilde{\Omega}^{\delta};\mathbf{p})$ represents the points whose images by $\Psi(\cdot;\mathbf{p})$ belongs to $\tilde{\Omega}^{\delta}$. Pixels outside of $\Omega \cap \Psi^{-1}(\tilde{\Omega}^{\delta};\mathbf{p}_j)$ are discarded to avoid incorrect interpolations. Finally, the computation are made on the pixels of $\Omega^{\delta,\mathbf{p}} = \Omega^{\delta} \cap \Psi^{-1}(\tilde{\Omega}^{\delta};\mathbf{p})$. We display in Figure 7.2 an example of domain $\Omega^{\delta,\mathbf{p}}$.

At step j of the incremental refinement, the boundary pixels are assumed to be located in $\Omega \setminus \Omega^{\delta,\mathbf{p}_{j-1}}$. Boundary pixels are discarded by replacing Ω by $\Omega^{\delta,\mathbf{p}_{j-1}}$ in the sums of the energies of Section 7.2.1. Equivalently it comes back to applying the mask $1_{\Omega^{\delta,\mathbf{p}_{j-1}}}(\mathbf{x})$. Consequently, the increment $\Delta\mathbf{p}_j$ is given by

$$\Delta\mathbf{p}_j = H_{\delta}^{-1}\mathbf{b}_{\delta} \tag{7.26}$$

where

$$H_\delta = H_\delta(\mathbf{p}_{j-1}) = \sum_{\mathbf{x}\in\Omega^{\delta,\mathbf{p}_{j-1}}} \tilde{\rho}'(\mathbf{x})\cdot G(\mathbf{x})^T G(\mathbf{x}) = \sum_{\mathbf{x}\in\Omega} 1_{\Omega^{\delta,\mathbf{p}_{j-1}}}(\mathbf{x})\tilde{\rho}'(\mathbf{x})\cdot G(\mathbf{x})^T G(\mathbf{x}), \tag{7.27}$$

$$\mathbf{b}_\delta = \mathbf{b}_\delta(\mathbf{p}_{j-1}) = \sum_{\mathbf{x}\in\Omega^{\delta,\mathbf{p}_{j-1}}} \tilde{\rho}'(\mathbf{x})\cdot G(\mathbf{x})^T DI(\mathbf{x}) = \sum_{\mathbf{x}\in\Omega} 1_{\Omega^{\delta,\mathbf{p}_{j-1}}}(\mathbf{x})\tilde{\rho}'(\mathbf{x})\cdot G(\mathbf{x})^T DI(\mathbf{x}). \tag{7.28}$$

7.3.3 Gradient Estimation on a Prefiltered Image

In Algorithm 7.1, the gradient ∇I_1 is estimated using the central differences scheme defined by (7.17) and (7.18). In [95], it was shown that the shift estimation with inverse compositional based-methods can be improved by using other gradient estimators. In particular, the shift estimation becomes more robust under noise. We extend this study to the general context of parametric motion estimation.

Smoothing the input image I_1 reduces its noise and its aliasing, which may lead to a better gradient estimation. Therefore, during the incremental refinement, we do not estimate directly the gradient of I_1 but we replace I_1 and I_2 by prefiltered versions $\tilde{I}_1$ and $\tilde{I}_2$ and estimate the gradient $\nabla\tilde{I}_1$. In order to be compatible with the gradient, the difference image DI is replaced by a prefiltered difference image $\tilde{DI}$. Both the gradient estimation and the prefiltering are computed by applying separable kernels. More precisely, a gradient estimation method is determined by a pair of matched prefilter and derivative kernels (stored as vectors):

- the prefilter kernel $\mathbf{k}$ is symmetrical,
- the derivative kernel $\mathbf{d}$, which is anti-symmetrical.

The prefilter is defined as $\mathbf{k}^T * \mathbf{k}$ while the horizontal and vertical gradient filters are defined respectively by $\mathbf{d}^T * \mathbf{k}$ and $\mathbf{k}^T * \mathbf{d}$. The prefiltered image $\tilde{I}_1$ whose gradient is estimated is given by

$$\tilde{I}_1 = \mathbf{k}^T * \mathbf{k} * I_1. \tag{7.29}$$

The gradient $\nabla\tilde{I}_1$ is estimated by computing the partial derivative estimates

$$\frac{\partial\tilde{I}_1}{\partial x} \simeq \mathbf{d}^T * \mathbf{k} * I_1, \tag{7.30}$$

$$\frac{\partial\tilde{I}_1}{\partial y} \simeq \mathbf{k}^T * \mathbf{d} * I_1. \tag{7.31}$$

Note that computing the gradient estimation with (7.30) and (7.31) does not require $\tilde{I}_1$. However it is still computed along with $\tilde{I}_2$ in order to get the difference image during the incremental refinement. At step j, the prefiltered difference image $\tilde{DI}$ is given by[2]

$$\tilde{DI}(\mathbf{x}) = \tilde{I}_2(\Psi(\mathbf{x};\mathbf{p}_{j-1})) - \tilde{I}_1(\mathbf{x}), \quad \mathbf{x}\in\Omega. \tag{7.32}$$

The convolutions with one-dimensional kernels are computed using the whole-symmetric boundary condition [46]. Let I be an image and $\mathbf{k}_1$ and $\mathbf{k}_2$ be two vectors. The filtering $\mathbf{k}_2^T * \mathbf{k}_1 * I$ is computed by convolving each column of I with $\mathbf{k}_1$ and then convolving each row of the result with $\mathbf{k}_2$.

All the considered gradient estimation kernels are shown in Table 7.4. In practice the kernels $\mathbf{k}$ and $\mathbf{d}$ were designed simultaneously in order to verify given properties [34, 110]. Note that the central differences estimator corresponds to $\mathbf{k} = 1$ and $\mathbf{d} = \frac{1}{2}(-1,0,1)^T$ so that there is no prefiltering required. In the following, when the central difference estimator is chosen, the gradient is computed as in Algorithm 7.1, i.e., with no prefiltering and the constant boundary condition.

[2] Interpolating the prefiltered image $\tilde{I}_2$ should be an easy task since the image is smooth.

Gradient estimator		Sample number				
		-2	-1	0	1	2
Central differences	**k**			1		
	d		-0.5	0	0.5	
Hypomode	**k**			0.5	0.5	
	d			-1	1	
Farid 3x3	**k**		0.229879	0.540242	0.229879	
	d		-0.425287	0	0.425287	
Farid 5x5	**k**	0.037659	0.249153	0.426375	0.249153	0.037659
	d	-0.109604	-0.276691	0	0.276691	0.109604
Gaussian $\sigma = 0.3$	**k**		0.003865	0.999990	0.003865	
	d		-0.707110	0	0.707110	
Gaussian $\sigma = 0.6$	**k**	0.003645	0.235160	0.943070	0.235160	0.003645
	d	-0.021915	-0.706770	0	0.706770	0.021915

Table 7.4: Prefilter kernel **k** and derivative kernel **d** of the proposed gradient estimators.

7.3.4 First Scale of the Gaussian Pyramid

When dealing with low quality input images (i.e. noisy), skipping the finest scales in the Gaussian pyramid usually yield results similar to using all the scales in the multiscale algorithm 7.2. Indeed the images are smoothed during the construction of the Gaussian pyramid, this reducing noise, alias, chromatic aberration, and zipper effects [73]. The motion estimation at coarser scales is less affected by the artifacts of the input images and the improvement at the fine scales can be negligible. Another advantage of not using the finest scales is that it significantly reduces the complexity of the algorithm. Because of its recursive construction the whole Gaussian pyramid has to be computed, but the motion estimation is only performed at the coarser scales.

In order to select the first scale used in the Gaussian Pyramid, we introduce a new parameter $s_0 \in \{0, \dots, N_{\text{scales}} - 1\}$. The modified multiscale approach only refines the motion estimation for the scales $N_{\text{scales}} - 1$ to s_0. Note that if $s_0 \geq 2$, $\mathbf{p}^0$ is computed from $\mathbf{p}^{s_0-1}$ thanks to the update rule with zoom factor η^{s_0-1} (see Table 7.3).

7.3.5 Modified Inverse Compositional Algorithm

Incorporating the proposed modifications to Algorithms 7.1 and 7.2, we obtain the modified inverse compositional algorithm. The one-scale and multiscale versions are respectively presented in Algorithms 7.3 and 7.4. The additional parameters are:

1. the non-negative integer δ for discarding boundary pixels (see Section 7.3.2),
2. the pair $(\mathbf{k}, \mathbf{d})$ of kernels for the gradient estimation and the prefiltering (see Section 7.3.3),
3. the first scale $s_0 \in \{0, \dots, N_{\text{scales}} - 1\}$ used in the pyramid (see Section 7.3.4).

In addition, the user has to specify if the grayscale conversion of Section 7.3.1 is used.

Algorithm 7.3: One-scale modified inverse compositional algorithm

input : I_1, I_2, $\mathbf{p}$, ε, $j_{\max}$, ρ, δ, $(\mathbf{k},\mathbf{d})$
output: Motion parameter $\mathbf{p} \in \mathbb{R}^n$
// **Precomputations**
Estimate the gradient $\nabla \tilde{I}_1$ from I_1 and $(\mathbf{k},\mathbf{d})$ as in (7.30) and (7.31).
Compute the prefiltered images $\tilde{I}_1 = \mathbf{k}^T * \mathbf{k} * I_1$ and $\tilde{I}_2 = \mathbf{k}^T * \mathbf{k} * I_2$.
Compute the Jacobian matrix J as in (7.6) (see Table 7.1 for typical planar transformations).
Compute $G = \nabla \tilde{I}_1^T J$.
Compute $G^T G$.
// **Incremental refinement**
Initialize $\mathbf{p}_0 = \mathbf{p}$ and $j = 1$.
repeat
  Compute the domain $\Omega^{\delta,\mathbf{p}_{j-1}}$ as in (7.25).
  for $\mathbf{x} \in \Omega^{\delta,\mathbf{p}_{j-1}}$ **do**
    Compute $\tilde{I}_2(\Psi(\mathbf{x},\mathbf{p}_{j-1}))$ by bicubic interpolation with constant boundary condition.
    Compute $\tilde{DI}(\mathbf{x}) = \tilde{I}_2(\Psi(\mathbf{x},\mathbf{p}_{j-1})) - \tilde{I}_1(\mathbf{x})$.
  Update the threshold parameter of the error function ρ as explained in Section 7.2.3.
  Compute $\tilde{\rho}' = \rho'(\|\tilde{DI}\|^2)$ on $\Omega^{\delta,\mathbf{p}_{j-1}}$.
  Compute the vector $\mathbf{b}_\delta = \sum_{\mathbf{x}\in\Omega^{\delta,\mathbf{p}_{j-1}}} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T \tilde{DI}(\mathbf{x})$.
  Compute the Hessian $H_\delta = \sum_{\mathbf{x}\in\Omega^{\delta,\mathbf{p}_{j-1}}} \tilde{\rho}'(\mathbf{x}) \cdot G(\mathbf{x})^T G(\mathbf{x})$ and invert it.
  Compute $\Delta\mathbf{p}_j = H_\delta^{-1}\mathbf{b}_\delta$.
  $\Psi(\cdot;\mathbf{p}_j) \leftarrow \Psi(\cdot;\mathbf{p}_{j-1}) \circ \Psi(\cdot;\Delta\mathbf{p}_j)^{-1}$.
  $j \leftarrow j+1$.
until $j > j_{max}$ ***or*** $\|\Delta\mathbf{p}_{j-1}\| \le \varepsilon$;
Return $\mathbf{p} = \mathbf{p}_j$.

Algorithm 7.4: Multiscale modified inverse compositional algorithm

input : I_1, I_2, ε, $j_{\max}$, ρ, N_{scales}, $s_0 \in \{0,\dots,N_{\text{scales}}-1\}$, η, δ, $(\mathbf{k},\mathbf{d})$
output: Transformation $\mathbf{p} \in \mathbb{R}^n$
If the grayscale conversion is specified by the user, replace I_1 and I_2 by the average of their channels.
Create a Gaussian pyramid of images I_1^s, I_2^s for $s = 0,\dots,N_{\text{scales}}-1$
Initialize with $\mathbf{p}^{N_{\text{scales}}-1} = 0$
for $s = N_{scales}-1$ **to** s_0 **do**
  Compute $\mathbf{p}^s$ with Algorithm 7.3 applied to I_1^s, I_2^s, $\mathbf{p}^s$, ε, $j_{\max}$, ρ, δ, $(\mathbf{k},\mathbf{d})$.
  if $s > 0$ **then**
    Compute $\mathbf{p}^{s-1}$ from $\mathbf{p}^s$ by zoom of factor η (see Table 7.3 for typical planar transformations).
if $s_0 > 1$ **then**
  Compute $\mathbf{p}^0$ from $\mathbf{p}^{s_0-1}$ by zoom of factor η^{s_0-1} (see Table 7.3 for typical planar transformations).
Return $\mathbf{p} = \mathbf{p}^0$

7.4 Experiments

In this section, we evaluate experimentally the impact of the modifications proposed in Section 7.3 on the motion estimation performance and on the computation time. First, we describe the experimental setup and the error measure used to evaluate the performance on synthetic data. Then we study the influence of the modifications and show in what extent each improves the performance of the algorithm. Finally, we compare the non-modified and modified inverse compositional algorithms with a classic parametric motion estimation based on the SIFT keypoints and the RANSAC algorithm.

7.4.1 Experimental Setup

To evaluate the performance of a motion estimation method we build, from a reference image, a sequence of noisy warped images whose transformations are known. For each image, the error between the estimated transformation and the ground truth transformation is expressed in terms of end-point error. The mean of the error among the sequence gives an evaluation of the performance of the method.

End-point error. Let I_1 and I_2 be two images linked by a transformation parametrized by $\mathbf{p}^\star$. Let $\mathbf{p}$ be the estimated motion parameters provided by a given motion estimation method. The end-point error $\mathrm{EPE}(\mathbf{p},\mathbf{p}^\star)(\mathbf{x})$ at a pixel $\mathbf{x}\in\Omega$ is defined by

$$\mathrm{EPE}(\mathbf{p},\mathbf{p}^\star)(\mathbf{x}) = \|\Psi(\mathbf{x};\mathbf{p}) - \Psi(\mathbf{x};\mathbf{p}^\star)\|_2. \tag{7.33}$$

It is the distance between the images of the estimated transformation and the ground truth transformation. The end-point error is a measure of the error that is commonly used in optical flow estimation [102, 103]. An example of end-point error field, i.e the image of end-point errors on Ω, is shown in Figure 7.3. The average end-point error $\overline{\mathrm{EPE}}(\mathbf{p},\mathbf{p}^\star)$ is defined as the mean of the end-point errors $\mathrm{EPE}(\mathbf{p},\mathbf{p}^\star)(\mathbf{x})$ over the domain Ω i.e.

$$\overline{\mathrm{EPE}}(\mathbf{p},\mathbf{p}^\star) = \frac{1}{MN}\sum_{\mathbf{x}\in\Omega}\mathrm{EPE}(\mathbf{p},\mathbf{p}^\star)(\mathbf{x}). \tag{7.34}$$

Building the test sequence. As the error varies with the images, the transformations and the noise, we evaluate the performance of methods by considering the mean of the errors over a sequence of images, which is built as follows. Let I be a reference input image. We draw N_{images} homographies parametrized by $(\mathbf{p}_i^\star)_{1\le i\le N_{\text{images}}}$ by randomly shifting the four corners of the domain Ω along both directions. The shifts are drawn independent and uniform in $[-L,L]$ for a given non-negative integer L (see Algorithm 6.4). We build a sequence $(I^i)_{1\le i\le N_{\text{images}}}$ of warped images using bicubic interpolation[3] with whole-symmetric boundary condition, verifying $I^i = I(\Psi(\cdot;\mathbf{p}_i^\star))$. Finally, we add Gaussian white noise of standard deviation σ to the reference image and the warped images. The role of the reference and warped images is swapped in the estimation algorithm to avoid the accumulation of interpolation error. In other words, we use $I_1 = I^i$ and $I_2 = I$.

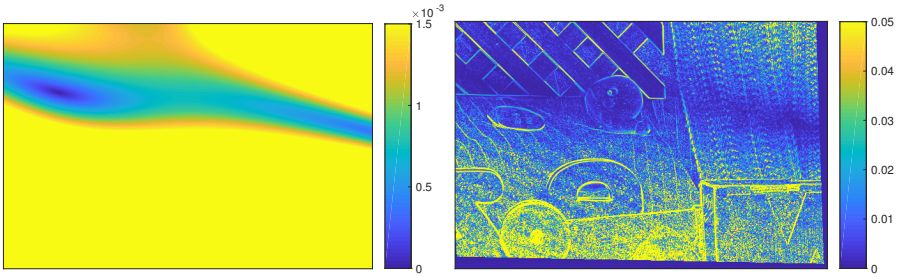
We compute the average error $\overline{\mathrm{EPE}}_i$ as in (7.34). The error of the method for the image I and the noise level σ, noted EPE to simplify, is evaluated as the mean

$$\mathrm{EPE} = \frac{1}{N_{\text{images}}}\sum_{i=1}^{N_{\text{images}}}\overline{\mathrm{EPE}}_i. \tag{7.35}$$

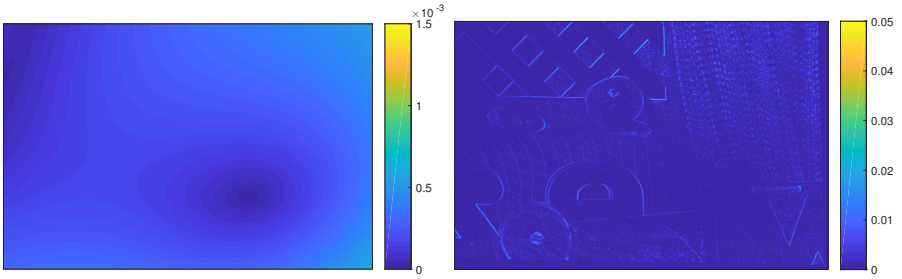
[3]The bicubic interpolation is prefered to fine-tuned methods introduced in Chapter 6 because it is used to transform images during the iterative scheme.

(a) Reference image $I_1 = I_2(\Psi(\cdot;\mathbf{p}^\star))$

(b) Warped image I_2



(c) End-point error field $\mathrm{EPE}(\mathbf{p},\mathbf{p}^\star)(\mathbf{x})$ (left) and residual (right) using the IC algorithm



(d) End-point error field $\mathrm{EPE}(\mathbf{p},\mathbf{p}^\star)(\mathbf{x})$ (left) and residual (right) using the modified IC algorithm

Figure 7.3: Example of motion estimation on synthetic data. The warped image I_2 is the image used in the experiments of Section 7.4.2. It is a color image of size 584×388 taken from the Middlebury database [10]. The reference image $I_1 = I_2(\Psi(\cdot;\mathbf{p}^\star))$ is an example of synthetic image related to I_2 by an homography and obtained by bicubic interpolation with whole-symmetric boundary condition. On the second and third line, the estimated motion $\mathbf{p}$ is obtained using either the inverse compositional algorithm or the modified inverse compositional algorithm. For the modifications we use $\delta = 5$, the Farid 5x5 kernel estimator, the grayscale conversion and $s_0 = 0$. For both methods, the right image is actually the root mean square over the channels of the residual $I_1 - I_2(\Psi(\cdot;\mathbf{p}))$, which is obtained by bicubic interpolation. Without modification, we have EPE=0.00460 and $\mathrm{RMSE}(I_1(\mathbf{x}), I_2(\Psi(\mathbf{x};\mathbf{p}))) = 0.042838$. With modification, we have EPE=0.00022 and $\mathrm{RMSE}(I_1(\mathbf{x}), I_2(\Psi(\mathbf{x};\mathbf{p}))) = 0.001790$.

Note that this error is not deterministic since it depends on the transformations and the noise realizations.

7.4.2 Influence of the Modifications

In order to evaluate the influence of the modifications proposed in Section 7.3, we use the experimental setup described in Section 7.4.1. We introduce one by one the modifications to simplify the presentation of the results.

We take as reference image I the image presented in Figure 7.3, which is a color image of size 584×388 taken from the Middlebury database [10]. For all the experiments, the sequence of transformations used is the same and is obtained by taking $N_{\text{images}} = 1000$ and $L = 20$. The noise level σ varies in $\{0,3,5,10,20,30,50\}$. In order to perform reliable comparisons, for a given noise level, the noisy images used are the same for all the methods. We use the following parameter values: $\eta = \frac{1}{2}$, $\varepsilon = 0.001$, $j_{\max} = 30$ and $N_{\text{scales}} = N_{\text{scales}}^{\max}$ (see (7.22)). The error functions used are the L2 and the Lorentzian functions, for which the threshold parameter varies during the incremental refinement as explained in Section 7.2.3. In order to study the performance of each method, we consider the end-point error (EPE) and the computation time. The displayed computation time corresponds to the CPU time used for the $N_{\text{images}} = 1000$ motion estimations and is expressed in seconds. Note that it also corresponds to the average computation time per image in milliseconds. The experiments were made using an Intel(R) Xeon(R) CPU E5-2650 @ 2.60GHz (on a single thread).

Discarding Boundary Pixels. To analyze the influence of the boundary handling, we compare the results of the inverse compositional algorithm described in Section 7.2 and of the modified version that discards boundary pixels with $\delta \in \{0,5\}$. The gradient estimation is done using the central difference scheme, all scales are used and there is no grayscale conversion. Note that $\delta = 0$ corresponds to the case where outside pixels are discarded during the interpolation (instead of setting the interpolated values to 0). Considering that, in the following, we use the gradient kernels of Table 7.4 and bicubic interpolation, the value $\delta = 5$ is large enough to discard all boundary pixels.

The results are presented in Table 7.5. It clearly shows that discarding boundary pixels always provides significantly better results in terms of precision and computation time. Discarding boundary pixels with $\delta = 5$ provides slightly better results than with $\delta = 0$ since it handles all the boundary pixels and not only outside pixels during the interpolation (which introduce more error than inside boundary pixels). By discarding boundary pixels, the incremental refinement is not perturbed by arbitrarily introduced outliers so that it has the following consequences.

- **On the precision.** The precision of the estimated model is increased by a factor ranging from 3.7 to 780 for the L2 function and from 1.4 to 22 for the Lorentzian function. Since the robust error functions handle more correctly the outliers, the precision improvements are less important for the Lorentzian error function than for the L2 function. The improvement factor decreases as the noise level increases i.e. as the noise becomes the main source of estimation error. It explains why the ranges of improvement factor are large.

- **On the computation time.** The computation time is divided by a factor ranging from 1.1 to 2 for the L2 function and from 1.1 to 2.4 for the Lorentzian function. At each step the incremental refinement complexity is lessened but it is not sufficient to explain such a reduction. The main reason is that less iterations are required to converge since the incremental refinement does not try to fit the model with the outliers. Globally the improvement factor tends to decrease with the noise level.

Finally, because it improves the precision and the computation time, we strongly recommend to discard boundary pixels with $\delta = 5$. It is done in the following experiments.

Color Handling. To analyze the influence of the grayscale conversion, we compare the results of the modified inverse compositional algorithm with and without grayscale conversion. Boundary pixels are discarded with $\delta = 5$, the gradient estimation is done using the central differences scheme and all scales are used.

The results are presented in Table 7.6. By using the grayscale conversion the computation time is reduced by a factor ranging from 1.4 to 2.6. Indeed, a reduction was expected since the input number of channels is divided by 3. For large noise value the reduction is globally more important because the input noise is divided by $\sqrt{3}$. For the precision, the results are similar but the grayscale conversion provides the best results for large noise values.

Note that the precision is similar as long as the main structures of the image are preserved by the grayscale conversion. On the other hand, we have introduced independent Gaussian noise in each channel, which may not be realistic. For a real image the noise reduction due to the grayscale conversion may not be so important. Nevertheless, we recommend the use of the grayscale conversion because it is much faster and the difference in precision is usually not remarkable.

Gradient Estimation on a Prefiltered Image. To analyze the influence of the gradient estimation, we compare the results of the modified inverse compositional algorithm using each one of the gradient kernels of Table 7.4. Boundary pixels are discarded with $\delta = 5$, all scales and the grayscale conversion are used.

The results are presented in Table 7.7 for the L2 error function and in Table 7.8 for the Lorentzian error function. The results analysis is not as simple as for the boundary handling influence and the color handling. In general, all the estimators provide similar results in terms of precision (except for the hypomode estimator that gives worse results). However, the central differences estimator provides slightly better results for small noise level, while it is the Farid 3x3 and 5x5 estimators [34] for larger noise levels. The main difference between the gradient estimators lies in the computation time. For low noise levels the computation times are similar but for large noise levels the Farid 5x5 estimator provides significantly better results.

In general, computing the gradient on a prefiltered image provides a gradient estimation more robust to noise. In addition, the prefiltered images contain less aliasing and the interpolated values are computed more precisely. Finally, it allows for a faster convergence of the incremental refinement and a more precise motion estimation. Therefore, in the following we use the Farid 5x5 estimator.

First Scale in the Gaussian Pyramid. To analyze the influence of the first scale s_0 used in the Gaussian pyramid, we compare the results of the modified inverse compositional algorithm using $s_0 \in \{0,1,2,3\}$. Boundary pixels are discarded with $\delta = 5$, the Farid 5x5 gradient estimator and the grayscale conversion are used. The results are presented in Table 7.9 for the L2 error function and in Table 7.10 for the Lorentzian error function.

- **Evolution with the number of skipped scales s_0.** As expected, the precision and the computation time both decrease with s_0, i.e. the number of scales skipped. There is a trade-off between precision and speed. By comparing the evolution between the column s_0 and $s_0 + 1$ we notice that it is less and less interesting to remove scales. Indeed, the decreasing factor for the computation time decreases with s_0 while the increasing factor for the estimation error increases with s_0. Therefore it is reasonable to only consider $s_0 \in \{0,1\}$.

- **Evolution with the noise level σ.** As σ increases, it is more and more interesting to skip the finest scale, i.e. to take $s_0 = 1$. Indeed, the decreasing factor for the computation time increases with σ while the increasing factor for the estimation error decreases with σ. For large values of σ, the estimation error is similar for $s_0 = 0$ and $s_0 = 1$ while the computation time is divided by a factor up to 2.3 for the L2 function and 4.4 for the Lorentzian function.

		L2			Lorentzian		
$\sigma = 0$	EPE	0.06268	0.00008	**0.00008**	0.00221	0.00010	**0.00010**
	Time	438	320	**311**	915	419	**404**
$\sigma = 3$	EPE	0.06296	0.00208	**0.00192**	0.00341	0.00207	**0.00192**
	Time	453	333	**323**	893	387	**378**
$\sigma = 5$	EPE	0.06394	0.00338	**0.00282**	0.00517	0.00337	**0.00282**
	Time	524	373	**355**	903	402	**391**
$\sigma = 10$	EPE	0.06646	0.00962	**0.00711**	0.01163	0.00963	**0.00714**
	Time	669	406	**386**	951	481	**459**
$\sigma = 20$	EPE	0.09194	0.02901	**0.01730**	0.03340	0.03101	**0.02147**
	Time	1104	640	**544**	1290	1109	**1069**
$\sigma = 30$	EPE	0.14683	0.06508	**0.03393**	0.06763	0.06569	**0.04447**
	Time	1319	1008	**827**	1741	1651	**1581**
$\sigma = 50$	EPE	0.24210	0.12385	**0.06515**	0.10959	0.10487	**0.07646**
	Time	1507	1454	**1313**	2139	2000	**1918**

Table 7.5: Influence of the boundary handling. Comparison between the inverse compositional algorithm presented in Section 7.2 (noted IC) and the modified inverse compositional algorithm that discards boundary pixels (noted DBP) with $\delta = 0$ and $\delta = 5$. The gradient estimation is done using the central difference scheme, all scales are used and there is no grayscale conversion. It clearly shows that discarding boundary pixels always provides significantly better results in terms of precision and computation time. The gain is less and less important as the noise level increases.

Finally, we recommend to use $s_0 \in \{0, 1\}$ with a choice depending on the context of application and the aim of the user (precision or speed).

Summary of the improvements. The study of the modifications proposed previously can be summarized as follows:

- Boundary pixels must be discarded with $\delta = 5$.
- The grayscale conversion must be used for high noise level and should be used in general since the eventual loss in precision is negligible with respect to the gain in speed.
- The central differences and the Farid 5x5 gradient estimators provide similar results. But the Farid 5x5 estimator is prefered because it is more robust to noise.
- Discarding the finest scale, i.e. taking $s_0 = 1$, allows for a significant speed up of the motion estimation with a moderate loss of precision, which decreases with the noise level.

Using this recommended modifications:

- Using $s_0 = 0$. The computation time is at least reduced by a factor 2.2. The estimation accuracy is at least improved by a factor 5 for the L2 function and 1.3 for the Lorentzian function.
- Using $s_1 = 1$. The computation time is at least reduced by a factor 3.4. For the L2 function, the estimation accuracy is at least improved by a factor 3.4. For the Lorentzian function the estimation accuracy is at most reduced by a factor 0.6.

		L2		Lorentzian	
Color handling		Color	Grayscale	Color	Grayscale
$\sigma = 0$	EPE	**0.00008**	0.00008	**0.00010**	0.00010
	Time	311	**176**	404	**261**
$\sigma = 3$	EPE	**0.00192**	0.00216	**0.00192**	0.00215
	Time	323	**208**	378	**262**
$\sigma = 5$	EPE	**0.00282**	0.00300	**0.00282**	0.00300
	Time	355	**195**	391	**261**
$\sigma = 10$	EPE	0.00711	**0.00707**	0.00714	**0.00707**
	Time	386	**210**	459	**281**
$\sigma = 20$	EPE	**0.01730**	0.01810	0.02147	**0.01929**
	Time	544	**255**	1069	**449**
$\sigma = 30$	EPE	0.03393	**0.03299**	0.04447	**0.03941**
	Time	827	**319**	1581	**806**
$\sigma = 50$	EPE	0.06515	**0.06192**	0.07646	**0.07060**
	Time	1313	**527**	1918	**1066**

Table 7.6: Influence of the color handling. Boundary pixels are discarded with $\delta = 5$, the gradient estimation is done using the central differences scheme and all scales are used. By using the grayscale conversion the computation time is reduced by a factor ranging from 1.4 to 2.6. Indeed, a reduction was expected since the input number of channels is divided by 3. For large noise value the reduction is globally more important because the input noise is divided by $\sqrt{3}$. For the precision, the results are similar but the grayscale conversion provides the best results for large noise values.

		Central Differences	Hypomode	Farid 3x3	Farid 5x5	Gaussian 3	Gaussian 6
$\sigma = 0$	EPE	**0.00008**	0.03146	0.00017	0.00026	0.00019	0.00015
	Time	**176**	210	187	202	213	207
$\sigma = 3$	EPE	**0.00216**	0.03169	0.00241	0.00269	0.00219	0.00235
	Time	208	249	216	**206**	216	208
$\sigma = 5$	EPE	**0.00300**	0.03212	0.00317	0.00351	0.00305	0.00312
	Time	**195**	230	202	205	221	209
$\sigma = 10$	EPE	0.00707	0.03451	**0.00702**	0.00749	0.00707	0.00694
	Time	**210**	252	220	**210**	240	226
$\sigma = 20$	EPE	0.01810	0.04133	0.01698	0.01782	0.01800	**0.01694**
	Time	255	306	245	**238**	281	252
$\sigma = 30$	EPE	0.03299	0.04992	**0.02917**	0.02941	0.03243	0.02940
	Time	319	375	291	**258**	349	290
$\sigma = 50$	EPE	0.06192	0.06848	0.04644	**0.04491**	0.05804	0.04788
	Time	527	568	435	**370**	572	450

Table 7.7: Influence of the gradient estimator (L2 error function). Comparison of the modified inverse compositional algorithm using the L2 error function and each one of the gradient kernels of Table 7.4. Boundary pixels are discarded with $\delta = 5$, all scales and the grayscale conversion are used. In general, all the estimators provide similar results in terms of precision (except for the hypomode estimator that gives worse results). However, the central differences estimator provides slightly better results for small noise level while it is the Farid 3x3 and 5x5 estimators for larger noise levels. For low noise levels the computation times are similar but for large noise levels the Farid 5x5 estimator provides significantly better results.

		Central Differences	Hypomode	Farid 3x3	Farid 5x5	Gaussian 3	Gaussian 6
$\sigma = 0$	EPE	**0.00010**	0.03140	0.00016	0.00024	**0.00010**	0.00012
	Time	261	277	**251**	252	279	261
$\sigma = 3$	EPE	**0.00215**	0.03162	0.00240	0.00268	0.00217	0.00234
	Time	262	285	255	**254**	284	255
$\sigma = 5$	EPE	**0.00300**	0.03203	0.00316	0.00349	0.00303	0.00311
	Time	261	297	254	**253**	283	262
$\sigma = 10$	EPE	0.00707	0.03437	0.00700	0.00746	0.00705	**0.00692**
	Time	281	341	272	**262**	318	285
$\sigma = 20$	EPE	0.01929	0.04145	**0.01693**	0.01778	0.01906	0.01698
	Time	449	452	334	**303**	514	361
$\sigma = 30$	EPE	0.03941	0.05042	0.02934	**0.02933**	0.03638	0.03091
	Time	806	739	481	**375**	814	617
$\sigma = 50$	EPE	0.07060	0.07333	0.04934	**0.04717**	0.06203	0.05439
	Time	1066	940	837	**704**	1041	942

Table 7.8: Influence of the gradient estimator (Lorentzian error function). Comparison of the modified inverse compositional algorithm using the Lorentzian error function and each one of the gradient kernels of Table 7.4. Boundary pixels are discarded with $\delta = 5$, all scales and the grayscale conversion are used. In general, all the estimators provide similar results in terms of precision (except for the hypomode estimator that gives worse results). However, the central differences estimator provides slightly better results for small noise level while it is the Farid 3x3 and 5x5 estimators for larger noise levels. For low noise levels the computation times are similar but for large noise levels the Farid 5x5 estimator provides significantly better results.

		$s_0 = 0$	$s_0 = 1$	$s_0 = 2$	$s_0 = 3$
$\sigma = 0$	EPE	**0.00026**	0.00328	0.01265	0.05672
	Time	202	131	107	**97**
$\sigma = 3$	EPE	**0.00269**	0.00562	0.01446	0.05829
	Time	206	130	110	**104**
$\sigma = 5$	EPE	**0.00351**	0.00684	0.01777	0.06094
	Time	205	128	109	**107**
$\sigma = 10$	EPE	**0.00749**	0.01261	0.02611	0.06899
	Time	210	132	113	**104**
$\sigma = 20$	EPE	**0.01782**	0.02641	0.04101	0.09040
	Time	238	132	114	**107**
$\sigma = 30$	EPE	**0.02941**	0.04276	0.07127	0.12303
	Time	258	131	112	**106**
$\sigma = 50$	EPE	**0.04491**	0.06679	0.10165	0.19240
	Time	370	160	129	**123**

Table 7.9: Influence of the first scale s_0 used in the Gaussian pyramid (L2 error function). Boundary pixels are discarded with $\delta = 5$, the Farid 5x5 gradient estimator and the grayscale conversion are used. The precision and the computation time both decrease with s_0. There is a trade-off between precision and speed. By comparing the evolution between the column s_0 and $s_0 + 1$ we notice that it is less and less interesting to remove scales. As the noise level σ increases, it is more and more interesting to skip the finest scale, i.e. to take $s_0 = 1$. For large values of σ, the estimation error is similar for $s_0 = 0$ and $s_0 = 1$ while the computation time is divided by a factor up to 2.3.

		$s_0 = 0$	$s_0 = 1$	$s_0 = 2$	$s_0 = 3$
$\sigma = 0$	EPE	**0.00024**	0.00327	0.01265	0.05667
	Time	252	122	92	**87**
$\sigma = 3$	EPE	**0.00268**	0.00561	0.01446	0.05826
	Time	254	140	114	**110**
$\sigma = 5$	EPE	**0.00349**	0.00682	0.01777	0.06101
	Time	253	143	119	**107**
$\sigma = 10$	EPE	**0.00746**	0.01257	0.02609	0.06901
	Time	262	142	115	**106**
$\sigma = 20$	EPE	**0.01778**	0.02639	0.04099	0.09019
	Time	303	146	112	**110**
$\sigma = 30$	EPE	**0.02933**	0.04272	0.07131	0.12286
	Time	375	150	116	**109**
$\sigma = 50$	EPE	**0.04717**	0.06675	0.10165	0.19220
	Time	704	158	116	**105**

Table 7.10: Influence of the first scale used in the Gaussian pyramid (Lorentzian error function). Boundary pixels are discarded with $\delta = 5$, the Farid 5x5 gradient estimator and the grayscale conversion are used. The precision and the computation time both decrease with s_0. There is a trade-off between precision and speed. By comparing the evolution between the column s_0 and $s_0 + 1$ we notice that it is less and less interesting to remove scales. As the noise level σ increases, it is more and more interesting to skip the finest scale, i.e. to take $s_0 = 1$. For large values of σ, the estimation error is similar for $s_0 = 0$ and $s_0 = 1$ while the computation time is divided by a factor up to 4.4.

7.4.3 Comparison with a SIFT+RANSAC Based Algorithm

In order to put in perspective the performance of the proposed modified inverse compositional algorithm, it is compared with a classical feature-based motion estimation algorithm. It uses as features the SIFT keypoints [72, 97] and estimates the model with a RANSAC algorithm [39]. Note that the grayscale conversion is used. A similar algorithm can be found in [85].

For the comparison, we consider the SIFT+RANSAC algorithm, the inverse compositional (IC) algorithm described in Algorithm 7.2 and the modified inverse compositional (mIC) algorithm. For the modifications we follow the recommendations of Section 7.4.2, i.e., we discard boundary pixels with $\delta = 5$ and use the Farid 5x5 kernel estimator, the grayscale conversion and $s_0 \in \{0, 1\}$.

On Synthetic Data. First we compare the algorithms using the same experimental setup and synthetic data as in Section 7.4.2. The results are presented in Table 7.11.

• **On the computation time.** The inverse compositional algorithm, modified or not, is faster than the SIFT+RANSAC algorithm. The computation time increases with the noise level while it decreases for the SIFT+RANSAC algorithm. Using $s_0 = 0$, the mIC algorithm is 5.6 to 11 times faster for the L2 function and 3 to 8.6 times faster for the Lorentzian function. Using $s_0 = 1$, the mIC algorithm is 13 to 17 times faster for the L2 function and 13 to 18 times faster for the Lorentzian function.

• **On the precision.** The IC algorithm is in general more accurate than the SIFT+RANSAC algorithm (except using the L2 function for low noise levels because of the incorrect boundary handling). On the opposite, the mIC algorithm always clearly outperforms the SIFT+RANSAC

			L2			Lorentzian		
		SIFT+R	IC	mIC $s_0 = 0$	mIC $s_0 = 1$	IC	mIC $s_0 = 0$	mIC $s_0 = 1$
$\sigma = 0$	EPE	0.03131	0.06268	**0.00026**	0.00328	0.00221	**0.00024**	0.00327
	Time	2171	438	202	**131**	915	252	**122**
$\sigma = 3$	EPE	0.04063	0.06296	**0.00269**	0.00562	0.00341	**0.00268**	0.00561
	Time	2173	453	206	**130**	893	254	**140**
$\sigma = 5$	EPE	0.04805	0.06394	**0.00351**	0.00684	0.00517	**0.00349**	0.00682
	Time	2144	524	205	**128**	903	253	**143**
$\sigma = 10$	EPE	0.15998	0.06646	**0.00749**	0.01261	0.01163	**0.00746**	0.01257
	Time	2208	669	210	**132**	951	262	**142**
$\sigma = 20$	EPE	0.82330	0.09194	**0.01782**	0.02641	0.03340	**0.01778**	0.02639
	Time	2198	1104	238	**132**	1290	303	**146**
$\sigma = 30$	EPE	0.67803	0.14683	**0.02941**	0.04276	0.06763	**0.02933**	0.04272
	Time	2177	1319	258	**131**	1741	375	**150**
$\sigma = 50$	EPE	1.22713	0.24210	**0.04491**	0.06679	0.10959	**0.04717**	0.06675
	Time	2075	1507	370	**160**	2139	704	**158**

Table 7.11: Comparison of motion estimation methods: SIFT+RANSAC (SIFT+R), inverse compositional (IC) algorithm described in Algorithm 7.2 and the modified inverse compositional (mIC) algorithm with $s_0 \in \{0,1\}$. The mIC algorithm discards boundary pixels with $\delta = 5$ and uses the Farid 5x5 kernel estimator and the grayscale conversion. The inverse compositional algorithm, modified or not, is always faster than the SIFT+RANSAC algorithm. The computation time increases with the noise level while it decreases for the SIFT+RANSAC algorithm. The IC algorithm is in general more accurate than the SIFT+RANSAC algorithm (except using the L2 function for low noise levels because of the incorrect boundary handling). On the opposite, the mIC algorithm always clearly outperforms the SIFT+RANSAC algorithm and in particular for high noise levels.

algorithm and in particular for high noise levels. The precision is 14 to 130 times better for $s_0 = 0$ and 7 to 31 times better for $s_0 = 1$.

• **General remarks on the experiments.** We observed that the behavior of the algorithm for other reference images is similar to the reported results. However the improvement factors may differ with the input images. Also, we noticed that in the context of low quality input images the precision may be better while using $s_0 = 1$.

In our tests the reference and warped images are linked with moderate deformations without occlusion nor contrast change. For larger deformations, the inverse compositional based algorithms may be outperformed by feature-based methods. The occlusions highly increase the error for the L2 error function but are well handled by the robust error functions.

On Real Data. Secondly, we compare the methods on two images taken from the "Lunch Room" sequence of the PASSTA Dataset [80]. Between the acquisitions, the camera was rotated around its optical center so that the images are linked by an homography. The images I_1 and I_2, of size 2048×2048 and stored in the JPEG format, are displayed in Figure 7.4. For the IC and mIC algorithms, we only use the Lorentzian error function.

Since the real motion $\mathbf{p}^\star$ is unknown, the end-point error cannot be computed. Let denote by $\mathbf{p}$ the estimated motion by a given method. The algorithm precision is indirectly evaluated by considering the residual $I_1 - I_2(\Psi(\cdot;\mathbf{p}))$, which is computed by bicubic interpolation. The root mean square over the channels of the residuals are displayed in Figure 7.5. The residuals are inevitably corrupted by JPEG artifacts, noise and interpolation error. The JPEG artifacts and noise are noticeable on flat areas. The interpolation error is localized at discontinuities and can be confounded with the consequence of a bad registration. The SIFT+RANSAC residual has

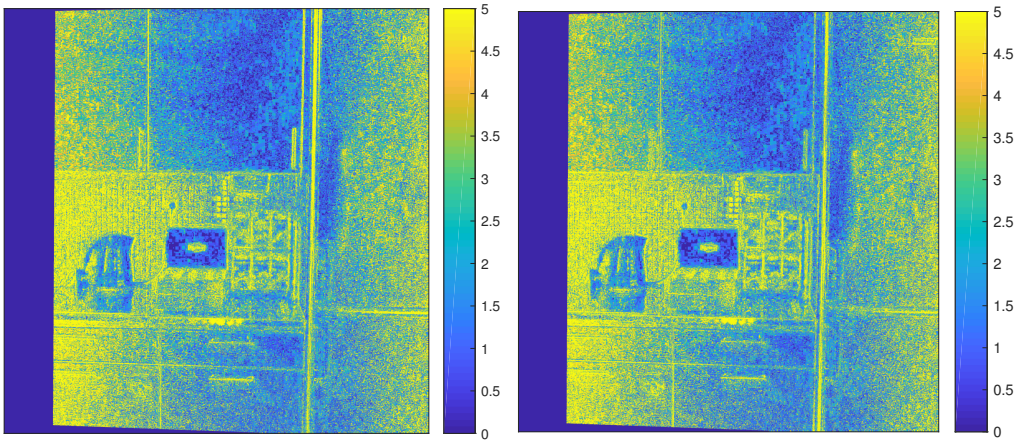
(a) Reference image I_1 (b) Warped image I_2

Figure 7.4: Real data used in Section 7.4.3. The images are taken from the "Lunch Room" sequence of the PASSTA Dataset [80]. Between the acquisitions, the camera was rotated around its optical center so that the images are linked by an homography. The images of size 2048×2048 are stored in JPEG format.

significantly higher values than the IC and mIC residuals. In particular, at the top-right corner of the image (refrigerator edge) the registration is not precise enough. It can be explained by the low density of SIFT keypoints in this zone. The mIC $s_0 = 0$ residual is slightly lower than the IC and mIC $s_0 = 1$ residuals on average but more importantly on the image discontinuities, which is interpreted as a better motion estimation.

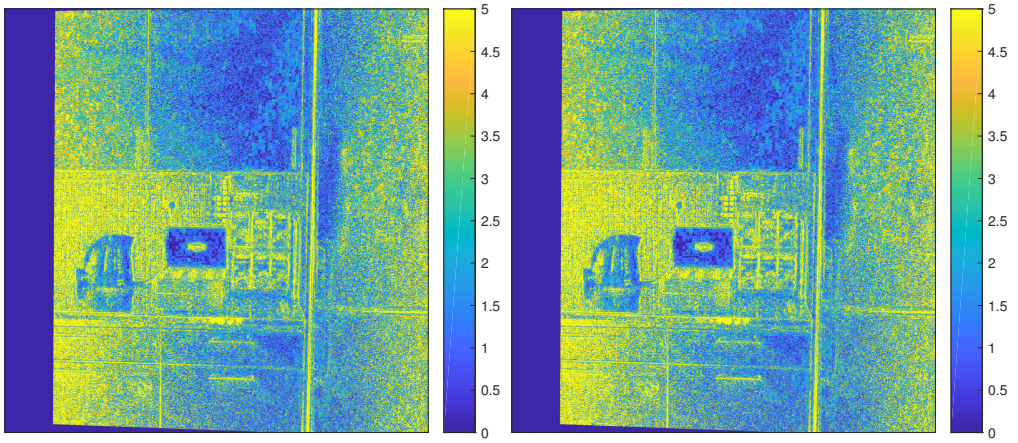
By replacing the real motion $\mathbf{p}^\star$ by an estimated motion in the end-point error definition in (7.33), we define the end-point difference between two motions. It allows for a comparison of the estimated motions. The end-point difference fields between the mIC $s_0 = 0$ and the three other methods are shown in Figure 7.6. The mIC s_0 and SIFT+RANSAC results are considerably different. On average, the end-point difference is greater than 0.5 pixel. They mainly differ at the top-right corner of the image, where the SIFT+RANSAC residual is higher. The mIC $s_0 = 0$ and mIC $s_0 = 1$ mainly differ at the top-left corner of the image. The average end-point difference of 0.15 pixel is not negligible. The finest scale must be used to achieve sufficient precision. The IC and mIC s_0 results are closer but still significantly different. On average, the end-point difference is greater than 0.05 pixel. They mainly differ at the boundaries of the domain, which is a consequence of the different boundary pixels handling but also of the images content.

The estimated motions are computed in respectively 39, 42, 18 and 2 seconds for the SIFT+RANSAC, IC and mIC algorithms. The IC algorithm is slower because at each scale the incremental refinement requires a large amount of iterations. For the mIC algorithm it is the case only for the finest scale. It explains why not using the finest scale divides the computation time by 9.



(a) SIFT+RANSAC (3.865058, 39)

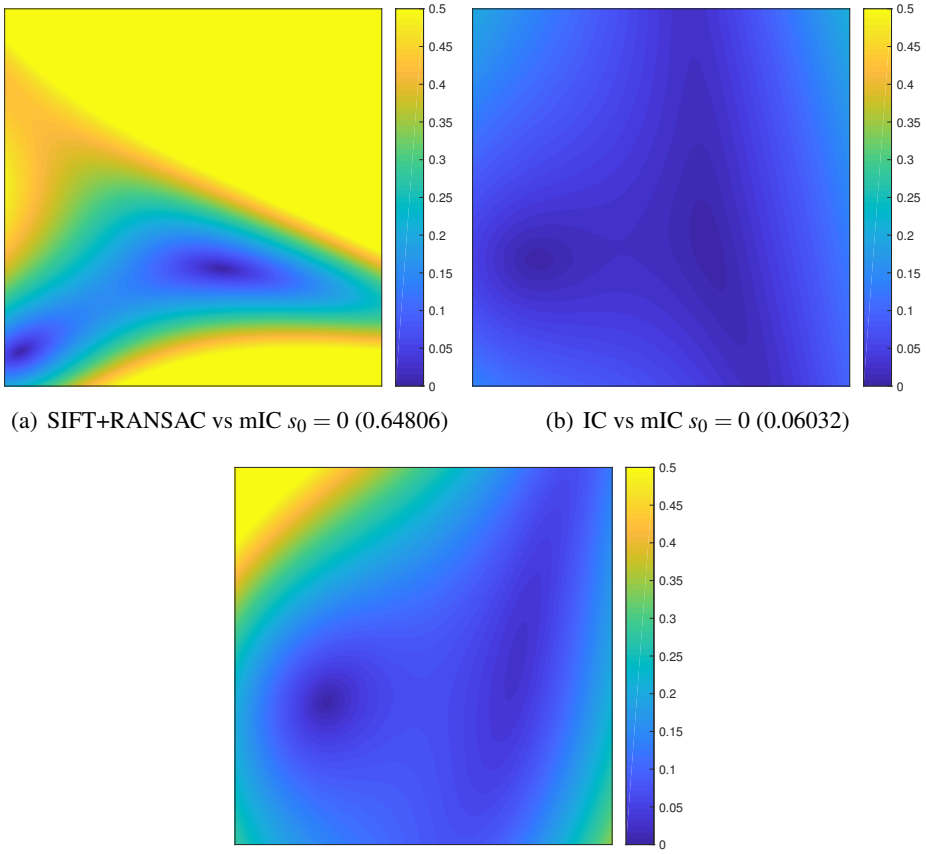
(b) IC (3.601141, 42)



(c) mIC $s_0 = 0$ (3.598951, 18)

(d) mIC $s_0 = 1$ (3.599503, 2)

Figure 7.5: Example of motion estimation on real data. The images correspond to the root mean square over the channels of the residuals $I_1 - I_2(\Psi(\cdot;\mathbf{p}))$, which is computed using bicubic interpolation. The first value between parentheses is the RMSE between I_1 and $I_2(\Psi(\cdot;\mathbf{p}))$. The second value is the computation time expressed in seconds. The residuals are inevitably corrupted by JPEG artifacts, noise and interpolation error. The JPEG artifacts and noise are noticeable on flat areas. The interpolation error is localized at discontinuities and can be confounded with the consequence of a bad registration. The SIFT+RANSAC residual has significantly higher values than the IC and mIC residuals. In particular, at the top-right corner of the image (refrigerator edge) the registration is not precise enough. It can be explained by the low density of SIFT keypoints in this zone. The mIC $s_0 = 0$ residual is slightly lower than the IC and mIC $s_0 = 1$ residuals on average but more importantly on the image discontinuities, which is interpreted as a better motion estimation.



(a) SIFT+RANSAC vs mIC $s_0 = 0$ (0.64806)

(b) IC vs mIC $s_0 = 0$ (0.06032)

(c) mIC $s_0 = 1$ vs mIC $s_0 = 0$ (0.14505)

Figure 7.6: Example of end-point difference fields on real data. The value between parentheses is the average end-point difference. The mIC s_0 and SIFT+RANSAC results are considerably different. On average, the end-point difference is greater than 0.5 pixel. They mainly differ at the top-right corner of the image, where the SIFT+RANSAC residual is higher. The mIC $s_0 = 0$ and mIC $s_0 = 1$ mainly differ at the top-left corner of the image. The average end-point difference of 0.15 pixel is not negligible. The finest scale must be used to achieve sufficient precision. The IC and mIC s_0 results are closer but still significantly different. On average, the end-point difference is greater than 0.05 pixel. They mainly differ at the boundaries of the domain, which is a consequence of the different boundary pixels handling but also of the images content.

7.5 Conclusion

In this chapter we detailed the inverse compositional algorithm and proposed to modify it with a correct boundary handling, the grayscale conversion, a more robust gradient estimation applied to a prefiltered image and by skipping scales in the multiscale coarse-to-fine scheme. Discarding boundary pixels is the main source of improvements and must always be done. In general, we recommend to discard boundary pixels with $\delta = 5$ and to use the grayscale conversion and the Farid 5x5 gradient estimator. With this settings, the estimation accuracy is at least improved by a factor 1.3 while the computation time is at least reduced by a factor 2.2 when all the scales are used and by a factor 3.4 when the finest scale is not used. For moderate transformations, the modified algorithm outperforms the classical feature-based methods using the SIFT keypoints and the RANSAC algorithm.

For low quality images, for instance because of noise, using the Farid 5x5 gradient estimator and the grayscale conversion provides the best results. For high quality images, using the central differences gradient estimator without grayscale conversion may provide slightly better results. When efficiency is preferred over accuracy, the grayscale conversion must be used and the finest scale must be skipped.

The modified inverse compositional algorithm will be used in Chapter 8 in the proposed two-step method for mosaicked images.

Chapter 8

Registration of Mosaicked (or CFA) Images

Abstract

A mosaicked (or CFA) image is a digital image where only one of the three color channels has been captured at each pixel location. The corresponding grayscale image contains a mosaic structure incoming from the color filter array (CFA). The acquired RAW image is a typical example of mosaicked image. The most common pattern for the CFA, and the one that is considered in this chapter, is the Bayer pattern [12]. In many applications, the geometric transformation between two mosaicked images has to be estimated without knowing the underlying color images. Unfortunately there is no standard and satisfactory method for mosaicked images. Existing registration methods designed for classical images cannot be used directly and a preprocessing step is required. In this chapter, we introduce two-step methods for the registration of mosaicked images. First, the two mosaicked images are converted into non-mosaicked (grayscale) images by lowpass filtering. According to [6], these filtered images estimate the luminance information contained in the mosaicked images. Then, the transformation is estimated by applying a pre-existing registration method designed for classical images. The performances of the proposed methods are evaluated experimentally for several lowpass filters and pre-existing registration methods. We conclude that a perfect lowpass filter should be applied and that the modified inverse compositional algorithm with robust error function (see Chapter 7) should be used. This recommended method is both accurate and efficient, and will be used in our image formation algorithm from RAW images (see Chapter 11).

Contents

8.1 Introduction

At least three color components per pixel location are required to produce a digital color image. Typically, the red, green and blue colors are used. An RGB image is composed of the three corresponding grayscale images. However, to reduce production cost, most digital cameras use a single sensor covered by a color filter array (CFA), which is arranged in a mosaic pattern. The acquired RAW image is a typical example of what is called a mosaicked (or CFA) image. The most common pattern for the CFA, and the one that is considered in this chapter, is the Bayer pattern [12].

Color demosaicking is the image reconstruction process consisting in estimating the missing color samples [68, 73]. As demosaicking involves some form of interpolation from neighboring pixel values, this may lead to blurring and false colors (a.k.a color aliasing). To avoid the introduction of these artefacts, multi-image fusion methods taking as input RAW images have been developed [35, 36, 37, 48, 61, 131, 136]. Such methods perform joint demosaicking, denoising and, possibly, super-resolution. Multi-image fusion methods can generally be decomposed into two main steps: the registration step, where the images are expressed in a common system of coordinates, and the combination step, where the data are combined to form an image.

In this chapter, we focus on the registration of mosaicked images where the transformation between the two images has to be estimated while the underlying color images are unknown. Although an accurate subpixel registration is crucial in many applications, there is no standard and satisfactory method for mosaicked images. Because of the particular content of these images, existing registration methods designed for classical grayscale or color images cannot be used directly and a preprocessing step is required.

The most natural solution is to register demosaicked versions of the mosaicked images. However the registration precision is limited by the artefacts introduced during the demosaicking. In the super-resolution method from RAW images of [48], translations are first roughly estimated on pairs of demosaicked images obtained by bilinear interpolation [6]. This is done by minimizing the simple sum of squared difference (SSD) criterion followed by a parabola fitting. Then, the estimations are refined during the iterative scheme that builds the high-resolution color image from all the RAW images.

Another solution proposed in [131] is to only use the low-frequency content of the mosaicked images. The registration is done by applying the Fourier-based method of [130], which is designed for aliased images, directly to the mosaicked images. Although the motion estimation is not accurate enough and is limited to rigid transformations, this work is based on the two following fundamental ideas.

1. First, the knowledge of the intensity of the light, i.e. the luminance information, is generally sufficient to perform accurate registration. Indeed, in many registration algorithms, color images are first converted into grayscale images by (weighted) average of the channels. For instance, such a conversion was proposed in Chapter 7 to handle color.

2. Second, it is shown in [6] that the luminance information is mainly localized in the low-frequency of the mosaicked images. Thus, the luminance component of a mosaicked image is not directly available but can be estimated by lowpass filtering.

In this chapter, we introduce two-step methods for the registration of mosaicked images. First, the two mosaicked images are converted into non-mosaicked (grayscale) images by lowpass filtering. According to [6], these filtered images estimate the luminance information contained in the mosaicked images. Then, the transformation is estimated by applying a pre-existing registration method designed for classical images. The performances of the proposed methods are evaluated experimentally for several lowpass filters and pre-existing registration methods. We recommend the use of a perfect lowpass filter, which cancels half of the spectrum along

each direction, and of the modified inverse compositional algorithm with robust error function (see Chapter 7). These recommendations lead to a method that is both accurate and efficient.

The chapter is organized as follows. First, two-step methods for the registration of mosaicked images are presented in Section 8.2. Then, in Section 8.3, the performances are evaluated experimentally.

8.2 Registration of Mosaicked Images

In the rest of this chapter, the color filter array considered is the classical Bayer filter RGGB [12]. Let I be a color image of size $M \times N$ whose red, green and blue channels are respectively denoted R, G and B. The mosaicked image I_{CFA}, corresponding to I, is the grayscale image of size $M \times N$ defined by

$$\forall (k,l) \in \Omega_{M,N}, \quad (I_{\text{CFA}})_{k,l} = \begin{cases} R_{k,l} & \text{if } k \text{ and } l \text{ are even,} \\ B_{k,l} & \text{if } k \text{ and } l \text{ are odd,} \\ G_{k,l} & \text{otherwise.} \end{cases} \tag{8.1}$$

An example of mosaicked image is shown in Figure 8.1.

Let I^1 and I^2 be two color images of size $M \times N$. We recall that the registration of I^1 and I^2 consists in estimating the geometric transformation $\varphi \in \sigma(\mathbb{R}^2)$ such that

$$\forall \mathbf{x} \in \Omega_{M,N}, \quad I^1(\mathbf{x}) \simeq I^2(\varphi(\mathbf{x})). \tag{8.2}$$

The registration of mosaicked images consists in estimating φ from the mosaicked images I^1_{CFA} and I^2_{CFA} (without knowing the underlying color images I^1 and I^2). Existing registration methods cannot be applied directly on the mosaicked images because of the mosaic structure (see Figure 8.1(b)) and of the high-frequencies artefacts (see Figure 8.1(d)).

In this section, we present two-step registration methods for mosaicked images. The principle is to apply pre-existing registration methods on grayscale images that are built from the mosaicked ones. First, the conversion from mosaicked to non-mosaicked images is considered in Section 8.2.1. Then, Section 8.2.2 details the proposed registration algorithm for mosaicked images.

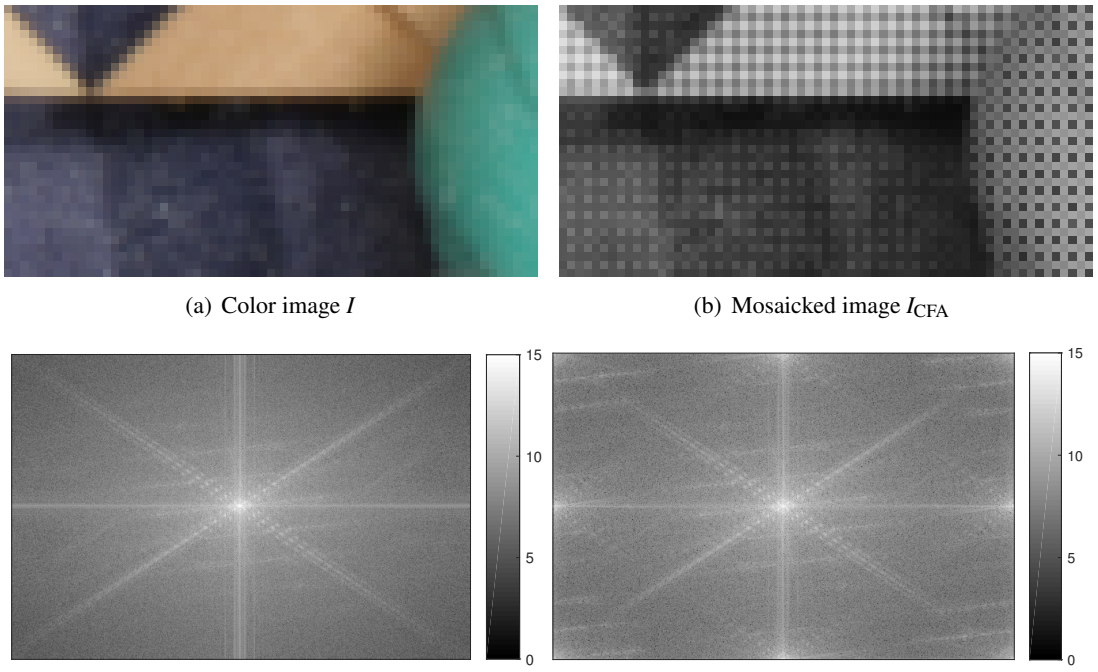
8.2.1 From Mosaicked to Non-mosaicked Images

The mosaicked images are grayscale images containing artefacts that deteriorate the performance of registration algorithms. A natural way of handling these artefacts consists in transforming the mosaicked images before the motion estimation. The considered transformation should convert mosaicked images into grayscale images whose contents allow for an accurate registration. The mosaic structure and the high-frequencies artefacts should be removed/reduced. In the following, this process is called the conversion from mosaicked to non-mosaicked images.

First, two classical, but not satisfactory, conversions are presented.

Channel extraction. A sub-sampled version of a channel can be directly obtained by extracting one in four pixels from a mosaicked image. However, the motion estimation on extracted channels is not in general precise enough since these images are strongly aliased and correspond to a coarser scale.

Image demosaicking. As presented in the introduction, a color image can be reconstructed from a mosaicked image using a color demosaicking method. Then, the mean of the three channels gives a grayscale image of the same size as the mosaicked one. Even though demosaicking may introduce blurring and color aliasing, the registration on demosaicked images provides in general significantly better results than the registration on extracted channels.



(a) Color image I (b) Mosaicked image I_{CFA}

(c) Spectrum of (a) (d) Spectrum of (b)

Figure 8.1: Example of a color image and its mosaicked version. Only details are shown in (a) and (b). The spectra correspond to the unnormalized discrete Fourier transform in logarithmic scale $u \mapsto \log(1+u)$. The root mean square over the channels is shown in (c). The mosaic structure is clearly visible in (b). The spectrum of the mosaicked image in (d) contains high-frequency structures that are not present in original spectrum in (a). This corresponds to the chrominance information.

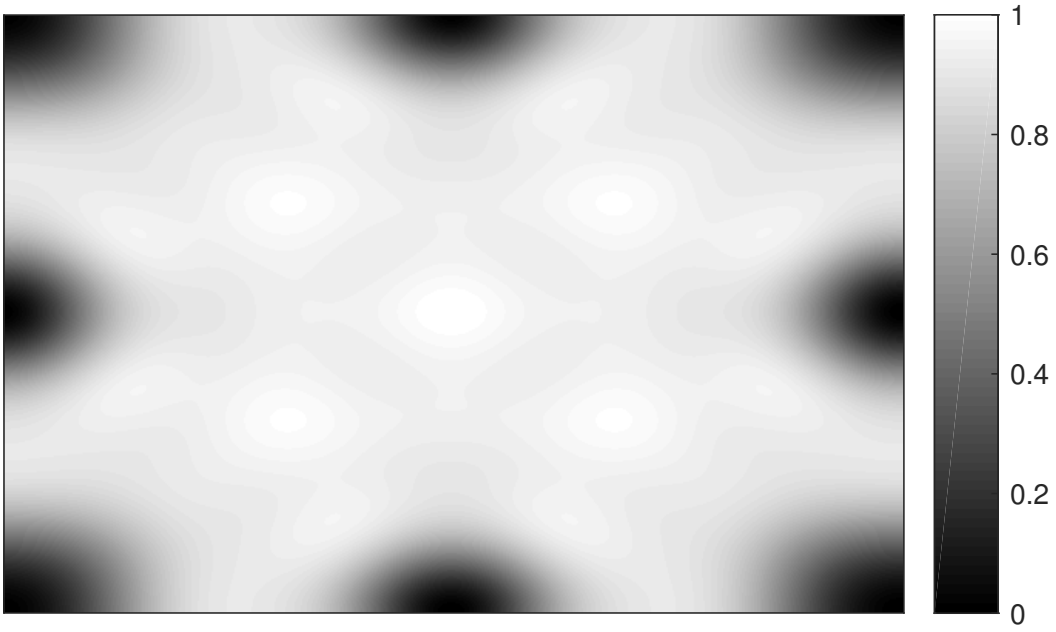


Figure 8.2: Frequency response of the HVS filter F defined in (8.6). The eight dark spots correspond to the location of the chrominance information in the Fourier domain.

In the following we consider the simple, but computationally efficient, demosaicking method by bilinear interpolation [6]. The reconstructed channels are obtained by filtering the extracted channels, which are filled by zeros at unknown pixel locations. The bilinear filter for the red and blue channels is given by

$$F_{R,B} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{8.3}$$

while the bilinear filter for the green channel is given by

$$F_G = \frac{1}{4}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{8.4}$$

Proposed conversion by lowpass filtering. Better results can be obtained by considering the following conversion approach that does not require demosaicking. It relies on the two fundamental ideas that were used in [131]:

1. The knowledge of the intensity of the light, i.e. the luminance information, is generally sufficient to perform accurate registration.
2. As shown in [6], the luminance information is mainly localized in the low-frequency of the mosaicked images while the chrominance information is mainly localized around the eight dark spots of Figure 8.2.

Thus, we propose to build non-mosaicked images, which result from the estimation of the luminance information by lowpass filtering of the mosaicked images. As the frequency locations of the luminance and chrominance overlap, a compromise has to be made between bandwidth and separation. A too small bandwidth gives a blurry image while a wrong separation of the two components leads to mosaic artefacts. In the case of a strong aliasing, for instance in a super-resolution context, a smaller bandwidth may be preferable. We propose to consider the following filters.

• **Gaussian filter.** The Gaussian filter of standard deviation σ_0, noted G_{σ_0}, is defined by

$$G_{\sigma_0}(x,y) = \frac{1}{2\pi\sigma_0^2}\exp\left(-\frac{x^2+y^2}{2\sigma_0^2}\right). \tag{8.5}$$

Among all the possible ways of applying G_{σ_0} [45], the finite impulse response (FIR) method is chosen. The higher σ_0 the better the separation, but also the smoother the filtered image. In the following we take $\sigma_0 = 1$, which is a good compromise.

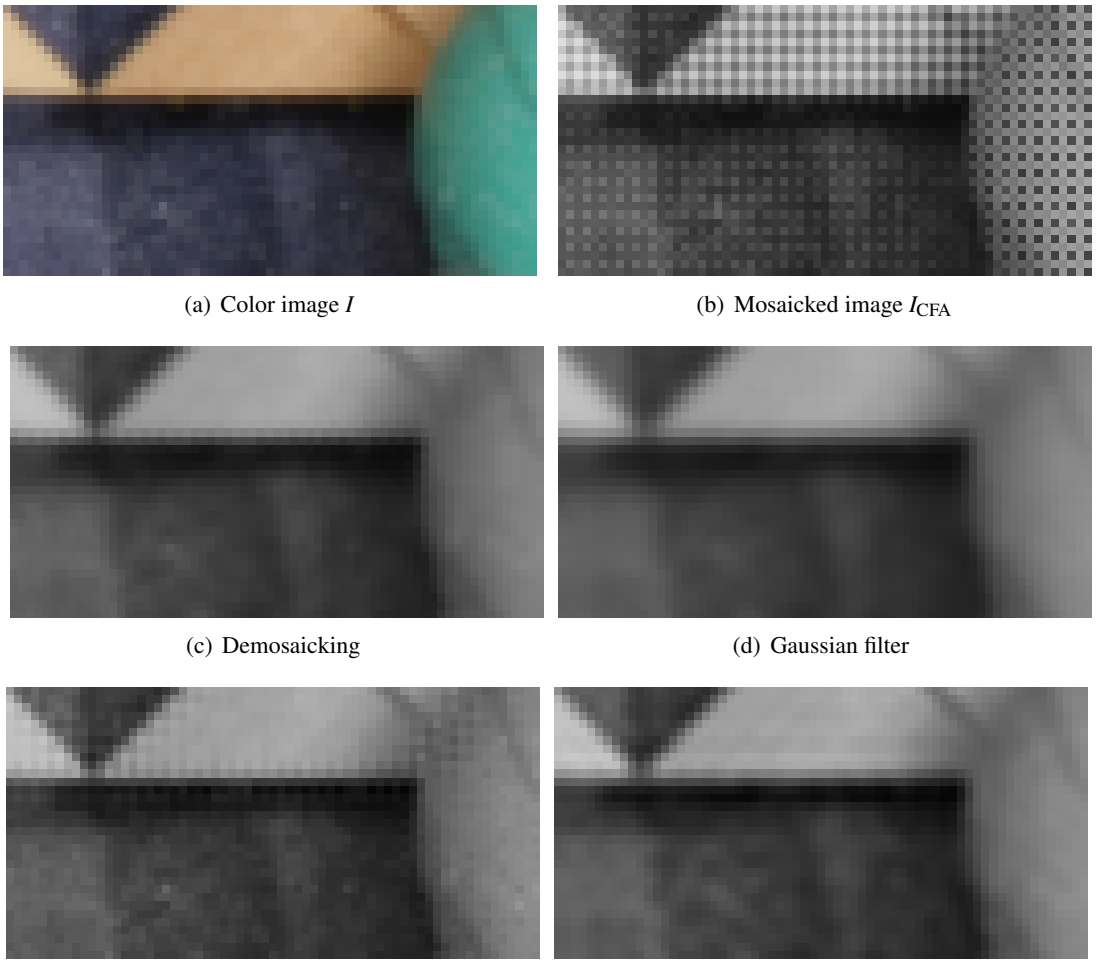
• **The HVS filter.** In [6], the luminance information is estimated using the 11×11 filter F defined by

$$F = \frac{1}{128}\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & -5 & 3 & -9 & 3 & -5 & 1 & -1 & 0 \\ 1 & 0 & 2 & 3 & 1 & 7 & 1 & 3 & 2 & 0 & 1 \\ 0 & -2 & 1 & -9 & 7 & 104 & 7 & -9 & 1 & -2 & 0 \\ 1 & 0 & 2 & 3 & 1 & 7 & 1 & 3 & 2 & 0 & 1 \\ 0 & -1 & 1 & -5 & 3 & -9 & 3 & -5 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{8.6}$$

The frequency response of the filter F is shown in Figure 8.2. The eight dark spots correspond to the location of the chrominance information in the Fourier domain. The experimental process leading to this filter is described in Section IV of [6]. As its theory is based on the human visual system (HVS), we refer to F as the HVS filter.

• **A perfect lowpass filter.** The perfect lowpass filter with ratio r is defined in the Fourier domain by the indicator function of $[-(1-r)\pi,(1-r)\pi]^2$. Along each dimension a ratio r of the frequencies is killed. Because the indicator function is discontinuous, the filtering theory of Chapter 4 does not apply. However in practice the DFT computations of Algorithm 4.1 can still be used. This application of the perfect lowpass filter is equivalent to the spectrum clipping with ratio r introduced in Section 6.2.1. In the following we take $r = \frac{1}{2}$ because $[-\frac{\pi}{2},\frac{\pi}{2}]^2$ is a part of the spectrum of mosaicked images that in general does not seem affected by the artefacts (see Figure 8.4(a)-(b)).

Examples of conversion from mosaicked to non-mosaicked images are shown in Figure 8.3. The corresponding spectra are in Figure 8.4. The mosaicked image I_{CFA} in Figure 8.3(b) is obtained by applying the Bayer filter to the color image I in Figure 8.3(a). The channel average of the demosaicked image obtained by bilinear interpolation is compared to the lowpass filtering of I_{CFA} by the three filters introduced previously. Mosaic structures around the edges are still visible for the demosaicked image and the HVS filter. Indeed, the high-frequency structures are not removed from the spectra. The HVS filter gives the image with the most details (in particular in the textured areas). The other results are blurry since more high-frequency content is filtered. The filtered image by the perfect lowpass seems to contain more details than the demosaicked one.

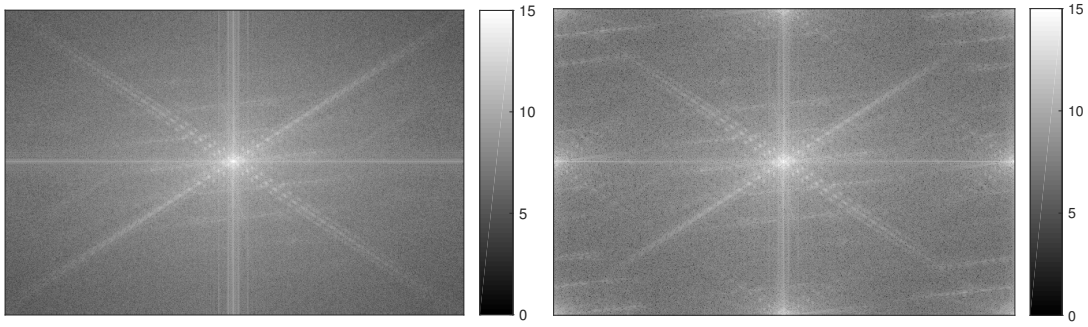


(a) Color image I (b) Mosaicked image I_{CFA}

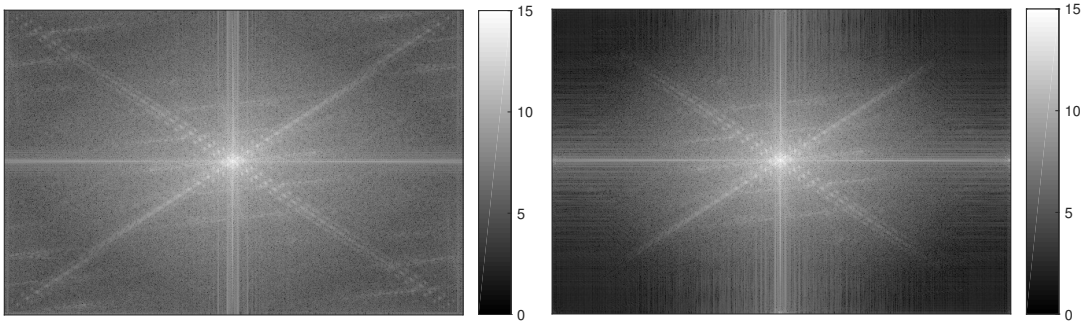
(c) Demosaicking (d) Gaussian filter

(e) HVS filter (f) Perfect lowpass filter

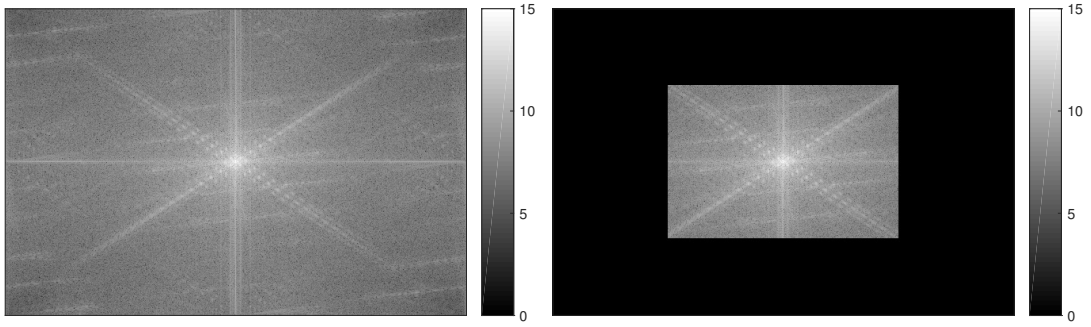
Figure 8.3: Conversions from mosaicked to non-mosaicked images (spatial domain). The mosaicked image I_{CFA} in (b) is obtained by applying the Bayer filter to the color image I in (a). The channel average of the demosaicked image obtained by bilinear interpolation in (c) is compared to the lowpass filtering of I_{CFA} by the three filters introduced in Section 8.2.1. Mosaic structures around the edges are still visible in (c) and (e). The image with the most details, in particular in the textured areas, is (e). The other results are blurry. (f) seems to contain more details than (c).



(a) Color image I (b) Mosaicked image I_{CFA}



(c) Demosaicking (d) Gaussian filter



(e) HVS filter (f) Perfect lowpass filter

Figure 8.4: Conversion from mosaicked to non-mosaicked images (spectra). The spectra correspond to the unnormalized discrete Fourier transform in logarithmic scale $u \mapsto \log(1+u)$. The root mean square over the channels is shown in (a). The high-frequency structures are not removed in (c) and (e), which explains the mosaic structure around the edges in Figure 8.3.

8.2.2 Algorithm

Given a conversion method from mosaicked to non-mosaicked images and a base registration method, mosaicked images are registered using Algorithm 8.1. Intensity-based registration methods should be preferred to Fourier-based methods as they are more robust and not limited to rigid transformations. Typically, the modified inverse compositional algorithm, introduced in Chapter 7, with a robust error function is a good candidate for the base registration method as it is efficient and robust. In particular, Algorithm 8.2 is obtained by following the recommendations of Section 8.3, i.e., using a perfect lowpass filter and the modified inverse compositional algorithm.

Algorithm 8.1: Registration of mosaicked images (general algorithm)

input : A reference mosaicked image I^1_{CFA}, a warped mosaicked image I^2_{CFA}, a conversion method and a base registration method
output: An estimated transformation φ

1 Compute the images I^1_{gray} and I^2_{gray} from I^1_{CFA} and I^2_{CFA} using the conversion method
2 Compute the estimated transformation φ between I^1_{gray} and I^2_{gray} using the base registration method

Algorithm 8.2: Registration of mosaicked images (recommended algorithm)

input : A reference mosaicked image I^1_{CFA} and a warped mosaicked image I^2_{CFA}
output: An estimated transformation φ

1 Compute the images I^1_{gray} and I^2_{gray} from I^1_{CFA} and I^2_{CFA} using the perfect lowpass filter in $[-\frac{\pi}{2}, \frac{\pi}{2}]^2$
2 Compute the estimated transformation φ between I^1_{gray} and I^2_{gray} using the modified inverse compositional algorithm with Lorentzian error function (see 7)

Remark: In practice, for real-world mosaicked images, a preprocessing step may be required to achieve better performance. For instance, a variance stabilizing transformation (VST) should be applied to RAW images to approximate a homoscedastic noise. In addition, the base registration method may require an equalization of the input image constrasts. These two operations are used in the preprocessing step of Chapter 11.

8.3 Experiments

In this section, we evaluate experimentally the performance of the registration methods for mosaicked images that have been proposed in Section 8.2 (see Algorithm 8.1). The results confirm the choices of the perfect lowpass filter and of the modified inverse compositional (see Algorithm 8.2). First, the experimental setup for generating synthetic data and measuring the registration error is described in Section 8.3.1. Then, the impact of the conversion from mosaicked to non-mosaicked images is studied in Section 8.3.2. Finally, the results obtained using different base registration methods are compared in Section 8.3.3.

8.3.1 Experimental Setup

To evaluate the performance of a registration method, we consider an experimental setup similar to the one developed in Chapter 7. As described in Section 7.4.1, a sequence of warped images, whose transformations are known, is built from a color image. The mosaicked images are then obtained by applying the RGGB Bayer filter. For each image, the error between the estimated transformation and the ground truth transformation is expressed in terms of end-point error (EPE). The mean of the error among the sequence gives an evaluation of the performance of the method.

		Demosaicking	Gaussian	HVS	Perfect	Red channel
$\sigma = 0$	EPE	0.00464	0.00906	0.00389	**0.00360**	0.01615
	Time	188	404	191	**149**	38
$\sigma = 3$	EPE	0.00669	0.01254	0.00583	**0.00559**	0.01877
	Time	195	378	197	**151**	55
$\sigma = 5$	EPE	0.01127	0.01595	0.00998	**0.00979**	0.02230
	Time	195	373	200	**153**	56
$\sigma = 10$	EPE	0.01734	0.02179	0.01561	**0.01550**	0.03834
	Time	208	461	216	**165**	59
$\sigma = 20$	EPE	0.03867	0.03813	**0.03340**	0.03352	0.08158
	Time	244	451	259	**204**	67
$\sigma = 30$	EPE	0.06081	0.06485	**0.05252**	0.05327	0.11601
	Time	369	472	393	**344**	107
$\sigma = 50$	EPE	0.12019	0.11296	**0.10655**	0.10812	0.27328
	Time	564	598	571	**531**	139

Table 8.1: Influence of the conversion from mosaicked to non-mosaicked images. The modified inverse compositional algorithm with Lorentzian error function (see Chapter 7) is used as the base registration method. Using the extracted red channel is the most computationally efficient method but the end-point error is the highest. All the other methods are computationally less efficient but lead to more accurate estimations. For moderate noise levels ($\sigma \leq 5$), the end-point error is below or around one hundredth of pixel. As the noise level increases, the performances of the methods converge. Globally, the filtering with the perfect lowpass provides the best results both in terms of accuracy and efficiency.

More precisely, we take as input image I the *RubberWhale* image, which is a color image of size 584×388 taken from the Middlebury database [10]. Similar results are obtained using other images. For all the experiments, the same sequence of $N_{\text{images}} = 1000$ non-noisy mosaicked images is used. The transformations are random homograhies obtained using Algorithm 6.4 with $L = 5$. The interpolation method used to transform I is the periodic plus smooth version of the B-spline of order 11, i.e., the *p+s-spline11* interpolation method, with half-symmetric extension (see Chapter 6). The mosaicked sequence is obtained by applying the RGGB Bayer filter as described in (8.1). Then, to get noisy images, a Gaussian white noise of level $\sigma \in \{0, 3, 5, 10, 20, 30, 50\}$ is added. As preprocessing, each mosaicked noisy image is cropped in a band of $\delta = 20$ pixels.

In order to study the performance of each method, we consider the end-point error and the computation time. The displayed computation time corresponds to the CPU time used for the $N_{\text{images}} = 1000$ motion estimations and is expressed in seconds. Note that it also corresponds to the average computation time per image in milliseconds. The experiments were made using an Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz (on a single thread).

8.3.2 Impact of the Conversion

First, we consider the impact of the conversion from mosaicked to non-mosaicked images. This is done using the five methods described in Section 8.2.1. The modified inverse compositional algorithm, introduced in Chapter 7, with the Lorentzian error function (and the recommended parameter) is then used as the base registration method. The results are presented in Table 8.1.

As expected, performing the base registration on one of the channels is the most computationally efficient method since the extracted channels are four times smaller images obtained by subsampling. The counterpart is that the end-point error is the highest because these images

		SIFT + RANSAC	IC	M-IC
$\sigma = 0$	EPE	0.01932	0.01173	**0.00360**
	Time	1298	458	**149**
$\sigma = 3$	EPE	0.03002	0.01309	**0.00559**
	Time	1334	453	**151**
$\sigma = 5$	EPE	0.04929	0.01678	**0.00979**
	Time	1393	451	**153**
$\sigma = 10$	EPE	0.15314	0.02391	**0.01550**
	Time	1671	447	**165**
$\sigma = 20$	EPE	0.49535	0.04903	**0.03352**
	Time	2575	424	**204**
$\sigma = 30$	EPE	0.83937	0.07455	**0.05327**
	Time	3029	463	**344**
$\sigma = 50$	EPE	1.84189	0.13512	**0.10812**
	Time	3003	579	**531**

Table 8.2: Influence of the base registration method. The conversion to non-mosaicked images is done by perfect lowpass filtering. For the inverse compositional (IC) and the modified inverse compositional (M-IC) algorithms, the Lorentzian error function is used. The M-IC algorithm leads to the best results for all noise levels both in terms of efficiency and accuracy. Thanks to the correct handling of the boundary pixels and to the prefiltering before the gradient estimation, it is able to achieve a precise motion estimation in only a few iterations even if the non-mosaicked images contain artefacts.

correspond to a coarser scale and are strongly aliased.

All the other methods are computationally less efficient but lead to more accurate estimations. For moderate noise levels ($\sigma \leq 5$), the end-point error is below or around one hundredth of pixel. Note that as the noise level increases the performances of the methods converge. Globally, filtering with the perfect lowpass provides the best results both in terms of accuracy and efficiency.

Conclusion: The perfect lowpass filter should be used to convert from mosaicked to non-mosaicked images.

8.3.3 Comparison of Different Base Registration Methods

We now compare the results obtained using different base registration methods. The conversion from mosaicked to non-mosaicked images is done by the perfect lowpass filtering. The base registration methods considered are the SIFT+RANSAC algorithm, the inverse compositional (IC) algorithm and its modified version (M-IC). These methods were introduced in Chapter 7. For the IC and M-IC algorithm the Lorentzian error function and the recommended parameters are used. The results are presented in Table 8.2.

The modified inverse compositional algorithm leads to the best results for all noise levels both in terms of efficiency and accuracy. Thanks to the correct handling of the boundary pixels and to the prefiltering before the gradient estimation, it is able to achieve a precise motion estimation in only a few iterations even if the non-mosaicked images contain artefacts.

Conclusion: The modified inverse compositional algorithm with robust error function should be used as the base registration method.

8.4 Conclusion

In this chapter, we introduced two-step methods for the registration of mosaicked images. First, the two mosaicked images are converted into non-mosaicked images by lowpass filtering. Following [6], these filtered images estimate the luminance information contained in the mosaicked images. Then, the transformation between the mosaicked images is estimated by applying a pre-existing registration method designed for classical images.

The performances of the proposed methods were evaluated experimentally for several lowpass filters and pre-existing registration methods. We recommend the use of a perfect lowpass filter, which cancels half the spectrum along each direction, and of the modified inverse compositional algorithm with robust error function (see Chapter 7). This recommended method is both accurate and efficient, and will be used in our image formation algorithm from RAW images (see Chapter 11).

Part III

Image Fusion/Formation

Chapter 9

Fast and Low Memory Image Fusion using Classical Kernel Regression

Abstract

In this chapter we consider irregularly sampled data fitting by kernel regression (KR) [117] and propose a fast and low memory image fusion method that is designed for a large number of images. Using KR, the intensity value of the data is locally approximated by a polynomial expansion, whose coefficients are obtained by solving a weighted linear regression (with weights built from a kernel function). We show that the linear systems involved can be obtained by a data accumulation. Classical kernel regression (CKR) is the most simple and efficient case where the weights only depend on the data spatial repartition. As it is equivalent to a local linear filtering [41], CKR introduces blur. We introduce the asymptotic equivalent filter (AEF), an approximation of the actual equivalent filter. In the proposed image fusion algorithm the combination part, using CKR with a Gaussian kernel, is split into an accumulation part, where the images are processed sequentially, and an image computation part. The blur, introduced by CKR, is inverted by applying the inverse of the AEF. The registration method used is the one introduced in Chapter 7. The choice of the optimal parameters and the evaluation of the performance of our algorithm are the subjects of Chapter 10. The combination part of our image fusion method is adapted to mosaicked images in Chapter 11.

Contents

9.1 Introduction

In this chapter we consider the problem of image fusion from a large number of images. While registration can be performed sequentially on the images, most combination methods use an iterative scheme and require the availability of all the data in memory at once [38, 87, 91, 117, 118]. They might be able to achieve high performance but the amount of accumulated data, i.e. the number of input images, is necessarily limited by the computer memory capacity. A fast and low memory method requires a simpler combination. Note that burst denoising methods [19, 71] rely on interpolation and are not applicable in a super-resolution context.

An image can be computed pixel-by-pixel without any iterative scheme using classical kernel regression [117]. This is the simplest and most efficient kernel regression method as it only takes into account the spatial distance but not the photometric distance. However, this provides low quality results in areas containing textures or edges. Actually, the final image is blurry. As shown in [41], each output pixel value is a local weighted average of the samples, i.e. obtained by an underlying local linear filtering. But the equivalent filter depends on the data spatial repartition, which varies with the pixel position.

In this chapter we analyze how to fit irregularly sampled data by kernel regression. We show that the linear systems involved can be obtained by a data accumulation. Then, we focus on classical kernel regression and introduce the asymptotic equivalent filter, an approximation of the actual equivalent filter. Finally, we propose a fast and low memory image fusion method that is designed for a large number of images. The combination part, using classical kernel regression with a Gaussian kernel, is split into an accumulation part, where the images are processed sequentially, and an image post-processing part. The quality of the result is significantly improved in a sharpening step where the blur is successfully inverted by applying the inverse of the asymptotic equivalent filter. The registration method used is the one introduced in Chapter 7. The performance of the proposed algorithm is evaluated in Chapter 10.

This chapter is organized as follows: Section 9.2 presents the irregularly sampled data fitting by kernel regression. In Section 9.3 we focus on classical kernel regression and introduce the asymptotic equivalent filter. Finally, Section 9.4.1 describes our image formation algorithm.

9.2 Kernel Regression

In this section we present the fitting method for irregularly sampled data using a kernel regression [117]. This includes methods known as normalized convolution [64, 91] and bilateral filtering [20, 32, 55, 90, 124]. We focus on order $N_{\mathrm{KR}} \in \{0,1,2\}$ because higher order are more likely to fit noise and the computational cost increases rapidly with the order. First we introduce two notations. Denote by $d_{N_{\mathrm{KR}}}$ the number of coefficients of a two-dimensional polynomial of degree (at most) N_{KR}. We have

$$d_{N_{\mathrm{KR}}} = \frac{(N_{\mathrm{KR}}+1)(N_{\mathrm{KR}}+2)}{2} = \begin{cases} 1 & \text{if } N_{\mathrm{KR}} = 0 \\ 3 & \text{if } N_{\mathrm{KR}} = 1 \\ 6 & \text{if } N_{\mathrm{KR}} = 2. \end{cases} \tag{9.1}$$

We define the function $X : \mathbb{R}^2 \to \mathbb{R}^{d_{N_{\mathrm{KR}}}}$ by

$$X(x,y)^T = \begin{cases} 1 & \text{if } N_{\mathrm{KR}} = 0 \\ (1,x,y) & \text{if } N_{\mathrm{KR}} = 1 \\ (1,x,y,x^2,xy,y^2) & \text{if } N_{\mathrm{KR}} = 2. \end{cases} \tag{9.2}$$

9.2.1 Principle

Let $\mathcal{S}$ be a set of irregularly sampled data. Let $\mathbf{x}^0 = (x^0, y^0) \in \mathbb{R}^2$ be a pixel position for which the intensity value $I_{\mathbf{x}^0}$ has to be computed. Within a local neighborhood $B(\mathbf{x}^0, r)$ of radius $r > 0$ around $\mathbf{x}^0$, the intensity value at position $\mathbf{x} = (x + x^0, y + y^0)$ is approximated by a polynomial expansion

$$P_{\mathbf{x}^0}(\mathbf{x}, \mathbf{v}) = X(\mathbf{x} - \mathbf{x}^0)^T \mathbf{v} = \begin{cases} v_1 & \text{if } N_{\mathrm{KR}} = 0 \\ v_1 + v_2 x + v_3 y & \text{if } N_{\mathrm{KR}} = 1 \\ v_1 + v_2 x + v_3 y + v_4 x^2 + v_5 xy + v_6 y^2 & \text{if } N_{\mathrm{KR}} = 2. \end{cases} \tag{9.3}$$

An example of irregularly sampled data $\mathcal{S}$ and of neighborhood $B(\mathbf{x}^0, r)$ is shown in Figure 9.1.

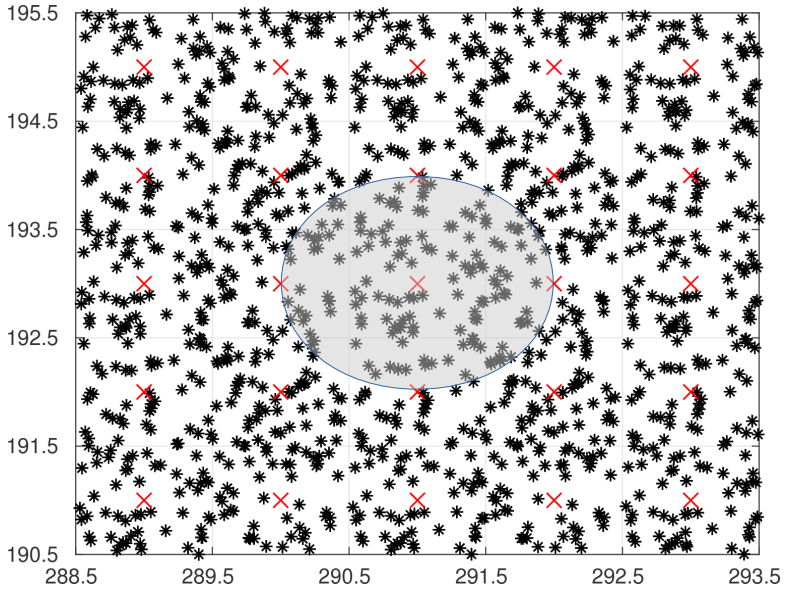


Figure 9.1: Example of irregularly sampled data $\mathcal{S}$ and of neighborhood $B(\mathbf{x}^0, r)$. The red crosses correspond to the integer grid. Here we have $\mathbf{x}^0 = (290, 193)$ and $r = 1$.

Denote by $(I_i, \mathbf{x}_i)_{1 \le i \le N_{\mathbf{x}^0}}$ the samples of $\mathcal{S}$ located in $B(\mathbf{x}^0, r)$ at position $\mathbf{x}_i$ and with intensity value I_i. To each location $\mathbf{x}_i$ is associated a weight $w_i \ge 0$, whose choice is discussed in Section 9.2.4. The parameter $\mathbf{v} = \mathbf{v}_{\mathbf{x}^0} = (v_i)^T_{1 \le i \le d_{N_{\mathrm{KR}}}} \in \mathbb{R}^{d_{N_{\mathrm{KR}}}}$, representing the polynomial coefficients, is chosen as the solution to a weighted linear regression on the intensities I_i, which amounts to minimizing the energy

$$\mathbf{u} \in \mathbb{R}^{d_{N_{\mathrm{KR}}}} \mapsto E_{\mathbf{x}^0}(\mathbf{u}) = \sum_{i=1}^{N_{\mathbf{x}^0}} w_i \left(I_i - P_{\mathbf{x}^0}(\mathbf{x}_i, \mathbf{u})\right)^2. \tag{9.4}$$

The intensity value $I_{\mathbf{x}^0}$ at location $\mathbf{x}^0$ is then given by

$$I_{\mathbf{x}^0} = P_{\mathbf{x}^0}(\mathbf{x}^0, \mathbf{v}) = v_1. \tag{9.5}$$

An image can be computed from $\mathcal{S}$ by applying the kernel regression method to pixels $\mathbf{x}^0$ in a regular grid.

9.2.2 Resolution of the Weighted Linear Regression Problem

We now detail how the polynomial coefficients $\mathbf{v}$, and consequently the intensity value $I_{\mathbf{x}^0}$, are obtained.

Notations. To the sample location $\mathbf{x}_i = (x^0 + x_i, y^0 + y_i)$ we associate the vector $X_i \in \mathbb{R}^{d_{N_{\mathrm{KR}}}}$ defined by

$$X_i^T = X(\mathbf{x}_i - \mathbf{x}^0)^T = \begin{cases} 1 & \text{if } N_{\mathrm{KR}} = 0 \\ (1, x_i, y_i) & \text{if } N_{\mathrm{KR}} = 1 \\ (1, x_i, y_i, x_i^2, x_i y_i, y_i^2) & \text{if } N_{\mathrm{KR}} = 2. \end{cases} \tag{9.6}$$

Denote by B the matrix defined by

$$B = \begin{pmatrix} X_1^T \\ \cdots \\ X_{N_{\mathbf{x}^0}} \end{pmatrix} = (X_i^T)_{1 \le i \le N_{\mathbf{x}^0}} \in \mathcal{M}_{N_{\mathbf{x}^0}, N_{\mathrm{KR}}}(\mathbb{R}) \tag{9.7}$$

and by W the diagonal matrix defined by

$$W = \begin{pmatrix} w_1 & & \\ & \ddots & \\ & & w_{N_{\mathbf{x}^0}} \end{pmatrix} \in \mathcal{M}_{N_{\mathbf{x}^0}}(\mathbb{R}) \tag{9.8}$$

Finally, set $\mathbf{I} = (I_i)_{1 \le i \le N_{\mathbf{x}^0}}^T \in \mathbb{R}^{d_{N_{\mathrm{KR}}}}$. Note that, to simplify, the dependencies on $\mathbf{x}^0$ of X_i, B, W and $\mathbf{I}$ are not specified.

Symmetric linear system. With these notations we can write

$$E_{\mathbf{x}^0}(\mathbf{u}) = \|W(B\mathbf{u} - \mathbf{I})\|_2^2 \tag{9.9}$$

so that the solution to the minimization problem $\mathbf{v} = \operatorname{argmin}_{\mathbf{u} \in \mathbb{R}^{d_{N_{\mathrm{KR}}}}} E_{\mathbf{x}^0}(\mathbf{u})$ is given by the normal equation

$$(B^T W B)\mathbf{v} = B^T W \mathbf{I}. \tag{9.10}$$

In other words, the parameter $\mathbf{v}$ satisfies the N_{KR}-by-N_{KR} symmetric linear system

$$A_{\mathbf{x}^0}\mathbf{v} = \mathbf{b}_{\mathbf{x}^0} \tag{9.11}$$

where

$$A_{\mathbf{x}^0} = (B^T W B) \quad \text{and} \quad \mathbf{b}_{\mathbf{x}^0} = B^T W \mathbf{I}. \tag{9.12}$$

Practical system resolution. In a non-degenerate case $A_{\mathbf{x}^0}$ is positive definite and the system in (9.11) admits a unique solution. For $N_{\mathrm{KR}} = 0$, the 1-by-1 system is solved by a simple division

$$\mathbf{v} = v_1 = \frac{\sum_{i=1}^{N_{\mathbf{x}^0}} w_i I_i}{\sum_{i=1}^{N_{\mathbf{x}^0}} w_i}. \tag{9.13}$$

For $N_{\mathrm{KR}} = 1$, the closed-form expression for the inverse of a 3-by-3 matrix is available. For $N_{\mathrm{KR}} = 2$, the Cholesky decomposition [44, p. 93] is used to solve the 6-by-6 system. In practice, to save computations, only the first component $v_1 = \left(A_{\mathbf{x}^0}^{-1}\mathbf{b}_{\mathbf{x}^0}\right)_1$ is computed.

9.2.3 Summative Expressions of the System Coefficients

Set $A_i = X_i X_i^T$. It is possible to rewrite $A_{\mathbf{x}^0}$ and $\mathbf{b}_{\mathbf{x}^0}$ as

$$A_{\mathbf{x}^0} = \sum_{i=1}^{N_{\mathbf{x}^0}} w_i A_i \tag{9.14}$$

and

$$\mathbf{b}_{\mathbf{x}^0} = \sum_{i=1}^{N_{\mathbf{x}^0}} w_i X_i I_i. \tag{9.15}$$

The corresponding expressions in terms of w_i, x_i, y_i and I_i for the three considered orders are the following.

For $N_{\mathbf{KR}} = 0$: We have $A_{\mathbf{x}^0} = \sum_{i=1}^{N_{\mathbf{x}^0}} w_i$ and $\mathbf{b}_{\mathbf{x}^0} = \sum_{i=1}^{N_{\mathbf{x}^0}} w_i I_i$.

For $N_{\mathbf{KR}} = 1$: We have

$$A_{\mathbf{x}^0} = \sum_{i=1}^{N_{\mathbf{x}^0}} w_i \begin{pmatrix} 1 & x_i & y_i \\ x_i & x_i^2 & x_i y_i \\ y_i & x_i y_i & y_i^2 \end{pmatrix} \tag{9.16}$$

and

$$\mathbf{b}_{\mathbf{x}^0} = \sum_{i=1}^{N_{\mathbf{x}^0}} w_i \begin{pmatrix} 1 \\ x_i \\ y_i \end{pmatrix} I_i. \tag{9.17}$$

For $N_{\mathbf{KR}} = 2$: We have

$$A_{\mathbf{x}^0} = \sum_{i=1}^{N_{\mathbf{x}^0}} w_i \begin{pmatrix} 1 & x_i & y_i & x_i^2 & x_i y_i & y_i^2 \\ x_i & x_i^2 & x_i y_i & x_i^3 & x_i^2 y_i & w_i x_i y_i^2 \\ y_i & x_i y_i & y_i^2 & x_i^2 y_i & x_i y_i^2 & y_i^3 \\ x_i^2 & x_i^3 & x_i^2 y_i & x_i^4 & x_i^3 y_i & x_i^2 y_i^2 \\ x_i y_i & x_i^2 y_i & x_i y_i^2 & x_i^3 y_i & x_i^2 y_i^2 & x_i y_i^3 \\ y_i^2 & x_i y_i^2 & y_i^3 & x_i^2 y_i^2 & x_i y_i^3 & y_i^4 \end{pmatrix} \tag{9.18}$$

and

$$\mathbf{b}_{\mathbf{x}^0} = \sum_{i=1}^{N_{\mathbf{x}^0}} \begin{pmatrix} I_i \\ x_i \\ y_i \\ x_i^2 \\ x_i y_i \\ y_i^2 \end{pmatrix} I_i. \tag{9.19}$$

Thanks to these summative expressions the contribution of a sample to the system coefficients can be computed by accumulation. This property is crucial to guarantee the low memory requirement of our image fusion algorithm described in Algorithm 9.1.

9.2.4 Choice of the Weights

In Section 9.2.1 we associated a weight w_i to the sample $(I_i, \mathbf{x}_i)$. This weight controls the relative contribution of $(I_i, \mathbf{x}_i)$, with respect to the other samples, in order to compute $\mathbf{v}$ and thus $I_{\mathbf{x}^0}$.

Among all the kernel regression methods there is a large variability in the choice of the weights but they rely on the common idea that they are built from a kernel function $w : \mathbb{R}^2 \to \mathbb{R}^+$. This function is assumed to reach its maximum at $(0,0)$ and to be a radial function. Depending on the way weights are built from the kernel, the methods can be categorized into two groups.

Classical kernel regression. In classical kernel regression methods [117] the weights only depend on the spatial repartition of the data around the current location $\mathbf{x}^0$. The idea is to have the highest contributions for the samples close to $\mathbf{x}^0$. The weights are given by

$$\forall 1 \leq i \leq N_{\mathbf{x}^0}, \quad w_i = w(\mathbf{x}_i - \mathbf{x}^0). \tag{9.20}$$

In general the kernel $w = w_{\sigma_s}$ is parametrized by a scaling parameter σ_s, which is linked to the neighborhood radius r. For instance in the Gaussian case we will take $r = 4\sigma_s$ in Section 9.4. A large scale allows for pixels far from $\mathbf{x}^0$ to contribute in a non-negligible way, which may be interesting for denoising. The choice of the scale is either made manually or automatically depending on the density of the data.

When the data density varies significantly, classical kernel regression methods can be slightly modified by taking into account the local density [91, 117]. A scale $\sigma_s(\mathbf{x}^0)$ is computed from the density around $\mathbf{x}^0$. The weights are then given by

$$\forall 1 \leq i \leq N_{\mathbf{x}^0}, \quad w_i = w_{\sigma_s(\mathbf{x}^0)}(\mathbf{x}_i - \mathbf{x}^0). \tag{9.21}$$

Classical kernel regressions methods are the simplest and most efficient methods but they do not perform well in areas containing textures or edges. When an image is built it appears blurry. In Section 9.3 we focus on classical kernel regression and show that it can be interpreted as local linear filtering of the data.

Data-adapted. In data-adapted kernel regression methods the weights also depend on the intensity values of the data.

- **Bilateral kernel regression:** In bilateral kernel regression methods [20, 32, 55, 90, 91, 124] the penalizations over the spatial distance and the intensities are separated. The weights are given by

$$\forall 1 \leq i \leq N_{\mathbf{x}^0}, \quad w_i = w_{\sigma_s}(\mathbf{x}_i - \mathbf{x}^0) w_{\sigma_r}(I_i - \tilde{I}(\mathbf{x}^0)) \tag{9.22}$$

  where $\sigma_r > 0$ is the scale for the radiometric information and $\tilde{I}(\mathbf{x}^0)$ is an estimate of $\tilde{I}_{\mathbf{x}^0}$. The scale $\sigma_r > 0$ can, for instance, be chosen depending on the data noise level.

  For $N_{\mathrm{KR}} = 0$ the results of the bilateral kernel regression is given by

$$I_{\mathbf{x}^0} = \frac{\sum_{i=1}^{N_{\mathbf{x}^0}} w_{\sigma_s}(\mathbf{x}_i - \mathbf{x}^0) w_{\sigma_r}(I_i - \tilde{I}(\mathbf{x}^0)) I_i}{\sum_{i=1}^{N_{\mathbf{x}^0}} w_{\sigma_s}(\mathbf{x}_i - \mathbf{x}^0) w_{\sigma_r}(I_i - \tilde{I}(\mathbf{x}^0))}. \tag{9.23}$$

  This is nothing but the bilateral filter [124]. For $N_{\mathrm{KR}} > 0$ it corresponds to high-order bilateral filters [116].

  In the denoising case, where regularly sampled data provide from an image, the estimate $\tilde{I}_{\mathbf{x}^0}$ is the noisy intensity value at $\mathbf{x}^0$. Otherwise, $\tilde{I}_{\mathbf{x}^0}$ has to be estimated using, for instance, another irregularly sampled data fitting method. In [91] and [117] the authors propose to use a classical kernel regression method. More precisely, in [91] they replace $\tilde{I}(\mathbf{x}^0)$ by the polynomial expansion $P_{\mathbf{x}^0}(\mathbf{x}^i, \mathbf{v})$ where $\mathbf{v}$ is the parameter obtained during the regression at location $\mathbf{x}^0$.

- **Steered kernel regression** The separation of the penalizations terms in (9.22) does not take into account the link between spatial and radiometric distances. For instance, it is clear that the intensity value varies more rapidly across an edge than along. In steered kernel regression methods [82, 91, 117, 119] the spatial penalization depends on the local structure of the data that is estimated from the intensity values. To simplify, the idea is to estimate the local gradient of the data and, accordingly, to associate to each location $\mathbf{x}_i$ a steering matrix $H_i^{\mathrm{steer}} \in \mathcal{M}_2(\mathbb{R})$. The weights are given by

$$\forall 1 \leq i \leq N_{\mathbf{x}^0}, \quad w_i = w_{\sigma_s}\left(H_i^{\mathrm{steer}}(\mathbf{x}_i - \mathbf{x}^0)\right). \tag{9.24}$$

  The steering matrices were built in order to give higher contribution to samples located far from $\mathbf{x}^0$ along the edges and less contribution across.

Data-adapted kernel regression methods provide better results than classical kernel regression. The data-adapted weights make these methods equivalent to a local nonlinear filtering of the data. The drawback is a significant increase of the computational cost. These methods rely on an pre-estimation of the output and an iterative scheme is usually required.

9.3 Classical Kernel Regression

In Section 9.2 we presented the general theory and computations of kernel regression methods. We now focus on the classical kernel regression, which is the most simple particular case where the weights only depend on the spatial repartition (see Section 9.2.4). Given a kernel w the weights are given by (9.20). The variation of the local density is not taken into account.

In Section 9.3.1 the pixel computation by classical kernel regression is interpreted as a local linear filtering process, which defines an equivalent filter and explains the blur introduction. The asymptotic equivalent filter, an approximation of the equivalent filter that does not depend on the pixel location, is introduced in Section 9.3.2. The asymptotic equivalent filter corresponding to a Gaussian kernel is presented in Section 9.3.3. The inverse of this filter will be used in our image fusion algorithm, described in Algorithm 9.1, to remove the blur and, consequently, to improve the results of classical kernel regression.

9.3.1 Local Linear Filtering and Equivalent Filter

As pointed out in [41, 117], classical kernel regression can be interpreted as a local linear filtering of the data. Indeed, we have

$$I_{\mathbf{x}^0} = v_1 = \left(A_{\mathbf{x}^0}^{-1}\mathbf{b}_{\mathbf{x}^0}\right)_1 = \sum_{j=1}^{d_{N_{\mathrm{KR}}}} (A_{\mathbf{x}^0}^{-1})_{[1,j]} (\mathbf{b}_{\mathbf{x}^0})_j \tag{9.25}$$

$$= \sum_{j=1}^{d_{N_{\mathrm{KR}}}} (A_{\mathbf{x}^0}^{-1})_{[1,j]} \sum_{i=1}^{N_{\mathbf{x}^0}} w_i (X_i)_j I_i \tag{9.26}$$

$$= \sum_{i=1}^{N_{\mathbf{x}^0}} w_i \left(\sum_{j=1}^{d_{N_{\mathrm{KR}}}} (A_{\mathbf{x}^0}^{-1})_{[1,j]} (X_i)_j \right) I_i. \tag{9.27}$$

This can be written as

$$I_{\mathbf{x}^0} = \sum_{i=1}^{N_{\mathbf{x}^0}} \tilde{w}(\mathbf{x}_i, \mathbf{x}_0) I_i \tag{9.28}$$

where

$$\tilde{w}(\mathbf{x}_i, \mathbf{x}^0) = w(\mathbf{x}_i - \mathbf{x}^0) \left(\sum_{j=1}^{d_{N_{\mathrm{KR}}}} (A_{\mathbf{x}^0}^{-1})_{[1,j]} (X(\mathbf{x}_i - \mathbf{x}^0))_j \right). \tag{9.29}$$

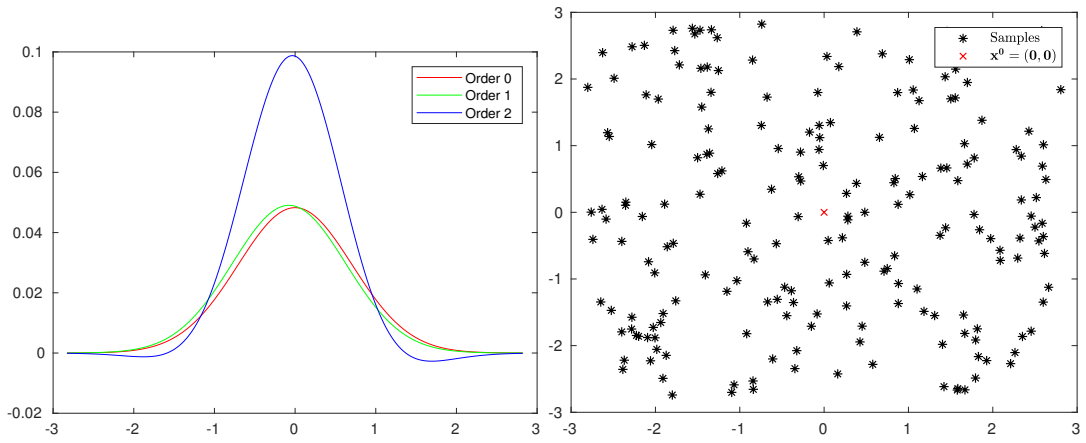
Thus, the computed intensity value $I_{\mathbf{x}^0}$ is the weighted average of the intensities $(I_i)_{1\le i\le N_{\mathbf{x}^0}}$ where the weights $\tilde{w}(\mathbf{x}_i, \mathbf{x}^0)$ depends on the spatial repartition of the data in the neighborhood $B(\mathbf{x}^0, r)$, i.e., on the locations $(\mathbf{x}_i)_{1\le i\le N_{\mathbf{x}^0}}$. Note that, similarly, data-adapted kernel regression can be interpreted as a local nonlinear filtering since the corresponding weights also depend on the intensities $(I_i)_{1\le i\le N_{\mathbf{x}^0}}$.

Equivalent filter. The local linear filtering defines equivalent filters (or kernel). The equivalent filter corresponding to the location $\mathbf{x}^0$ is given by

$$\mathbf{x} \mapsto \tilde{w}(\mathbf{x}, \mathbf{x}^0). \tag{9.30}$$

Note that classical kernel regression reproduces constants so that we have $\sum_{i=1}^{N_{\mathbf{x}^0}} \tilde{w}(\mathbf{x}_i, \mathbf{x}_0) = 1$. The equivalent filter $\tilde{w}(\mathbf{x}, \mathbf{x}^0)$, however, is not normalized, i.e., its integral may be different from one.

An example of equivalent filter is shown in Figure 9.2. The equivalent filters for $N_{\mathrm{KR}} = 0$ and $N_{\mathrm{KR}} = 1$ are similar. More generally, classical kernel regression with orders $N_{\mathrm{KR}} = 2q$



(a) Equivalent filter (slice $y = 0$) (b) Spatial data repartition

Figure 9.2: Example of equivalent filter for a Gaussian kernel of standard deviation $\sigma_s = \frac{\sqrt{2}}{2}$ and $r = 4\sigma_s$. For the data whose spatial repartition is represented in (b), the equivalent filters at location $\mathbf{x}^0 = (0,0)$ for $N_{KR} \in \{0,1,2\}$ are computed. The curves corresponding to the slice $y = 0$ are shown in (a). The equivalent filters for $N_{KR} = 0$ and $N_{KR} = 1$ are similar.

and $N_{KR} = 2q+1$ lead asympotically (see Section 9.3.2 for a proper definition) to the same results [117]. This property is proved in Section 9.3.3 for the Gaussian kernel.

In general the equivalent filter is a low-pass approximation (more or less tight) of the Dirac function $\delta_{\mathbf{x}^0}$. This explains why an image computed by classical kernel regression seems blurry: the equivalent filter has attenuated the high-frequencies.

9.3.2 Asymptotic Equivalent Filter

The blur introduced by a linear filter can be inverted by applying the inverse filter (provided there is one). The problem here is that the equivalent filter is a local linear filter, which depends on the data spatial repartition. In the following we show that the equivalent filter can be approximated "asymptotically" by a linear filter, that does not depend on the pixel location.

Additional hypothesis on the kernel. To guarantee that the quantities introduced below are well-defined, an additional hypothesis is made on the kernel w. We assume that for all non-negative integers m, n such that $m+n \leq 2N_{KR}$,

$$\int_{\mathbb{R}^2} w(x,y)|x|^m|y|^n dxdy < +\infty. \tag{9.31}$$

Asymptotic formulation. Consider the ideal case where the data in $\mathcal{S}$ correspond to the sampling, with a uniform spatial distribution, of a bounded function $I : \mathbb{R}^2 \to \mathbb{R}$. The energy $E_{\mathbf{x}^0}$, introduced in (9.4) and defining the weighted linear regression problem, can be written as

$$E_{\mathbf{x}^0}(\mathbf{u}) = \sum_{i=1}^{N_{\mathbf{x}^0}} w(\mathbf{x}_i - \mathbf{x}^0) \left(I(\mathbf{x}_i) - P_{\mathbf{x}^0}(\mathbf{x}_i, \mathbf{u})\right)^2. \tag{9.32}$$

As the number of local samples $N_{\mathbf{x}^0}$ and the neighborhood radius r go to infinity, thanks to Monte-Carlo integration [51], we have

$$\frac{1}{N_{\mathbf{x}^0}} E_{\mathbf{x}^0}(\mathbf{u}) \longrightarrow \int_{\mathbb{R}^2} w(\mathbf{x} - \mathbf{x}^0) \left(I(\mathbf{x}) - P_{\mathbf{x}^0}(\mathbf{x}, \mathbf{u})\right)^2 d\mathbf{x} \doteq \mathcal{E}_{\mathbf{x}^0}(\mathbf{u}). \tag{9.33}$$

The energy $\mathcal{E}_{\mathbf{x}^0}$ is called the asymptotic energy. As for the discrete case, to the minimizer $\mathbf{v}$ of $\mathcal{E}_{\mathbf{x}^0}$ (provided there is one) is associated the intensity value $I_{\mathbf{x}^0} = v_1$.

Minimization of the asymptotic energy. Using similar computations as for the discrete case, a continuous version of the normal equation (9.10) is obtained. The parameter $\mathbf{v}$ is a critical point of $\mathcal{E}_{\mathbf{x}^0}$ if and only if

$$\left(\int_{\mathbb{R}^2} w(\mathbf{x}-\mathbf{x}^0)(XX^T)(\mathbf{x}-\mathbf{x}^0)d\mathbf{x} \right) \mathbf{v} = \int_{\mathbb{R}^2} w(\mathbf{x}-\mathbf{x}^0)X(\mathbf{x}-\mathbf{x}^0)I(\mathbf{x})d\mathbf{x}. \tag{9.34}$$

After a change of variable this can be written as the linear system

$$A\mathbf{v} = \mathbf{b}(\mathbf{x}^0) \tag{9.35}$$

where

$$A = A_{w,N_{\mathrm{KR}}} = \int_{\mathbb{R}^2} w(\mathbf{x})(XX^T)(\mathbf{x})d\mathbf{x} \in \mathcal{M}_{d_{N_{\mathrm{KR}}}}(\mathbb{R}) \tag{9.36}$$

is a positive semi-definite matrix that does not depend on $\mathbf{x}^0$ and, denoting by $\star$ the cross-correlation operator,

$$\mathbf{b}(\mathbf{x}^0) = (wX \star I)(\mathbf{x}^0). \tag{9.37}$$

In a non-degenerate case (i.e. for an appropriate choice of kernel w), A is positive definite and the asymptotic energy has a unique minimum $\mathbf{v}$, which is defined by (9.35).

Linear filtering. The intensity value $I_{\mathbf{x}^0}$ is given by

$$I_{\mathbf{x}^0} = \int_{\mathbb{R}^2} \left(A^{-1}X(\mathbf{x}-\mathbf{x}^0)\right)_1 w(\mathbf{x}-\mathbf{x}^0)I(\mathbf{x})d\mathbf{x}. \tag{9.38}$$

Set $X^* = X(-\cdot)$. We define the asymptotic equivalent filter $\tilde{w}_\infty : \mathbb{R}^2 \to \mathbb{R}$ by

$$\tilde{w}_\infty = \left(A^{-1}X^*\right)_1 w. \tag{9.39}$$

Then, the intensity value $I_{\mathbf{x}^0}$ given by classical kernel regression in the asymptotic case is expressed as the linear filtering of the underlying image I by $\tilde{w}_\infty$,

$$I_{\mathbf{x}^0} = (\tilde{w}_\infty * I)(\mathbf{x}^0). \tag{9.40}$$

Global approximation of the equivalent kernel. Contrarily to the equivalent filter $\tilde{w}$, whose expression is given in (9.29), the asymptotic equivalent filter $\tilde{w}_\infty$ does not depend on the location $\mathbf{x}^0$ nor on the spatial data repartition.

A and $\mathbf{b}(\mathbf{x}^0)$ can be obtained respectively from the discrete quantities $A_{\mathbf{x}^0}$ and $\mathbf{x}^0$ (see (9.14) and (9.15)) thanks to Monte-Carlo integration. Thus, $\tilde{w}_\infty$, which was at first place obtained from the minimizer of the approximate energy $\mathcal{E}_{\mathbf{x}^0}$, is also an approximation of the equivalent filter $\tilde{w}$.

Interpretation. To summarize, an image computed by classical kernel regression can be asymptotically interpreted as the convolution between the ideal unknown target image I and a filter $\tilde{w}_\infty$. The image may seem blurry since $\tilde{w}_\infty$ attenuates the high-frequencies (see Gaussian case in the next section). This justifies, in Algorithm 9.1, the application of an enhancement filter reverting the blur.

9.3.3 Gaussian Case

Denote by w_{σ_s} the two-dimensional Gaussian kernel of scale $\sigma_s > 0$, i.e.,

$$w_{\sigma_s}(\mathbf{x}) = \frac{1}{2\pi\sigma_s^2} \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma_s^2}\right). \tag{9.41}$$

The asymptotic equivalent filter depends on the kernel and on the order N_{KR}. We now present the equivalent filter $\tilde{w}_{\sigma_s,\infty}$ corresponding to w_{σ_s} for $N_{\mathrm{KR}} \in \{0,1,2\}$.

Expression of A. The matrix A, defined in (9.36), is composed of elements of the form

$$W_{\sigma_s,m,n} = \int_{\mathbb{R}^2} x^m y^n w_{\sigma_s}(x,y)dxdy = \underbrace{\left(\sqrt{2\pi\sigma_s^2}\int_{\mathbb{R}} w_{\sigma_s}(x,0)x^m dx\right)}_{\doteq W_{\sigma_s,m}} \underbrace{\left(\sqrt{2\pi\sigma_s^2}\int_{\mathbb{R}} w_{\sigma_s}(0,y)y^n dy\right)}_{\doteq W_{\sigma_s,n}} \tag{9.42}$$

for n,m non-negative integers. By symmetry, for n odd we have $W_{\sigma_s,n} = 0$. For n even, we use the following formula

$$\int_0^\infty x^n e^{-ax^2} dx = \frac{\Gamma(\frac{n+1}{2})}{2a^{\frac{n+1}{2}}}. \tag{9.43}$$

where $a > 0$ and $\Gamma : z \in \mathbb{C} \mapsto \int_0^\infty x^{z-1}e^{-x}dx$ is the Gamma function. With $a = \frac{1}{2\sigma_s^2}$, we have for n even,

$$W_{\sigma_s,n} = \frac{1}{\sqrt{2\pi\sigma_s^2}}(2\sigma_s^2)^{\frac{n+1}{2}}\Gamma\left(\frac{n+1}{2}\right). \tag{9.44}$$

Finally, we have for $n \in \{0,1,2,3,4\}$,

$$\begin{cases} W_{\sigma_s,0} = & 1 \\ W_{\sigma_s,2} = & \sigma_s^2 \\ W_{\sigma_s,4} = & 3\sigma_s^4 \\ W_{\sigma_s,1} = W_{\sigma_s,3} = & 0. \end{cases} \tag{9.45}$$

Note that we used the well-known formulas $\Gamma(\frac{1}{2}) = \sqrt{\pi}$ and $\Gamma(x+1) = x\Gamma(x)$ (for $x > 0$). The explicit expressions of A and A^{-1} for $N_{\mathrm{KR}} \in \{0,1,2\}$ are the following.

For $N_{\mathrm{KR}} = 0$: We have $A = 1$ and $A^{-1} = 1$.

For $N_{\mathrm{KR}} = 1$: A is a diagonal matrix given by

$$A = \begin{pmatrix} W_{\sigma_s,0} & 0 & 0 \\ 0 & W_{\sigma_s,2}W_{\sigma_s,0} & 0 \\ 0 & 0 & W_{\sigma_s,2}W_{\sigma_s,0} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sigma_s^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix} \tag{9.46}$$

and its inverse A^{-1} is given by

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sigma_s^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_s^2} \end{pmatrix}. \tag{9.47}$$

For $N_{\mathbf{KR}} = 2$: A is given by

$$A = \begin{pmatrix} W^2_{\sigma_s,0} & 0 & 0 & W_{\sigma_s,0}W_{\sigma_s,2} & 0 & W_{\sigma_s,0}W_{\sigma_s,2} \\ 0 & W_{\sigma_s,0}W_{\sigma_s,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & W_{\sigma_s,0}W_{\sigma_s,2} & 0 & 0 & 0 \\ W_{\sigma_s,0}W_{\sigma_s,2} & 0 & 0 & W_{\sigma_s,0}W_{\sigma_s,4} & 0 & W^2_{\sigma_s,2} \\ 0 & 0 & 0 & 0 & W^2_{\sigma_s,2} & 0 \\ W_{\sigma_s,0}W_{\sigma_s,2} & 0 & 0 & W^2_{\sigma_s,2} & 0 & W_{\sigma_s,0}W_{\sigma_s,4} \end{pmatrix} \tag{9.48}$$

$$= \begin{pmatrix} 1 & 0 & 0 & \sigma_s^2 & 0 & \sigma_s^2 \\ 0 & \sigma_s^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_s^2 & 0 & 0 & 0 \\ \sigma_s^2 & 0 & 0 & 3\sigma_s^4 & 0 & \sigma_s^4 \\ 0 & 0 & 0 & 0 & \sigma_s^4 & 0 \\ \sigma_s^2 & 0 & 0 & \sigma_s^4 & 0 & 3\sigma_s^4 \end{pmatrix} \tag{9.49}$$

and its inverse A^{-1} is given by

$$A^{-1} = \begin{pmatrix} 2 & 0 & 0 & -\frac{1}{2\sigma_s^2} & 0 & -\frac{1}{2\sigma_s^2} \\ 0 & \frac{1}{\sigma_s^2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sigma_s^2} & 0 & 0 & 0 \\ -\frac{1}{2\sigma_s^2} & 0 & 0 & \frac{1}{2\sigma_s^4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sigma_s^4} & 0 \\ -\frac{1}{2\sigma_s^2} & 0 & 0 & 0 & 0 & \frac{1}{2\sigma_s^4} \end{pmatrix}. \tag{9.50}$$

Expression of $\tilde{w}_{\sigma_s,\infty}$. Combining the above expressions of A^{-1} and (9.39), we obtain an explicit formula for the asymptotic equivalent filter $\tilde{w}_{\sigma_s,\infty}$.

For $N_{\mathbf{KR}} \in \{0,1\}$: As discussed in Section 9.3.1, order 0 and 1 share the same asymptotic equivalent filter and are asymptotically equivalent. The asymptotic equivalent filter coincides with the Gaussian kernel w_{σ_s}, i.e.,

$$\tilde{w}_{\sigma_s,\infty}(\mathbf{x}) = w_{\sigma_s}(\mathbf{x}). \tag{9.51}$$

Its Fourier transform is given by

$$\mathcal{F}(\tilde{w}_{\sigma_s,\infty})(\mathbf{x}) = \exp\left(-\sigma_s^2 \frac{\|\mathbf{x}\|^2}{2}\right). \tag{9.52}$$

For $N_{\mathbf{KR}} = 2$: The asymptotic equivalent filter is given by

$$\tilde{w}_{\sigma_s,\infty}(\mathbf{x}) = w_{\sigma_s}(\mathbf{x})\left(2 - \frac{\|\mathbf{x}\|^2}{2\sigma_s^2}\right) \tag{9.53}$$

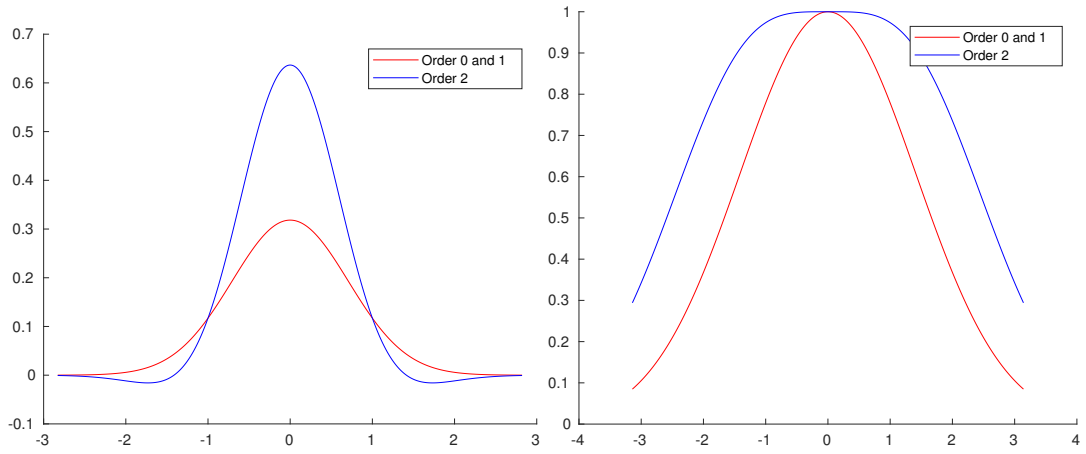
and its Fourier transform is

$$\mathcal{F}(\tilde{w}_{\sigma_s,\infty})(\mathbf{x}) = \left(1 + \frac{\sigma_s^2}{2}\|\mathbf{x}\|^2\right)\exp\left(-\sigma_s^2 \frac{\|\mathbf{x}\|^2}{2}\right). \tag{9.54}$$

Note that we can write

$$\tilde{w}_{\sigma_s,\infty} = w_{\sigma_s} - \frac{\sigma_s^2}{2}\Delta(w_{\sigma_s}) \tag{9.55}$$

where Δ denotes the Laplacian operator.



(a) $x \mapsto \tilde{w}_{\sigma_s,\infty}(x,0)$ (b) $x \mapsto \mathcal{F}(\tilde{w}_{\sigma_s,\infty})(x,0)$

Figure 9.3: Asymptotic equivalent filter for the Gaussian kernel ($\sigma_s = 0.7$). In both spatial (a) and Fourier domains (b), the functions decrease as $\|\mathbf{x}\|$ increases. In particular, the linear filtering by $\tilde{w}_{\sigma_s,\infty}$ attenuates the high-frequency content. This attenuation is less important for $N_{KR} = 2$. The shape of $\tilde{w}_{\sigma_s,\infty}$ is similar to the shape of the equivalent filter $\tilde{w}$ shown in Figure 9.2. The asymptotic filters have an integral of one. For $N_{KR} = 2$ it only seems larger because the filter takes negative values.

In both cases the asymptotic equivalent filters are radial functions with positive Fourier transforms. The asymptotic equivalent filters for $N_{KR} \in \{0,1,2\}$, and the corresponding Fourier transforms, are shown in Figure 9.3. In both spatial and Fourier domains, the functions decrease as $\|\mathbf{x}\|$ increases. In particular, the linear filtering by $\tilde{w}_{\sigma_s,\infty}$ attenuates the high-frequency content. This attenuation is less important for $N_{KR} = 2$. The shape of $\tilde{w}_{\sigma_s,\infty}$ is similar to the shape of the equivalent filter $\tilde{w}$ shown in Figure 9.2. The asymptotic filters have an integral of one. For $N_{KR} = 2$ it only seems larger because the filter takes negative values.

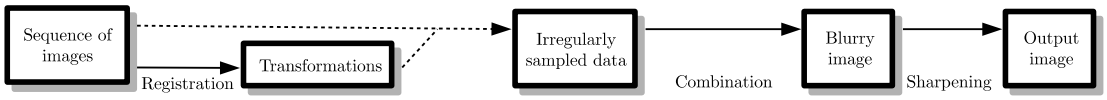
9.4 Fast and Low Memory Image Fusion

In this section we propose a fast and low memory image fusion method that is designed for a large number of images. The combination part, using classical kernel regression with Gaussian kernel, is split into an accumulation part and an image computation part. Using the particular structure of the linear systems involved for computing the pixel values (see Section 9.2.3), the input images are processed sequentially during the accumulation. Then, an image is computed by resolving the small systems at each pixel locations. The counterpart of this efficient image computation, which do not require an iterative scheme, is the blurry aspect of the image. The quality of the result is significantly improved in the sharpening step. The blur is inverted by applying the inverse of the asymptotic equivalent filter, introduced in Section 9.3.3.

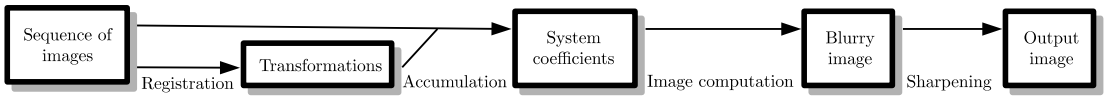
Our algorithm is detailed in Section 9.4.1. The low memory requirement of the method is discussed in Section 9.4.2. The choice of the optimal parameters and the evaluation of the performance of our algorithm are the subjects of Chapter 10.

9.4.1 Proposed Algorithm

Given a sequence $(I^j)_{1 \leq j \leq N_{im}}$ of N_{im} images of common size $M \times N$, a zoom factor $\lambda > 0$, an order $N_{KR} \in \{0,1,2\}$ and a scale $\sigma_s > 0$, our image fusion method builds an image I of size $[\lambda M] \times [\lambda N]$. Our algorithm is presented in Algorithm 9.1 and its main steps are summarized in



(a) Memory greedy algorithm



(b) Our algorithm

Figure 9.4: Main steps of our image fusion method. (a) and (b) are theoretically equivalent. The combination from the irregularly sampled data is replaced by the accumulation step, where the images are sequentially processed to get the system coefficients, and the image computation step, where the systems are solved.

Figure 9.4(b).

Registration. The first image I^1 is arbitrarily chosen as the reference image for the registration part, i.e., the irregularly sampled data are expressed in the system of coordinates defined by I^1. The transformation φ^1 is set to the identity. The homographic transformation φ^j between I^1 and I^j, so that $I^1 \simeq I^j(\varphi^j(\cdot))$, is estimated using the modified inverse compositional algorithm with Lorentzian error function (see Algorithm 7.4 in Chapter 7).

Accumulation. The system coefficents in $A_{\mathbf{x}^0}$ and $\mathbf{b}_{\mathbf{x}^0}$ for each output pixel $\mathbf{x}^0 \in \Omega_{[\lambda M],[\lambda N]}$ are computed by accumulation. First, they are initialized to zero i.e. $A_{\mathbf{x}^0} = 0_{\mathcal{M}_{d_{N_{\mathrm{KR}}}}(\mathbb{R})}$ and $\mathbf{b}_{\mathbf{x}^0} = 0_{\mathbb{R}^{d_{N_{\mathrm{KR}}}}}$. Then, the images are processed sequentially. For each pixel $(k,l) \in \Omega_{M,N}$, the corresponding location $\mathbf{x} = \lambda(\varphi^j)^{-1}(k,l)$ in the zoomed reference system is computed. The sample $(\mathbf{x}, I^j_{k,l})$ contributes to the system coefficients of the output pixel $\mathbf{x}^0 \in B(\mathbf{x}, 4\sigma_s)$ as written in (9.14) and (9.15).

Image computation. For each output pixel $\mathbf{x}^0 \in \Omega_{[\lambda M],[\lambda N]}$, the intensity value $I_{\mathbf{x}^0}$ is computed from $A_{\mathbf{x}^0}$ and $\mathbf{b}_{\mathbf{x}^0}$ as described in Section 9.2.2. It defines an image I of size $[\lambda M] \times [\lambda N]$, which is blurry.

Sharpening. To remove the blur, the inverse of the equivalent asymptotic filter $\tilde{w}_{\sigma_s,\infty}$ (see Section 9.3.3) is applied to I using the DCT convolution method [45, 77]. Note that the DCT convolution method only applies when the filter is symmetrical (and real-valued). Thanks to the symmetrical boundary handling, it provides better results than the filtering method presented in Chapter 4, which implicitly uses a periodic extension.

Precisions.

- Our method naturally extends to color images by processing the channels independently (except for the registration part that is shared).
- The evaluation of the Gaussian weights is costly. Therefore they are tabulated.
- As discussed in Section 9.4.2, in practice there is no need to compute the whole matrix $A_{\mathbf{x}^0}$ because of redundancies. This saves both memory and computations.
- Contrast change may deteriorate the algorithm performance (during the registration and the fitting). This can be handled by equalizing the input image contrasts, for instance using the Midway Image Equalization algorithm [28, 50]. In Chapter 10 we perform a simple mean equalization.

Algorithm 9.1: Image fusion algorithm using classical kernel regression.

Input : A sequence $(I^j)_{1\leq j\leq N_{\text{im}}}$ of N_{im} images of size $M\times N$, a zoom factor $\lambda>0$, an order $N_{\text{KR}}\in\{0,1,2\}$ and a scale $\sigma_s>0$

Output: A fused image I of size $[\lambda M]\times[\lambda N]$

// Registration

Set φ^1 to the identity transformation

for $j\in\{2,\dots,N_{im}\}$ **do**

  Compute the estimated homographic transformation φ^j between I^1 and I^j using Algorithm 7.4 with Lorentzian error function

// Accumulation

for $\mathbf{x}^0\in\Omega_{[\lambda M],[\lambda N]}$ **do**

  Initialize $A_{\mathbf{x}^0}$ and $\mathbf{b}_{\mathbf{x}^0}$ to zero

for $j\in\{1,\dots,N_{im}\}$ **do**

  for $(k,l)\in\Omega_{M,N}$ **do**

    Compute $\mathbf{x}=\lambda(\varphi^j)^{-1}(k,l)$

    for $\mathbf{x}^0\in\Omega_{[\lambda M],[\lambda N]}\cap B(\mathbf{x},4\sigma_s)$ **do**

      Compute the Gaussian weight $w=w_{\sigma_s}(\mathbf{x}-\mathbf{x}^0)$ using (9.41)

      Compute $X=X(\mathbf{x}-\mathbf{x}^0)$ using (9.2)

      Update $A_{\mathbf{x}^0}\leftarrow A_{\mathbf{x}^0}+wXX^T$

      Update $\mathbf{b}_{\mathbf{x}^0}\leftarrow\mathbf{b}_{\mathbf{x}^0}+wXI^j_{k,l}$

// Image computation

for $\mathbf{x}^0\in\Omega_{[\lambda M],[\lambda N]}$ **do**

  Compute $I_{\mathbf{x}^0}$ from $A_{\mathbf{x}^0}$ and $\mathbf{b}_{\mathbf{x}^0}$ as described in Section 9.2.2

// Sharpening

Apply the inverse of the equivalent asymptotic filter $\tilde{w}_{\sigma_s,\infty}$ (see Section 9.3.3) to I using the DCT convolution

Remarks.

- A zoom factor $\lambda>1$ corresponds to a super-resolution context. For $\lambda<1$, it can be seen as a down-sampling.
- The scale σ_s is relative to the zoomed system of coordinates. In the reference system of coordinates the corresponding scale is $\lambda\sigma_s$.
- Assume an additional reference image is provided to the algorithm. A variant of the algorithm consists in using this image as the reference (instead of I^1) during the registration and not during the data fitting part. The idea is to remove the asymmetry between the images since, in Algorithm 9.1, I^1 plays a particular role. Indeed, the samples of I^1 always have the maximal weight during the fitting. This variant is used in the experimental evaluation of our algorithm in Chapter 10.
- Our method does not perform deblurring (as in [33] for instance). The sharpening step only corrects the blur introduced by the classical kernel regression.

9.4.2 A Low Memory Requirement

Our image fusion algorithm is designed for a large number of images N_{im}. As the images are processed sequentially, the memory requirement does not depend on the number of input images

N_{im} but on the size of the output image. The majority of the memory requirement comes from the storage of the system coefficients. In theory, the storage of $A_{\mathbf{x}^0}$ and $\mathbf{b}_{\mathbf{x}^0}$ requires $d_{N_{\text{KR}}}^2 + d_{N_{\text{KR}}}$ (double-precision floating point) numbers. Thanks to the redundancies in $A_{\mathbf{x}^0}$ (see Section 9.2.3), it comes back down to the storage of

$$N_{\text{mem}} = \begin{cases} 2 & \text{if } N_{\text{KR}} = 0 \\ 9 & \text{if } N_{\text{KR}} = 1 \\ 21 & \text{if } N_{\text{KR}} = 2 \end{cases} \tag{9.56}$$

numbers. Finally, the overall storage of the system coefficient represents $N_{\text{mem}} [\lambda M] [\lambda N]$ numbers.

Theoretically the accumulation and image computation steps are equivalent to a single combination step taking as input the irregularly sampled data. The structure of the resulting memory greedy algorithm is shown in Fig. 9.4(a). The knowledge of the irregularly sampled data requires to store three numbers per sample (two for the location and one for the intensity). The overall storage represents $3N_{\text{im}}MN$ numbers. The storage of the system coefficients is smaller than the storage of the data as soon as the number of images N_{im} becomes larger than $\frac{\lambda^2 N_{\text{mem}}}{3}$. For $\lambda = 1$ and $N_{\text{KR}} \in \{0,1,2\}$ it corresponds respectively to 1, 3 and 7 images. For $\lambda = 2$ the values are multiplied by 4.

Note that the number of updates per sample depends on the scale σ_{s}. For instance, assume the value $\sigma_{\text{s}} = 1/\sqrt{2}$ is used. Then, each sample contributes to at most 20 surrounding pixels, which represents $20N_{\text{mem}}$ updates.

9.5 Conclusion

In this chapter we presented irregularly sampled data fitting by kernel regression. The intensity value of the data is locally approximated by a polynomial expansion, whose coefficients are obtained by solving a weighted linear regression (with weights built from a kernel function). We showed that summative expressions for the system coefficients are available.

Classical kernel regression is the most simple and efficient case where the weights only depend on the data spatial repartition. As it is equivalent to a local linear filtering, classical kernel regression introduces blur. We introduced the asymptotic equivalent filter, an approximation of the actual equivalent filter, that does not depend on the spatial repartition. We obtained the expression of this filter in the Gaussian case.

Based on these results we proposed a fast and low memory image fusion algorithm that is designed for a large number of images. The registration method used is the one introduced in Chapter 7. Using the particular structure of the system coefficients, the combination part, using classical kernel regression with a Gaussian kernel, is split into an accumulation part, where the images are processed sequentially, and an image computation part. The blur, introduced by classical kernel regression, is inverted by applying the inverse of the asymptotic equivalent filter.

The choice of the optimal parameters and the evaluation of the performance of our algorithm are the subjects of Chapter 10. The combination part of our image fusion method is adapted to mosaicked images in Chapter 11.

Chapter 10

Experimental Evaluation of our Image Fusion Algorithm

Abstract

In this chapter we evaluate experimentally the performance of our proposed image fusion algorithm on synthetic and real data, as it has been described in Algorithm 9.1. Our analysis of the synthetic case shows that the sharpening step is crucial to obtain a blur-free denoised image. We also figure out the best configuration for the order and the scale, depending on the sub-sampling factor. Assuming a uniform repartition of the samples, we conclude that it is better to use the order 0 when no super-resolution is required, and the order 2 for super-resolution. We find that a Gaussian kernel scale of about 0.7 is the best choice. We show that for a large amount of data (synthetic or real) our image fusion algorithm provides similar results as slower and memory greedy methods. The residual noise on both synthetic and real examples decreases as expected and our algorithm can perform super-resolution. From the experiments on real data we see that the performance of image fusion methods is limited by uncontrolled processes (demosaicking, JPEG compression, 8-bit quantization). This is why we propose in Chapter 11 an image formation algorithm taking RAW images as input.

Contents

10.1 Introduction

In this experimental chapter we evaluate the performance of our image fusion algorithm, described in Algorithm 9.1, on synthetic and real data and estimate the best configuration for the order and the scale, depending on the sub-sampling factor. In Chapter 9 we saw that order 0 and 1 are asymptotically equivalent. Therefore in the following we consider $N_{\mathrm{KR}} \in \{0,2\}$.

Our image fusion algorithm is compared to similar methods where the data fitting part (by classical kernel regression) is respectively replaced by the ACT method [38] and by a burst denoising method. We show that for a large amount of data (synthetic or real) our image fusion algorithm provides similar results as both of these slower and memory greedy methods. The residual noise decay proves to be as theoretically expected, and our algorithm also allows super-resolution.

The experiments were made using an Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz (on a single thread) and a C language implementation. We used the *p+s-spline11* interpolation method, introduced in Chapter 6, for the burst denoising. Using a fine-tuned interpolation method makes the burst denoising slower but leads to significantly better results. First, the two error measurements $\mathcal{E}_{\mathrm{ref}}$ and $\mathcal{E}_{\mathrm{set}}$, which are used to evaluate the algorithm performances, are introduced in Section 10.2. Section 10.3 presents the experiments on synthetic data. The results on real data are analyzed in Section 10.4.

10.2 Evaluation of the Error

Let $(I_i)_{1\leq i\leq n}$ be a sequence of images and consider an image fusion method.

Distance to the reference. Assume that the reference target image I_{ref} (of size $M\times N$) is known. Denote by I the image built from the sequence $(I_i)_{1\leq i\leq n}$. The error $\mathcal{E}_{\mathrm{ref}}$ is defined as the root mean square error (RMSE) of the reconstructed image I. It is given by

$$\mathcal{E}_{\mathrm{ref}}^2 = \frac{1}{MN}\sum_{(k,l)\in\Omega_{M,N}} (I-I_{\mathrm{ref}})_{k,l}^2. \tag{10.1}$$

Distance between reconstructed image. When no reference image is available (for instance for real data), it is not possible to define $\mathcal{E}_{\mathrm{ref}}$. We propose to consider the distance between reconstructed images from two separated sequences. Let $(I'_i)_{1\leq i\leq n}$ be another sequence. Denote by I and I' the two reconstructed images, which are assumed to be expressed in the same system of coordinates. The error $\mathcal{E}_{\mathrm{set}}$ is defined as the root mean square difference (RMSD) between I and I'. It is given by

$$\mathcal{E}_{\mathrm{set}}^2 = \frac{1}{MN}\sum_{(k,l)\in\Omega_{M,N}} (I'-I)_{k,l}^2. \tag{10.2}$$

For instance, the two sequences can be obtained by splitting a sequence $(I_i)_{1\leq i\leq 2n}$ in two.
Remark: In practice, to discard boundary artefacts, a crop of 20 pixels is done before computing $\mathcal{E}_{\mathrm{ref}}$ and $\mathcal{E}_{\mathrm{set}}$.

Link with the residual noise. The ideal denoising case corresponds to the case where each intensity value is computed as the average of n_{sample} noisy samples. Denote by σ_{n} the noise level. For the pixel $(k,l)\in\Omega_{M,N}$ denote by $J_s^{k,l}$ the samples. We assume that we can write $J_s^{k,l} = (I_{\mathrm{ref}})_{k,l} + \varepsilon_s^{k,l}$ where the $(\varepsilon_s^{k,l})$, which represent the noise, are independent identically distributed with $\mathbb{E}(\varepsilon_s^{k,l}) = 0$ and $\mathrm{Var}(\varepsilon_s^{k,l}) = \sigma_{\mathrm{n}}^2$. Then, we have

$$I_{k,l} = \frac{1}{n_{\mathrm{sample}}}\sum_{s=1}^{n_{\mathrm{sample}}} J_s^{k,l} = (I_{\mathrm{ref}})_{k,l} + \frac{1}{n_{\mathrm{sample}}}\sum_{s=1}^{n_{\mathrm{sample}}} \varepsilon_s^{k,l}. \tag{10.3}$$

The residual noise $\varepsilon_{k,l}$ at location (k,l) is then given by

$$\varepsilon_{k,l} = \frac{1}{n_{\text{sample}}} \sum_{s=1}^{n_{\text{sample}}} \varepsilon_s^{k,l} \tag{10.4}$$

and it verifies $\mathbb{E}(\varepsilon_{k,l}^2) = \frac{\sigma_{\text{n}}^2}{n_{\text{sample}}}$. Finally, thanks to the strong law of large numbers, we can write as MN goes to infinity

$$\mathcal{E}_{\text{ref}}^2 = \frac{1}{MN} \sum_{i=1}^{n} (\varepsilon^{k,l})^2 \to \frac{\sigma_{\text{n}}^2}{n_{\text{sample}}}. \tag{10.5}$$

Similarly, we obtain that

$$\mathcal{E}_{\text{set}}^2 \to 2 \frac{\sigma_{\text{n}}^2}{n_{\text{sample}}}. \tag{10.6}$$

When a number n of images are combined and a zoom factor λ is used, the number of samples per output pixel is around $n_{\text{sample}} \simeq \frac{n}{\lambda^2}$. Therefore, in the ideal denoising case, we have

$$\mathcal{E}_{\text{ref}} \simeq \lambda \frac{\sigma_{\text{n}}}{\sqrt{n}} \quad \text{and} \quad \mathcal{E}_{\text{set}} \simeq \sqrt{2}\lambda \frac{\sigma_{\text{n}}}{\sqrt{n}}. \tag{10.7}$$

10.3 Experiments on Synthetic Data

First, we evaluate on synthetic data the performance of our image fusion algorithm, described in Algorithm 9.1. More precisely, we focus on the performance of the irregularly sampled data fitting part, which uses classical kernel regression. In this evaluation, the transformations between the images are therefore assumed to be known. We refer to Chapter 7 for more details about the performance of the registration part.

Section 10.3.1 describes our experimental setup. The interest of the sharpening is discussed in Section 10.3.2. The best configuration for the order N_{KR} and the scale σ_{s}, depending on the sub-sampling factor λ, is obtained in Section 10.3.3. Section 10.3.4 compares our method with the burst denoising and the ACT methods. Finally, the comparison between two reconstructed images (using our method) from separated sets is considered in Section 10.3.5.

10.3.1 Experimental Setup

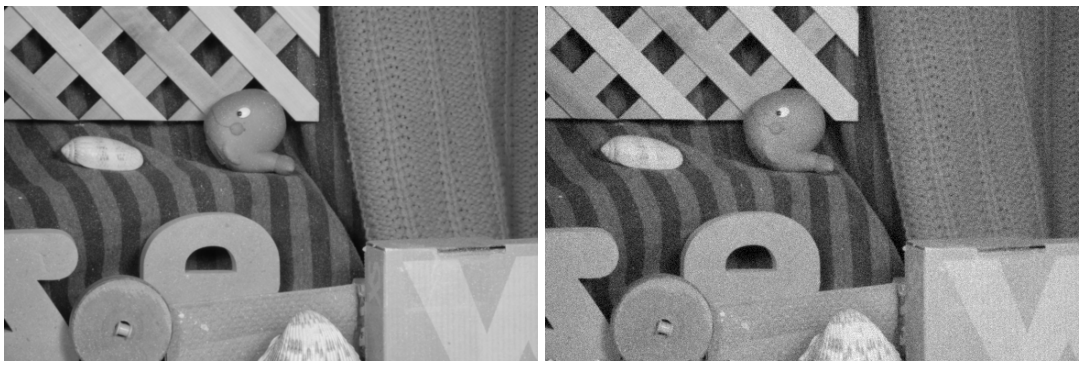
To evaluate the performance of an image fusion method we build a sequence of noisy warped and sub-sampled images from a reference image. The image reconstructed from the sequence using the image fusion method is compared to the reference image.

Building of the test sequences. We build several test sequences from the same reference image, noted I_{ref}, as follows. First, we build a base sequence $(I_i)_{1 \le i \le N_{\text{im}}}$ of warped non-noisy images, which have the same size as I_{ref}. The transformation φ_i between I_{ref} and I_i is a random homography obtained using Algorithm 6.4 with $L = 3$. The geometric transformations are computed using the *p+s-spline11* interpolation method (with half-symmetric boundary condition) that was introduced in Chapter 6. Let λ be an integer sub-sampling factor and σ_{n} be a noise level. The test sequence $(I_i^{\lambda,\sigma_{\text{n}}})_{1 \le i \le N_{\text{im}}}$ is computed from $(I_i)_{1 \le i \le N_{\text{im}}}$ by sub-sampling of factor λ (keeping only one sample over λ^2) and by adding Gaussian white noise of standard deviation σ_{n}.

Exact registration. As we focus on the performance of the irregularly data fitting part, the registration part of Algorithm 9.1, which uses the first image of the sequence as reference, is replaced by an exact registration, which uses I_{ref} as reference. The transformations used during the data fitting part are exactly the random homographies $(\varphi_i)_{1 \le i \le N_{\text{im}}}$ that were generated.

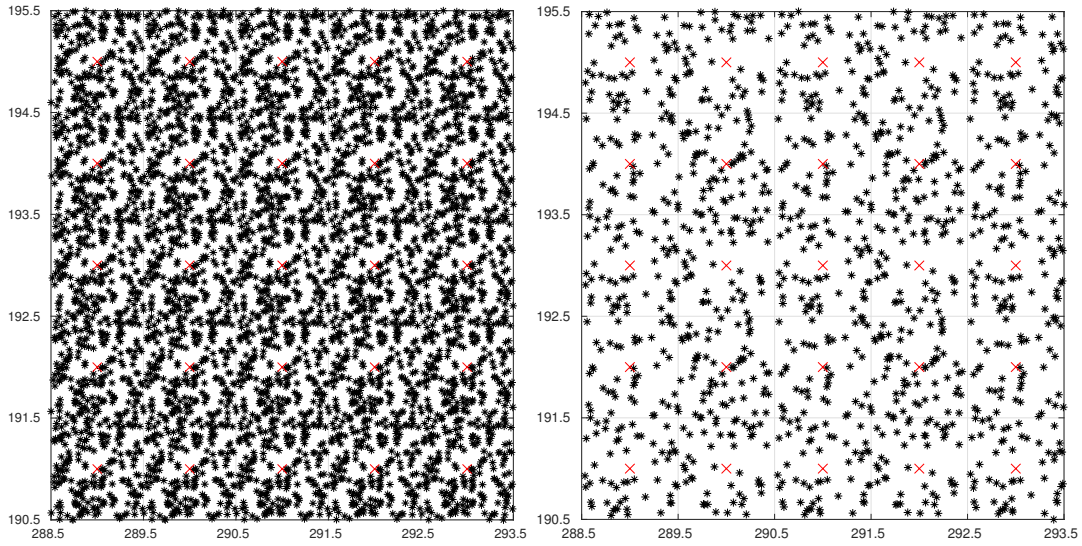
We take as reference image I_{ref} the image presented in Figure 10.1(a). It is the grayscale version of the *RubberWhale* image, which is of size 584×388 and is taken from the Middlebury database [10]. We checked that similar results are obtained using other images. The test sequences are built using $N_{\text{im}} = 200$ images for the noise levels $\sigma_{\text{n}} \in \{0, 1, 3, 5, 10, 20, 50\}$. We consider the sub-sampling factors $\lambda = 1$ and $\lambda = 2$, which correspond respectively to no super-resolution and to super-resolution by 2.

An example of noisy warped image is shown in Figure 10.1(b). The spatial repartition of the irregularly sampled data, with and without super-resolution, is shown in Figure 10.2. As the homographies φ_i are locally similar to translations, the data repartition seems to have a periodicity of λ along each direction.



(a) Reference image I_{ref} (b) Noisy warped image $I_1^{1,20}$

Figure 10.1: Synthetic data for the performance evaluation. The reference image I_{ref} in (a) is used in Section 10.3 to generate the test sequences. It is the grayscale version of the *RubberWhale* image, which is of size 584×388 and is taken from the Middlebury database [10]. The noisy warped image $I_1^{1,20}$ in (b) is the first image of the test sequence without sub-sampling and for the noise level $\sigma_{\text{n}} = 20$.



(a) $\lambda = 1$ (b) $\lambda = 2$ (super-resolution)

Figure 10.2: Spatial repartition of the irregularly sampled data, around the center of the reference image, in the synthetic case (Section 10.3). The factor λ corresponds to the sub-sampling factor used to generate the test sequences or, equivalently, to the zoom factor used in the image formation algorithm. As the homographies φ_i are locally similar to translations, the data repartition is approximately λ-periodic in each direction.

10.3.2 Discussion of the Sharpening Step

In this section, we discuss the impact of the sharpening step. We recall that it consists in applying the inverse of the asymptotic equivalent filter introduced in Section 9.3. To do that we compare the results with and without performing the sharpening. The comparisons for the scale $\sigma_s = 0.7$ and for various values of sub-sampling factor λ, noise level σ_n and order N_{KR} are shown in Figure 10.3 to Figure 10.8. The sets of parameters used for each figure are the following:

	Figure 10.3	Figure 10.4	Figure 10.5	Figure 10.6	Figure 10.7	Figure 10.8
λ	1	1	1	2	2	2
σ_n	0	5	50	0	0	5
N_{KR}	0	0	0	0	2	2

Blur removal. As expected, in every case the reconstructed image without sharpening is blurry (see (a)). The high frequency content is not well reconstructed. The spectrum seems to have been low-pass filtered (see (c)). The difference to the reference image is high at the edges or in textured areas (see (e)). The reconstructed image with sharpening does look sharper (see (b)) and has a richer high frequency content (see (d)).

However, performing the sharpening does not necessarily provide a better result, in terms of $\mathcal{E}_{ref}$ or visually. The impact of the sharpening highly depends on the noise level and on the spatial data repartition.

Residual noise increase. The sharpening increases the high frequency content of the reconstructed image. Therefore it also inevitably increases the residual noise. There is a trade-off between removing the blur and increasing the noise. For instance consider the results without sub-sampling ($\lambda = 1$) for order $N_{KR} = 0$ and noise levels $\sigma_n \in \{0, 5, 50\}$. In Figure 10.3 and Figure 10.4, i.e. for a low level of noise, the sharpening leads to better results both visually and in terms of $\mathcal{E}_{ref}$. On the contrary for a high noise level, as in Figure 10.5, the distance to the reference $\mathcal{E}_{ref}$ is higher after sharpening. However this does not mean that it should not be performed. Indeed, the residual after sharpening is not signal dependent and is mostly composed of white noise in Figure 10.4(f),(h) and Figure 10.5(f),(h).

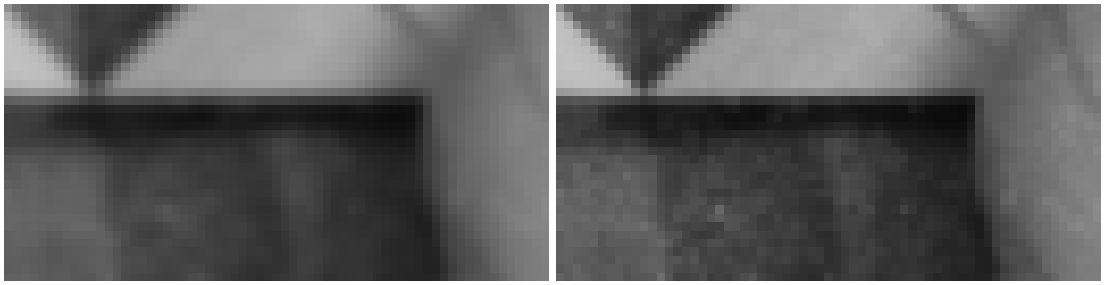
Spatial repartition. The sharpening step consists in applying the inverse of the asymptotic equivalent filter, which is an approximation of the equivalent filter. If the approximation is not tight enough, the sharpening may not be adapted and may introduce additional errors. Typically, this is the case when the data repartition is not dense or uniform enough. For instance in the super-resolution case ($\lambda = 2$) the data repartition seems to have a $(2,2)$-periodicity and is not as dense as for $\lambda = 1$ (see Figure 10.2). Consequently, in the results corresponding to $\lambda = 2$, shown in Figure 10.6 to Figure 10.8, the residual after sharpening remains signal dependent and contains some high-frequency structures. In Figure 10.8, for $\sigma_n = 5$ and $N_{KR} = 2$, the signal dependency is negligible with respect to the residual noise.

Sharpening and order. The asymptotic equivalent filter k_2 (corresponding to $N_{KR} = 2$) is closer to the Dirac function δ_0 than k_0 (corresponding to $N_{KR} = 0$). More precisely, we have

$$0 \leq \mathcal{F}(k_0) \leq \mathcal{F}(k_2) \leq 1. \tag{10.8}$$

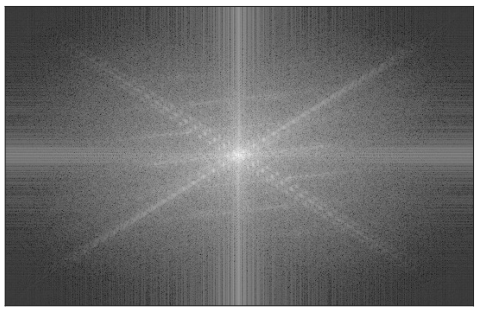
Therefore using $N_{KR} = 2$ introduces less blur during the image computation by classical kernel regression and requires less sharpening. In the super-resolution case without noise ($\lambda = 2$ and $\sigma_n = 0$), the sharpened image using $N_{KR} = 0$ contains zipper artefacts (see Figure 10.6). This is not the case for $N_{KR} = 2$ (see Figure 10.7).

To sharpen or not to sharpen ? In general the answer mostly depends on the user's goal, but sharpening will deliver a sharper image that is closer to the original, as illustrated in Figures 10.3 and 10.4. In the following, unless specified, we consider the results with sharpening.

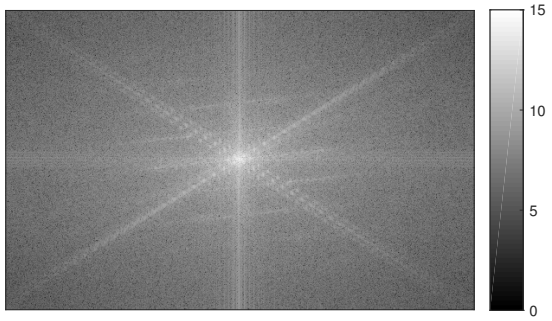


(a) Without sharpening

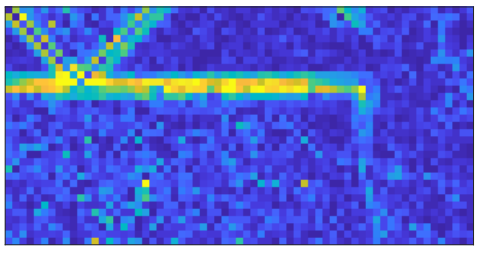
(b) With sharpening



(c) Spectrum of (a)

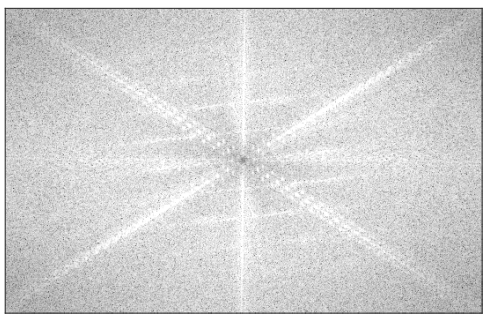


(d) Spectrum of (b)

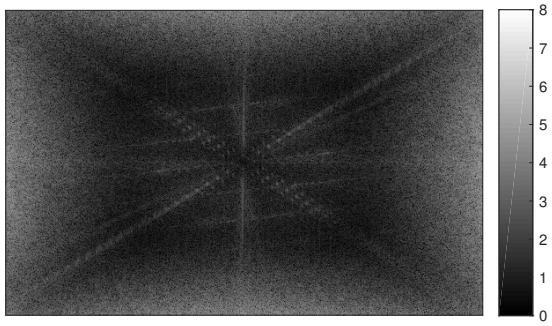


(e) Residual for (a) ($\mathcal{E}_{\text{ref}} = 2.90$)

(f) Residual for (b) ($\mathcal{E}_{\text{ref}} = 0.04$)

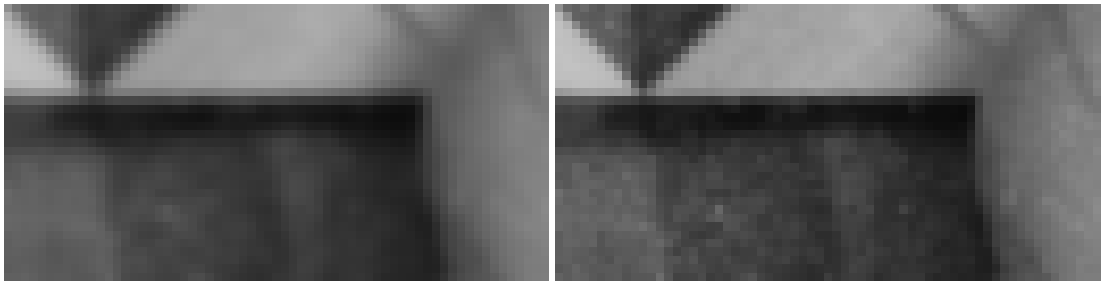


(g) Spectrum of (e)

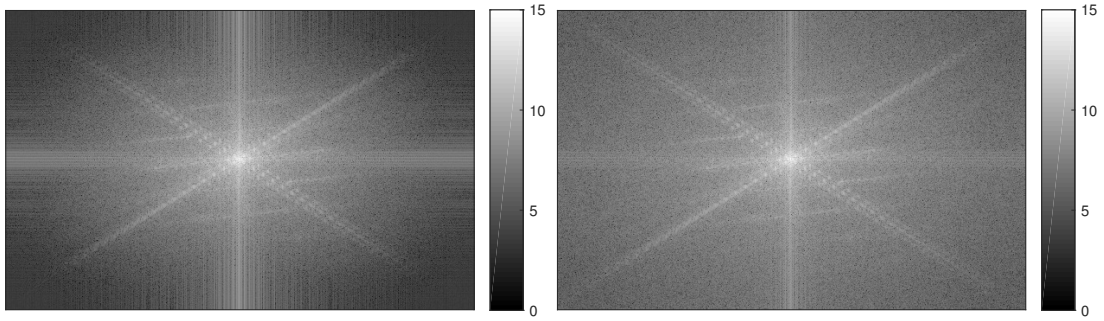


(h) Spectrum of (f)

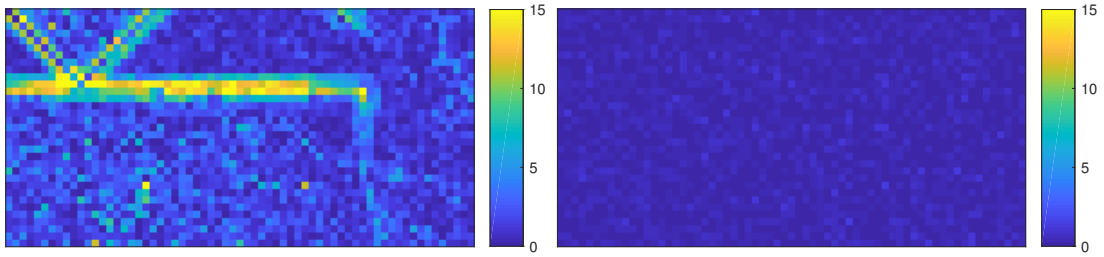
Figure 10.3: Comparison without and with sharpening for $\lambda = 1$, $N_{\text{KR}} = 0$ and $\sigma_{\text{n}} = 0$. The value $\sigma_{\text{s}} = 0.7$ is used. The reconstructed image after sharpening looks sharper and is closer to the reference image.



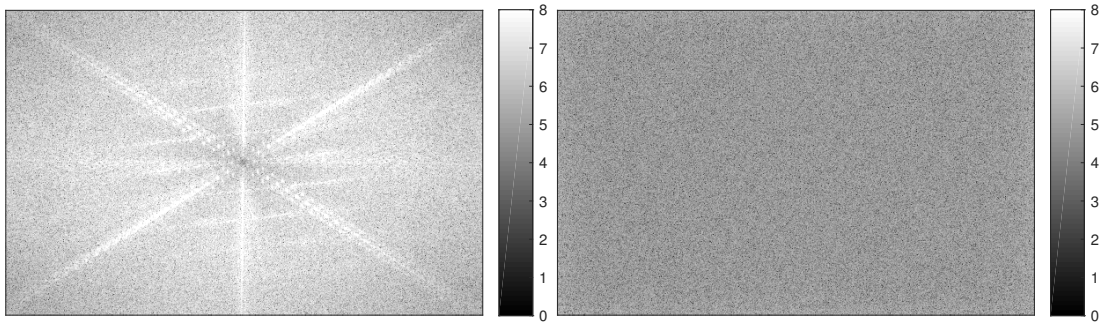
(a) Without sharpening (b) With sharpening



(c) Spectrum of (a) (d) Spectrum of (b)

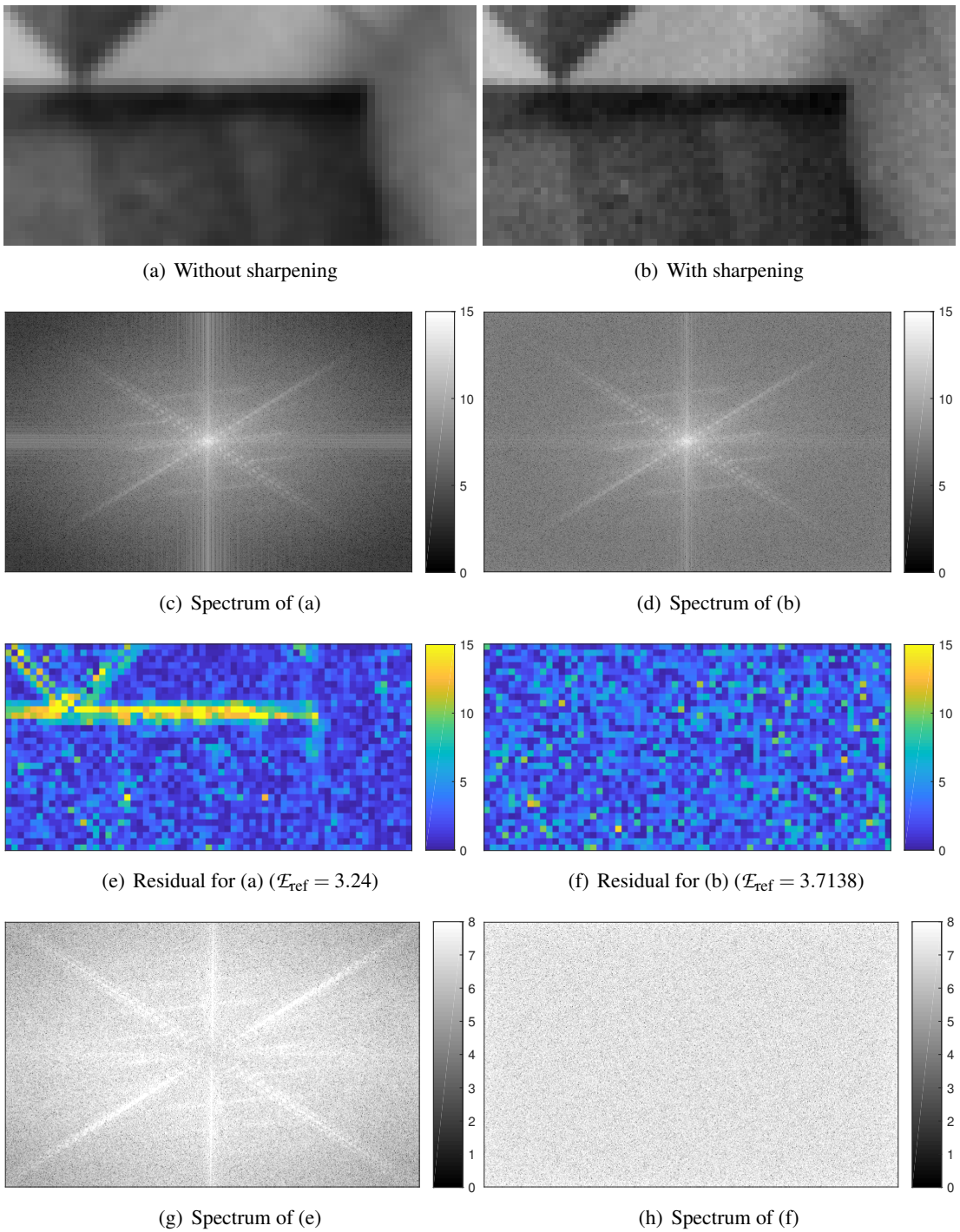


(e) Residual for (a) ($\mathcal{E}_{\text{ref}} = 2.90$) (f) Residual for (b) ($\mathcal{E}_{\text{ref}} = 0.37$)



(g) Spectrum of (e) (h) Spectrum of (f)

Figure 10.4: Comparison without and with sharpening for $\lambda = 1$, $N_{\text{KR}} = 0$ and $\sigma_{\text{n}} = 5$. The value $\sigma_{\text{s}} = 0.7$ is used. The reconstructed image after sharpening looks sharper and is closer to the reference image. The residual after sharpening is mostly composed of white noise.



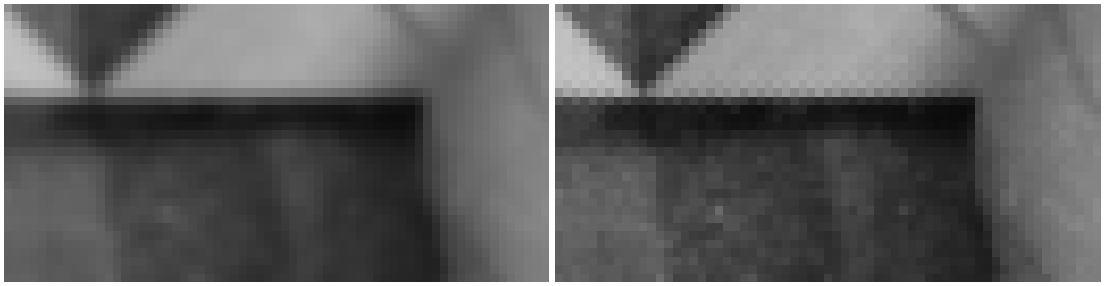
(a) Without sharpening (b) With sharpening

(c) Spectrum of (a) (d) Spectrum of (b)

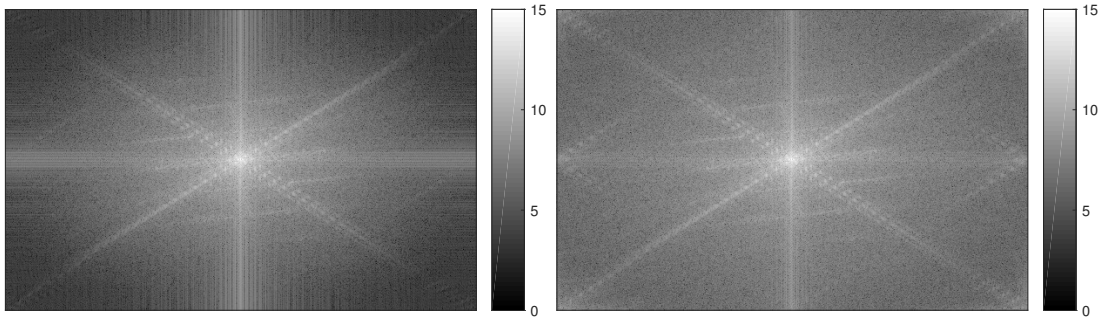
(e) Residual for (a) ($\mathcal{E}_{\text{ref}} = 3.24$) (f) Residual for (b) ($\mathcal{E}_{\text{ref}} = 3.7138$)

(g) Spectrum of (e) (h) Spectrum of (f)

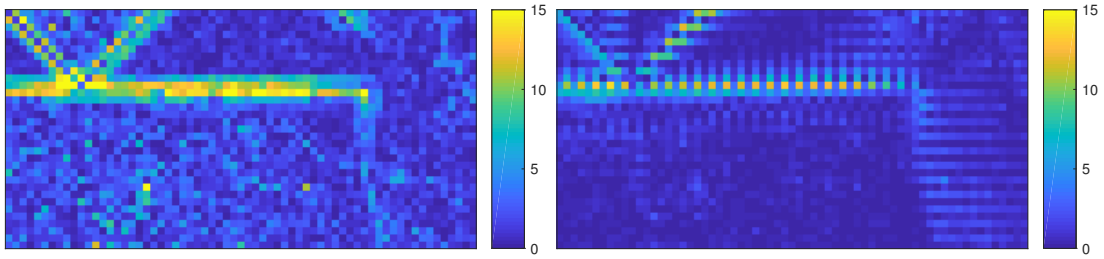
Figure 10.5: Comparison without and with sharpening for $\lambda = 1$, $N_{\text{KR}} = 0$ and $\sigma_{\text{n}} = 50$. The value $\sigma_{\text{s}} = 0.7$ is used. The reconstructed image after sharpening looks sharper but is not closer to the reference image. The sharpening increases the residual noise but the residual after sharpening is mostly composed of white noise.



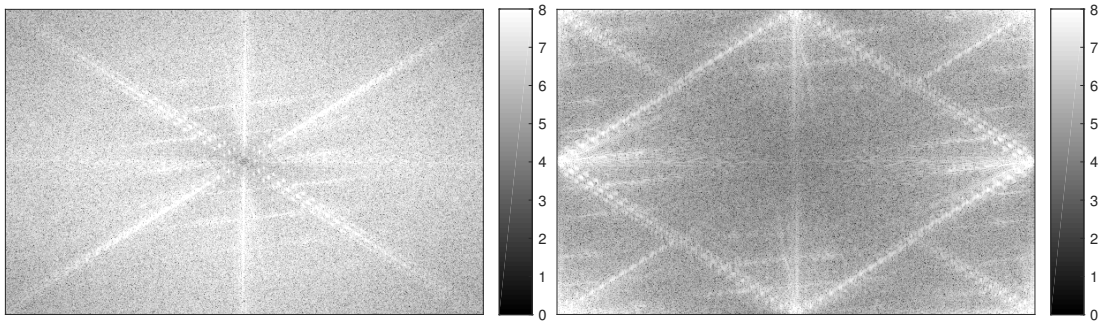
(a) Without sharpening (b) With sharpening



(c) Spectrum of (a) (d) Spectrum of (b)

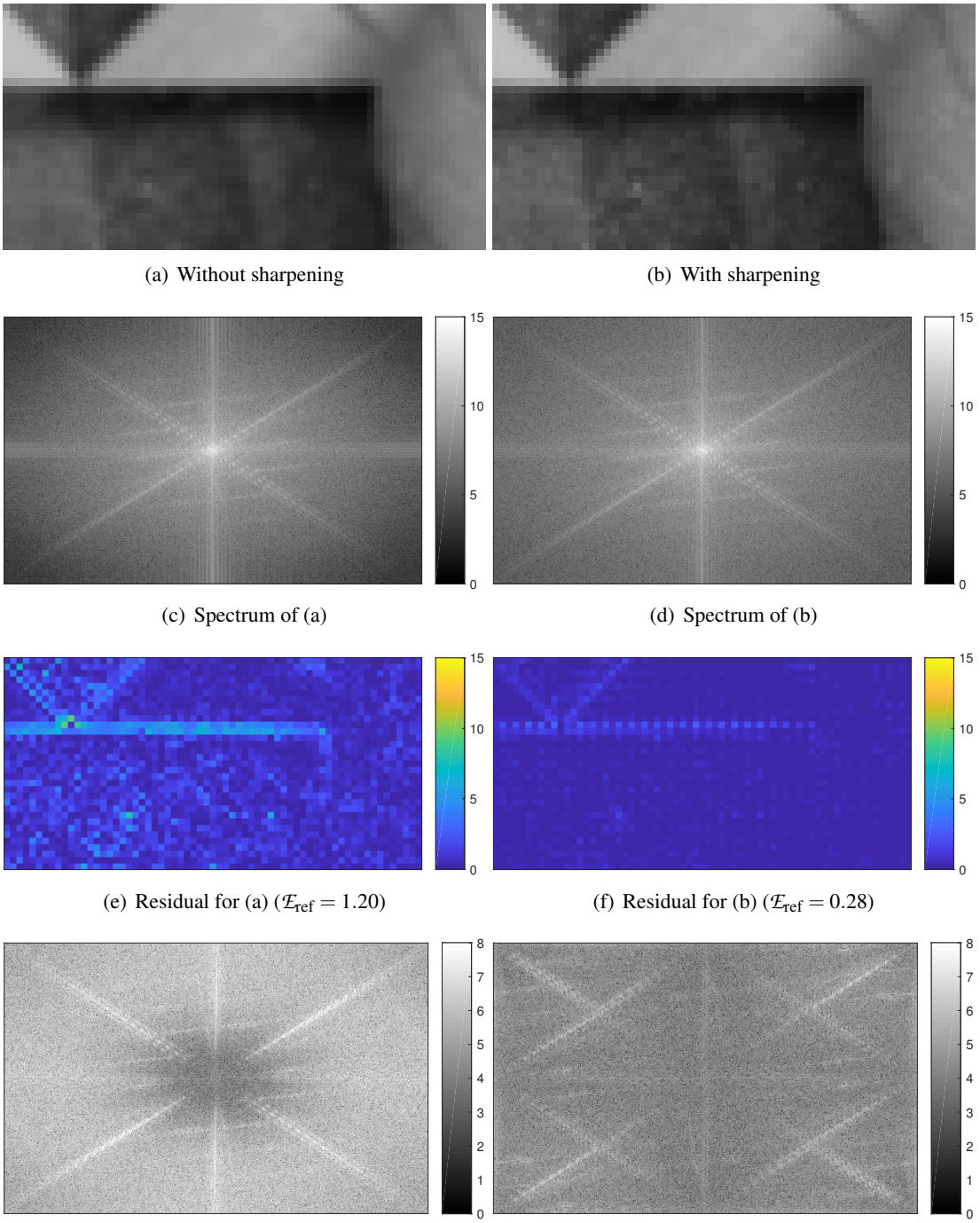


(e) Residual for (a) ($\mathcal{E}_{\text{ref}} = 2.92$) (f) Residual for (b) ($\mathcal{E}_{\text{ref}} = 1.50$)



(g) Spectrum of (e) (h) Spectrum of (f)

Figure 10.6: Comparison without and with sharpening for $\lambda = 2$, $N_{\text{KR}} = 0$ and $\sigma_{\text{n}} = 0$. The value $\sigma_{\text{s}} = 0.7$ is used. The reconstructed image after sharpening is closer to the reference image but contains zipper artifacts. With $N_{\text{KR}} = 0$ and $\lambda = 2$ the sharpening is not adapted and introduces artifacts.



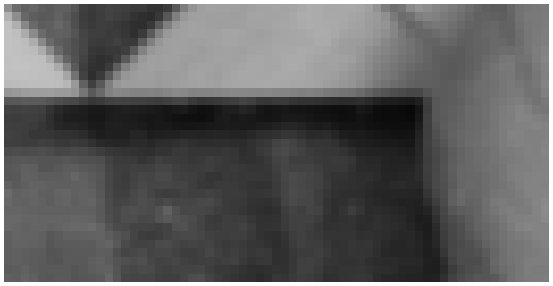
(a) Without sharpening (b) With sharpening

(c) Spectrum of (a) (d) Spectrum of (b)

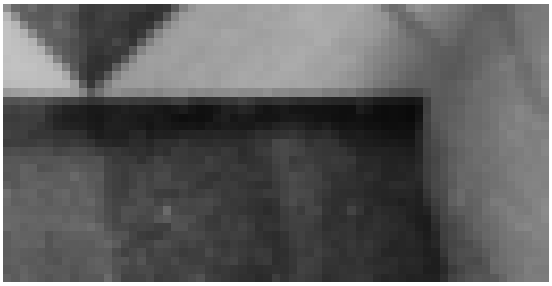
(e) Residual for (a) ($\mathcal{E}_{\text{ref}} = 1.20$) (f) Residual for (b) ($\mathcal{E}_{\text{ref}} = 0.28$)

(g) Spectrum of (e) (h) Spectrum of (f)

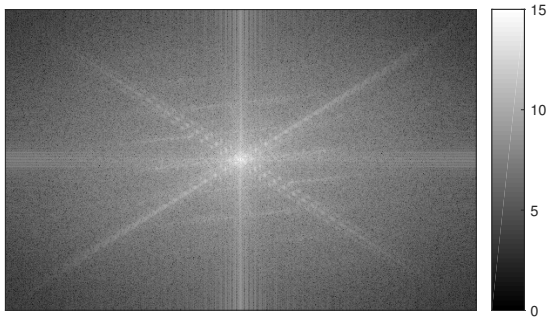
Figure 10.7: Comparison without and with sharpening for $\lambda = 2$, $N_{\text{KR}} = 2$ and $\sigma_{\text{n}} = 5$. The value $\sigma_{\text{s}} = 0.7$ is used. The reconstructed image after sharpening looks sharper and is closer to the reference image. Because of the inadequate data repartition the residual is slightly signal dependent and contains a moderate amount of high frequency artifacts. The results are better than the ones of Figure 10.6 where $N_{\text{KR}} = 0$ is used.



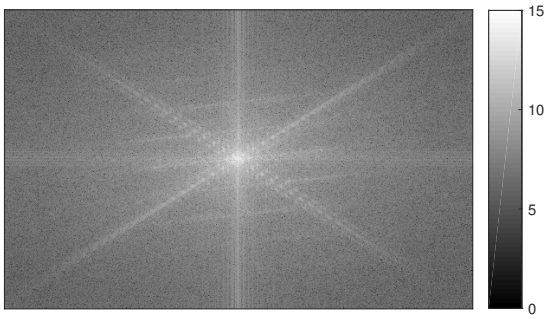
(a) Without sharpening



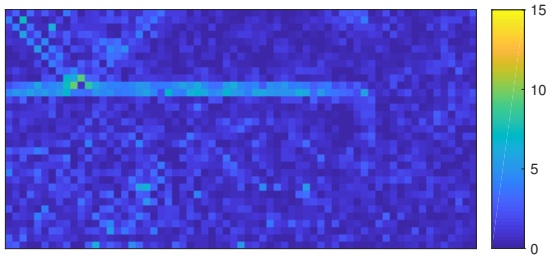
(b) With sharpening



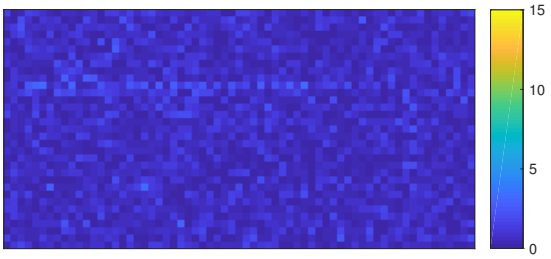
(c) Spectrum of (a)



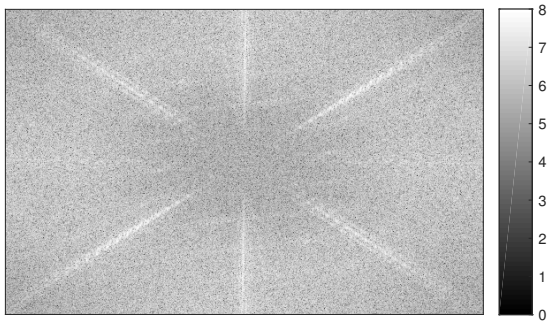
(d) Spectrum of (b)



(e) Residual for (a) ($\mathcal{L}_{\text{ref}} = 1.28$)



(f) Residual for (b) ($\mathcal{L}_{\text{ref}} = 0.81$)



(g) Spectrum of (e)

(h) Spectrum of (f)

Figure 10.8: Comparison without and with sharpening for $\lambda = 2$, $N_{\text{KR}} = 2$ and $\sigma_{\text{n}} = 5$. The value $\sigma_{\text{s}} = 0.7$ is used. The reconstructed image after sharpening looks sharper and is closer to the reference image. The residual after sharpening is signal dependent but is mostly composed of white noise.

10.3.3 Choice of Order and Scale

In this section we find the best options for the order N_{KR} and scale σ_s, when the sharpening is performed, depending on the sub-sampling factor λ and the noise level σ_n. To do that we compare the results using $N_{im} = 200$ images for $\sigma_s \in \{0.10, 0.15, \ldots, 0.85\}$ and $N_{KR} \in \{0, 2\}$. Our conclusions do not differ for different numbers of images exceeding 50.

Without super-resolution ($\lambda = 1$). The evolution of $\mathcal{E}_{ref}$ with the scale σ_s for $N_{KR} \in \{0, 2\}$ and $\sigma_n \in \{0, 5, 20, 50\}$ is shown in Figure 10.9. Without noise using $N_{KR} = 2$ leads to the smallest error for small values of σ_s while as soon as there is noise using $N_{KR} = 0$ is always (slightly) better. Indeed, the denoising capacity of higher order is less important because more parameters are estimated (here 6 against 1) and the model is more likely to fit noise.

The reconstructed images of the two orders (with $\sigma_s = 0.7$) are compared, respectively without noise and with $\sigma_n = 5$, in Figure 10.11 and Figure 10.12. Without noise the results are difficult to interpret since $\mathcal{E}_{ref}$ is of the same order as the interpolation error used to generate the data. With noise the results are similar but slighlty better for $N_{KR} = 0$ and the residuals are mostly composed of white noise. Another advantage of using $N_{KR} = 0$ is that less computations are required (see Figure 10.10). Therefore $N_{KR} = 0$ should be used for $\lambda = 1$. Note that this analysis is valid only when the data repartition is uniform and dense enough. Otherwise the incoming conclusions corresponding to $\lambda = 2$ should be considered.

For $N_{KR} = 0$ the error $\mathcal{E}_{ref}$ decreases with the scale σ_s (see Figure 10.9). The decay is slower and slower and is almost negligible after $\sigma_s = 0.7$. The residuals for $\sigma_s \in \{0.3, 0.5, 0.7\}$ are compared in Figure 10.13 for $\sigma_n \in \{0, 5, 20\}$. It is clear that the best results are obtained for $\sigma_s = 0.7$. As σ_s increases the number of neighboring pixels used to compute the pixel values by kernel regression increases. As shown in Figure 10.10 the computation time increases with σ_s. Therefore, we choose to use σ_s around 0.7.

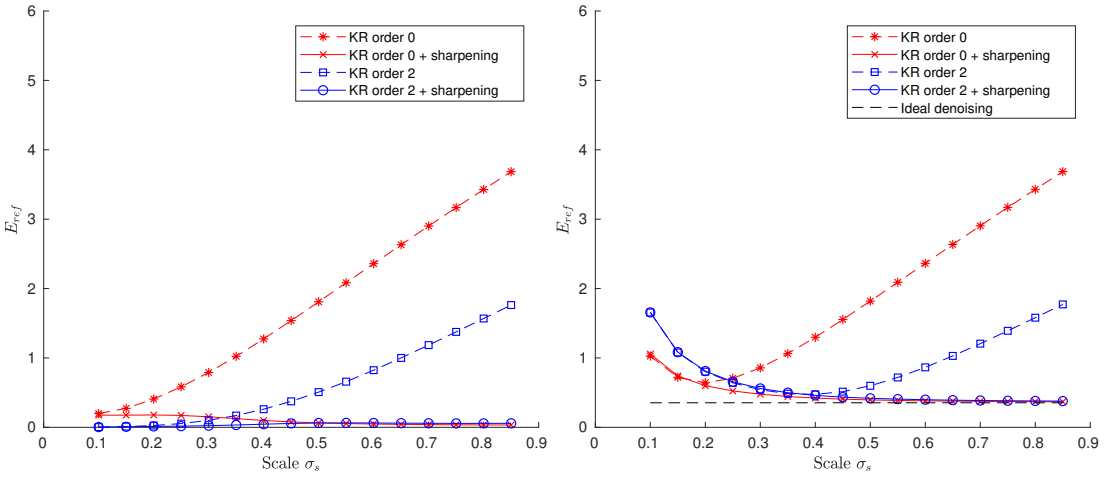
With super-resolution ($\lambda = 2$). The evolution of $\mathcal{E}_{ref}$ with the scale σ_s for $N_{KR} \in \{0, 2\}$ and $\sigma_n \in \{0, 5, 20, 50\}$ is shown in Figure 10.14. Without noise using $N_{KR} = 2$ leads to the smallest error. As soon as the data are noisy, the error $\mathcal{E}_{ref}$ is smaller for small values of σ_s when $N_{KR} = 0$ is used and smaller for high values when $N_{KR} = 2$. Actually for the largest noise level, $\sigma_n = 50$, using $N_{KR} = 0$ leads to a smaller error $\mathcal{E}_{ref}$ thanks to a better denoising but the results are not visually satisfactory for the reasons evoked below.

The reconstructed images of the two orders (with $\sigma_s = 0.7$) are compared, respectively without noise and with $\sigma_n = 5$, in Figure 10.15 and Figure 10.16. A zipper effect is visible around the edges of the results of $N_{KR} = 0$. As seen in Section 10.3.2, this is introduced by the sharpening when the data repartition is not uniform or not dense enough, which is the case here. Therefore, using $N_{KR} = 2$ is recommended (even though the denoising capacity may be lessened). The residuals for $\sigma_s \in \{0.3, 0.5, 0.7\}$ are compared in Figure 10.17 for $\sigma_n \in \{0, 5, 20\}$. Without noise small values of σ_s provides the best results because it corresponds to a less important sharpening. For noisy data the best results are obtained for $\sigma_s = 0.7$, which is the value we recommend.

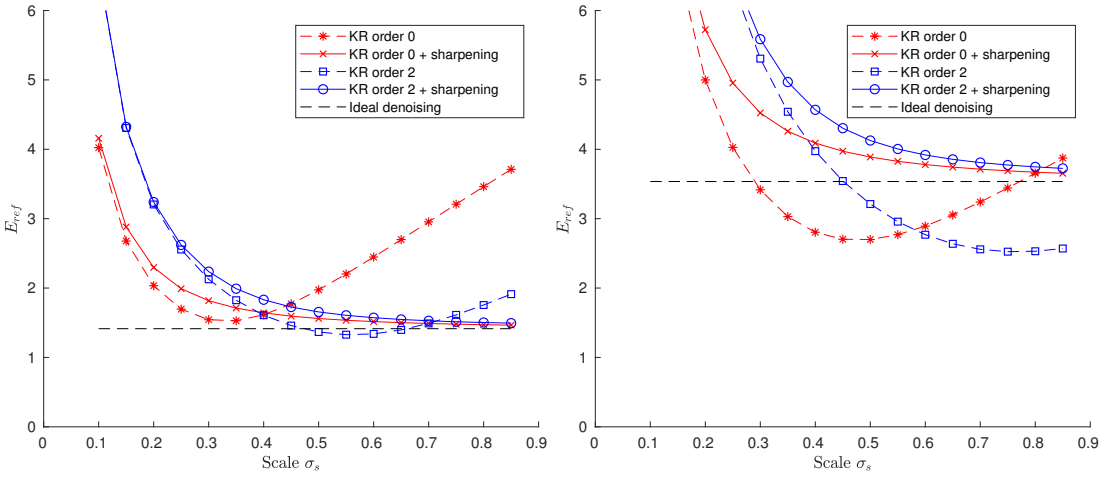
About the sharpening. In Figure 10.9 and Figure 10.14, the evolution of $\mathcal{E}_{ref}$ when the sharpening is not performed is also shown. For high noise levels ($\sigma_n \geq 20$) the error $\mathcal{E}_{ref}$ without sharpening may be smaller than the residual noise in the ideal denoising case. In other words the blur removes more noise than it introduces error. There is a compromise between using more neighboring pixels (to denoise) and increasing the high-frequency error (because of the blur). Note that after the sharpening the error is always higher than the ideal residual noise.

Summary. The denoising ability of the classical kernel regression is more important with $N_{KR} = 0$ and with high values of σ_s. However this implies a strong sharpening step that may introduce artifacts. We choose the following compromise: using $N_{KR} = 0$ when $\lambda = 1$ and $N_{KR} = 2$

otherwise ($\lambda > 1$). In both cases a scale around 0.7 should be considered (for instance $\sigma_s = \frac{\sqrt{2}}{2} \simeq 0.71$).



(a) $\sigma_n = 0$ (b) $\sigma_n = 5$



(c) $\sigma_n = 20$ (d) $\sigma_n = 50$

Figure 10.9: Evolution of the error $\mathcal{E}_{\text{ref}}$ with the scale σ_s for $N_{\text{KR}} \in \{0,2\}$ and $\sigma_n \in \{0,5,20,50\}$ ($N_{\text{im}} = 200$ images with $\lambda = 1$), and comparing the sharpening-no sharpening options. Without noise using $N_{\text{KR}} = 2$ leads to the smallest error for small values of σ_s while as soon as there is noise using $N_{\text{KR}} = 0$ is always (slightly) better. Indeed, the denoising capacity of higher order convolution decays because more parameters are being estimated at each pixel (here 6 against 1). Therefore the model is more likely to fit noise. $\mathcal{E}_{\text{ref}}$ decreases with the scale σ_s. The decay is slower and slower and is almost negligible after $\sigma_s = 0.7$. For high noise levels ($\sigma_n \geq 20$) the error $\mathcal{E}_{\text{ref}}$ without sharpening may be smaller than the residual noise in the ideal denoising case. In other words the blur removes more noise than it introduces error. There is a compromise between using more neighboring pixels (to denoise) and increasing the high-frequency error (because of the blur). Note that after the sharpening the error is always higher than the ideal residual noise.

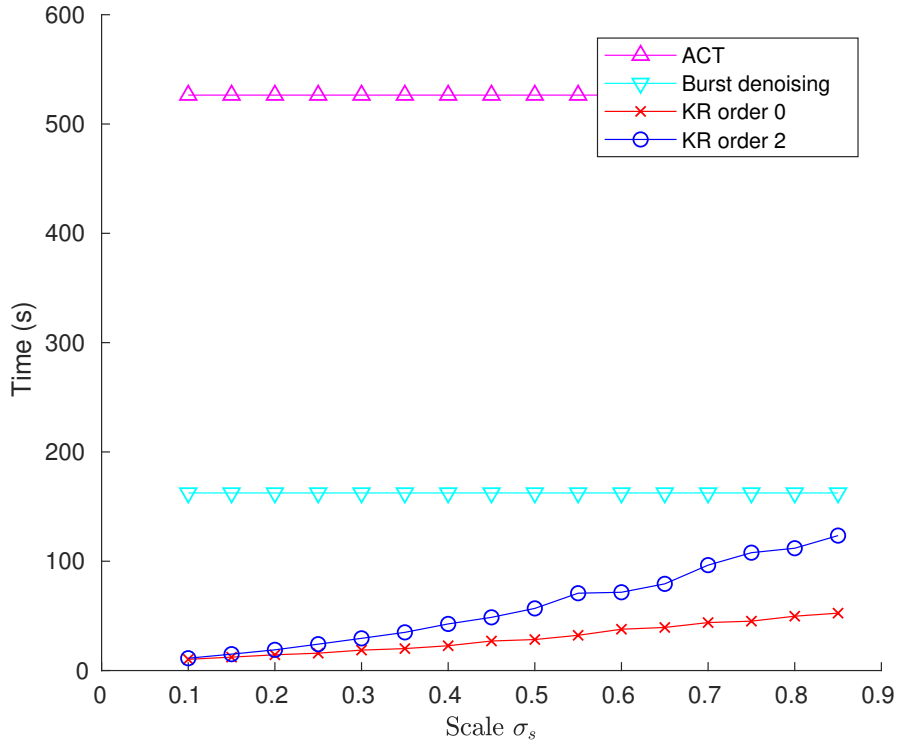
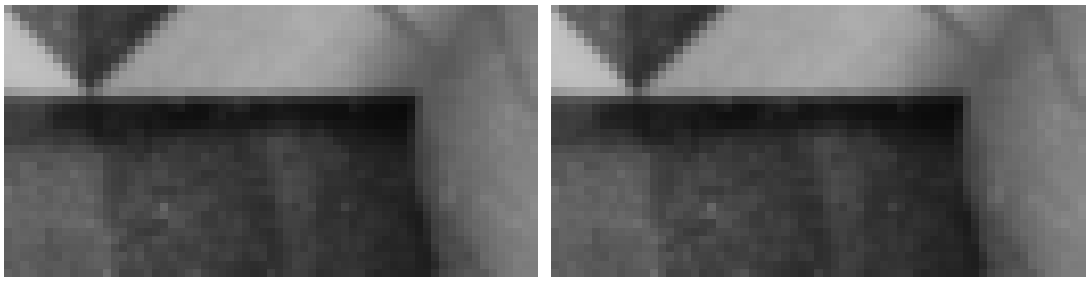
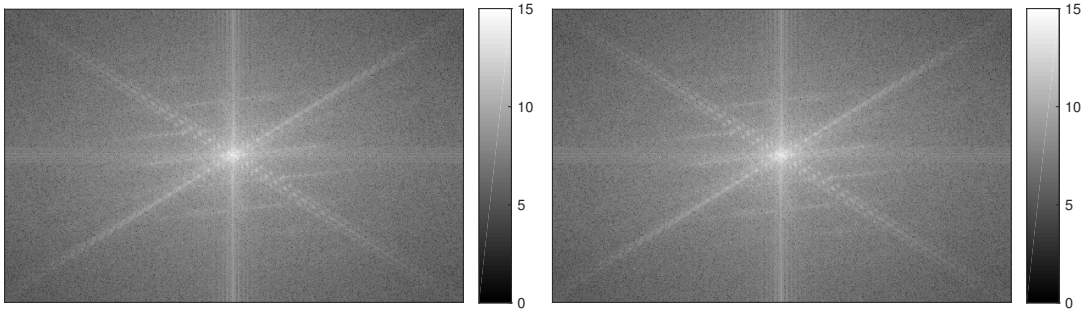


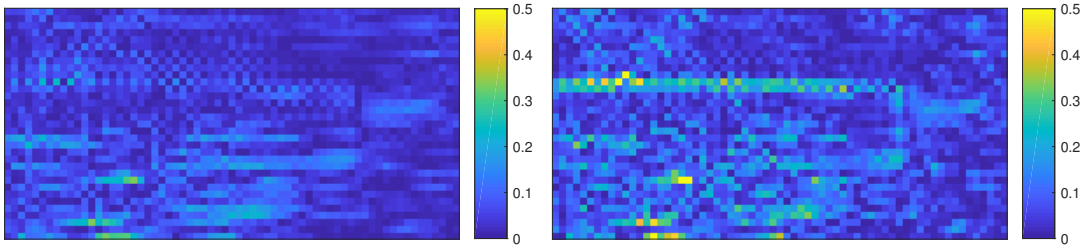
Figure 10.10: Evolution of the computation time with the scale σ_s for $N_{\text{im}} = 200$ images and $\lambda = 1$. The sharpening step is taken into account for the kernel regression methods. The kernel regression methods are most efficient. The computation time increases with the scale σ_s but remains way smaller than for the burst combination and the ACT methods. The ACT method is computationally and memory greedy.



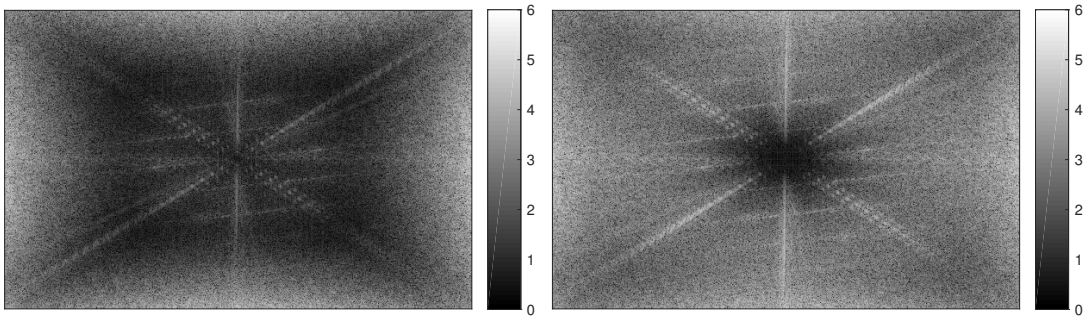
(a) Order 0 (b) Order 2



(c) Spectrum of (a) (d) Spectrum of (b)

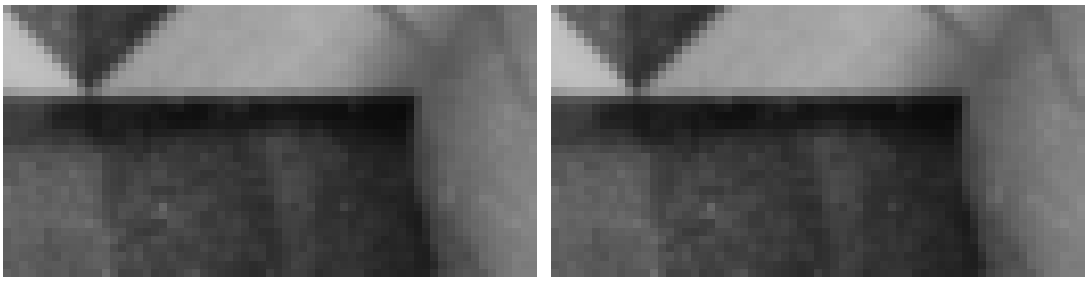


(e) Residual for (a) ($\mathcal{E}_{\text{ref}} = 0.039$) (f) Residual for (b) ($\mathcal{E}_{\text{ref}} = 0.0578$)

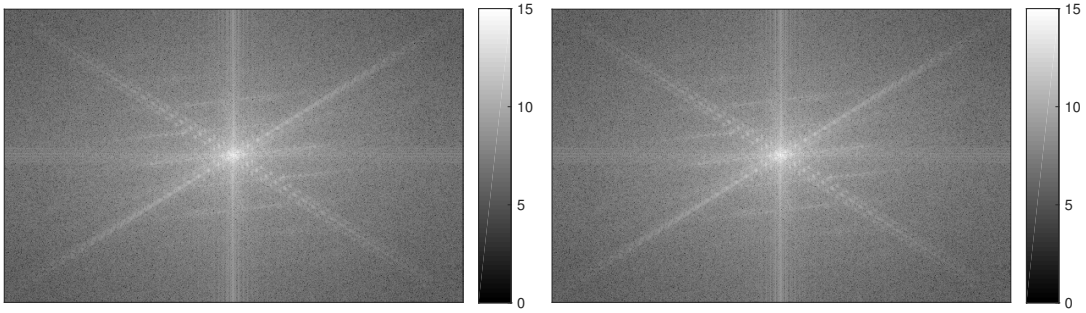


(g) Spectrum of (e) (h) Spectrum of (f)

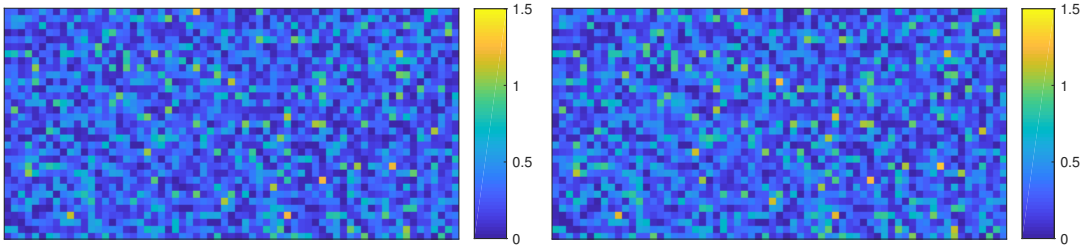
Figure 10.11: Comparison of the reconstructed images of the two orders for $\lambda = 1$ and $\sigma_n = 0$ with $\sigma_s = 0.7$. The sharpening is performed. The results are difficult to interpret since $\mathcal{E}_{\text{ref}}$ is of the same order as the interpolation error used to generate the data.



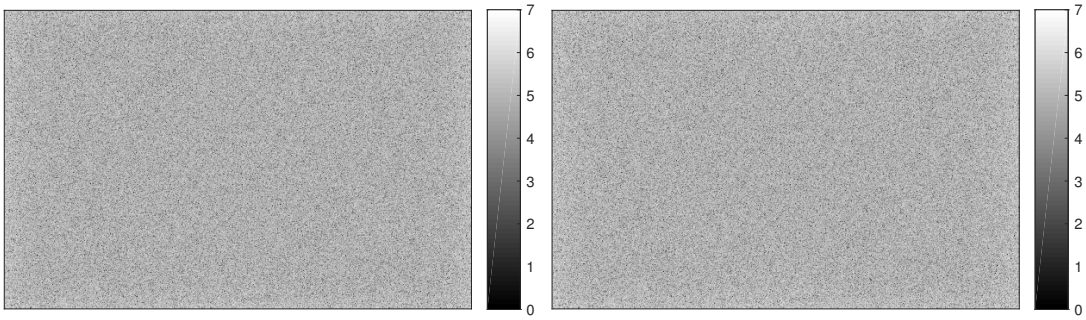
(a) Order 0 (b) Order 2



(c) Spectrum of (a) (d) Spectrum of (b)

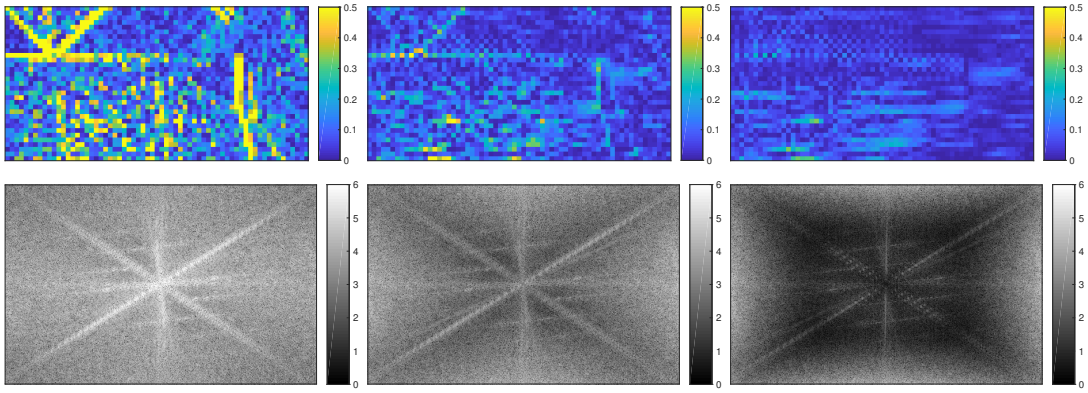


(e) Residual for (a) ($\mathcal{E}_{\text{ref}} = 0.3744$) (f) Residual for (b) ($\mathcal{E}_{\text{ref}} = 0.3862$)

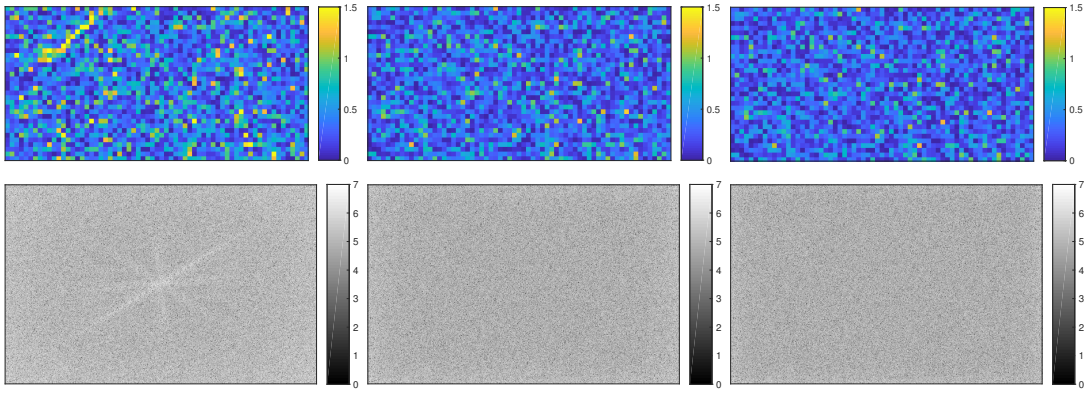


(g) Spectrum of (e) (h) Spectrum of (f)

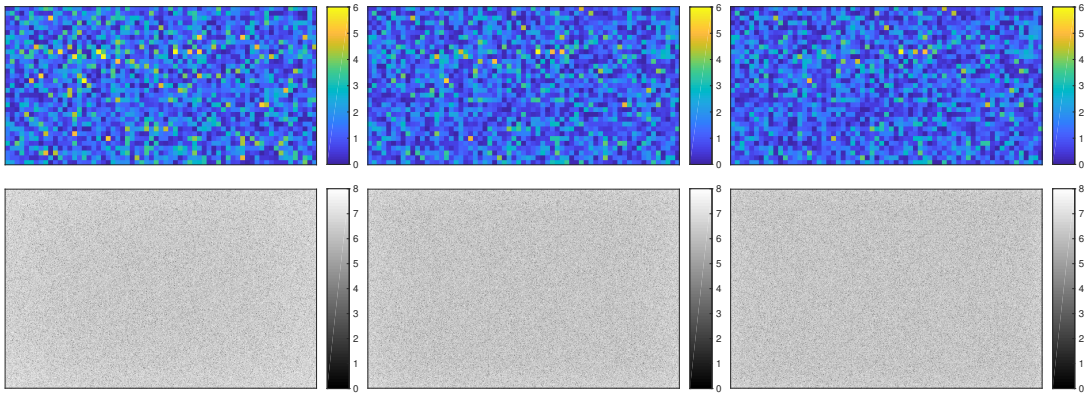
Figure 10.12: Comparison of the reconstructed images of the two orders for $\lambda = 1$ and $\sigma_n = 5$ with $\sigma_s = 0.7$. The sharpening is performed. The results are similar but slightly better for $N_{\text{KR}} = 0$. The residuals are mostly composed of white noise.



(a) $\sigma_n = 0$ (0.1517,0.0637,0.0390)

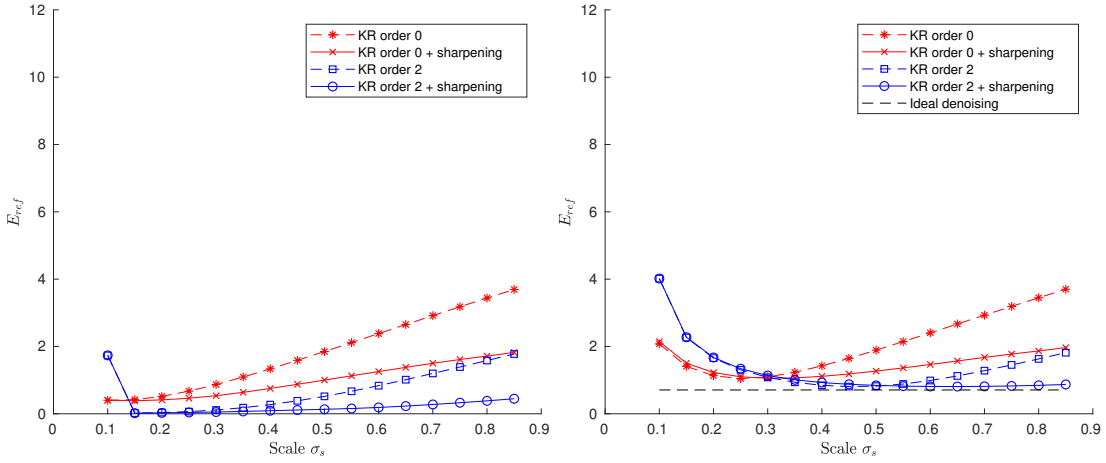


(b) $\sigma_n = 5$ (0.4771,0.3948,0.3744)



(c) $\sigma_n = 20$ (1.8178,1.5609,1.4897)

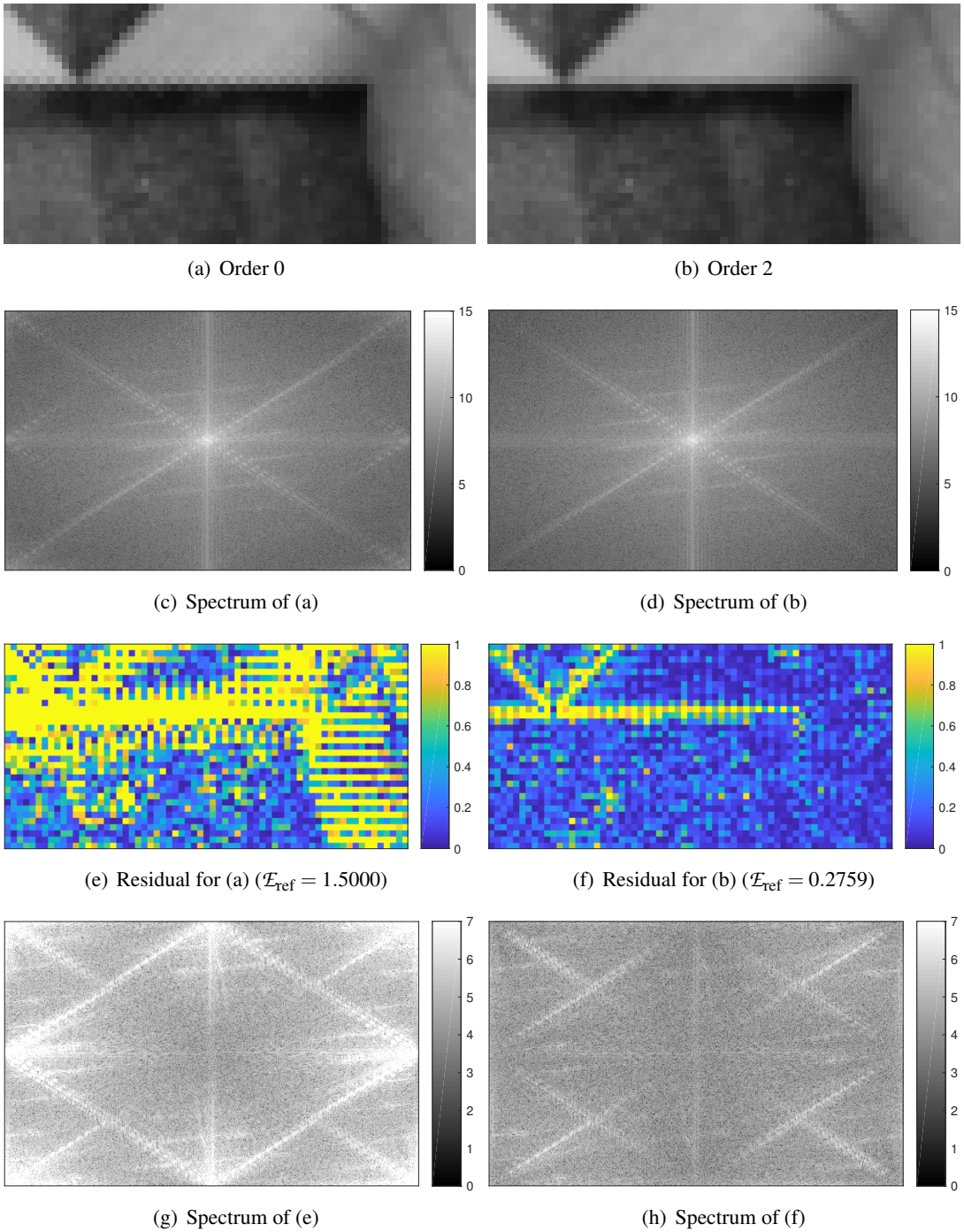
Figure 10.13: Comparison of the residuals for $\sigma_s \in \{0.3, 0.5, 0.7\}$ (from left to right) when $\lambda = 1$ and $N_{KR} = 0$. The sharpening is performed. The values between parentheses are the corresponding error $\mathcal{E}_{ref}$. It is clear that the best results are obtained for $\sigma_s = 0.7$. When the data are noisy the residuals are mostly composed of white noise (except for $\sigma_n = 0.3$).



(a) $\sigma_n = 0$ (b) $\sigma_n = 5$

(c) $\sigma_n = 20$ (d) $\sigma_n = 50$

Figure 10.14: Evolution of the error $\mathcal{E}_{\text{ref}}$ with the scale σ_s for $N_{\text{KR}} \in \{0,2\}$ and $\sigma_n \in \{0,5,20,50\}$ ($N_{\text{im}} = 200$ images with $\lambda = 2$). The sharpening and no-sharpening options are considered for kernel regression. Without noise using $N_{\text{KR}} = 2$ leads to the smallest error. As soon as the data are noisy, the error $\mathcal{E}_{\text{ref}}$ is smaller for small values of σ_s when $N_{\text{KR}} = 0$ is used and smaller for high values when $N_{\text{KR}} = 2$. Actually for the largest noise level, $\sigma_n = 50$, using $N_{\text{KR}} = 0$ leads to a smaller error $\mathcal{E}_{\text{ref}}$. For high noise levels ($\sigma_n \geq 20$) the error $\mathcal{E}_{\text{ref}}$ without sharpening may be smaller than the residual noise in the ideal denoising case. In other words the blur removes more noise than it introduces error. There is a compromise between using more neighboring pixels (to denoise) and increasing the high-frequency error (because of the blur). Note that after the sharpening the error is always higher than the ideal residual noise.



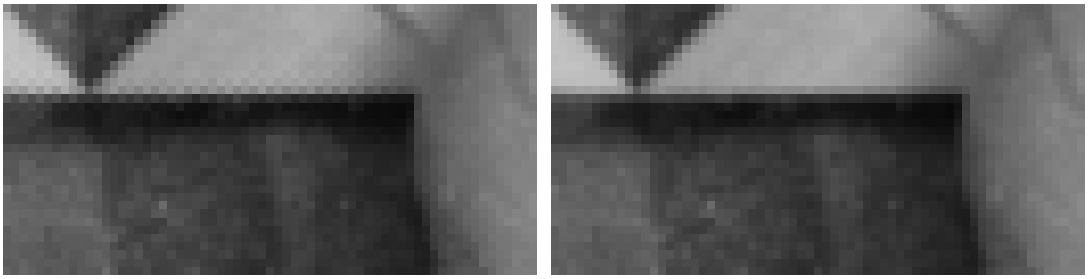
(a) Order 0 (b) Order 2

(c) Spectrum of (a) (d) Spectrum of (b)

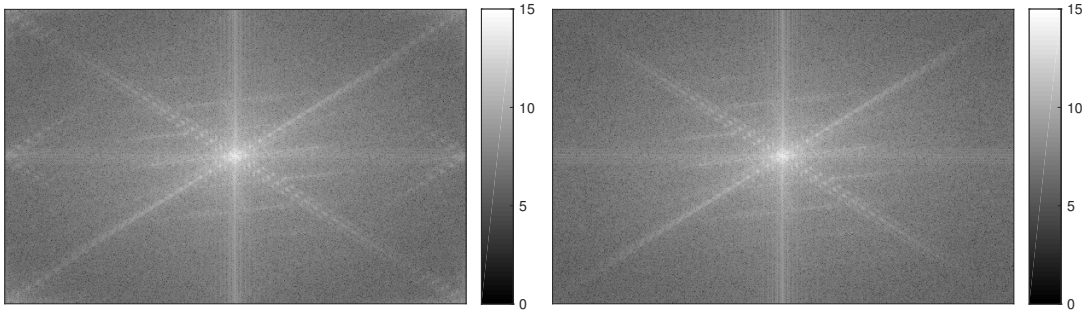
(e) Residual for (a) ($\mathcal{E}_{\text{ref}} = 1.5000$) (f) Residual for (b) ($\mathcal{E}_{\text{ref}} = 0.2759$)

(g) Spectrum of (e) (h) Spectrum of (f)

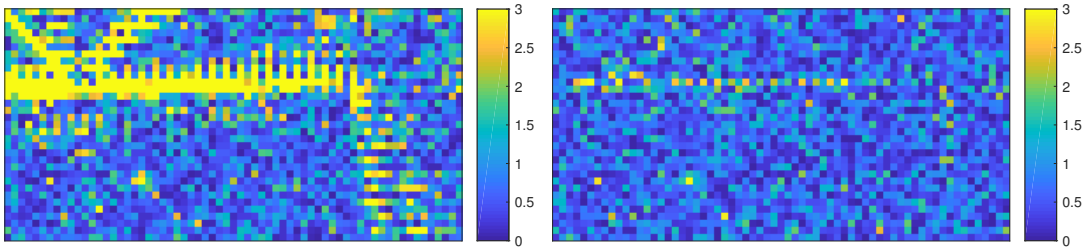
Figure 10.15: Comparison of the reconstructed images of the two orders for $\lambda = 2$ and $\sigma_n = 0$ with $\sigma_s = 0.7$. The sharpening is performed. A zipper effect is visible around the edges of the results of $N_{\text{KR}} = 0$. As seen in Section 10.3.2, this is introduced by the sharpening when the data repartition is not uniform or not dense enough, which is the case here.



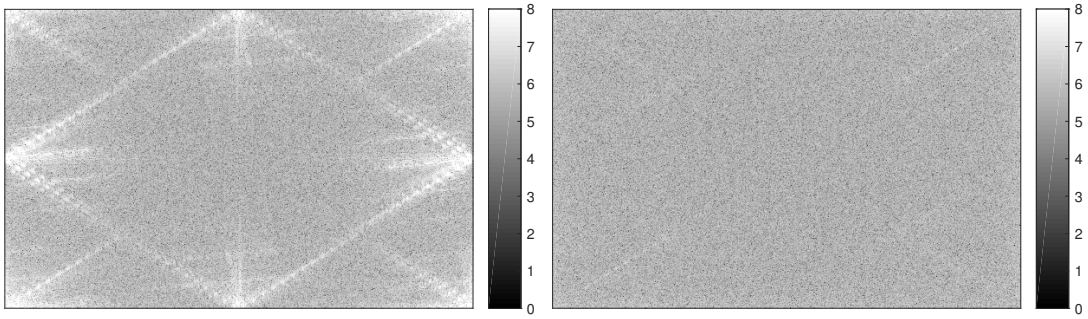
(a) Order 0 (b) Order 2



(c) Spectrum of (a) (d) Spectrum of (b)

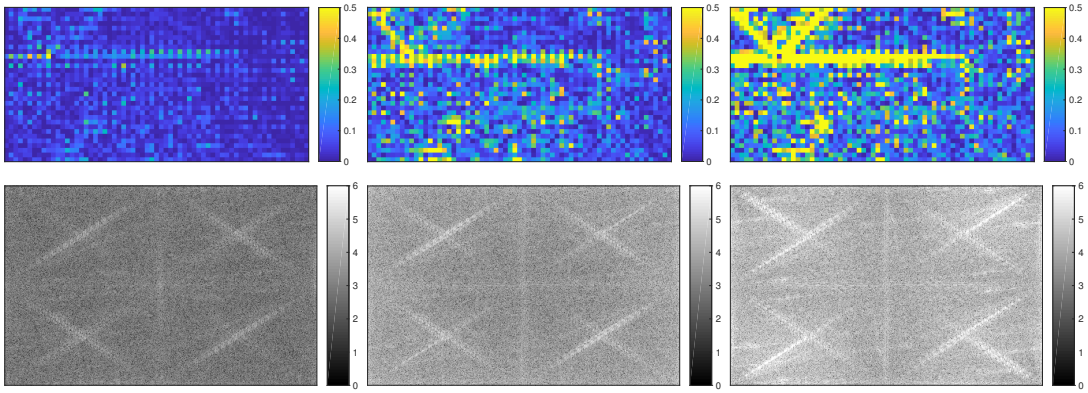


(e) Residual for (a) ($\mathcal{E}_{\text{ref}} = 1.67$) (f) Residual for (b) ($\mathcal{E}_{\text{ref}} = 0.81$)

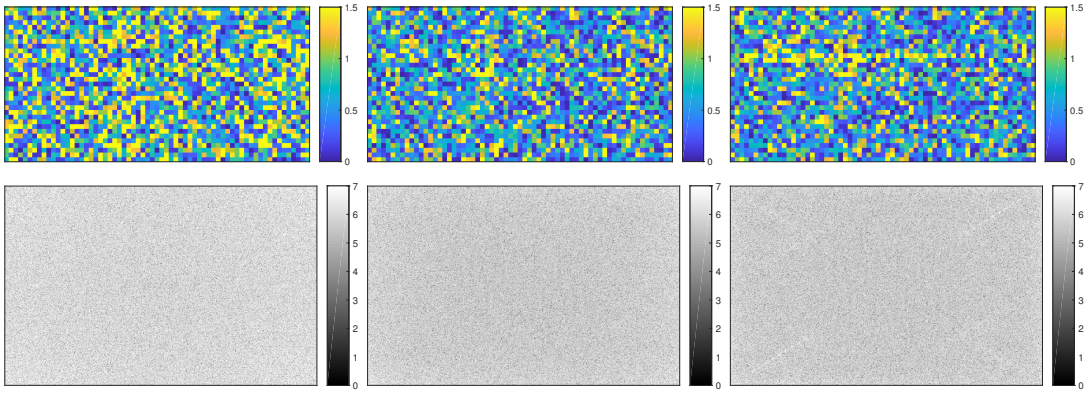


(g) Spectrum of (e) (h) Spectrum of (f)

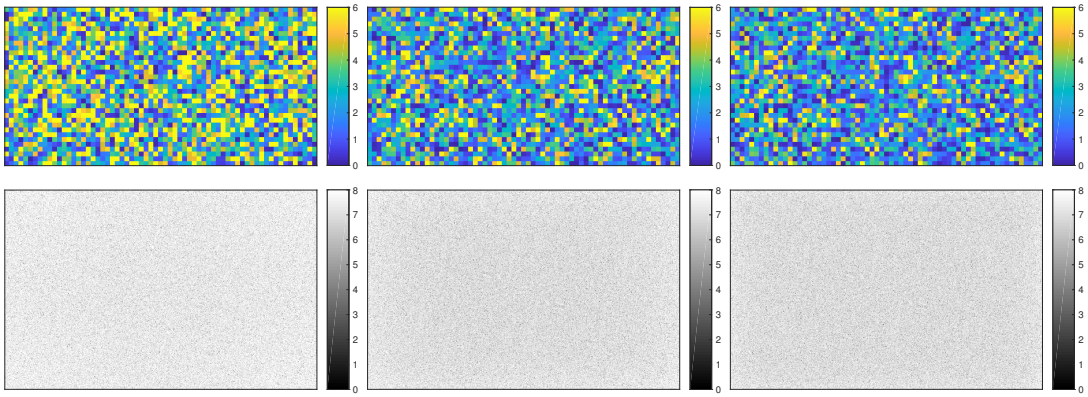
Figure 10.16: Comparison of the reconstructed images of the two orders for $\lambda = 2$ and $\sigma_n = 5$ with $\sigma_s = 0.7$. The sharpening is performed. A zipper effect is visible around the edges of the results of $N_{\text{KR}} = 0$. As seen in Section 10.3.2, this is introduced by the sharpening when the data repartition is not uniform or not dense enough, which is the case here.



(a) $\sigma_n = 0$ (0.0522,0.1325,0.2759)



(b) $\sigma_n = 5$ (1.13,0.84,0.81)



(c) $\sigma_n = 20$ (4.52,3.33,3.07)

Figure 10.17: Comparison of the residuals for $\sigma_s \in \{0.3, 0.5, 0.7\}$ (from left to right) when $\lambda = 2$ and $N_{KR} = 2$. The sharpening is performed. The values between parentheses are the corresponding error $\mathcal{E}_{ref}$. Without noise small values of σ_s provides the best results because it corresponds to a less important sharpening. For noisy data the best results are obtained for $\sigma_s = 0.7$.

10.3.4 Comparison with Burst Denoising and the ACT

In this section we compare our method, based on classical kernel regression, with the burst denoising and the ACT methods. We take for the order N_{KR} and the scale σ_{s} the values chosen in Section 10.3.3.

Evolution with the number of images. First, we consider the evolution of the error $\mathcal{E}_{\mathrm{ref}}$ with the number n of images combined. This number n varies from 50 to $N_{\mathrm{im}} = 200$ (by step of 10). The results for noise levels $\sigma_{\mathrm{n}} \in \{0, 5, 20, 50\}$ are presented in Figure 10.18 for $\lambda = 1$ and in Figure 10.19 for $\lambda = 2$. The ACT method clearly provides the best results in every situation. For noisy data all the curves are similar to the ideal denoising one, which is $n \mapsto \frac{\lambda \sigma_{\mathrm{n}}}{\sqrt{n}}$, but the ACT and burst denoising curves are closer.

The error for the classical kernel regression decreases with the number of images, even without noise, since having more samples allows for a better fit and sharpening. For $\lambda = 1$ the burst denoising performs well but without noise it is beaten by our method for a high number of images. As already seen, using $N_{\mathrm{KR}} = 0$ for $\lambda = 2$ leads to high errors. However it does better than $N_{\mathrm{KR}} = 2$ for $\sigma_{\mathrm{n}} = 50$ (in terms of $\mathcal{E}_{\mathrm{ref}}$ but artifacts are introduced).

Evolution with the noise level. The evolution of the error $\mathcal{E}_{\mathrm{ref}}$ for $N_{\mathrm{im}} = 200$ images with the noise level σ_{n} is presented in Figure 10.20. The results confirm the analysis made previously. The ACT and burst denoising have the best results but for all the methods the error is close to the ideal residual noise. For $\lambda = 2$ the bad results of our method with $N_{\mathrm{KR}} = 0$ are clearly noticible.

Residual comparison. For $\lambda = 1$, the residuals of the three methods are compared in Figure 10.21. For $\lambda = 2$, the residuals of our method and of the ACT are compared in Figure 10.22. The noise levels considered are $\sigma_{\mathrm{n}} \in \{0, 5, 20\}$. In every situation the ACT method leads to the best result. The burst denoising (for $\lambda = 1$) provides similar results. Without noise, the spectrum of the burst denoising is similar to the spectrum of the residual during the consistency measurement evaluation (see Figure 6.4(g)) and is mostly composed of high-frequency content. For noisy data, with and without sub-sampling, the residuals of all methods are mostly composed of white noise.

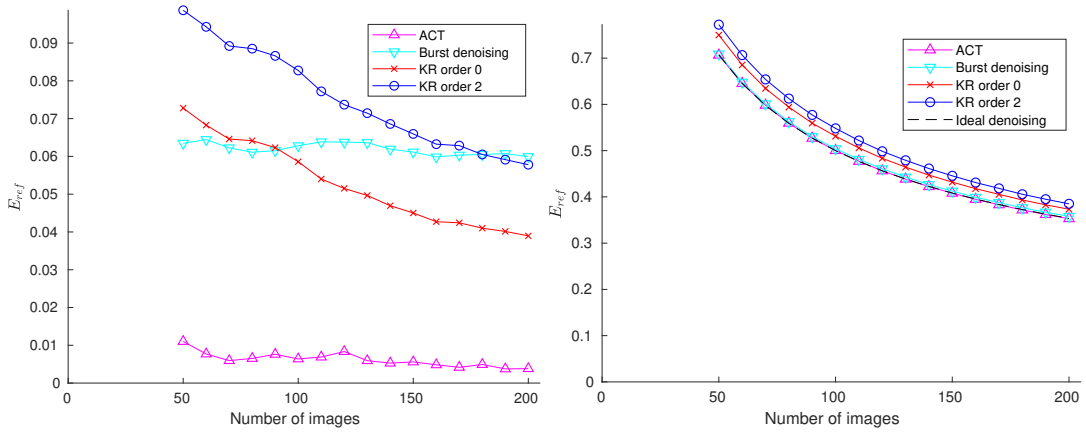
Computation time. The evolution of the computation time with the number of images n for $\lambda = 1$ is shown in Figure 10.23. For all the methods the computation time increases linearly with the number of images. The ACT method is the most costly method and is followed by the burst denoising. The gap between the methods increases with the number of images. For instance the factor between the ACT and our method (with $N_{\mathrm{KR}} = 0$) varies from 5 to 10. Note that the computation time for the $N_{\mathrm{im}} = 200$ images using our method is smaller than the one of the ACT for 50 images.

Memory requirement. The maximum memory usages of the methods in kilobytes (kB) are the following.

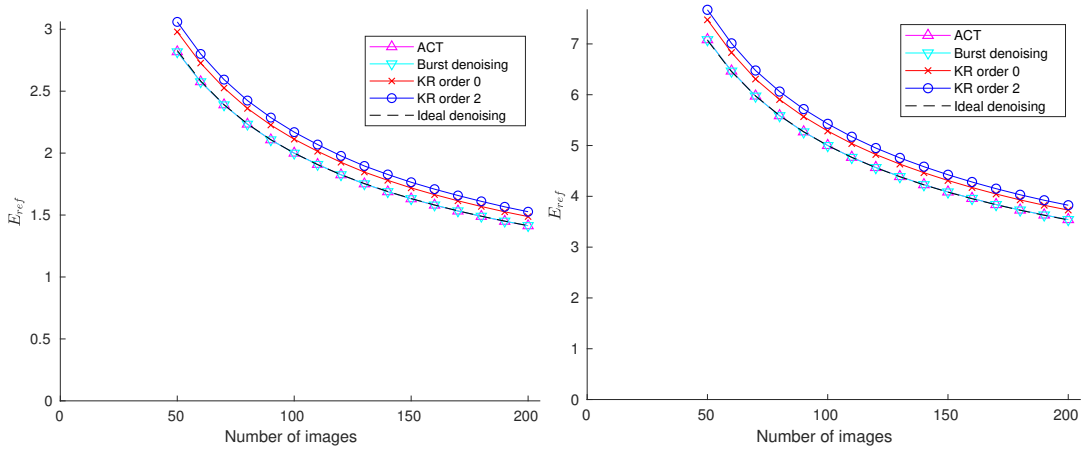
	Burst denoising	ACT	Our method (KR)
$\lambda = 1$	82608	4101764	**17428**
$\lambda = 2$	ND	1185360	**95832**

Compared to the ACT, our method uses 235 times less memory for $\lambda = 1$ and 12 times less for $\lambda = 2$. Besides the different sub-sampling factor, the difference between the two factors is due to the use of a different order N_{KR}. Indeed, using $N_{\mathrm{KR}} = 2$ requires to store $21/2 \simeq 10$ times more system coefficients than for $N_{\mathrm{KR}} = 0$.

For $\lambda = 1$, the burst denoising method is in between since the images are processed sequentially but with a costly interpolation method.

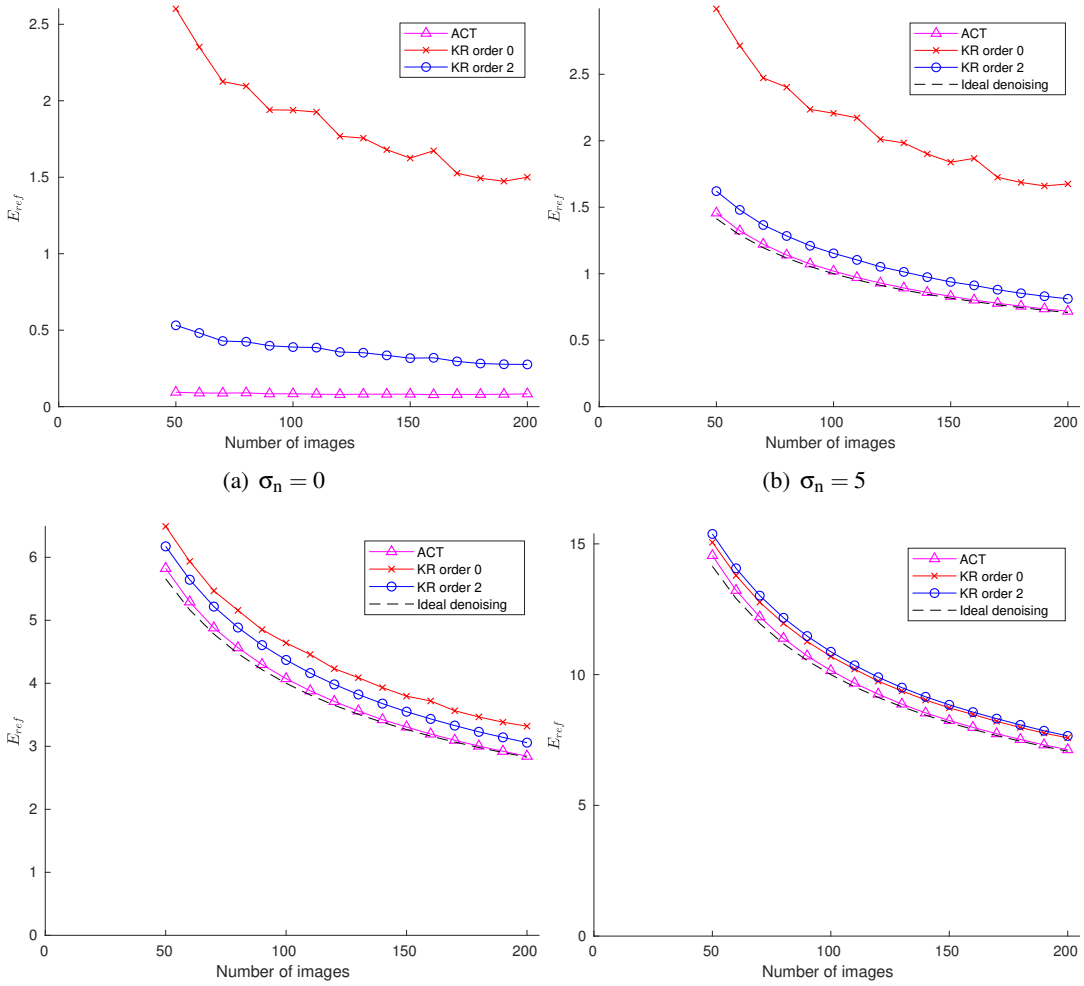


(a) $\sigma_n = 0$ (b) $\sigma_n = 5$



(c) $\sigma_n = 20$ (d) $\sigma_n = 50$

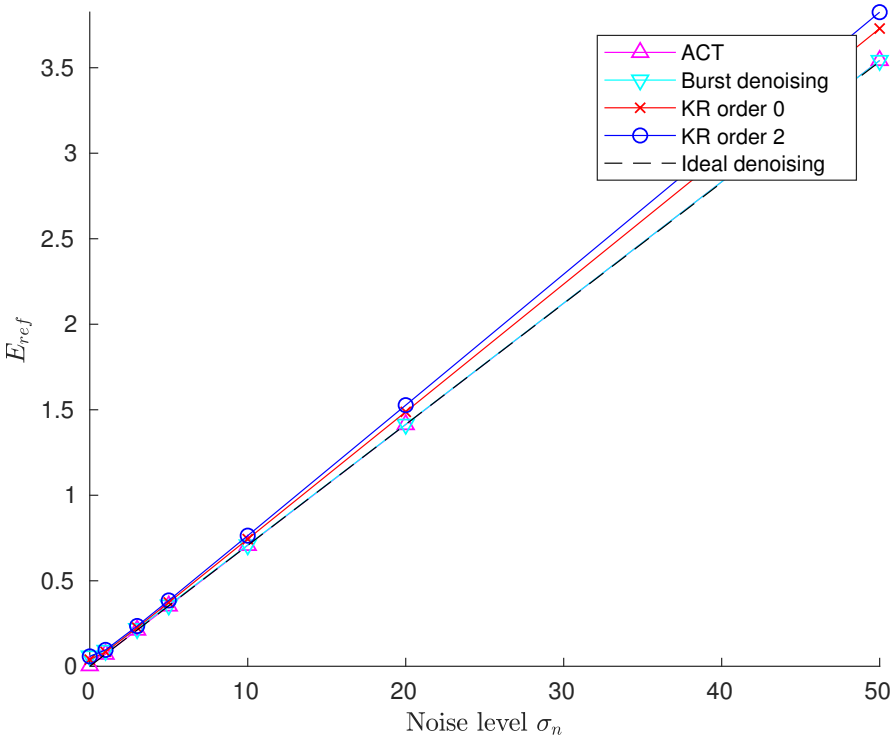
Figure 10.18: Evolution of the error $\mathcal{E}_{\text{ref}}$ with the number n of images combined for $\lambda = 1$. The scale $\sigma_s = 0.7$ is used for both orders. The ACT method clearly provides the best results in every situation. For noisy data all the curves are similar to the ideal denoising one, which is $n \mapsto \frac{\sigma_n}{\sqrt{n}}$, but the ACT and burst denoising curves are closer. The error for the classical kernel regression decreases with the number of images, even without noise, since having more samples allows for a better fit and sharpening. The burst denoising performs well but it is beaten by our method for a high number of images without noise.



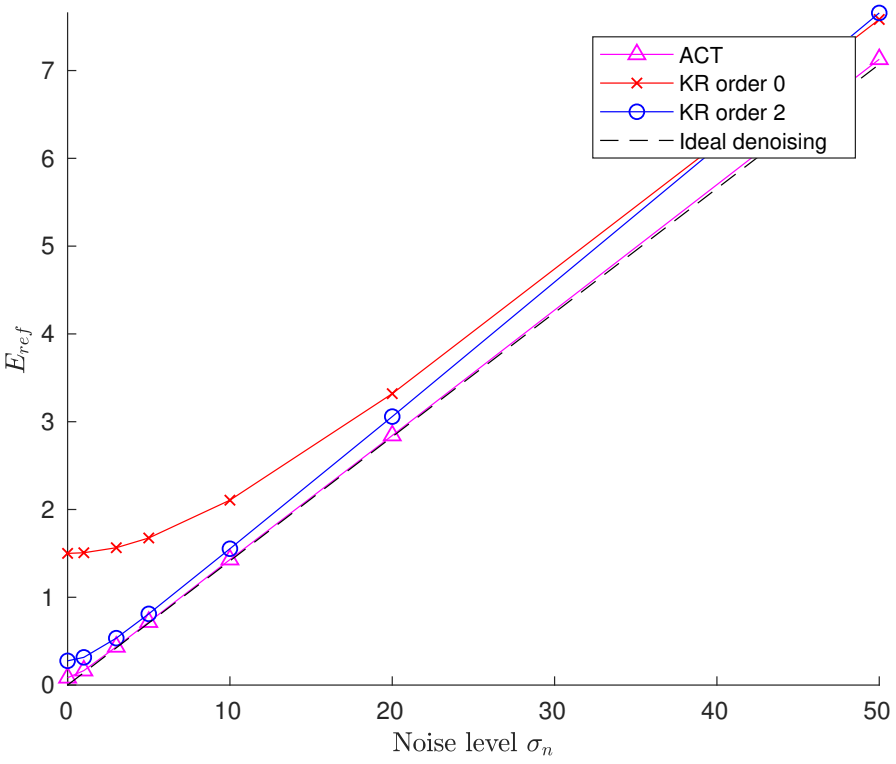
(a) $\sigma_n = 0$ (b) $\sigma_n = 5$

(c) $\sigma_n = 20$ (d) $\sigma_n = 50$

Figure 10.19: Evolution of the error $\mathcal{E}_{\text{ref}}$ with the number n of images combined for $\lambda = 2$. The scale $\sigma_s = 0.7$ is used for both orders. The ACT method clearly provides the best results in every situation. For noisy data all the curves are similar to the ideal denoising one, which is $n \mapsto \frac{2\sigma_n}{\sqrt{n}}$. The error for the classical kernel regression decreases with the number of images, even without noise, since having more samples allows for a better fit and sharpening. As already seen, using $N_{\text{KR}} = 0$ for $\lambda = 2$ leads to high errors. However this choice performs better than $N_{\text{KR}} = 2$ for $\sigma_n = 50$ (in terms of $\mathcal{E}_{\text{ref}}$, but artifacts are introduced).

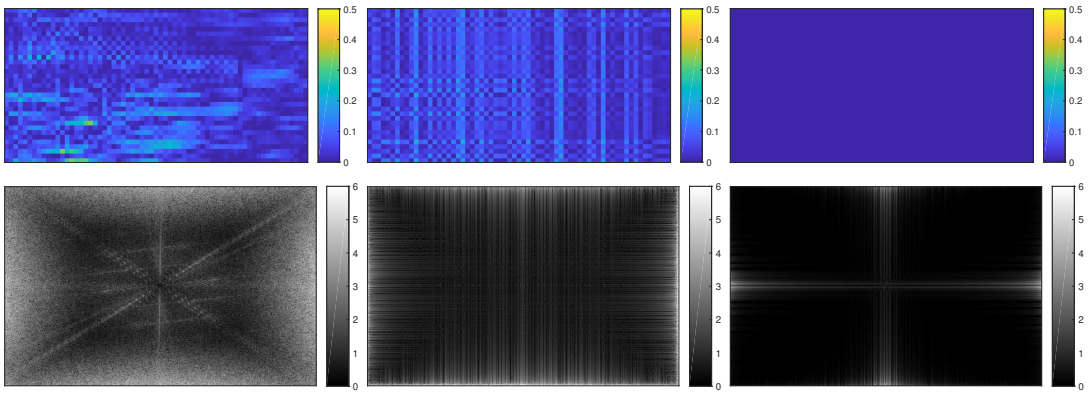


(a) Without super-resolution $\lambda = 1$

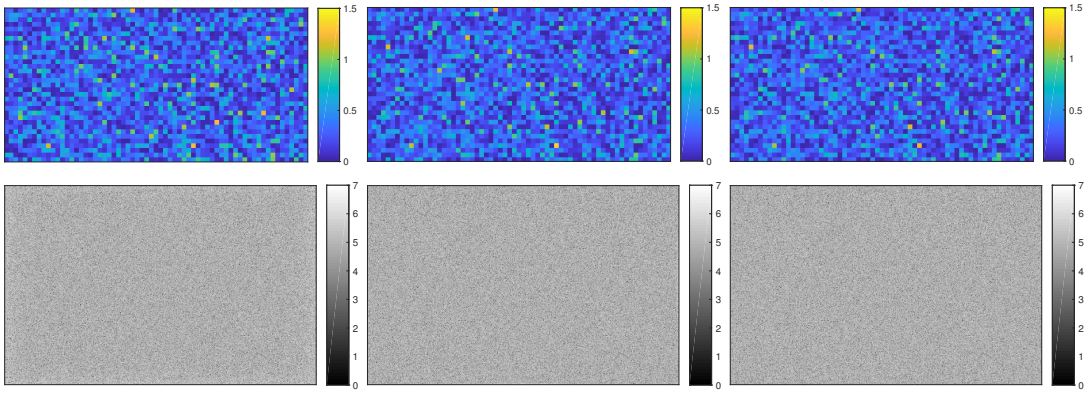


(b) With super-resolution $\lambda = 2$

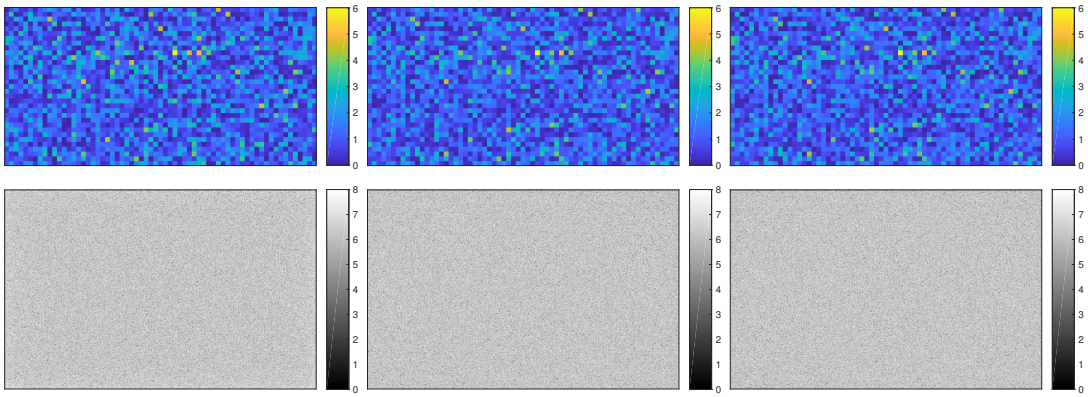
Figure 10.20: Evolution of the error $\mathcal{E}_{\text{ref}}$ for $N_{\text{im}} = 200$ images with the noise level σ_{n}. The results confirm the analysis made previously. The scale $\sigma_{\text{s}} = 0.7$ is used for both orders. The ACT and burst denoising have the best results but for all the methods the error is close to the ideal residual noise. For $\lambda = 2$ the bad results of our method with $N_{\text{KR}} = 0$ are clearly noticible.



(a) $\sigma_n = 0$ (0.0390,0.0599,0.0038)

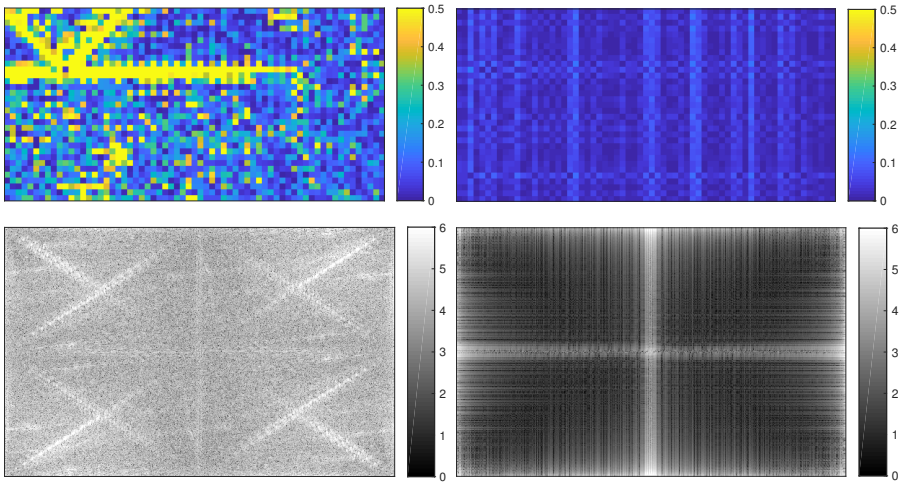


(b) $\sigma_n = 5$ (0.3744,0.3596,0.3544)

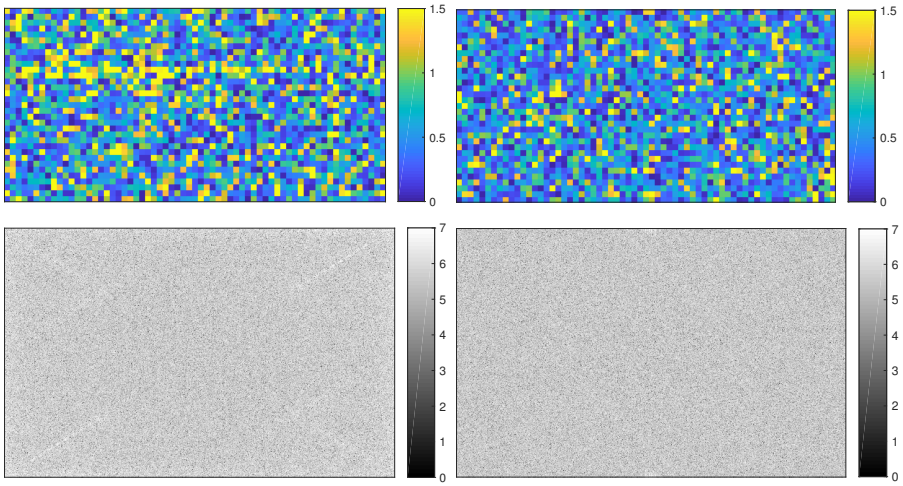


(c) $\sigma_n = 20$ (1.4887,1.4154,1.4164)

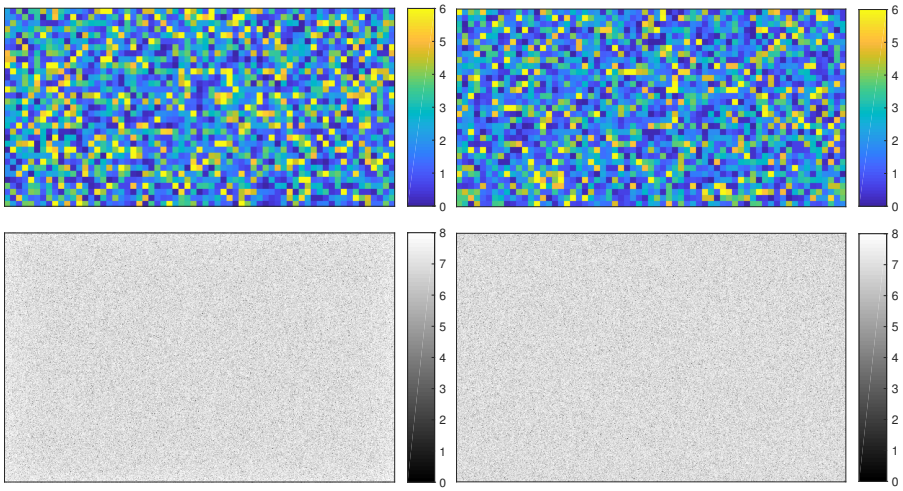
Figure 10.21: Residuals of our method, burst denoising and ACT (from left to right) for $\lambda = 1$. Our method uses $N_{KR} = 0$ and $\sigma_s = 0.7$. The values between parentheses are the corresponding error $\mathcal{E}_{ref}$. The ACT method yields the best result. The burst denoising provides similar results. Without noise, the spectrum of the burst denoising is similar to the spectrum of the residual during the consistency measurement evaluation (see Figure 6.4(g)) and is mostly composed of high-frequency content. For noisy data, the residuals of all methods are mostly composed of white noise.



(a) $\sigma_n = 0$ (0.2759,0.0835)



(b) $\sigma_n = 5$ (0.8101,0.7164)



(c) $\sigma_n = 20$ (3.0718,2.8502)

Figure 10.22: Residuals of our method (left) and ACT (right) for $\lambda = 2$. Our method uses $N_{KR} = 2$ and $\sigma_s = 0.7$. The values between parentheses are the corresponding error $\mathcal{E}_{ref}$. The ACT method yields the best result. For noisy data, the residuals of both methods are mostly composed of white noise.

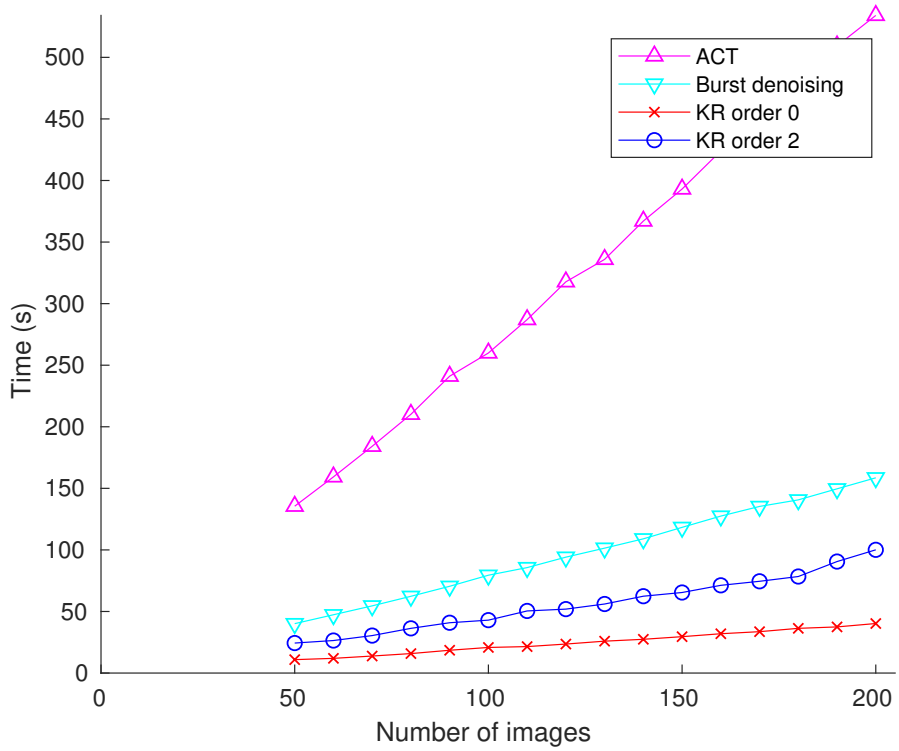


Figure 10.23: Evolution of the computation time with the number of images n for $\lambda = 1$. For all the methods the computation time increases linearly with the number of images. The ACT method is the most costly method and is followed by the burst denoising. The gap between the methods increases with the number of images. For instance the factor between the ACT and our method (with $N_{\text{KR}} = 0$) varies from 5 to 10. Note that the computation time for the $N_{\text{im}} = 200$ images using our method is smaller than the one of the ACT for 50 images.

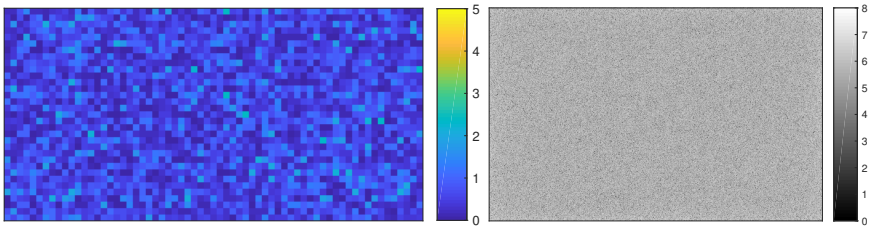
Conclusion. The best results are obtained using the ACT method but it is computationally and memory greedy. The burst denoising method provides similar results while being faster and not memory greedy (since images are processed sequentially). But it is not applicable when the input images are aliased (for instance for $\lambda > 1$). Our method shows similar performance for a high enough number of images. But is much faster and only requires a small amount of memory. The denoising factor is close to the ideal one and it is open to super-resolution.

10.3.5 Comparison between Two Reconstructed Images

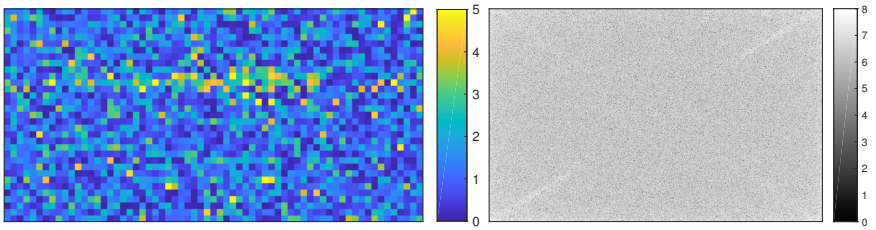
For real data, in Section 10.4, the reference image is not available. The error $\mathcal{E}_{\text{set}}$, where two images built from separate sets are compared, has to be considered. Therefore, in this section we consider the same process in order to have a baseline for real data. The scale $\sigma_s = \frac{\sqrt{2}}{2}$ is used.

Evolution with the number of images. First, we consider the evolution of the error $\mathcal{E}_{\text{set}}$ with the number n of images used in each set. This number n varies from 50 to $N_{\text{im}}/2 = 100$. The results for noise levels $\sigma_n \in \{0,5,20,50\}$ are presented in Figure 10.25 for $\lambda = 1$ and in Figure 10.26 for $\lambda = 2$. For the same reason as for $\mathcal{E}_{\text{ref}}$, the error $\mathcal{E}_{\text{set}}$ decreases with the number of images even though there is no noise. For noisy data the curves for both orders are similar to the ideal denoising one, which is $n \mapsto \frac{\sqrt{2}\lambda\sigma_n}{\sqrt{n}}$. For $\lambda = 1$ using $N_{\text{KR}} = 0$ gives the smallest error while for $\lambda = 2$ it is $N_{\text{KR}} = 2$ (except for high noise $\sigma_n = 50$).

Difference images for $\sigma_n = 5$. The difference images between the two reconstructed images using 100 images for $\sigma_n = 5$ and $\lambda \in \{1,2\}$ are shown in Figure 10.24. We used $N_{\text{KR}} = 0$ for $\lambda = 1$ and $N_{\text{KR}} = 2$ for $\lambda = 2$. The differences are mostly composed of residual white noise. For $\lambda = 2$ the structure of the image can still be noticed.

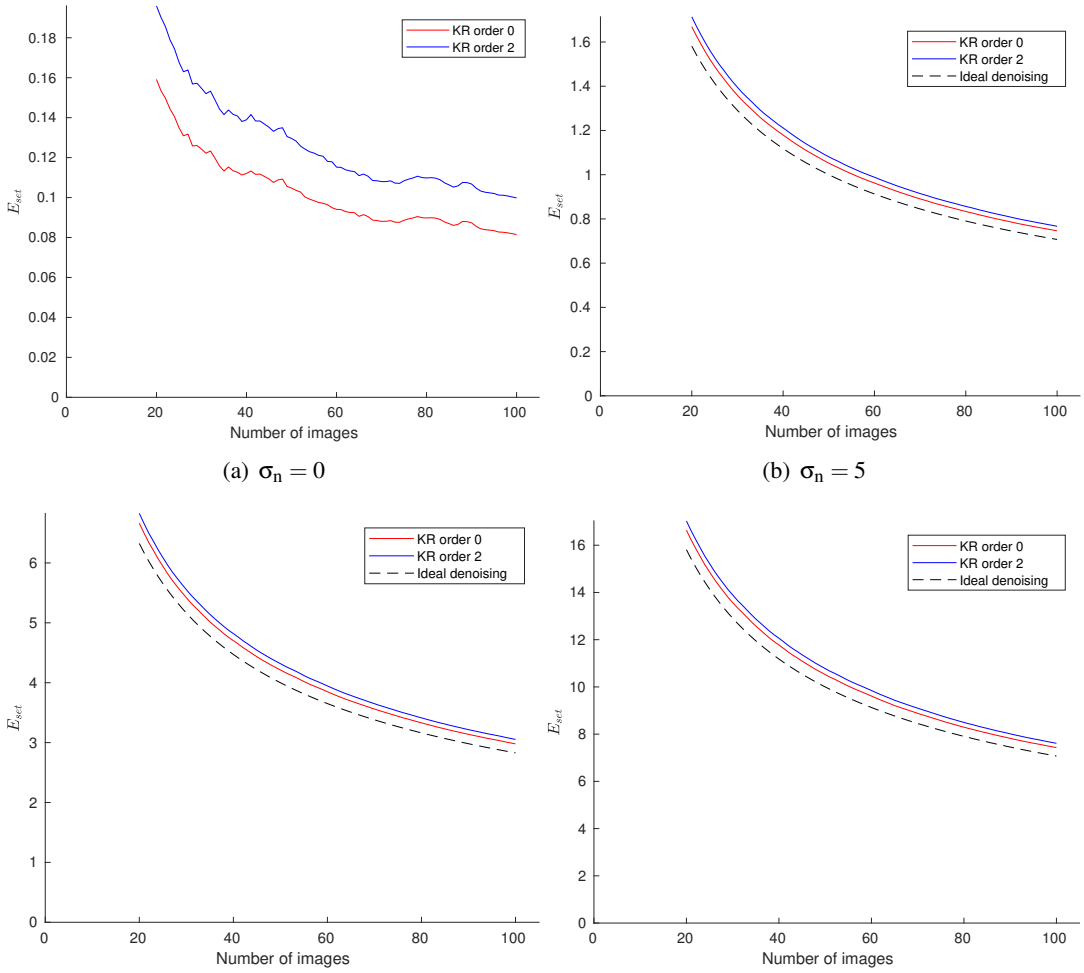


(a) Without super-resolution ($\lambda = 1$): $\mathcal{E}_{\text{set}} = 0.75$



(b) With super-resolution ($\lambda = 2$): $\mathcal{E}_{\text{set}} = 1.70$

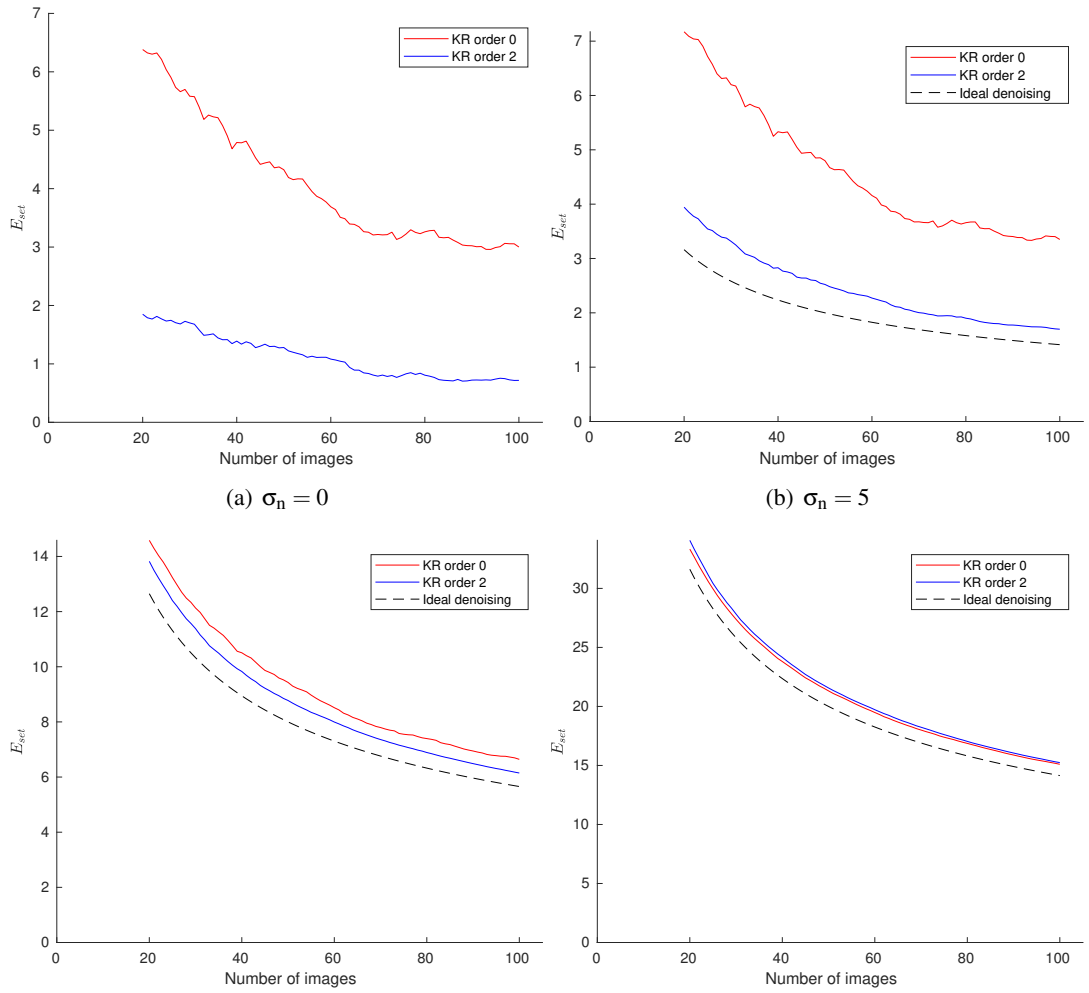
Figure 10.24: Difference images between the two reconstructed images using 100 images for $\sigma_n = 5$. We used with $N_{\text{KR}} = 0$ for $\lambda = 1$ and $N_{\text{KR}} = 2$ for $\lambda = 2$ (with $\sigma_s = \frac{\sqrt{2}}{2}$). The differences are mostly composed of residual white noise. For $\lambda = 2$ the structure of the image can still be noticed.



(a) $\sigma_n = 0$ (b) $\sigma_n = 5$

(c) $\sigma_n = 20$ (d) $\sigma_n = 50$

Figure 10.25: Evolution of the error $\mathcal{E}_{\text{set}}$ with the number n of images used in each set for $\lambda = 1$. The scale $\sigma_s = \frac{\sqrt{2}}{2}$ is used. For the same reason as for $\mathcal{E}_{\text{ref}}$, the error $\mathcal{E}_{\text{set}}$ decreases with the number of images even though there is no noise. For noisy data the curves for both orders are similar to the ideal denoising one, which is $n \mapsto \frac{\sqrt{2}\sigma_n}{\sqrt{n}}$. Using $N_{\text{KR}} = 0$ gives the smallest error.



(a) $\sigma_n = 0$ (b) $\sigma_n = 5$

(c) $\sigma_n = 20$ (d) $\sigma_n = 50$

Figure 10.26: Evolution of the error $\mathcal{E}_{\text{set}}$ with the number n of images used in each set for $\lambda = 2$. The scale $\sigma_s = \frac{\sqrt{2}}{2}$ is used. For the same reason as for $\mathcal{E}_{\text{ref}}$, the error $\mathcal{E}_{\text{set}}$ decreases with the number of images even though there is no noise. For noisy data the curves for both orders are similar to the ideal denoising one, which is $n \mapsto \frac{2\sqrt{2}\sigma_n}{\sqrt{n}}$. Using $N_{\text{KR}} = 2$ gives the smallest error (except for high noise $\sigma_n = 50$).

10.4 Experiments on Real Data

In this section, we evaluate on real data the performance of our image fusion algorithm, described in Algorithm 9.1. First, the experimental setup is described in Section 10.4.1. Then, the results are analyzed in Section 10.4.2.

10.4.1 Experimental Setup

The considered dataset is composed of 201 color images of a static outdoor scene. They were acquired using an Olympus E-M5 Mark II camera (with a 17.0mm lens) that was set in sequential mode. We used a shutter speed of 1/100s, an aperture of f/5.6 and ISO 200. The camera was fixed on a tripod but the internal stabilization mode was desactivated. Thus, small motions/vibrations of the camera were allowed and super-resolution can be considered. Finally, the images differ because of small motions of the camera, noise, JPEG compression [132], quantization (8-bit), and possibly small illumination variations. Note that the corresponding RAW images are used in Chapter 11.

Because of the ACT memory limitations, we only keep the central part of the images of size 512×512, which is by the way distortion-free. We arbitrarily set the first image as the reference image (see Figure 10.28(a)). Small illumination variations are handled by a mean equalization of the images to the reference one. As for the synthetic data the reference image is used for the registration but not for the combination. The irregularly data fitting is performed on the remaining images (here at most $N_{\text{im}} = 200$).

Details of the reference image and of another image of the sequence can be seen in Figure 10.28(b)-(c). The difference between these images after registration and the corresponding spectrum are displayed in Figure 10.28(d)-(e). The difference is mostly composed of noise since the image structure is hardly noticible. However it does not seem to be white noise (probably because of the JPEG compression). The root mean square difference (RMSD) is around 4.30, which corresponds (assuming a perfect registration and white noise) to a standard deviation $\sigma_{\text{n}} \leq 3$.

In our experiments we consider the zoom factors $\lambda = 1$ (without super-resolution) and $\lambda = 2$. Note that, contrary to the synthetic case, the output image using $\lambda = 2$ is of size 1024×1024. As recommended in Section 10.3.3, our method uses $N_{\text{KR}} = 0$ for $\lambda = 1$ and $N_{\text{KR}} = 2$ for $\lambda = 2$, with the scale $\sigma_{\text{s}} = \frac{\sqrt{2}}{2}$ in both cases.

10.4.2 Results

The registration step for the $N_{\text{im}} = 200$ images is done in 128s, i.e., a single transformation is estimated in approximately 0.6s. The estimated data repartitions with and without super-resolution are shown in Figure 10.27. As the transformations between the images are locally similar to translations, the data repartition seems to have a periodicity of λ along each direction.

Computation time and memory requirement. The computation times and maximal memory requirements of the methods are the following.

		Burst denoising	ACT	Our method (KR)
$\lambda = 1$	Time (s)	442	1784	**135**
	Max memory (kB)	175468	5562344	**82192**
$\lambda = 2$	Time (s)	ND	1942	**444**
	Max memory (kB)	ND	6224528	**1200492**

Our method is significantly more efficient and requires less memory.

Comparison with the reference image. For $\lambda = 1$, the result of our method is compared to the reference image in Figure 10.29. The output image has less high frequency content than the reference image thanks to the denoising. Note that no blur is introduced. The difference image is mostly composed of noise but the structure of the image can be noticed. The RMSD is around 3.11, which is (assuming $\sigma_n = 3$) close to the expected value $\sigma_n\sqrt{1+1/200} \simeq 3$.

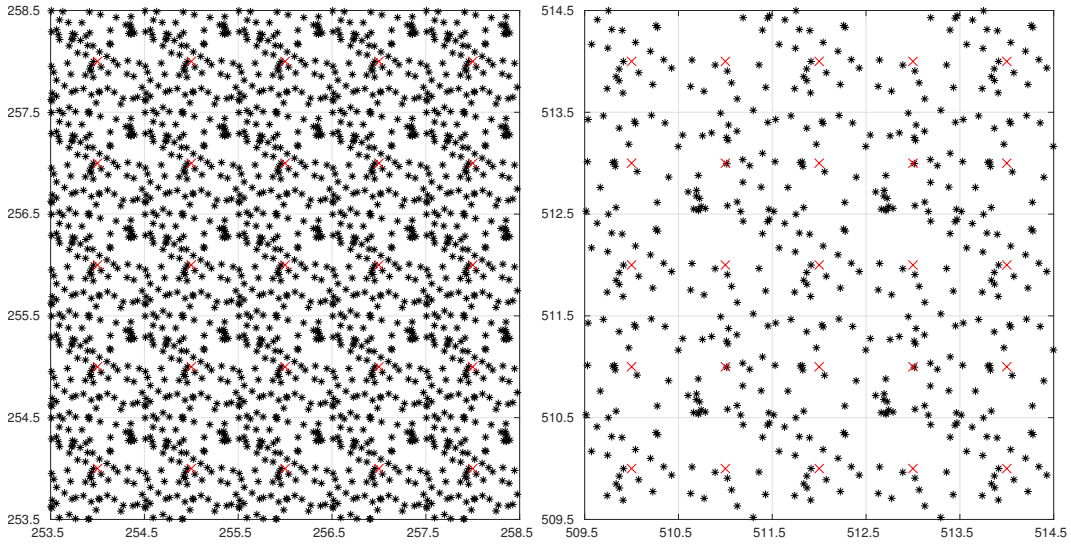
Comparison between reconstructed images. The difference between two reconstructed images from separate sets of 100 images is shown in Figure 10.30. For $\lambda = 1$ the difference is mostly composed of noise while for $\lambda = 2$ the structure of the image can be noticed.

The evolution of the error $\mathcal{E}_{\text{set}}$ with the number n of images used in each set is shown in Figure 10.31. The error tends to decrease with n but, contrarily to synthetic data, it may increase temporarily when adding images. However it has to be noted that an increase of the error does not necessarily mean a deterioration of the results. Anyway the error $\mathcal{E}_{\text{set}}$ for both λs is approximately divided by 2 between 20 and 100 images instead of $\sqrt{100/20} = \sqrt{5} \simeq 2.2$ in the ideal denoising case.

Comparison with other methods. Our method is compared to the ACT and the burst denoising methods in Figure 10.32. The results are really close and cannot be distinguished to the naked eye. Therefore the difference images are shown. They are not composed of noise. For $\lambda = 2$ the difference is more important in particular around the edges (where zipper structures can be seen).

Conclusion. Our method performs well on real data since it provides similar results as slower and memory greedy methods. The residual noise decreases as expected. The output image is denoised and no blur is introduced. In addition, super-resolution can be performed.

The results of the three methods are not as good as for synthetic data because of demosaicking, JPEG compression and 8-bit quantization. These uncontrolled processings deteriorate both the registration and the irregularly data fitting part. This is why we propose in Chapter 11 an image formation algorithm taking RAW images as input.



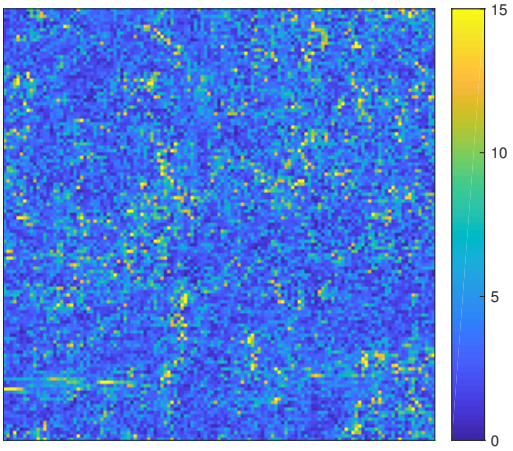
(a) $\lambda = 1$ (b) $\lambda = 2$

Figure 10.27: Estimated spatial repartition of the irregularly sampled data, around the center of the reference image, for real data (Section 10.4). The factor λ corresponds to the zoom factor used in the image formation algorithm. As the transformations between the images are locally similar to translations, the data repartition seems to have a periodicity of λ along each direction.

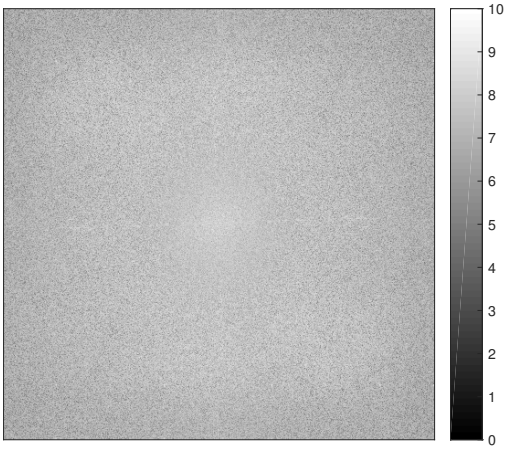
(a) Reference image

(b) Details of (a)

(c) Details of another image



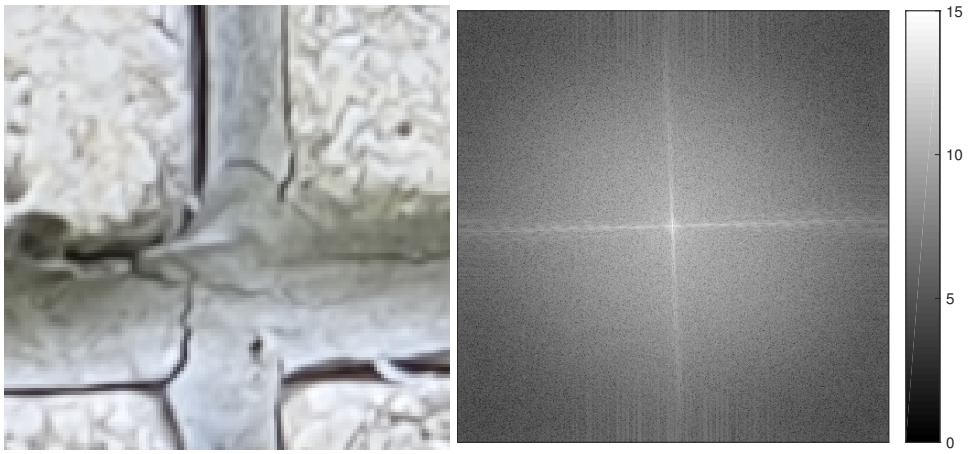
(d) Difference after registration (RMSD=4.30)



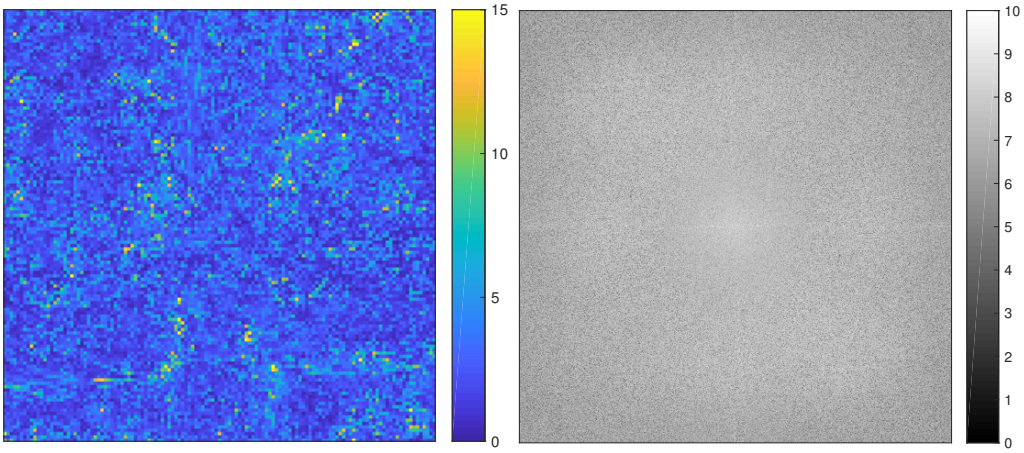
(e) Spectrum of (d)

Figure 10.28: Real data used in Section 10.4. The reference image in (a) is a color image of size 512×512. In (b) and (c) details in a zone of size 128×128 are shown. The difference in (d) is mostly composed of noise since the image structure is hardly noticible. However it does not seem to be white noise (probably because of the JPEG compression). The RMSD is around 4.30, which corresponds (assuming a perfect registration and white noise) to a standard deviation $\sigma_n \leq 3$.

(a) Reference (b) Spectrum of (a)

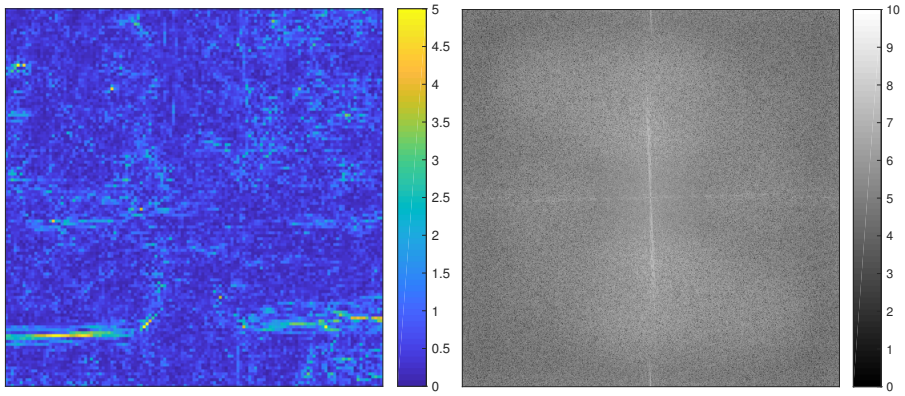


(c) Output (d) Spectrum of (c)

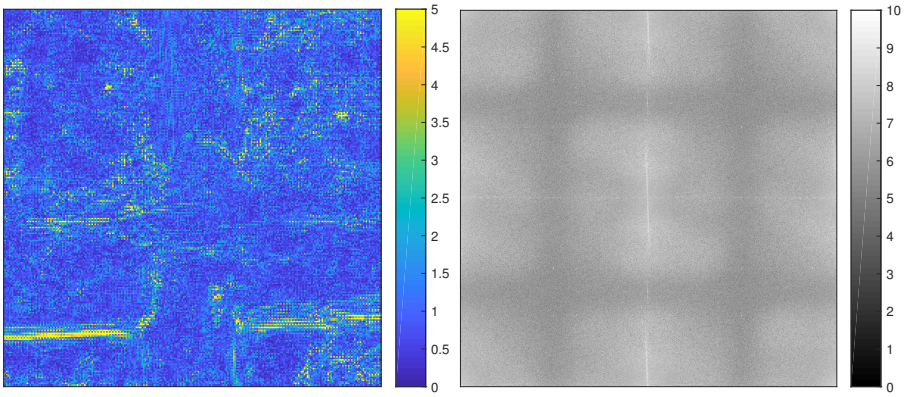


(e) Difference between (a) and (c) (RMSD=3.11) (f) Spectrum of (e)

Figure 10.29: Comparison to reference image for our method ($\lambda = 1$, $N_{\text{im}} = 200$). The output image (c) has less high frequency content than the reference image (a) thanks to the denoising. Note that no blur is introduced. The difference image (e) is mostly composed of noise but the structure of the image can be noticed (probably because of the JPEG compression). The RMSD is around 3.11, which is (assuming $\sigma_n = 3$) close to the expected value $\sigma_n\sqrt{1+1/200} \simeq 3$.



(a) Difference image $\lambda = 1$ ($\mathcal{E}_{\text{set}} = 0.63$) (b) Spectrum of (a)



(c) Difference image $\lambda = 2$ ($\mathcal{E}_{\text{set}} = 1.27$) (d) Spectrum of (c)

Figure 10.30: Comparison between two reconstructed images from separate sets of $N_{\text{im}}/2 = 100$ images. For $\lambda = 1$ the difference is mostly composed of noise while for $\lambda = 2$ the structure of the image can be noticed.

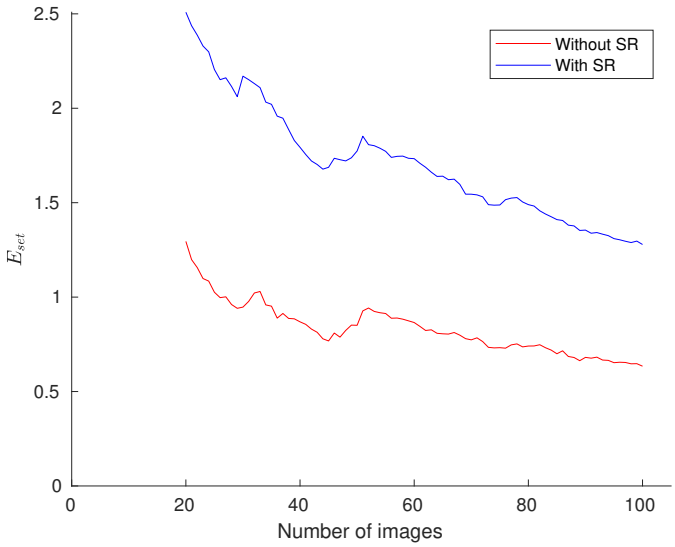
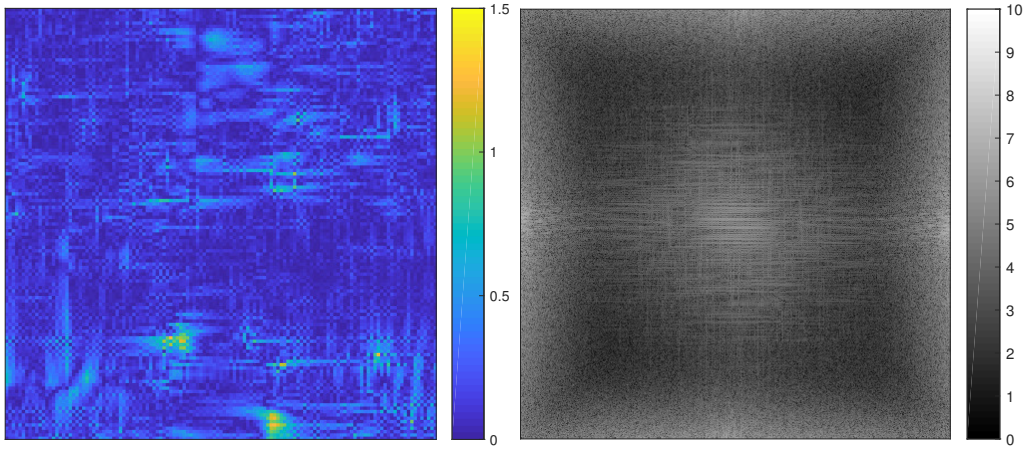
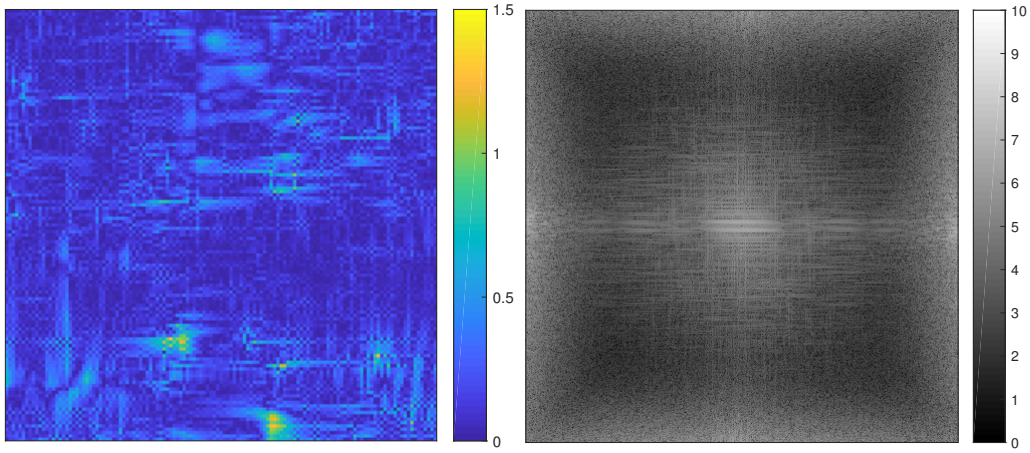


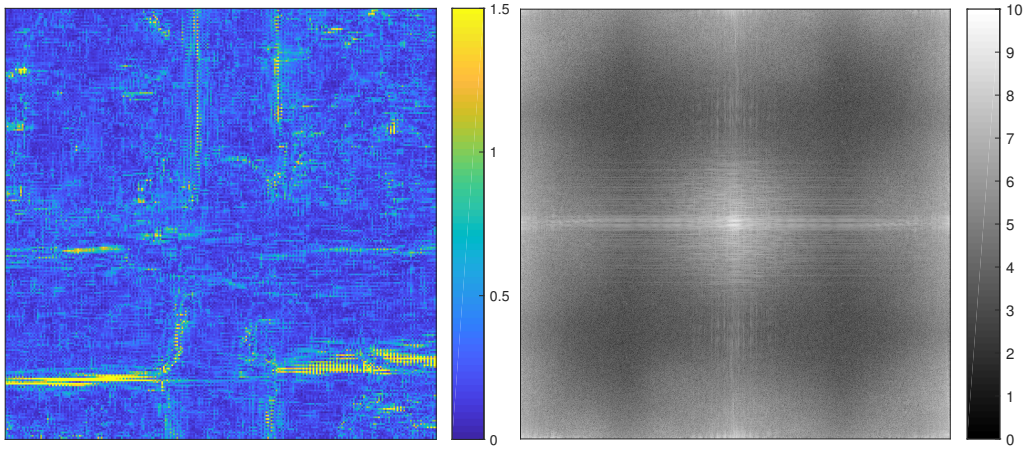
Figure 10.31: Evolution of the error $\mathcal{E}_{\text{set}}$ with the number n of images used in each set. The error tends to decrease with n but, contrarily to synthetic data, it may increase when adding images. However it has to be noted that an increase of the error does not necessarily mean a deterioration of the results. Anyway the error $\mathcal{E}_{\text{set}}$ for both λ is approximately divided by 2 between 20 and 100 images instead of $\sqrt{100/20} = \sqrt{5} \simeq 2.2$ in the ideal denoising case.



(a) Burst denoising (RMSD = 0.17)



(b) ACT for $\lambda = 1$ (RMSD = 0.19)



(c) ACT for $\lambda = 2$ (RMSD = 0.48)

Figure 10.32: Comparison to the ACT and the burst denoising methods. The results are really close and cannot be distinguished to the naked eye. Therefore the difference images are shown. They are not composed of noise. For $\lambda = 2$ the difference is more important in particular around the edges (where zipper structures can be seen).

10.5 Conclusion

In this chapter we evaluated experimentally the performance of our image fusion algorithm, described in Algorithm 9.1, on synthetic and real data. The experiments on synthetic data allowed us to highlight the interest of the sharpening step and to find the best configuration for the order and the scale. The choice of the order mostly depends on the data spatial repartition. Assuming a uniform sample repartition, we have been led to recommend the use of order 0 without super-resolution and order 2 with super-resolution. The optimal kernel scale turns out to be around 0.7. We showed that for a large amount of data (synthetic or real) our image fusion algorithm provides results nearly as good as those obtained with slower and memory greedy methods. The residual noise decreases as expected and it is able to perform super-resolution.

From the experiments on real data we saw that the performance of image fusion methods is limited by uncontrolled processes (demosaicking, JPEG compression, 8-bit quantization). This is why we propose in Chapter 11 an image formation algorithm taking RAW images as input.

Chapter 11

Image Formation from a Large Sequence of RAW images

Abstract

In this chapter, we propose an image formation method from a large sequence of RAW images. This is an adapted version of our image fusion method, described in Chapter 9, for RAW images and it follows the work of [13]. The RAW images are first preprocessed to transform the noise curve and to adjust the histograms. Then the two-step registration method for mosaicked images, introduced in Chapter 8, is used to align the images. As in Chapter 9, the images are combined by classical kernel regression of second order with Gaussian kernel. The process has two stages. The first one is an accumulation process where the images are processed sequentially. In the second stage, the blur introduced by the classical kernel regression is inverted by a sharpening filter. We show experimentally that, for a large number of RAW images, our image formation method provides, efficiently and with a low memory usage, a high-quality result. Contrarily to image fused from processed images (as in Chapter 10), the images formed from RAW images do not contain artifacts coming from the unknown image processing pipeline. Our method successfully performs super-resolution and the residual noise decreases as expected. We obtained results similar to those obtained by slower and memory greedy methods.

Contents

11.1 Introduction

A RAW image is an unprocessed (or minimally processed) image acquired by a camera. The CCD or CMOS captor transforms incoming light photons into voltage output to which is associated, after quantization, an intensity value [4]. Usually a 12-bit (4096 values) or 14-bit (16384 values) quantization is used. Contrarily to a JPEG image [132], which is quantized in 8-bit (256 values), there is no compression. As already discussed in Chapter 8, a RAW image is a mosaicked (or CFA) image. The color image provided by a commercial camera is the result of a generally unknown image processing pipeline [49]. Among others, gamma correction, demosaicking, denoising, deblurring, compression and quantization may have been applied.

Most multi-image algorithms take as input a sequence of such preprocessed images, which leads to artifacts as noticed in Chapter 10. Therefore, good methods should rather handle RAW images [35, 36, 37, 48, 61, 131]. Forming an image from a sequence of RAW images [35, 36, 37, 48, 131] implies solving simultaneously demosaicking, denoising, and possibly super-resolution. Such methods can generally be decomposed into two main steps: the registration step, where the images are expressed in a common system of coordinates, and the combination step, where the data are combined to form an image. In Chapter 8 we showed that there was no standard satisfactory registration method for mosaicked images, and we proposed an accurate and efficient two-step method. Assuming the images correctly registered, existing methods produce high-quality images [37, 38, 131] but the number of input images is limited by severe computational and memory costs. In Chapter 9, we broke this limitation by developing a simple, efficient and low memory combination method.

In this chapter, we propose an image formation method taking as input a large sequence of RAW images. This method is a version adapted to RAW images of our image fusion method described in Chapter 9, published in [13].

The RAW images are first preprocessed to transform the noise curve and to adjust the histograms. Then the two-step registration method for mosaicked images introduced in Chapter 8 is used to align the images. As in Chapter 9, the image combination stage, using classical kernel regression of second order with Gaussian kernel, is split into an accumulation step, where the images are processed sequentially, and an image computation step. Finally, the blur introduced by the classical kernel regression is inverted in a sharpening step. We show experimentally that, for a large number of RAW images, our image formation method provides, efficiently and with a low memory usage, a high-quality result. Our method successfully performs super-resolution and the residual noise decreases as expected. Our results are similar to slower and memory greedy methods.

The chapter is organized as follows: Section 11.2 describes the preprocessing step of our method. Our image formation algorithm from RAW images is presented in Section 11.3 and is experimentally validated in Section 11.4.

11.2 Preprocessing of RAW images

Let $(I^j_{\mathrm{RAW}})_{1\le j\le N_{\mathrm{im}}}$ be a sequence of N_{im} RAW images. In this section we describe how the input sequence $(I^j_{\mathrm{RAW}})_{1\le j\le N_{\mathrm{im}}}$ is preprocessed in our image formation algorithm. First, a variance stabilizing transform (VST) is applied to the images in order to approximate an additive, homoscedastic, white, and Gaussian noise (see Section 11.2.2). Then, the histograms of the images are ajusted (see Section 11.2.3). The aim is to optimize the performances of the registration and combination steps.

11.2.1 Color Handling

We start with notations and remarks about color channels that will prove useful.

Channel index. Let I be a color image. The three channels of I are index by $c \in \{1,2,3\}$. The intensity value at location (k,l) in the channel c is noted $I_{k,l,c}$.

RAW image and channel. Let I_{RAW} be a RAW image. In this chapter we assume that the color filter array is given by the classical Bayer filter RGGB [12]. The intensity value at location (k,l) is an intensity value corresponding to the channel c given by

$$c = k\%2 + l\%2 + 1 \tag{11.1}$$

where

$$n\%2 = \begin{cases} 0 & \text{if } n \text{ is even,} \\ 1 & \text{if } n \text{ is odd.} \end{cases} \tag{11.2}$$

11.2.2 Variance Stabilizing Transform

The registration and combination parts of our algorithm are designed for additive, homoscedastic, white, Gaussian noise. However, this is not a realistic assumption for RAW images for which the noise is the combination of a Poisson noise, coming from the photon emission, a thermal noise and an electronic noise [4, 23, 40]. A good approximation for such noise is an additive, white and Gaussian noise with a signal-dependent variance.

Variance stabilizing transform. To transform a signal-dependent noise into a nearly homoscedastic noise, a variance stabilizing transform (VST) has to be applied [23, 66]. Applying a VST to an image I consists in computing an image $f(I)$ (pixel-by-pixel operation) where $f : \mathbb{R}^+ \to \mathbb{R}^+$ is chosen so that the noise level in $f(I)$ is approximately independent from the intensity value. The choice of f depends on the function $g : \mathbb{R}^+ \to \mathbb{R}^+$ that associates to an intensity value u the variance $g(u)$ of the noise (in I). In [66] it is shown that a good choice for f is given by

$$f(u) \propto \int_0^u \frac{ds}{\sqrt{g(s)}}. \tag{11.3}$$

Anscombe transform. Using the classical assumption that the variance of the noise is an affine function of the intensity value, i.e., $g(u) = au + b$, (11.3) can be rewritten as

$$f(u) \propto \sqrt{au + b}. \tag{11.4}$$

This corresponds to a generalized Anscombe transform [7], which was originally designed for Poisson noise.

Practical application of the VST. We apply the VST to the sequence $(I^j_{\text{RAW}})_{1 \le j \le N_{\text{im}}}$ as follows. First, the noise curve $g_c(u)$ is estimated for each channel $c \in \{1,2,3\}$ of the first image I^1_{RAW} by the Ponomarenko et al. method [22, 92]. The parameters a_c and b_c such that $g_c(u) \simeq a_c u + b_c$ are computed by linear least squares. Then, for each image I^j_{RAW} the image I^j_{CFA} is obtained as

$$(I^j_{\text{CFA}})_{k,l} = \sqrt{a_c (I^j_{\text{RAW}})_{k,l} + b_c} \tag{11.5}$$

where $c = k\%2 + l\%2 + 1$.

Remarks. The VST is not inverted in the end of Algorithm 11.2. Note that in order to do it, one should not just use the algebraic inverse of the VST, which is biased. A discussion around the unbiased inverse VST is proposed in [76].

11.2.3 Adjustments of the Histograms

After the VST, the histograms of the images I^j_{CFA} are adjusted as follows.

Reference image. First, the histogram of the first image I^1_{CFA} is modified so that its channels have the same mean. This equalization is done multiplicatively. This color balance is interesting for two reasons: (1) it corrects the eventual unnatural color balance introduced during the VST and (2) a CFA image with equalized channels is likely to show an attenuated mosaic structure after the lowpass filtering of Algorithm 8.2. Then, the maximal value of I^1_{CFA} is arbitrarily set (multiplicatively) to 255.

Histogram equalization channel-by-channel. As the illumination of the scene may slightly vary during the acquisition, the histograms of the images have to be equalized. For $j \in \{2,\dots,N_{\text{im}}\}$ the means of the channels of I^j_{CFA} are set multiplicatively to the mean of I^1_{CFA} (same mean for the three channels). Note that by construction all the CFA images have equalized channels and a maximal value around 255.

The overall preprocessing step is summarized in Algorithm 11.1. It will be used in Algorithm 11.2. Without loss of generality the reference image can be any other image than the first one. Note that advanced photographic effects such as vignetting, distortion and chromatic aberration [112] are neglected.

Algorithm 11.1: Preprocessing of RAW images

Input : A sequence $(I^j_{\text{RAW}})_{1\le j\le N_{\text{im}}}$ of N_{im} RAW images
Output: A preprocessed sequence $(I^j_{\text{CFA}})_{1\le j\le N_{\text{im}}}$ of N_{im} mosaicked images

```
// Variance Stabilizing Transform
```
for $c \in \{1,2,3\}$ **do**
  Estimate the noise curve g_c of the channel c of I^1_{CFA}
  Compute a_c and b_c such that $g_c(u) \simeq a_c u + b_c$ by linear least squares
for $j \in \{1,\dots,N_{im}\}$ **do**
  Compute I^j_{CFA} from I^j_{RAW} using (11.5)

```
// Adjustments of the histograms
```
Equalize multiplicatively the means of the channels of I^1_{CFA}
Set multiplicatively the maximal value of I^1_{CFA} to 255
for $j \in \{2,\dots,N_{im}\}$ **do**
  Set multiplicatively the means of the channels of I^j_{CFA} to the mean of I^1_{CFA}

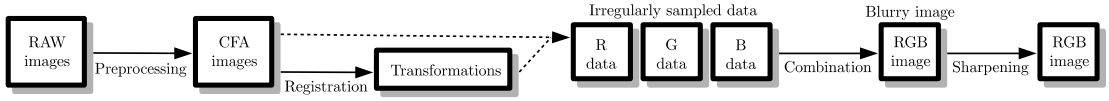
11.3 Image Formation from RAW images

In this section we propose a fast and low memory image formation method from RAW images that is designed for a large number of images. This is an adapted version of our image fusion method, described in Chapter 9, for RAW images. The RAW images are first preprocessed as described in Section 11.2. The considered registration method for mosaicked images was introduced in Chapter 8.

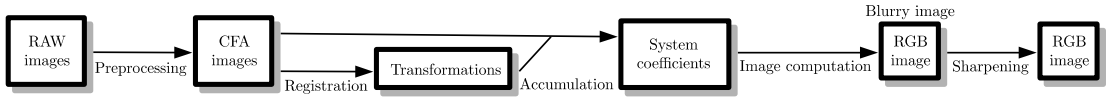
11.3.1 Proposed Algorithm

Given a sequence $(I^j_{\text{RAW}})_{1\le j\le N_{\text{im}}}$ of N_{im} RAW images of common size $M \times N$, a zoom factor $\lambda > 0$ and a scale $\sigma_s > 0$, our image formation builds a color image I of size $[\lambda M] \times [\lambda N]$. Our

algorithm is presented in Algorithm 11.2 and its main steps are summarized in Figure 11.1(b). The first image is arbitrarily chosen as the reference image.



(a) Memory greedy algorithm



(b) Our algorithm

Figure 11.1: Main steps of our image formation method. (a) and (b) are theoretically equivalent. The combination from the irregularly sampled data is replaced by the accumulation step, where the images are sequentially processed, and the image computation step.

Preprocessing. First, the sequence $(I^j_{\text{RAW}})_{1\le j\le N_{\text{im}}}$ is preprocessed as described in Section 11.2. The sequence $(I^j_{\text{CFA}})_{1\le j\le N_{\text{im}}}$ of mosaicked images is computed using Algorithm 11.1.

Registration. The transformation φ^1 is set to the identity. The homographic transformation φ^j between I^1_{CFA} and I^j_{CFA}, so that $I^1_{\text{CFA}} \simeq I^j_{\text{CFA}}(\varphi^j(\cdot))$, is estimated using a two-step method for the registration of mosaicked images, introduced in Chapter 8. More precisely, we use the recommended method that is summarized in Algorithm 8.2. It uses a perfect lowpass filter and the modified inverse compositional algorithm with robust error function.

Accumulation. The accumulation part is mostly the same as in Algorithm 9.1 except that the input images are mosaicked. Multi-image demosaicking can be considered as a form of super-resolution with factor 2. Therefore following the recommendation of Chapter 10 we use $N_{\text{KR}} = 2$. The system coefficents in $A_{\mathbf{x}^0,c}$ and $\mathbf{b}_{\mathbf{x}^0,c}$ for each output pixel $(\mathbf{x}^0,c) \in \Omega_{[\lambda M],[\lambda N]} \times \{1,2,3\}$ are computed by accumulation. First, they are initialized to zero i.e. $A_{\mathbf{x}^0,c} = 0_{\mathcal{M}_6(\mathbb{R})}$ and $\mathbf{b}_{\mathbf{x}^0,c} = 0_{\mathbb{R}^6}$. Then, the CFA images are processed sequentially. For each pixel $(k,l) \in \Omega_{M,N}$, the channel $c = k\%2 + l\%2 + 1$ and the corresponding location $\mathbf{x} = \lambda(\varphi^j)^{-1}(k,l)$ in the zoomed reference system are computed. The sample $(\mathbf{x}, I^j_{k,l})$ contributes to the system coefficients of the output pixel $\mathbf{x}^0 \in B(\mathbf{x}, 4\sigma_s)$ in the channel c as written in (9.14) and (9.15).

From here the steps are exactly the same as in Algorithm 9.1. We recall them for clarity.

Image computation. For each output pixel $(\mathbf{x}^0,c) \in \Omega_{[\lambda M],[\lambda N]} \in \{1,2,3\}$, the intensity value $I_{\mathbf{x}^0,c}$ is computed from $A_{\mathbf{x}^0,c}$ and $\mathbf{b}_{\mathbf{x}^0,c}$ as described in Section 9.2.2. It defines a color image I of size $[\lambda M] \times [\lambda N]$, which is blurry.

Sharpening. To remove the blur, the inverse of the equivalent asymptotic filter $\tilde{w}_{\sigma_s,\infty}$ (see Section 9.3.3) is applied to I using the DCT convolution method [45, 77].

Remarks.

- The same remarks as in Section 9.4.1 apply.
- In the preprocessing step the mean of the channels were equalized. The color images may look grayish. Depending on the context of use, the color balance of the output image can be adjusted in a post-processing step [31].

Algorithm 11.2: Image formation algorithm from RAW images

Input : A sequence $(I^j_{\text{RAW}})_{1\le j\le N_{\text{im}}}$ of N_{im} RAW images of size $M\times N$, a zoom factor $\lambda>0$ and a scale $\sigma_s>0$

Output: A color image I of size $[\lambda M]\times[\lambda N]$

// Preprocessing

Compute the preprocessed sequence $(I^j_{\text{CFA}})_{1\le j\le N_{\text{im}}}$ from $(I^j_{\text{RAW}})_{1\le j\le N_{\text{im}}}$ using Algorithm 11.1

// Registration

Set φ^1 to the identity transformation

for $j\in\{2,\dots,N_{im}\}$ **do**

  Compute the estimated homographic transformation φ^j between I^1_{CFA} and I^j_{CFA} using Algorithm 8.2

// Accumulation

Set $N_{\text{KR}}=2$

for $(\mathbf{x}^0,c)\in\Omega_{[\lambda M],[\lambda N]}\times\{1,2,3\}$ **do**

  Initialize $A_{\mathbf{x}^0,c}$ and $\mathbf{b}_{\mathbf{x}^0,c}$ to zero

for $j\in\{1,\dots,N_{im}\}$ **do**

  for $(k,l)\in\Omega_{M,N}$ **do**

    Compute $c=k\%2+l\%2+1$

    Compute $\mathbf{x}=\lambda(\varphi^j)^{-1}(k,l)$

    for $\mathbf{x}^0\in\Omega_{[\lambda M],[\lambda N]}\cap B(\mathbf{x},4\sigma_s)$ **do**

      Compute the Gaussian weight $w=w_{\sigma_s}(\mathbf{x}-\mathbf{x}^0)$ using (9.41)

      Compute $X=X(\mathbf{x}-\mathbf{x}^0)$ using (9.2)

      Update $A_{\mathbf{x}^0,c}\leftarrow A_{\mathbf{x}^0,c}+wXX^T$

      Update $\mathbf{b}_{\mathbf{x}^0,c}\leftarrow\mathbf{b}_{\mathbf{x}^0,c}+wX(I^j_{\text{CFA}})_{k,l}$

// Image computation

for $(\mathbf{x}^0,c)\in\Omega_{[\lambda M],[\lambda N]}\times\{1,2,3\}$ **do**

  Compute $I_{\mathbf{x}^0,c}$ from $A_{\mathbf{x}^0,c}$ and $\mathbf{b}_{\mathbf{x}^0,c}$ as described in Section 9.2.2

// Sharpening

Apply the inverse of the equivalent asymptotic filter $\tilde{w}_{\sigma_s,\infty}$ (see Section 9.3.3) to I using the DCT convolution

11.3.2 A Low Memory Requirement

As the low memory requirement is ensured in the combination stage, the following analysis is similar to the one made in Section 9.4.2 for the image fusion algorithm.

Algorithm 11.2 is designed for a large number of images N_{im}. As the images are processed sequentially, the memory requirement does not depend on the number of input images N_{im} but on the size of the output image. The majority of the memory requirement comes from the storage of the system coefficients. As we use $N_{\text{KR}}=2$, 21 coefficients have to be stored per output pixel. As the output color image is of size $[\lambda M]\times[\lambda N]$, the overall storage is of $63\,[\lambda M]\,[\lambda N]$ (double-precision floating point) numbers.

Theoretically the accumulation and image computation steps are equivalent to a single combination step taking as input the irregularly sampled data. The structure of the resulting memory greedy algorithm is shown in Fig. 11.1(a). The knowledge of the irregularly sampled data requires to store three numbers per sample (two for the location and one for the intensity). The overall storage represents $3N_{\text{im}}MN$ numbers. The storage of the system coefficients is smaller

than the storage of the data as soon as the number of images N_{im} becomes larger than $21\lambda^2$.

11.4 Experiments

In this section we evaluate on real data the performance of our image formation algorithm, described in Algorithm 11.2. Two sequences of RAW images are considered. First, the common experimental setup is described in Section 11.4.1. In Section 11.4.2 the results for the first sequence show the limits of our method when the spatial repartition of the samples is not uniform enough. On the contrary, our method performs well in Section 11.4.3 for the second sequence, which has a (more) uniform sample repartition.

We recall that the performance of the registration method was evaluated on synthetic data in Chapter 8. For the combination method (using classical kernel regression) it was done in Chapter 10.

11.4.1 Experimental Setup

The two considered sequences are composed of 201 RAW images of a static outdoor scene. In both cases, they were acquired using an Olympus E-M5 Mark II camera (with a 17.0mm lens) that was set in sequential mode. The camera was fixed on a tripod but the internal stabilization mode was desactivated. Thus, small motions/vibrations of the camera were allowed and demosaicking and super-resolution can be considered. Finally, the images of a sequence differ because of small motions of the camera, noise, quantization (14-bit), and possibly small illumination variations.

In both cases, we only keep the central part of the images of size 512×512, which is by the way distortion-free. As in Chapter 10 we arbitrarily set the first image as the reference image, which is used for the preprocessing and registration but not for the combination. The irregularly data fitting is performed on the remaining images (here at most $N_{\text{im}} = 200$).

In our experiments we considered the zoom factors $\lambda = 1$ (demosaicking without super-resolution) and $\lambda = 1.5$ (demosaicking and super-resolution). We recall that this respectively corresponds to zooms of factor 2 and 3 of the red and blue channels. As recommended in Chapter 10, our method uses the scale $\sigma_s = \frac{\sqrt{2}}{2}$ in both cases.

11.4.2 A First Sequence with Inadequate Spatial Repartition

The first sequence was obtained using a shutter speed of 1/50s, an aperture of f/5.6 and ISO 250. The camera was strongly fixed on the tripod so that the movements of the camera were limited. The reference CFA image (after preprocessing) is shown in Figure 11.3(a). The mosaic structure is clearly visible in Figure 11.3(b).

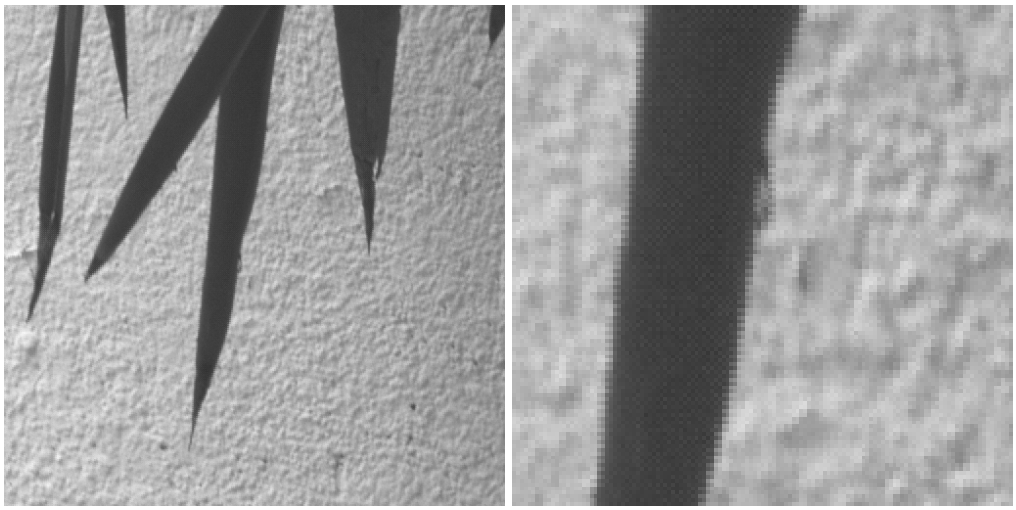
Spatial repartition. The estimated spatial repartition of the irregularly sampled data is shown in Figure 11.3. As the transformations between the images are locally similar to translations, the data repartition seems to have a periodicity of 2λ along each direction. As the camera did not move enough the samples are clustered and the spatial repartition is not uniform.

Output images. The output color images obtained from the $N_{\text{im}} = 200$ RAW images using our method are presented in Figure 11.4. For both zoom factors, chromatic aberration [99] can be seen near the edges. As expected the results for $\lambda = 1.5$ have a lower quality and artifacts, caused by the periodicity of the spatial data repartition, that can also be seen in the Fourier domain.

Comparison between reconstructed images. The difference between two reconstructed images from separate sets of 100 images is shown in Figure 11.5. For both zoom factors the structure of the image is visible. In the spatial domain, the difference is mostly composed of

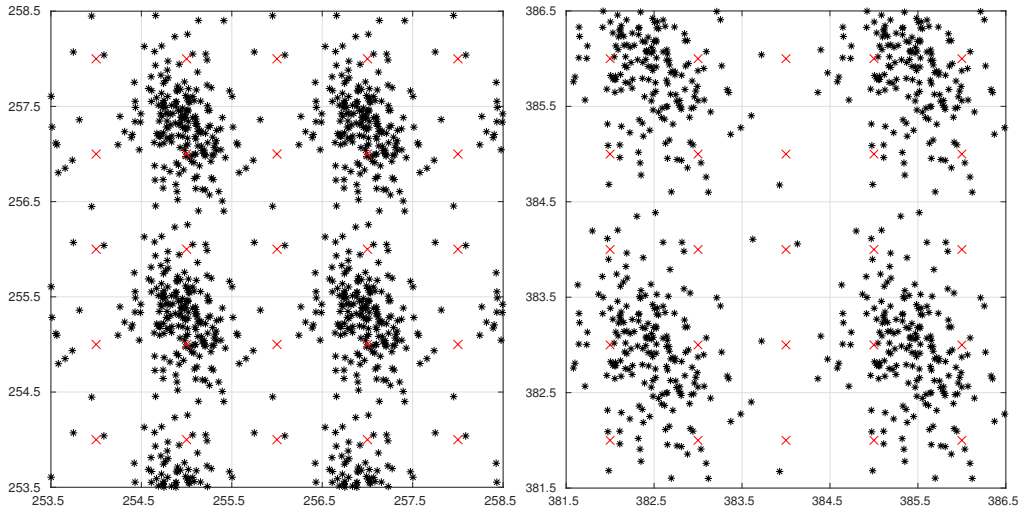
signal-dependent noise for $\lambda = 1$ while there is a lot of zipper structure [73] for $\lambda = 1.5$. In the Fourier domain, artifacts caused by the periodicity of the spatial data repartition are present for both zoom factors.

Conclusion. The non-uniform spatial repartition of the data has a negative impact on the results for two main reasons. First, this introduces a periodicity in the image content during the data fitting by classical kernel regression. The resulting high-frequency structures are boosted during the sharpening. Secondly, the asymptotic equivalent filter is not adapted and poorly approximates the actual equivalent filter.



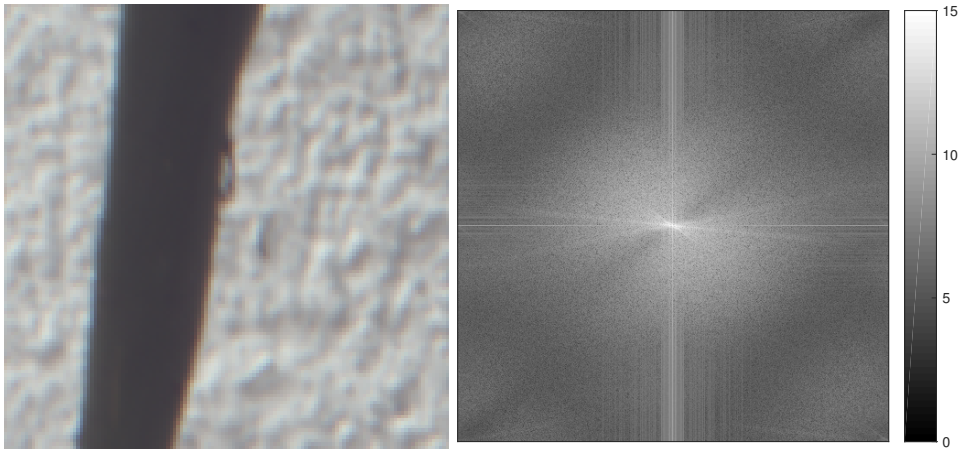
(a) Reference CFA image (b) Details of (a)

Figure 11.2: Dataset used in Section 11.4.2. The reference CFA image in (a) is of size 512×512 and corresponds to the preprocessed version of the reference RAW image. Details in a zone of size 128×128 are shown in (b). The mosaic structure is clearly visible near the edges.

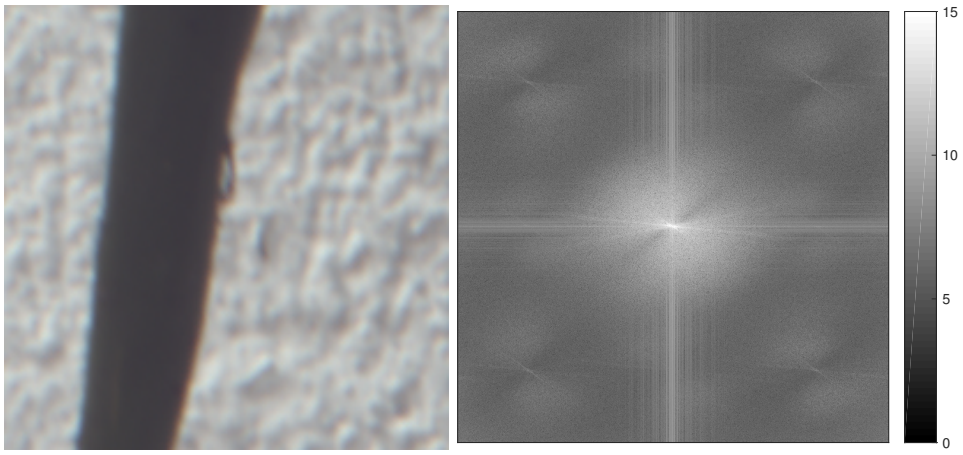


(a) $\lambda = 1$ (b) $\lambda = 1.5$

Figure 11.3: Estimated spatial repartition of the irregularly sampled data (red channel) for the first sequence (used in Section 11.4.2). Only a small region around the center of the image is shown. The factor λ corresponds to the zoom factor used in the image formation algorithm. As the transformations between the images are locally similar to translations, the data repartition seems to have a periodicity of 2λ along each direction. As the camera did not move enough the samples are clustered and the spatial repartition is not uniform.

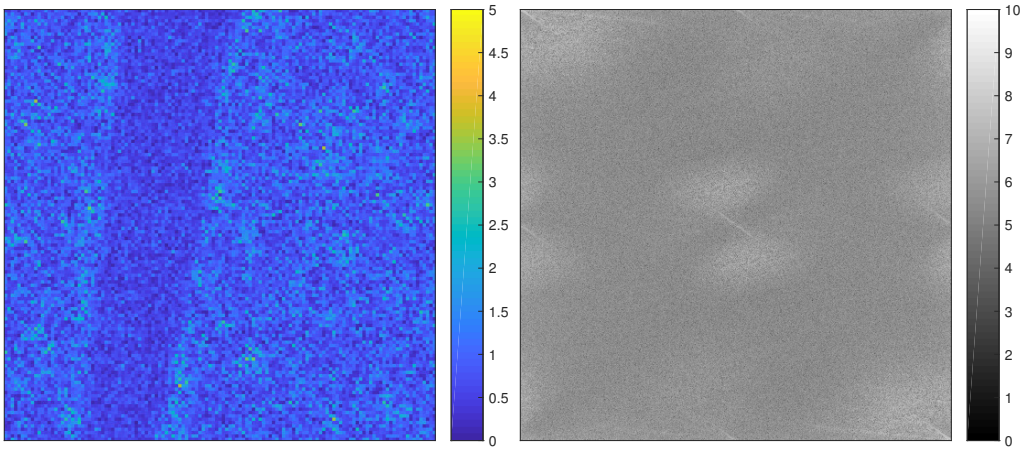


(a) Output image ($\lambda = 1$) (b) Spectrum of (a)



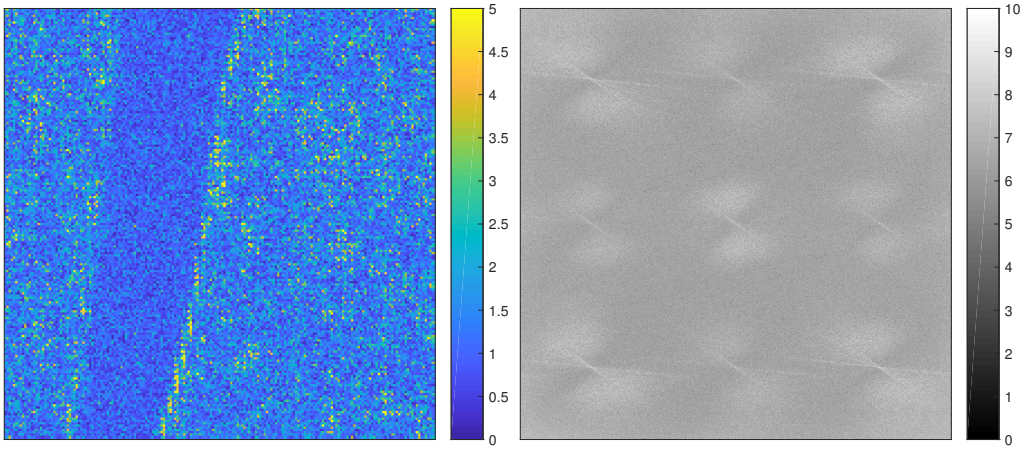
(c) Output image ($\lambda = 1.5$) (d) Spectrum of (c)

Figure 11.4: Output color images obtained from the first sequence of $N_{\text{im}} = 200$ RAW images using our method. For both zoom factors, chromatic aberration [99] can be seen near the edges. As expected the results for $\lambda = 1.5$ have a lower quality and artifacts, caused by the periodicity of the spatial data repartition, which can be seen in the Fourier domain.



(a) Difference image $\lambda = 1$ ($\mathcal{E}_{\text{set}} = 1.01$)

(b) Spectrum of (a)



(c) Difference image $\lambda = 1.5$ ($\mathcal{E}_{\text{set}} = 1.63$)

(d) Spectrum of (c)

Figure 11.5: Comparison between two reconstructed images from separate sets of $N_{\text{im}}/2 = 100$ images (first sequence). For both zoom factors the structure of the image is visible. In the spatial domain, the difference is mostly composed of signal-dependent noise for $\lambda = 1$ while there is a lot of zipper structure for $\lambda = 1.5$. In the Fourier domain, artifacts caused by the periodicity of the spatial data repartition are present for both zoom factors.

11.4.3 A Second Sequence with Adequate Spatial Repartition

The second sequence was obtained using a shutter speed of 1/100s, an aperture of f/5.6 and ISO 200. The camera was fixed on the tripod but larger movements of the camera were authorized compared to the first sequence. The reference CFA image (after preprocessing) is shown in Figure 11.7(a). The mosaic structure is clearly visible in Figure 11.7(b). Note that the corresponding JPEG images were used in Chapter 10 (see Figure 10.28).

Spatial repartition. The estimated spatial repartition of the irregularly sampled data is shown in Figure 11.7. As the transformations between the images are locally similar to translations, the data repartition seems to have a periodicity of 2λ along each direction. As the camera did move enough the data spatial repartition is close to be uniform.

Output images. The output color images obtained from the $N_{\text{im}} = 200$ RAW images using our method are presented in Figure 11.8. The images are sharp, without artifacts and there is no apparent noise. Contrarily to the results for the first sequence, the spectral artifacts caused by the periodicity can hardly be seen. Note that the corresponding output image for JPEG images can be seen in Figure 10.29.

Comparison between reconstructed images. The difference between two reconstructed images from separate sets of 100 images is shown in Figure 11.9. For both zoom factors the structure of the image is visible. For $\lambda = 1$ the difference is mostly composed of signal-dependent noise. However the structure is significantly less visible compared to the results for JPEG images (see Figure 10.30). This is an argument in favour of taking RAW images as input in multi-image methods. For $\lambda = 1.5$ the difference image contains zipper structures [73] but the impact on the spectrum is more important as for the first sequence.

The evolution of the error $\mathcal{E}_{\text{set}}$ with the number n of images used in each set is shown in Figure 11.10. The error tends to decrease with n but may increase temporarily when adding images. As noted in Chapter 10, this does not mean a deterioration of the results. For both zoom factors, the error $\mathcal{E}_{\text{set}}$ is approximately divided by 2 between 20 and 100 images instead of $\sqrt{100/20} \simeq 2.2$ in the ideal denoising case.

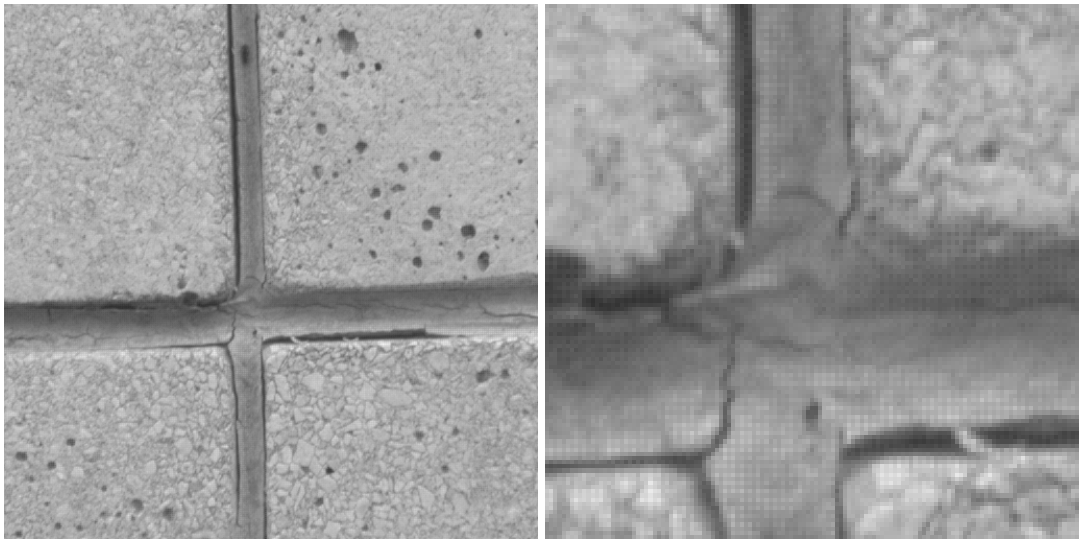
Comparison to the ACT method. Our image formation method is compared to a similar method where the combination part (by classical kernel regression) is replaced by the ACT method [38]. The results of the two methods are similar and cannot be distinguished to the naked eye. The difference images for $\lambda \in \{1, 1.5\}$ are shown in Figure 11.11. For both zoom factors the structure of the image and zipper structures are visible. The difference images do not seem to contain (residual) noise.

Computation time and maximal memory usage. The preprocessing and registration steps were performed in 108s, i.e., around 0.5s per image. Note that the experiments were made using an Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz (on a single thread) and a C language implementation. The computation times and maximal memory usages of our method and of the ACT are the following.

		ACT	Our method (KR)
$\lambda = 1$	Time (s)	612	**114**
	Max memory (kB)	3084984	**306604**
$\lambda = 1.5$	Time (s)	675	**133**
	Max memory (kB)	3651948	**675432**

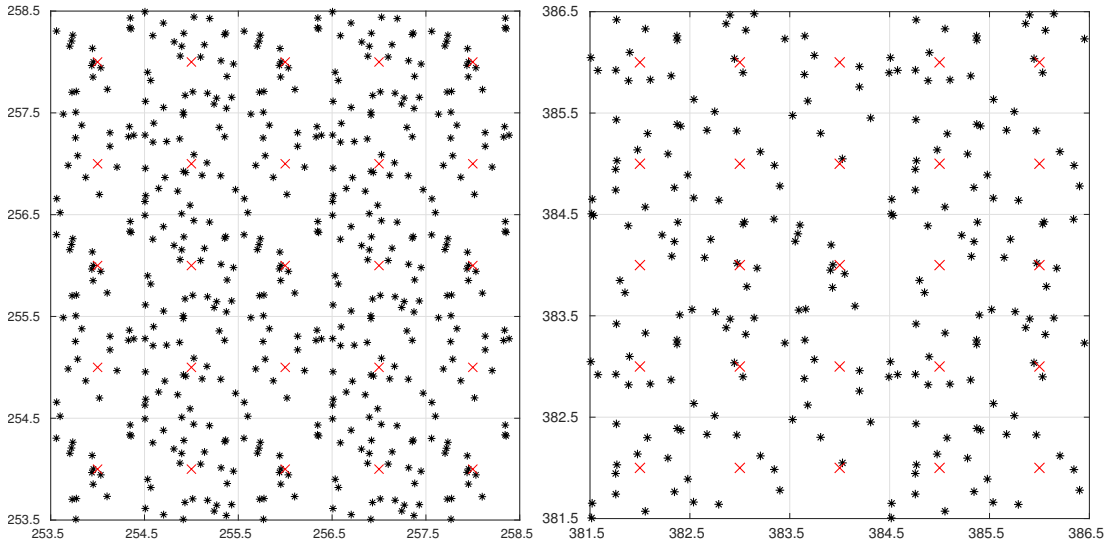
Our method is around 5 times faster. For $\lambda = 1$ it requires 10 times less memory. The ratio decreases to 5 for $\lambda = 1.5$ because the main memory cost of our method depends on the size of the output image.

Conclusion. Contrarily to image fused from processed images (as in Chapter 10), the images formed from RAW images do not contain artifacts coming from the unknown image processing pipeline. For a large number of RAW images and under the assumption of a uniform data spatial repartition, our image formation method provides, efficiently and with a low memory usage, a high-quality result. Our method successfully performs super-resolution and the residual noise decreases as expected. Our results are similar to slower and memory greedy methods.



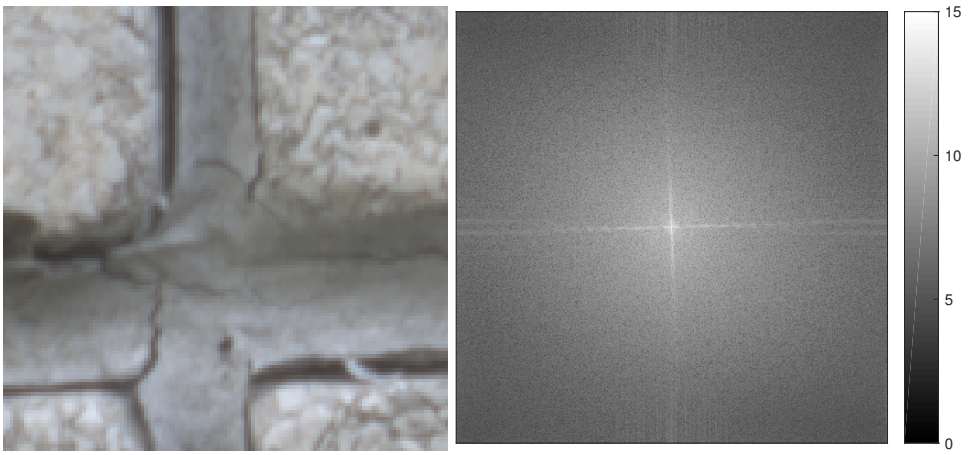
(a) Reference CFA image (b) Details of (a)

Figure 11.6: Dataset used in Section 11.4.3. The reference CFA image in (a) is of size 512×512 and corresponds to the preprocessed version of the reference RAW image. Details in a zone of size 128×128 are shown in (b). The mosaic structure is clearly visible, in particular near the edges. Note that the corresponding JPEG images were used in Chapter 10 (see Figure 10.28).

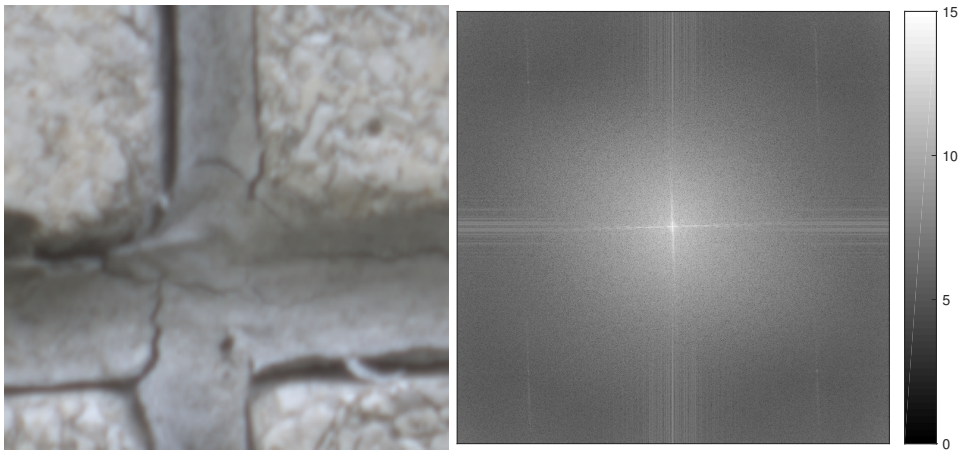


(a) $\lambda = 1$ (b) $\lambda = 1.5$

Figure 11.7: Estimated spatial repartition of the irregularly sampled data (red channel) for the second sequence (used in Section 11.4.3). Only a small region around the center of the image is shown. The factor λ corresponds to the zoom factor used in the image formation algorithm. As the transformations between the images are locally similar to translations, the data repartition seems to have a periodicity of 2λ along each direction. As the camera did move enough the data spatial repartition is close to be uniform.

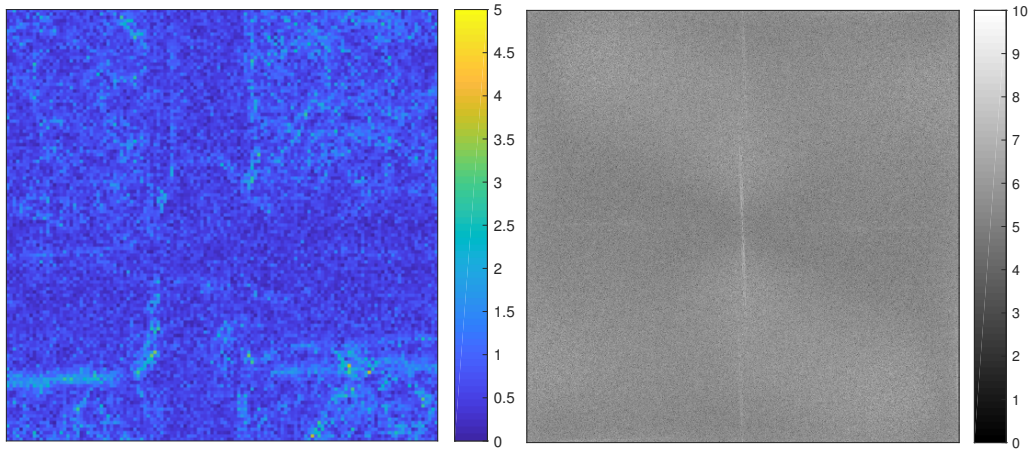


(a) Output image ($\lambda = 1$) (b) Spectrum of (a)



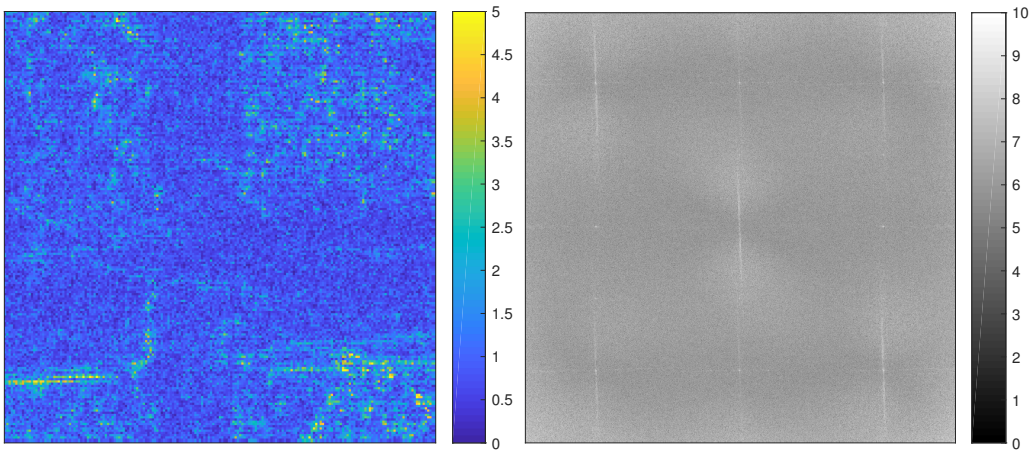
(c) Output image ($\lambda = 1.5$) (d) Spectrum of (c)

Figure 11.8: Output color images obtained from the second sequence of $N_{\text{im}} = 200$ RAW images using our method. The images are sharp, without artifacts and there is no apparent noise. Contrarily to the results for the first sequence, the spectral artifacts caused by the periodicity can hardly be seen. Note that the corresponding output image for JPEG images can be seen in Figure 10.29.



(a) Difference image $\lambda = 1$ ($\mathcal{E}_{\text{set}} = 0.80$)

(b) Spectrum of (a)



(c) Difference image $\lambda = 1.5$ ($\mathcal{E}_{\text{set}} = 1.15$)

(d) Spectrum of (c)

Figure 11.9: Comparison between two reconstructed images from separate sets of $N_{\text{im}}/2 = 100$ images (second sequence). For both zoom factors the structure of the image is visible. For $\lambda = 1$ the difference is mostly composed of signal-dependent noise. However the structure is significantly less visible compared to the results for JPEG images (see Figure 10.30). This is an argument in favour of taking RAW images as input in multi-image methods. For $\lambda = 1.5$ the difference image contains zipper structures but the impact on the spectrum is more important as for the first sequence.

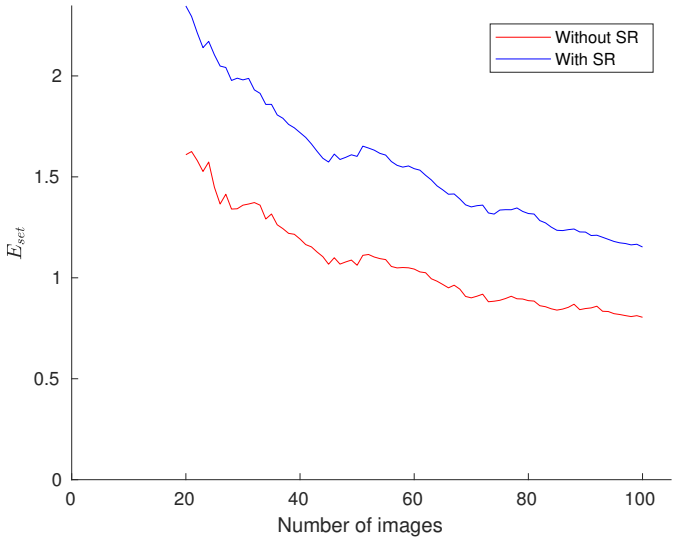
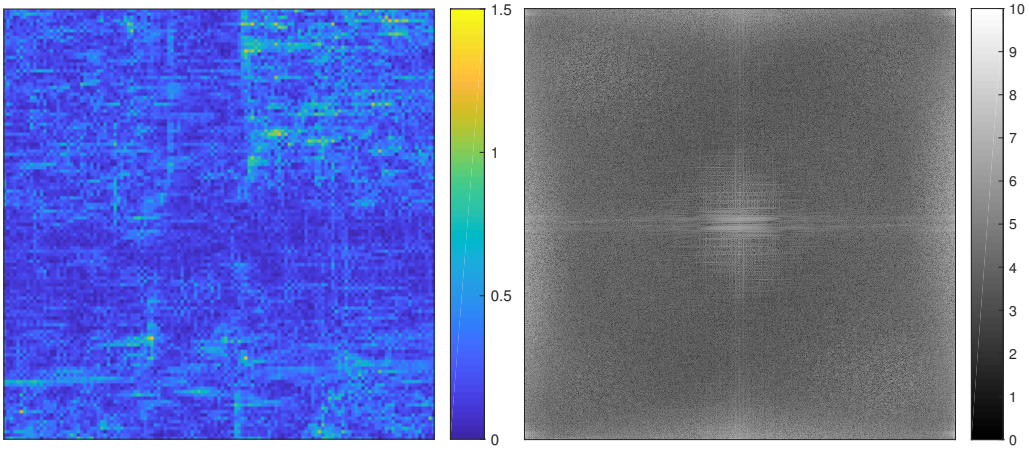
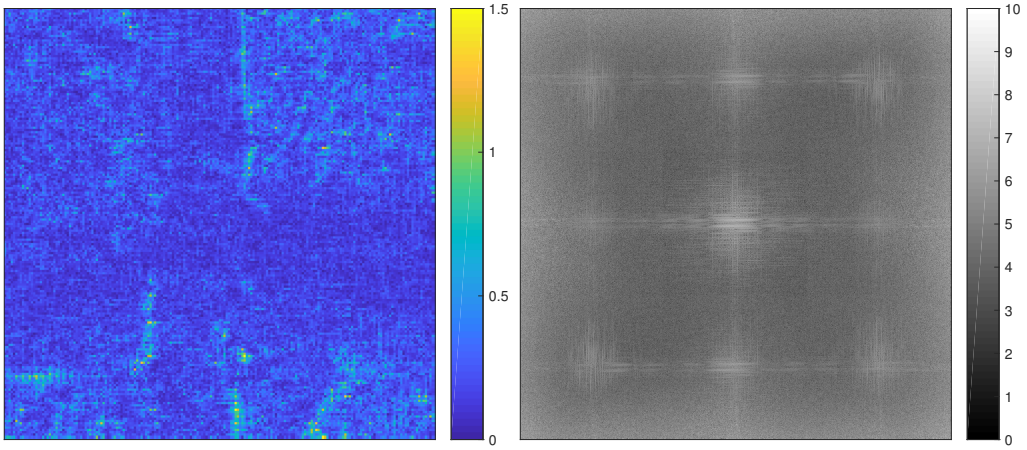


Figure 11.10: Evolution of the error $\mathcal{E}_{\text{set}}$ with the number n of images used in each set (second sequence). The error tends to decrease with n but may increase temporarily when adding images. As noted in Chapter 10, this does not mean a deterioration of the results. For both zoom factors, the error $\mathcal{E}_{\text{set}}$ is approximately divided by 2 between 20 and 100 images instead of $\sqrt{100/20} \simeq 2.2$ in the ideal denoising case.



(a) Difference image $\lambda = 1$ (RMSD = 0.32)

(b) Spectrum of (a)



(c) Difference image $\lambda = 1.5$ (RMSD = 1.03)

(d) Spectrum of (c)

Figure 11.11: Comparison to the ACT method. The results of the two methods are similar and cannot be distinguished to the naked eye. Therefore, the difference images are shown in (a) and (c). For both zoom factors the structure of the image and zipper structures are visible. The difference images do not seem to contain (residual) noise.

11.5 Conclusion

In this chapter, we proposed an image formation method from a large sequence of RAW images, which is an adapted version of the image fusion method described in Chapter 9.

The RAW images are first preprocessed to transform the noise curve and to adjust the histograms. Then the two-step registration method for mosaicked images, introduced in Chapter 8, is used to align the images. As in Chapter 9, the combination part, using classical kernel regression of second order with Gaussian kernel, is split into an accumulation part, where the images are processed sequentially, and an image computation part. Finally, the blur introduced by the classical kernel regression is inverted in a sharpening step.

We showed experimentally that, for a large number of RAW images, our image formation method provides, efficiently and with a low memory usage, a high-quality result. Our method sucessfully performs super-resolution and the residual noise decreases as expected. Our results are similar to slower and memory greedy methods. Contrarily to image fused from processed images (as in Chapter 10), the images formed from RAW images do not contain artifacts coming from the unknown image processing pipeline.

This efficient method opens the way to real time image formation from RAW images. The images are processed sequentially during the preprocessing, registration and accumulation steps so that on the fly computations can be performed. In our experiments, without any optimization of the code, the processing time per image (in these three steps) was around one second. Considering multi-threading (or using a GPU), this value may decrease significantly.

Chapter 12

Conclusion and perspectives

We addressed in this dissertation the problem of building a high-quality color image, containing a low level of noise and aliasing, from a large sequence (e.g. hundreds or thousands) of RAW images taken with a consumer camera. This was a challenging issue requiring to perform on the fly demosaicking, denoising and super-resolution. Existing methods were limited by inaccurate registration methods for mosaicked images and by severe computational and memory costs. We broke these limitations by providing an efficient and accurate two-step registration method for mosaicked images, and by developping a simple, efficient and low memory combination method.

During the dissertation each step of the proposed image formation method was introduced and analyzed separately. The performance and the accuracy were evaluated on synthetic and real data. This procedure allowed for a control of the error at each step but also for a proper analysis of the proposed improvements of existing methods. Generating synthetic data required an interpolation method and controlling the interpolation error was crucial for analyzing the performance of our method. From our study of interpolation methods we derived new fine-tuned interpolation methods that we used to generate synthetic data. It may be interesting to undertake a more thorough study of the properties of such new methods.

In our analysis, we found that for a large sequence of RAW images, our image formation method from RAW images successfully performs super-resolution with a residual noise decreasing as expected. We obtained results similar to those obtained by slower and memory greedy methods. Contrarily to images fused from previously processed images (e.g. JPEG images), the images formed from RAW images do not contain artifacts coming from the previous unknown image processing pipeline.

The key of our method efficiency and low memory requirement is that the small linear systems involved in classical kernel regression can be computed by data accumulation. The key of our method accuracy is its sharpening step. Thanks to our analysis of the classical kernel regression we were able to understand the behavior of the method for large datasets and we introduced the asymptotic equivalent kernel. The low-quality of the image computed by classical kernel regression is compensated in the sharpening step where the blur introduced is successfully removed.

Improvements of the method can still be envisaged. For instance, it is possible to slightly modify the registration method so that it takes into account lens distortion. The homography transformation can be replaced by local translations (in small blocks). In addition, badly registered images could be automatically detected. Also, a post-processing enhancement may be considered. Notably, chromatic aberration can be handled using [99].

Our method opens the way to highly accurate real time image formation from RAW images. The images are processed sequentially during the preprocessing, registration and accumulation steps so that on the fly computations can be performed. In our experiments, which did not involve any code optimization, the processing time per image of size 512×512 (in these three

steps) was around one second. Considering multi-threading (or using a GPU), this value will decrease significantly.

Open-source Tools

The following open-source tools were used:

- *imscript* by Enric Meinhardt-Llopis: a collection of small and standalone utilities for image processing, written in pure C. Available at https://github.com/mnhrdt/imscript/.
- *pvflip* by Gabriele Facciolo: an OpenGL accelerated image viewer. Available at https://github.com/gfacciol/pvflip/.
- *vpv* by Jérémy Anger: a viewer of image sequences to analyze image and video processing results. Available at https://github.com/kidanger/vpv/.
- *GNU parallel* by Ole Tange: a shell tool for executing jobs in parallel [120, 121]. Available at https://www.gnu.org/software/parallel/.
- *dcraw* by Dave Coffin: an ANSI C program that decodes raw images. Available at https://www.cybercom.net/~dcoffin/dcraw/.

Bibliography

[1] R. Abergel and L. Moisan. "The Shannon Total Variation". In: *Journal of Mathematical Imaging and Vision* (2017), pp. 1–30. ISSN: 1573-7683. DOI: 10.1007/s10851-017-0733-5 (see pp. 18, 38, 58, 62, 64, 134, 142, 146).

[2] C. Aguerrebere, J. Delon, Y. Gousseau, and P. Musé. "Best Algorithms for HDR Image Generation. A Study of Performance Bounds". In: *SIAM Journal on Imaging Sciences* 7.1 (2014), pp. 1–34. DOI: 10.1137/120891952 (see pp. 15, 35).

[3] C. Aguerrebere, A. Almansa, Y. Gousseau, J. Delon, and P. Muse. "Single shot high dynamic range imaging using piecewise linear estimators". In: *2014 IEEE International Conference on Computational Photography (ICCP)*. Vol. 00. 2014, pp. 1–10. DOI: 10.1109/ICCPHOT.2014.6831807 (see pp. 15, 35).

[4] C. Aguerrebere, J. Delon, Y. Gousseau, and P. Muse. "Study of the digital camera acquisition process and statistical modeling of the sensor raw data". In: (2013). URL: https://hal.archives-ouvertes.fr/hal-00733538 (see pp. 258, 259).

[5] A. Aldroubi, M. Unser, and M. Eden. "Cardinal Spline Filters: Stability and Convergence to the Ideal Sinc Interpolator". In: *Signal Processing* 28.2 (1992), pp. 127–138. DOI: 10.1016/0165-1684(92)90030-Z (see pp. 20, 40, 101, 103, 104, 134).

[6] D. Alleysson, S. Susstrunk, and J. Herault. "Linear demosaicing inspired by the human visual system". In: *IEEE Transactions on Image Processing* 14.4 (2005), pp. 439–449. ISSN: 1057-7149. DOI: 10.1109/TIP.2004.841200 (see pp. 26, 46, 187, 188, 191, 192, 198).

[7] F. Anscombe. "The transformation of Poisson, binomial and negative-binomial data". In: *Biometrika* 35.3/4 (1948), pp. 246–254. DOI: 10.2307/2332343 (see p. 259).

[8] S. Baker and I. Matthews. "Equivalence and efficiency of image alignment algorithms". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990652 (see pp. 24, 44, 161, 162, 165, 166).

[9] S. Baker and I. Matthews. "Lucas-kanade 20 years on: A unifying framework". In: *International journal of computer vision* 56.3 (2004), pp. 221–255. DOI: 10.1023/B:VISI.00000 (see p. 162).

[10] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. "A Database and Evaluation Methodology for Optical Flow". In: *International Journal of Computer Vision* 92.1 (2011), pp. 1–31. DOI: 10.1007/s11263-010-0390-2 (see pp. 24, 44, 149, 175, 176, 196, 220).

[11] S. Baker, R. Gross, I. Matthews, and T. Ishikawa. *Lucas-Kanade 20 Years On: A Unifying Framework: Part 2*. Tech. rep. CMU-RI-TR-03-01. Pittsburgh, PA: Robotics Institute , Carnegie Mellon University, 2003 (see pp. 162, 166).

[12] B. Bayer. *Color imaging array*. US Patent 3,971,065. 1976 (see pp. 26, 46, 187–189, 259).

[13] T. Briand. "Low Memory Image Reconstruction Algorithm from RAW Images". In: *2018 IEEE 13th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*. 2018, pp. 1–5. DOI: 10.1109/IVMSPW.2018.8448561 (see pp. 30, 50, 257, 258).

[14] T. Briand and P. Monasse. "Theory and Practice of Image B-Spline Interpolation". In: *Image Processing On Line* 8 (2018), pp. 99–141. DOI: 10.5201/ipol.2018.221 (see pp. 20, 31, 40, 53, 99).

[15] T. Briand and J. Vacher. "How to Apply a Filter Defined in the Frequency Domain by a Continuous Function". In: *Image Processing On Line* 6 (2016), pp. 183–211. DOI: 10.5201/ipol.2016.116 (see pp. 18, 31, 38, 53, 59, 62, 75, 82).

[16] T. Briand, G. Facciolo, and J. Sánchez. "Improvements of the Inverse Compositional Algorithm for Parametric Motion Estimation". In: *IPOL* (PREPRINT 2018). URL: https://www.ipol.im/pub/pre/222/ (see pp. 31, 53, 161).

[17] T. Briand, J. Vacher, B. Galerne, and J. Rabin. "The Heeger & Bergen Pyramid Based Texture Synthesis Algorithm". In: *Image Processing On Line* 4 (2014), pp. 276–299. DOI: 10.5201/ipol.2014.79 (see pp. 76, 83, 84, 86, 97, 146).

[18] A. Buades, B. Coll, and J-M Morel. "A non-local algorithm for image denoising". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 2. IEEE. 2005, pp. 60–65. DOI: 10.1109/CVPR.2005.38 (see pp. 15, 35).

[19] A. Buades, Y. Lou, J.M. Morel, and Z. Tang. "A note on multi-image denoising". In: *Int. Workshop on Local and Non-Local Approximation in Image Processing*. 2009, pp. 1–15. DOI: 10.1109/LNLA.2009.5278408 (see pp. 15, 28, 35, 48, 202).

[20] A. Buades, B. Coll, and J. M. Morel. "The staircasing effect in neighborhood filters and its solution". In: *IEEE Transactions on Image Processing* 15.6 (2006), pp. 1499–1505. ISSN: 1057-7149. DOI: 10.1109/TIP.2006.871137 (see pp. 202, 206).

[21] E.W. Cheney. *Approximation theory III*. Vol. 12. ISBN 9780121710507. Academic Press New York, 1980 (see pp. 17, 37).

[22] M. Colom and A. Buades. "Analysis and Extension of the Ponomarenko et al. Method, Estimating a Noise Curve from a Single Image". In: *Image Processing On Line* 3 (2013), pp. 173–197. DOI: 10.5201/ipol.2013.45 (see p. 259).

[23] M. Colom, A. Buades, and J-M Morel. "Nonparametric noise estimation method for raw images". In: *JOSA A* 31.4 (2014), pp. 863–871. DOI: 10.1364/JOSAA.31.000863 (see p. 259).

[24] J. W Cooley and J. Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of computation* 19.90 (1965), pp. 297–301. DOI: 10.2307/2003354 (see pp. 58, 61, 68, 69).

[25] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *IEEE Transactions on image processing* 16.8 (2007), pp. 2080–2095. DOI: 10.1109/TIP.2007.901238 (see pp. 15, 35).

[26] M. Delbracio and G. Sapiro. "Removing camera shake via weighted Fourier burst accumulation". In: *IEEE Transactions on Image Processing* 24.11 (2015), pp. 3293–3307. DOI: 10.1109/TIP.2015.2442914 (see pp. 15, 35).

[27] F. Dellinger, J. Delon, Y. Gousseau, J. Michel, and F. Tupin. "SAR-SIFT: A SIFT-Like Algorithm for SAR Images". In: *IEEE Transactions on Geoscience and Remote Sensing* 53.1 (2015), pp. 453–466. ISSN: 0196-2892. DOI: 10.1109/TGRS.2014.2323552 (see pp. 24, 44).

[28] J. Delon. "Midway image equalization". In: *Journal of Mathematical Imaging and Vision* 21.2 (2004), pp. 119–134. DOI: 10.1023/B:JMIV.0000035178.72139.2d (see pp. 165, 213).

[29] P. Dierckx. *Curve and surface fitting with splines*. ISBN 9780198534402. Oxford University Press, 1995 (see pp. 17, 37).

[30] W. Dorn. "Generalizations of Horner's rule for polynomial evaluation". In: *IBM Journal of Research and Development* 6.2 (1962), pp. 239–245. DOI: 10.1147/rd.62.0239 (see p. 127).

[31] M. Ebner. *Color constancy*. Vol. 6. John Wiley & Sons, 2007. DOI: 10.1002/9780470510490 (see p. 261).

[32] M. Elad. "On the origin of the bilateral filter and ways to improve it". In: *IEEE Transactions on image processing* 11.10 (2002), pp. 1141–1151. DOI: 10.1109/TIP.2002.801126 (see pp. 202, 206).

[33] G. Facciolo, A. Almansa, J-F Aujol, and V. Caselles. "Irregular to Regular Sampling, Denoising, and Deconvolution". In: *Multiscale Modeling and Simulation* (2009), pp. 1574–1608. DOI: 10.1137/080719443 (see p. 214).

[34] H. Farid and E. P. Simoncelli. "Differentiation of discrete multidimensional signals". In: *IEEE Transactions on image processing* 13.4 (2004), pp. 496–508. DOI: 10.1109/TIP.2004.823819 (see pp. 171, 177).

[35] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. "Advances and challenges in super-resolution". In: *International Journal of Imaging Systems and Technology* 14.2 (2004), pp. 47–57. ISSN: 1098-1098. DOI: 10.1002/ima.20007 (see pp. 15, 35, 188, 258).

[36] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. "Dynamic demosaicing and color superresolution of video sequences". In: *Image Reconstruction from Incomplete Data III*. Vol. 5562. International Society for Optics and Photonics. 2004, pp. 169–179. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.8268&rep=rep1&type=pdf (see pp. 15, 35, 188, 258).

[37] S. Farsiu, M. Elad, and P. Milanfar. "Multiframe demosaicing and super-resolution of color images". In: *IEEE transactions on image processing* 15.1 (2006), pp. 141–159. DOI: 10.1109/TIP.2005.860336 (see pp. 15, 16, 35, 188, 258).

[38] H. Feichtinger, K. Gröchenig, and T. Strohmer. "Efficient numerical methods in non-uniform sampling theory". In: *Numerische Mathematik* 69.4 (1995), pp. 423–440. DOI: 10.1007/s002110050 (see pp. 16, 35, 202, 218, 258, 267).

[39] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395. DOI: 10.1145/358669.358692 (see pp. 162, 181).

[40] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian. "Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data". In: *IEEE Transactions on Image Processing* 17.10 (2008), pp. 1737–1754. DOI: 10.1109/TIP.2008.2001399 (see p. 259).

[41] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, 2001 (see pp. 16, 28, 36, 48, 201, 202, 207).

[42] M. Frigo and S.G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005), pp. 216–231. DOI: 10.1109/JPROC.2004.840301 (see pp. 69, 82).

[43] B. Galerne, Y. Gousseau, and J-M Morel. "Random phase textures: Theory and synthesis". In: *IEEE Transactions on image processing* 20.1 (2011), pp. 257–267. DOI: 10.1109/TIP.2010.2052822 (see p. 146).

[44] J. Gentle. *Numerical linear algebra for applications in statistics*. Springer Science & Business Media, 2012 (see p. 204).

[45] P. Getreuer. "A Survey of Gaussian Convolution Algorithms". In: *Image Processing On Line* 3 (2013), pp. 286–310. DOI: 10.5201/ipol.2013.87 (see pp. 76, 84, 191, 213, 261).

[46] P. Getreuer. "Linear Methods for Image Interpolation". In: *Image Processing On Line* 1 (2011). DOI: 10.5201/ipol.2011.g_lmii (see pp. 17, 37, 101, 103, 142, 165, 167, 171).

[47] D. Glasner, S. Bagon, and M. Irani. "Super-resolution from a single image". In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 349–356. DOI: 10.1109/ICCV.2009.5459271 (see pp. 15, 35).

[48] T. Gotoh and M. Okutomi. "Direct super-resolution and registration using raw CFA images". In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 2. IEEE, pp. II–II. DOI: 10.1109/CVPR.2004.1315219 (see pp. 15, 16, 35, 188, 258).

[49] M. D. Grossberg and S. K. Nayar. "Determining the camera response from images: what is knowable?" In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.11 (2003), pp. 1455–1467. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2003.1240119 (see p. 258).

[50] T. Guillemot and J. Delon. "Implementation of the Midway Image Equalization". In: *Image Processing On Line* 6 (2016), pp. 114–129. DOI: 10.5201/ipol.2016.140 (see pp. 165, 213).

[51] J. Hammersley. *Monte carlo methods*. Springer Science & Business Media, 2013 (see p. 208).

[52] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. ISBN 9780521540513. Cambridge university press, 2nd edition, 2004 (see pp. 24, 44, 128).

[53] S. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. Barron, F. Kainz, J. Chen, and M. Levoy. "Burst Photography for High Dynamic Range and Low-light Imaging on Mobile Cameras". In: *ACM Trans. Graph.* 35.6 (Nov. 2016), 192:1–192:12. ISSN: 0730-0301. DOI: 10.1145/2980179.2980254 (see pp. 15, 35).

[54] D.J. Heeger and J.R. Bergen. "Pyramid-based texture analysis/synthesis". In: *Proceedings of the 22nd Annual Conference On Computer Graphics And Interactive Techniques*. ACM. 1995, pp. 229–238. DOI: 10.1145/218380.218446 (see p. 84).

[55] C. Hessel. "The automatic decomposition of an image in base and detail : Application to contrast enhancement". Theses. Université Paris-Saclay, 2018. URL: https://tel.archives-ouvertes.fr/tel-01826213 (see pp. 202, 206).

[56] B. Horn and B. Schunck. "Determining optical flow". In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203. DOI: 10.1016/0004-3702(81)90024-2 (see pp. 24, 44).

[57] H. Hou and H. Andrews. "Cubic Splines for Image Interpolation and Digital Filtering". In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 26.6 (1978), pp. 508–517. DOI: 10.1109/TASSP.1978.1163154 (see p. 101).

[58] D.H. Hubel and T.N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of Physiology* 160.1 (1962), p. 106. DOI: 10.1113/jphysiol.1962.sp006837 (see p. 84).

[59] AJ. Jerri. "The Shannon sampling theorem; Its various extensions and applications: A tutorial review". In: *Proceedings of the IEEE* 65.11 (1977), pp. 1565–1596. ISSN: 0018-9219. DOI: 10.1109/PROC.1977.10771 (see pp. 15, 35).

[60] E.I. Jury. *Theory and Application of the Z-Transform Method*. ISBN 9780882751221. Wiley, 1964 (see p. 104).

[61] W. C. Kao. "High Dynamic Range Imaging by Fusing Multiple Raw Images and Tone Reproduction". In: *IEEE Transactions on Consumer Electronics* 54.1 (2008), pp. 10–15. ISSN: 0098-3063. DOI: 10.1109/TCE.2008.4470017 (see pp. 15, 35, 188, 258).

[62] Y. Katznelson. *An introduction to harmonic analysis*. ISBN 0521543592. chapter 1. Cambridge University Press, 2004 (see p. 77).

[63] J. Keiner, S. Kunis, and D. Potts. "Using NFFT 3—a software library for various nonequispaced fast Fourier transforms". In: *ACM Transactions on Mathematical Software (TOMS)* 36.4 (2009), p. 19. DOI: 10.1145/1555386.1555388 (see pp. 69, 134).

[64] H. Knutsson and C-F Westin. "Normalized and differential convolution". In: *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*. IEEE. 1993, pp. 515–523. URL: 10.1109/CVPR.1993.341081 (see p. 202).

[65] S. Kunis and D. Potts. *Time and memory requirements of the nonequispaced FFT*. Technische Universität Chemnitz. Fakultät für Mathematik, 2006 (see p. 69).

[66] M. Lebrun, M. Colom, A. Buades, and J-M Morel. "Secrets of image denoising cuisine". In: *Acta Numerica* 21 (2012), pp. 475–576. DOI: 10.1017/S0962492912000062 (see p. 259).

[67] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *CVPR*. Vol. 2. 3. 2017, p. 4. DOI: https://arxiv.org/abs/1609.04802 (see pp. 15, 35).

[68] X. Li, B. Gunturk, and L. Zhang. "Image demosaicing: A systematic survey". In: *Visual Communications and Image Processing 2008*. Vol. 6822. International Society for Optics and Photonics. 2008, 68221J. DOI: 10.1117/12.766768 (see pp. 15, 35, 188).

[69] T. Lin and J. Barron. *Image reconstruction error for optical flow*. Citeseer, 1994 (see p. 142).

[70] M. Liou. "Spline fit made easy". In: *IEEE Transactions on Computers* 100.5 (1976), pp. 522–527. DOI: 10.1109/TC.1976.1674640 (see p. 101).

[71] Z. Liu, L. Yuan, X. Tang, M. Uyttendaele, and J. Sun. "Fast burst images denoising". In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), p. 232. DOI: 10.1145/2661229.2661277 (see pp. 15, 28, 35, 48, 202).

[72] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94 (see pp. 162, 181).

[73] W. Lu and Y. Tan. "Color filter array demosaicking: new method and performance measures". In: *IEEE Transactions on Image Processing* 12.10 (2003), pp. 1194–1210. DOI: 10.1109/TIP.2003.816004 (see pp. 172, 188, 264, 267).

[74] B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679. URL: http://dl.acm.org/citation.cfm?id=1623264.1623280 (see pp. 24, 44, 161, 162, 165).

[75] J.B.Antoine Maintz and Max A. Viergever. "A survey of medical image registration". In: *Medical Image Analysis* 2.1 (1998), pp. 1 –36. ISSN: 1361-8415. DOI: 10.1016/S1361-8415(01)80026-8 (see pp. 24, 44).

[76] M. Makitalo and A. Foi. "A Closed-Form Approximation of the Exact Unbiased Inverse of the Anscombe Variance-Stabilizing Transformation". In: *IEEE Transactions on Image Processing* 20.9 (2011), pp. 2697–2698. ISSN: 1057-7149. DOI: 10.1109/TIP.2011.2121085 (see p. 259).

[77] S. A. Martucci. "Symmetric convolution and the discrete sine and cosine transforms". In: *IEEE Transactions on Signal Processing* 42.5 (1994), pp. 1038–1051. ISSN: 1053-587X. DOI: 10.1109/78.295213 (see pp. 213, 261).

[78] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H. Shum. "Full-frame video stabilization with motion inpainting". In: *IEEE Transactions on pattern analysis and Machine Intelligence* 28.7 (2006), pp. 1150–1163. DOI: 10.1109/TPAMI.2006.141 (see pp. 24, 44).

[79] E. Meinhardt-Llopis, J. Sánchez Pérez, and D. Kondermann. "Horn-Schunck Optical Flow with a Multi-Scale Strategy". In: *Image Processing On Line* 3 (2013), pp. 151–172. DOI: 10.5201/ipol.2013.20 (see p. 167).

[80] G. Meneghetti, M. Danelljan, M. Felsberg, and K. Nordberg. "Image Alignment for Panorama Stitching in Sparsely Structured Environments". In: *Image Analysis*. Cham: Springer International Publishing, 2015, pp. 428–439. ISBN: 978-3-319-19665-7. DOI: 10.1007/978-3-319-19665-7_36 (see pp. 182, 183).

[81] T. Mertens, J. Kautz, and F. Van Reeth. "Exposure fusion: A simple and practical alternative to high dynamic range photography". In: *Computer Graphics Forum*. Vol. 28. 1. Wiley Online Library. 2009, pp. 161–171. DOI: 10.1111/j.1467-8659.2008.01171.x (see pp. 15, 35).

[82] P. Milanfar. "A Tour of Modern Image Filtering: New Insights and Methods, Both Practical and Theoretical". In: *IEEE Signal Processing Magazine* 30.1 (2013), pp. 106–128. ISSN: 1053-5888. DOI: 10.1109/MSP.2011.2179329 (see p. 206).

[83] G. Milovanović and Z. Udovičić. "Calculation of Coefficients of a Cardinal B-spline". In: *Applied Mathematics Letters* 23.11 (2010), pp. 1346–1350. DOI: 10.1016/j.aml.2010.06.029 (see p. 127).

[84] L. Moisan. "Periodic plus smooth image decomposition". In: *Journal of Mathematical Imaging and Vision* 39.2 (2011), pp. 161–179. DOI: 10.1007/s10851-010-0227-1 (see pp. 142, 146, 147).

[85] L. Moisan, P. Moulon, and P. Monasse. "Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers". In: *Image Processing On Line* 2 (2012), pp. 56–73. DOI: 10.5201/ipol.2012.mmm-oh (see pp. 128, 181).

[86] O. V. Morozov, M. Unser, and P. Hunziker. "Reconstruction of Large, Irregularly Sampled Multidimensional Images. A Tensor-Based Approach". In: *IEEE Transactions on Medical Imaging* 30.2 (2011), pp. 366–374. ISSN: 0278-0062. DOI: 10.1109/TMI.2010.2078832 (see pp. 16, 35).

[87] K. Nasrollahi and T. Moeslund. "Super-resolution: a comprehensive survey". In: *Machine Vision and Applications* 25.6 (2014), pp. 1423–1468. ISSN: 1432-1769. DOI: 10.1007/s00138-014-0623-4 (see pp. 15, 16, 35, 202).

[88] F. Oliveira and J. Tavares. "Medical image registration: a review". In: *Computer Methods in Biomechanics and Biomedical Engineering* 17.2 (2014), pp. 73–93. DOI: 10.1080/10255842.2012.670855 (see pp. 24, 44).

[89] A. Papandreou-Suppappola. *Applications in time-frequency signal processing*. ISBN 1420042467. CRC press, 2002 (see p. 76).

[90] S. Paris, P. Kornprobst, J. Tumblin, F. Durand, et al. "Bilateral filtering: Theory and applications". In: *Foundations and Trends® in Computer Graphics and Vision* 4.1 (2009), pp. 1–73. DOI: 10.1561/0600000020 (see pp. 202, 206).

[91] T. Pham, L. J Van Vliet, and K. Schutte. "Robust fusion of irregularly sampled data using adaptive normalized convolution". In: *EURASIP Journal on Advances in Signal Processing* 2006.1 (2006), pp. 1–12. DOI: 10.1155/ASP/2006/83268 (see pp. 16, 35, 202, 206).

[92] N. Ponomarenko, V. Lukin, M. Zriakhov, A. Kaarna, and J. Astola. "An automatic approach to lossy compression of AVIRIS images". In: *2007 IEEE International Geoscience and Remote Sensing Symposium*. 2007, pp. 472–475. DOI: 10.1109/IGARSS.2007.4422833 (see p. 259).

[93] J. Portilla and E.P. Simoncelli. "A parametric texture model based on joint statistics of complex wavelet coefficients". In: *International Journal of Computer Vision* 40.1 (2000), pp. 49–70. DOI: 10.1023/A:1026553619983 (see p. 84).

[94] D. Potts, G. Steidl, and M. Tasche. "Fast Fourier Transforms for Nonequispaced Data: A Tutorial". In: *Modern Sampling Theory: Mathematics and Applications*. Boston, MA: Birkhäuser Boston, 2001, pp. 247–270. ISBN: 978-1-4612-0143-4. DOI: 10.1007/978-1-4612-0143-4_12 (see pp. 58, 69).

[95] M. Rais. "Fast and accurate image registration. Applications to on-board satellite imaging." Theses. Université Paris-Saclay, 2016. URL: https://tel.archives-ouvertes.fr/tel-01485321 (see p. 171).

[96] M. Reed and B. Simon. *Methods of Modern Mathematical Physics: Vol.: 1.: Functional Analysis*. ISBN 0125850506. Academic press, 1972 (see p. 78).

[97] I. Rey Otero and M. Delbracio. "Anatomy of the SIFT Method". In: *Image Processing On Line* 4 (2014), pp. 370–396. DOI: 10.5201/ipol.2014.82 (see pp. 162, 181).

[98] I. Rey Otero and M. Delbracio. "Computing an Exact Gaussian Scale-Space". In: *Image Processing On Line* 6 (2016), pp. 8–26. DOI: 10.5201/ipol.2016.117 (see p. 84).

[99] V. Rudakova and P. Monasse. "Precise Correction of Lateral Chromatic Aberration in Images". In: *Image and Video Technology*. Ed. by Reinhard Klette, Mariano Rivera, and Shin'ichi Satoh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 12–22. ISBN: 978-3-642-53842-1. DOI: 10.1007/978-3-642-53842-1_2 (see pp. 263, 265, 275).

[100] W. Rudin. *Real and complex analysis*. Tata McGraw-Hill Education, 1987 (see p. 60).

[101] N. Sabater, J.M. Morel, and A. Almansa. "How Accurate Can Block Matches Be in Stereo Vision?" In: *SIAM Journal on Imaging Sciences* 4 (2011), p. 472. DOI: 10.1137/100797849 (see pp. 15, 35).

[102] J. Sánchez Pérez, N. Monzón López, and A. Salgado de la Nuez. "Robust Optical Flow Estimation". In: *Image Processing On Line* 3 (2013), pp. 252–270. DOI: 10.5201/ipol.2013.21 (see p. 174).

[103] J. Sánchez Pérez, E. Meinhardt-Llopis, and G. Facciolo. "TV-L1 Optical Flow Estimation". In: *Image Processing On Line* 3 (2013), pp. 137–150. DOI: 10.5201/ipol.2013.26 (see p. 174).

[104] J. Sánchez. "The Inverse Compositional Algorithm for Parametric Registration". In: *Image Processing On Line* 6 (2016), pp. 212–232. DOI: 10.5201/ipol.2016.153 (see pp. 24, 44, 162, 165–167, 170).

[105] I.J Schoenberg. "Cardinal Interpolation and Spline Functions". In: *Journal of Approximation theory* 2.2 (1969), pp. 167–206. DOI: 10.1016/0021-9045(69)90040-9 (see p. 104).

[106] I.J Schoenberg. "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions". In: *I.J Schoenberg Selected Papers*. Springer, 1988, pp. 3–57. DOI: 10.1007/978-1-4899-0433-1_1 (see pp. 101, 103).

[107] L. Schwartz. "Théorie des distributions". In: *Actualités Scientifiques et Industrielles, Institut de Mathématique, Université de Strasbourg* 1 (1966), p. 2. DOI: 10.1090/S0002-9904-1952-09555-0 (see pp. 58, 77).

[108] C.E. Shannon. "Communication in the Presence of Noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21. DOI: 10.1109/JRPROC.1949.232969 (see p. 58).

[109] L. Simon and JM Morel. "Influence of Unknown Exterior Samples on Interpolated Values for Band-limited Images". In: *SIAM Journal on Imaging Sciences* 9.1 (2016), pp. 152–184. DOI: 10.1137/140978338 (see pp. 58, 101).

[110] E. P. Simoncelli. "Design of multi-dimensional derivative filters". In: *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*. Vol. 1. IEEE. 1994, pp. 790–794. DOI: 10.1109/ICIP.1994.413423 (see p. 171).

[111] E.P. Simoncelli and W.T. Freeman. "The steerable pyramid: A flexible architecture for multi-scale derivative computation". In: *Proceedings of the International Conference on Image Processing*. Vol. 3. IEEE. 1995, pp. 444–447. DOI: 10.1109/ICIP.1995.537667 (see pp. 71, 76, 84).

[112] B. Steinert, H. Dammertz, J. Hanika, and H. Lensch. "General spectral camera lens simulation". In: *Computer Graphics Forum*. Vol. 30. 6. Wiley Online Library. 2011, pp. 1643–1654. URL: https://jo.dreggn.org/home/2011_lens_simulation.pdf (see p. 260).

[113] R.S. Strichartz. *A guide to distribution theory and Fourier transforms*. ISBN 0849382734. World Scientific, 2003 (see pp. 58, 77).

[114] R. Szeliski. *Computer vision: algorithms and applications*. https://dx.doi.org/10.1007/978-1-84882-935-0. Springer Science & Business Media, 2010 (see pp. 24, 44).

[115] R. Szeliski et al. "Image alignment and stitching: A tutorial". In: *Foundations and Trends® in Computer Graphics and Vision* 2.1 (2007), pp. 1–104 (see pp. 24, 44).

[116] H. Takeda, S. Farsiu, and P. Milanfar. "Higher order bilateral filters and their properties". In: *Computational Imaging V*. Vol. 6498. International Society for Optics and Photonics. 2007, 64980S. DOI: 10.1117/12.714507 (see p. 206).

[117] H. Takeda, S. Farsiu, and P. Milanfar. "Kernel Regression for Image Processing and Reconstruction". In: *IEEE Transactions on Image Processing* 16.2 (2007), pp. 349–366. ISSN: 1057-7149. DOI: 10.1109/TIP.2006.888330 (see pp. 16, 28, 35, 36, 48, 201, 202, 205–208).

[118] H. Takeda, S. Farsiu, and P. Milanfar. "Robust kernel regression for restoration and reconstruction of images from sparse noisy data". In: *Image Processing, 2006 IEEE International Conference on*. IEEE. 2006, pp. 1257–1260. DOI: 10.1109/ICIP.2006.312573 (see pp. 16, 35, 202).

[119] H. Takeda, P. Milanfar, M. Protter, and M. Elad. "Super-Resolution Without Explicit Subpixel Motion Estimation". In: *IEEE Transactions on Image Processing* 18.9 (2009), pp. 1958–1975. ISSN: 1057-7149. DOI: 10.1109/TIP.2009.2023703 (see pp. 15, 35, 206).

[120] O. Tange. "GNU Parallel - The Command-Line Power Tool". In: *;login: The USENIX Magazine* 36.1 (2011), pp. 42–47. DOI: 10.5281/zenodo.16303 (see p. 277).

[121] O. Tange. *GNU Parallel 2018*. Ole Tange, 2018. DOI: 10.5281/zenodo.1146014 (see p. 277).

[122] P. Thévenaz, U. Ruttimann, and M. Unser. "A pyramid approach to subpixel registration based on intensity". In: *IEEE transactions on image processing* 7.1 (1998), pp. 27–41. DOI: 10.1109/83.650848 (see pp. 24, 44, 161, 162).

[123] P. Thévenaz, T. Blu, and M. Unser. "Interpolation Revisited [Medical Images Application]". In: *Medical Imaging, IEEE Transactions on* 19.7 (2000), pp. 739–758. DOI: 10.1109/42.875199 (see pp. 17, 37, 103, 106, 142).

[124] C. Tomasi and R. Manduchi. "Bilateral filtering for gray and color images". In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. 1998, pp. 839–846. DOI: 10.1109/ICCV.1998.710815 (see pp. 202, 206).

[125] R. Tsai and T. Huang. "Multiframe image restoration and registration". In: *Advances in computer vision and Image Processing* 1.2 (1984), pp. 317–339 (see pp. 15, 35).

[126] M. Unser. "Splines: A Perfect Fit for Signal and Image Processing". In: *Signal Processing Magazine, IEEE* 16.6 (1999), pp. 22–38. DOI: 10.1109/79.799930 (see pp. 101–103).

[127] M. Unser, A. Aldroubi, and M. Eden. "B-spline Signal Processing. I. Theory". In: *Signal Processing, IEEE Transactions on* 41.2 (1993), pp. 821–833. DOI: 10.1109/78.193220 (see pp. 101, 103).

[128] M. Unser, A. Aldroubi, and M. Eden. "B-spline Signal Processing. II. Efficiency Design and Applications". In: *Signal Processing, IEEE Transactions on* 41.2 (1993), pp. 834–848. DOI: 10.1109/78.193221 (see p. 101).

[129] M. Unser, A. Aldroubi, and M. Eden. "Fast B-spline Transforms for Continuous Image Representation and Interpolation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.3 (1991), pp. 277–285. DOI: 10.1109/34.75515 (see pp. 20, 40, 101, 103).

[130] P. Vandewalle, S. Süsstrunk, and M. Vetterli. "A frequency domain approach to registration of aliased images with application to super-resolution". In: *EURASIP journal on advances in signal processing* 2006.1 (2006), p. 071459. DOI: 10.1155/ASP/2006/71459 (see p. 188).

[131] P. Vandewalle, K. Krichane, D. Alleysson, and S. Süsstrunk. "Joint demosaicing and super-resolution imaging from a set of unregistered aliased images". In: *Electronic Imaging 2007*. International Society for Optics and Photonics. 2007, 65020A–65020A. DOI: 10.1117/12.703980 (see pp. 15, 16, 35, 188, 191, 258).

[132] G. Wallace. "The JPEG still picture compression standard". In: *Communications of the ACM* 34.4 (1991), pp. 30–44. DOI: 10.1109/30.125072 (see pp. 249, 258).

[133] E. T. Whittaker. “XVIII.—On the functions which are represented by the expansions of the interpolation-theory”. In: *Proceedings of the Royal Society of Edinburgh* 35 (1915), pp. 181–194. DOI: 10.1017/S0370164600017806 (see p. 58).

[134] Y. Wu, J. Lim, and M. Yang. “Online object tracking: A benchmark”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 2411–2418. DOI: 10.1109/CVPR.2013.312 (see pp. 24, 44).

[135] J. Yang, J. Wright, T. S. Huang, and Y. Ma. “Image Super-Resolution Via Sparse Representation”. In: *IEEE Transactions on Image Processing* 19.11 (2010), pp. 2861–2873. ISSN: 1057-7149. DOI: 10.1109/TIP.2010.2050625 (see pp. 15, 35).

[136] L. Yuan and J. Sun. “High quality image reconstruction from RAW and JPEG image pair”. In: *2011 International Conference on Computer Vision*. 2011, pp. 2158–2165. DOI: 10.1109/ICCV.2011.6126492 (see p. 188).

[137] B. Zitová and J. Flusser. “Image registration methods: a survey”. In: *Image and Vision Computing* 21.11 (2003), pp. 977 –1000. ISSN: 0262-8856. DOI: 10.1016/S0262-8856(03)00137-9 (see pp. 24, 44).

Titre: Formation d'image à partir d'une grande séquence d'images RAW: performance et précision

Mots clefs: Formation d'image • Débruitage • Dématriçage • Super-résolution • Haute précision • Interpolation • B-spline • Recalage • Fusion d'images • Ajustement de données irrégulièrement échantillonnées

Résumé:
Le but de cette thèse est de construire une image couleur de haute qualité, contenant un faible niveau de bruit et d'aliasing, à partir d'une grande séquence (e.g. des centaines) d'images RAW prises avec un appareil photo grand public. C'est un problème complexe nécessitant d'effectuer à la volée du dématriçage, du débruitage et de la super-résolution. Les algorithmes existants produisent des images de haute qualité, mais le nombre d'images d'entrée est limité par des coûts de calcul et de mémoire importants. Dans cette thèse, nous proposons un algorithme de fusion d'images qui les traite séquentiellement de sorte que le coût mémoire ne dépend que de la taille de l'image de sortie. Après un pré-traitement, les images mosaïquées sont recalées en utilisant une méthode en deux étapes que nous introduisons. Ensuite, une image couleur est calculée par accumulation des données irrégulièrement échantillonnées en utilisant une régression à noyau classique. Enfin, le flou introduit est supprimé en appliquant l'inverse du filtre équivalent asymptotique correspondant (que nous introduisons). Nous évaluons la performance et la précision de chaque étape de notre algorithme sur des données synthétiques et réelles. Nous montrons que pour une grande séquence d'images, notre méthode augmente avec succès la résolution et le bruit résiduel diminue comme prévu. Nos résultats sont similaires à des méthodes plus lentes et plus gourmandes en mémoire. Comme la génération de données nécessite une méthode d'interpolation, nous étudions également les méthodes d'interpolation par polynôme trigonométrique et B-spline. Nous déduisons de cette étude de nouvelles méthodes d'interpolation affinées.

Title: Image Formation from a Large Sequence of RAW Images: Performance and Accuracy

Keywords: Image formation • Denoising • Demosaicking • Super-resolution • High-precision • Interpolation • B-spline • Registration • Image fusion • Irregularly sampled data fitting

Abstract:
The aim of this thesis is to build a high-quality color image, containing a low level of noise and aliasing, from a large sequence (e.g. hundreds or thousands) of RAW images taken with a consumer camera. This is a challenging issue requiring to perform on the fly demosaicking, denoising and super-resolution. Existing algorithms produce high-quality images but the number of input images is limited by severe computational and memory costs. In this thesis we propose an image fusion algorithm that processes the images sequentially so that the memory cost only depends on the size of the output image. After a preprocessing step, the mosaicked (or CFA) images are aligned in a common system of coordinates using a two-step registration method that we introduce. Then, a color image is computed by accumulation of the irregularly sampled data using classical kernel regression. Finally, the blur introduced is removed by applying the inverse of the corresponding asymptotic equivalent filter (that we introduce). We evaluate the performance and the accuracy of each step of our algorithm on synthetic and real data. We find that for a large sequence of RAW images, our method successfully performs super-resolution and the residual noise decreases as expected. We obtained results similar to those obtained by slower and memory greedy methods. As generating synthetic data requires an interpolation method, we also study in detail the trigonometric polynomial and B-spline interpolation methods. We derive from this study new fine-tuned interpolation methods.