Background reconstruction from multiple images

Xiaoyi Yang

NNT: 2018SACLT020

Background Reconstruction From Multiple Images

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom ParisTech

École doctorale n°580: sciences et technologies
de l'information et de la communication (STIC)
Spécialité de doctorat: traitement du signal et des images

Thèse présentée et soutenue à Paris, le 18 décembre 2018, par

Mme Xiaoyi Yang

Composition du Jury :

M. Liming Chen Professeur, École Centrale de Lyon, LIRIS	Président
Mme Françoise Dibos Professeur Emérite, Université Paris 13, Institut Galilée, LAGA, L2TI	Rapporteur
Mme Valérie Gouet-Brunet Directrice de Recherche, l'IGN, MATIS	Rapporteur
M. Antoine Manzanera Enseignant-Chercheur, ENSTA ParisTech, U2IS	Examinateur
M. Henri Maître Professeur Emérite, Télécom ParisTech, LTCI	Directeur de thèse
M. Yohann Tendero Maître de Conférences, Télécom ParisTech, LTCI	Co-Directeur de thèse
M. Yann Gousseau Professeur, Télécom ParisTech, LTCI	Co-Directeur de thèse, Invité

Thèse de doctorat

Titre: Reconstruction d'une scène masquée à partir de multi-image

Mots clés: reconstruction d'image, suppression de masque, estimation d'arrière-plan.

Résumé: La problématique générale de cette thèse est de reconstituer la scène de fond à partir d'une séquence d'images en présence de masques d'avant-plan. Nous nous sommes intéressés aux méthodes pour détecter ce qui constitue le fond ainsi que les solutions pour corriger les parties cachées et les distorsions géométrique et chromatique introduites lors de la photographie.

Une série de processus est proposée, dont la mise en œuvre comporte dans l'ordre l'alignement géométrique, le réglage chromatique, la fusion des images et la correction des défauts.

Nous nous plaçons dans l'hypothèse où le fond est porté sur une surface plane. L'alignement géométrique est alors réalisé par calcul de l'homographie entre une image quelconque et l'image qui sert de référence, suivi d'une interpolation bilinéaire.

Le réglage chromatique vise à retrouver un même contraste dans les différentes images. Nous proposons de modéliser la mise en correspondance chromatique entre images par une approximation linéaire dont les paramètres sont déterminés par les résultats de la mise en correspondance des points de contrôle (SIFT).

Ces deux étapes sont suivies par une étape de fusion. Plusieurs techniques sont comparées.

La première proposition est d'étendre la définition de la médiane dans l'espace vectoriel. Elle est robuste lorsqu'il y a plus de la moitié des images qui voient les pixels d'arrière-plan. En outre, nous concevons un algorithme original basé sur la notion de clique. Il permet de détecter le plus grand nuage de pixels dans l'espace RGB. Cette approche est fiable même lorsque les pixels d'arrière-plan sont minoritaires.

Lors de la mise en œuvre de ce protocole, on constate que certains résultats de fusion présentent des défauts de type flou dus à l'existence d'erreurs d'alignement géométrique. Nous proposons donc un traitement complémentaire. Il est basé sur une comparaison entre le résultat de fusion et les images alignées après passage d'un filtre gaussien. Sa sortie est un assemblage des morceaux très détaillés d'image alignés qui ressemblent le plus au résultat de fusion associés.

La performance de nos méthodes est évaluée par un ensemble de données contenant de nombreuses images de qualités différentes. Les expériences confirment la fiabilisé et la robustesse de notre conception dans diverses conditions de photographie.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Title: Background reconstruction from multiple images

Keywords: image reconstruction, mask removal, background estimation.

Abstract: The general topic of this thesis is to reconstruct the background scene from a burst of images in presence of masks. We focus on the background detection methods as well as on solutions to geometric and chromatic distortions introduced during photography.

A series of process is proposed, which consists of geometric alignment, chromatic adjustment, image fusion and defect correction.

We consider the case where the background scene is a flat surface. The geometric alignment between a reference image and any other images in the sequence, depends on the computation of a homography followed by a bilinear interpolation.

The chromatic adjustment aims to attach a similar contrast to the scene in different images. We propose to model the chromatic mapping between images with linear approximations whose parameters are decided by matched pixels of SIFT .

These two steps are followed by a discussion on image fusion. Several methods have been compared.

The first proposition is a generation of typical median filter to the vector range. It is robust when more than half of the images convey the background information. Besides, we design an original algorithm based on the notion of clique. It serves to distinguish the biggest cloud of pixels in RGB space. This approach is highly reliable even when the background pixels are the minority.

During the implementation, we notice that some fusion results bear blur-like defects due to the existence of geometric alignment errors. We provide therefore a combination method as a complementary step to ameliorate the fusion results. It is based on a comparison between the fusion image and other aligned images after applying a Gaussian filter. The output is a mosaic of patches with clear details issued from the aligned images which are the most similar to their related fusion patches.

The performance of our methods is evaluated by a data set containing extensive images of different qualities. Experiments confirm the reliability and robustness of our design under a variety of photography conditions.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Contents

Chapter 1

Introduction

Computational photography is a very active research field, and one of its branches, occlusion removal, has a vast scope of prospects in terms of image editing and video surveillance, motion detection and image understanding. However, due to the impacts of critical conditions such as illumination changes and motionless foreground, the actual approaches are either less robust or highly conditional. We are hence motivated to work on this promising topic.

My thesis is dedicated to find out a reliable occlusion removal method along with the solution of problems which will occur during the process. A burst of images taken at different camera positions are used so that every parts of the background are revealed at least one time. We try to realize an automatic occlusion detection without assumptions on the mask shape, color or motion.

This chapter is devoted to a general introduction of our work. We present here a modeling of problem and the difficulties we may meet. Then we give a description on how we organize the manuscript.

1.1 Challenges

Our purpose is to investigate the possibility to reconstruct the background scene with a small number of images captured by a camera or continuous frames issued from a video. This challenge is well illustrated by a popular example of viewing an animal through a cage. More attractive applications may be found when dealing with some general cases such as a landscape seen through foliage, a sport event through attendance foreground, a monument through tourists, etc. The choice

of method is critical as the separation of occlusion should be proceeded without awareness of the mask type.

In previous works (see Chapter 2), images or video frames are usually captured by stand-still devices to avoid the alignment problem. These methods loose their robustness under situations of moving background, camera jitter, motionless foreground objects, etc. Besides, the number of images impacts determinately the scope of application. A single image is not enough to provide information behind big masks. On the contrary, a large quantity of images required by many estimation models are difficult to collect at one time by hand-held cameras.

Based on these considerations, we plan to add alignment process in our work and we focus on the problems with a small set of source images. As the start of our work, we define a context with the following requirements which will be partially released later:

1) The background to restore is plane and quasi-Lambertian.
2) Images in the sequence (typically 5) reveal the entire target scene several times depending on the combined motion of the photographer and the occlusions.
3) The angles of view do not change too much among images.
4) The camera parameters are manually fixed during photography.
5) Images should be captured in a very short time.

The purpose of this thesis is not limited to bringing solution to background reconstruction. Another objective is to determine a sound methodology that is able to adapt itself to the specific conditions of image quality in terms of contrast, colors, relative positioning and movement of objects. Factors that may disturb a good functioning of the background reconstruction process include but are not limited to:

- **Illumination changes.** The change of lighting conditions, caused for example by the drifting clouds or by a light switch in an indoor scene, results in color changes in images.
- **Automatic adjustments of camera.** Many internal adjustments of cameras such as white balance, auto focus and auto brightness control may lead to color differences of the same view in different images.
- **Optical aberration.** The imperfection of lens deviates from the model of a perfect optical system. Images captured by the lens with defects become blurred or distorted depending on the type of aberration.

These problems should all be taken into consideration during our design of process. Our work is expected to manage a general case where the user takes casually

a few pictures of a view with occlusion with a hand-held equipment. The process should output a perfect mask-removed result in a short time.

1.2 Overview

The main portion of this thesis consists of three parts which are organized in a logical order, i.e. image alignment, background pixel selection and experiments. Chapters under these parts are as follows:

Part I: Image Alignment

Chapter 3: Geometrical Alignment. Based on the assumption of a planar target scene, we suppose that images can be mapped onto each other by homography. We apply a standard alignment algorithm between the selected reference image and every other images in the sequence. The process consists of three steps which are: the search of matched pixels using SIFT features, the computation of homography based on DLT method with RANSAC selected samples, the re-sampling of images on the reference grid through bilinear interpolation. The experimental errors are controlled within a satisfying range.

Chapter 4: Photometric Correction. We notice the existence of contrast differences in images caused by illumination variations and/or the automatic adjustments of camera. Based on a study of the camera processing pipeline, we set up the color formation model and finally the color transfer relationship between images. These color transfer functions are approximated by several linear models whose parameters are estimated with the matched pixels selected using again RANSAC strategy. Experiments show that a quadratic polynomial is the best model. It helps to reduce significantly the RMSE between images.

Part II: Fusion Methods

Chapter 5: Median Filter. The robust performance of median filtering makes it a simple but good choice for pixel-level image fusion. While the classical median filter is limited in gray-scale, we propose either to filter the scalars degenerated from RGB values or to extend the median definition to 3D space where the median vector minimizes the sum of its Euclidean distances to other vectors. Experiments confirm that the extended median filter performs better than its channel-wise counterpart since the RGB vectors convey more information than scalars. The results are fairly satisfying as long as more than a half of the pixels belong to the background.

Chapter 6: Meaningful Clique. The strong constraint of median filtering on background pixel quantity motivates us to design another process for pixel selection. Notice that the background pixels which share similar RGB values stay close to each other while the mask pixels scatter in RGB space. We design a clique based algorithm to figure out the biggest cloud of similar vectors in this space. The pixels in this cloud are supposed to convey the background information. This method provides the best performance in terms of quality and reliability compared with median filtering and RPCA method. It shows a good robustness in cases where background pixels are the minority.

Chapter 7: Combination method. Blur-like defects occur in the fusion results when we test the process with images captured by lens of poor or average quality. They are caused by the geometrical alignment errors introduced by the lens optical aberrations. Our idea is to replace the mosaic fusion results by the clear patches of an aligned sequence. The source pixels are estimated according to their Euclidean distance to the pixels in the fusion results. To eliminate the effects of blur, the resolution of images is degraded through a Gaussian filter before comparison. This approach is an efficient alternative to median or clique filtering in case of serious distortions.

Part III: Experiments

Chapter 8: Experimental performances. We test the performance of our methods through an extensive experimental study with images of different qualities. Our data set includes synthetic sequences and real sequences captured by prime or zoom lenses with manually controlled parameters or by embodied lenses of a smart phone with automatic adjustment of lighting conditions. Also, we estimate the effects of an image processing software (*DxO Optics Pro11*) on mask removal results. Experiments confirm the reliability and the robustness of our design.

A part of work in this thesis leads to the publication:
A fast algorithm for occlusion detection and removal. Xiaoyi Yang, Yann Gousseau, Henri Maitre, Yohann Tendero. *International Conference on Image Processing (ICIP), 2018.*

A webpage displaying the experimental results can be found at this address:
https://combinaison-images.telecom-paristech.fr

Chapter 2

Related Work

A variety of works has been done on mask removal. All processes go through the stages of mask/foreground detection and background restoration which are achieved either by independent algorithms or by a synthetic method. In any cases, the detection technology, as a link between the real data and the mathematical model, plays a decisive role in the choice of approaches.

Hence, we classify the previous works according to their mask/foreground detection methods, i.e. manual selection, photographic technology and automatic evaluation. We will end up with a search of tools for background detection, which is the our domain of interests.

2.1 Manual selection of masks

These approaches require the user to select manually the areas of occlusion in the images [CPT03] or in the frames of videos [PSB05] where the pixel information will be discarded. When background information is missing, the reconstruction of missing parts is usually achieved by inpainting methods in which the sources are found within a single frame or the whole video.

The searching process in many recent inpainting approaches relies on patch similarity that takes into account both structural and textural consistency of the nearby region. The missing background is completed either gradually from the border to the center [CPT04, WR06, YWW+14, WLP+14], or is generated globally by iteratively minimizing an energy function [Kom06, KT07, KT07, NAF+13, LAGM17].

However, the performance of these methods is limited when the gaps to be completed are large and of varying textures. Many problems such as unconnected edges, smoothing and blurring artifacts may occur in the results [GLM14, Tha15].

2.2 Special photographic technology

This section involves the methods relying on special experimental technologies. They distinguish occlusions based on their distances to the camera, which can be measured by varying the focal length or by stereo estimation. However, all these designs are based on stringent experimental conditions.

2.2.1 Multi-focus

The multi-focus methods address a case where the occluders are: 1) thin (such as fence and branches); 2) distant to the background; 3) out-of focus in the target image. The irradiance at a pixel in an image is supposed to be a weighted sum of the blurred occluder radiance and the background radiance [AFM98, HK07].

In this case, [YMK10, YTKA12, YKA13] propose to reverse the projection of blurring masks by eliminating occluders. It involves an image focusing on the background and two images focusing on the occluders with/without flashlight captured by a finite aperture lens. The flashlight, introducing the radiance difference on the occluders, helps to extract their information. This information is then used to remove masks in the background-clear image. This method is later tested by [MLS10] with real camera images. The model and its parameters are discussed in [FS03]. Still, camera calibration and parameter decision are always difficult to carry out.

2.2.2 Light field equipments

Different from conventional cameras, the light field equipments provide both intensity and direction information of the incident ray. They allow to proceed the 3D reconstruction as well as occlusion removal by analyzing the depth of objects in sight. Two kinds of cameras are usually mentioned:

Stereo cameras This is the general name of cameras with two (binocular) or multiple lenses in front of their independent image sensors. The camera deduces several images with parallax at one time. The early occlusion researches

focus mostly on binocular stereo [GLY92, GLY95, EW02, BW11] while the lack of information causes many problems such as holes in the depth map, failure estimation on large occlusions, etc [SBBB86, KKPD04]. Therefore, more and more works are found in the domain of multiple lenses stereo, ranging from several lens to a camera array [LH96, WJV+05, VWJL04].

Plenoptic cameras Represented by Lytro [Ng17] and Raytrix [PW12], plenoptic cameras can be regarded as an application of camera array. They are made by placing an array of micro-lenses between the conventional lens and the sensor of the camera. The light field estimation is similar to that of camera arrays in [LH96], while the results are single light-field images after processing.

For all camera types, the idea lies in creating a depth map and selecting the elements of deep depths for image fusion. For every two identical parallel calibrated cameras, the depth of world points equal to $d = \frac{fT}{\delta \mathbf{x}}$, where f is the focal length, T is the baseline between cameras and $\delta\mathbf{x}$ is the coordinate difference (disparity) between the related pixels on images [Aya91, Zha13]. Given a pixel in one image, its related pixel in another image can be computed with d, then the search of depth is converted to a search of matched pixels (mostly replaced by patches). This process, proceeded along the epipolar line or in a global range, aims to find the patch that minimizes an energy function established in Markov random field [HS07], which can be generally represented as: $E(d, I) = E_{match}(d, I) + E_{constraint}(d)$. The terms E_{match} and $E_{constraint}$ are matching and constraint costs that estimate respectively the intensity and depth differences in the neighborhood. The depth map is set up with $(d, I) = \arg\min_{d,I} E$.

The improvement of energy function has become an important research field among the works of mask removal. Apart from the traditional SSA, SAD functions, the constraints are also designed based on monotonicity of pixel order [GLY95], the uniqueness of disparity maps [ZK00], the entropy [VLS+06], the visibility of occlusion [SLKS05], the color similarity and spatial distance [YK06, JM15]. Further more, there are many ameliorating methods such as adding a k-means cluster to discard some mask pixels in multi-view stereo [DJ10, XSZ17], optimizing the results by graph-cut segmentation [KZ01, VETC07, XWSZ14].

Even so, the results of stereo estimation are often inaccurate, The matching process is often affected by factors such as reflection, foreshortening, optical distortion, low texture and repetitive/ambiguous patterns.

2.3 Automatic evaluation

Here we focus on the mask removal approaches with no requirements on human intervention or special equipments. These methods set up hypotheses on occlusion or background based on their characteristics. We divide them into the methods detecting masks and the methods modeling background.

2.3.1 Mask detection

Compared with background, occlusions are more often chosen as the object to be detected since they can be easily described by some properties. The most common strategies make use of continuity of intensity variation or special hypotheses on certain type of masks.

Continuous change of frames These methods rely on the gradual variation of pixel intensity over time. Therefore, the input images are required to be a series of continuous frames or several images following a consistent varying trend.

Optical flow Optical flow aims at computing a motion information at each pixel in an image. It is based on the brightness constancy hypothesis which assume that the intensity of brightness flows does not change from one frame to another (see e.g.[HS81, LK+81, BSL+11]). If the brightness at (x, y) and at time t is denoted by $I(x, y, t)$, then we have: $\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$, with $u = \frac{\partial x}{\partial t}$, $v = \frac{\partial y}{\partial t}$ the velocity elements in the direction x and y. With an additional smoothness constraint, the velocity field can be further computed (see e.g.[BWS05]). The optical flow helps to extract occlusion from background since they are usually at different distances to the moving camera. However, an exact discrimination of their velocity is always difficult. The easiest way to achieve it is to set up a threshold on velocity values [MLY14]. Besides, [GTCS+01] attempts to find out the stable regions all along the time. [YBF04, YHCB05] propose a Bayesian framework and estimate the likelihood of velocity values being the motion layers. [DJB14] provides constraints on the camera coordinate system to refine the motion analysis.

Kalman filtering The Kalman filter is an optimal estimator for a linear system affected by Gaussian white noises during process and measurement. At time t_i, it estimates the current state $\hat{X}(t_i)$ from the previous state $\hat{X}(t_{i-1})$ and the current measurement $\tilde{X}(t_i)$ by computing: $\hat{X}(t_i) = A\hat{X}(t_{i-1}) + K(t_i)[\tilde{X}(t_i) - CA\hat{X}(t_{i-1})]$, where A and C are respectively the transition of system and measurement, $K(t_i)$ is the Kalman gain updated by minimizing the mean-square of residual error

[AM79, VAS82, BH+92]. For mask detection, $\hat{X}(t_i)$ and $\tilde{X}(t_i)$ are assigned to the predicted and actual intensities of the current frame. If their difference surpasses threshold, a change is supposed to be detected and the foreground/background mark reverses [KWM94]. This design is not robust to the illumination changes of background. To fix this problem, [RMK95] applies an adaptive threshold based on the current background variance; [Z+03] proceeds filtering after principle component analysis; [GZX01] generates the temporal filter to spatial range; [MMSZ05] adds the estimation of an illumination map.

Hypothesis on occlusion Some methods are designed according to the properties of some special occlusions. Two common hypotheses are the approx-regular patterns and the randomly repeating particles in image.

Regularity Fence is always discussed as the typical regular occlusion in many works. Its detection is usually based on the search of periodic patterns and specific orientation. The early approaches [HLEL06, PCL08] propose to select the most repetitive patches, along with their directions. Then, the growth of lattice progresses along the directions according to a likelihood estimation of patches. The obtained area, including fence and its neighboring background, will go through a segmentation of pixels by k-means clustering [LBHL08] or by support vector machine [PBCL10] to extract the exact region of the fence. In later works, the repetitive stick-like structure is extracted directly by a linear clustering in [YWLH15] or by Bayes classifier in [FMG16]. The background reconstruction is finally achieved by an inpainting method.

Repeatability Although rain and snow are distributed randomly in the spatial domain, their repeating forms yield a high frequency contents in their spectrum in Fourier domain [LZ03]. Methods in different works simulate in fact certain kinds of low-pass filters that reserve detail components among high frequency elements. The most straightforward expression is found in [ZLG+13] where an edge-preserving filter is used to smooth the image. Further estimations in [CH13, SO13] demonstrate that the rain patches in high frequency domain are expressed as low-rank matrix due to the repetition of similar appearance which can be removed by matrix decomposition. Similarly, [KLF12, SFW14] apply dictionary learning strategy to separate sparse components from low-rank elements. A more detailed description of rain assumes it to be similar to a Gaussian kernel elongated along motion, then [BKN07, BNK10] remove the rain by a subtraction of its corresponding filter in the Fourier domain.

2.3.2 Background detection

Directly detecting the background is the most straightforward track for mask removal. However, this path is difficult due to the diversity and complexity of backgrounds. The current works always need a large quantity of continuous frames for the estimation. The process is rather time-consuming while the result is often not accurate.

Motivated to work on this direction, we focus particularly on the approaches dealing with a small number of input images. Difficult as it is, this task can still be accomplished by analyzing the static-state of background based on its highly frequent appearance in a burst of aligned images. In order to achieve this target, the following tools can be used:

Principal component analysis (PCA) $\mathbf{\Phi} = \{\mathbf{I}_1, \ldots, \mathbf{I}_N\}$ denotes a set of RGB vectors issued from the same pixel position of N aligned frames. The effects of foreground values, which are random and sparse, can be eliminated by modeling the primary distribution of all pixel values [MBV12]. The low-rank principal vectors, containing the discriminative information of background, are computed by decomposing original vectors along the principal component direction e_0 which retains the largest variance:

$$e_0 = \arg\max_e \frac{1}{N} \sum_{i=1}^{N} [\, \mathbf{I}_i e - \bar{\mathbf{I}} \,]^2, \quad \bar{\mathbf{I}} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{I}_j e.$$

The value of e_0 can be obtained by SVD of data matrix [Jol11]. For the sake of on-line application, [Row98] proposes to compute e_0 using EM strategy until convergence ($\mathbf{x}_i$ is the mean-removed value of $\mathbf{I}_i$):

$$\tilde{e} \leftarrow \sum_{i=1}^{N} (\mathbf{x}_i^T e_{t-1}) \mathbf{x}_i \quad \textit{and} \quad e_t \leftarrow \frac{\tilde{e}}{\|\tilde{e}\|}.$$

However, the classical principal component analysis is often corrupted by the extreme outliers in data set. [CLMW11] starts the topic of robust principal component analysis which is based on the low-rank and sparse decomposition. They prove that if M is a $3 \times N$ matrix whose columns are made of RGB vectors of $\mathbf{\Phi}$, it may be decomposed as:

$$M = L_0 + S_0,$$

where L_0 is a low-rank matrix conveying the background intensities of each frame and S_0 is a sparse matrix of the foreground scene (noise). The decomposition is

solved by estimating the principal component pursuit (PCP):

$$\begin{aligned} \textit{minimize} \quad & \|L\|_* + \lambda\|S\|_1, \\ \textit{subject to} \quad & L + S = M, \end{aligned}$$

where $\lambda = 1/\sqrt{\max(3, N)}$, $\|L\|_* := \sum_i \sigma_i(L)$ is the sum of singular values of L and $\|S\|_1 = \sum_{ij} \|S_{ij}\|$ denotes the l_1-norm of S. A popular solution of PCP problem recovers L_0 and S_0 depending on the augmented Lagrange multiplier (ALM) [YY09, CGL+09, LCM10]:

$$(L_0, S_0) = \arg\min_{L,S} \|L\|_* + \lambda\|S\|_1 + tr(Y(M-L-S)) + \frac{\mu}{2}\|M - L - S\|^2,$$

The Lagrange multiplier matrix $Y_{K+1} = Y_k + \mu(M - L_k - S_k)$ is updated iteratively, and μ is a chosen parameter.

Many attempts have been made on RPCA [GBZ12, GCW14, JBJ15, CWS+16, JJMB16], but most of them suffer from while the expensive computation. Recently in [HFB14], Grassmann average is introduced as a subspace model to replace ordinary PCA. We consider that this method, benefiting from both computational efficiency and robustness, is an efficient alternative to the work we will develop later for background detection.

[HFB14] proves that when the vectors in $\mathbf{\Phi}$ are sampled from a Gaussian distribution, their Grassmann average coincides with their first principal component. We continue using $\mathbf{x}_i$ to represent mean-removed value of $\mathbf{I}_i$, the Grassmann average q is computed by iterating:

$$\omega_i \leftarrow sgn(\mathbf{x}_i^T q_{t-1})\|\mathbf{x}_i\|,$$

$$\mu(\omega_{1:N}, \mathbf{x}_{1:N}) \leftarrow (\sum_{i=1}^{N} \omega_i)^{-1} \sum_{i=1}^{N} \omega_i \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|} \tag{2.1}$$

$$q_t \leftarrow \frac{\mu(\omega_{1:N}, \mathbf{x}_{1:N})}{\|\mu(\omega_{1:N}, \mathbf{x}_{1:N})\|}. \tag{2.2}$$

To calculate the average robustly, [Hub11] removes $p\%$ of largest and smallest values in data set and estimates a trimmed result. The computation of weighted average in Equation (2.1) no longer covers every vectors but selected ones, $\mu(\omega_{1:N}, \mathbf{x}_{1:N})$ in Equation (2.1) and Equation (2.2) should be rewritten as $\mu_{trim}(\omega_{1:N}, \mathbf{x}_{1:N})$. In all, a trimmed Grassmann average is used to estimate the components in each $\mathbf{x}_i$.

Median filtering Median filtering serves to find out the number that separate the greater and lesser halves of a set. Given a sorted scalar data set $\mathbf{S}$, its median is defined as (see Chapter 5):

$$med(\mathbf{S}) := \begin{cases} \mathbf{S}_{(\#\mathbf{S}+1)/2} & \#\mathbf{S} \in 2\mathbb{N}+1 \\ \frac{1}{2}\left(\mathbf{S}_{\#\mathbf{S}/2} + \mathbf{S}_{\#\mathbf{S}/2+1}\right) & \#\mathbf{S} \in 2\mathbb{N}. \end{cases}$$

The use of median for mask removal is based on two assumptions: 1) The image sequence has been aligned; 2) At each position, the number of background pixels exceeds half of the total number of pixels. A pixel-wise application on an image sequence ensures that filtering results belong to the background.

Temporal median filter has been widely used to create a single mosaic from several images. In early applications, median results are directly used as background for movement detection [MS95, TOF02], or as a layer for image reconstruction [WFZ02]. Later, filtering is often associated with other methods (pre-segmentation in [FdWE03] or post-inpainting in [KR07]) to get better results.

However, no obvious improvement has been done in terms of median filtering conception, applications have always been limited within scalar range. The recent works are found either in medical imaging [JOd+15], or in an independent implementation per channel [MP14, LPBVD15]. It is however very practical to adapt the gray-scale filtering into vector space (see Chapter 5).

Part I

Image Alignment

Chapter 3

Geometrical Alignment

To make use of elements from different images, the first problem is to compensate projective distortions. It is a common phenomenon that occurs when a scene of 3D space is mapped onto a 2D plane. Under such distortion, objects loose their original shape such that opposite sides of a rectangular intersect and circles become ellipses.

In our application, we want to pre-process the sequence so that the background scenes in different images superpose perfectly on each others. We proceed to a geometrical alignment as the first step of our process where an image is chosen as the reference and every other images are mapped on its coordinates. As explained earlier, the scene to restore is assumed to be planar. It is reasonable to express the mapping between two coordinate spaces as a homography. In order to achieve it, we rely on a classical pipeline where the local descriptors are selected and then the homography is decided by a RANSAC procedure.

This chapter consists of three sections. First, we give a review on SIFT features estimation, which is the standard process to extract matched pixels. Then, we focus on the homography and the method to map original image on a reference image grid. Finally, we give some results in our experiments.

3.1 Extraction of matched pixels

This step prepares samples (i.e.matched pixels) for further calculation of homography. It consists of two major steps: 1) selection of pixels and their features which are robust to geometric distortions. 2) matching process carried out by

estimating feature-similarity. We apply SIFT algorithm to achieve the first step as is presented in Section 3.1.1. We introduce the second step in Section 3.1.2

3.1.1 Chosen pixels and their invariant features by SIFT

Scale Invariant Feature Transform (SIFT) is a widely used image matching algorithm introduced by D.Lowe in 1999 [Low99]. It shows a strong robustness to scene deformation of images under slight changes of viewpoint which is exactly the case of our assumption. Although numerous detectors such as SURF [BTVG06] and ASIFT [MHB+10] have been proposed ever since, experiences show that SIFT is fast and accurate enough in our case. So we apply SIFT in our process to establish the geometric relationship between images. Here is a general description of this algorithm.

Keypoints in an image

Not all the pixels of image are suitable for feature matching. The shape of object may bear various deformations such as translation, rotation and scaling. Keypoints are chosen so as to be discriminative and as robust as possible to changes such as translations, rotations or blur.

If we define the original digital image I on a grid $\{\mathbf{x} \,|\, (x, y) \in \{1, \ldots, M\} \times \{1, \ldots, N\}\}$, a sequence of images of different blur levels are created by discrete convolutions of the input image with a series of variable-scale Gaussian kernels $G_\sigma(\mathbf{x}, \sigma)$:

$$L_\sigma(\mathbf{x}) = (I * G_\sigma)(\mathbf{x}).$$

These convolutions simulate the optical blur of camera. A Gaussian kernel of blur level σ is expressed as:

$$G_\sigma(\mathbf{x}) = \frac{1}{2\pi\sigma^2} exp(-\frac{\|\mathbf{x}\|^2}{2\sigma^2})$$

Previous work [MS04] shows that local extrema of scale-normalized Laplacian of Gaussian $\sigma^2\nabla^2 G_\sigma$ provide the most stable features despite of scale and position changes. However, such a differential operation is rather expensive in terms of computation. In practice, it is replaced by a lower cost approximation, i.e. the difference of Gaussian (DoG), followed by a precise localization of quadratic estimation. Here, DoG is the difference of two filtered images at nearby scales separated by a multiple factor k [Low04]:

$$D_\sigma(\mathbf{x}) = L_{k\sigma}(\mathbf{x}) - L_\sigma(\mathbf{x}). \tag{3.1}$$

The local maxima of DoG give a rough estimation of keypoints. Equation (3.1) is rewritten in form of second order Taylor expansion:

$$D_\sigma\left(\mathbf{x}+\varepsilon_{\mathbf{x}}\right)=D_\sigma\left(\mathbf{x}\right)+\mathbf{g}^T\varepsilon_{\mathbf{x}}+\frac{1}{2}\varepsilon_{\mathbf{x}}^T H\varepsilon_{\mathbf{x}}. \tag{3.2}$$

$\mathbf{g}=\left[\,(D_{\sigma\ x+1,y}-D_{\sigma\ x-1,y})/2\,,\,(D_{\sigma\ x,y+1}-D_{\sigma\ x,y-1})/2\,\right]^T$ is the gradient vector of DoG, and H is the 4×4 Hessian matrix of DoG:

$$H=\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix},$$

$$h_{11}=D_{\sigma\ x+1,y}+D_{\sigma\ x-1,y}-2D_{\sigma\ x,y}\,,\ h_{22}=D_{\sigma\ x,y+1}+D_{\sigma\ x,y-1}-2D_{\sigma\ x,y},$$
$$h_{12}=h_{21}=(D_{\sigma\ x+1,y+1}+D_{\sigma\ x-1,y-1}-D_{\sigma\ x+1,y-1}-D_{\sigma\ x-1,y+1}).$$

The exact position of maxima is corrected by an offset that sets derivative of Function (3.2) to zero. A more accurate candidate keypoint $\breve{\mathbf{x}}$ becomes:

$$\breve{\mathbf{x}}=\mathbf{x}+\varepsilon_{\mathbf{x}}=\mathbf{x}-H^{-1}\mathbf{g}.$$

After the process above, pixels belonging to complex local structures should be included in the set $\{\breve{\mathbf{x}}\}$. But some of them are characterized with certain unstable features that would be seriously affected by perturbations. So it still requires a further selection to filter candidate keypoints. Two cases are taken into account by SIFT: 1) points of low contrast to its surrounding which risk being enormously affected by noise; 2) points lying on edges which are hardly localized in space.

The first case is eliminated by setting up a threshold t_d for DoG value. Keypoints with their DoG value below this threshold are rejected. The second kind of points are picked out by analyzing their Hessian matrix of DoG. In the second case, edges always have large principal curvatures which can be estimated by ratio between the largest and smallest eigenvalues of Hessian matrix. So another threshold t_h is set up as the upper limit of this ratio. The keypoints finally remained are those that meet both conditions:

$$\left\{\breve{\mathbf{x}}\in\mathbb{R}^2\,\middle|\,\begin{array}{c} D_\sigma\left(\breve{\mathbf{x}}\right)>t_d \\ \dfrac{Tr(H)^2}{\|H\|}(\breve{\mathbf{x}})<t_h \end{array}\right\}.$$

Until now, the keypoint-search process is complete. It is designed based on the mathematical properties of local structure. However, it is not enough to describe

the invariant features of image, because distortion conditions have not been fully taken into account.

For the sake of scale invariance and speed, the original image is sub-sampled at several rates to simulate all possible zoom-outs of the image. The keypoint-search process is carried out once for each image. So, on the whole, a family of images at different sampling rates δ and different blur levels σ are created. This family is called the digital Gaussian scale-space. To classify these images, those that share a same sampling rate belong to a subfamily called octave. In each octave, images are of the same size, but of different blur level. To simplify the representation of each image layer, σ is defined as a variable across octaves the value of which depends on δ. So, an image layer of Gaussian scale-space is represented as $L_\sigma(\mathbf{x})$.

Besides, to ensure rotation invariance, each keypoint is attached to a reference orientation. It is estimated according to the gradient orientation distribution over a keypoint neighborhood Π_{grad}. Given a keypoint on image layer L_σ, all pixels in range $\{\mathbf{x} \,|\, (x,y) \in \Pi_{grad}\}$ get their gradient magnitude M_{grad} as well as their orientation θ_{grad} following:

$$\begin{aligned} M_{grad}(\mathbf{x}) &= \sqrt{(L_{\sigma\ x+1,y} + L_{\sigma\ x,y+1} - L_{\sigma\ x-1,y} - L_{\sigma\ x,y-1})^2}, \\ \theta_{grad}(\mathbf{x}) &= atan2(\,(L_{\sigma\ x,y+1} - L_{\sigma\ x,y-1}), (L_{\sigma\ x+1,y} - L_{\sigma\ x-1,y})\,) \mod 2\pi. \end{aligned} \tag{3.3}$$

These orientations are used to form a histogram whose maximum peak corresponds to the reference orientation of keypoint. To estimate θ_{grad} in a discrete way, the angle range $[0, 2\pi]$ is divided into n_{keybin} bins of which the kth bin is centered at $2k\pi/n_{keybin}$. Each θ_{grad} contributes to the histogram its gradient magnitude M_{grad} weighted on a Gaussian window. So, the reference orientation of a keypoint θ_{key} is its dominant direction of local gradients.

Finally, a keypoint comes out to be a triplet $(\mathbf{x}_{key}, \sigma_{key}, \theta_{key})$ consisting of its position, scale and reference orientation. It gives the center position of area to calculate invariant features.

Descriptors of keypoints

Descriptors record the distribution of gradient orientations in the neighborhood of keypoint. The basic idea is to form histograms in the same way as what we did to estimate the reference orientation of keypoints. However, since a descriptor takes into account gradient condition of not only the keypoint but also its surrounding samples, the process in this section is far more than a simple copy of previous one.

The first change lies in the definition of the keypoint neighbor area $\hat{\Pi}_{key}$. To ensure invariance of features, it is aligned by keypoint reference orientation. Following this adjustment, the coordinate axes are rotated to the direction of reference orientation and centered at keypoint position. If we suppose a keypoint to be $\mathbf{k} = (\mathbf{x}_{key}, \sigma_{key}, \theta_{key})$ with $\mathbf{x}_{key} = (x_{key}\,,\, y_{key})$, a pixel $\mathbf{x} = (x, y)$ of input image is transferred to its new coordinate $\hat{\mathbf{x}} = (\hat{x}, \hat{y})$ on image layer σ_{key} by:

$$\hat{x} = \frac{1}{\sigma_{key}}\,[\,(x - x_{key})\cos\theta_{key} + (y - y_{key})\sin\theta_{key}\,],$$
$$\hat{y} = \frac{1}{\sigma_{key}}[-(x - x_{key})\sin\theta_{key} + (y - y_{key})\cos\theta_{key}\,].$$

All pixels $\hat{\mathbf{x}}$ in range of $\hat{\Pi}_{key}$ are assigned to a gradient magnitude and an orientation. Their value are obtained according to Function (3.3) despite of change of coordinates.

More over, the neighborhood patch $\hat{\Pi}_{key}$ is no longer taken as a whole to form histogram. Instead, we choose an array of $n_{win} \times n_{win}$ points within $\hat{\Pi}_{key}$ and estimate histograms of the patches $\delta\hat{\Pi}_{key} \in \hat{\Pi}_{key}$ centered by them. The center point $(\hat{x}_i, \hat{y}_j)$ for $(i, j) \in \{1, \ldots, n_{win}\}^2$ is calculated by:

$$\hat{x}_i = \frac{2\lambda_{des}}{n_{win}}(i - \frac{1 + n_{win}}{2})\ ,\ \hat{y}_j = \frac{2\lambda_{des}}{n_{win}}(j - \frac{1 + n_{win}}{2}),$$

where λ_{des} is a Gaussian weight parameter [Ote15].

Finally, to accumulate contributions of gradient magnitude in $\delta\hat{\Pi}_{key}$, the angle range is divided into n_{desbin} bins. The accumulated results form a vector $\mathbf{d}_{ini}(\mathbf{k})$ of length $n_{win} \times n_{win} \times n_{desbin}$, which is the prototype of descriptor. It still needs to be normalized in order to reduce the impact of non-linear illumination changes. The final result $\mathbf{d}(\mathbf{k})$ is obtained by:

$$\mathbf{d}(\mathbf{k}) = \frac{\mathbf{d}_{ini}(\mathbf{k})}{\|\mathbf{d}_{ini}(\mathbf{k})\|}.$$

In original SIFT method, n_{win} and n_{desbin} are respectively set to 4 and 8. So the descriptor $\mathbf{d}(\mathbf{k})$ is by default a vector of 128 components.

3.1.2 Matching pixels between images

In this section, suppose that we have already detected a set of keypoints $\mathbf{K}_{ori}$ of image to be aligned (and $\mathbf{K}_{ref}$ of referential image respectively) and their descriptors. Matching process aims to establish correspondence between images

by searching keypoint pairs. It is composed by a general search for similar pixels and a rejection of false pairs.

At first, each keypoint in image to be aligned is assumed to have a matched keypoint in referential image. Similarity between two keypoints is estimated according to the Euclidean distance of their descriptors. The candidate keypoint is defined as the point that minimizes the descriptor distance:

$$\forall \hat{\mathbf{k}} \in \mathbf{K}_{ori}, \quad \mathbf{k} := \underset{\mathbf{k}_r \in \mathbf{K}_{ref}}{\arg\min} \|\mathbf{d}(\hat{\mathbf{k}}) - \mathbf{d}(\mathbf{k}_r)\|. \tag{3.4}$$

However, this definition is not reliable as some keypoint pairs do not share the same content. The selection of reliable matches counts on a further analysis of distance value between keypoint descriptors. If two keypoints share a same scene, their descriptor distance should be rather small. These keypoints can be distinguished among all possible pairs by a threshold. In order to avoid dependency on absolute distance, a relative threshold t_{match} is designed as a ratio between the minimum distance and the second minimum distance. Therefore, apart from the keypoint defined in Function (3.4), it requires to define a second nearest keypoint as:

$$\mathbf{k}' := \underset{\mathbf{k}_r \in \mathbf{K}_{ref} \setminus \{\hat{\mathbf{k}}\}}{\arg\min} \|\mathbf{d}(\hat{\mathbf{k}}) - \mathbf{d}(\mathbf{k}_r)\|.$$

As indicated in [Low99], a keypoint pair $(\hat{\mathbf{k}}, \mathbf{k})$ will be kept if it satisfies the inequality:

$$\frac{\|\mathbf{d}(\hat{\mathbf{k}}) - \mathbf{d}(\mathbf{k})\|}{\|\mathbf{d}(\hat{\mathbf{k}}) - \mathbf{d}(\mathbf{k}')\|} \leq t_{match}.$$

3.2 Homography

This section concerns the way to set up a coordinate mapping from images to be aligned (defined on grid $\Omega_{ori} = \{1, \ldots, M_{ori}\} \times \{1, \ldots, N_{ori}\}$) to referential image (defined on grid $\Omega_{ref} = \{1, \ldots, M_{ref}\} \times \{1, \ldots, N_{ref}\}$).

Images can be mapped onto each other by a homography if they are captured by a camera (a pinhole model) under either of the conditions [HZ03]: 1) The captured scene is a plane. 2) These images are acquired with the same camera center. Since our assumption of planar scenes conforms to the first case, we assume that the mapping between any image pair of the observed sequence can be represented as a homography.

We will explain the chosen model (i.e. homography) according to some basic knowledge of projective transformation in Section 3.2.1. Section 3.2.2 focuses on RANSAC algorithm which allows us to calculate homography with the most correct samples. In the end, we apply homography and bi-linear interpolation to align images in Section 3.2.3.

Remark that the keypoint pairs obtained in previous section will be used here as candidate samples. In the following text, we are particularly interested in pixel coordinates among three elements of each keypoint (see Section 3.1.1), which can be expressed as: $(\hat{\mathbf{x}}, \mathbf{x})$ with $\hat{\mathbf{x}} \in \Omega_{ori}$ and $\mathbf{x} \in \Omega_{ref}$.

3.2.1 A short review of projective geometry

The explanation should start with an extension of space. In order to represent the real world, Euclidean space is no longer convenient to be used as a geometry frame. The drawback lies in its lack of definition at infinity, which makes it impossible to explain why parallel lines such as railway tracks may intersect at horizon in image.

Therefore, a more general expression – projective space is introduced to computer vision problems. It is an extension of Euclidean space with an additional definition of infinity. Under projective assumption, there is no distinction between parallelism and intersection as parallel elements (lines in 2D space for example) meet at infinity. As a result, we are able to map every points of real world to pixels in image without worrying about non-defined cases.

Based on this space, we try to find out geometrical properties of pinhole camera model. It belongs in the category of central projection where the camera center corresponds to center of projection in Definition 3.1.

Definition 3.1. Central projection is a mapping that associates a set of points with another set of points by projection lines passing through a common point, namely the center of projection.

Since our problem context defines the scene to restore as a plane, we restrict therefore our condition as a specific case of central projection between 2D planes. This kind of projections preserves col-linearity when mapping points from one plane to another. The mapping can be expressed as a projective transformation according to Definition 3.2.

Definition 3.2. A mapping $h : \mathbb{P}^2 \to \mathbb{P}^2$ is a homography (or a projective transformation) if and only if the mapping results of three collinear points remain collinear. [HZ03].

Here, $\mathbb{P}^2$ represents a 2D projective space. To represent a pixel in this space, we add a non-zero ratio k (usually $k = 1$) to its Euclidean coordinates $\mathbf{x} = (x, y)^T$ and form homogeneous coordinates $\mathbf{x} = (kx, ky, k)^T$. The vector rises up to a triplet while the degree of freedom remains two for a point. Different from Euclidean coordinates, it is the proportion rather than the absolute value that makes sense and the degree of freedom is always one smaller than the number of elements. Back to the property of projective transformation, the expression of homogeneous coordinates makes it easier to understand dimension of matrix in Theorem 3.3 [HZ03].

Theorem 3.3. *A mapping $h : \mathbb{P}^2 \to \mathbb{P}^2$ is a homography if and only if $\forall \mathbf{x} \in \mathbb{P}^2$, $h(\mathbf{x}) = \mathbf{H}\mathbf{x}$. $\mathbf{H}$ is a non-singular 3×3 matrix.*

In brief, mapping from a planar scene to an image is a homography which can be mathematically expressed as a non-singular matrix. With help of this conclusion, we are able to analyze coordinate relationship between two images in our problem. Since projection from planar scene to each image is a homography, mapping between two of these images should be a composition of two homographies, which is still a homography. An example is illustrated in Figure 3.1.

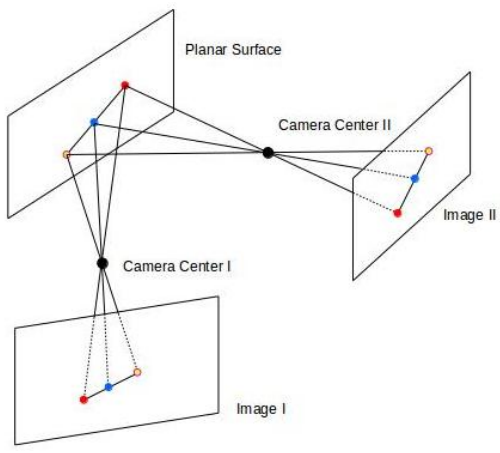


FIGURE 3.1: **Mapping between images of a planar scene.** Collinear points of planar surface remain collinear on image. A point $\mathbf{x}_0$ maps respectively to its position on image I and image II by $\mathbf{x}_1 = \mathbf{H}_1\mathbf{x}_0$ and $\mathbf{x}_2 = \mathbf{H}_2\mathbf{x}_0$. Since $\mathbf{H}_1$ and $\mathbf{H}_2$ are non-singular matrix, it is possible to set up a mapping between $\mathbf{x}_1$ and $\mathbf{x}_2$ as: $\mathbf{x}_2 = (\mathbf{H}_2\mathbf{H}_1^{-1})\mathbf{x}_1$, where $(\mathbf{H}_2\mathbf{H}_1^{-1})$ is a non-singular 3×3 matrices. So mapping between images of a planar scene is again a homography.

Recalling that our goal is to align photos taken from different camera positions for a planar scene, we plan to map the coordinates of each image onto that of an image selected as reference. The projection is carried out by calculating matrix of homography and then reconstructing pixels by interpolation (see Section 3.2.3). Remark that masks do not belong to the planar background and may be improperly reconstructed by homography in aligned image. But as they will be removed in final result, we do not emphasize here on their accurate alignment.

Coordinates of matched pixel pairs $\{(\hat{\mathbf{x}}, \mathbf{x})\}$ are used to estimate the matrix $\mathbf{H}$. For all $(\hat{\mathbf{x}}, \mathbf{x})$ belonging to planar background in image, we have theoretically $\mathbf{H}\hat{\mathbf{x}} = \mathbf{x}$. Under homogeneous coordinates, $\mathbf{H}$ is a 3×3 matrix with 8 degrees of freedom. It requires four pairs of matched pixels in general position (points in 2D space with 2 degrees of freedom) to calculate the matrix.

3.2.2 Sample selection and model decision

During the calculation, several conditions restrain us to select randomly four pairs of matched pixels for homography estimation. In the first place, the distribution of four pairs of selected matched pixels is decisive to the accuracy of homography estimation. Due to the existence of noise, a sample which covers a large area will give a more robust result than a sample concentrating on a small area. The homography of small area risks introducing errors later in the alignment of sequence. More particularly to our case, some of the pixel pairs belonging to the masks do not obey the homography of background. Their presence will lead to a wrong projective transformation.

Thus, we need a robust method to select samples so that the matrix fits best the data set $\{(\hat{\mathbf{x}}, \mathbf{x})\}$ despite of undesirable effects. We apply here a popular method named RANSAC (random sample consensus) [FB87]. It is an iterative algorithm that each time extracts randomly a minimum number of data to calculate model. It selects inliers by filtering data set members according to an acceptable error threshold. The model with largest number of inliers is finally selected as best choice. In our case, the data set indicates $\{(\hat{\mathbf{x}}, \mathbf{x})\}$ and we select every time four samples to calculate the model $\mathbf{H}$ using the DLT method ([HZ03], chapter 3). The error to estimate is $\|\mathbf{H}\hat{\mathbf{x}} - \mathbf{x}\|$.

Both problems of noise and outliers are resolved by this algorithm. The design of error threshold ε_{ransac} defines a tolerance of noise and the selection of candidate with largest number of inliers ensures the result to be a model applicable in an

Algorithm 1: Estimation of homography by RANSAC

Data: Matched pixel pairs $\mathbf{X}_0 = \{(\hat{\mathbf{x}}, \mathbf{x}) | \, \hat{\mathbf{x}} \in \Omega_{ori}, \mathbf{x} \in \Omega_{ref}\}$
Result: Homography $\mathbf{H}$, set of inlier matched pixels $\mathbf{X}_{in}$
Initialize $\mathbf{X}_{in} = \emptyset$, $\mathbf{H} = 0_{4\times4}$, $N_{inlier} = 0$;
for $i = 1$ **to** 10000 **do**
  Select randomly four pixel pairs among $\mathbf{X}_0$ to calculate $\mathbf{H}_{iter}$;
  foreach $(\hat{\mathbf{x}}, \mathbf{x}) \in \mathbf{X}_0$ **do**
    if $\|\mathbf{H}_{iter}\hat{\mathbf{x}} - \mathbf{x}\| \leq \varepsilon_g$ **then**
      $\mathbf{X}_{iter} \leftarrow (\hat{\mathbf{x}}, \mathbf{x})$;
    end
  end
  if $card(\mathbf{X}_{iter}) > N_{inlier}$ **then**
    Update $N_{inlier} = card(\mathbf{X}_{iter})$, $\mathbf{H} = \mathbf{H}_{iter}$, $\mathbf{X}_{in} = \mathbf{X}_{iter}$;
  end
end
return $\mathbf{H}$ and $\mathbf{X}_{in}$

area as big as possible. Since we assume that background takes up most area of image, the output $\mathbf{H}$ belongs certainly to the planar background.

Remark that a set of inlier pixel coordinates $\mathbf{X}_{in}$ is also derived from Algorithm 1. They are pixel positions of scene shared by both aligned image and referential image and will be used for photometric correction in Chapter 4.

3.2.3 Reconstruction and interpolation

Having obtained the homography $\mathbf{H}$ between two images, we create in the end an aligned image defined on pixel grid Ω_{ref} of referential image. The RGB values of pixels in aligned result is obtained by searching related pixel values in image to be aligned.

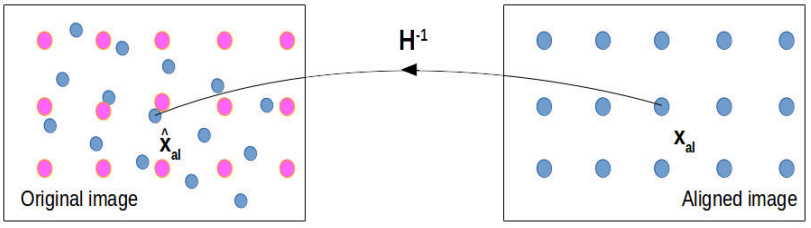


FIGURE 3.2: **Problem of inverse mapping.** the purple grid in original image and the blue grid in aligned image represent respectively Ω_{ori} and Ω_{ref} with definition of RGB vectors. The blue grid in the original image is the inverse mapping result of Ω_{ref}. Its elements rarely superpose on Ω_{ori}. As an example, the point $\mathbf{x}_{al} \in \Omega_{ref}$ is mapped to $\hat{\mathbf{x}}_{al}$ under $\mathbf{H}^{-1}$ with no definition of RGB vector in original image

We define here the RGB vector of image to be aligned as $I_{ori}(\mathbf{x}_{ori}) \in \{0, \ldots, 255\}^3$, $\mathbf{x}_{ori} \in \Omega_{ori}$ (resp. $I_{al}(\mathbf{x}_{al}) \in \{0, \ldots, 255\}^3$, $\mathbf{x}_{al} \in \Omega_{ref}$ for its aligned result).

In order to get pixel values of $\mathbf{x}_{al} \in \Omega_{ref}$, we need to know its related position in image to be aligned. Since $\mathbf{H}$ is a non-singular matrix, we set up an inverse mapping and find out: $\hat{\mathbf{x}}_{al} = \mathbf{H}^{-1}\mathbf{x}_{al}$ in image to be aligned. However, $\hat{\mathbf{x}}_{al}$ is very likely to be a position out of Ω_{ori} and there is no such a definition of RGB vector as: $I_{ori}(\hat{\mathbf{x}}_{al})$, $\hat{\mathbf{x}}_{al} \notin \Omega_{ori}$. A graphical description is shown in Figure 3.2.

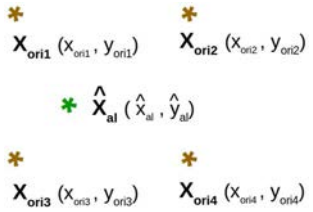


FIGURE 3.3: **Inverse mapping point and its surrounding.** $\hat{\mathbf{x}}_{al}$ is inverse mapping result within scope of Ω_{ori}. It is surrounded by four pixels defined on grid associated with RGB vectors $\mathbf{x}_{ori1}$, $\mathbf{x}_{ori2}$, $\mathbf{x}_{ori3}$ and $\mathbf{x}_{ori4}$.

This problem is solved by interpolating the RGB vector at $\hat{\mathbf{x}}_{al}$ according to its surrounding RGB values on the grid. In our application, we use a bi-linear interpolation to estimate the value of each vector $I_{ori}(\hat{\mathbf{x}}_{al})$. For the no-edge areas, we assume that an inverse mapping result $\hat{\mathbf{x}}_{al} = (\hat{x}_{al}, \hat{y}_{al})$, $1 \leq \hat{x}_{al} \leq M_{ori}$, $1 \leq \hat{y}_{al} \leq N_{ori}$ is within the scope of Ω_{ori}. It exists four pixels $\mathbf{x}_{ori\,i} = (x_{ori\,i}, y_{ori\,i})$, $i \in \{1, 2, 3, 4\}$ on grid Ω_{ori} in the neighborhood of $\hat{\mathbf{x}}_{al}$ as is shown in Figure 3.3 that relate to each other according to:

$$x_{ori1} + 1 = x_{ori2} = x_{ori3} + 1 = x_{ori4}\,;$$
$$y_{ori1} + 1 = y_{ori2} + 1 = y_{ori3} = y_{ori4}\,.$$

$I_{ori}(\hat{\mathbf{x}}_{al})$ is computed by two linear interpolations in both directions x and y with the value of these pixels. If we interpolate at the first step the value in direction of x, we get two results at $(\hat{x}_{al}, y_{ori1})$ and $(\hat{x}_{al}, y_{ori3})$ as:

$$I_{ori}(\hat{x}_{al}, y_{ori1}) = \frac{x_{ori2} - \hat{x}_{al}}{x_{ori2} - x_{ori1}} I_{ori}(\mathbf{x}_{ori1}) + \frac{\hat{x}_{al} - x_{ori1}}{x_{ori2} - x_{ori1}} I_{ori}(\mathbf{x}_{ori2});$$
$$I_{ori}(\hat{x}_{al}, y_{ori3}) = \frac{x_{ori4} - \hat{x}_{al}}{x_{ori4} - x_{ori3}} I_{ori}(\mathbf{x}_{ori3}) + \frac{\hat{x}_{al} - x_{ori3}}{x_{ori4} - x_{ori3}} I_{ori}(\mathbf{x}_{ori4}).$$

In a similar way, we get the final result by another interpolation in the direction y as is presented by Function (3.5) ([[ ]] means keeping integer part of result).

$$I_{ori}(\hat{\mathbf{x}}_{al}) = I_{ori}(\hat{x}_{al}, \hat{y}_{al}) = [[\frac{y_{ori3} - \hat{y}_{al}}{y_{ori3} - y_{ori1}} I_{ori}(\hat{x}_{al}, y_{ori1}) + \frac{\hat{y}_{al} - y_{ori1}}{y_{ori3} - y_{ori1}} I_{ori}(\hat{x}_{al}, y_{ori3})]]. \tag{3.5}$$

The obtained RGB vector is finally defined as RGB vector in aligned result:

$$I_{al}(\mathbf{x}_{al}) := I_{ori}(\mathbf{H}^{-1}\mathbf{x}_{al}) = I_{ori}(\hat{\mathbf{x}}_{al}).$$

3.3 Experiments

Our experiments are carried out by repeating the alignment process between a reference image and every other images in the sequence. The reference image in Algorithm 2 refers to the first image in the sequence. In examples we show at the end of this chapter, the reference images are pre-cut and only their areas of interest are put into use. This pre-processing is not essential while it ensures the concentration of matched pixels on the plane, which is an important factor for the proper operation of RANSAC.

Algorithm 2: Geometrical alignment

Data: A set of n images $\mathbf{\Phi}_{ori} = \{I_{ori\,i}(\Omega_i)\}, i \in \{1 \dots n\}$
Result: A set of geometrical aligned result $\mathbf{\Phi}_{al} = \{I_{al\,i}(\Omega_1)\}, i \in \{1 \dots n\}$
Initialize $\mathbf{\Phi}_{al} = \emptyset$;
$\mathbf{\Phi}_{al} \leftarrow I_{ori\,1}(\Omega_1)$;
foreach $I_{ori\,i}(\Omega_i) \in \mathbf{\Phi}_{ori} \backslash I_{ori\,1}(\Omega_1)$ **do**
  Compute matched pixels $\{(\hat{\mathbf{x}}, \mathbf{x}) | \hat{\mathbf{x}} \in \Omega_i, \mathbf{x} \in \Omega_1\}$ through SIFT estimation on $I_{ori\,i}(\Omega_i)$ and $I_{ori\,1}(\Omega_1)$;
  Calculate homography H mapping $\Omega_{ori\,i}$ to $\Omega_{ori\,1}$ (see algorithm 1);
  Re-sample $I_{ori\,i}$ on Ω_1 by bi-linear interpolation of each channel as $I_{al\,i}(\Omega_1)$;
  $\mathbf{\Phi}_{al} \leftarrow I_{al\,i}(\Omega_1)$;
end
return $\mathbf{\Phi}_{al}$

Since the algorithm is very classical and has been used in many previous works, its correctness is no need to be proved. However, there are some details to notice in actual use.

First, the assumption on homography is easy to break down due to the optical aberration introduced by the camera lenses. To reduce the potential effects of the aberration, we take either of the following measures during our experiments:

a) We use a prime lenses of good quality. The aim is to reduce geometrical aberration which results in the lose of rigidity of objects in images. It appears either as pincushion or as barrel shape which depends strongly on the zoom of lens. A poor quality of lenses will aggravate this phenomenon. We use a prime lens (SIGMA 30mm f1.5 DC HSM) to take most of our photos.

b) We arrange our target planes in the center of image. The distortion of view has the tendency to be more and more serious from the center to the side of a lens. We put therefore our interest objects in the center.

c) We take pictures with a modest change of camera positions. That means the reference image has less difference of visual angle with other images. In this way, the projection errors will be small even under the existence of distortion.

The adjustment of parameters is another key point to the success of this process. The iteration number of RANSAC should be decided by the matched pixel numbers and the inlier probability which are difficult to predict. The best combination of samples may not be included and the model may be inaccurate without enough iteration. To simplify the problem, we set it at a large number (10000). It makes the process slow while provides enough sampling in every cases. Besides, the relative threshold between nearest and second nearest neighbors in SIFT matching is set to 15 and the threshold of residual errors is set at 1 pixel. These thresholds are rather strict while in actual tests we get usually about 500 pairs of matched pixels for an image of size 1000×2000, which are enough for our estimation.

Based on the discussion above, we tested our program with several image sequences and we give here three examples in Figures 3.4, 3.6, 3.8 along with their results: Figures 3.5, 3.7, 3.9. The first images in original sequences are set as the reference and a direct cut-off of these images are presented as the first images in aligned results. Other images are resampled according to their vision and range.

The results turn out to be satisfying. Most of the images superpose perfectly onto each other with errors below one or two pixels which are acceptable for the correct operation of further steps. We will see more challenging sequences (e.g.acquired by smart phone) in Part III.

Of course, these observations are based on the ideal condition supported by the perfect equipments. We will see errors occur with an increasing trend from the center to the border of image when using lenses of poor quality or with zoom. A further discussion will be later found in Chapter 7.

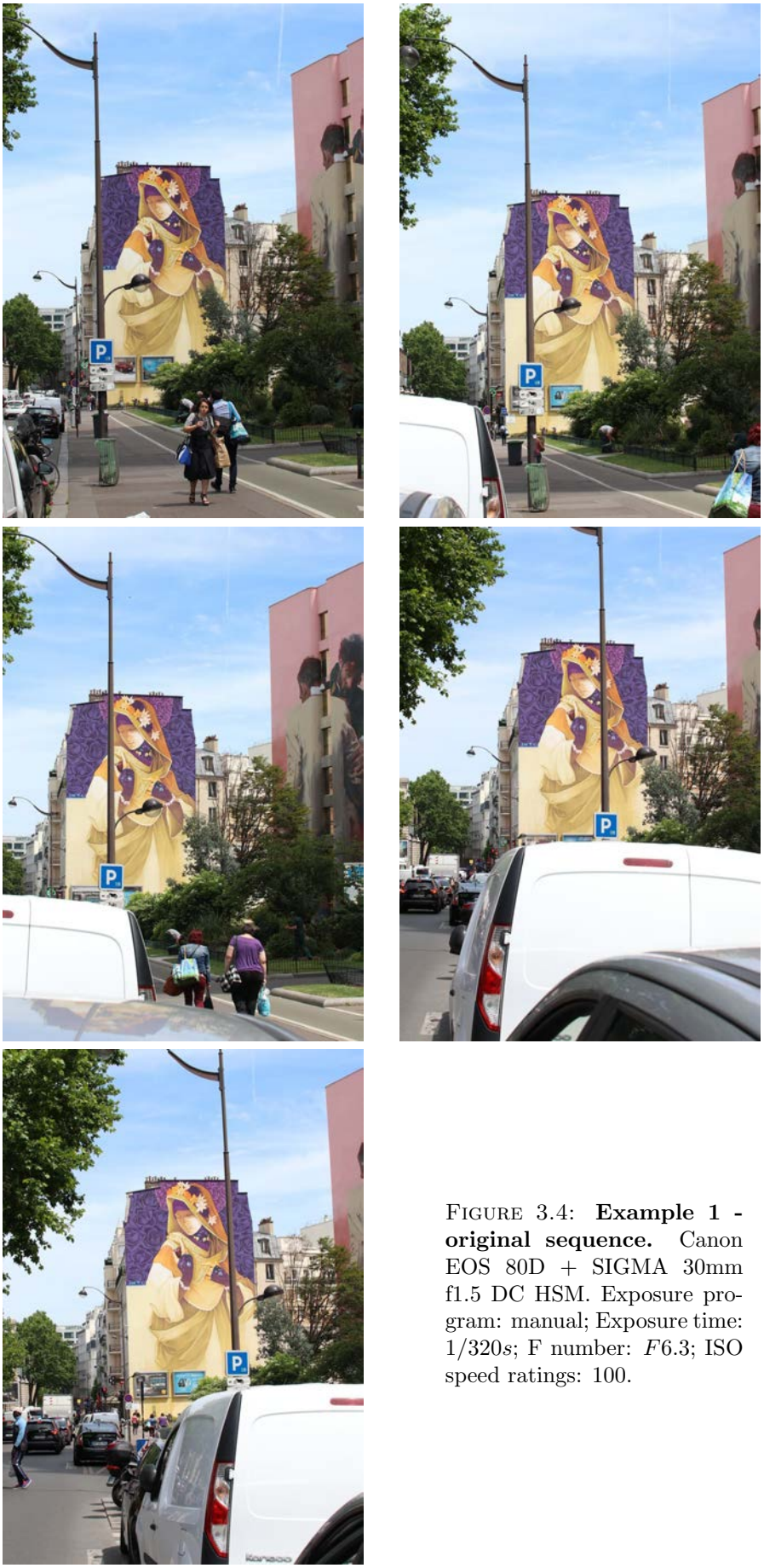


FIGURE 3.4: **Example 1 - original sequence.** Canon EOS 80D + SIGMA 30mm f1.5 DC HSM. Exposure program: manual; Exposure time: $1/320s$; F number: $F6.3$; ISO speed ratings: 100.

FIGURE 3.5: **Example 1 - aligned results.** The maximum errors appear on the borders of images. The images superpose perfectly onto each other in the center areas.

FIGURE 3.6: **Example 2 - original sequence.** Canon EOS 80D + SIGMA 30mm f1.5 DC HSM. Exposure program: manual; Exposure time: 1/100s; F number: F6.3; ISO speed ratings: 160.

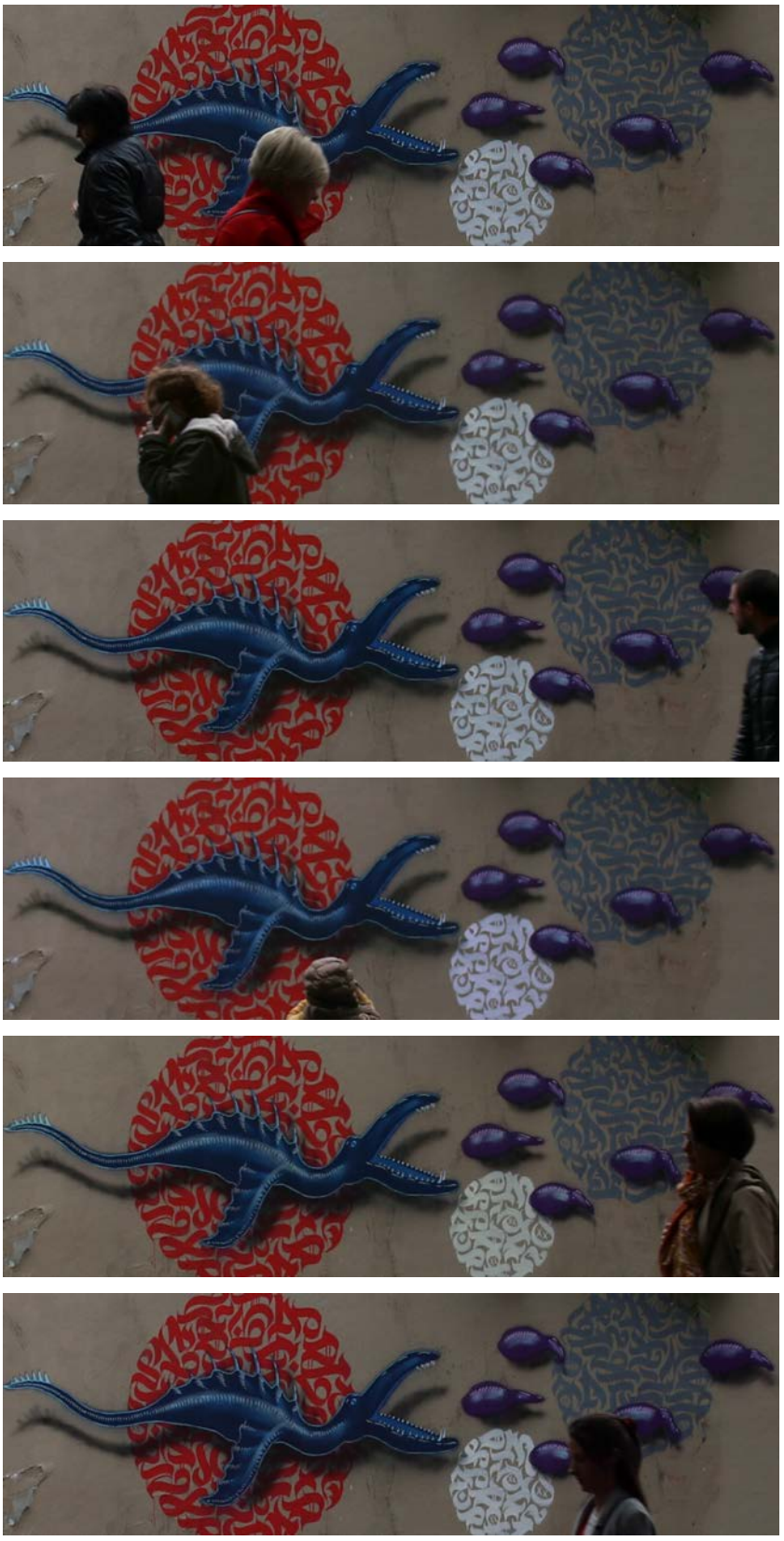


FIGURE 3.7: **Example 2 - aligned results.** The aligned results show no obvious errors in the center. The errors appear generally on the borders of images. The maximum errors are under 2 pixels.

FIGURE 3.8: **Example 3 - original sequence.** Canon EOS 80D + SIGMA 30mm f1.5 DC HSM. Exposure program: manual; Exposure time: $1/100s$; F number: $F10$; ISO speed ratings: 100.

FIGURE 3.9: **Example 3 - aligned results.** The maximum aligned errors are about 1 pixel. The distribution of maximum errors does not concentrate on certain areas.

Chapter 4

Photometric Adjustment

Having been registered, every images in sequence convey the similar background pixels at the same positions. Yet due to the existence of color mismatches, these images can not be directly put into use for fusion procedure. The photometric difference between images is a common phenomenon which may be caused by many factors. The most possible reasons are listed as follows:

a) A variation of white balance adjustment caused by lighting condition changes (e.g.clouds, shadow of moving objects).
b) Changeable reflected lights from non-Lambertian objects under different observing angles.
c) Effects of spatial chromatic aberration and vignetting on lenses.
d) Automatically changed camera parameters (aperture size, time of exposure, ISO sensitivity) according to scene.
e) Unknown effects of camera processing (e.g.noise filtering, white balance, content enhancement) on different scenes.

In reverse, the effects of chromatic differences can be reduced by:

- capturing images in a short delay under stable illuminating conditions;
- avoiding extreme viewing angles;
- applying professional cameras and optical lenses of good quality;
- setting manually as many as possible the parameters (mode M)
- choosing file format with less automatic processing in camera (e.g.RAW)

Even so, it is not possible to eliminate totally the problem and obtain photometrically identical images. Without any adjustment, fusion results of such sequences risk being mottled, or even worse, bearing wrongly selected masks. We attempt thus to design a method to unify object colors in different images.

This chapter is dedicated to our discussion on color adjustment methods. It is arranged as follows: Section 4.2 introduces the pipeline of color formation process in camera. Section 4.3 includes our propositions of color alignment method. Finally, Section 4.4 provides experimental results of photometric correction along with our analysis on the performance of different approaches.

4.1 Experimental assumptions

In order to set up a stable model of luminance, we try to restrict the unpredictable effects of lighting conditions. At the stage of method conception, our experiences are carried out under three assumptions:

1) All surfaces obey Lambert hypothesis without great specular highlights. Luminance value remains the same regardless of observation directions.
2) Each group of images is taken in a short time to prevent changes of lighting conditions. Impacts of any unexpected factors are negligible.
3) Camera parameters such as aperture size, ISO value and shutter speed are fixed when taking an image sequence.

These requirements are easy to meet when taking photos of a matte surface with parameters fixed in a very short time. We suppose from then on that the luminance issued from the natural space remains approximately the same when it is received by camera at any positions.

Mathematically, we define the natural space as $\Pi_s \subset \mathbb{R}^3$. We denote the luminance values, issued from $\mathbf{x}_s \in \Pi_s$, received by camera at two different positions by $L_1(\mathbf{x}_s)$ and $L_2(\mathbf{x}_s)$. Their relationship may be expressed as:

$$L_1(\mathbf{x}_s) = L_2(\mathbf{x}_s) + \varepsilon_s(\mathbf{x}_s). \tag{4.1}$$

where ε_s is a small value. The luminance will be further quantified and finally mapped onto a 2D image grid after a complicated transformation through the optical and electronic systems in digital camera.

4.2 Color formation process in digital camera

Every manufacturers have their own image processing algorithms, but they tend to share a rather fixed pipeline for color adjustment. Neglecting some procedures

for noise and artifacts removal, we present the major steps along with their output formats in Figure 4.1.

In this section, we shall introduce each step of the pipeline and make clear of its role in color formation process. Generally speaking, most procedures, other than an optional sRGB transformation, are approximately linear. Therefore, as later we shall see, the conversion from scene luminance to digital RGB image can be approximately modeled by matrix.

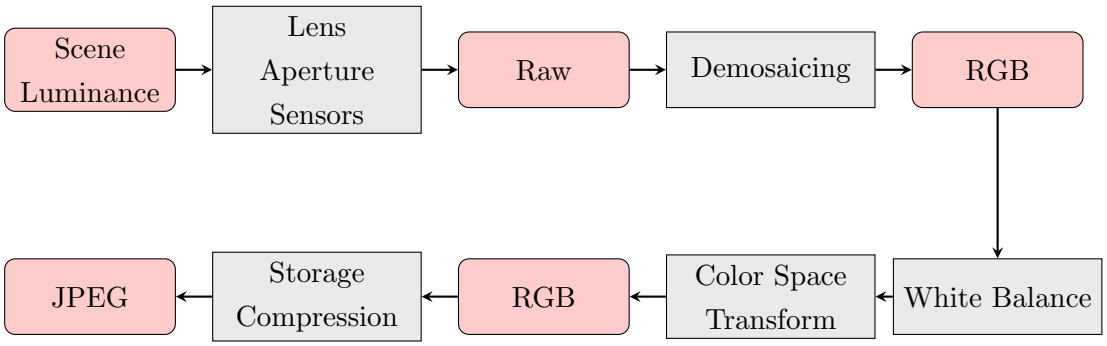


FIGURE 4.1: **Pipeline of camera processing.** Gray blocks represent main steps during camera processing while pink blocks indicate data formats after corresponding steps. There are also some manufacturers who put white balance in front of demosaicing, but it makes no big difference later in image color model because both steps are approximately linear.

4.2.1 From luminance to raw

Information of scene is firstly recorded by raw file. It is the most elementary image format that contains almost unprocessed data from the camera sensor ([M+17] Chapter 8). To state briefly this step, the scene luminance passes through a color-filter system then hits on a sensor array. Sensors convert photon energy into intensity values after a series of analog and digital conversions such as sampling and quantization. The result is finally recorded by raw files (usually on 8 to 14 bits), whose data preserves as closely as possible information of natural scene. Most adjustments are carried out based on this original image format.

If we denote intensity value (red, green or blue) at position $\mathbf{x}$ by $U(\mathbf{x}) \in \mathbf{R}$, the response function that projects luminance of 3D scene space to raw data on a 2D image grid Ω is approximately linear unless the signal is too weak or the sensor is saturated (see e.g.[Ham13] for CCD sensors):

$$\forall \mathbf{x} \in \Omega,\ \exists \mathbf{x}_s \in \Pi_s, \quad U(\mathbf{x}) = GL(\mathbf{x}_s) + B + n(\mathbf{x}_s).$$

Here, G is the ISO sensitivity function of camera that represents global gain of the whole acquisition process. The black level B is a value purposely added in the result by camera manufacturers in order to keep it positive. The term n is additive noise with respect to Gaussian distribution [ADGM14].

4.2.2 From raw to RGB

The sensor we mentioned above consists of a color filter array (CFA) on top of CCD or CMOS elements. Taking Bayer filter as an example, it is the most commonly used CFA in camera (see Figure 4.2). A periodic pattern of red, green and blue filters separates light elements according to their wavelength range. The raw file, which records the color information received by sensors, presents therefore the image by mosaic of red, green and blue intensities.

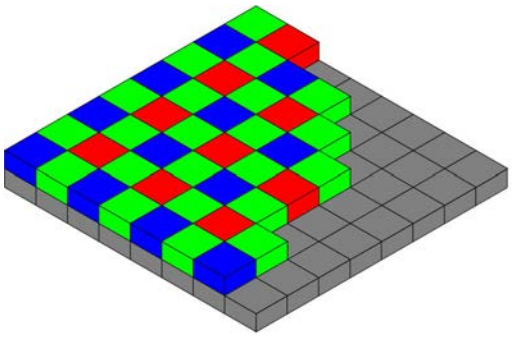


FIGURE 4.2: **Example of Bayer filter array.** [Wik06] It is the three-color filter layer covering a sensor array (gray grid). Incoming light is decomposed into blue, red and green elements. The filter leaves element of its own color passing through its layer and arriving at sensor array. These sensors record then intensity of light and output an image called Bayer pattern image in Raw file. It requires a further demosaicing process to become a full-color image.

The part of work to compute a full-color image from the mosaic of three primary colors is known as demosaicing. Many algorithms [ZW05, CC06, BCMS09, Get12] have been proposed to accomplish this conversion, and most of them share the idea of interpolation. Hence, this process is often approximately regarded as a linear conversion.

Following Function (4.1), we further assume the approximate equality of RGB vectors between images after demosaicing. We define respectively two digital images on grid Ω_1 and Ω_2. If a point $\mathbf{x}_s \in \Pi_s$ is mapped on these images at $\mathbf{x} \in \Omega_1$ and $\hat{\mathbf{x}} \in \Omega_2$, the RGB vectors after demosaicing of these two pixels are related by (ε_{rgb} is a small value):

$$\mathbf{I}_{mosaic1}(\mathbf{x}) = \mathbf{I}_{mosaic2}(\hat{\mathbf{x}}) + \varepsilon_{rgb}(\hat{\mathbf{x}}). \tag{4.2}$$

4.2.3 Color adjustments in RGB space

Three color components obtained from the sensor are strongly dependent on the physical properties of both the CFA and the sensors. Yet in actual use, the signal is required to be compatible with various devices such as screen and printer in absence of the camera sensor information. It is thus necessary to convert the RGB vectors obtained from previous steps into a conventional color space. Moreover, to get rid of the specific illuminating conditions of scene (as the human visual system does), it may be appreciated to apply a adjustment named white balance. Both operations, which will be described in this section, may introduce photometric differences to the images. Some other occasional effects, such as non-Lambertian reflexion and coloring lights of moving objects, will not be discussed here.

White balance

White balance, or more generally color balance, is used to correct color intensities of image in order to reduce the influence of ambient color. This adjustment is originally raised based on the fact that camera sensors do not have color constancy ability as human vision system. Human is able to perceive color of objects regardless of ambient lights, while as a device without subjective perception, camera records objectively a combined result of all illuminants. The recorded result is physically correct but subjectively deviates from perceived colors of objects.

White balance serves as a corrector of this perceived unreality. The basic idea is to set up a map between a chosen pixel vector in image (neutral color) and a defined intensity vector. Other colors in image are then calibrated by this map in order to return back to their wanted appearance. In RGB channel space, this procedure is interpreted as a rotation of color vector in color space [Pro17]). It is expressed as a 3×3 diagonal matrix whose parameters are calculated independently according to intensity distribution of each channel. We continue representing original RGB vector at $\mathbf{x} \in \Omega$ as $\mathbf{I}_{mosaic}(\mathbf{x})$, and we define $\mathbf{I}_{balance}(\mathbf{x})$ as the vector after white balance. The adjustment is, at most of the time, written as a change of scale on each axis [VK05]:

$$\mathbf{I}_{balance}(\mathbf{x}) = \mathbf{W}\,\mathbf{I}_{mosaic}(\mathbf{x}), \quad \mathbf{W} = \begin{bmatrix} w_r & 0 & 0 \\ 0 & w_g & 0 \\ 0 & 0 & w_b \end{bmatrix} \; w_r, w_g, w_b \neq 0. \tag{4.3}$$

There are many possible choices for diagonal factors. To give some examples, they would be inverse of neutral color value [WCF05], deviation of intensity average from gray [Lam05], or inverse of maximal channel value. All these methods indicate that diagonal factors are decided according to lighting conditions of the whole image. As a result, white balance matrix of two images are different as long as their content is not exactly the same. That explains where comes the color difference of same content in different images.

Color space transformation

For the sake of image interoperability on diverse devices, it is necessary to map pixel values onto a suitable color space (such as HSI, CIE, YCbCr and CMYK) according to certain requirements of using condition. In our case, vector transformation on CIE space helps improving visual experience, a further optional transformation on sRGB space provides a perfect color expression for 8-bit storage and a good compatibility with display systems.

1) CIE-xyz color space which is derived from physical experiments is regarded as a good estimate of physiologically perceived color space [WS82, Hun05]. Due to the different spectral sensitivities between camera sensor and human vision system, data captured by camera sensors is not adapted to describe exactly what people see. In this case, we interpret the captured color in CIE-xyz color space in order to simulate human color sensation. This transformation is approximately described as a linear projection from color balance result $\mathbf{I}_{balance}(\mathbf{x})$ to $\mathbf{I}_{cie}(\mathbf{x})$ [FBH97]:

$$\mathbf{I}_{cie}(\mathbf{x}) = \mathbf{C}\,\mathbf{I}_{balance}(\mathbf{x}), \quad \mathbf{C} = \begin{bmatrix} 2.7688 & 1.7517 & 1.1301 \\ 1.0000 & 4.5906 & 0.0601 \\ 0.0000 & 0.0565 & 5.5942 \end{bmatrix}. \tag{4.4}$$

2) sRGB is a type of RGB color space well adapted to physical display devices. It maps image data from 16-bit to 8-bit on a limited gamut. Such a transformation will certainly lead to a loss in dynamic range. Even so, it is still a popular optional step in camera processing based on the consideration of 8-bit format requirement of most output devices. What is more, the transfer function of sRGB takes into account a gamma correction, from which the image benefits a better adaptation to display character of some devices such as CRT systems.

A transformation from CIE-xyz space to sRGB color space consists of two steps. The first step is a linear projection carried out by a matrix multiplication. We define the vector obtained after linear step as $\mathbf{I}_{linear}(\mathbf{x})$. The linear transformation step can be described by Function (4.5) as:

$$\mathbf{I}_{linear}(\mathbf{x}) = \mathbf{S}\,\mathbf{I}_{cie}(\mathbf{x}), \quad \mathbf{S} = \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix}. \tag{4.5}$$

The second step is gamma correction in form of power-law equation [SACM12]. It is now widely used for sake of a better visual quality and storage efficiency (The original log compression was designed for the sake of compatibility with CRT displays and tube cameras). sRGB defines its gamma correction as a piecewise function performing identically at each channel of color vector. To simplify our expression, we represent uniformly channel element $i \in \{red, green, blue\}$ of $\mathbf{I}_{linear}(\mathbf{x})$ as $I_{linear\,i}(\mathbf{x})$. It relates to channel element $I_i(\mathbf{x})$ of output vector $\mathbf{I}(\mathbf{x})$. The function is expressed as:

$$I_i(\mathbf{x}) = f_\gamma(I_{linear}(\mathbf{x})) = \begin{cases} 12.92\, I_{linear\,i}(\mathbf{x}), & I_{linear\,i}(\mathbf{x}) \leq b \\ (1+a)\, I_{linear\,i}(\mathbf{x})^{1/2.4} - a, & I_{linear\,i}(\mathbf{x}) > b \end{cases} \tag{4.6}$$

with b usually set at 0.0031308 and $a = \frac{12.92b - b^{1/2.4}}{b^{1/2.4} - 1} = 0.055$.

By the end of this step, the image processing inside camera is almost complete. There is still a step of compression in pipeline (Figure 4.1). It is proceeded at cost of losing patch details, which will interfere our analysis on pixels in further steps. Fortunately, the compression options, along with file types, become selectable in many cameras. Therefore, we choose in our experiments the image files with less compression and $\{\mathbf{I}(\mathbf{x}) | \mathbf{x} \in \Omega\}$ is regarded approximately as the final output of camera in our analysis.

4.2.4 A short resume

To sum up, color variation is an associate result of multiple causes. It can be reduced by fixing parameters such as aperture size, shutter speed and ISO value. But the effects of internal adjustment in camera can not be totally removed by manual control.

Some internal algorithms such as white balance are involved in image lighting conditions of the whole scene captured by image. As we change camera positions

to get information of areas blocked by masks, the variation of lighting conditions, as well as distortion of color in images is inevitable.

Fortunately, the major procedures of color adjustment in camera are clear and rather fixed. They can be expressed as a series of linear transformations followed by a gamma correction. This knowledge gives us the possibility to set up a color mapping between two images, and we will discuss our mapping propositions in the next section.

4.3 Color alignment methods

This section is devoted to the discussion of color alignment propositions and sampling methods. Since internal parameters of camera may vary with scene, it is impossible to establish a transfer function shared by all images. Similar to our geometric alignment method in Chapter 3, our idea is to define a referential image and adjust each time one image in sequence according to a color transfer function. If it works well, we will get a new sequence of images sharing photometric information of referential image.

Our work is organized as follows: Section 4.3.1 summarizes mathematical model of color formation process in camera and then defines our adjustment problem. Next, Section 4.3.2 is dedicated to propose possible methods for photometric correction. At the end, Section 4.3.3 focuses on the choice of samples for transfer function estimation.

4.3.1 Problem model

Given two digital images after geometric alignment defined on grid Ω_{tr} (image to be adjusted) and Ω_{ref} (referential image), $\mathbf{X}_0 = \{(\hat{\mathbf{x}}, \mathbf{x}) | \hat{\mathbf{x}} \in \Omega_{tr}, \mathbf{x} \in \Omega_{ref}\}$ is a set of pixel pairs sharing the same content of natural scene. $\mathbf{I}_{tr}(\hat{\mathbf{x}})$ and $\mathbf{I}_{ref}(\mathbf{x})$ defined on $[0,1]^3$ refer to normalized output vectors after sRGB transformation. Our objective is to define a color transfer function:

$$\forall (\hat{\mathbf{x}}, \mathbf{x}) \in \mathbf{X}_0, \quad g : \mathbf{I}_{tr}(\hat{\mathbf{x}}) \mapsto \mathbf{I}_{ref}(\mathbf{x}).$$

We focus first of all on color vector transformation of Section 4.2.3. We represent all procedures changing a demosaicing RGB vector to its output vector after sRGB adjustment with a single symbol f. Among all color adjustment steps in RGB space, the only procedure introducing a nonlinear operation is gamma

correction f_γ. Thus, we set it aside and define the linear part as $f_l := \mathbf{SCW}$ (with $\mathbf{S}$ defined in Function (4.5), $\mathbf{C}$ defined in Function (4.4), $\mathbf{W}$ defined in Function (4.3)). Then f is expressed as a composition of two parts:

$$f = f_\gamma \circ f_l.$$

Here, f_l is a non-singular matrix. The coefficients of some linear transform methods (e.g. white balance) vary with the content of images. So f_l and f vary with images (noted as $f_{ref.l}$ and f_{ref} for reference image, $f_{tr.l}$ and f_{tr} for image to be adjusted). f_γ (see Function (4.6)) has its inverse function expressed as:

$$f_\gamma{}^{-1}(I_i(\mathbf{x})) = \begin{cases} I_i(\mathbf{x})/12.92, & I_i(\mathbf{x}) \leq 12.92b \\ [(I_i(\mathbf{x}) + a)/(1+a)]^{2.4} & I_i(\mathbf{x}) > 12.92b \end{cases} \tag{4.7}$$

with $b = 0.0031308$, $a = \frac{12.92b - b^{1/2.4}}{b^{1/2.4} - 1} = 0.055$.
Thus, f is an invertible function and the original RGB vectors of $\mathbf{I}_{tr}(\hat{\mathbf{x}})$ and $\mathbf{I}_{ref}(\mathbf{x})$ are represented as: $f_{tr}{}^{-1}(\mathbf{I}_{tr}(\hat{\mathbf{x}})) = f_{tr.l}{}^{-1} \circ f_\gamma{}^{-1}(\mathbf{I}_{tr}(\hat{\mathbf{x}}))$ and $f_{ref}{}^{-1}(\mathbf{I}_{ref}(\mathbf{x})) = f_{ref.l}{}^{-1} \circ f_\gamma{}^{-1}(\mathbf{I}_{ref}(\mathbf{x}))$. Since two images convey the same content at $\mathbf{x}$ and $\hat{\mathbf{x}}$, it is possible to set up a mapping between original RGB vectors of the two images. From Function (4.2), we assume that:

$$\begin{aligned} f_{ref}{}^{-1}(\mathbf{I}_{ref}(\mathbf{x})) &= f_{tr}{}^{-1}(\mathbf{I}_{tr}(\hat{\mathbf{x}})) + \varepsilon_{rgb}(\hat{\mathbf{x}}) \\ f_{ref.l}{}^{-1} \circ f_\gamma{}^{-1}(\mathbf{I}_{ref}(\mathbf{x})) &= f_{tr.l}{}^{-1} \circ f_\gamma{}^{-1}(\mathbf{I}_{tr}(\hat{\mathbf{x}})) + \varepsilon_{rgb}(\hat{\mathbf{x}}). \end{aligned} \tag{4.8}$$

In the end, we get the expression of g as:

$$g(\mathbf{I}_{tr}(\hat{\mathbf{x}})) = \mathbf{I}_{ref}(\mathbf{x}) = f_{ref} \circ (f_{tr}{}^{-1}(\mathbf{I}_{tr}(\hat{\mathbf{x}})) + \varepsilon_{rgb}(\hat{\mathbf{x}})). \tag{4.9}$$

This function can not be further simplified as f is nonlinear. Our problem will be solved if we can describe Function (4.9) or its equivalent form, Function (4.8).

4.3.2 Propositions of method

As we have discussed, the key point to this problem is to decide a mathematical formula for color transfer model. It is necessary to be noted that the pipeline in Section 4.2 is rather general without taking into account any unpredictable factors such as vignetting and aberration. What's more, there may be still some algorithms added optionally by camera manufacturers. As a result, if the associated effects of these factors are important enough, a delicate explication of theoretical model may introduce errors to the result as well.

Thus, although we have given parameters for $\mathbf{S}$, $\mathbf{C}$ and $\mathbf{W}$ in Section 4.2.3, it is computational costly and unnecessary to calculate f with these parameters. On the contrary, if we can find a easier way to estimate a model with the help of samples in images, the result may be more appropriate to the real case.

We have principally two ideas based on different expressions of model. The first method is derived from Function (4.8). We attempt to set up a linear relationship by removing nonlinear part of function. Our second design is an estimation of g. We abandon details of Function (4.9) and approximate g with polynomials. The degree of polynomial will be limited to 2.

sRGB reduction

Our idea is first to eliminate the gamma correction in sRGB conversion, and then to apply a linear transformation between two images. It means that we regard $\tilde{\mathbf{I}}_{tr}(\hat{\mathbf{x}}) = {f_\gamma}^{-1}(\mathbf{I}_{tr}(\hat{\mathbf{x}}))$ and $\tilde{\mathbf{I}}_{ref}(\mathbf{x}) = {f_\gamma}^{-1}(\mathbf{I}_{ref}(\mathbf{x}))$ as our new estimators. Function (4.8) can be rewritten as:

$$\tilde{\mathbf{I}}_{ref}(\mathbf{x}) = f_{ref.l} \circ ({f_{tr.l}}^{-1}(\tilde{\mathbf{I}}_{tr}(\hat{\mathbf{x}})) + \varepsilon_{rgb}).$$

In consideration of unknown factors involved in color formation process of camera, we will not simplify this expression any further. Yet it is convincing enough that the transfer relation between $\tilde{\mathbf{I}}_{tr}(\hat{\mathbf{x}})$ and $\tilde{\mathbf{I}}_{ref}(\mathbf{x})$ can be described by an affine g_s. The transfer process is composed of five steps:

1) Mapping $\mathbf{I}_{tr}(\hat{\mathbf{x}})$ to $\tilde{\mathbf{I}}_{tr}(\hat{\mathbf{x}})$ (resp. $\mathbf{I}_{ref}(\mathbf{x})$ to $\tilde{\mathbf{I}}_{ref}(\mathbf{x})$) by Function (4.7);
2) Estimating g_s by selected pixel pairs $\{(\tilde{\mathbf{I}}_{tr}(\hat{\mathbf{x}}), \tilde{\mathbf{I}}_{ref}(\mathbf{x}))\}$;
3) Removing effect of gamma correction on $\mathbf{I}_{tr}$;
4) Adjusting colors with g_s;
5) Applying gamma correction by Function (4.6) to create a new image.

We express g_s as a 3×4 affinity matrix. We extend vectors as triplets: $\tilde{\mathbf{I}}_{tr}(\hat{\mathbf{x}}) = [\tilde{I}_{tr\,1}(\hat{\mathbf{x}}), \tilde{I}_{tr\,2}(\hat{\mathbf{x}}), \tilde{I}_{tr\,3}(\hat{\mathbf{x}})]^T$ and $\tilde{\mathbf{I}}_{ref}(\mathbf{x}) = [\tilde{I}_{ref\,1}(\mathbf{x}), \tilde{I}_{ref\,2}(\mathbf{x}), \tilde{I}_{ref\,3}(\mathbf{x})]^T$. An affine mapping is of the form:

$$\begin{bmatrix} \tilde{I}_{ref\,1}(\mathbf{x}) \\ \tilde{I}_{ref\,2}(\mathbf{x}) \\ \tilde{I}_{ref\,3}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} \tilde{I}_{tr\,1}(\hat{\mathbf{x}}) \\ \tilde{I}_{tr\,2}(\hat{\mathbf{x}}) \\ \tilde{I}_{tr\,3}(\hat{\mathbf{x}}) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix},$$

where a_{ij} and b_i with $i, j \in \{1, 2, 3\}$ are coefficients to estimate. We modify the form of input vectors $\tilde{\mathbf{I}}_{tr}(\hat{\mathbf{x}})$ by adding a constant 1 as its fourth element. Then its equivalent expression can be written as the multiplication between a vector

and a matrix:

$$\begin{bmatrix} \tilde{I}_{ref\,1}(\mathbf{x}) \\ \tilde{I}_{ref\,2}(\mathbf{x}) \\ \tilde{I}_{ref\,3}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{bmatrix} \begin{bmatrix} \tilde{I}_{tr\,1}(\hat{\mathbf{x}}) \\ \tilde{I}_{tr\,2}(\hat{\mathbf{x}}) \\ \tilde{I}_{tr\,3}(\hat{\mathbf{x}}) \\ 1 \end{bmatrix}. \tag{4.10}$$

where g_s is represented as:

$$g_s := \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{bmatrix}.$$

Its 12 coefficients will be decided by 4 pairs of vectors $(\tilde{\mathbf{I}}_{tr}(\hat{\mathbf{x}}_r), \tilde{\mathbf{I}}_{ref}(\mathbf{x}_r))$, $r \in \{1, 2, 3, 4\}$. We reorganize the function as:

$$\begin{bmatrix} \tilde{I}_{ref\,1}(\mathbf{x}_1) & \tilde{I}_{ref\,1}(\mathbf{x}_2) & \tilde{I}_{ref\,1}(\mathbf{x}_3) & \tilde{I}_{ref\,1}(\mathbf{x}_4) \\ \tilde{I}_{ref\,2}(\mathbf{x}_1) & \tilde{I}_{ref\,2}(\mathbf{x}_2) & \tilde{I}_{ref\,2}(\mathbf{x}_3) & \tilde{I}_{ref\,2}(\mathbf{x}_4) \\ \tilde{I}_{ref\,3}(\mathbf{x}_1) & \tilde{I}_{ref\,3}(\mathbf{x}_2) & \tilde{I}_{ref\,3}(\mathbf{x}_3) & \tilde{I}_{ref\,3}(\mathbf{x}_4) \end{bmatrix} = g_s \cdot \begin{bmatrix} \tilde{I}_{tr\,1}(\hat{\mathbf{x}}_1) & \tilde{I}_{tr\,1}(\hat{\mathbf{x}}_2) & \tilde{I}_{tr\,1}(\hat{\mathbf{x}}_3) & \tilde{I}_{tr\,1}(\hat{\mathbf{x}}_4) \\ \tilde{I}_{tr\,2}(\hat{\mathbf{x}}_1) & \tilde{I}_{tr\,2}(\hat{\mathbf{x}}_2) & \tilde{I}_{tr\,2}(\hat{\mathbf{x}}_3) & \tilde{I}_{tr\,2}(\hat{\mathbf{x}}_4) \\ \tilde{I}_{tr\,3}(\hat{\mathbf{x}}_1) & \tilde{I}_{tr\,3}(\hat{\mathbf{x}}_2) & \tilde{I}_{tr\,3}(\hat{\mathbf{x}}_3) & \tilde{I}_{tr\,3}(\hat{\mathbf{x}}_4) \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.11}$$

The least-square solution of g_s is computed and further applied for adjustment.

Polynomial regression

Since Function (4.9) is difficult to calculate, we attempt to approach g by regression analysis. What is more, our experience shows that the correlation of channels is weak. To simplify this problem, we decide to estimate each channel independently. Our regression model is a n-degree polynomial of the form:

$$g(I_{tr\,i}(\hat{\mathbf{x}})) = I_{ref\,i}(\mathbf{x}) = \sum_{k=0}^{n} \alpha_{k\,i}\, I_{tr\,i}(\hat{\mathbf{x}})^{\,k}, \quad i \in \{1, 2, 3\}.$$

The choice of n is a decisive factor to estimation accuracy. A high degree approximates a curve by a precise description of samples, but at the same time, it makes the curve too sensitive to noises. In our tests, we have found once the false color of tiny area at degree 2. Therefore, we stop trying higher degrees and we decide to choose polynomial of degree 1 and 2 as two possible models of our

estimation. The function becomes either linear or quadratic:

$$g_l(I_{tr\,i}(\hat{\mathbf{x}})) = I_{ref\,i}(\mathbf{x}) = \alpha_{l0\,i} + \alpha_{l1\,i}\, I_{tr\,i}(\hat{\mathbf{x}}) \qquad or \tag{4.12}$$

$$g_q(I_{tr\,i}(\hat{\mathbf{x}})) = I_{ref\,i}(\mathbf{x}) = \alpha_{q0\,i} + \alpha_{q1\,i}\, I_{tr\,i}(\hat{\mathbf{x}}) + \alpha_{q2\,i}\, I_{tr\,i}(\hat{\mathbf{x}})^2. \tag{4.13}$$

Regression parameters are computed by pixel pairs. Taken Function (4.13) as example, We rewrite the relationship of a single channel between reference and its matched pixels in form of vector multiplication as:

$$I_{ref\,i}(\mathbf{x}) = \left[\ 1\ I_{tr\,i}(\hat{\mathbf{x}})\ I_{tr\,i}(\hat{\mathbf{x}})^2\ \right] \cdot \left[\ \alpha_{q0\,i}\ \ \alpha_{q1\,i}\ \ \alpha_{q2\,i}\ \right]^T$$

It requires three pairs of vectors $(\mathbf{I}_{tr}(\hat{\mathbf{x}}_r), \mathbf{I}_{ref}(\mathbf{x}_r))$, $r \in \{1, 2, 3\}$ to calculate the coefficients $\alpha_{q0\,i}$, $\alpha_{q1\,i}$, $\alpha_{q2\,i}$. We write them together in an matrix as:

$$\begin{bmatrix} I_{ref\,i}(\mathbf{x}_1) \\ I_{ref\,i}(\mathbf{x}_2) \\ I_{ref\,i}(\mathbf{x}_3) \end{bmatrix} = \begin{bmatrix} 1 & I_{tr\,i}(\hat{\mathbf{x}}_1) & I_{tr\,i}(\hat{\mathbf{x}}_1)^2 \\ 1 & I_{tr\,i}(\hat{\mathbf{x}}_2) & I_{tr\,i}(\hat{\mathbf{x}}_2)^2 \\ 1 & I_{tr\,i}(\hat{\mathbf{x}}_3) & I_{tr\,i}(\hat{\mathbf{x}}_3)^2 \end{bmatrix} \begin{bmatrix} \alpha_{q0\,i} \\ \alpha_{q1\,i} \\ \alpha_{q2\,i} \end{bmatrix}. \tag{4.14}$$

A similar expression, obtained from Function (4.12) with two pairs of vectors, is presented as:

$$\begin{bmatrix} I_{ref\,i}(\mathbf{x}_1) \\ I_{ref\,i}(\mathbf{x}_2) \end{bmatrix} = \begin{bmatrix} 1 & I_{tr\,i}(\hat{\mathbf{x}}_1) \\ 1 & I_{tr\,i}(\hat{\mathbf{x}}_2) \end{bmatrix} \begin{bmatrix} \alpha_{l0\,i} \\ \alpha_{l1\,i} \end{bmatrix}. \tag{4.15}$$

The coefficients in Function (4.14) and Function (4.15) can be calculated by least-square estimation.

Remark that we use the same pairs of matched pixels to compute parameters of different channels in order to avoid errors caused by the estimation on samples of different deviations. There are in total 6 parameters to decide for linear approximation (9 for quadratic approximation), the number of vector pairs in need remains 2 (3 for quadratic approximation).

4.3.3 Sample selection

Our designs require respectively 2, 3 or 4 pairs of matched pixels to decide the mapping parameters which will be used in all range of image. A random selection of samples is highly risky and that may lead to the failure of adjustment. For sample selection, two major sources of mistakes should be taken into account:

1) Some samples do not convey the same content of scene. That will lead to a total mistake in the computation of parameters. Images corrected by this kind of models will bear chromatic distortion in all range of scene.

2) Samples convey the content of masks. Chromatic transformation models of different objects are probably not the same. A mapping that adjusts successfully certain occlusion may perform incorrectly in the background region.

Therefore, we apply RANSAC strategy (see Chapter 3) to enhance the robustness of sample selection. The model with largest number of inliers should describe the chromatic transformation of background since it is supposed to occupy the majority of image range. Models computed by incorrect samples (either case 1 or 2) will be rejected during the selection due to its little amount of inliers.

4.4 Experiments

Discussion in previous sections give a general description on our designs while many details such as the selection of samples, the choice of parameters are still not decided. This section concentrates on the algorithms and provides evaluation on the performance of different models. Section 4.4.1 presents the algorithms of transfer models and the related choices. Section 4.4.2 evaluates quantitatively and qualitatively the results of chromatic adjustments.

4.4.1 Algorithm details

Our idea is to uniform the contrast of images by proceeding chromatic adjustment between the reference image and any other images in sequence. sRGB method, linear and quadratic approximations in Section 4.3.2 are described by Algorithms 3-5. As we come to the stage of experimental tests, parameters in algorithms and sampling resources should be discussed now. Remark that chromatic adjustment is proceeded after geometric alignment (see Chapter 3), our design will be based on aligned images and some results of previous step.

Despite of the computation of coefficients, RANSAC strategy is applied similarly in these algorithms. It is therefore reasonable to share RANSAC parameters which are the number of iteration times and threshold of residual differences between reference and adjusted RGB vectors. We choose 1000 as the number of iterations times which provides us with enough computed models for selection. In linear and quadratic adjustments, threshold for residual errors per channel

Algorithm 3: Chromatic adjustment with sRGB model

Data: Pixel pair positions $\mathbf{X}_0 = \{(\hat{\mathbf{x}}, \mathbf{x}) | \, \hat{\mathbf{x}} \in \Omega_{tr}, \mathbf{x} \in \Omega_{ref}\}$;
Image to be adjusted $\mathbf{I}_{tr}(\Omega_{tr})$; reference image $\mathbf{I}_{ref}(\Omega_{ref})$.
Result: Chromatic corrected image $\mathbf{I}_s(\Omega_{tr})$.
Initialize $g_s = 0_{3\times4}$, $N_{inlier} = 0$;
Map $\mathbf{I}_{tr}(\Omega_{tr})$ to $\tilde{\mathbf{I}}_{tr}(\Omega_{tr})$ (resp. $\mathbf{I}_{ref}(\Omega_{ref})$ to $\tilde{\mathbf{I}}_{ref}(\Omega_{ref})$) by Function (4.7);
for $n = 1$ **to** 1000 **do**
  Select randomly 4 pixel pairs in $\mathbf{X}_0$ to calculate g_{iter} by solving Function (4.11);
  foreach $(\hat{\mathbf{x}}, \mathbf{x}) \in \mathbf{X}_0$ **do**
    Compute adjusted vector $\tilde{\mathbf{I}}_{new}(\hat{\mathbf{x}})$ of $\tilde{\mathbf{I}}_{tr}(\hat{\mathbf{x}})$ by Function (4.10);
    if $\forall i \in \{1, 2, 3\}$, $|\tilde{I}_{ref\,i}(\mathbf{x}) - \tilde{I}_{new\,i}(\hat{\mathbf{x}})| \leq \tilde{\epsilon}_c$ **then**
      $\mathbf{X}_{iter} \leftarrow (\hat{\mathbf{x}}, \mathbf{x})$;
    end
  end
  if $card(\mathbf{X}_{iter}) > N_{inlier}$ **then**
    Update $N_{inlier} = card(\mathbf{X}_{iter})$, $g_s = g_{iter}$;
  end
  Clear $\mathbf{X}_{iter}$;
end
Compute adjusted results $\tilde{\mathbf{I}}_s(\Omega_{tr})$ of $\tilde{\mathbf{I}}_{tr}(\Omega_{tr})$ by Function (4.10);
Map $\tilde{\mathbf{I}}_s(\Omega_{tr})$ back to $\mathbf{I}_s(\Omega_{tr})$ with g_s by Function (4.6);
return $\mathbf{I}_s(\Omega_{tr})$.

Algorithm 4: Chromatic adjustment with linear model

Data: Pixel pair positions $\mathbf{X}_0 = \{(\hat{\mathbf{x}}, \mathbf{x}) | \, \hat{\mathbf{x}} \in \Omega_{tr}, \mathbf{x} \in \Omega_{ref}\}$;
Image to be adjusted $\mathbf{I}_{tr}(\Omega_{tr})$; reference image $\mathbf{I}_{ref}(\Omega_{ref})$.
Result: Chromatic corrected image $\mathbf{I}_l(\Omega_{tr})$.
Initialize $N_{inlier} = 0$, $\forall i \in \{1, 2, 3\}$ $\alpha_{l0\,i} = 0$, $\alpha_{l1\,i} = 0$;
for $n = 1$ **to** 1000 **do**
  Select randomly 2 pixel pairs in $\mathbf{X}_0$ to calculate $\alpha_{iter0\,i}$ $\alpha_{iter1\,i}$,$\forall i \in \{1, 2, 3\}$ by solving Function (4.15) ;
  foreach $(\hat{\mathbf{x}}, \mathbf{x}) \in \mathbf{X}_0$ **do**
    Compute adjusted vector $\mathbf{I}_{new}(\hat{\mathbf{x}})$ of $\mathbf{I}_{tr}(\hat{\mathbf{x}})$ by Function (4.12);
    if $\forall i \in \{1, 2, 3\}$, $|I_{ref\,i}(\mathbf{x}) - I_{new\,i}(\hat{\mathbf{x}})| \leq \epsilon_c$ **then**
      $\mathbf{X}_{iter} \leftarrow (\hat{\mathbf{x}}, \mathbf{x})$;
    end
  end
  if $card(\mathbf{X}_{iter}) > N_{inlier}$ **then**
    Update $N_{inlier} = card(\mathbf{X}_{iter})$;
    $\forall i \in \{1, 2, 3\}$, $\alpha_{l0\,i} = \alpha_{iter0\,i}$, $\alpha_{l1\,i} = \alpha_{iter1\,i}$;
  end
  Clear $\mathbf{X}_{iter}$;
end
Compute adjusted results $\mathbf{I}_l(\Omega_{tr})$ of $\tilde{\mathbf{I}}_{tr}(\Omega_{tr})$ with $\alpha_{l0\,i}$, $\alpha_{l1\,i}$ by Function (4.12);
return $\mathbf{I}_l(\Omega_{tr})$.

Algorithm 5: Chromatic adjustment with quadratic model

Data: Pixel pair positions $\mathbf{X}_0 = \{(\hat{\mathbf{x}}, \mathbf{x}) | \, \hat{\mathbf{x}} \in \Omega_{tr}, \mathbf{x} \in \Omega_{ref}\}$;
Image to be adjusted $\mathbf{I}_{tr}(\Omega_{tr})$; reference image $\mathbf{I}_{ref}(\Omega_{ref})$.
Result: Chromatic corrected image $\mathbf{I}_q(\Omega_{tr})$.
Initialize $N_{inlier} = 0$, $\forall i \in \{1, 2, 3\}$ $\alpha_{q0\,i} = 0$ $\alpha_{q1\,i} = 0$ $\alpha_{q2\,i} = 0$;
for $n = 1$ **to** 1000 **do**
  Select randomly 3 pixel pairs in $\mathbf{X}_0$ to calculate $\alpha_{iter0\,i}$ $\alpha_{iter1\,i}$ $\alpha_{iter2\,i}$, $\forall i \in \{1, 2, 3\}$ by solving Function (4.14) ;
  foreach $(\hat{\mathbf{x}}, \mathbf{x}) \in \mathbf{X}_0$ **do**
    Compute adjusted vector $\mathbf{I}_{new}(\hat{\mathbf{x}})$ of $\mathbf{I}_{tr}(\hat{\mathbf{x}})$ by Function (4.13);
    if $\forall i \in \{1, 2, 3\}$, $|I_{ref\,i}(\mathbf{x}) - I_{new\,i}(\hat{\mathbf{x}})| \leq \epsilon_c$, **then**
      $\mathbf{X}_{iter} \leftarrow (\hat{\mathbf{x}}, \mathbf{x})$;
    end
  end
  if $card(\mathbf{X}_{iter}) > N_{inlier}$ **then**
    Update $N_{inlier} = card(\mathbf{X}_{iter})$;
    $\forall i \in \{1, 2, 3\}$, $\alpha_{q0\,i} = \alpha_{iter0\,i}$, $\alpha_{q1\,i} = \alpha_{iter1\,i}$, $\alpha_{q2\,i} = \alpha_{iter2\,i}$;
  end
  Clear $\mathbf{X}_{iter}$;
end
Compute adjusted results $\mathbf{I}_q(\Omega_{tr})$ of $\tilde{\mathbf{I}}_{tr}(\Omega_{tr})$ with $\alpha_{q0\,i}$, $\alpha_{q1\,i}$ $\alpha_{q2\,i}$ by Function (4.13);
return $\mathbf{I}_q(\Omega_{tr})$.

is set at $\epsilon_c = 0.01$ for normalized intensities ranging within $[0, 1]$. This value is decided after lots of tests confirming that the threshold can be assigned to a fixed number and 0.01 is a good choice which always leaves enough inliers for models. In sRGB approach, the threshold is modified as $\tilde{\epsilon}_c = 0.0007$ following the non-linear mapping in Function (4.7).

Apart from parameters in algorithms, the choice of samples has not been decided. Following previous discussion, most of the samples should be pixel pairs conveying the same view of background. A small amount of badly matched pairs are tolerable as they will be discarded by RANSAC selection. While the majority of samples should be accurately matched and be able to represent multiple colors appearing in image. Based on these considerations, we come up with two ideas which both have their pros and cons:

SIFT results The first idea is to use SIFT results issued from geometric alignment step. These matched pixel pairs are reliable while they often present a limitation on color range. Because SIFT chooses pixels located in the area with highly variation of contrast despite of their colors. Therefore, colors of matched

pixels concentrate probably on certain types and the accuracy of transfer function computed with these pixels remains to be tested.

Pixels on grid We assume that pixels from different images superpose one to another except in area with masks. It is possible to define a grid (for example with elements of $5px \times 5px$) and select pixels on grid as our samples. This design benefits from a well distributed samples on a large color range, but at the same time, the grid increases the complexity of outlier cases. Thus, this design counts on an accurate geometric alignment and a careful RANSAC selection.

(a) Reference image (woman)

(b) Adjusted result (woman)

(c) Reference image (planet)

(d) Adjusted result (planet)

FIGURE 4.3: **Examples of samples on grid** They are adjusted by Algorithm 5 with samples on grids. 'Woman' is well aligned, 'planet' has 2-pixel errors.

Experiments confirms that SIFT results are better choices than pixels en grid. Figure 4.3 helps to explain our abandon of grid. After geometric alignment, series 'woman' performs well with nearly no error while 'planet' bears a few pixels of

mismatch. The result of 'woman ' is well presented which means the minority of badly matched pixels caused by masks are correctly eliminated by RANSAC. However, 'planet' is wrongly adjusted with chromatic distortion. That means its transfer model is computed with mismatched pixels and RANSAC selection has failed in case where lots of pixels on grid are badly matched because of alignment errors. This kind of failures have never occurred in the approaches proceeded with SIFT results. In addition, tests show that both the quantity and the chromatic distribution of SIFT results meet the sampling requirements as is shown in Figure 4.4. It is thus the robust sample resources for our adjustments.

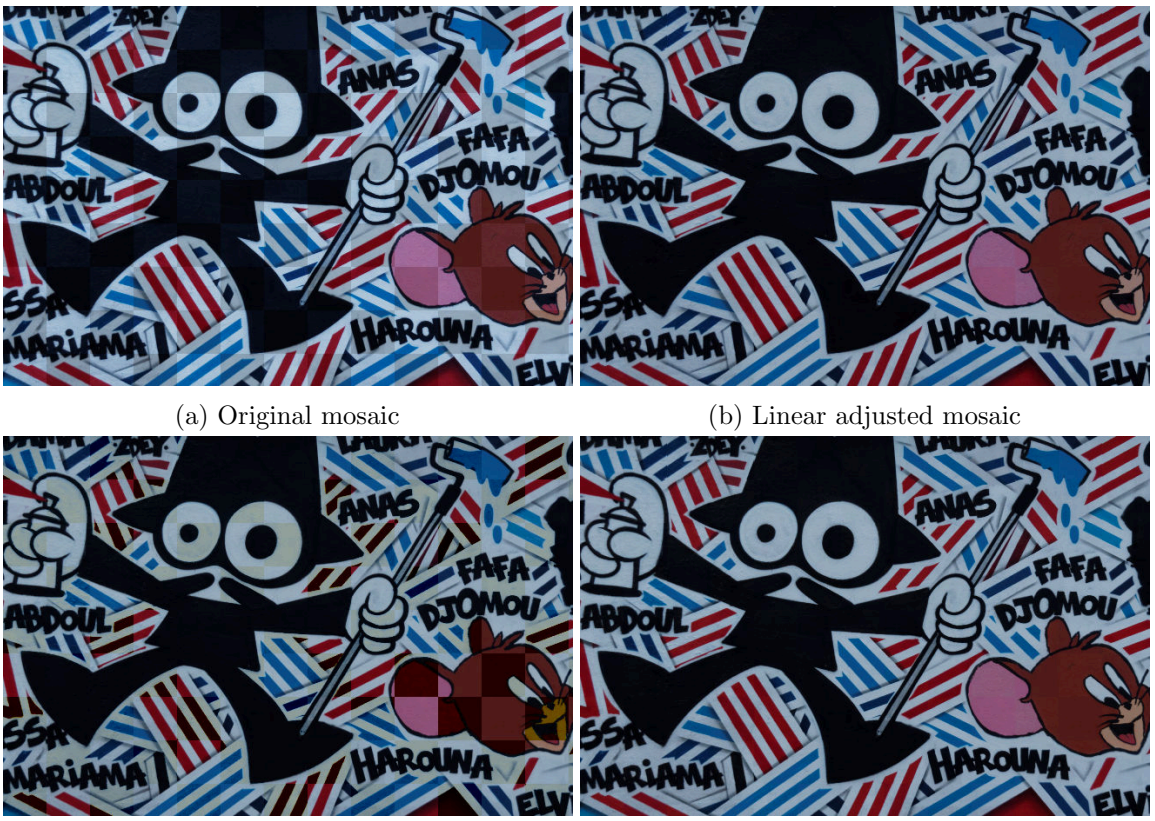
(a) Matched pixels in reference image (b) Matched pixels in aligned image

FIGURE 4.4: **SIFT sampling** Positions marked with red cross present 137 pairs of inliers of the color transfer model chosen by RANSAC among 746 SIFT samples. They are distributed in varies parts of background scene. Most of representative colors in background are included ranging from yellow to purple.

4.4.2 Estimation

In order to observe the color differences between images, we arrange the elements of two images in one mosaic as is shown in Figures 4.5, 4.6, 4.7. Differences are obvious in the areas that should be of uniform colors in the original view. In all the examples, linear and quadratic approximation plays always a positive role to align the contrast in images while sRGB performs randomly and even provides serious mistakes in Figures 4.5c, 4.6c. More concretely, the performance of quadratic adjustment is better than that of linear model. It provides always the most uniform mosaics among all. On the contrary, sRGB adjustment seems to worsen the contrast difference and the trend of change is out of control.

To further evaluate the performance of three methods, we calculate the root mean square error (RMSE) of matched RGB vectors before and after adjustments. The



(a) Original mosaic (b) Linear adjusted mosaic

(c) sRGB adjusted mosaic (d) Quadratic adjusted mosaic

FIGURE 4.5: **Chromatic adjusted results (School).** Linear and quadratic adjustment provide similar mosaic results of uniform colors. sRGB adjustment seems to have eliminated the differences of white area. But its result presents extreme mistakes in the region of red colors.

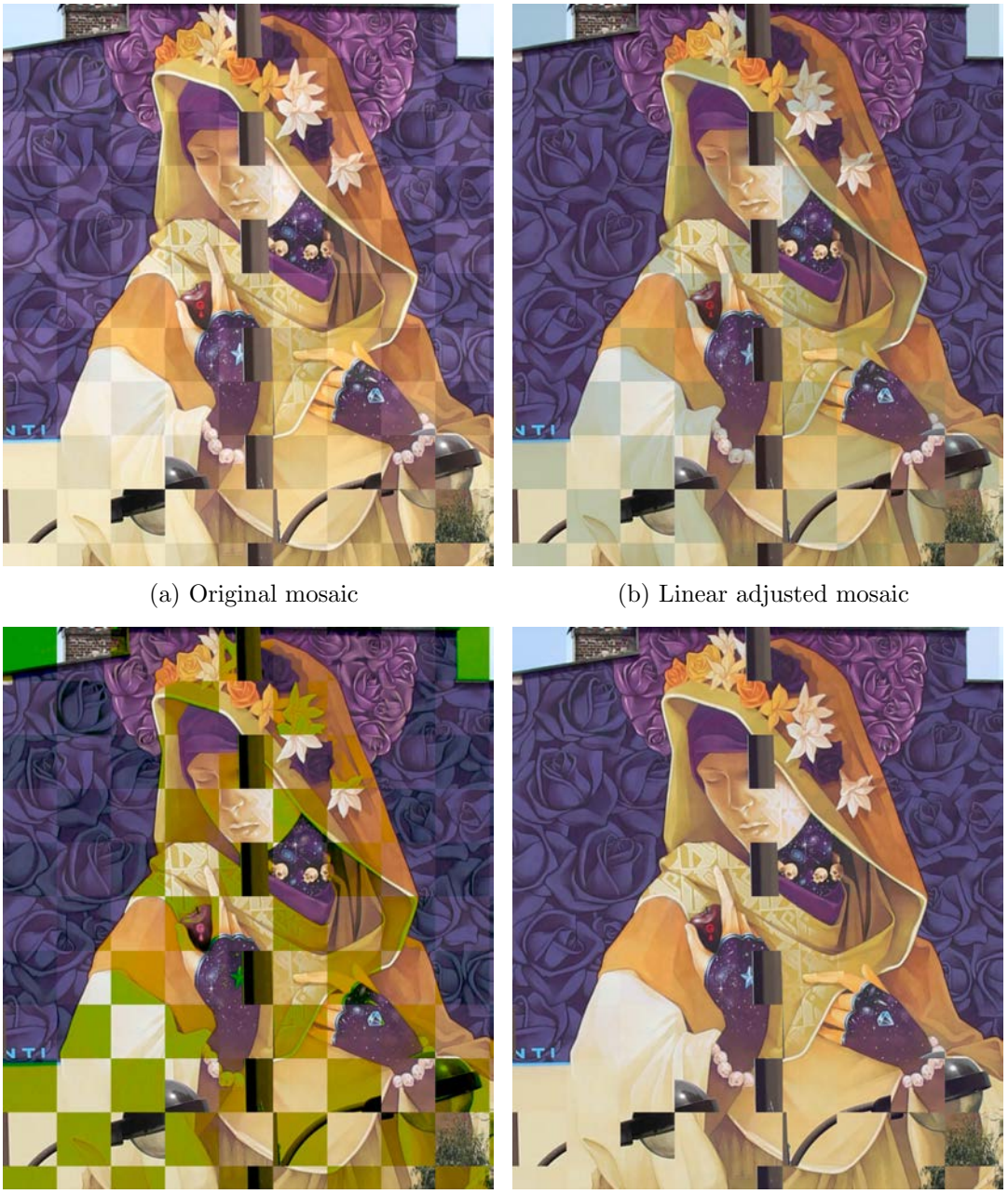
estimation is limited to samples so as to exclude mask pixels. Table 4.1 shows the results of five sequences computed between the reference image and ith image (marked as $1{-}i$) with normalized values ranging from 0 to 1. More experimental facts can be found in Chapter 8.

The results of RMSE confirm our judgment through observation. Quadratic and linear approximation reduce the contrast differences between images and their RMSE decrease nearly all the time. These methods usually give similar results while quadratic model shows its advantage in case where the original differences are important. What's more, quadratic approximation performs robustly as it never results in RMSE larger than its value before adjustment. On the contrary, sRGB approach is not an appropriate method for color alignment as it always causes failures, A possible explication is that sRGB model may not include entirely the image processing steps inside the camera we use, some more non linear transformations apart from Function (4.6) probably exist in its pipeline.

Over all, we come to a conclusion that both polynomial approximation methods are effective for chromatic adjustment and quadratic polynomial provide better performance. In all tests, RMSE values after quadratic adjustment never exceed 15 (i.e. 0.067 in range of 0 to 1). This value provides a criterion for determining whether two pixels express the same content. It will be used in the later designs.

		Original	**Linear**	**Quadratic**	**sRGB**
School	1-2	0.038	0.017	0.015	0.057
	1-3	0.044	0.015	0.015	0.432
	1-4	0.072	0.021	0.018	0.350
	1-5	0.054	0.014	0.013	0.391
Rose	1-2	0.125	0.046	0.037	1.051
	1-3	0.144	0.084	0.033	1.652
	1-4	0.139	0.085	0.035	3.919
	1-5	0.149	0.089	0.036	2.794
Dancer	1-2	0.214	0.056	0.039	2.219
	1-3	0.238	0.058	0.042	7.667
	1-4	0.231	0.069	0.047	5.477
	1-5	0.253	0.067	0.042	4.525
Woman	1-2	0.015	0.013	0.013	0.316
	1-3	0.017	0.014	0.014	0.123
	1-4	0.034	0.014	0.014	8.593
	1-5	0.024	0.014	0.014	2.807
Bakery	1-2	0.021	0.021	0.021	0.421
	1-3	0.028	0.021	0.024	1.906
	1-4	0.024	0.025	0.024	1.733
	1-5	0.034	0.024	0.025	0.488

TABLE 4.1: **RMSE of matched pixels.** $1{-}i$ means the photometric transformation is proceeded between the first and the i-th images in sequence. Values in the table are the RMSE of SIFT matches before/after adjustment. For normalized images, intensities vary by steps of $1/256 = 0.004$. So we assume that color differences under 0.004 can not be distinguished by eyes and RMSEs under $0.004 \times \sqrt{3} = 0.007$ are negligible.



(a) Original mosaic (b) Linear adjusted mosaic

(c) sRGB adjusted mosaic (d) Quadratic adjusted mosaic

FIGURE 4.6: **Chromatic adjusted results (Rose).** Linear approximation correct a part of the contrast differences in two images. It performs well in the adjustment of purple area as the color of roses becomes uniform. But the yellow region is not correctly expressed. sRGB gives a total failure of adjustment in this example with wrong colors everywhere. The result of quadratic adjustment is acceptable in all range of background with every types of color well aligned. No obvious contrast difference is found in mosaic.

(a) Original mosaic

(b) Linear adjusted mosaic

(c) sRGB adjusted mosaic

(d) Quadratic adjusted mosaic

FIGURE 4.7: **Chromatic adjusted results (Dancer).** Original mosaic shows an obvious contrast difference, the reference is much clear than another image. Linear approximation ameliorate the mosaic but it is still easy to distinguish the dark patches from the bright ones. sRGB model gives incorrect result which becomes green in all range of image. Quadratic approximation provides the best result with a uniform appearance of image mosaic.

Part II

Fusion Methods

Chapter 5

Median Filtering

From now on, we suppose that images have been aligned and photometrically corrected. Here starts a discussion on how to restore the hidden parts. We consider the case where we do not make any additional assumption on masks.

Inpainting methods are a possible choices which have been widely used to complete lost parts of images by extending geometrical and statistical properties from the existing region to the missing area [BSCB00, CPT04]. However, it depends on the most likely patches searched outside the occlusion in the image to fill the gaps. These patches are often quite different from the real missing parts when the texture of background varies greatly or has nothing in common with the surrounding area.

Since we have several images at our disposal, it is possible to restore the real scene behind masks. So we attempt to distinguish the background according to their appearance frequency. At each pixel position, we get a stack of RGB vectors each being issued from one image. These vectors can be divided into two categories that convey respectively the background and the masks. If we take into account the quantity of the two sorts of pixels, there should be four possible situations for each stack:

1) All pixels are of background.
2) A majority of pixels are of background and a minority of pixels are of masks;
3) A minority of pixels are of background and a majority of pixels are of masks;
4) All pixels are of masks.

In this chapter, we focus particularly on case 1) and 2). We choose to apply a filter which is not sensitive to unknown values of masked pixels. A good candidate would be the median filter which was once applied in [YSL04] for gray-scale light

field completion. It provides a good estimate under two constraints: 1) There exists certain camera positions where hidden parts can be seen. 2) The areas to be restored are revealed in more than half of the aligned images.

Compared with previous work, our case is more complicated as the pixels of color images convey 3-dimensional vectors rather than gray-scale scalars. We need to choose an adequate definition of median filtering for the this case.

This chapter consists of three parts. Section 5.1 introduces the median filter. Then, we propose some possible vector filtering methods for color image fusion in Section 5.2. Section 5.3 estimates the performance of different approaches. Finally, we will analyze their results and give our conclusions.

5.1 Median and median filter

We start from a short review on median definition and properties. Then, we present the previous application of median filtering and explain our reason to choose it as the mask removal method.

5.1.1 Median

The median is a common concept in scalar field. It refers to the middle number of a sorted data set. Its position-dependency character results in two definitions depending on the parity (odd/even) of member quantity [Hub11]:

Definition 5.1. Given $n \in \mathbb{N}$, $\forall 1 \leq i \leq n$, $s_i \in \mathbb{R}$. Let $\mathbf{S} = \{s_i\}$ be a sorted set (in descending or ascending order). The median of $\mathbf{S}$ is :

$$med(\mathbf{S}) := \begin{cases} s_{\frac{n+1}{2}} & n \in 2\mathbb{N}+1 \\ \frac{1}{2}\left(s_{\frac{n}{2}} + s_{\frac{n}{2}+1}\right) & n \in 2\mathbb{N}. \end{cases}$$

Definition shows that a median value is characterized by the order rather than the value of members in data set. It benefits from a low sensitivity to the extreme values. The median satisfies the optimality property described by Theorem 5.2.

Theorem 5.2. *Median of a set* $\mathbf{S} = \{s_i\}$, $i \in \{1, \ldots, n\}$ *minimizes the sum of absolute difference between a set member and itself:*

$$\forall j \in \{1, \ldots, n\}, \quad \sum_{i=1}^{n} \mid med(\mathbf{S}) - s_i \mid \leq \sum_{i=1}^{n} \mid s_j - s_i \mid .$$

A search of median can be achieved by approximating iteratively the value that minimizes its distance to data set members. However, the data set is required to have odd number of members. In Definition 5.1, when n is even, median will be a value out of data set between $s_{\frac{n}{2}}$ and $s_{\frac{n}{2}+1}$.

Proposition 5.3. *Given an odd-member data set* $\mathbf{S}=\{s_i\}$, $s_i \in \mathbb{R}$, $i \in \{1, \ldots, n\}$, $n \in 2\mathbb{N}$. *Median of the set is a member that minimizes its distance to other members:*

$$med(\mathbf{S}) = \arg\min_{s \in \mathbb{R}} \sum_{i=1}^{n} \mid s - s_i \mid .$$

This limitation can be broken in image fusion problems as the median value will be selected among the data set members. This is not only for convenience, but also because we need to avoid new created pixel values (artificial colors) in fusion results. Therefore, strictly speaking, we search for a member of the data set which is closest to the median value.

5.1.2 Median filter

When proceeding the median filter in image processing, a mobile window passes through a data area and leaves the median value in each window. In all, previous applications of median filter are always limited to gray-scale despite of problem contexts.

A widely used application is image noise reduction as median filter is able to eliminate the extreme values in the neighborhood. A window of fixed-pattern passes over each pixel and replaces its intensity by the median value of neighborhood [SN94]. The image is smoothed with less noise. Median filter is especially effective in reducing impulse noises.

Another application of median filter is to remove masks in gray-scale image sequences [YSL04]. The data is a sequence of aligned images with masks in some areas. A window stretches across different images at each pixel position under an assumption that the number of background pixels exceeds half of the number of images in sequence. Therefore, the median of data set will be the background pixel and the hidden part is restored after filtering.

Our problem is very similar to the second case but we need to replace gray-scale images (as in [YSL04]) by color images. It is reasonable to make use of the basic idea of median selection, but actually there is not much reference on the vector median filtering. We need to find an appropriate median definition in vector range which performs well for mask removal of color image sequences.

5.2 Possible choices of vector median

We recall here our model. Given $n \in \mathbb{N}$ digital images defined on grid Ω after geometric and photometric alignments (see Chapters 3 and 4). For all pixel position $\mathbf{x} \in \Omega$, we wish to define median $\mathbf{med}(\mathbf{\Phi}(\mathbf{x}))$ of its vector data set $\mathbf{\Phi}(\mathbf{x}) = \{\mathbf{I}_1(\mathbf{x}), \ldots, \mathbf{I}_n(\mathbf{x})\}$, where $\mathbf{I}_i(\mathbf{x}) = (I_{ir}(\mathbf{x}), I_{ig}(\mathbf{x}), I_{ib}(\mathbf{x}))^T, \forall i \in \{1, \ldots, n\}$.

This section gives several possible definitions of the vector median. After a brief list of these possibilities, we will analyze their behaviors and select some of them for further tests in Section 5.3.

5.2.1 Definitions of vector median

We have four propositions of vector median. Definitions 5.4 and 5.5 relates to a direct search of vector while Definitions 5.6 and 5.7 pick out a vector corresponding to a certain type of scalar median.

Definition 5.4. Marginal median is a vector formed by the median on each channel [Pur71, MGS11] :

$$\mathbf{med}(\mathbf{\Phi}(\mathbf{x})) := (med(\{I_{ir}(\mathbf{x})\}), med(\{I_{ig}(\mathbf{x})\}), med(\{I_{ib}(\mathbf{x})\}))^T, i \in \{1, \ldots, n\}.$$

Definition 5.5. Geometric median is a vector minimizing the sum of its Euclidean distances to all other vector of data set [AHN90] :

$$\mathbf{med}(\mathbf{\Phi}(\mathbf{x})) := \underset{\mathbf{I}(\mathbf{x}) \in \mathbf{\Phi}(\mathbf{x})}{\arg\min} \sum_{i=1}^{n} \|\mathbf{I}(\mathbf{x}) - \mathbf{I}_i(\mathbf{x})\|. \tag{5.1}$$

Definition 5.6. Degenerated median is a vector corresponding to the median of intensity value. The conversion function from RGB vector to its intensity value is defined according to [AMCS96].

$$s_i(\mathbf{x}) := 0.2126 I_{ir}(\mathbf{x}) + 0.7152 I_{ig}(\mathbf{x}) + 0.0722 I_{ib}(\mathbf{x}). \tag{5.2}$$
$$\mathbf{med}(\mathbf{\Phi}(\mathbf{x})) := \mathbf{I}_j(\mathbf{x}) \in \mathbf{\Phi}(\mathbf{x}) \text{ such that } s_j(\mathbf{x}) = med(\{s_i(\mathbf{x}), i \in \{1, \ldots, n\}\}).$$

Definition 5.7. Space-curve median is a vector corresponding to the median of space-filling curve orders (see Figure 5.1). The order is length of curve from origin to the end of vector [SLP83]. If we represent order mapping as $\mathbf{I}(\mathbf{x}) \to c(\mathbf{I}(\mathbf{x}))$, then space-curve median is defined as:

$$\mathbf{med}(\mathbf{\Phi}(\mathbf{x})) := \mathbf{I}(\mathbf{x}) \in \mathbf{\Phi}(\mathbf{x}) \quad \text{such that} \quad c(\mathbf{I}(\mathbf{x})) = med(\{c(\mathbf{I}_i(\mathbf{x}))\}),\ i \in \{1, \ldots, n\}.$$

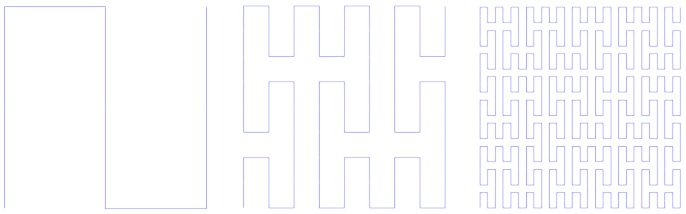


FIGURE 5.1: **Example of 2D space-filling curve construction [Wik07].** Space-filling curve is a continuous curve that passes through every elements in digital space. Its pattern depends on the function chosen for extension. This example shows how a space-filling curve (here a Peano curve) is constructed iteratively. It is possible to set up a bijection between vector and length of curve it cuts, namely curve order.

5.2.2 Performance prediction

Definitions in Section 5.2.1 are carefully designed and some of them have already been used in some applications. For sake of some particular demands in our case (prevention of artificial colors for example), we analyze here their behaviors and reject some choices before further tests.

Marginal median is the first definition to be abandoned. Indeed, medians of each channel belong probably to different vectors. In order to avoid generation of artificial colors, our application requires that the median result should belong to its original data set, which is not the case for the marginal median.

Geometric median extends median operation to vector space. It is well-designed as its definition degenerates to the definition of scalar median when the space dimension reduces to one. Unlike marginal median, geometric median is unique and belongs to the original data set by definition. Thus, we regard it as a good candidate of vector median for our further tests.

Degenerated median makes use of intensity value to express content of scene. Its feasibility is proved by previous mask removal method on gray-scale images [YSL04]. However, as we convert colored images to gray-scale images for median selection, pixel expression bears certainly a loss of information during their degeneration from vectors to scalars. It is still not sure if this process would lead to an ambiguity of colors, so the robustness of this method is to be estimated in further tests.

Space-curve median is directly rejected based on two major considerations: first, it is difficult to decide an appropriate pattern of space-filling lines to express vectors. Figure 5.2 shows how a bad choice of curve pattern leads to a wrong

selection of median. What's more, the estimation of mapping between vector and curve length introduces great computation. Space-curve median is not a good solution to our application.

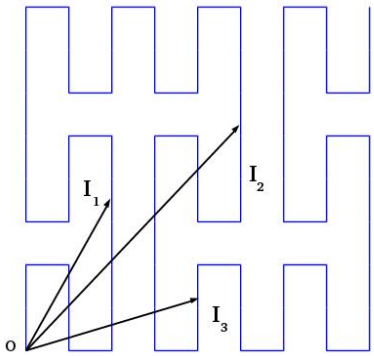


FIGURE 5.2: **Incorrect median caused by improper choice of space-filling curve pattern** We take 2D Peano curve as an example. Suppose that $\mathbf{I_1}$, $\mathbf{I_2}$ and $\mathbf{I_3}$ three vectors in 2D space. According to their ending points on Peano curve, their order obeys an inequality $c(\mathbf{I_1}) < c(\mathbf{I_2}) < c(\mathbf{I_3})$. Then the median of these orders should be $c(\mathbf{I_2})$ and $\mathbf{I_2}$ represents 'middle state' of three vectors, which is obviously incorrect.

Based on the analysis above, geometric median and degenerated median can be selected as the candidate methods for mask removal. Their performance remains to be evaluated in our experiments.

5.3 Experiments

Based on the discussion in Section 5.2, we test the performance of geometric and degenerated median according to Algorithms 6 and 7. The minimal sum of distance is taken as a criterion to pick out the median pixel for fusion results. Algorithm 6 calculates the distance based on Definition 5.5. Algorithm 7 computes at first the the gray-scale image related to each colored image according to Definition 5.2, then estimates the scalar distances following Proposition 5.3.

Many sequences have been put into use for evaluation, we present here three examples shown by Figures 5.3, 5.4 and 5.5. A more systematic assessment with quantitative and qualitative estimations of pixel selection methods can be found in Chapter 8.

Both median methods proceed effectively for mask removal when the assumption on background pixel quantity is achieved. The general appearance of their results are very similar to each other as is shown by Figures 5.3a and 5.3b, Figures 5.4a and 5.4d, Figures 5.5a and 5.5c.

However, the filtering qualities of these approaches are not exactly the same as we find many differences in the enlarged details. Degenerated median results present a lot of small areas of remaining masks (see Figures 5.3d, 5.4c and 5.5d). Geometric median results present better the background scene with less incorrect

Algorithm 6: Image fusion by geometric median filtering

Data: Aligned image sequence $\{\mathbf{I}_i(\Omega)\}$, $i \in \{1 \dots n\}$
Result: Median image $\mathbf{I}_{med}$
foreach $\mathbf{x} \in \Omega$ **do**
  Initialize $S_{min} = +\infty$;
  for $i = 1$ **to** n **do**
    Compute $S = \sum_{j=1}^{n} \|\mathbf{I}_i(\mathbf{x}) - \mathbf{I}_j(\mathbf{x})\|$;
    if $S < S_{min}$ **then**
      $S_{min} = S$;
      $I_{med}(\mathbf{x}) = \mathbf{I}_i(\mathbf{x})$;
    end
  end
end
return $\mathbf{I}_{med}(\Omega)$

Algorithm 7: Image fusion by degenerated median filtering

Data: Aligned image sequence $\{\mathbf{I}_i(\Omega)\}$, $i \in \{1 \dots n\}$
Result: Median image $\mathbf{I}_{med}$
foreach $\mathbf{I}_i(\Omega)$ *in sequence* **do**
  Compute $s_i(\Omega) = 0.2126 I_{ir}(\Omega) + 0.7152 I_{ig}(\Omega) + 0.0722 I_{ib}(\Omega)$;
end
foreach $\mathbf{x} \in \Omega$ **do**
  Initialize $S_{min} = +\infty$;
  for $i = 1$ **to** n **do**
    Compute $S = \sum_{j=1}^{n} |s_i(\mathbf{x}) - s_j(\mathbf{x})|$;
    if $S < S_{min}$ **then**
      $S_{min} = S$;
      $I_{med}(\mathbf{x}) = \mathbf{I}_i(\mathbf{x})$;
    end
  end
end
return $\mathbf{I}_{med}(\Omega)$

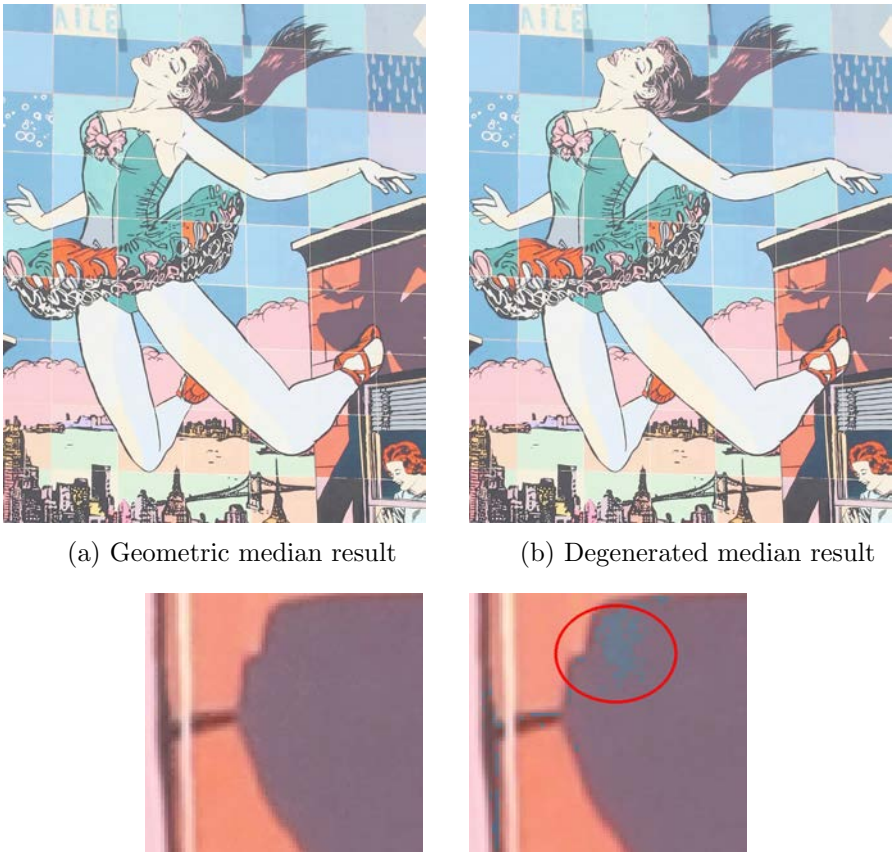
pixel selections. This phenomenon exist more or less in every image sequences depending on the intensity differences between the occlusion and the background. Later in Chapter 8, more obvious differences can be found between two median results in some examples.

The observations are consistent with our prediction that the degenerated computation results in a lose of color information. Mistakes may occur when pixels of significant color differences are expressed by similar intensity values. In reverse, geometric median, benefiting from the full information of RGB vectors,

can properly express the color relationship of pixels.

Therefore, we confirms that both extensions of vector median help to remove some of the masks in image sequence while the geometric median performs better. It is therefore the robust vector median filtering method for image fusion.

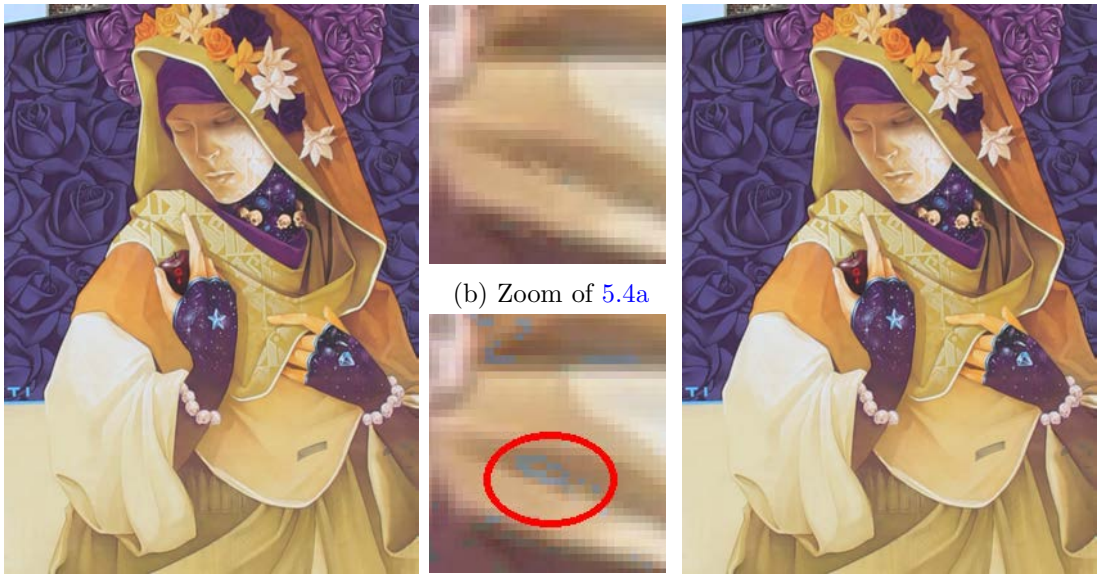
Nevertheless, the assumption on background pixel quantity is rather stringent. Many sequences, as for example the sequence in Figure 5.4, can not meet the requirement in all the range of image and their fusion results bear much remaining masks. Thus, it is necessary to design another method which is able to deal with more general cases. That will be discussed in Chapter 6.



(a) Geometric median result (b) Degenerated median result

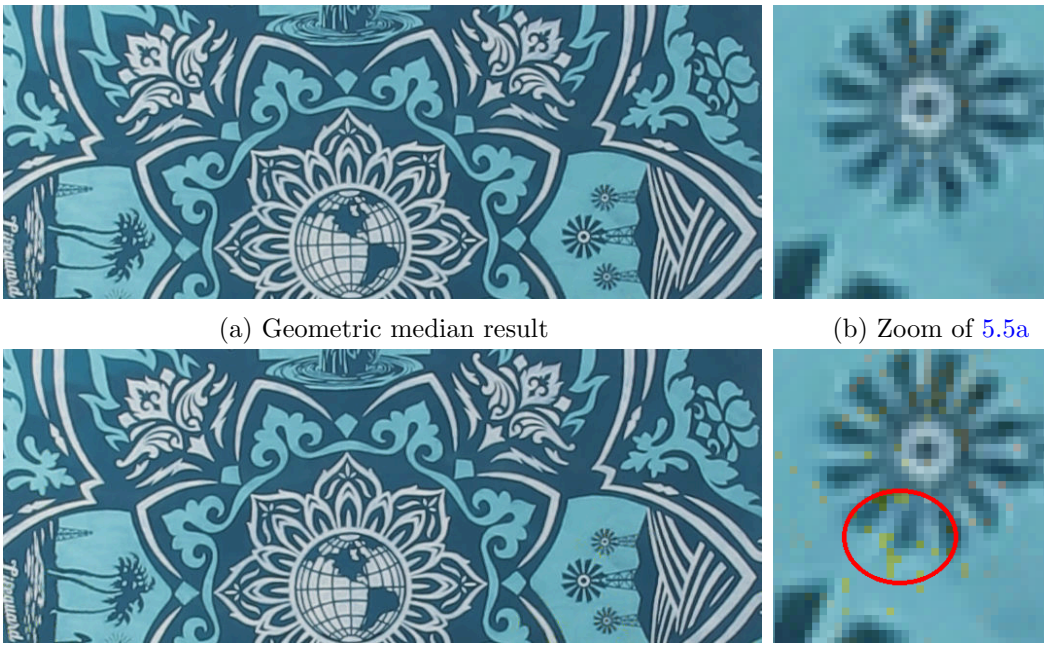
(c) Zoom of 5.3a (d) Zoom of 5.3b

FIGURE 5.3: **Fusion results (Dance).** The general appearance of two median results are similar to each other. But the enlarged details show that it exists many small areas of mask in the result of degenerated median filtering. Geometric median provides better result with accurate details.



(a) Geometric median result (b) Zoom of 5.4a (c) Zoom of 5.4d (d) Degenerated median result

FIGURE 5.4: **Fusion results (Rose).** Both median results leave a short bar on the girl's clothes of the fusion results. The quality of geometric median is better than degenerated median since small area of occlusions remains in the enlarged result of degenerated median image.



(a) Geometric median result (b) Zoom of 5.5a

(c) Degenerated median result (d) Zoom of 5.5c

FIGURE 5.5: **Fusion results (Orient).** No obvious mask has been left in this example. The enlarged figures show that geometric median performs better the details than degenerated median. Some yellow spots belonging to the mask remains in degenerated median result

Chapter 6

Meaningful Clique

As shown in Chapter 5, median filter works well under the assumption that more than 50% of the pixels belong to the background, but this requirement is rather stringent to reach within all range of images. As for example, we meet often the case where several pedestrians wander into the view one after another, only to leave a few revealed pixels of the background. Consequently, the median decision fails and selects blindly the pixels of mask according to their orders.

Essentially, these failures are due to the difference between problem features and the assumption of median filtering. Median filtering concerns the rank of elements in a data set. Yet, for our problem, the pixels of background share similar intensities while the masks vary randomly. Therefore, the density of distribution, rather than the order of intensities, is more appropriate to describe the background pixels.

Hence, this chapter is devoted to a design of pixel selection process based on the intensity distribution. Inspired by the concepts in graph theory, we gather the background information with a new-defined data set named meaningful clique. The algorithm is shown to fit with more general mask removal cases. It provides a better performance than median filter in terms of quality and reliability.

This chapter is organized as follows: first, Section 6.1 describes the special characteristics of background pixels. Then, Section 6.2 gives a short review of some concepts in graph theory. Later, Section 6.3 goes into details the definition of meaningful clique and the procedures of our new algorithm. Finally, Section 6.4 evaluates qualitatively and quantitatively the performances of median filter and our meaningful clique method.

6.1 Feature of background pixels

After the steps presented in Chapter 3 and Chapter 4, pixels at the same position on image grid convey the view of a fixed point captured at different moments. On the one hand, the background pixels bear the similar colors as they share the same content, on the other hand, the mask pixels show big differences due to the movement of both photographer and obstructions.

Mathematically, these pixels are expressed as discrete points in RGB space. The chromatic similarity (or difference) is expressed as the close (or long) distance between points. Therefore, the background pixels, which are similar to each other, gather as dense clouds. The mask pixels of various colors are more likely to spread about as the isolated points.

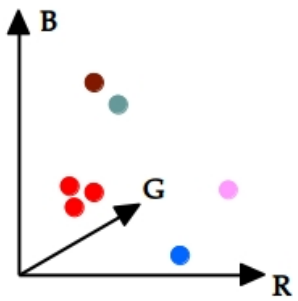


FIGURE 6.1: **Pixel distribution in color space.** We express the pixels as the points in RGB space. The three red points represent the background pixels form a dense cloud. The other points are the pixels of masks changing all the time. The brown and green points posit occasionally close. While they are not as numerous as the background pixels.

Remark that some mask pixels are also possible to gather as clouds. It happens when the motion is not enough to span the homogenous area of mask or when several occlusions appearing at the same position share coincidentally the similar colors. This phenomenon rarely occurs along with the perspective variation during photography and the clouds of these pixels are usually smaller than the cloud of background.

Still, to ensure the precision of our description, we assume that the repetition of similar masks should be less than that of the background. Different from the median filter which separates points into two groups regardless of their distances (see Chapter 5), we are in need of a method more adapted to the pixel distribution characteristic in RGB space that is able to distinguish the biggest dense cloud apart from other pixels.

6.2 Graph and clique

The cloud of points close to each other briefly described in Section 6.1 reminds us of the definition of clique in graph theory [BM+76]. Therefore, we give first a short review on several concepts that help us to design our own method, and we discuss then the modification according to our case.

Two basic notions should be mentioned here. We start from the concept of graph which is widely applied to define the data structure in many fields. It is often used to model a binary relationship between the objects.

Definition 6.1. A graph is a pair of sets $G(V, E)$ consisting of a set of vertices V and a set of edges E. An undirected graph is a graph in which edges have no orientation. [GR13]

There is a further definition on the subset of vertices whose members have links to each other, namely clique (first raised by [LP49]).

Definition 6.2. In an undirected graph $G(V, E)$, a subset $C \subseteq V$ of vertices V is called a clique if there exists edges to connect every two vertices of this subset. The maximum cliques are the cliques of the biggest cardinality [Gol04].

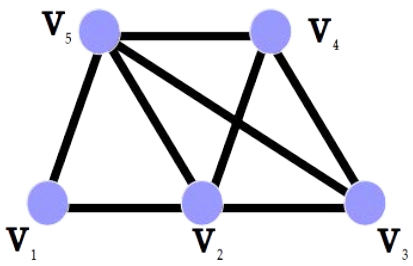


FIGURE 6.2: **Cliques in an undirected graph.** The diagram shows an example of undirected graph with its vertices $V = \{V_1, V_2, V_3, V_4, V_5\}$ and its edges represented by the black bars. According to definition 6.2, the vertices with bars connecting every two of them constitute cliques (for example $\{V_3, V_4\}$, $\{V_1, V_2, V_5\}$ and $\{V_2, V_4, V_5\}$). The clique of biggest cardinality, which refers to $\{V_2, V_3, V_4, V_5\}$ here, is the maximum clique.

The clique represents a set of vertices which are somehow related to each other among all the vertices in graph. Its definition is similar but not exactly identical to the description of cloud in Section 6.1. We regard the points in RGB space as the vertices and the Euclidean distances between points as the edges. According to Definition 6.2, a point can be classified simultaneously into several cliques as long as it is connected to every other members of the cliques. It means that a pixel can be marked at the same time as both background and masks.

Therefore, we should modify Definition 6.2 so as to eliminate the compatibility of cliques. Our idea is to cut off the connections between vertices (points) of different cliques by marking these edges (Euclidean distances) as invalid. The proposed definition will be presented in Section 6.3.

6.3 Clique based algorithm

We come back now to the core section of this chapter that gives a new algorithm for background simulation. We name a set of neighboring pixels as the dense clique.

This section consists of two parts. The first part focuses on the definition of dense clique and its search process. The second part gives a description of the meaningful clique as well as a general view of the whole iteration.

In our discussion, we input a stack of images after the geometric and photometric alignments described in Chapters 3 and 4

$$\mathbf{\Phi}(\mathbf{x}) = \{\mathbf{I}_i(\mathbf{x}),\ i \in \{1, \ldots, n\}\} \tag{6.1}$$

defined for all $\mathbf{x} \in \Omega \subset \mathbb{R}^2$, where $\mathbf{I}_i(\mathbf{x}) \in \mathbb{R}^3$. To estimate the background, for each pixel $\mathbf{x}$, we need to decide which value $\tilde{\mathbf{I}}(\mathbf{x})$ represents best the background.

6.3.1 Dense clique and search

Following the analysis in Section 6.2, we define dense cliques of cardinality m which are exclusive to each other by adding the valid/invalid mark on edges:

1) The edges are defined as Euclidean distance of points in $\mathbb{R}^3$;
2) For a set of m vertices $C \subset V$, if the edges linking $\forall v \in C$ and every other vertices in $C \setminus \{v\}$ are the $m-1$ shortest edges issued from v, then the edges between every two vertices of C are valid. Otherwise, all of them are invalid.

By doing so, the edges become distinguishing links. A m-member dense clique is a m-member subset $C \subseteq V$ of vertices if there exists valid edges to connect every two vertices of C. The definition is organized as follows:

Definition 6.3. (Dense clique) Let $v_1, \ldots, v_n \in \mathbb{R}^3$ and $V := \{v_1, \ldots, v_n\}$. A clique $C \subset V$ such that card $V = m$ is said dense if $\forall\ v \in C$ its $m-1$ nearest neighbors in V are in $C \setminus \{v\}$.

This definition ensures that the intersection of dense cliques of the same cardinality is always empty. It can be proved by contradiction: For $v \in V$, we suppose that it exists two m-member dense cliques $C_1 \neq C_2$ such that $v \in C_1$ and $v \in C_2$, then the $m-1$ nearest neighbors of v in V are also included in C_1 and C_2. Since C_1 and C_2 are of m members, $C_1 = C_2 = \{v\} \bigcup \{m-1 \text{ nearest neighbors of v}\}$.

In actual use, dense clique is the subsets of $\mathbf{\Phi}(\mathbf{x})$ and the points indicate pixel color channel intensity $\mathbf{I}_i(\mathbf{x})$. For every $\mathbf{x} \in \Omega$, the cliques given in Definition 6.3, applied with $\mathbf{\Phi}(\mathbf{x})$, can be computed using Algorithm 8.

Algorithm 8: Dense clique computation.

Data: Set $\mathbf{\Phi}(\mathbf{x})$ (see (6.1)), positive integer m
Result: Dense clique set $S(\mathbf{x})$.
Set $S = \emptyset$ and compute the $n \times m$ matrix M made with indexes of nearest neighbors (NN) of $\mathbf{I}_i(\mathbf{x})$ s.t. $\forall i \in \{1,\ldots,n\}, \text{row}(M,i) = (i, 1\text{st-NN} \ldots, m-1\text{th-NN})$.

```
for i=1,..., n do
    Compute the set E_1 := {M(i, 1 : m)}
    for j=2,..., m do
        Compute the set E_2 := {M(M(i, j), 1 : m)}
        if E_1 ≠ E_2 then
            Break
        end
        if j==m then
            S := S ∪ E_1
        end
    end
end
return S
```

The algorithm is a direct translation of Definition 6.3 by code. Each time, the search is carried out based on the observed value at $\mathbf{x}$, namely $\mathbf{I}_i(\mathbf{x}) \in \mathbf{\Phi}(\mathbf{x}) \, \forall i \in \{1,\ldots,n\}$ and its $m-1$ nearest neighbors that form together a candidate clique E_1. For every members $\mathbf{I} \in E_1$, we compute a set E_2 including $\mathbf{I}$ and its $m-1$ nearest neighbors. The candidate clique E_1 will be kept to the end only if it equals every time to such E_2.

6.3.2 Meaningful clique and iteration

As we have argued in Section 6.1, if a group of images display similar values, then one of these groups is assumed to be the background. Under ideal conditions, the clique of background should have the biggest cardinality. But in practice,

the case is more complicated as it involves some pseudo-dense cliques. It requires thus more constraints to define the clique we need, namely meaningful clique.

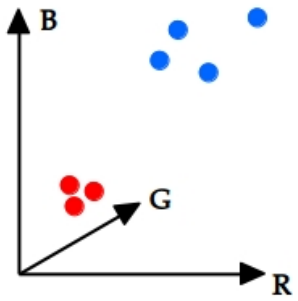


FIGURE 6.3: **Pseudo-dense clique.** The diagram shows an example of pseudo-dense clique. The four blue points is among the nearest three neighbors of each other although they are not really adjacent. If we search for clique of three members, we will get the real dense clique of the red points and an other clique of three blue points. As what we have designed, we will pass to the search of four-member cliques and we get only a clique of blue points. Without any constraint on density, this clique will be output by mistake.

We start with the explanation of pseudo-dense clique. As is represented by the group of blue points in Figure 6.3, it consists of several points that are rather dispersive. Even so, their distance to each other is relatively smaller than the distance to other points in the data set. Consequently, they are also selected as the candidate cliques according to Definition 6.3.

This problem comes from the lack of description on the density of clique. Definition 6.3 explains only the relationship between a isolated subset and other members in data set. No constraint is added to evaluate the intensity variance of pixels in the subset.

The unrestricted search of clique is designed due to the difficulty of threshold choice. A large threshold is useless while a small value may limit the growth of background clique (Sometimes the subset of repeated masks may have a smaller variance than the set of background, a badly selected threshold may reject the background clique but save the mask clique).

Therefore, instead of passing checks on every searches of dense clique, we choose to add a restriction at the last moment of the process in order to avoid its impact on the growth of clique. Also, this design is out of consideration that a pseudo-dense clique will introduce errors only if its size is no less than the real dense clique. Otherwise, it will be rejected during the growth of clique. We define the inclusive clique based on which the clique size grows.

Definition 6.4. (Inclusive clique) We say that a clique C_{in} is inclusive if $\forall k \in \{2, \ldots, \text{card } C_{in} - 1\}$, it exists dense clique $\tilde{C}$, card $\tilde{C} = k$.

Finally, the clique which is supposed to contain background pixels of similar RGB values is defined as follows:

Definition 6.5. (Meaningful clique) Let $\sigma_T > 0$ be a given threshold. The meaningful clique C is a largest inclusive clique C_m with var $C_m \leq \sigma_T$, or a second largest inclusive clique of minimal variance.

In practice, we iterate the search of inclusive clique according to Definition 6.4 and we update each time the latest and previous results until only one inclusive clique is found. This clique will be output if its variance is below the threshold. Otherwise, it is suspicious to be a pseudo clique and the clique of smallest variance value among the previous results will be returned. The threshold in set to $\sigma_T = 15$ for the images valued in $\{0, \ldots, 255\}^3$, which is the empirical maximum RMSE of chromatic aligned results we've ever obtained in experiments (see Chapter 4). This process is described by Algorithm 9.

Algorithm 9: Meaningful clique computation (see Definition 6.5).

Data: Set $\mathbf{\Phi}(\mathbf{x})$ (see (6.1)), treshold σ_T.
Result: Meaningful clique $C(\mathbf{x})$.
Set $n :=$card $\mathbf{\Phi}(\mathbf{x})$, $m := 2$, $s := 0$, $S_{\text{pre}} := S_{\text{cur}} := \emptyset$
do
  Set $S_{\text{pre}} := S_{\text{cur}}$, $S_{\text{cur}} :=$ Algorithm 8 $(\mathbf{\Phi}(\mathbf{x}), m)$, $m := m + 1$ and $s :=$ card S_{cur}
while $s \geq 2$
Compute $\sigma^2 := \begin{cases} +\infty \text{ if } S_{\text{cur}} = \emptyset \\ \sigma^2 := \text{var } C, \text{ for } C \in S_{\text{cur}} \end{cases}$
if $\sigma^2 \leq \sigma_T^2$ **then**
  return $C \in S_{\text{cur}}$
else
  return $\arg\min_{C \in S_{\text{pre}}} \text{var } C$
end

Theoretically, if the minimum size of background clique is α, the number of images should be controlled under $3\alpha + 2$ (limit case with an α-member background clique and two $(\alpha + 1)$-member pseudo-dense cliques) so that there will be at most one pseudo-dense clique bigger than the background clique . Yet, in practice the appearance of double pseudo-dense cliques never occurred since we use at most 7 images which ensures a minimum size of background clique at 2. What's more, even the single pseudo-dense clique is rather rare during the search, needless to say the double case.

We consider that the meaningful clique contains pixels of the same view. Since it is almost the maximum inclusive clique, the highly repeated view is very likely

to be the background. To strengthen the robustness of result, we estimate $\tilde{\mathbf{I}}(\mathbf{x})$ given $\mathbf{\Phi}(\mathbf{x})$ by computing a median point of the meaningful clique $C(\mathbf{x})$ obtained with Algorithm 9:

$$\tilde{\mathbf{I}}(\mathbf{x}) \in \underset{\mathbf{I}(\mathbf{x})\in C(\mathbf{x})}{\arg\min} \sum_{\mathbf{I}_i(\mathbf{x})\in C(\mathbf{x})} \|\mathbf{I}_i(\mathbf{x}) - \mathbf{I}(\mathbf{x})\|_2^2$$

6.4 Experiments

We now turn to evaluate the performance of the proposed algorithm. Both simulated and real sequences are used to evaluate qualitatively and quantitatively our approach. Two other algorithms, the median based method and the robust PCA method [HFB14], are also estimated. In the following, we first give quantitative tests on simulated sequences and then exhibit the results on real image sequences. More experiments can be found in [YGMT18].

6.4.1 Simulated sequences

In our experiments, we benefit from the quantitative estimation on simulated sequences that eliminate the impact of alignment errors. An image sequence is generated by choosing an image as the ground-truth, and then superimposing randomly occlusions on the background. Also, we add white Gaussian noise to simulate the temporal images noise. In a such sequence, we denote, hereinafter, I_0 the ground-truth and $\tilde{I}$ the estimated result. Two examples are given by Figures 6.4 and 6.5.

Our idea is to compute the proportion of pixels conveying the mask content, namely the error rate. The estimations are carried out on different areas of the whole scene where every positions share the same number of background pixels out of all images in sequence. In all, we compute the error rates under a given number of correct sources.

To do so, the pixels where the background is observed $k \in \{1, \ldots, n\}$ times, in a sequence of n images, are given by the set

$$T(k) := \{\mathbf{x} \in \Omega : \varphi(\mathbf{x}) = k\}, \tag{6.2}$$

where $\varphi(\mathbf{x}) := \text{card } \{i \in \{1, \ldots, n\} : \|I_i(\mathbf{x}) - I_0(\mathbf{x})\|_2 < \varepsilon\}$ and $\varepsilon \geq 0$ is some noise dependent threshold. For $k \in \{1, \ldots, n\}$, the error rate is defined by:

$$R(k) := \frac{\text{card } \left\{ \left\| I_0(\mathbf{x}) - \tilde{I}(\mathbf{x}) \right\|_2 < \varepsilon \right\}}{\text{card } T(k)}. \tag{6.3}$$

Tables 6.1-6.2 give the error rates for four simulated sequences of five images in the noiseless and noisy cases. In these experiments, the clique method based on Algorithm 9 performs for most of the time better than two other methods. Especially when the background pixels account for less than 50% of pixels in the stacks (first and second column in the Tables 6.1-6.2). On the one hand, the median filtering becomes less reliable and errors is very likely to occur unless the background pixels happen to rank in the median positions. On the other hand, the clique method keeps its reliability when the background pixels form a clique of cardinality 2 or more.

		1	2	3	4	5
Seq.1	Clique	**91.1**	**3.2**	**0.0**	**0.0**	**0.0**
	Median	96.7	57.8	0.0	0.0	0.0
	RPCA	97.8	79.8	0.0	0.0	0.0
Seq.2	Clique		**26.0**	**0.0**	**0.0**	**0.0**
	Median		77.4	0.0	0.0	0.0
	RPCA		98.1	0.0	0.0	0.0
Seq.3	Clique		**0.0**	**2.1**	**0.0**	**0.0**
	Median		0.0	60.7	0.0	0.0
	RPCA		100	99.9	98.4	97.9
Seq.4	Clique		**0.0**	**0.0**	**0.0**	**0.0**
	Median		0.0	47.7	0.0	0.0
	RPCA		100	100	99.9	99.8

TABLE 6.1: Error rates for four noiseless simulations with clique, median and RPCA methods. The table provides the erroneous decision percentages when the background is observed k times. Each sequence has 5 images. Blue cells indicate that the error rates there are negligible. The median decision is correct if $k \geq 3$. The clique method performs better.

The robust PCA method shows poor performance although its results look very similar to the median images through a direct observation (see Figures 6.1 and 6.2). That is because the image contrast has been changed during the estimation of low-rank component and the distance $\left\| I_0(\mathbf{x}) - \tilde{I}(\mathbf{x}) \right\|_2$ turns out to be larger.

To explain this phenomenon, we recall here the main idea of robust PCA method. It supposes that the background information is conveyed by the low-rank component of intensity vectors which can be obtained by aligning original vectors along the direction of principal component [GVL12, CLMW11]. [SBZ16] provides a method to compute correctly the first principal component despite of

		1	2	3	4	5
Seq.1	Clique	**94.7**	**10.5**	8.4e-2	2.9e-3	**0.0**
	Median	97.4	61.8	**2.0e-2**	**0.0**	0.0
	RPCA	92.7	71.9	5.3e-2	4.5e-3	2.7e-3
Seq.2	Clique		**47.6**	8.5e-2	5.5e-3	**0.0**
	Median		80.3	1.7e-2	0.0	0.0
	RPCA		89.3	**0.0**	**0.0**	0.0
Seq.3	Clique		**0.0**	**6.7**	1.4e-2	**0.0**
	Median		0.0	63.3	**5.9e-3**	0.0
	RPCA		94.7	53.7	59.3	51.3
Seq.4	Clique	100	**1.0**	1.2e-2	1.1e-3	**0.0**
	Median	100	52.9	**0.0**	**0.0**	0.0
	RPCA	0.0	83.6	41.8	23.8	18.4

TABLE 6.2: Error rates in percentage. Noisy simulations with $\sigma = 5$ additive Gaussian noise. The table is organized as Table 6.1 ($\varepsilon := 35$). The clique method always performs better or very similarly.

the existence of outliers. The background reconstruction depends on a selection among the partial intensity after decomposition (the detailed method is presented in Chapter 2). Hence, the estimation on robust PCA method should be based on a synthesis of observation and qualitative analysis.

6.4.2 Real sequences

Figures 6.6 and 6.7 show two examples on real image sequences. In the areas where the background pixels account for more than half of the whole sequence, all methods are reliable. When this assumption breaks down, median filtering loose its robustness while robust PCA method seems to perform better sometimes. The proposed clique based method always works the best even for cases where there are only two background pixels to constitute minimum cliques.

To conclude, our new method that relies on the computation of meaningful clique is fast, simple and robust without assumption on the occlusion shapes, textures, colors or motions. It is demonstrated to be advantageous in terms of quality and reliability when comparing with a color median and with a much more sophisticated method, RPCA.

(a) Images in sequence(1st-5th image) and ground-truth(6th image)

(b) Median result (c) RPCA result (d) Clique result

FIGURE 6.4: **Simulated sequence with $\sigma = 5$ additive Gaussian noise.** Median filtering and robust PCA method give the similar results. Meaningful clique method removes most of the masks.

(a) Images in sequence(1st-5th image) and ground-truth(6th image)

(b) Median result (c) RPCA result (d) Clique result

FIGURE 6.5: **Noiseless simulated sequence.** In this example, the median filtering and robust PCA method give the similar results. The meaningful clique method performs the best with the least mask left in its result.

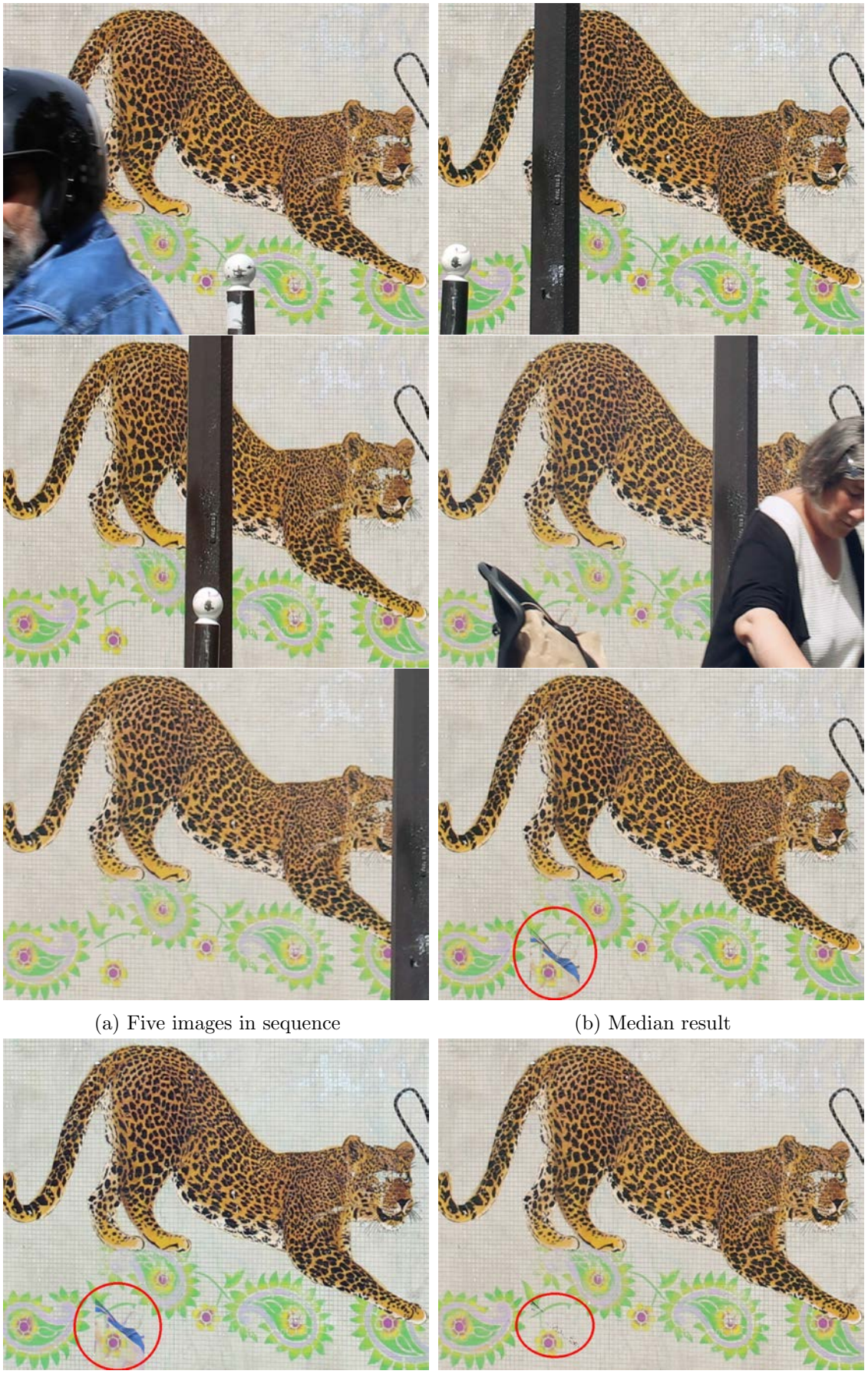
(a) Images in sequence

(b) Median result

(c) RPCA result

(d) Meaningful clique result

FIGURE 6.6: **Real sequence 1.** Median filtering losse its robustness in the areas where 2 pixels out of 4 belong to foreground. The robust PCA and meaningful clique method give both the results with all masks removed. They are more reliable when the assumption on background number breaks down.



(a) Five images in sequence

(b) Median result

(c) RPCA result

(d) Meaningful clique result

FIGURE 6.7: **Real sequence 2.** The meaningful clique method gives better result than median filtering and robust PCA method in this example.

Chapter 7

Jitter Blur Correction

During the conception of our process, we were at first dedicated to justifying the correctness of our model. The experiments were carried out with carefully selected lenses (SIGMA 30mm f1.5 DC HSM) in order to eliminate the effects such as distortion and chromatic aberration. As we moved to the next stage, we need to test the robustness of our method so we released the requirements on lenses. It turned out that the fusion results were no longer perfect with blur-pattern defects getting worse from the center to the border.

This phenomenon reminds us that the accuracy of steps presented in previous chapters may be affected by lenses quality. If so, certain parts of the algorithm should be modified in order to deal with broader conditions of use.

This chapter focuses particularly on this issue. It opens with a discussion on error sources followed by the search for possible solutions. In Section 7.1, experiments show that the errors come mainly from the lenses distortions which lead to misalignment between images. Section 7.2 reviews several existing tools either to correct the distortion or to replace the homography resampling method. Analysis shows that these methods are not appropriate to our case. Therefore, we propose in Section 7.3 an original method as an additional step of the whole process.

Remark that this method is an optional process aiming at reducing the jitter blur artifacts. The combination will not help eliminating the mask area in fusion results if pixels issued from the masks remain in the background after pixel selection step. So this approach is not an essential step when image sequences are well aligned.

7.1 Error sources

In this section, we present the defects observed in fusion results. Then, we analyze the possible sources of error and the steps they may affect. We focus here particularly on the types of problem caused by aberration. The existence of each type of error is inspected either by a direct observation or by a further test and we will come to a conclusion in the end of this section.

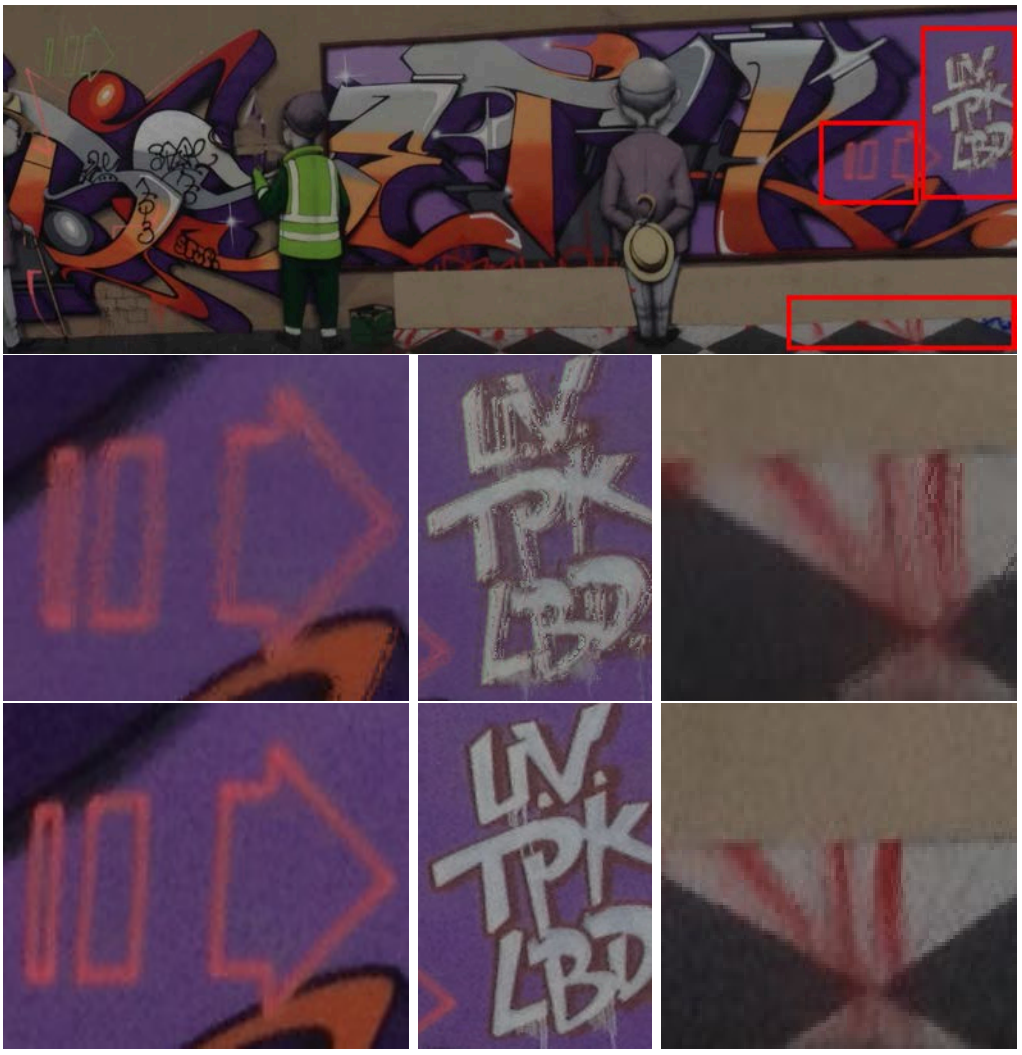


FIGURE 7.1: **Fusion result with defects and its partial enlarged details** The upper figure is a fusion result of a sequence of images captured by a zoom lenses. From left to right, it corresponds to the center till the left border of the original images. The middle figures show the enlarged detail of areas framed up in the upper figure. The lower figures gives a comparison issued from one of the aligned images in the original sequence.

7.1.1 Problem description

Figure 7.1 gives a good example of the defect pattern in fusion results. It shows that a pixel, which should be present only one time in the sharp image, may appear repeatedly in its neighborhood. In serious cases, the patterns look as blurry areas out of focus. What is more, the appearance of these defects concentrates mainly on the border of image. As to Figure 7.1, its left part, issuing from the center of original image, is almost free of defects, while its right part, which is also the right border of the original image, bears important defects everywhere.

7.1.2 Aberrations

Since the problem is brought about by the change of lenses, we focus on the defect which is named as optic aberrations. It explains the performance of real optical system apart from the ideally predicted model. Strictly speaking, every images bear the associated effects of aberration while the seriousness and the types of these effects varies from lenses. Basically, the aberration results in either distortion or blur (chromatically or mono-chromatically). Blur can be directly observed by an enlargement of image details while the distortion is difficult to be judged without a comparison with the corrected exemplar.

In case of blur, the accuracy of matched key-points of SIFT is no longer guaranteed. One of the possible problems is that the blur may worsen the redundancy of interest points. It is a character of SIFT that it tends to build local descriptors in a redundant way. A structure may be represented by several key-points in a slight change of positions or scales [RDGM10]. This phenomenon exists but is not serious in normal case. However, a blurry area may introduce many redundant points while the standard RANSAC algorithm does not have the mechanism to detect and discard them. As a result, both geometrical and chromatic alignment will be affected since they make use of the matched key-points to calculate the homography and the color transfer model.

In case of distortion, the transfer model between two images can not be expressed as a homography. Because of distortion, each object looses its rigidity in images. The original shapes will be modified in a way that straight lines become curves and figures deform increasingly from the center to the borders. No object will keep its form, not even to say the particular case of planar. If so, errors will occur in our process during the generation and the application of homography, which involves the RANSAC algorithm and the interpolation step of geometrical alignment.

7.1.3 Our judgment

The factor of blur is first excluded. Our operation ensures the image quality as we carried out the outdoor experiments under good lighting conditions with high shutter speed and we used tripod for indoor photography. Through a direct observation on enlarged details, images in most of the sequences remain sharp everywhere. Blur occurs occasionally in small areas of very few images. Therefore, even if the blur problem results in inaccuracy during alignment, it should not have a general effect on the fusion results of every sequences.

In contrast, distortion is more suspicious as it is particularly associated with zoom lenses (although it may also be found in prime lenses of high quality). Further more, the presence of defects in fusion results which is severer on the border than in the center is also in accordance with the characteristics of distortion. The jitter blur defects may probably be explained as the fusion results of several misaligned images that present the same content within a certain range of neighborhood.

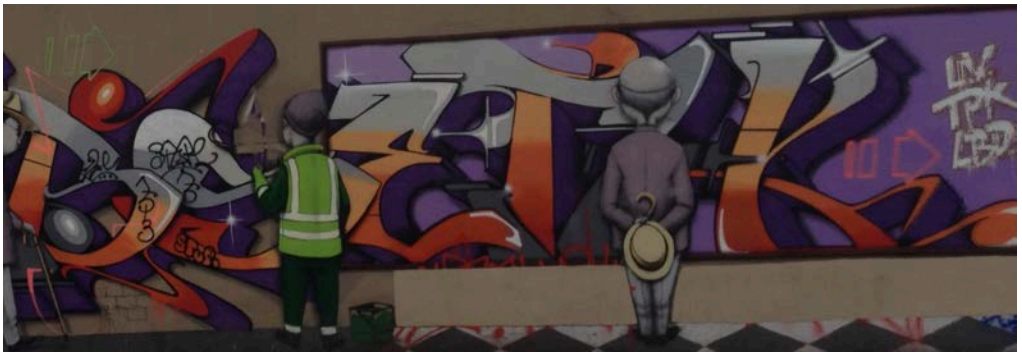
To justify our judgment, we derived the geometrically aligned images and counted their errors of alignment. As predicted, these errors varying from one pixel up to twenty pixels are rather important. The most significant errors appear in the sequences where the angle of view between two images changes a lot. That is consistent with the losing-rigidity prediction as these sequences require dramatic transformations that introduce more obvious errors than other cases.

Moreover, we tried to correct the distortion with a commercial image processing software DxO Optics Pro11 which takes into account the tabulated values of aberration for each pair of lenses and camera. The errors decrease but can not be totally removed. The fusion results become sharper but not good enough. (see Figure 7.2)

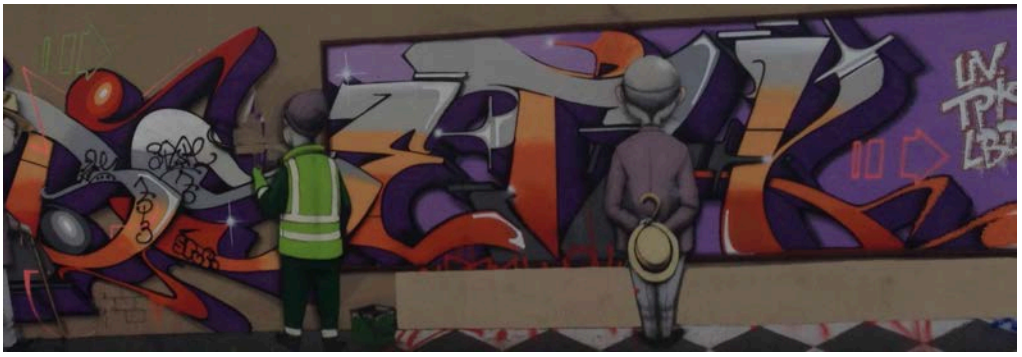
Here, we come to a conclusion that the major source of defects is the misalignment errors caused by the distortion of zoom lenses. It breaks down the hypothesis on rigid transformation model between images. Therefore, our following work will be dedicated to defect restoration.

7.2 Existing tools

Two applications are dedicated to alignment errors. Lens distortion correction may be added as the post-processing to deal with original images, while patch



(a) Median image of original sequence



(b) Median image of sequence after distortion correction

(c) Enlarged detail of 7.2a

(d) Enlarged detail of 7.2b

FIGURE 7.2: **Comparison between the median filtering results of the images with/without distortion correction.** Figure 7.2a, along with its enlarged detail Figure 7.2c, is the fusion result of the original sequence without additional process. Figures 7.2b and 7.2d show the median result of the same sequence while every images are geometrically corrected using the software DxO Optics Pro11 before being put into use. The maximum error among the aligned images reduces from 17 pixels to 6 pixels. The details of Figure 7.2d is clearer than Figure 7.2c but it is still not sharp enough.

match would be a substitution of geometrical alignment. However, they turn out to be either less effective or unfavorable to image fusion.

7.2.1 Lens distortion correction

The algorithms for lenses distortion correction have been studied for many years. The general idea is to map the distorted coordinates onto the corrected grids. Different models [Dua71, Len87, Fit01, CF05] have been raised ever since to deal with this problem, while the most popular expressions are those given by Fitzgibbon [Fit01]:

$$x = x_c + \frac{\tilde{x} - x_c}{1 + k_1 r^2 + k_2 r^4 + \dots},$$
$$y = y_c + \frac{\tilde{y} - y_c}{1 + k_1 r^2 + k_2 r^4 + \dots}.$$

Here $(\tilde{x}, \tilde{y})$ and (x, y) are respectively the distorted and corrected coordinates, (x_c, y_c) is the center of camera distortion model which is often represented by image center, $(k_1, k_2 \dots)$ are the distortion parameters.

The accuracy of parameter estimation has a direct impact on the effects of correction. Without camera information, most algorithms compute the parameters by minimizing the distortion error of string-like objects in image [DF01]. The selection of lines, either by manual identification [WQS09, AGS11] or by detection algorithms [CGPV03, BD13], gives very few results at disposal. Consequently, these methods tend to define less parameters in their model (2 for example), their work can not correct the slight distortion in real image.

In contrast, the calibrated systems such as DxO Optics Pro11 make use of the correction tables adapted to every pairs of camera and lenses. Their model benefits from more parameters and targeted description of photography equipments. We choose thus this software to reverse the effects of distortion. However, this measure is not enough to solve totally our problem. Just as is shown in Figure 7.2, the fusion result ameliorates but bears still some defects.

7.2.2 Patch match

Patch match is a randomized method for searching approximate nearest-neighbor matches between image patches in a very short time [BSFG09]. When we input a reference image I_{ref} and an image to be aligned I_{align}, the algorithm estimates

the position of patch in I_{align} which is most similar to the patch of I_{ref}. A map of patch coordinates, namely nearest-neighbor field [BSGF10], helps to edit a new image using the patches of I_{align} to replace the patches of I_{ref}. In other words, we apply I_{align} as the source of patch to recreate I_{ref}. If we replace the homography transformation by patch match, there will be no alignment errors between the aligned images.

Even so, this approach is not suitable for our application. In the mask region of reference image, the aligned images will express either the same mask (if they have found the similar source patches) or a random pattern. Both cases, losing the truth of background, will lead to mistakes in fusion result.

7.3 Combination method

Since the existing methods can not satisfy our requirements, we turn to design our own approach. Remark that the ideal geometrical theory is too delicate to bear the associated effects of photography, we abandon directly the idea of improving alignment process. Instead, we try to add an additional step to correct the fusion results. This section goes through details of our basic idea, the method pipeline as well as the adjustment of parameters.

7.3.1 General description

Under the condition of perfect alignment, the fusion result can be explained as the restoration of one of the images in a sequence by removing its masks. Therefore, we come up with an idea based on the mixture of the fusion result and a selected image after alignment.

In fact, such two images have their qualities and imperfection which are just complementary. The selected image presents perfectly most parts of the background except those blocked by the occlusions. On the contrary, the fusion result bears defects caused by the alignment errors while it reveals the content behind the masks. Our attempt takes advantage of both images by covering the content of fusion result onto the masks of the selected image.

Still, there are two problems to be solved during the conception. First, the aligned image must be selected rather than be picked randomly due to the existence of alignment errors. What is more, the comparison between two images should take into account the defects in fusion result.

The process consists of three steps. First, we select an image that will be later combined with the fusion result. Next, a difference map is set up to indicate the mask areas in the chosen image. In the end, we redefine an associated result of these two images according to the map. In the following, the selected image and the fusion result of either median or clique method, mapped on the same grid Ω, will be denoted with $\mathbf{I}_s(\mathbf{x})$, $\mathbf{I}_f(\mathbf{x}) : \Omega \to \{0, \ldots, 255\}^3$.

Before going into details, we declare the limitations of our design even though they are extreme cases that rarely occur in our experiments:

a) Tiny masks such as spots or lines will be re-added into the result.
b) Defects located in the masks area will not be removed.

7.3.2 Pipeline

The procedures involve the choice of sharp image with masks, estimation of blurry area and region replacement. Here is a detailed description and analysis of the proposed approach.

Image selection

Strictly speaking, it is not an independent step in the program, while the choice of image should be logically finished before all other process in the pipeline. The selected image is required to have the least alignment difference from the fusion result in order to avoid combining errors in final result.

In practice, the reference image, which is used as the template to align other images in Chapter 3 and Chapter 4, holds the post of $\mathbf{I}_s$. A justification is that the reference image can be considered as the average form of view in a sequence if we regard the alignment errors as a stretch of view towards a random direction. Its pixels are more often to be chosen during the fusion process and it turns out to be the best choice for a combination with the fusion result.

Difference map

The difference map is a binary image of size Ω that distinguish the points of masks from those of the background. Ideally, it can be obtained by filtering the intensity difference between $\mathbf{I}_s(\mathbf{x})$ and $\mathbf{I}_f(\mathbf{x})$ with a threshold. The area with a distance exceeding the threshold is recorded as the mask in the map. However, this simple procedure is not robust to the effects of alignment errors. Some differences between two images, as for instance the misaligned contours, will be classified as the masks.

Hence, it is necessary to somehow modify the images before calculating their intensity distances. We have noticed that the pattern of defects looks like a fitful repetition of the adjacent texture. In extreme cases, it is similar with the blurry parts of image. These blur-like defects, along with the slight misalignment of contours, can be covered up in images of low resolution. So our strategy is:

a) to degrade resolution of $\mathbf{I}_s$ and $\mathbf{I}_f$ by low pass filtering;
b) to filter the intensity distances between low resolution images;
c) to create a difference map from the filtered results.

In order to get low resolution images, we convolve the original images with Gaussian filter. This method is widely used to reduce the details while enhancing the structure of images. The visual effect of this process is to produce images of the same view acquired at furthest distance.

The filtering proceeds as a weight matrix namely Gaussian kernel passes through the original image. The new value of each pixel is a weighted average of its neighborhood. In practice, we use the discrete approximation of Gaussian function (of the normal distribution) to build the Gaussian kernel. The weight values of the kernel are calculated by:

$$G_\sigma(x_d, y_d) = \frac{1}{2\pi\sigma^2} \exp(-\frac{x_d^2 + y_d^2}{2\sigma^2}),$$

where x_d and y_d are the horizontal and vertical pixel distances to the kernel center, σ is the standard deviation of the Gaussian distribution. A larger σ results in a flatter Gaussian distribution and further a more important blur. We are thus allowed to adjust the blur level of the convolution results.

Definition 7.1. A blurred image $B_\sigma : \Omega \to \{0, \ldots, 255\}^3$ of its original image $\mathbf{I} : \Omega \to \{0, \ldots, 255\}^3$ is the convolution result of $\mathbf{I}$ and the Gaussian kernel G_σ:

$$\forall \mathbf{x} \in \Omega, \quad B_\sigma(\mathbf{x}) := (G_\sigma * \mathbf{I})(\mathbf{x}).$$

Having eliminated the influence of contours and defects, we calculate now the intensity distance between blurred images $B_{\sigma,s}$ and $B_{\sigma,f}$ (see Definition 7.1). With a carefully chosen σ, the area with great difference reveals the mask in selected image and in fusion image (if there is any left). Since the images are no longer sharp, these areas will be a little bit larger than the mask themselves. Yet, we appreciate this imprecision because it ensures the total removal of masks latter in the new defined result.

We estimate the difference of each pixel with a threshold $\epsilon > 0$. The difference map is defined by Definition 7.2.

Definition 7.2. Given some $\epsilon > 0$, the difference map $M_\epsilon : \Omega \to \{0, 1\}$ of two blurred images $B_{\sigma,s}$ and $B_{\sigma,f}$ is a binary map whose value is defined by:

$$\forall \mathbf{x} \in \Omega, \quad M_\epsilon(\mathbf{x}) := \begin{cases} 0 & if \ \|B_{\sigma,s}(\mathbf{x}) - B_{\sigma,f}(\mathbf{x})\|_2 < \epsilon \\ 1 & otherwise. \end{cases}$$

Image generation

We generate in this step the new image without defects. Following the previous step, the area with $M_\epsilon = 0$ conveys the identical views of $\mathbf{I}_s$ and $\mathbf{I}_f$ despite of the misalignment errors and the defects. Here, $\mathbf{I}_s$ benefits from its spatial continuity while $\mathbf{I}_f$ risks being imperfect as it consists of several images. So we apply the pixel values of $\mathbf{I}_s$ to define this area in the new image. On the contrary, $\mathbf{I}_f$ explains better than $\mathbf{I}_s$ the content behind masks. It is thus selected to complete the area $M_\epsilon = 1$ of new image. Over all, the defect corrected image is created by the following definition:

Definition 7.3. The corrected image $\mathbf{I}_c$ of $\mathbf{I}_f$ is a combination of $\mathbf{I}_f$ and its reference image $\mathbf{I}_s$ based on their difference map M_ϵ.

$$\forall \mathbf{x} \in \Omega, \quad \mathbf{I}_c(\mathbf{x}) := \begin{cases} \mathbf{I}_s(\mathbf{x}) & if \quad M_\epsilon(\mathbf{x}) = 0 \\ \mathbf{I}_f(\mathbf{x}) & if \quad M_\epsilon(\mathbf{x}) = 1. \end{cases}$$

It should be noted that we express the masks as the difference between $\mathbf{I}_s$ and $\mathbf{I}_f$ marked with $M_\epsilon = 1$. This expression is generally correct except when there are masks left in $\mathbf{I}_f$. These masks, marked with either $M_\epsilon = 0$ (if they share the view of $\mathbf{I}_s$) or $M_\epsilon = 1$ (if they are issued from other images), will be remained in $\mathbf{I}_c$. That explains why we declare from the very beginning that our method helps removing misalignment defects (either spatially or photometrically) rather than improving mask removal efficiency.

7.3.3 Parameter selection

The standard deviation σ of Gaussian filter and the threshold ϵ of intensity distance are two parameters to be decided during the process. At present, we adjust manually these parameters according to the severity of misalignment in

different sequences. Nevertheless, it is necessary to start with reasonable values and to predict the effects of their variation.

In a discrete application of Gaussian filter (command 'imgaussfilt' in matlab), the standard deviation decides not only the pattern of distribution but also the size of kernel window. As for our case, the filter takes into account 95.45% probability as near certainty with a window $[-2\sigma, 2\sigma]^2$, centered at $\mathbf{x} \in \Omega$. It is reasonable to set up a relationship between σ and the maximum misalignment error δ_a as δ_a is related to the level of jitter blur defects. Within the error range, pixels conveying repetitively the same view should be distributed large weights during the blurry simulation. For a conservative estimate, we assume that pixels, within the range of $[-3\delta_a, 3\delta_a]^2$ centered at $\mathbf{x} \in \Omega$, are included in the window and the start value of standard deviation is assigned to $\sigma = 1.5\,\delta_a$. As we try to retain the accuracy of our estimation, the standard deviation is not desired to be of great value. It will be slightly raised only if the blurred image is too sensitive to the change of intensity threshold.

The start value of ϵ is chosen according to the statistic analysis on RMSE of identical patches after chromatic alignment. It is often set to 15 which is the maximum value we have got in all of our experiments. During the adjustment, the threshold value varies only in a limited range after the Gaussian filter has been chosen. On one hand, a small value is so sensitive that the defects and misalignment errors may be classified as the masks. On the other, a big value risks omitting some parts of the masks.

To get the optimal combination of two parameters, our strategy is to initialize them at $(\sigma, \epsilon) = (1.5\,\delta_a, 15)$. With σ fixed, we adjust ϵ to get a better result: if it exists defects of misalignment problem, ϵ should be increased; if masks of $\mathbf{I}_s$ remains in the result, ϵ should be reduced. When a slight change of ϵ leads to a mutuality between the problem of defects and masks, it means the dynamic range of ϵ is not enough and σ should be increased.

7.4 Experiments

We exploited the idea of Section 7.3 into two approaches. The first approach is to use the pipeline directly by combining a selected image with the fusion result. Further more, we modified the program in an iterative way in order that most area of the final image (even the masks) benefits from spatial continuity. The results of these methods are presented in this section.

7.4.1 Attempt on the pipeline

The key to success of our method is its ability to distinguish defects from masks. If it works well, the defects should be removed while no more masks other than those left in fusion results will be added into the output image.

Algorithm 10: Combination method

Data: $\mathbf{I}_s$, $\mathbf{I}_f$, σ, ϵ
Result: Corrected image $\mathbf{I}_c$
Calculate blurred image $B_{\sigma,s}$ of $\mathbf{I}_s$ and $B_{\sigma,f}$ of $\mathbf{I}_f$ (see Definition 7.1);
Generate difference map M_ϵ with $B_{\sigma,s}$ and $B_{\sigma,f}$ (see Definition 7.2);
Define an image $\mathbf{I}_c$ by combining $\mathbf{I}_s$ and $\mathbf{I}_f$ according to M_ϵ (see Definition 7.3);
return $\mathbf{I}_c$

The pipeline in Section 7.3.2 can be expressed by Algorithm 10. We want to make sure that this method can be implanted as a universal step of our whole work. Therefore, it is required to be able to deal with every types of sequence in our tests. These sequences are generally divided into four categories according to the imperfections in their fusion results.

	Type 1	Type 2	Type 3	Type 4
Defects	×	×	√	√
Mask	×	√	×	√

TABLE 7.1: **Imperfection types in fusion results.** There are respectively two criteria: with/without defects; with/without masks.

As is shown in Table 7.1, the imperfection type depends on the appearance of defects and masks in the fusion results. In our tests, the defects are mainly caused by the distortion introduced by the zoom lenses (see Section 7.1). So the option 'with' and 'without' defects corresponds to the sequences captured by zoom lenses or by the nice prime lenses. While the remain of masks is related to the density of masks in original sequence which has been discussed in Chapter 5 and Chapter 6. Thus, we classify our samples as:

1. captured by prime lenses + follow the hypotheses for pixel selection;
2. captured by prime lenses + break the hypotheses for pixel selection;
3. captured by zoom lenses + follow the hypotheses for pixel selection;
4. captured by zoom lenses + break the hypotheses for pixel selection.

Figures 7.3 - 7.6 give respectively one example for each sequence type. In these examples, we choose the median images as fusion results and we select the reference images of each sequence to replace the areas with defects. These choices are

reasonable as no significant misalignment problem is found in the areas where two images join together.

In terms of defects, Figures 7.3 and 7.4 show that our method will not add extra problems onto the fusion results when the alignments are accurate. While Figures 7.5 and 7.6 demonstrate the efficiency of our method in detecting and removing defects since the jitter blur parts located in the mask-free areas of selected images are replaced by the sharp view.

In terms of masks, Figure 7.4b (resp. Figure 7.6b) conveys visually the same occlusion as Figure 7.4a (resp. Figure 7.6a). No additional mask issued from the selected image is introduced into the final result.

What is more, the parameter setting discussed in Section 7.3.3 shows its convenience in the tests. Table 7.2 lists the alignment errors of Figures 7.3-7.6 and the parameters we set during the experiments. The applied values show no big differences to their start values and the adjustment is always unidirectional with σ increasing and with ϵ decreasing.

	Alignment error	Start values	Applied values
Figure 7.3	2px	$(3, 15)$	$(3, 13)$
Figure 7.4	1px	$(1.5, 15)$	$(2, 15)$
Figure 7.5	17px	$(25.5, 15)$	$(27, 15)$
Figure 7.6	13px	$(19.5, 15)$	$(21, 11)$

TABLE 7.2: **Parameter pairs (σ, ϵ) in Figures 7.3-7.6.** Alignment error δ_a indicates the maximum error between two images in the sequence. It is obtained based on a cursory observation. Start values are (σ, ϵ) mentioned in Section 7.3.3 which are set by $\sigma = 1.5\delta_a$ and $\epsilon = 15$. Applied values are the pairs of (σ, ϵ) deriving the best results in our experiments.

In all, the test results prove the efficiency and robustness of our method when dealing with defects. The new created images benefit from the sharp view of the selected image in most of the areas. Still, not all the defects are removed in this approach. It is limited by the lack of information behind masks in the selected image. Therefore, a further improvement focuses on a larger range of replacement of fusion result by the original images which will be presented in the next section.

(a) Original fusion result

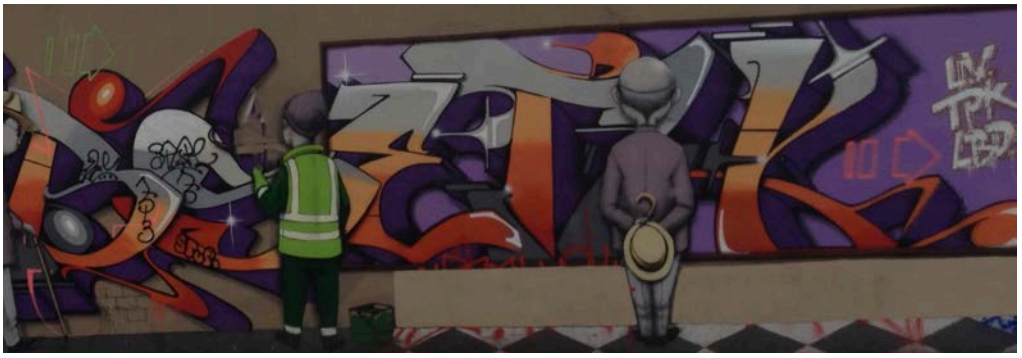
(b) Corrected image

FIGURE 7.3: **Defect-correction**
Sequence of imperfection type 1: no defects, no masks.

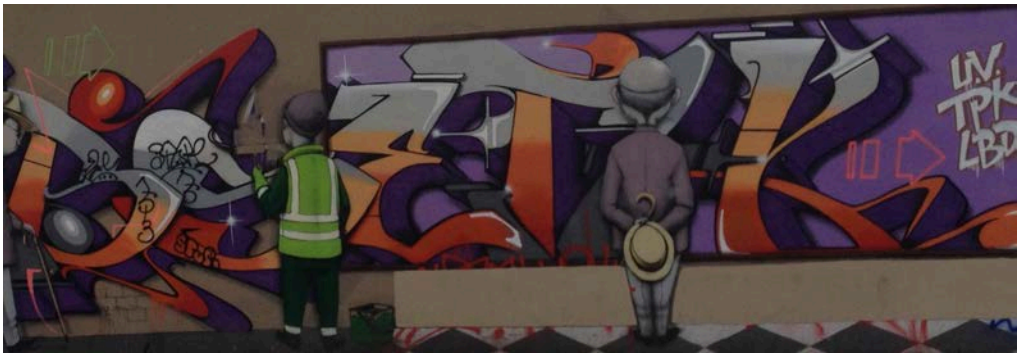
(a) Original fusion result

(b) Corrected image

FIGURE 7.4: **Defect-correction**
Sequence of imperfection type 2: no defects, with masks.



(a) Original fusion result



(b) Corrected image

(c) Enlarged detail of 7.5a

(d) Enlarged detail of 7.5b

FIGURE 7.5: **Defect-correction**
Sequence of imperfection type 3: with defects, no masks.

(a) Original fusion result

(b) Corrected image

(c) Enlarged detail of 7.6a

(d) Enlarged detail of 7.6b

FIGURE 7.6: **Defect-correction**
Sequence of imperfection type 4: with defects, with masks.

7.4.2 Attempt on the iteration of pipeline

Figure 7.7 and Figure 7.8 help to explain why we make a further attempt on the iteration of our method. The red circles in Figure 7.7a and Figure 7.8a lighten the areas where the defects left in the results of the no-iteration process. They are located in the mask areas of the selected images whose content is filled in by the fusion images. Consequently, the defects in these regions remain after the process as is shown in Figure 7.7b and Figure 7.8b.

In order to remove these defects, it requires further combinations to other original images conveying the background view. That inspires us to modify Algorithm 10 in an iterative way so that the fusion results can be replaced as much as possible by others images in the sequence.

Algorithm 11: Iterative combination method

Data: $\{\mathbf{I}_i\}\ i \in \{1, \ldots, n\},\ \mathbf{I}_f,\ \sigma,\ \epsilon$
Result: Corrected image $\mathbf{I}_c$
Initialize $S = \{\mathbf{I}_i\}$, $Z = \mathbf{1}_\Omega$, $\mathbf{I}_c = \mathbf{I}_f$;
while Z *is not equal to* $\mathbf{0}_\Omega$ **do**
  Set $N_{total} = 0$, $index = 0$, $M_\epsilon = \mathbf{0}_\Omega$;
  for $i = 1$ **to** n **do**
    `/* Select an image for combination */`
    Generate $M_{\epsilon,i}$ of $S\{i\}$ and $\mathbf{I}_f$ according to Definitions 7.1 and 7.2;
    Define combining area $M_{combine,i}(\mathbf{x}) := \begin{cases} 0 & M_{\epsilon,i}(\mathbf{x}) = 0 \,\&\&\, Z(\mathbf{x}) = 1 \\ 1 & else \end{cases}$
    if $N_{total} < card\,\{M_{combine,i}(\mathbf{x}) = 0\}$ **then**
      $N_{total} = card\,\{M_{combine,i}(\mathbf{x}) = 0\}$, $index = i$, $M_\epsilon = M_{combine,i}$;
    end
  end
  `/* Combine the selected image to the present result */`
  Define new $\mathbf{I}_c$ with present $\mathbf{I}_c$ and $S\{index\}$based on M_ϵ (see Definition 7.3);
  `/* Update the image set and the area left */`
  Update $Z(\mathbf{x}) := \begin{cases} 0 & M_\epsilon = 0 \\ Z(\mathbf{x}) & else \end{cases}$ and $S = S \setminus S\{index\}$;
  if $S = \emptyset$ **then**
    return $\mathbf{I}_c$
  end
end
return $\mathbf{I}_c$

Algorithm 11 shows the detail of the iterative method. The basic pipeline and the choice of parameters remain the same while the slight change lies in the selection of image to be joined. Instead of deciding an image on purpose, we estimate

the common area between the aligned sequence and the fusion region remained after the previous loop. We choose the image sharing the largest common area with the fusion region for the next combination. It is reasonable as decision because it allows us to use less images so as to reduce the mismatch risks along the combining boundary.

The experimental results turns out to be satisfying. Generally speaking, the iterative method gives similar results as the no-iterative method in most of the area. What is more, its outputs present perfectly the details everywhere despite of the masks in certain original images (see comparison between Figures 7.7b and 7.7c, between Figures 7.8b and 7.8c). That makes it a better choice than the method without loop.

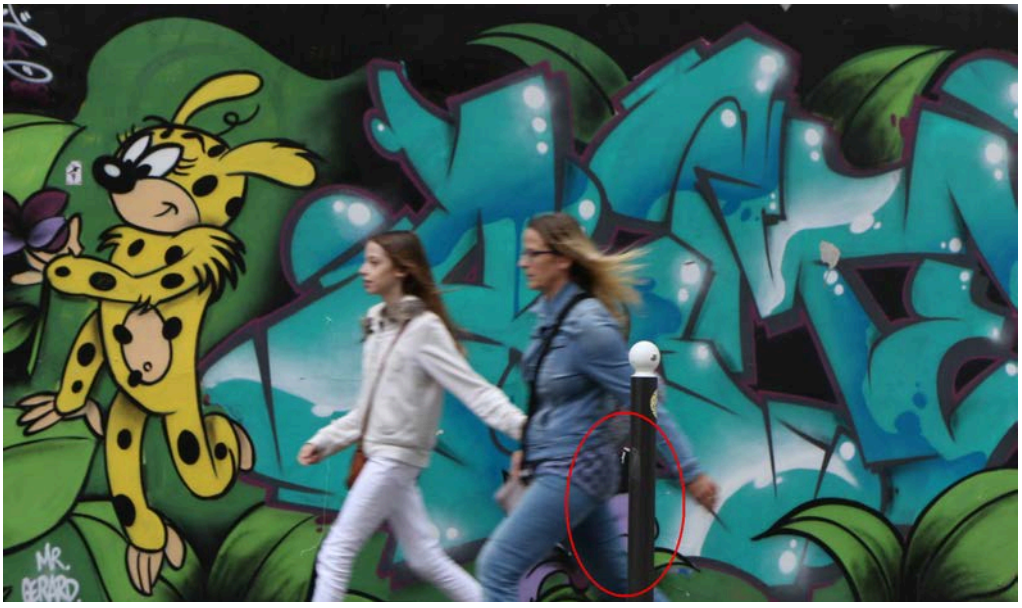
Up to now, we achieve our goal that the region, apart from the remaining masks in the fusion result, is replaced by the spatial continuous view of the original aligned images. The defects caused by the geometrical alignment errors has been removed by our approach. Since the method is shown to be robust in many cases, it is therefore added as a general step of our whole project.

(a) Location of the enlarged detail in the selected image

(b) Detail without iteration

(c) Detail after iteration

FIGURE 7.7: **Comparison between outputs of Algorithms 10 and 11**



(a) Location of the enlarged detail in the selected image

(b) Detail without iteration

(c) Detail after iteration

FIGURE 7.8: **Comparison between outputs of Algorithms 10 and 11.** Without iteration, obvious defects can be seen on the purple and green areas of result which are blocked by the mask in original image. The iterative approach provides a better result with clearer details.

Part III

Experiments

Chapter 8

Experimental Performances

In this chapter, we provide extensive experiments. They serve as a more in depth evaluation on methods compared with the experiments at the end of each previous chapter, which help basically to explain how we decide steps in algorithms. We set up here our own data set and give a general estimation on the performances of each step in the pipeline.

We start by presenting the data sets, then we go into details on experiments. The objective is to evaluate the robustness of approaches that have been presented in the previous chapters under different conditions. This chapter will end up with a summary of our design.

8.1 Data set

Geometrical alignment errors and contrast differences are very important under practical field conditions and the whole pipeline is deployed with the aim of eliminating their effects. Data selection should therefore account for these variations.

In order to fully test our algorithm, the data set should contain sequences of various qualities. According to the discussion in previous chapters, geometrical alignment errors and contrast differences are related respectively to the optical distortion of lenses and the color adjustment inside cameras. Hence, we choose three combinations of equipment in our experiments, respectively producing images of high, intermediary or poor quality.

Sequence	**Camera**	**Lenses Model**	**Parameters**			
			Focal	Aperture	Speed	ISO
Mario	Canon EOS 80D	Sigma 30mm F1.4 DC HSM	30mm	F11.0	1/80s	160
Dance	Canon EOS 80D	Sigma 30mm F1.4 DC HSM	30mm	F10.0	1/160s	100
Bakery	Canon EOS 80D	Sigma 30mm F1.4 DC HSM	30mm	F6.3	1/100s	160
Leopard	Canon EOS 80D	Sigma 30mm F1.4 DC HSM	30mm	F10.0	1/125s	100
Woman	Canon EOS 80D	Sigma 30mm F1.4 DC HSM	30mm	F10.0	1/100s	100
Monster	Canon EOS 80D	Canon EF S17-55mm F2.8 IS USM	17mm	F8.0	1/125s	640
Cartoon	Canon EOS 7D Mark II	Canon EF S17-55mm F2.8 IS USM	37mm	F6.3	1/80s	400
Inversion	Canon EOS 7D Mark II	Canon EF S17-50mm F2.8 EX DC OS HSM	21mm	F7.1	1/125s	250
Orient	Canon EOS 7D Mark II	Canon EF S17-50mm F2.8 EX DC OS HSM	35mm	F9.0	1/200s	100
Snoopy	Canon EOS 7D Mark II	Canon EF S17-50mm F2.8 EX DC OS HSM	38mm	F7.1	1/100s	100
Babs	IPhone 4S	IPhone 4S Lens	4.3mm	F2.4	Auto	Auto
Garfield	IPhone 4S	IPhone 4S Lens	4.3mm	F2.4	Auto	Auto
Banc	IPhone 4S	IPhone 4S Lens	4.3mm	F2.4	Auto	Auto
Mexican	IPhone 4S	IPhone 4S Lens	4.3mm	F2.4	Auto	Auto
Planet	IPhone 4S	IPhone 4S Lens	4.3mm	F2.4	Auto	Auto
Syn 1	–	–	–	–	–	–
Syn 2	–	–	–	–	–	–
Syn 3	–	–	–	–	–	–
Syn 4	–	–	–	–	–	–
Syn 5	–	–	–	–	–	–

TABLE 8.1: **Photographic information.** There are in total twenty sequences in our data set. Each sequence consists of 3, 4 or 5 images. They are divided into four groups according to the photographic equipments. The first five sequences are captured by a prime lens with every parameters fixed. The second five sequences are obtained using a zoom lenses of poor optical quality. The third group of sources are smart phone images whose contrasts are automatically adjusted. The last five series are synthetic sequences with background covered randomly by image patches as masks. Gaussian noise is then added onto these images.

Images with less errors are taken by good lenses of fixed focal length. (We call it hereinafter the prime lenses.) In our tests, we choose *Sigma 30mm F1.4 DC HSM* combined with a *Canon EOS 80D* camera to take a group of high-quality sequences. This combination, along with manually fixed parameters, is considered to accurately record the scene with little geometric or chromatic distortions.

The second choice is to employ lenses whose focal lengths are adjustable (zoom lenses). Images captured by these lenses may not be as good as those of the first group since zoom of lenses may introduce distortion problems. The lenses we use in experiments are *Canon EF S17-55mm F2.8 IS USM* and *Canon EF S17-50mm F2.8 EX DC OS HSM*.

The most uncontrollable case comes from the use of embodied lenses of smart phone with automatic adjustment on focusing and lighting conditions. Serious contrast differences and distortion problems may be introduced to images which challenge greatly the performance of our algorithm. We select five sequences issued from *IPhone 4S* that are thought to be of limited quality.

In addition, we create five synthetic sequences for quantitative evaluation of fusion methods in order to avoid the effects of alignment errors on the results. Each image is made of a background scene covered randomly by occlusions with Gaussian noise ($\sigma = 5$) independently distributed in RGB channels. In all, Table 8.1 shows the details of our data source.

Besides, *DxO Optics Pro11* is supposed to have a positive effect on the correction of distortion (see Chapter 7). It is also instructive to be aware of the performance of our algorithm when dealing with images exported from such image processing software. Therefore, each real sequence provides a new sequence with pre-processing of DxO geometric correction. As a whole, our data set consists of 6 kinds of real sequences and the synthetic sequences.

	Prime Lenses	**Zoom Lenses**	**Smart Phone**	**Synthetic**
DxO correction	✓/✗	✓/✗	✓/✗	✗

TABLE 8.2: **Seven kinds of sequences in data set.** Sequences captured by prime lenses, zoom lenses and smart phone will be pre-proceeded by *DxO Optics Pro11* to eliminate the geometric distortion. Along with three kinds of original groups and the synthetic series, they are taken as new kinds of sequences in our tests.

8.2 Summary of parameters

Table 8.3 summarizes all the parameters to be adjusted during the process and their values used in our experiments. These values are decided based on either their experimental performances (i_{iter}, ϵ_g, ϵ_c, σ), or on the evaluation of results in previous steps (σ_T, ϵ).

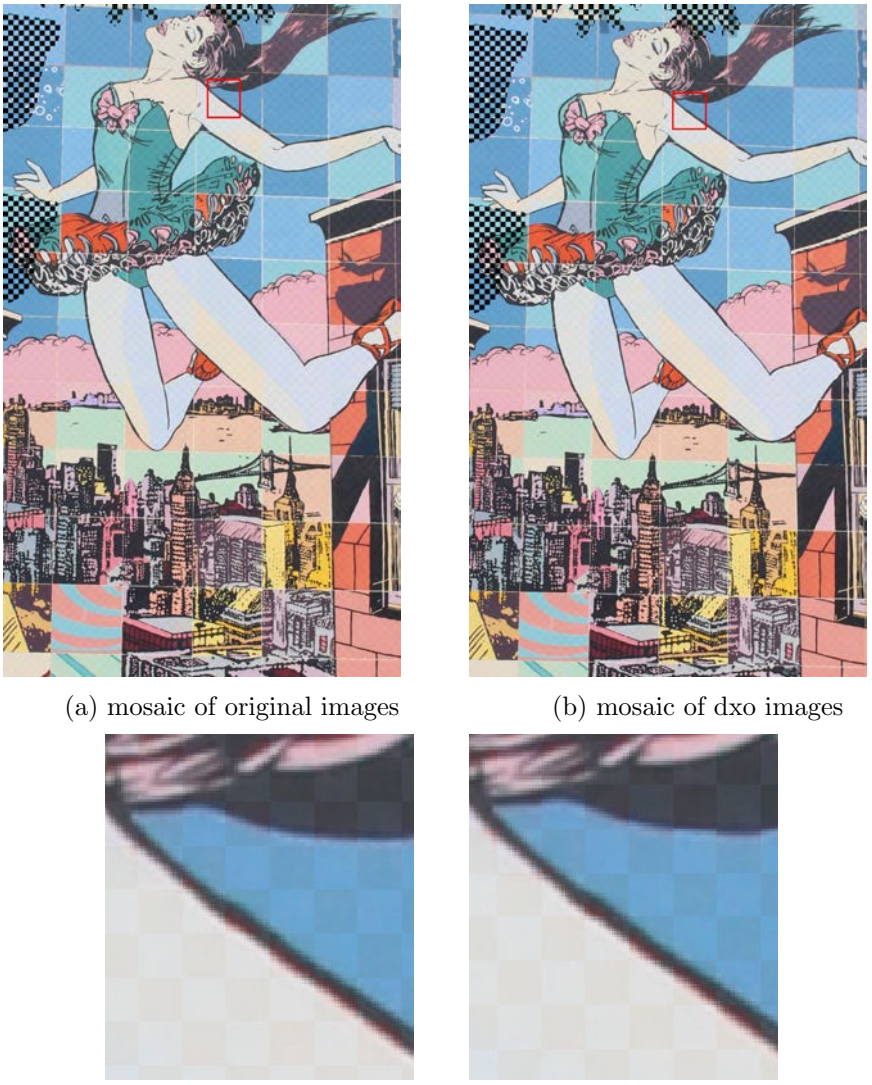
Geometric Alignment	relative threshold between nearest and second nearest neighbors in SIFT matching $t_{match} = 15$; RANSAC number of iteration: $i_{iter} = 10000$; maximum residual geometric error: $\epsilon_g = 1$.
Chromatic Adjustment	maximum residual chromatic error per channel: $\epsilon_c = 0.01$.
Meaningful Clique	maximum variance of vectors in clique: $\sigma_T = 15$.
Combination Method	standard deviation of the Gaussian distribution: $\sigma = 1.5\epsilon_a$; (ϵ_a is the size of blur-like defects.) maximum distance of similar blurred vector: $\epsilon = 15$.

TABLE 8.3: **Parameter values used during tests.** The explication of parameter choices are found in Chapter 3 Section 3.3 for t_{match} ,i_{iter}, ϵ_g; in Chapter 4 Section 4.4.1 for ϵ_c; in Chapter 6 Section 6.3.2 for σ_T; in Chapter 7 Section 7.3.3 for σ, ϵ.

8.3 Geometrical alignment

To visually examine the geometric alignment errors, we make mosaics of aligned images in sequences. They consist of patches issued from two images which are alternately arranged. If errors exist among aligned images, there will be twisted shapes in mosaic results. We test on real sequences, with or without DxO pre-processing, captured by prime lenses, zoom lenses and lenses of smart phone (six types of real sequences in Table 8.2).

Results show that the degree of distortion varies from lenses types as we have predicted. It exists a big difference of alignment errors based on the photography equipments. DxO pre-processing plays more or less a positive role to correct distortion while it is not enough to deal with this problem independently for the sake of an accurate pixel selection later in fusion step.



(a) mosaic of original images (b) mosaic of dxo images

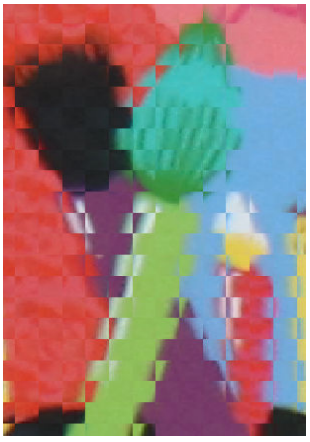
(c) Zoom of original mosaic (d) Zoom of dxo mosaic

FIGURE 8.1: **Mosaic - Prime lenses sequence (Dance).** The mosaic of original images has no obvious geometric distortion. Lines of patches connect smoothly even in the partial enlarged area marked by red square. It exists no much difference with/without DxO pre-processing.

Images captured by prime lenses are of good quality. Without any pre-processing, their aligned results superpose already perfectly one to another. Tests show similar results as what is presented in Chapter 3. We observed that most of the errors in original sequences are below one pixel. They are acceptable for use in further steps. The mosaic images, as is shown in the example of Figure 8.1, have no obvious distortion. DxO correction makes no big difference in terms of homography accuracy and its use here is therefore optional.

(a) mosaic of original images

(b) mosaic of dxo images



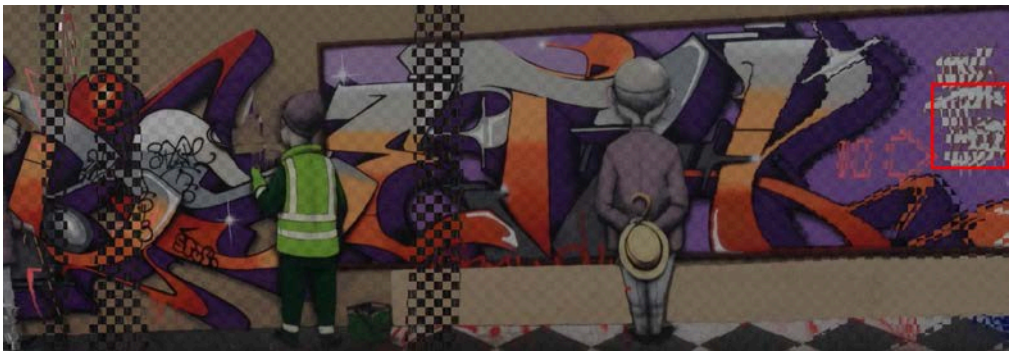
(c) Zoom of original mosaic

(d) Zoom of dxo mosaic

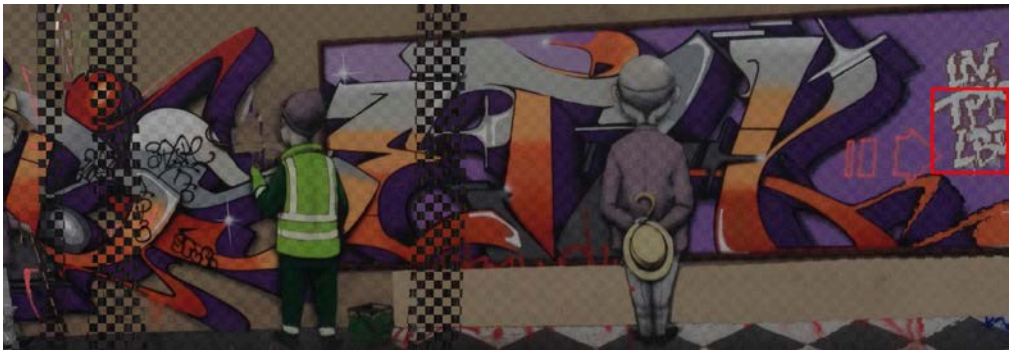
FIGURE 8.2: **Mosaic - Zoom lenses sequence (Monster).** Distortion becomes more and more serious from the center to the border of images. DxO corrected result is much better than original one while small errors exist still.

Zoom lenses introduce more geometric distortion to sequences. In Figure 8.2, mosaic in the center presents well the connection of texture in neighboring area. However, distortion is detected on the borders as dislocation of patches appears in the mosaic of two images. The DxO pre-processing corrects efficiently the errors as can be seen in Figure 8.2b, when compared with Figure 8.2a. Still,

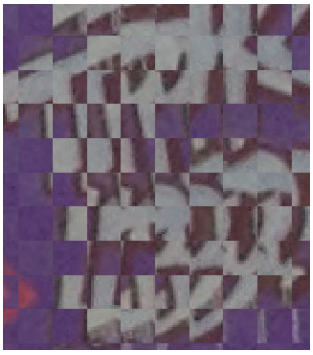
small errors of three to four pixels remain among corrected images. These errors may result in tiny blur-like profile of objects in fusion results.



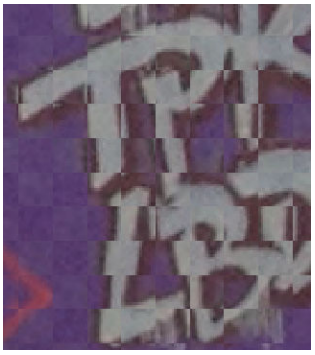
(a) mosaic of original images



(b) mosaic of dxo images



(c) Zoom of original mosaic



(d) Zoom of dxo mosaic

FIGURE 8.3: **Mosaic - Smart phone sequence (Babs).** Distortion exists in most range of the image. Serious errors result in a repeated expression of scene in close proximity. Letters in the enlarged Figure 8.3c cannot be clearly distinguished. DxO corrected result presents better the scene while errors are still evident on the borders. (see Figure 8.3d)

Distortion is so severe in sequences captured by smart phone that some alignment errors reach more than fifteen pixels and its effects spread even to the quasi-center range of image (see example in Figure 8.3). In this case, the DxO pre-processing provides a great improvement on these images which restore the

correct presentation of patches in their center area. However, it does not manage to compensate the important deformation on the borders where dislocation of patches is still obvious as is shown by Figures 8.3b and 8.3d.

8.4 Photometric correction

We estimate our chromatic adjustment methods in this section. In Chapter 4, we have found that sRGB model performs somehow randomly and brings about some extreme failure cases. Here, we concentrate on the performance of linear and quadratic models under different photography conditions. Tables 8.4 and 8.5 show the RMSE results for each sequence that are computed between the matched pixel vectors in reference image and any of another image in sequence.

The release of manual control on parameters introduce more chromatic differences to images as the original RMSEs of automatic adjusted sequences (smart phone) are generally much bigger than that of sequences with parameters fixed (prime/zoom lenses). While in any cases, linear and quadratic adjustment have nearly all the time positive effects on reducing contrast differences. The assumption of fixed camera parameters that we set up during model conception seems not to be an essential condition to the success of our adjustment. It may be released in later application.

Two kinds of approximation provide similar results. However, quadratic model performs better as it never gives updated RMSEs bigger than their original values. Linear model, even though its errors are always negligible (for example 0.001, 0.002), is less robust. We confirm therefore our judgment in Chapter 4 that quadratic approximation would be the best choice for chromatic alignment.

The DxO pre-processing does not have much impact on this step since there is no obvious trend of change between Table 8.4 and Table 8.5. A possible explication is that SIFT matched points are robust to deformation and distortion correction may only result in a change of sample choices.

		Original	Linear	Quadratic
Mario	1-2	0.022	**0.018**	0.019
	1-3	0.019	0.019	**0.019**
	1-4	0.019	**0.018**	0.019
	1-5	0.024	**0.020**	0.021
Dance	1-2	0.040	0.026	**0.026**
	1-3	0.044	**0.026**	0.027
Bakery	1-2	0.021	0.021	**0.021**
	1-3	0.028	**0.021**	0.024
	1-4	0.024	0.025	**0.024**
	1-5	0.034	**0.024**	0.025
Leopard	1-2	0.036	**0.026**	0.027
	1-3	0.034	0.033	**0.031**
	1-4	0.037	0.028	**0.028**
	1-5	0.047	0.083	**0.039**
Woman	1-2	0.015	0.013	**0.013**
	1-3	0.017	0.014	**0.014**
	1-4	0.034	0.014	**0.014**
	1-5	0.024	0.014	**0.014**
Cartoon	1-2	0.022	0.022	**0.021**
	1-3	0.022	0.022	**0.022**
	1-4	0.047	0.024	**0.023**
	1-5	0.061	0.025	**0.023**
Inversion	1-2	0.024	0.023	**0.023**
	1-3	0.031	0.032	**0.030**
	1-4	0.040	0.037	**0.037**
	1-5	0.036	0.034	**0.033**
Orient	1-2	0.025	0.022	**0.022**
	1-3	0.021	**0.017**	0.018
	1-4	0.023	0.020	**0.019**
Monster	1-2	0.041	0.041	**0.038**
	1-3	0.044	0.038	**0.036**
	1-4	0.052	0.039	**0.038**
	1-5	0.030	0.030	**0.030**

Table 8.4 – *Continued on next page*

Continued from previous page

		Original	Linear	Quadratic
Snoopy	1-2	0.022	**0.018**	0.019
	1-3	0.020	**0.014**	0.015
	1-4	0.025	0.019	**0.019**
	1-5	0.034	0.017	**0.017**
Babs	1-2	0.066	0.028	**0.026**
	1-3	0.043	**0.027**	0.028
	1-4	0.076	0.028	**0.027**
Planet	1-2	0.086	0.073	**0.060**
	1-3	0.053	0.054	**0.050**
	1-4	0.090	0.053	**0.052**
	1-5	0.072	0.056	**0.054**
Garfield	1-2	0.247	0.072	**0.063**
	1-3	0.042	**0.035**	0.036
	1-4	0.039	**0.030**	0.032
Banc	1-2	0.035	0.036	**0.035**
	1-3	0.208	**0.037**	0.038
Mexican	1-2	0.099	**0.033**	0.038
	1-3	0.106	0.039	**0.036**
	1-4	0.087	0.048	**0.041**
	1-5	0.345	**0.043**	0.054

TABLE 8.4: **RMSE of matched pixels in original sequences.** $1-i$ means evaluations are carried out between the reference image (1 st image) and another image (i th image) in sequence. Values are computed based on the normalized RGB vectors whose range has been transformed from $\{1,\ldots 255\}^3$ to $[0,1]^3$. Therefore, differences per channel under $1/255 = 0.004$ can be neglected. Similarly, differences of RMSE which are related to vector-distances are negligible under $\sqrt{3}/255 = 0.007$. The bold values gives are smallest RMSE of each row. The red values marks the values bigger than original RMSE.

		Original	Linear	Quadratic
Mario	1-2	0.022	0.018	**0.018**
	1-3	0.019	0.020	**0.019**
	1-4	0.020	0.020	**0.020**
	1-5	0.024	**0.019**	0.020
Dance	1-2	0.038	0.025	**0.025**
	1-3	0.043	0.026	**0.026**
Bakery	1-2	0.021	0.021	**0.021**
	1-3	0.028	0.021	**0.021**
	1-4	0.024	0.025	**0.024**
	1-5	0.034	**0.024**	0.025
Leopard	1-2	0.036	0.027	**0.027**
	1-3	0.033	**0.030**	0.031
	1-4	0.035	0.028	**0.028**
	1-5	0.047	0.043	**0.043**
Woman	1-2	0.016	0.014	**0.014**
	1-3	0.017	**0.014**	0.015
	1-4	0.034	0.013	**0.013**
	1-5	0.025	0.016	**0.015**
Cartoon	1-2	0.020	0.021	**0.020**
	1-3	0.021	0.022	**0.020**
	1-4	0.046	0.023	**0.022**
	1-5	0.061	0.025	**0.025**
Inversion	1-2	0.023	0.021	**0.021**
	1-3	0.033	0.032	**0.031**
	1-4	0.035	0.034	**0.033**
	1-5	0.034	0.027	**0.027**
Orient	1-2	0.064	0.047	**0.047**
	1-3	0.062	0.048	**0.048**
	1-4	0.044	0.041	**0.038**
Monster	1-2	0.077	**0.052**	0.059
	1-3	0.074	0.047	**0.045**
	1-4	0.068	**0.041**	0.042
	1-5	0.057	0.041	**0.039**

Table 8.5 – *Continued on next page*

Continued from previous page

		Original	Linear	Quadratic
Snoopy	1-2	0.047	0.030	**0.029**
	1-3	0.044	0.021	**0.020**
	1-4	0.080	0.038	**0.038**
	1-5	0.068	0.019	**0.019**
Babs	1-2	0.072	0.028	**0.027**
	1-3	0.042	0.027	**0.026**
	1-4	0.077	0.027	**0.027**
Planet	1-2	0.120	0.104	**0.083**
	1-3	0.067	0.061	**0.056**
	1-4	0.100	0.074	**0.070**
	1-5	0.094	0.074	**0.072**
Garfield	1-2	0.213	0.062	**0.057**
	1-3	0.057	0.064	**0.051**
	1-4	0.060	**0.038**	0.039
Banc	1-2	0.036	**0.035**	0.036
	1-3	0.206	0.039	**0.038**
Mexican	1-2	0.094	**0.034**	0.037
	1-3	0.107	0.034	**0.034**
	1-4	0.085	0.046	**0.045**
	1-5	0.342	0.051	**0.044**

TABLE 8.5: **RMSE of matched pixels in DxO sequences.** This table is organized as Table 8.4. The sequences are pre-proceeded with DxO Optics Pro11. Quadratic model performs the best with no obvious mistakes.

8.5 Image fusion

We compare four fusion methods in this section which are: meaningful clique (as is introduced in Chapter 6), geometric and degenerated medians (later called median 1 and 2) in Chapter 5 and RPCA, mentioned in Chapter 2. Synthetic sequences, benefiting from accurate alignment and reference without occlusion, are used to evaluate the error rates in fusion results. Real sequences, on the other hand, show the performance of different approaches in real field conditions.

Error rates in Table 8.6 give a general estimation on the ability of four methods to select the background. Apart from the badly evaluation on RPCA, probably due to the change of contrast, nearly all the worst cases come from the results of median 2. It is reasonable as median 2 is the only method depending on the analysis on intensity rather than RGB values. Degeneration leads to a loss of information. Median 1 shows high robustness when the background pixels exceed half of the total number (values in columns 4 and 5 are all zeros). However, its error rates increase significantly when the assumption on pixel number is no longer true. Its application is therefore limited.

Meaningful clique gives the best results based on its robustness under different conditions. Its error rates in columns 4 and 5 are nearly negligible, as for example, the number of wrongly-selected pixels in Seq.1 columns 4 equals approximately to: $60000 \times 0.0029\% = 1.74$. More importantly, it provides rather accurate choices until the case where only one pixel belongs to background and the minimum clique can not be set up.

In the tests on real sequences, the effects of geometric alignment errors involves in image resources which may affect fusion results by adding blur-like defects along contours of object. Hence, we focus here not only on the size of remaining masks, but also on the blurry defects of texture. In Figures 8.4, 8.5 and 8.6, we mark remaining masks with blue squares and we indicate areas to be enlarged for texture observation by red squares.

Figure 8.4 shows typical fusion results of prime lenses sequences. Meaningful clique method gives images with the least masks and the outputs of other methods are more or less the same. More concretely, RPCA performs sometimes better sometimes worse than median 1 while median 2 is always the worst. There is no obvious blur-like defect in results since aligned images do not leave much errors in previous steps. The enlarged areas of sequence with/without DxO pre-processing present hence the similar textures.

For zoom lenses sequences, both remaining masks and blur-like defects should be taken into consideration. As is shown by Figure 8.5, meaningful clique method leave again the least amount of masks in its results. Besides, we enlarge an area on the border of original image where there exists geometric errors of about 2 or 3 pixels. The chosen example represents a case where alignment errors are small but not negligible. Textures appear to be blurred due to a number of wrongly-selected pixels on contours (defects).

DxO pre-processing, which corrects efficiently errors of zoom lenses sequences, impacts positively the results of pixel selection. Enlarged details of DxO sequence (Figures 8.5j, 8.5l, 8.5n, 8.5p) are much sharper than those of original sequence (Figures 8.5i, 8.5k, 8.5m, 8.5o) for all fusion methods. There are still some tiny errors left along borders, nevertheless, applying DxO for distortion correction is an available way to ameliorate fusion quality of these sequences.

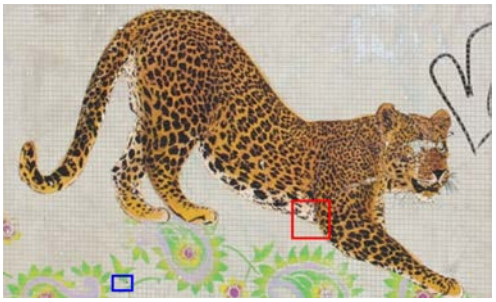
The performance differences of the four algorithms are more obvious in smart phone results with large error size. We select the sequence 'Mexican' as an example which is shown in Figure 8.6. This time, meaningful clique and RPCA leave similar amount of masks which is less than that of two median results. In enlarged area, defects are presented as gray or yellow shade along brim of hat marked by red circles. Different from tiny blur in previous Figure 8.5, it is possible to distinguish the pixel selection ability of each approach in existence of alignment errors.

Result of meaningful clique (Figure 8.6c) has the least defects, RPCA and median 1 results (Figures 8.6e and 8.6k) show similar range of gray area. Median 2 gives the worst result where it select even the yellow elements coming from a wrongly stretch of hat. These differences can be explained theoretically as meaningful cliques depend on the compact of elements while median selection is based on the order of elements. When the alignment errors account for more than half of the data set, median filter make more mistakes than meaningful clique method. RPCA refers to a decomposition of compact component in every elements (no element is discarded as meaningful clique). Logically, its performance should be between the above methods.

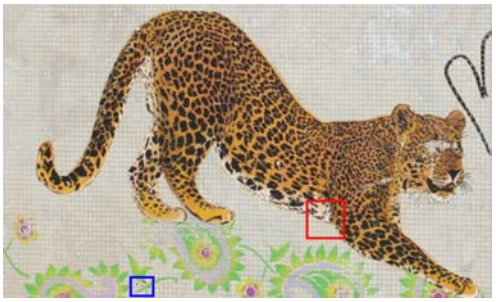
With DxO pre-processing, we notice no big change in terms of masks remaining in results. But the correction of aligned image indeed improves image quality as defects are reduced in enlarged area. Overall, meaningful clique method is our best choice for image fusion. It is consistently better to pre-proceed images with DxO correction for the sake of accuracy results.

		1	**2**	**3**	**4**	**5**	**Total**
Syn.1		(1e3)	(1e4)	(4e4)	(6e4)	(2e5)	(3e5)
	Clique	94.7	**10.5**	8.4e-2	2.9e-3	**0.0**	**9.0e-1**
	Median 1	97.4	61.8	**2.0e-2**	**0.0**	0.0	3.1
	Median 2	**91.3**	67.7	1.6	5.6e-1	0.0	3.6
	RPCA	92.7	71.9	5.3e-2	4.5e-3	2.7e-3	3.5
Syn.2		(0)	(1e3)	(2e4)	(5e4)	(5e5)	(6e5)
	Clique	–	**47.6**	8.5e-2	5.5e-3	**0.0**	**8.9e-2**
	Median 1	–	80.3	**1.7e-2**	**0.0**	0.0	1.4e-1
	Median 2	–	93.7	1.1	3.8e-1	0.0	2.4e-1
	RPCA	–	89.3	0.0	0.0	0.0	1.6e-1
Syn.3		(0)	(7e3)	(3e4)	(7e4)	(8e5)	(9e5)
	Clique	–	**6.7**	1.4e-2	**0.0**	**0.0**	**5.3e-2**
	Median 1	–	63.3	**5.9e-3**	0.0	0.0	4.9e-1
	Median 2	–	84.3	1.5	5.2e-1	0.0	7.6e-1
	RPCA	–	94.7	53.7	59.3	51.3	52.4
Syn.4		(1)	(1e4)	(2e4)	(9e4)	(8e5)	(9e5)
	Clique	100	**1.0**	1.2e-2	1.1e-3	**0.0**	**1.5e-2**
	Median 1	100	52.9	**0.0**	**0.0**	0.0	7.6e-1
	Median 2	0.0	63.2	2.1	6.9e-1	0.0	1.0
	RPCA	0.0	83.6	41.8	23.8	18.4	20.4
Syn.5		(2e3)	(1e4)	(5e4)	(1e5)	(1e5)	(3e5)
	Clique	**99.4**	**8.2**	4.4e-2	**0.0**	**0.0**	**1.1**
	Median 1	99.6	85.8	**3.5e-3**	0.0	0.0	5.4
	Median 2	99.5	90.2	1.3	5.6e-1	0.0	6.1
	RPCA	99.6	95.3	5.0e-1	3.7e-2	3.0e-2	6.0

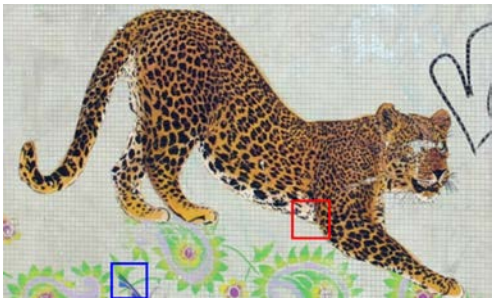
TABLE 8.6: Error rates in percentage. Computing method of this table has been presented in Chapter 6 Section 6.4.1. Its first row indicates the numbers of background-observed time. Values in parentheses refer to the quantity of pixels. The lowest rate in each column are marked in bold letter. The worst cases are marked in blue. Since the contrast of RPCA results are changed, their error rates are not taken into account.



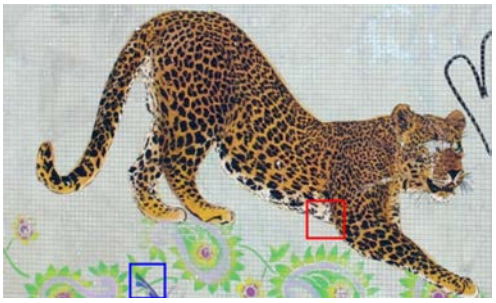
(a) Meaningful clique: original sequence



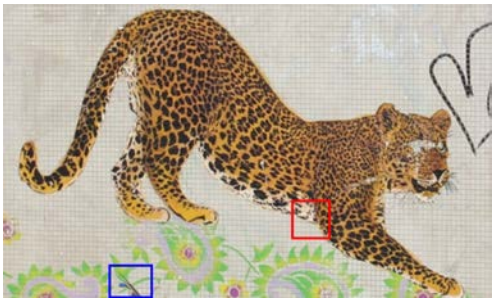
(b) Meaningful clique: DxO sequence



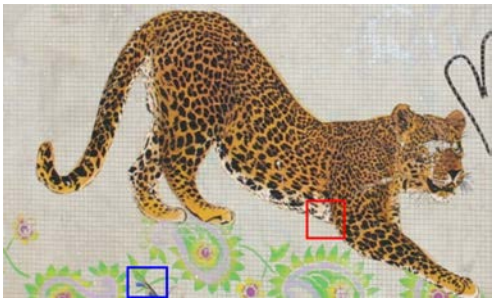
(c) RPCA: original sequence



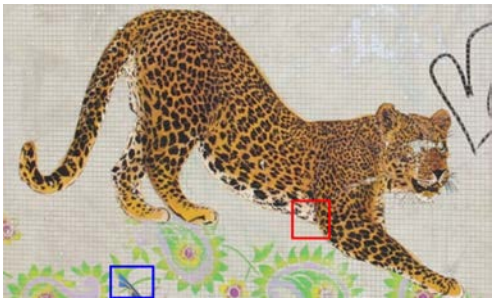
(d) RPCA: DxO sequence



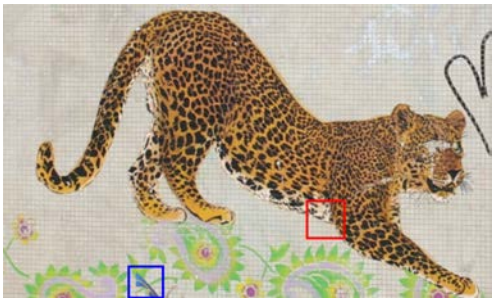
(e) Median 1: original sequence



(f) Median 1: DxO sequence



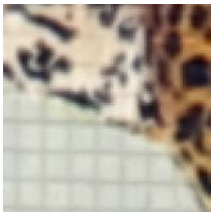
(g) Median 2: original sequence



(h) Median 2: DxO sequence

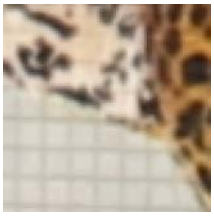
(i) Zoom of 8.4a

(j) Zoom of 8.4b



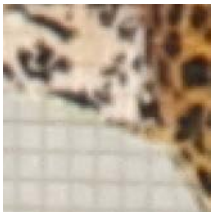
(k) Zoom of 8.4c

(l) Zoom of 8.4d



(m) Zoom of 8.4e

(n) Zoom of 8.4f



(o) Zoom of 8.4g

(p) Zoom of 8.4h

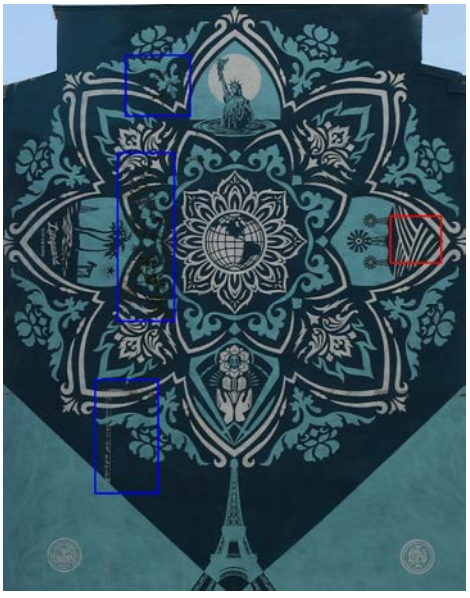
FIGURE 8.4: Fusion – Prime lenses sequence (Leopard).

(a) Meaningful clique: original sequence

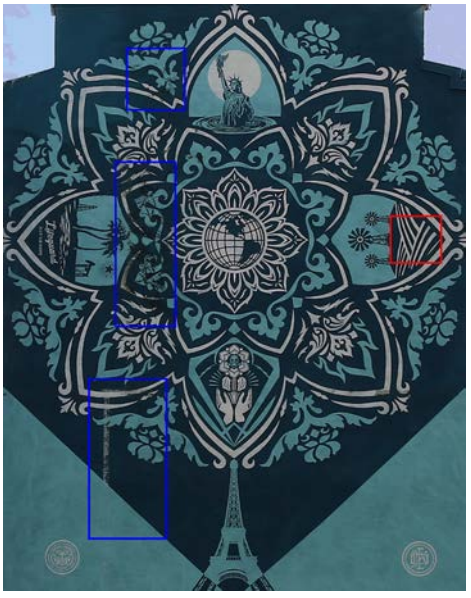
(b) Meaningful clique: DxO sequence

(c) RPCA: original sequence

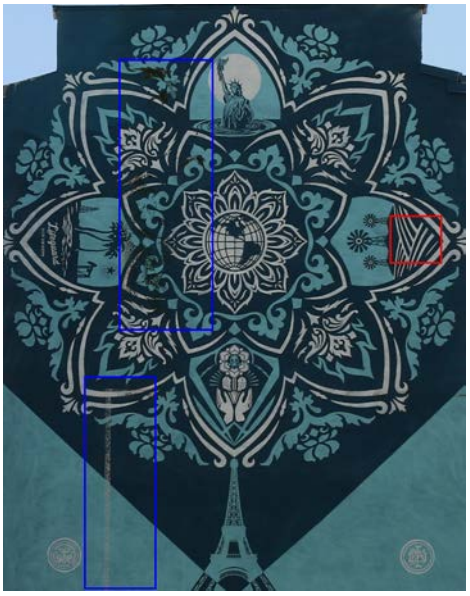
(d) RPCA: DxO sequence



(e) Median 1: original sequence



(f) Median1: DxO sequence



(g) Median 2: original sequence

(h) Median 2: DxO sequence

(i) Zoom of 8.5a

(j) Zoom of 8.5b

(k) Zoom of 8.5c

(l) Zoom of 8.5d

(m) Zoom of 8.5e

(n) Zoom of 8.5f

(o) Zoom of 8.5g

(p) Zoom of 8.5h

FIGURE 8.5: Fusion – Zoom lenses sequence (Orient).

(a) Meaningful clique: original sequence

(b) Meaningful clique: DxO sequence

(c) Zoom of 8.6a

(d) Zoom of 8.6b

(e) Zoom of 8.6g

(f) Zoom of 8.6h

(g) RPCA: original sequence

(h) RPCA: DxO sequence

(i) Median 1: original sequence

(j) Median 1: DxO sequence

(k) Zoom of 8.6i

(l) Zoom of 8.6j

(m) Zoom of 8.6o

(n) Zoom of 8.6p

(o) Median 2: original sequence

(p) Median 2: DxO sequence

FIGURE 8.6: Fusion – Smart phone sequence (Mexican).

8.6 Combination method

Finally, pixels in fusion results can be re-selected in order to remove totally the blur-like defects. Prime lenses results, usually accurate enough, show no big difference before/after this step. The enlarged patches in Figure 8.7 present very similar appearances without any newly introduced defects.

For results of zoom lenses sequences, some blur-like defects remain from previous step, as in Figure 8.8, the hand of boy is not clear enough in both zoom of original and DxO proceeded results. Our Gaussian strategy replaces this area by a more complete expression in original image, its results present textures with clear details even without using the DxO correction (see Figures 8.8d, 8.8h).

In smart phone sequences, defects are more serious as the bench and shoe loses their texture in Figures 8.9b and 8.9f. These patterns are restored by applying our method as is shown in Figures 8.9d and 8.9h. Again, sequences with/without DxO pre-processing present all perfect details. Because difference of fusion results lies in the size of defects, which will not affect the replacement of patches.

As a whole, our combination method appears as a simple yet very robust and efficient method to remove the blur-like defects left in fusion results.

8.7 Conclusion

To summarize, we propose a pipeline for mask removal consisting of four major steps which are geometric alignment, chromatic adjustment, image fusion and defect removal. It is able to deal with general cases where image sequences of nice or limited quality are taken with manually or automatically adjusted parameters. Best choices of each step are respectively homography transformation, quadratic approximation, meaningful clique method and combination method.

Quality of image determines the level of distortion which is major source of errors in homography transformation and pixel selection. DxO pre-processing ameliorate but can not eliminate this phenomenon when errors are rather important. Fortunately, the last step of our method enhances the robustness of pipeline and ensures result accuracy. Therefore, our approach has the ability to manage different conditions ranging from sequences captured by nice prime lenses to smart phone images without any intentional control. The use of other image processing software is not indispensable thanks to the efficient fallback offered by our defect removal strategy.

(a) Original fusion

(b) Original fusion after Gaussian processing

(c) Zoom of 8.7a

(d) Zoom of 8.7b

(e) Zoom of 8.7g

(f) Zoom of 8.7h

(g) DxO fusion

(h) DxO fusion after Gaussian processing

FIGURE 8.7: **Defect removal – Prime lenses sequence (Mario)**: maximum alignment errors are below 1 pixel, sigma of Gaussian filter is set at 1. The enlarged area is marked by red square in each image.

(a) Original fusion

(b) Zoom of 8.8a

(c) Original fusion after Gaussian processing

(d) Zoom of 8.8c

(e) DxO fusion

(f) Zoom of 8.8e

(g) DxO fusion after Gaussian processing

(h) Zoom of 8.8g

FIGURE 8.8: **Defect removal – Zoom lenses sequence (Inversion)**: maximum alignment errors are 4 to 5 pixels, sigma of Gaussian filter is set at 7.

(a) Original fusion

(b) Zoom of 8.9a

(c) Original fusion after Gaussian processing

(d) Zoom of 8.9c

(e) DxO fusion

(f) Zoom of 8.9e

(g) DxO fusion after Gaussian processing

(h) Zoom of 8.9g

FIGURE 8.9: **Defect removal – Smart phone sequence (Banc)**: maximum alignment errors are about 16 pixels, sigma of Gaussian filter is set at 25.

Chapter 9

Conclusion

Here we come to the conclusion of our work, in which we have addressed several problems to perform blind mask removal in digital images. We have focused on the multi-image problems so as to get enough information about the background (in particular avoiding inpainting techniques). Several problems appearing in real field conditions have been fixed by our algorithms. These solutions can be organized into three points:

1. **Adjustment of background contrast and color differences.** The differences can be effectively reduced by robustly estimated quadratic mappings on every RGB channels. The parameters of the mapping functions are computed with matched pixels issued from a feature extraction method (SIFT in our case) and then selected by a RANSAC strategy. This approach avoids the classical limitations of classical histogram adjustment methods in the case where the contents is changing a lot between images.
2. **Design of a pixel selection method with lenient assumptions.** We suppose that pixels originating from different images with the same content form a dense cloud of points in the RGB space. The cloud corresponding to the background is supposed to be the largest one and is picked out by a recursive algorithm searching for a dense clique. This method is proved to be more effective for background reconstruction than both a color median method and a robust PCA method.
3. **Elimination of fusion defects caused by geometrical alignment errors in the initial sequence.** The areas with blur-like defects are replaced by sharp patches of aligned images after a

similarity estimation. The estimation relies on a comparison between a threshold and the distance of pixels from images after a Gaussian filtering. This fallback solution permits us to relax many requirements ensuring that images can be perfectly aligned.

As a syntheses of the discussion in previous chapters, we propose a pipeline for mask removal shown by Figure 9.1. It is an automatic and fast method that releases the constraints in many previous works on the size and shape of occlusions. The missing areas are filled in with factual scene. Further more, although we have listed many assumptions in each chapter to ensure theoretically the performance of our design, the whole process shows its robustness when we release some of the constraints during the tests. Finally, the context can be generalized as:

1. The background to restore is a planar or a 3D view if it is captured with the optical center fixed.
2. Images in the sequence (typically 5) reveal the entire target scene several times depending on the combined motion of the photographer and the occlusions.
3. The angles of view do not change too much among images.
4. Cameras with manual or automatic sets are all available as long as the surface is quasi-Lambertian.
5. Images can be captured at different moments as long as there is no big change of lighting conditions.

Experiments in Chapter 8 confirms the good performance of our design. From another perspective, the assumptions on the context restrict the applying range of our approach. Therefore, the future work may focus on releasing limitations such as:

1. The target can not be a general 3D object.
2. No glossy or specular reflection should be found on the surface.
3. Masks should not be of similar color and be the predominant content in the stack of images.

To deal with the first constraint, the geometric alignment should be modified to a certain extent. If the 3D area is small, it is possible to keep actual steps and add patch-match [BSFG09, BSGF10] to ameliorate 3D region in aligned results. If the whole target is a complicated view, then the homography model should be totally

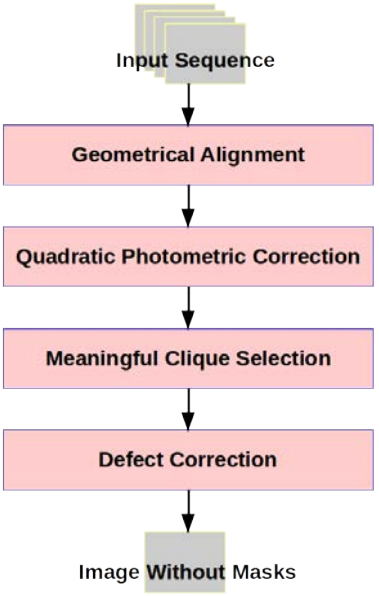


FIGURE 9.1: **Pipeline for mask removal.** It consists of four steps to restore the plane background from a sequence of images taken at different camera positions. In case of perfect shooting conditions, some of the steps can be omitted.

abandoned and we may account on 3D reconstruction in stereo-vision [JM15] or a more sophisticated application, bundle adjustment [HZ03, Mic18, TMHF99].

Many previous works have been done on reflection removal [LZW03, LZW04, BY08, SKG+12, GCM14]. These are of particular interest. Some of them set up their models with the motion information obtained by optical flow [SAA00, XRLF15] or by SIFT flow [LB13]. We may learn from these methods to modify either geometric alignment or image fusion step in our pipeline so as to release the second constraint.

In the case where the mask is very large and do not present color changes, temporal filtering is not enough and spatial information of patch is required. A possible solution is to generalize the meaningful clique approach to a spatio-temporal filtering method. Otherwise, the optical flow may help to distinguish the background and masks while it requires the images in sequence to be continuous frames in the video [MLY12, HS81].

Over all, we have discussed in detail the essential process for multi-image mask removal in this thesis. The final pipeline we proposed is simple and practical, but some assumptions in this design restrict its use to a certain extent. These limits require a future work on this topic that may relax the hypothesis in some steps of the pipeline.

Source Code

Geometric alignment and photometric adjustment (main.cpp)

```
#include <opencv2/opencv.hpp>
#include "opencv2/nonfree/nonfree.hpp"
#include <vector>
#include <sys/types.h>
#include <dirent.h>
#include <algorithm>

////////////////////////////////////////////////////////////////////////////
/// Geometric and photometric adjustment: align images in the sequence
/// @argv[1], directory of the folder of original images
////////////////////////////////////////////////////////////////////////////
int main(int, char *argv[])
{
    std::vector< cv::Mat > img_res, img_geo, homo_inliers;
    DIR* folder_dir = opendir(argv[1]);
    struct dirent* file_info;
    cv::Mat img_scene,descriptors_scene;
    std::vector<cv::KeyPoint> keypoints_scene;
    cv::SIFT sift;
    cv::BFMatcher matcher;
    std::vector< std::vector<cv::Point2f> > match_posi;
    const float ratio = 0.8;

    //**-- Step1: Geometric alignment --**//
    while( (file_info = readdir(folder_dir)) != NULL )
    {
       if( strcmp(file_info->d_name,".")!=0 && strcmp(file_info->d_name,"..")!=0 )
       {
           std::string dir_whole = argv[1];
           dir_whole = dir_whole+"/"+file_info->d_name;
           if( img_res.empty() )
           {
               img_res.push_back( cv::imread(dir_whole) );
               cv::cvtColor(img_res[0],img_scene,cv::COLOR_RGB2GRAY);
```

```
        sift(img_scene, cv::Mat(), keypoints_scene, descriptors_scene);
        img_res[0].convertTo(img_res[0],CV_32FC3);
    }
    else
    {
        cv::Mat img_object,descriptors_object;
        cv::Mat img_ori = cv::imread(dir_whole);
        cv::cvtColor(img_ori,img_object,cv::COLOR_RGB2GRAY);
        img_ori.convertTo(img_ori,CV_32FC3);

        // SIFT keypoints and descriptors //
        std::vector<cv::KeyPoint> keypoints_object;
        sift(img_object, cv::Mat(), keypoints_object, descriptors_object);

        // Match descriptors using FLANN matcher //
        std::vector<std::vector<cv::DMatch> > matches;
        matcher.knnMatch(descriptors_object, descriptors_scene, matches, 2);
        std::vector<cv::DMatch> good_matches;
        for (size_t i = 0; i < matches.size(); ++i)
        {
            if (matches[i][0].distance < ratio * matches[i][1].distance)
                good_matches.push_back(matches[i][0]);
        }

        // Get coordinates of matched pixels //
        std::vector<cv::Point2f> obj,scene;
        for( size_t i = 0; i < good_matches.size(); ++i )
        {
          obj.push_back( keypoints_object[ good_matches[i].queryIdx ].pt );
          scene.push_back( keypoints_scene[ good_matches[i].trainIdx ].pt );
        }
        match_posi.push_back(scene);

        // Compute homography //
        cv::Mat inlier_mask;
        cv::Mat H = findHomography( obj,scene,CV_RANSAC,1,inlier_mask );
        homo_inliers.push_back(inlier_mask);

        // Bilinear interpolation //
        std::vector<cv::Mat> img_ch,img_interp(3);
        cv::split(img_ori,img_ch);
        cv::warpPerspective( img_ch[0],img_interp[0],H,img_scene.size(),cv::
INTER_LINEAR );
        cv::warpPerspective( img_ch[1],img_interp[1],H,img_scene.size(),cv::
INTER_LINEAR );
        cv::warpPerspective( img_ch[2],img_interp[2],H,img_scene.size(),cv::
INTER_LINEAR );
        cv::Mat img_al;
```

```
            cv::merge(img_interp,img_al);
            img_geo.push_back(img_al);
        }
    }
 }

//**-- Step2: Chromatic adjustment --**//
 std::vector<cv::Mat> colchannel_scene, colchannel_obj;
 cv::Mat pow_two,obj_all;
 for( size_t img_num=0; img_num!=img_geo.size(); ++img_num)
 {
     // Prepare intensity matrix //
     cv::Mat obj_col, scene_col;
     std::vector<int> rand_posi;
     int match_num = 0;
     for(int i=0; i!=homo_inliers[img_num].rows; ++i)
     {
         if(homo_inliers[img_num].at<uchar>(i)==1)
         {
             obj_col.push_back(img_geo[img_num].at<cv::Vec3f>((match_posi[img_num])[i]))
;
             scene_col.push_back(img_res[0].at<cv::Vec3f>((match_posi[img_num])[i]));
             rand_posi.push_back(match_num);
             ++match_num;
         }
     }
     std::random_shuffle(rand_posi.begin(), rand_posi.end());
     cv::split(scene_col,colchannel_scene);
     cv::Mat all_one(obj_col.size(),CV_32FC3,cv::Scalar(1,1,1));
     cv::pow(obj_col,2,pow_two);
     cv::hconcat(all_one,obj_col,obj_all);
     cv::hconcat(obj_all,pow_two,obj_all);
     cv::split(obj_all,colchannel_obj);

     // Calculate transfer model with RANSAC //
     int inliers_max = 0;
     std::vector<cv::Mat> final_model;
     for( size_t i=0; i!=(rand_posi.size()/3); ++i )
     {
         cv::Mat inliers = cv::Mat::ones(colchannel_scene[0].size(),CV_8UC1);
         std::vector<cv::Mat> model(3);
         for( int j=0; j!=3; ++j )
         {
             cv::Mat ref = (cv::Mat_<float>(3,1)
                            <<colchannel_scene[j].at<float>(rand_posi[3*i]),
                              colchannel_scene[j].at<float>(rand_posi[3*i+1]),
                              colchannel_scene[j].at<float>(rand_posi[3*i+2]));
             cv::Mat tar(3,3,CV_32FC1);
```

```
                colchannel_obj[j].row(rand_posi[3*i]).copyTo( tar.row(0) );
                colchannel_obj[j].row(rand_posi[3*i+1]).copyTo( tar.row(1) );
                colchannel_obj[j].row(rand_posi[3*i+2]).copyTo( tar.row(2) );
                model[j] = tar.inv(cv::DECOMP_SVD)*ref;

                // Search inliers //
                cv::Mat check = (cv::abs(colchannel_obj[j]*model[j]-colchannel_scene[j])
    <=15)/255;
                cv::multiply(inliers,check,inliers);
            }
            int inliers_actual = cv::sum(inliers)[0];
            if( inliers_actual>inliers_max )
            {
                final_model.assign(model.begin(),model.end());
                inliers_max = inliers_actual;
            }
        }

        // Adjust image colors //
        std::vector<cv::Mat> color_ch(3),geo_channel(3);
        cv::split(img_geo[img_num],geo_channel);
        for(int j=0;j!=3;++j)
        {
            color_ch[j] = final_model[j].at<float>(0,0)*cv::Mat::ones(geo_channel[j].size()
    ,CV_32FC1)+
                    final_model[j].at<float>(1,0)*geo_channel[j];
            cv::Mat pow_interp;
            cv::pow(geo_channel[j],2,pow_interp);
            color_ch[j] = color_ch[j]+final_model[j].at<float>(2,0)*pow_interp;
        }
        cv::Mat img_colal;
        cv::merge(color_ch,img_colal);
        img_res.push_back(img_colal);
    }
    closedir(folder_dir);
    return 0;
}
```

Meaningful clique (main.cpp detail.h detail.cpp)

```
#include <opencv2/opencv.hpp>
#include <vector>
#include <math.h>
#include "detail.h"

////////////////////////////////////////////////////////////////////////////
/// meaningful clique: select the pixels belonging to the most dense clouds
/// @argv[1], directory of image folder
/// @argv[2], thershold of variance
////////////////////////////////////////////////////////////////////////////
int main(int, char *argv[])
{
    /* input images, convert them into float format */
    std::vector<cv::Mat> Input_image;
    size_t n_file = ReadImagesFromFolder( argv[1], Input_image);

    /* create pointers for original images */
    std::vector<cv::Mat_<cv::Vec3f>::iterator> Pt_img;
    for (size_t i=0; i!=n_file; ++i)
        Pt_img.push_back(Input_image[i].begin<cv::Vec3f>());
    cv::Mat_<cv::Vec3f>::iterator pt_end = Input_image[0].end<cv::Vec3f>();

    /* give an ascending order to the distance between every images  */
    std::vector<cv::Mat> Image_dist;
    PixelDistance( Input_image,n_file,Image_dist );

    /* create pointers for distance array */
    std::vector<cv::Mat_<float>::iterator> Pt_dist;
    for (size_t i=0; i!=Image_dist.size(); ++i)
        Pt_dist.push_back(Image_dist[i].begin<float>());

    /* create pointer for output image */
    cv::Mat Output( Input_image[0].rows, Input_image[0].cols, CV_32FC3 );
    cv::Mat_<cv::Vec3f>::iterator pt_out = Output.begin<cv::Vec3f>();

    /* traverse every pixel positions, select the cloud of pixels */
    while (Pt_img[0]!=pt_end)
    {
        /* examine if there are more than 2 pixels */
        int num_valable=0;
        for(size_t n_im=0; n_im!=n_file; ++n_im)
        {
            if( !isnan((*Pt_img[n_im])[0]) )
            {
                *pt_out = *Pt_img[n_im];
                ++num_valable;
```

```
            }
        }

        if(num_valable>1)
        {
            /* distance between every two images */
            std::vector<float> Member;
            for (size_t i=0; i!=Image_dist.size(); ++i)
                Member.push_back(*(Pt_dist[i]));

            /* get the distance order */
            cv::Mat matrix_order(n_file,n_file,CV_32FC1);
            getDistOrder( Member,n_file,matrix_order );

            /* get the candidate dense clouds */
            getCandiCloud(matrix_order,atof(argv[2]), Pt_img, pt_out);
        }
        else if(num_valable==0)
            *pt_out = cv::Vec3f(255,255,255);

        /* pass to the next pixel position */
        for (size_t i=0; i!=n_file; ++i)
            ++ Pt_img[i];
        for (size_t i=0; i!=Image_dist.size(); ++i)
            ++ Pt_dist[i];
        ++ pt_out;
    }

    /* save new image */
    std::string name_out = argv[1];
    name_out = name_out+"_clique.png";
    imwrite(name_out,Output);
    return 0;
}
```

```
#ifndef MAJOR_STEPS
#define MAJOR_STEPS
#include <opencv2/opencv.hpp>
#include <vector>

void valueOrder(std::vector<float> &source, std::vector<int> &image_order);
size_t searchVecElemNum( size_t i, size_t j, size_t n_file);
float calculateVari(cv::Mat &matchan3);
void findMedian(cv::Mat &cloud, cv::Mat_<cv::Vec3f>::iterator img_out);
size_t ReadImagesFromFolder(char *folder_name, std::vector<cv::Mat> &Input_Image);
void PixelDistance(std::vector<cv::Mat> &Image , size_t n_file, std::vector<cv::Mat> &
    Image_dist);
```

```
void getDistOrder(std::vector<float> &Member, size_t n_file, cv::Mat &order_mat);
void getCandiCloud(cv::Mat &order, float vari_thershold, std::vector<cv::Mat_<cv::Vec3f>::
    iterator> &Pt_img, cv::Mat_<cv::Vec3f>::iterator img_out);

#endif
```

```
#include <sys/types.h>
#include <dirent.h>
#include <string>
#include <vector>
#include <algorithm>
#include <math.h>
#include <opencv2/opencv.hpp>

size_t searchVecElemNum( size_t i, size_t j, size_t n_file)
{
    return j-1+i*(n_file-1)-i*(i+1)/2;
}

void valueOrder( std::vector<float> &source, std::vector<int> &image_order )
{
    std::vector<float> temporary(1,INFINITY);
    int accumulator=0;
    for(std::vector<float>::size_type src_num=0; src_num!=source.size(); ++src_num )
    {

        std::vector<int>::iterator iter_ord=image_order.begin();
        for(std::vector<float>::iterator iter_tem=temporary.begin();iter_tem!=temporary.end
    ();++iter_tem)
        {
            if(source[src_num]>=*iter_tem)
                ++iter_ord;
            else
            {
                temporary.insert(iter_tem,source[src_num]);
                break;
            }
        }
        image_order.insert(iter_ord,accumulator);
        ++accumulator;
    }
}

float calculateVari(cv::Mat &matchan3)
{
    cv::Mat matrix;
    matchan3.copyTo(matrix);
```

```
    float px_num = matrix.rows;
    cv::Scalar mean = cv::mean(matrix);
    matrix -= mean;
    cv::pow(matrix,2,matrix);
    cv::Scalar somme = cv::sum(matrix);
    float all = somme[0]+somme[1]+somme[2];
    all /= px_num;
    return sqrt(all);
}

/**
 * @brief find the median of a cloud of RGB vectors
 * @param cloud, Mat::CV_32FC3, a column of RGB vectors of the best cloud
 * @param img_out, Mat_<Vec3f>::iterator, save the median RGB vector of the cloud
 * @return none
 */
void findMedian(cv::Mat &cloud,cv::Mat_<cv::Vec3f>::iterator img_out )
{
    float dist_min = INFINITY;
    for (int i=0; i!=cloud.rows; ++i)
    {
        cv::Mat member;
        cloud.copyTo(member);

        cv::Mat center(member.rows,1,CV_32FC3,cv::Scalar(member.at<cv::Vec3f>(i,0)[0],
    member.at<cv::Vec3f>(i,0)[1],member.at<cv::Vec3f>(i,0)[2]));
        member -= center;
        cv::pow(member,2,member);

        std::vector< cv::Mat > channel(3);
        cv::split(member,channel);
        cv::Mat somme = channel[0]+channel[1]+channel[2];
        cv::pow(somme,0.5,somme);
        cv::Scalar dist = cv::sum(somme);

        if (dist[0]<dist_min)
        {
            dist_min = dist[0];
            *img_out = cloud.at<cv::Vec3f>(i,0);
        }
    }
}

/**
 * @brief read several image files in a folder, save them into a vector of Mat
 * @param Folder_name, char *, image folder
 * @param Input_Image, vector of Mat::CV_32FC3
 * @return number of files
```

```
 */
size_t ReadImagesFromFolder( char *folder_name, std::vector<cv::Mat> &Input_Image)
{
    DIR* folder_dir = opendir(folder_name);
    struct dirent* file_info;

    while( (file_info = readdir(folder_dir)) != NULL )
    {
       if( strcmp(file_info->d_name,".")!=0 && strcmp(file_info->d_name,"..")!=0 )
       {
           std::string whole_dir = folder_name;
           whole_dir = whole_dir+"/"+file_info->d_name;
           cv::Mat Image = cv::imread(whole_dir);
           Image.convertTo(Image,CV_32FC3);
           cv::Mat border_mask;
           cv::inRange(Image,cv::Scalar(255,255,255),cv::Scalar(255,255,255),border_mask);
           Image.setTo(cv::Scalar(NAN,NAN,NAN),border_mask);
           Input_Image.push_back(Image);
       }
    }

    closedir(folder_dir);
    return Input_Image.size();
}

/**
 * @brief calculate matrix of euclidean distance between RGB vectors at the same position
   of two images
 * @param Image, vector of Mat::CV_32FC3, Input images, already defined
 * @param n_file, unsigned, number of images
 * @param Image_dist, vector of Mat::CV_32FC1, matrix of euclidean distance, defined here
   such matrix between
 * image i and image j (i<j) is saved at position j-1+i*(n_file-1)-i*(i+1)/2. the search of
    position in
 * Image_dist is achieved by searchVecElemNum in detail_preparation.cpp
 * @return none
 */
void PixelDistance( std::vector<cv::Mat> &Image, size_t n_file, std::vector<cv::Mat> &
   Image_dist )
{
    for(size_t i=0; i<n_file-1; ++i)
    {
        for(size_t j=i+1; j<n_file; ++j)
        {
            std::vector<cv::Mat> Image_channel;
            cv::Mat image_diff = Image[i]-Image[j];
            cv::pow(image_diff,2,image_diff);
            cv::split(image_diff,Image_channel);
```

```
            cv::Mat dist_ij = Image_channel[0]+Image_channel[1]+Image_channel[2];
            cv::pow(dist_ij,0.5,dist_ij);
            Image_dist.push_back(dist_ij);
        }
    }
}

/**
 * @brief sort the distance of vector of images to the vector of a certain image, save
   orderly
 * the image numbers as a row of matrix
 * @param Member, vector of float, euclidean distance of two RGB vectors of every two
   images at a certain pixel
 * position
 * @param n_file, unsigned, number of images
 * @param order_mat, n_file*n_file Mat of CV_32FC1, map of distance order whose row save
   the image numbers on
 * ascending order, already defined
 * @return none
 */
void getDistOrder( std::vector<float> &Member, size_t n_file, cv::Mat &order_mat )
{
    for(size_t i=0; i<n_file; ++i)
    {
        /* pixel distances to pixel i */
        std::vector<float> dist_to_i;
        for(size_t j=0; j<n_file; ++j)
        {
            if(j==i)
                dist_to_i.push_back(0);
            else if(i<j)
                dist_to_i.push_back(Member[ searchVecElemNum(i,j,n_file) ]);
            else
                dist_to_i.push_back(Member[ searchVecElemNum(j,i,n_file) ]);
        }

        /* sort members in ascending order, return a map of image numbers */
        std::vector<int> vec_order;
        valueOrder( dist_to_i, vec_order );
        for (size_t k=0; k<n_file; ++k)
            order_mat.at<float>(i,k)=vec_order[k];
    }
}

/**
 * @brief get one or several clouds of RGB vectors sharing the similar values. for each
   cloud of pixels, save
 * the number of their belonging images as a vector
```

```
 * @param order, Mat::CV_32FC1, map of pixel distances
 * @param storage_cloud, vector of float vector, save image numbers of each cloud
 * @return none
 */
void getCandiCloud( cv::Mat &order, float vari_thershold, std::vector<cv::Mat_<cv::Vec3f>::
    iterator> &Pt_img, cv::Mat_<cv::Vec3f>::iterator img_out )
{
    const size_t size_mini(2), size_maxi(order.rows);
    std::vector< std::vector<float> > storage_cloud;

    /* increase the size of cloud by iteration */
    for (size_t size_current=size_mini; size_current<=size_maxi; ++size_current)
    {
        std::vector< std::vector<float> > actual_cloud;
        /* select each time a pixel to find possible dense cloud */
        for (size_t center=0; center!=size_maxi; ++center)
        {
            /* select the members most approche to the center */
            cv::Mat center_mat;
            order(cv::Rect(0,center,size_current,1)).copyTo(center_mat);
            std::set<float> center_set(center_mat.begin<float>(),center_mat.end<float>());

            /* initialize the first member set as center set */
            std::set<float> member_set1(center_set.begin(),center_set.end());

            /* calculate the intersection of two member sets */
            bool isFinish = 0;
            for (size_t mem_num=1; mem_num!=size_current;++mem_num)
            {
                size_t row_num = center_mat.at<float>(0,mem_num);
                cv::Mat member_mat;
                order(cv::Rect(0,row_num,size_current,1)).copyTo(member_mat);
                std::set<float> member_set2(member_mat.begin<float>(),member_mat.end<float
    >());

                std::set<float> intersect;
                std::set_intersection(member_set1.begin(), member_set1.end(), member_set2.
    begin(), member_set2.end(), inserter(intersect, intersect.begin()));

                /* analyze the result, prepare for the next iteration */
                if(intersect.size()!=size_current)
                    break;
                else
                {
                    member_set1.clear();
                    member_set1.insert(intersect.begin(),intersect.end());
                }
```

```
            /* ensure the iteration has been accomplished */
            if ( mem_num==(size_current-1) )
                isFinish = 1;
        }

        /* save a candidate result */
        if (isFinish)
        {
            std::vector<float> candidate_cloud;
            candidate_cloud.insert(candidate_cloud.begin(),member_set1.begin(),
member_set1.end());
            std::vector< std::vector<float> >::iterator repetation = std::find(
actual_cloud.begin(),actual_cloud.end(),candidate_cloud);
            if (repetation==actual_cloud.end())
                actual_cloud.push_back(candidate_cloud);
        }

    }

    /* examine the number clouds in storage */
    size_t cloud_size = actual_cloud.size();
    if (cloud_size>1) //  >1:continue searching; update storage
    {
        storage_cloud.clear();
        storage_cloud = actual_cloud;
    }
    else if(cloud_size==1) // 1:maybe find target cloud; examine variance
    {
         cv::Mat single_cloud(size_current,1,CV_32FC3);
         for(size_t candi_num=0; candi_num!=size_current; ++candi_num)
         {
             size_t img_num = (actual_cloud[0])[candi_num];
             single_cloud.at<cv::Vec3f>(candi_num,0) = *(Pt_img[img_num]);
         }
         float vari = calculateVari(single_cloud);
         if(vari<vari_thershold || size_current==2)
         {
             findMedian(single_cloud,img_out);
             break;
         }
         else
             cloud_size = 0;
    }

    if (cloud_size==0) // 0:select a cloud among previous result
    {
        /* search the values of pixels in different clouds */
        cv::Mat cloud_vari_mini;
```

```
            float vari_mini(INFINITY);
            size_t size_previous = size_current-1;
            for(size_t cloud_num=0; cloud_num!=storage_cloud.size(); ++cloud_num)
            {
                cv::Mat single_cloud(size_previous,1,CV_32FC3);
                for(size_t ele_num=0; ele_num!=size_previous; ++ele_num)
                {
                    size_t img_num = (storage_cloud[cloud_num])[ele_num];
                    single_cloud.at<cv::Vec3f>(ele_num,0) = *(Pt_img[img_num]);
                }
                float vari = calculateVari(single_cloud);
                if(vari<vari_mini)
                {
                    vari_mini = vari;
                    cloud_vari_mini = cv::Mat();
                    single_cloud.copyTo(cloud_vari_mini);
                }
            }
            findMedian(cloud_vari_mini,img_out);
            break;
        }
    }
}
```

Jitter blur correction (main.cpp)

```
#include <opencv2/opencv.hpp>
#include "opencv2/nonfree/nonfree.hpp"
#include <vector>
#include <sys/types.h>
#include <dirent.h>
#include <stdlib.h>

////////////////////////////////////////////////////////////////////////////
/// jitter blur corrector: replace the fusion area by patches in aligned images
/// @argv[1], directory of aligned image folder
/// @argv[2], name of the fusion image
/// @argv[3], sigma of Gaussian distribution
/// @argv[4], threshold of color distances
////////////////////////////////////////////////////////////////////////////
int main(int, char *argv[])
{
    int sigma = atoi(argv[3]);
    int wsize = sigma*4+1; //gaussian window size
    float thresh = atof(argv[4]);

    // read in images
```

```
cv::Mat fusion = cv::imread(argv[2]);
fusion.convertTo(fusion,CV_32FC3);
cv::Mat fusion_blur;
cv::GaussianBlur(fusion,fusion_blur,cv::Size(wsize,wsize),sigma,sigma,cv::
BORDER_REFLECT);
std::vector<cv::Mat> aligned;
DIR* folder_dir = opendir(argv[1]);
struct dirent* file_info;

std::vector<cv::Mat> similar; //'difference' is marked as 0, 'similarity' is marked as
1.
while( (file_info = readdir(folder_dir)) != NULL )
{
   if( strcmp(file_info->d_name,".")!=0 && strcmp(file_info->d_name,"..")!=0 )
   {
       std::string dir_whole = argv[1];
       dir_whole = dir_whole+"/"+file_info->d_name;
       cv::Mat img_ori = cv::imread(dir_whole);
       img_ori.convertTo(img_ori,CV_32FC3);
       aligned.push_back(img_ori);

       //**-- Step1: blur images --**//
       cv::Mat img_blur;
       cv::GaussianBlur(img_ori,img_blur,cv::Size(wsize,wsize),sigma,sigma,cv::
BORDER_REFLECT);

       //**-- Step2: create distance cards --**//
       cv::Mat vdist = img_blur-fusion_blur;
       cv::pow(vdist,2,vdist);
       std::vector< cv::Mat > cdist(3);
       cv::split(vdist,cdist);
       cv::Mat ndist = cdist[0]+cdist[1]+cdist[2];
       cv::pow(ndist,0.5,ndist);
       ndist.setTo(-1,ndist>=thresh);
       ndist.setTo(0,ndist!=-1);
       ndist = ndist+1;
       similar.push_back(ndist);
   }
}

cv::Mat index = cv::Mat::ones(aligned.size(),1,CV_32FC1); //0:the image has been used
cv::Mat left = cv::Mat::ones(fusion.size(),CV_32FC1);//1:area left to be corrected
cv::Mat result(fusion.size(),CV_32FC3,cv::Scalar(0,0,0));
std::vector< cv::Mat > cresult(3);
cv::split(result,cresult);
while( cv::sum(index)[0]!=0 && cv::sum(left)[0]!=0 )
{
    //**-- Step3: find candidate aligned image --**//
```

```
        int max_area = 0, best_choice = -1;
        for(int i=0; i!=similar.size(); ++i)
        {
            if( index.at<float>(i,0)==0 )
                continue;
            int actual_area = cv::sum(similar[i].mul(left))[0];
            if( actual_area>max_area )
            {
                max_area = actual_area;
                best_choice = i;
            }
        }

        //**-- Step4: fill in the final image --**//
        cv::Mat mask = similar[best_choice].mul(left);
        std::vector< cv::Mat > caligned(3);
        cv::split(aligned[best_choice],caligned);
        cresult[0] = cresult[0]+caligned[0].mul(mask);
        cresult[1] = cresult[1]+caligned[1].mul(mask);
        cresult[2] = cresult[2]+caligned[2].mul(mask);

        //**-- Step5: update index and area left to be corrected --**//
        index.at<float>(best_choice,0) = 0;
        left = left.mul(1-mask);
    }

    cv::merge(cresult,result);
    result.convertTo(result,CV_8UC3);
    cv::imshow( "result", result );
    cv::waitKey(0);

    return 0;
}
```

Bibliography

[ADGM14] Cecilia Aguerrebere, Julie Delon, Yann Gousseau, and Pablo Musé. Best algorithms for hdr image generation. a study of performance bounds. *SIAM Journal on Imaging Sciences*, 7(1):1–34, 2014.

[AFM98] Naoki Asada, Hisanaga Fujiwara, and Takashi Matsuyama. Analysis of photometric properties of occluding edges by the reversed projection blurring model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):155–167, 1998.

[AGS11] Luis Alvarez, Luis Gómez, and J Rafael Sendra. Accurate depth dependent lens distortion models: an application to planar view scenarios. *Journal of Mathematical Imaging and Vision*, 39(1):75–85, 2011.

[AHN90] Jaakko Astola, Petri Haavisto, and Yrjo Neuvo. Vector median filters. *Proceedings of the IEEE*, 78(4):678–689, 1990.

[AM79] Brian DO Anderson and John B Moore. Optimal filtering. *Englewood Cliffs*, 21:22–95, 1979.

[AMCS96] Matthew Anderson, Ricardo Motta, Srinivasan Chandrasekar, and Michael Stokes. Proposal for a standard default color space for the internetsrgb. In *Color and imaging conference*, pages 238–245. Society for Imaging Science and Technology, 1996.

[Aya91] Nicholas Ayache. *Artificial vision for mobile robots: stereo vision and multisensory perception.* Mit Press, 1991.

[BCMS09] Antoni Buades, Bartomeu Coll, Jean-Michel Morel, and Catalina Sbert. Self-similarity driven color demosaicking. *IEEE Transactions on Image Processing*, 18(6):1192–1202, 2009.

[BD13] Faisal Bukhari and Matthew N Dailey. Automatic radial distortion estimation from a single image. *Journal of mathematical imaging and vision*, 45(1):31–45, 2013.

[BH+92] Robert Grover Brown, Patrick YC Hwang, et al. *Introduction to random signals and applied Kalman filtering*, volume 3. Wiley New York, 1992.

[BKN07] Peter Barnum, Takeo Kanade, and Srinivasa Narasimhan. Spatio-temporal frequency analysis for removing rain and snow from videos. In *Proceedings of the First International Workshop on Photometric Analysis For Computer Vision-PACV 2007*, pages 8–p. INRIA, 2007.

[BM+76] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Citeseer, 1976.

[BNK10] Peter C Barnum, Srinivasa Narasimhan, and Takeo Kanade. Analysis of rain and snow in frequency space. *International journal of computer vision*, 86(2-3):256, 2010.

[BSCB00] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[BSFG09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (ToG)*, 28(3):24, 2009.

[BSGF10] Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. The generalized patchmatch correspondence algorithm. In *European Conference on Computer Vision*, pages 29–43. Springer, 2010.

[BSL+11] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[BW11] Randolph Blake and Hugh Wilson. Binocular vision. *Vision research*, 51(7):754–770, 2011.

[BWS05] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International journal of computer vision*, 61(3):211–231, 2005.

[BY08] Efrat Be'Ery and Arie Yeredor. Blind separation of superimposed shifted images using parameterized joint diagonalization. *IEEE Transactions on Image Processing*, 17(3):340–353, 2008.

[CC06] K-H Chung and Y-H Chan. Color demosaicing using variance of color differences. *IEEE Transactions on Image Processing*, 15(10):2944–2955, 2006.

[CF05] David Claus and Andrew W Fitzgibbon. A rational function lens distortion model for general cameras. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 213–219. IEEE, 2005.

[CGL+09] Minming Chen, Arvind Ganesh, Zhouchen Lin, Yi Ma, John Wright, and Leqin Wu. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. *Coordinated Science Laboratory Report no. UILU-ENG-09-2214*, 2009.

[CGPV03] Rita Cucchiara, Costantino Grana, Andrea Prati, and Roberto Vezzani. A hough transform-based method for radial lens distortion correction. In *Image Analysis and Processing, 2003. Proceedings. 12th International Conference on*, pages 182–187. IEEE, 2003.

[CH13] Yi-Lei Chen and Chiou-Ting Hsu. A generalized low-rank appearance model for spatio-temporally correlated rain streaks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1968–1975, 2013.

[CLMW11] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[CPT03] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. Object removal by exemplar-based inpainting. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2. IEEE, 2003.

[CPT04] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on image processing*, 13(9):1200–1212, 2004.

[CWS+16] Wenfei Cao, Yao Wang, Jian Sun, Deyu Meng, Can Yang, Andrzej Cichocki, and Zongben Xu. Total variation regularized tensor rpca for background subtraction from compressive measurements. *IEEE Transactions on Image Processing*, 25(9):4075–4090, 2016.

[DF01] Frederic Devernay and Olivier Faugeras. Straight lines have to be straight. *Machine vision and applications*, 13(1):14–24, 2001.

[DJ10] Cuong Manh Do and Bahram Javidi. 3d integral imaging reconstruction of occluded objects using independent component analysis-based k-means clustering. *Journal of display technology*, 6(7):257–262, 2010.

[DJB14] Daniel D Doyle, Alan L Jennings, and Jonathan T Black. Optical flow background estimation for real-time pan/tilt camera object tracking. *Measurement*, 48:195–207, 2014.

[Dua71] C Brown Duane. Close-range camera calibration. *Photogramm. Eng*, 37(8):855–866, 1971.

[EW02] Geoffrey Egnal and Richard P Wildes. Detecting binocular half-occlusions: Empirical comparisons of five approaches. *IEEE Transactions on pattern analysis and machine intelligence*, 24(8):1127–1133, 2002.

[FB87] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision*, pages 726–740. Elsevier, 1987.

[FBH97] Hugh S Fairman, Michael H Brill, and Henry Hemmendinger. How the cie 1931 color-matching functions were derived from wright-guild data. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 22(1):11–23, 1997.

[FdWE03] Dirk Farin, Peter HN de With, and Wolfgang Effelsberg. Robust background estimation for complex video sequences. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I–145. IEEE, 2003.

[Fit01] Andrew W Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[FMG16] Muhammad Shahid Farid, Arif Mahmood, and Marco Grangetto. Image de-fencing framework with hybrid inpainting algorithm. *Signal, Image and Video Processing*, 10(7):1193–1201, 2016.

[FS03] Paolo Favaro and Stefano Soatto. Seeing beyond occlusions (and other marvels of a finite lens aperture). In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2003.

[GBZ12] Charles Guyon, Thierry Bouwmans, and El-hadi Zahzah. Robust principal component analysis for background subtraction: Systematic evaluation and comparative analysis. In *Principal component analysis*. InTech, 2012.

[GCM14] Xiaojie Guo, Xiaochun Cao, and Yi Ma. Robust separation of reflection from multiple images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2187–2194, 2014.

[GCW14] Zhi Gao, Loong-Fah Cheong, and Yu-Xiang Wang. Block-sparse rpca for salient motion detection. *IEEE transactions on pattern analysis and machine intelligence*, 36(10):1975–1987, 2014.

[Get12] Pascal Getreuer. Image demosaicking with contour stencils. *Image Processing On Line*, 2:22–34, 2012.

[GLM14] Christine Guillemot and Olivier Le Meur. Image inpainting: Overview and recent advances. *IEEE signal processing magazine*, 31(1):127–144, 2014.

[GLY92] Davi Geiger, Bruce Ladendorf, and Alan Yuille. Occlusions and binocular stereo. In *European Conference on Computer Vision*, pages 425–433. Springer, 1992.

[GLY95] Davi Geiger, Bruce Ladendorf, and Alan Yuille. Occlusions and binocular stereo. *International Journal of Computer Vision*, 14(3):211–226, 1995.

[Gol04] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.

[GR13] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013.

[GTCS+01] Daniel Gutchess, M Trajkovics, Eric Cohen-Solal, Damian Lyons, and Anil K Jain. A background model initialization algorithm for video surveillance. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 733–740. IEEE, 2001.

[GVL12] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[GZX01] Da-shan Gao, Jie Zhou, and Le-ping Xin. A novel algorithm of adaptive background estimation. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 2, pages 395–398. IEEE, 2001.

[Ham13] Hamamatsu. Learning center in digital imaging : Charge coupled device ccd linearity. on line, 2013.

[HFB14] Soren Hauberg, Aasa Feragen, and Michael J Black. Grassmann averages for scalable robust pca. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3810–3817, 2014.

[HK07] Samuel W Hasinoff and Kiriakos N Kutulakos. A layer-based restoration framework for variable-aperture photography. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[HLEL06] James Hays, Marius Leordeanu, Alexei A Efros, and Yanxi Liu. Discovering texture regularity as a higher-order correspondence problem. In *European Conference on Computer Vision*, pages 522–535. Springer, 2006.

[HS81] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[HS07] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[Hub11] Peter J Huber. Robust statistics. In *International Encyclopedia of Statistical Science*, pages 1248–1251. Springer, 2011.

[Hun05] Robert William Gainer Hunt. *The reproduction of colour*. John Wiley & Sons, 2005.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[JBJ15] Sajid Javed, Theirry Bouwmans, and Soon Ki Jung. Depth extended online rpca with spatiotemporal constraints for robust background subtraction. In *Frontiers of Computer Vision (FCV), 2015 21st Korea-Japan Joint Workshop on*, pages 1–6. IEEE, 2015.

[JJMB16] Sajid Javed, Soon Ki Jung, Arif Mahmood, and Thierry Bouwmans. Motion-aware graph regularized rpca for background modeling of complex scenes. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 120–125. IEEE, 2016.

[JM15] Laura Fernández Julià and Pascal Monasse. Bilaterally weighted patches for disparity map computation. *Image Processing On Line*, 5:73–89, 2015.

[JOd+15] NP Jerome, MR Orton, JA dArcy, T Feiweier, N Tunariu, DM Koh, MO Leach, and DJ Collins. Use of the temporal median and trimmed mean mitigates effects of respiratory motion in multiple-acquisition abdominal diffusion imaging. *Physics in Medicine & Biology*, 60(2):N9, 2015.

[Jol11] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

[KKPD04] Sabine Kurz, Frank Krummenauer, Norbert Pfeiffer, and H Burkhard Dick. Monocular versus binocular pupillometry. *Journal of Cataract & Refractive Surgery*, 30(12):2551–2556, 2004.

[KLF12] Li-Wei Kang, Chia-Wen Lin, and Yu-Hsiang Fu. Automatic single-image-based rain streaks removal via image decomposition. *IEEE Transactions on Image Processing*, 21(4):1742, 2012.

[Kom06] Nikos Komodakis. Image completion using global optimization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 442–452. IEEE, 2006.

[KR07] Thommen Korah and Christopher Rasmussen. Spatiotemporal inpainting for recovering texture maps of occluded building facades. *IEEE Transactions on Image Processing*, 16(9):2262–2271, 2007.

[KT07] Nikos Komodakis and Georgios Tziritas. Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *IEEE Transactions on Image Processing*, 16(11):2649–2661, 2007.

[KWM94] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. In *European Conference on Computer Vision*, pages 189–196. Springer, 1994.

[KZ01] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 508–515. IEEE, 2001.

[LAGM17] Thuc Le, Andrés Almansa, Yann Gousseau, and Simon Masnou. Motion-consistent video inpainting. In *ICIP 2017: IEEE International Conference on Image Processing*, 2017.

[Lam05] Edmund Y Lam. Combining gray world and retinex theory for automatic white balance in digital photography. In *Consumer Electronics, 2005.(ISCE 2005). Proceedings of the Ninth International Symposium on*, pages 134–139. IEEE, 2005.

[LB13] Yu Li and Michael S Brown. Exploiting reflection change for automatic reflection removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2432–2439, 2013.

[LBHL08] Yanxi Liuy, Tamara Belkina, James H Hays, and Roberto Lublinerman. Image de-fencing. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[LCM10] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[Len87] Reimar Lenz. Linsenfehlerkorrigierte eichung von halbleiterkameras mit standardobjektiven für hochgenaue 3dmessungen in echtzeit. In *Mustererkennung 1987*, pages 212–216. Springer, 1987.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996.

[LK+81] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[Low99] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LP49] R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[LPBVD15] Benjamin Laugraud, Sébastien Piérard, Marc Braham, and Marc Van Droogenbroeck. Simple median-based method for stationary background generation using background subtraction algorithms. In *International Conference on Image Analysis and Processing*, pages 477–484. Springer, 2015.

[LZ03] M Langer and Q Zhang. Rendering falling snow using an inverse fourier transform. *ACM SIGGRAPH technical sketches program*, 2003.

[LZW03] Anat Levin, Assaf Zomet, and Yair Weiss. Learning to perceive transparency from the statistics of natural scenes. In *Advances in Neural Information Processing Systems*, pages 1271–1278, 2003.

[LZW04] Anat Levin, Assaf Zomet, and Yair Weiss. Separating reflections from a single image using local features. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004.

[M+17] Henri Maître et al. *From Photon to pixel: the digital camera handbook*. John Wiley & Sons, 2017.

[MBV12] C Marghes, T Bouwmans, and R Vasiu. Background modeling and foreground detection via a reconstructive and discriminative subspace learning approach. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012.

[MGS11] Samuel Morillas, Valentín Gregori, and Almanzor Sapena. Adaptive marginal median filter for colour images. *Sensors*, 11(3):3205–3213, 2011.

[MHB+10] Elmar Mair, Gregory D Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *European conference on Computer vision*, pages 183–196. Springer, 2010.

[Mic18] MicMac. MicMac main page, 2018.

[MLS10] Scott McCloskey, Michael Langer, and Kaleem Siddiqi. Removing partial occlusion from blurred thin occluders. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 4400–4403. IEEE, 2010.

[MLY12] Yadong Mu, Wei Liu, and Shuicheng Yan. Video de-fencing. *arXiv preprint arXiv:1210.2388*, 2012.

[MLY14] Yadong Mu, Wei Liu, and Shuicheng Yan. Video de-fencing. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(7):1111–1121, 2014.

[MMSZ05] Stefano Messelodi, Carla Maria Modena, Nicola Segata, and Michele Zanin. A kalman filter based background updating algorithm robust to sharp illumination changes. In *International Conference on Image Analysis and Processing*, pages 163–170. Springer, 2005.

[MP14] L Maddalena and A Petrosino. Background model initialization for static cameras. *Background Modeling and Foreground Detection for Video Surveillance*, pages 3–1, 2014.

[MS95] Nigel JB McFarlane and C Paddy Schofield. Segmentation and tracking of piglets in images. *Machine vision and applications*, 8(3):187–193, 1995.

[MS04] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.

[NAF+13] Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, and Patrick Pérez. Towards fast, generic video inpainting. In *Proceedings of the 10th European Conference on Visual Media Production*, page 7. ACM, 2013.

[Ng17] Ren Ng. Lytro redefines photography with light field cameras, 2017.

[Ote15] Ives Rey Otero. *Anatomy of the SIFT Method*. PhD thesis, École normale supérieure de Cachan-ENS Cachan, 2015.

[PBCL10] Minwoo Park, Kyle Brocklehurst, Robert T Collins, and Yanxi Liu. Image de-fencing revisited. In *Asian Conference on Computer Vision*, pages 422–434. Springer, 2010.

[PCL08] Minwoo Park, Robert T Collins, and Yanxi Liu. Deformed lattice discovery via efficient mean-shift belief propagation. In *European Conference on Computer Vision*, pages 474–485. Springer, 2008.

[Pro17] Edoardo Provenzi. *Computational Color Science: Variational Retinex-like Methods*. John Wiley & Sons, 2017.

[PSB05] Kedar A Patwardhan, Guillermo Sapiro, and Marcelo Bertalmio. Video inpainting of occluding and occluded objects. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 2, pages II–69. IEEE, 2005.

[Pur71] Madan Lal Puri. Nonparametric methods in multivariate analysis. Technical report, 1971.

[PW12] Christian Perwass and Lennart Wietzke. Single lens 3d-camera with extended depth-of-field. In *Human Vision and Electronic Imaging XVII*, volume 8291, page 829108. International Society for Optics and Photonics, 2012.

[RDGM10] Julien Rabin, Julie Delon, Yann Gousseau, and Lionel Moisan. Macransac: a robust algorithm for the recognition of multiple objects. In *Fifth International Symposium on 3D Data Processing, Visualization and Transmission (3DPTV 2010)*, page 051, 2010.

[RMK95] Christof Ridder, Olaf Munkelt, and Harald Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *Proceedings of International Conference on recent Advances in Mechatronics*, pages 193–199. Citeseer, 1995.

[Row98] Sam T Roweis. Em algorithms for pca and spca. In *Advances in neural information processing systems*, pages 626–632, 1998.

[SAA00] Richard Szeliski, Shai Avidan, and P Anandan. Layer extraction from multiple images containing reflections and transparency. In *cvpr*, page 1246. IEEE, 2000.

[SACM12] Michael Stokes, Mattew Anderson, Srinivasan Chandrasekar, and Ricardo Motta. A standard default color space for the internetsrgb, 1996. *URL http://www. w3. org/Graphics/Color/sRGB*, 2012.

[SBBB86] James E Sheedy, Ian L Bailey, Markus Buri, and Eric Bass. Binocular vs. monocular task performance. *American journal of optometry and physiological optics*, 63(10):839–846, 1986.

[SBZ16] Andrews Sobral, Thierry Bouwmans, and E-h Zahzah. Lrslibrary: Low-rank and sparse tools for background modeling and subtraction in videos. *Robust Low-Rank and Sparse Matrix Decomposition: Applications in Image and Video Processing*, 2016.

[SFW14] Shao-Hua Sun, Shang-Pu Fan, and Yu-Chiang Frank Wang. Exploiting image structural similarity for single image rain removal. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 4482–4486. IEEE, 2014.

[SKG+12] Sudipta N Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. Image-based rendering for scenes with reflections. *ACM Trans. Graph.*, 31(4):100–1, 2012.

[SLKS05] Jian Sun, Yin Li, Sing Bing Kang, and Heung-Yeung Shum. Symmetric stereo matching for occlusion handling. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 399–406. IEEE, 2005.

[SLP83] RJ Stevens, AF Lehar, and FH Preston. Manipulation and presentation of multidimensional image data using the peano scan. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 520–526, 1983.

[SN94] Tong Sun and Yrjö Neuvo. Detail-preserving median based filters in image processing. *Pattern Recognition Letters*, 15(4):341–347, 1994.

[SO13] Hayden Schaeffer and Stanley Osher. A low patch-rank interpretation of texture. *SIAM Journal on Imaging Sciences*, 6(1):226–262, 2013.

[Tha15] Bansi B Thanki. Overview of an image inpainting techniques. *International Journal For Technological Research In Engineering*, 2(5):388–391, 2015.

[TMHF99] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmenta modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[TOF02] Emanuele Trucco, Francesca Odone, and Andrea Fusiello. Layered representation of a video shot with mosaicing. *Pattern Analysis & Applications*, 5(3):296–305, 2002.

[VAS82] KK VASILEV. Kalman filter theory. *Radioelektronika*, 25:102–104, 1982.

[VETC07] George Vogiatzis, Carlos Hernández Esteban, Philip HS Torr, and Roberto Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2241–2246, 2007.

[VK05] Johannes Von Kries. Influence of adaptation on the effects produced by luminous stimuli. *handbuch der Physiologie des Menschen*, 3:109–282, 1905.

[VLS+06] Vaibhav Vaish, Marc Levoy, Richard Szeliski, C Lawrence Zitnick, and Sing Bing Kang. Reconstructing occluded surfaces using synthetic apertures: Stereo, focus and robust measures. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2331–2338. IEEE, 2006.

[VWJL04] V. Vaish, B. Wilburn, N. Joshi, and M. Levoy. Using plane+ parallax for calibrating dense camera arrays. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proc. 2004 IEEE Comp. Soc. Conf.*, volume 1, pages I–I. IEEE, 2004.

[WCF05] Ching-Chih Weng, Homer Chen, and Chiou-Shann Fuh. A novel automatic white balance method for digital still cameras. In *Circuits*

and Systems, 2005. ISCAS 2005. IEEE International Symposium on, pages 3801–3804. IEEE, 2005.

[WFZ02] Yonatan Wexler, Andrew Fitzgibbon, and Andrew Zisserman. Bayesian estimation of layers from multiple images. In *European Conference on Computer Vision*, pages 487–501. Springer, 2002.

[Wik06] Wikipedia. Bayer pattern on sensor, 2006.

[Wik07] Wikipedia. Three iterations of a peano curve construction, whose limit is a space-filling curve., 2007.

[WJV+05] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High performance imaging using large camera arrays. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 765–776. ACM, 2005.

[WLP+14] Jing Wang, Ke Lu, Daru Pan, Ning He, and Bing-kun Bao. Robust object removal with an exemplar-based image inpainting approach. *Neurocomputing*, 123:150–155, 2014.

[WQS09] Aiqi Wang, Tianshuang Qiu, and Longtan Shao. A simple method of radial distortion correction with centre of distortion estimation. *Journal of Mathematical Imaging and Vision*, 35(3):165–172, 2009.

[WR06] Jiying Wu and Qiuqi Ruan. Object removal by cross isophotes exemplar-based inpainting. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 810–813. IEEE, 2006.

[WS82] Gunter Wyszecki and Walter Stanley Stiles. *Color science*, volume 8. Wiley New York, 1982.

[XRLF15] Tianfan Xue, Michael Rubinstein, Ce Liu, and William T Freeman. A computational approach for obstruction-free photography. *ACM Transactions on Graphics (TOG)*, 34(4):79, 2015.

[XSZ17] Zhaolin Xiao, Lipeng Si, and Guoqing Zhou. Seeing beyond foreground occlusion: A joint framework for sap-based scene depth and appearance reconstruction. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):979–991, 2017.

[XWSZ14] Zhaolin Xiao, Qing Wang, Lipeng Si, and Guoqing Zhou. Reconstructing scene depth and appearance behind foreground occlusion

using camera array. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 41–45. IEEE, 2014.

[YBF04] Hulya Yalcin, Michael J Black, and Ronan Fablet. The dense estimation of motion and appearance in layers. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, pages 165–165. IEEE, 2004.

[YGMT18] Xiaoyi Yang, Yann Gousseau, Henri Maître, and Yohann Tendero. Webpage of Occlusion Removal Method, 2018.

[YHCB05] Hulya Yalcin, Martial Hebert, Robert Collins, and Michael J Black. A flow-based approach to vehicle detection and background mosaicking in airborne video. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 1202–vol. IEEE, 2005.

[YK06] Kuk-Jin Yoon and In So Kweon. Adaptive support-weight approach for correspondence search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):650–656, 2006.

[YKA13] Atsushi Yamashita, Toru Kaneko, and Hajime Asama. Precise extraction of visual information from images by image processing techniques. In *International Symposium on Ultraprecision Engineering and Nanotechnology, Tokyo, Japan*, pages 37–42, 2013.

[YMK10] Atsushi Yamashita, Akiyoshi Matsui, and Toru Kaneko. Fence removal from multi-focus images. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 4532–4535. IEEE, 2010.

[YSL04] Liron Yatziv, Guillermo Sapiro, and Marc Levoy. Lightfield completion. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 3, pages 1787–1790. IEEE, 2004.

[YTKA12] Atsushi Yamashita, Fumiya Tsurumi, Toru Kaneko, and Hajime Asama. Automatic removal of foreground occluder from multi-focus images. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 5410–5416. IEEE, 2012.

[YWLH15] Jingyu Yang, Jun Wang, Leijie Liu, and Chunping Hou. Rifo: Restoring images with fence occlusions. In *MMSP*, pages 1–6, 2015.

[YWW+14] Jianjun Yang, Yin Wang, Honggang Wang, Kun Hua, Wei Wang, and Ju Shen. Automatic objects removal for scene completion. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 553–558. IEEE, 2014.

[YY09] Xiaoming Yuan and Junfeng Yang. Sparse and low-rank matrix decomposition via alternating direction methods. *preprint*, 12:2, 2009.

[Z+03] Jing Zhong et al. Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 44–50. IEEE, 2003.

[Zha13] Song Zhang. *Handbook of 3D machine vision: Optical metrology and imaging*. CRC press, 2013.

[ZK00] C Lawrence Zitnick and Takeo Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on pattern analysis and machine intelligence*, 22(7):675–684, 2000.

[ZLG+13] Xianhui Zheng, Yinghao Liao, Wei Guo, Xueyang Fu, and Xinghao Ding. Single-image-based rain and snow removal using multi-guided filter. In *International Conference on Neural Information Processing*, pages 258–265. Springer, 2013.

[ZW05] Lei Zhang and Xiaolin Wu. Color demosaicking via directional linear minimum mean square-error estimation. *IEEE Transactions on Image Processing*, 14(12):2167–2178, 2005.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France