



# Video inpainting and semi-supervised object removal

Thuc Trinh Le

## ► To cite this version:

Thuc Trinh Le. Video inpainting and semi-supervised object removal. Image Processing [eess.IV]. Université Paris Saclay (COMUE), 2019. English. NNT : 2019SACLT026 . tel-02382805

**HAL Id: tel-02382805**

**<https://pastel.hal.science/tel-02382805>**

Submitted on 27 Nov 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Video inpainting and semi-supervised object removal

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

Ecole doctorale n°580 Sciences et technologies de l'information et de la  
communication (L'ED STIC)  
Spécialité de doctorat : Signal et Image

Thèse présentée et soutenue à Paris, le 06/06/2019, par

**M. THUC TRINH LE**

## Composition du Jury :

Mme. Christine GUILLEMOT Directrice de Recherche, IRISA	Rapporteur
M. Guillermo SAPIRO Professeur, Duke University	Rapporteur
M. Gabriele FACCIOLO Professeur, Ecole Normale Supérieure Paris-Saclay	Examineur
M. Patrick PEREZ Directeur de R&D, valeo.ai	Examineur
M. David TSCHUMPERLE Chargé de Recherche CNRS, ENSICAEN	Examineur
M. Andrés ALMANSA Directeur de Recherche CNRS, Université Paris Descartes	Directeur de thèse
M. Yann GOUSSEAU Professeur, Télécom ParisTech	Co-directeur de thèse
M. Simon MASNOU Professeur, Université Claude-Bernard Lyon 1	Co-directeur de thèse
M. Alasdair NEWSON Maître de Conférences, Télécom ParisTech	Invité



**Titre :** Inpainting de vidéos et suppression d'objets semi-supervisée

**Mots clés :** édition vidéo, segmentation vidéo, inpainting vidéo, suppression d'objets

**Résumé :** De nos jours, l'augmentation rapide des vidéos crée une demande massive d'applications d'édition de vidéos. Dans cette thèse, nous nous concentrons sur l'application de suppression d'objets en vidéo. Pour mener à bien cette tâche, nous l'avons divisée en deux sous-problèmes: (1) une étape de segmentation des objets vidéo pour sélectionner les objets à supprimer et (2) une étape d'inpainting vidéo pour remplir les zones desocculées.

Pour le problème de la segmentation vidéo, nous concevons un système adapté aux applications de suppression d'objets et leurs exigences particulières en termes de précision et d'efficacité. Notre approche repose sur la combinaison de réseaux de neurones convolutifs (CNN) pour la segmentation et sur des méthodes classiques de suivi d'objets. Nous adoptons des réseaux de segmentation d'images et les appliquons au cas des vidéos en effectuant une segmentation image par image. En exploitant à la fois des apprentissages en ligne et hors ligne avec des annotations sur la première image seulement, les réseaux sont en mesure de produire une segmentation extrêmement précise des objets vidéo. En outre, nous proposons un module de suivi de masques pour assurer la continuité temporelle et un module de liaison de masques pour assurer la cohérence temporelle. De plus, nous présentons un moyen simple d'apprendre une dilatation optimale du masque, ce qui nous aide à créer des masques appropriés pour l'application de suppression d'objets vidéo.

Pour le problème d'inpainting vidéo, nous divisons notre travail en deux catégories : Pour un arrière-plan statique nous montrons que le problème peut être résolu efficacement par une technique simple de propagation basée sur le mouvement. Pour traiter le fond dynamique, l'inpainting est obtenu en optimisant une fonction d'énergie globale basée sur des patches. Pour accélérer l'algorithme, nous avons proposé une parallélisation de l'algorithme PatchMatch 3D. Pour améliorer la précision, nous intégrons systématiquement le flux optique dans le processus global. Le résultat est une méthode d'inpainting vidéo capable de reconstruire dans un temps raisonnable aussi bien des objets en mouvement que des textures dynamiques.

Enfin, nous combinons les méthodes de segmentation des objets vidéo et d'inpainting vidéo dans un système unifié pour supprimer les objets non souhaités dans les vidéos. A notre connaissance, il s'agit du premier système de ce type. Dans notre système, l'utilisateur n'a qu'à délimiter approximativement dans le premier cadre les objets à modifier. Ce processus d'annotation est facilité par l'aide de superpixels. Ces annotations sont ensuite affinées et propagées dans la vidéo par la méthode de segmentation. Un ou plusieurs objets peuvent ensuite être supprimés automatiquement à l'aide de nos méthodes d'inpainting vidéo. Il en résulte un outil informatique flexible pour le montage vidéo, avec de nombreuses applications potentielles, allant de la suppression de la foule à la correction de scènes non physiques.

**Title :** Video inpainting and semi-supervised object removal

**Keywords :** video editing, video segmentation, video inpainting, objects removal

**Abstract :** Nowadays, the rapid increase of video creates a massive demand for video editing applications. In this dissertation, we focus on the objects removal application in video. To complete this task, we divided it into two problems: (1) A video objects segmentation step to select which objects to remove and (2) a video inpainting step to filling the disoccluded regions.

For the video segmentation problem, we design a system which is suitable for object removal applications and their particular requirements in terms of accuracy and efficiency. Our approach relies on the combination of Convolutional Neural Networks (CNNs) for segmentation and more classical mask tracking methods. In particular, we employ image segmentation networks and apply them in a frame-by-frame basis. By exploiting both offline and online training with first frame annotation only, the networks are able to produce a highly accurate video object segmentation. Besides, we propose a mask tracking module to ensure temporal continuity and a mask linking module to ensure the identity coherence across frames. Moreover, we introduce a simple way to learn the dilation layer in the mask, which helps us create suitable masks for video objects removal.

For the video inpainting problem, we divide our work into two categories based on the type of background. In the static

background case we show that the problem can be solved efficiently using a simple motion-based propagation technique. To deal with dynamic background, inpainting is obtained by optimizing a global patch-based energy function. To increase the speed of the algorithm, we proposed a parallel extension of the 3D PatchMatch algorithm. To improve accuracy, we systematically incorporate the optical flow in the overall process. The resulting video inpainting method is able to reconstruct moving objects as well as to reproduce dynamic textures while running in a reasonable time.

Finally, we combine the video objects segmentation and video inpainting methods into a unified system to remove undesired objects in videos. To the best of our knowledge, this is the first system of this kind. In our system, the user only needs to approximately delimit in the first frame the objects to be edited. These annotation process is facilitated by the help of superpixels. Then, these annotations are refined and propagated through the video by the video objects segmentation method. One or several objects can then be removed automatically using our video inpainting methods. This results in a flexible computational video editing tool, with numerous potential applications, ranging from crowd suppression to unphysical scenes correction.







# 1

## Résumé en français

### Contents

<b>1.1</b>	<b>Segmentation d'objets vidéo</b>	<b>2</b>
<b>1.2</b>	<b>L'inpainting de vidéo</b>	<b>3</b>
<b>1.3</b>	<b>Suppression d'objets dans des vidéos complexes en quelques coups de pinceau</b>	<b>5</b>
<b>1.4</b>	<b>Organisation de la thèse</b>	<b>5</b>

Aujourd'hui, avec le développement de plusieurs nouveaux types de caméras, de plus en plus de vidéos de haute qualité sont produites chaque jour pour saisir tous les aspects de notre vie. De nos jours, les gens portent une caméra vidéo partout où ils vont et prennent des centaines d'heures de vidéos chaque année. Cela crée une demande massive d'applications de montage vidéo, telles que la sélection automatique des scènes, l'homogénéisation des couleurs, le montage par réflexion, etc.

Une tâche de montage vidéo particulièrement intéressante est la suppression d'objets. Imaginez que vous tournez une grande vidéo d'un ami devant un monument fantastique, mais plus tard, vous vous rendez compte que la vidéo est ruinée par des piétons qui passent au hasard. Le meilleur moment étant déjà passé, il est trop tard pour refaire le tournage, il ne vous reste plus qu'à retirer ces piétons de la vidéo. Cependant, les outils existants pour réaliser ce type de tâche, par exemple Content-Aware-Fill in Photoshop, prennent souvent beaucoup de temps et reposent sur une édition manuelle image par image. Ce n'est pas un bon choix car la cohérence temporelle n'est pas préservée, ce qui crée de forts artefacts lorsqu'une série

d'images est lue en séquence. Il s'avère que la suppression d'objets des vidéos est un problème extrêmement difficile à résoudre, et même les professionnels des effets visuels les résolvent à l'aide d'un montage manuel qui prend beaucoup de temps.

Motivé par ce problème, nous visons à créer un outil qui permet à un utilisateur de sélectionner les objets à supprimer dans la première trame, puis de laisser l'algorithme les supprimer de manière plausible. Deux questions principales se posent lors de la création d'un tel outil : (1) Comment pouvons-nous segmenter automatiquement ces objets à partir de la vidéo, avec seulement la première annotation d'image ? et (2) Une fois que ces objets indésirables ont été détectés, comment pouvons-nous remplir le trou restant dans la vidéo ?

Notre thèse tente de répondre à ces questions en considérant deux techniques de montage vidéo avancées : (i) la segmentation d'objets vidéo et (ii) la peinture vidéo (également connue sous le nom d'achèvement vidéo). La première tâche traite du problème de l'extraction d'objets multiples dans une vidéo, tandis que la deuxième tâche vise à combler les zones manquantes dans la vidéo.

## 1.1 Segmentation d'objets vidéo

La première partie de la thèse porte sur une méthode de segmentation d'objets vidéo pour l'extraction de masques d'objets. La segmentation d'objets vidéo est une tâche fondamentale en vision par ordinateur qui vise à séparer chaque image de la vidéo en deux ou plusieurs régions ; chaque région correspond à un objet spécifique dans la vidéo ou au fond. Dans cette thèse, nous nous concentrons sur la tâche d'extraire des objets multiples à partir d'annotations données dans le premier cadre. Dans la littérature, cette tâche est souvent appelée segmentation semi-supervisée d'objets vidéo. Plus précisément, compte tenu d'une première image avec des annotations délimitant les masques objets, le but est de segmenter précisément la même instance dans les images vidéo suivantes.

Ces dernières années, avec l'augmentation significative des données disponibles et un matériel informatique plus rapide, les méthodes axées sur les données, comme l'apprentissage approfondi, ont dominé le domaine de la vision par ordinateur. Elles sont devenues les techniques standard en matière de segmentation de l'image et de la vidéo. Alors que les réseaux de segmentation d'images basés sur l'apprentissage en profondeur obtiennent des résultats impressionnants ([Long, Shelhamer, and Darrell 2015](#); [Chen et al. 2016](#)), le problème de segmentation vidéo reste difficile en raison de la haute dimension des vidéos, des changements dans l'apparence des objets dans le temps et du manque de vidéos annotées pour la formation. Dans cette thèse, nous abordons ces problèmes en combinant des réseaux de segmentation basés sur l'apprentissage en profondeur avec des approches classiques de suivi de masque et de liaison de masque. Alors que les réseaux de segmentation basés sur l'apprentissage en profondeur permettent la formation avec des images statiques uniquement et produisent des résultats de séparation précis entre l'arrière-plan et les objets, les méthodes

de suivi utilisent des informations de localisation et suivent les objets pour assurer une cohérence temporelle. Enfin, les méthodes de liaison de masques permettent d'assurer la cohérence des identités d'objets dans le temps. Cette combinaison nous permet de segmenter différentes instances de la même classe sémantique et de gérer des situations difficiles telles que l'occlusion ou des objets se croisant. De plus, l'application de suppression d'objets spécifiques exige que le masque couvre tous les détails des objets, ce qui n'est généralement pas obtenu par un réseau de segmentation régulier. En d'autres termes, nous recherchons une approche de segmentation qui favorisera le rappel plutôt que la précision. Si ce n'est pas le cas, des artefacts gênants apparaîtront au moment de la suppression de l'objet. Nous abordons ce problème en introduisant une couche de dilatation intelligente qui apprend la zone de transition entre le fond et les objets.

En expérimentant avec différents ensembles de données, à la fois pour les cas de segmentation d'objets simples et multiples, nous montrons que notre méthode permet d'obtenir des résultats de segmentation vidéo de haute qualité, cohérents dans le temps et adaptés aux applications de suppression des objets.

## 1.2 L'inpainting de vidéo

La deuxième partie de la thèse traite du problème de l'inpainting vidéo. Dans le domaine de la vision par ordinateur et de l'infographie, l'inpainting vidéo fait référence à une technique permettant de combler les trous d'une vidéo en utilisant des informations spatiales et temporelles provenant de régions voisines. Les trous peuvent correspondre à des pièces manquantes ou à des objets retirés des scènes. L'objectif premier des approches de la peinture vidéo est de compléter ces trous afin que le résultat soit le plus réaliste possible par rapport au contexte connu, tant dans l'espace que dans le temps. Pour obtenir un résultat naturel, la cohérence spatiale et temporelle doit être conservée à travers la vidéo. Le problème est d'autant plus compliqué que la caméra et les objets de premier plan peuvent se déplacer, que l'arrière-plan peut être dynamique, que des interactions peuvent se produire entre les objets de premier plan, etc.

Dans le passé, de nombreux travaux ont été proposés pour résoudre des cas spécifiques d'incrustation vidéo tels que la restauration vidéo de vieux films, le retrait d'installations, le retrait de personnes des vidéos de surveillance pour la protection de la vie privée, le retrait d'un logo ou d'un filigrane placé sur une vidéo, la récupération de blocs vidéo perdus suite à une compression avec perte ou des erreurs de transmission image / vidéo. Cependant, ces approches supposent souvent des hypothèses fortes sur les vidéos d'entrée, telles qu'une caméra fixe, un fond statique, de petites occlusions ou des mouvements particuliers.

Plus spécifiquement, les premières approches de l'inpainting d'images ont été variationnel ([Masnou and Morel 1998](#)), ou PDE-based ([Bertalmio et al. 2000](#)) et consacrées à la préservation de la géométrie. Elles ont été suivies par des méthodes basées sur des

patches ([Drori, Cohen-Or, and Yeshurun 2003](#); [Criminisi, Pérez, and Toyama 2004](#)), héritées des méthodes de synthèse de texture ([Efros and Leung 1999](#)). Certaines de ces méthodes ont été adaptées aux vidéos, souvent en mélangeant des approches basées sur les pixels pour reconstruire l'arrière-plan et des stratégies gourmandes en patches pour les objets en mouvement ([Patwardhan, Sapiro, and Bertalmio 2005](#); [Patwardhan, Sapiro, and Bertalmio 2007](#)).

Une autre famille d'œuvres qui fonctionne très bien lorsque l'arrière-plan est statique repose sur la propagation des pixels par le mouvement. L'idée est d'abord de déduire un champ de mouvement à l'extérieur et à l'intérieur des régions manquantes. En utilisant le champ de mouvement complété, les valeurs de pixels provenant de l'extérieur de la région manquante sont ensuite propagées à l'intérieur de celle-ci. Par exemple, plusieurs méthodes tentent de restaurer le champ de mouvement à l'intérieur de ces régions manquantes en propageant progressivement les vecteurs de mouvement ([Matsushita et al. 2006](#)), en échantillonnant les patches de mouvement spatio-temporels ([Shiratori et al. 2006](#); [Tang et al. 2011](#)), ou en interpolant le mouvement manquant ([You et al. 2013](#); [Bokov and Vatolin 2018](#)).

Récemment, différentes approches ont été proposées pour résoudre des situations plus complexes. Parmi eux, ([Newson et al. 2014](#)) étend le travail de ([Wexler, Shechtman, and Irani 2007](#)). Les vidéos sont complétées par l'optimisation d'une énergie d'optimisation globale, basée sur des patches. Plus précisément, ce travail introduit une extension spatio-temporelle à l'algorithme PatchMatch pour accélérer le problème de recherche de patch, utilise une pyramide de texture multi-résolution pour améliorer la préservation des détails et estime le mouvement de fond en utilisant un modèle affine.

A partir de cette littérature, nous proposerons dans cette thèse deux moyens complémentaires pour réaliser l'étape d'inpainting nécessaire à l'élimination des objets dans les vidéos.

Une première méthode est rapide et repose sur l'achèvement image par image du flux optique, suivi de la propagation des valeurs de voxel, inspirée de la méthode récemment introduite ([Bokov and Vatolin 2018](#)), elle-même partageant des idées avec l'approche de ([Huang et al. 2016](#)) et permettant des gains impressionnants en termes de temps de calcul. De telles approches sont efficaces sur le plan informatique, mais ne sont pas en mesure de traiter les fonds mouvants et les textures dynamiques.

Pour ces cas complexes, nous nous appuyons sur une seconde approche plus sophistiquée qui améliore la stratégie globale de ([Newson et al. 2014](#)) en termes de précision et de rapidité. En particulier, nous apportons une amélioration significative à la méthode de ([Newson et al. 2014](#)) dans la reconstruction des objets en mouvement par l'introduction de termes de flux optique. Sur la base de ces termes, un nouveau schéma d'initialisation, une distance de patch modifiée, une stratégie de recherche de patch guidée par flux optique et une carte de séparation sont proposés. Nous atteignons également l'objectif de réduire le temps de calcul en parallélisant l'algorithme. En expérimentant et en comparant le résultat avec d'autres

résultats de pointe, nous montrons que notre méthode a la capacité de préserver la cohérence spatio-temporelle sur fond dynamique ainsi que de reconstruire des objets en mouvement dans une longue occlusion temporelle.

### **1.3 Suppression d’objets dans des vidéos complexes en quelques coups de pinceau**

Après avoir résolu le problème de la segmentation des objets vidéo et de la peinture vidéo, nous les combinons en un système complet pour la suppression des objets des vidéos. En entrée, le système n’a besoin que de quelques coups de pinceau sur le premier cadre, délimitant grossièrement les objets à enlever. Ces objets sont ensuite découpés et les trous sont remplis automatiquement. Les étapes clés de notre système sont les suivantes : après l’initialisation, les masques sont d’abord affinés et ensuite automatiquement propagés à travers la vidéo par notre algorithme de segmentation vidéo, puis les régions manquantes sont synthétisées en utilisant des techniques de peinture vidéo. Notre système peut traiter des objets multiples se croisant éventuellement avec des mouvements complexes, ainsi que des textures dynamiques. De plus, il permet également à l’utilisateur de corriger les erreurs, ce qui permet un mode d’interaction flexible. Il en résulte un outil de calcul qui permet d’alléger les opérations manuelles fastidieuses d’édition de vidéos de haute qualité.

Nous évaluons l’ensemble du pipeline de notre système à l’aide de séquences classiques ainsi que de plusieurs nouvelles situations difficiles et réalistes. Ces expériences montrent que notre système d’enlèvement d’objets surpasse les travaux précédents qui utilisent des masques sélectionnés manuellement et est utile dans des situations difficiles de la vie réelle.

### **1.4 Organisation de la thèse**

Ce mémoire commence par un chapitre préalable, Chapitre 2, dans lequel nous introduisons quelques notions, définitions et quelques notions communes dans les domaines de l’apprentissage profond, de la segmentation et de la peinture vidéo. Plus spécifiquement, ce chapitre comprend une introduction rapide sur l’apprentissage profond et les réseaux neuronaux convolutifs avec un accent particulier sur les réseaux de segmentation d’images. Il présente également en détail chaque bloc et les architectures de réseaux qui sont touchés dans notre méthode de segmentation. Pour l’inpainting vidéo, ce chapitre donne quelques notations, la formulation du problème, et un bref détour sur l’estimation du mouvement avec un accent sur le calcul du flux optique basé sur l’apprentissage profond.

Ensuite, le reste de la thèse est divisé en trois parties présentant nos contributions à l’application de suppression d’objets en vidéo. Dans chaque partie, nous commençons par un examen attentif des différentes approches de pointe, puis nous présentons notre méthode proposée et fournissons des résultats expérimentaux.

**Segmentation vidéo** Dans le chapitre 3, nous présentons la première partie de la thèse, qui est une méthode de segmentation d’objets vidéo d’une manière semi supervisée. Nos méthodes de segmentation d’objets multiples permettent non seulement de distinguer l’arrière-plan et l’avant-plan, mais aussi de séparer différentes instances d’objets dans des conditions difficiles. Par comparaison qualitative et quantitative avec les travaux précédents, nous montrons que notre méthode permet d’obtenir des performances compétitives sur différents ensembles de données.

La deuxième partie de la thèse aborde le problème de la reconstruction des régions endommagées dans une vidéo. Dans cette partie, nous divisons notre travail en deux catégories selon le type de contexte.

**Vidéo inpainting avec fond statique** Dans le chapitre 4, nous présentons une méthode simple de propagation de pixels guidée par le mouvement pour traiter les cas de fond statique. Nous montrons que le problème de la suppression d’objets avec un fond statique peut être résolu efficacement en utilisant une technique simple basée sur le mouvement.

**Vidéo inpainting with complex motions and dynamic textures** Dans le chapitre 5, nous présentons une méthode d’inpainting vidéo s’appuyant sur l’optimisation d’une fonction énergétique globale basée sur des patches pour traiter les fonds dynamiques. Nous nous appuyons sur le travail de ([Newson et al. 2014](#)) et l’améliorons en vitesse et en précision. Pour augmenter la vitesse de l’algorithme, nous avons proposé une extension parallèle de l’algorithme 3D PatchMatch. Pour améliorer la précision, nous intégrons systématiquement le flux optique dans l’ensemble du processus. De cette façon, nous pouvons reconstruire des objets en mouvement et reproduire des textures dynamiques avec une grande cohérence temporelle.

**Suppression d’objet** Dans la troisième partie de la thèse, Chapitre 6, nous combinons nos méthodes de segmentation d’objets et de peinture vidéo dans un système unifié pour supprimer les objets des vidéos avec seulement quelques traits des utilisateurs.

Enfin, le chapitre 7 conclut sur les contributions de la thèse et présente quelques perspectives.





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Video objects segmentation . . . . .	2
1.2	Video inpainting . . . . .	3
1.3	Objects removal in complex videos from a few strokes . . . . .	4
1.4	Thesis organization . . . . .	5
1.5	Publications . . . . .	6
1.6	Supplementary websites . . . . .	6
<b>2</b>	<b>Prerequisites</b>	<b>7</b>
2.1	Video objects segmentation . . . . .	8
2.1.1	Convolutional Neural Networks . . . . .	8
2.1.2	Segmentation Networks . . . . .	21
2.1.3	Evaluation metrics . . . . .	28
2.2	Inpainting . . . . .	29
2.2.1	Definition and notation . . . . .	29
2.2.2	Motion estimation . . . . .	32
<b>3</b>	<b>Video Objects Segmentation</b>	<b>39</b>
3.1	Introduction . . . . .	40
3.1.1	What is video objects segmentation? . . . . .	40
3.1.2	Applications . . . . .	42
3.1.3	Challenges . . . . .	43
3.1.4	Video segmentation dataset . . . . .	44
3.2	Literature reviews . . . . .	45
3.2.1	Classical video segmentation . . . . .	47
3.2.2	Deep learning-based objects segmentation. . . . .	54
3.3	Method . . . . .	69
3.3.1	Choosing the best method for video objects segmentation . . . . .	69
3.3.2	Overview . . . . .	71
3.3.3	Segmentation networks . . . . .	72
3.3.4	Multiple object masks tracking . . . . .	79

3.3.5	Masks linking . . . . .	83
3.3.6	Post-processing . . . . .	86
3.4	Experimental Results . . . . .	87
3.4.1	Implementation details . . . . .	87
3.4.2	Evaluation metrics . . . . .	88
3.4.3	Evaluation . . . . .	89
3.5	Concluding remarks . . . . .	99
3.5.1	Limitations . . . . .	99
3.5.2	Conclusions and future works . . . . .	100
<b>4</b>	<b>Objects removal with a static background</b>	<b>101</b>
4.1	Introduction to image/video inpainting . . . . .	102
4.1.1	Overview . . . . .	102
4.1.2	Applications . . . . .	103
4.1.3	Challenges . . . . .	104
4.1.4	Video objects removal with static background . . . . .	105
4.2	Related works . . . . .	106
4.3	Proposed method . . . . .	108
4.3.1	Motion field reconstruction . . . . .	109
4.3.2	Motion-guided pixel reconstruction . . . . .	110
4.3.3	Poisson blending . . . . .	111
4.4	Results . . . . .	113
4.4.1	Implementation details . . . . .	113
4.4.2	Evaluation . . . . .	114
4.5	Conclusion . . . . .	120
<b>5</b>	<b>Video inpainting with a complex background</b>	<b>121</b>
5.1	Introduction . . . . .	122
5.2	State-of-the-art in inpainting . . . . .	123
5.2.1	Image inpainting . . . . .	124
5.2.2	Video inpainting . . . . .	137
5.3	Proposed method . . . . .	143
5.3.1	Pre-processing . . . . .	144
5.3.2	Energy . . . . .	146
5.3.3	Coarse initialization . . . . .	150
5.3.4	Source patch searching . . . . .	151
5.3.5	Pixel reconstruction . . . . .	155
5.4	Result . . . . .	156
5.4.1	Implementation details . . . . .	156
5.4.2	Experimental results . . . . .	157

5.4.3	Contribution of each component . . . . .	170
5.5	Limitations . . . . .	176
5.6	Conclusion and Future Works . . . . .	177
<b>6</b>	<b>A complete system for objects removal in videos</b>	<b>179</b>
6.1	Introduction . . . . .	180
6.2	Proposed method . . . . .	181
6.2.1	Overview . . . . .	181
6.2.2	First frame annotation . . . . .	182
6.2.3	Objects segmentation . . . . .	184
6.2.4	Objects removal . . . . .	184
6.3	Experimental results . . . . .	186
6.3.1	Comparison with state-of-the-arts objects removal methods . . . . .	186
6.3.2	Results with our new challenging sequences . . . . .	192
6.3.3	Application in real-life situations . . . . .	194
6.3.4	Impact of the segmentation masks on the inpainting performances . . . . .	196
6.3.5	Some failure cases . . . . .	197
6.4	Conclusion . . . . .	199
<b>7</b>	<b>Conclusion, limitations and future works</b>	<b>201</b>
7.1	Conclusion . . . . .	201
7.2	Limitations and perspectives . . . . .	203



# 1

## Introduction

### Contents

---

<b>1.1</b>	<b>Video objects segmentation</b>	<b>2</b>
<b>1.2</b>	<b>Video inpainting</b>	<b>3</b>
<b>1.3</b>	<b>Objects removal in complex videos from a few strokes</b>	<b>4</b>
<b>1.4</b>	<b>Thesis organization</b>	<b>5</b>
<b>1.5</b>	<b>Publications</b>	<b>6</b>
<b>1.6</b>	<b>Supplementary websites</b>	<b>6</b>

---

Nowadays, with the development of several new types of cameras, more and more high-quality videos are being produced every day to capture every aspects of our lives. People nowadays carry a video camera with them wherever they go and take hundreds of hours of videos every year. This creates a massive demand for video-based editing applications, such as automatic selection of scenes, color homogenization, reflectance editing, etc.

A particularly interesting video editing task is object removal. Imagine you shoot a great video of a friend in front of a fantastic monument, but later, you realize that the video is ruined by random pedestrians passing by. Since the best moment has already passed, it is too late to re-shoot, all you want to do is to remove these pedestrians out of the video. However, existing tools to achieve this kind of task, e.g., Content-Aware-Fill in Photoshop, are often time-consuming and rely on frame-by-frame manual editing. This is not a good choice because the temporal consistency is not preserved, which creates strong artifacts when a series of images is played as a sequence. It turns out that removing objects from videos is an extremely

challenging problem, and even visual effects professionals solve them with time-consuming manual editing.

Motivated by this problem, we aim at creating a tool that enables a user to select the objects to remove in the first frame, and then let the algorithm suppress them in a plausible way. Two main questions arise when creating such a tool: (1) How can we automatically segment these objects from the video, given only the first frame annotation? and (2) Once these undesired objects have been detected, how can we fill in the remaining hole in the video?

Our thesis tries to answer these questions by considering two main advanced video editing techniques: (i) video objects segmentation and (ii) video inpainting (also known as video completion). The first task addresses the problem of extracting multiple objects in a video, while the second task aims at filling-in the missing regions in the video.

## 1.1 Video objects segmentation

The first part of the thesis focuses on a video object segmentation method for extracting object masks. Video object segmentation is a fundamental task in computer vision which aims at separating each frame in the video into two or more regions; each region corresponds to a specific object in the video or to the background. In this thesis, we concentrate on the task of extracting multiple objects from given annotations in the first frame. In the literature, this task is frequently referred to as semi-supervised video objects segmentation. More specifically, given a first frame with annotations delimiting the object masks, the goal is to accurately segment the same instance in the next video frames.

In recent years, with the significant increase of the available data and faster computing hardware, data-driven methods such as deep learning have dominated the field of computer vision. They became the standard techniques in image and video segmentation. While deep learning-based image segmentation networks achieve impressive results ([Long, Shelhamer, and Darrell 2015](#); [L.-C. Chen, Papandreou, Kokkinos, et al. 2016](#)), the video segmentation problem remains challenging because of the high dimension of the videos, the changes of object appearances over time and the lack of annotated videos for training. In this thesis, we address these problems by combining deep learning-based segmentation networks with classical mask tracking and mask linking approaches. While deep learning-based segmentation networks allow training with static images only and produce accurate separation results between background and objects, tracking methods use location information and track the objects to ensure temporal consistency. Eventually, mask linking methods enable the consistency of object identities across time. This combination allows us to segment different instances of the same semantic class and handles difficult situations such as occlusion or objects crossing each other. Moreover, the specific objects removal application requires that the mask must cover all details of the objects, which is usually not obtained by a regular segmentation network. In other words, we seek for a segmentation approach that will favor recall instead of precision. If

this is not the case, annoying artifacts will appear at object removal time. We address this issue by introducing a smart dilation layer which learns the transition zone between background and objects.

By experimenting with different datasets, both for the cases of single and multiples object segmentation, we show that our method achieves high-quality temporally coherent video segmentation results which are suitable for objects removal application.

## 1.2 Video inpainting

The second part of the thesis deals with the problem of video inpainting. In the field of computer vision and computer graphic, video inpainting refers to a technique to fill-in holes in a video using spatial and temporal information from neighboring regions. The holes may correspond to missing parts or to removed objects from the scenes. The primary objective of video inpainting approaches is to complete these holes so that the result is as realistic as possible in relation to the known context, both in space and time. To obtain a natural result, the spatial and temporal coherency must be kept through the video. The problem is further complicated because both the camera and foreground objects may move, the background may be dynamic, interactions may happen between foreground objects, etc.

In the past, many works have been proposed to solve specific cases of video inpainting such as video restoration of old films, rig removal, removing people from surveillance videos for privacy protection, removing a logo or watermark placed on a video, recovering lost video blocks due to lossy compression or image/video transmission errors. However, these approaches often assume strong hypothesis on the input videos, such as a fixed camera, a static background, small occlusions or particular motions.

More specifically, the first approaches in image inpainting were variational ([Masnou and Morel 1998](#)), or PDE-based ([Bertalmio, Sapiro, et al. 2000a](#)) and dedicated to the preservation of geometry. They were followed by patch-based methods ([Drori, Cohen-Or, and Yeshurun 2003](#); [Criminisi, Pérez, and Toyama 2004](#)), inherited from texture synthesis methods ([Efros and Leung 1999](#)). Some of these methods have been adapted to videos, often by mixing pixel-based approaches for reconstructing the background and greedy patch-based strategies for moving objects ([Patwardhan, Sapiro, and Bertalmio 2005](#); [Patwardhan, Sapiro, and Bertalmio 2007](#)).

Another family of works which performs very well when the background is static relies on motion-based pixel propagation. The idea is to first infer a motion field outside and inside the missing regions. Using the completed motion field, pixel values from outside the missing region are then propagated inside it. For example, several methods try to restore the motion field inside these missing regions by gradually propagating motion vectors ([Matsushita, Ofek, Ge, et al. 2006](#)), by sampling spatial-temporal motion patches ([Shiratori et al. 2006](#); [N. C. Tang et al. 2011](#)), or by interpolating the missing motion ([S. You et al. 2013](#); [Bokov and Vatolin](#)

2018).

Recently, different approaches have been proposed to solve more complex situations. Among them, (Newson, Almansa, Fradet, et al. 2014) extend the work of (Wexler, Shechtman, and Irani 2007). Videos are completed by optimizing a global, patch-based optimization energy. Specifically, this work introduces a spatiotemporal extension to the PatchMatch algorithm to accelerate the patch search problem, makes use of a multi-resolution texture feature pyramid to improve the preservation of details and estimates background motion using an affine model.

Drawing on this literature, we will propose in this thesis two complementary ways to perform the inpainting step needed to remove objects in videos.

A first method is fast and relies on a frame-by-frame completion of the optical flow, followed by voxel values propagation, inspired by the recently introduced method (Bokov and Vatolin 2018), itself sharing ideas with the approach from (J.-B. Huang et al. 2016) and yielding impressive gains in terms of computational times. Such approaches are computationally efficient but are not able to deal with moving backgrounds and dynamic textures.

For these complex cases, we rely on a more sophisticated second approach which improves the global strategy of (Newson, Almansa, Fradet, et al. 2014) in terms of both accuracy and speed. In particular, we make a significant improvement to the method of (Newson, Almansa, Fradet, et al. 2014) in moving objects reconstruction by the introduction of optical flow terms. Based on these terms, a novel initialization scheme, a modified patch distance, an optical flow-guided patch searching strategy, and a separation map are proposed. We also attain the goal of reducing the computation time by parallelizing the algorithm. By experimenting and comparing the result with other state-of-the-art results, we show that our method has the capability of preserving the spatio-temporal coherency under dynamic background as well as reconstructing moving objects within a long temporal occlusion.

### 1.3 Objects removal in complex videos from a few strokes

After solving the problem of video objects segmentation and video inpainting, we combine them into a complete system for the removal of objects from videos. As an input, the system only needs a user to draw a few strokes on the first frame, roughly delimiting the objects to be removed. These objects are then cut out and the holes are filled automatically. The key steps of our system are the following: after initialization, the masks are first refined and then automatically propagated through the video by our video segmentation algorithm, then the missing regions are synthesized using video inpainting techniques. Our system can deal with multiple objects possibly crossing with complex motions, as well as with dynamic textures. Moreover, it also allows the user to correct errors, which enables a flexible interaction mode. This results in a computational tool that can alleviate tedious manual operations for editing high-quality videos.



We evaluate the whole pipeline of our system using classical sequences as well as several new challenging and realistic situations. These experiments show that our object removal system outperforms previous works which use manually selected masks and is helpful in real-life difficult situations.

## 1.4 Thesis organization

This dissertation starts with a prerequisite chapter, Chapter 2, in which we introduce some notions, definitions and some common background in the domains of deep learning, segmentation, and video inpainting. More specifically, this chapter includes a quick introduction about deep learning and convolutional neural network with a focus on image segmentation networks. It also introduces in details each block and networks architectures which are touched in our segmentation method. For video inpainting, this chapter gives some notations, problem formulation, and a brief detour on motion estimation with a focus on deep learning-based optical flow calculation.

Then, the rest of the dissertation is divided in three parts presenting our contributions for objects removal application in videos. In each part, we start with a careful review of various state-of-the-art approaches, then we present our proposed method and provide experimental results.

**Video segmentation** In Chapter 3, we present the first part of the thesis, which is a video objects segmentation method in a semi-supervised manner. Our multiple objects segmentation methods not only can distinguish background/foreground but also can separate different object instances under challenging conditions. By qualitative and quantitative comparison to previous works, we show that our method achieves competitive performance over different datasets.

The second part of the thesis addresses the problem of reconstructing damaged regions in a video. In this part, we divide our work into two categories based on the type of background.

**Video inpainting with static background** In chapter 4, we present a simple motion-guided pixel propagation method to deal with static background cases. We show that the problem of objects removal with a static background can be solved efficiently using a simple motion-based technique.

**Video inpainting with complex motions and dynamic textures** In chapter 5, we introduce a video inpainting method relying on the optimization a global patch-based energy function to deal with dynamic backgrounds. We build on the work of (Newson, Almansa, Fradet, et al. 2014) and improve it in both speed and accuracy. To increase the speed of the algorithm, we proposed a parallel extension of the 3D PatchMatch algorithm. To improve accuracy,

we systematically incorporate the optical flow in the overall process. By this way, we can reconstruct moving objects and reproduce dynamic textures with high temporal consistency.

**Object removal** In the third part of the thesis, Chapter 6, we combine our objects segmentation and video inpainting methods into one unified system to remove objects in videos with only a few strokes from the users.

Finally, Chapter 7 concludes on the contributions of the dissertation and discusses some perspectives.

## 1.5 Publications

The works contained in this thesis have led to the following publication or submissions:

1. Le, T. T., Almansa, A., Gousseau, Y., & Masnou, S. (2019, March) **Removing objects from videos with a few strokes**. Submitted to *Computer Visual Media Journal (CVMJ)*.
2. Le, T. T., Almansa, A., Gousseau, Y., & Masnou, S. (2018, December). **Removing objects from videos with a few strokes**. In *SIGGRAPH Asia 2018 Technical Briefs* (p. 22). ACM.
3. Le, T. T., Almansa, A., Gousseau, Y., & Masnou, S. (2017, September). **Motion-consistent video inpainting**. In *2017 IEEE International Conference on Image Processing (ICIP)* (pp. 2094-2098). IEEE.
4. Th. T. Le, A. Almansa, Y. Gousseau et S. Masnou, **Inpainting vidéo préservant le mouvement**, *GRETSI XXVIème Colloque*, Août 2017.

## 1.6 Supplementary websites

This thesis is devoted to video processing and editing methods, whose results can of course not fully be evaluated from still illustrations. Therefore, two websites are provided to illustrate the videos result of our algorithms. These websites may be found with the following links.

- Motion-consistent video inpainting:  
[https://purl.org/vid\\_inp\\_motion](https://purl.org/vid_inp_motion)
- Removing objects from videos with a few strokes  
<https://object-removal.telecom-paristech.fr>

# 2

## Prerequisites

### Contents

---

<b>2.1</b>	<b>Video objects segmentation</b>	<b>8</b>
2.1.1	Convolutional Neural Networks	8
	Building blocks	8
	Properties of CNNs	11
	Common CNN architectures	12
	Learning	15
2.1.2	Segmentation Networks	21
	Transfer learning	21
	Fully Convolutional Networks (FCNs)	22
	Encoder-Decoder architecture	23
	Deeplab architecture	24
2.1.3	Evaluation metrics	28
<b>2.2</b>	<b>Inpainting</b>	<b>29</b>
2.2.1	Definition and notation	29
2.2.2	Motion estimation	32

---

To better understand our works, this chapter gives an overview of concepts that are touched in this dissertation. More specifically, the first part of this chapter, Section 2.1, focuses on deep

learning methods for video objects segmentation. This section starts with a brief overview of convolutional neural networks, the backbone of our method, and then goes deeper into different ways to use deep learning for the segmentation task. The second part of this chapter, Section 2.2, gathers some notions and definitions related to the inpainting problem. In this chapter, we do not aim to provide details about the available approaches for each task. For a more thorough description of the related works, we refer the reader to the state-of-the-art section of each chapter.

## 2.1 Video objects segmentation

Figure 2.1 shows the evolution of the applications of deep learning to image understanding tasks, from image classification up to instance segmentation. The original task addressed by deep learning architectures is classification which aims to classify an image into different categories. Detection is the next step, providing both classes and the spatial location of those classes. At the next step is segmentation, which predicts labels for every pixel to separate different object or region. Further progress has brought deep learning to video. In this section, several segmentation techniques using deep learning are presented to understand how these modern techniques can bring advantages to the video editing domain.

To start, we give a brief review of visual representation learning with deep networks, especially convolutional neural networks which are the core network architectures of our method. Next, we explain how to adapt these networks to the segmentation task. We also explain in more details each component on which our segmentation network will be built upon.

### 2.1.1 Convolutional Neural Networks

Convolutional Neural Networks (LeCun et al. 1998) (CNNs) are special cases of standard neural networks which are designed for processing data that has a grid-like topology (e.g., images). Contrary to standard approaches for vision problems based on hand-crafted features, CNNs-based methods aim at learning the features required to the task at hand. They have two key properties: spatially shared weights and spatial pooling. The former property makes them learn features that are shift-invariant while the latter is responsible for reducing the sensitivity and enlarge the receptive field. These two properties make CNNs extremely useful for image applications.

#### Building blocks

A Convolutional Neural Network is a stack of layers which may or may not have parameters. A layer receives 3D volume (height, width, channels) as input and it outputs a 3D volume with some differentiable function. An example of a CNN for image classification is demonstrated

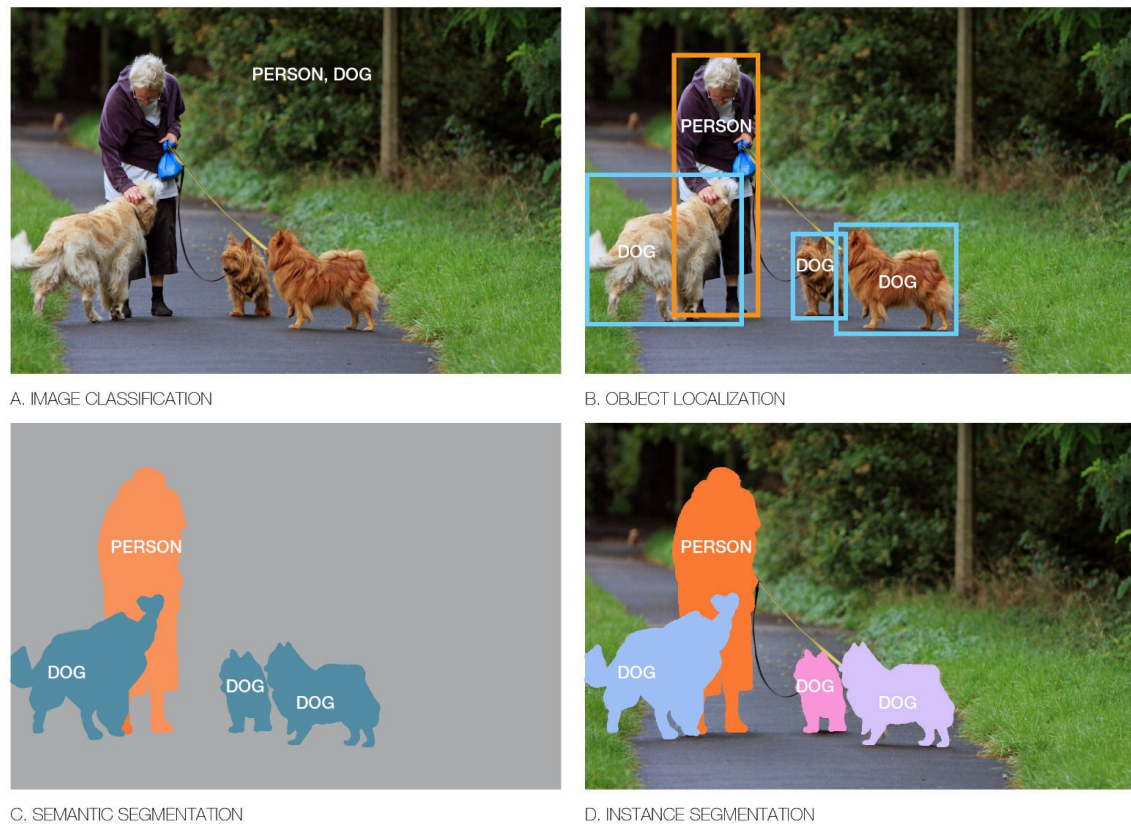


Figure 2.1 – The evolution of deep learning applications to image understanding, from image classification (a) to object detection (b), then semantic segmentation (c) and instance segmentation (d).

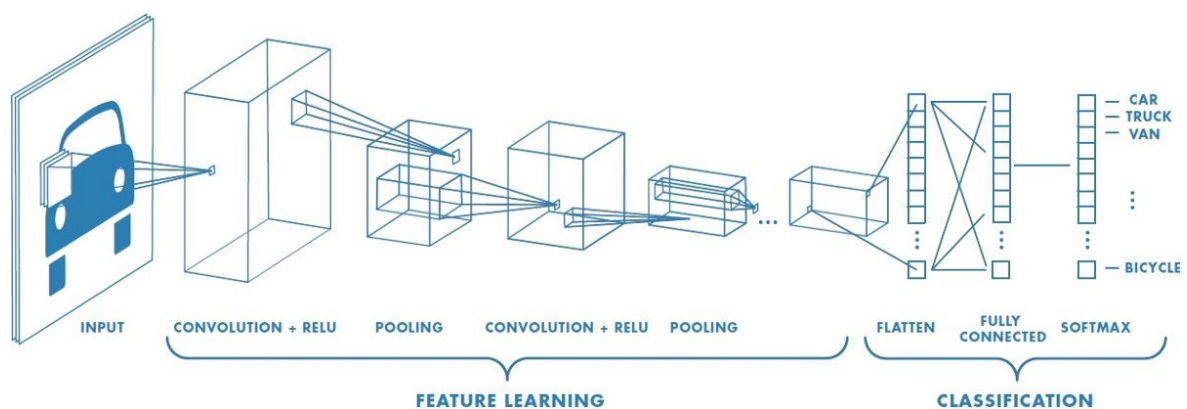


Figure 2.2 – An example architecture of a CNN for image classification. Source [MATLAB Tech Talk](#).

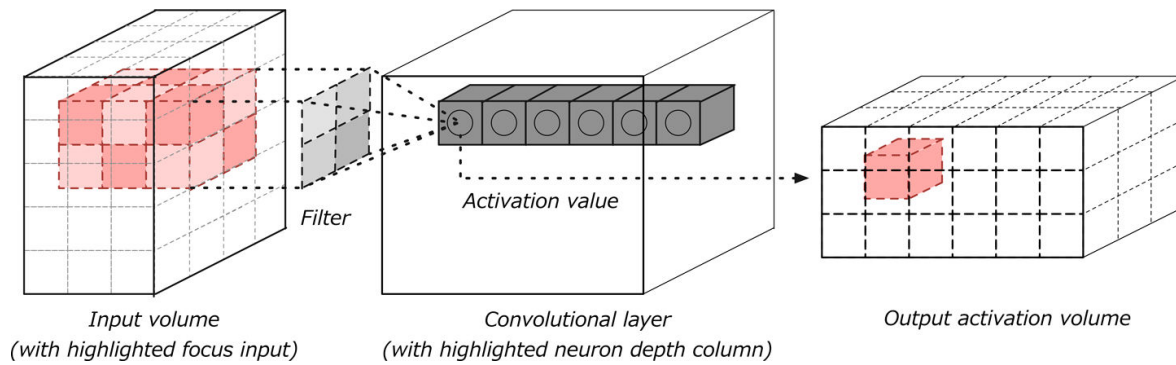


Figure 2.3 – Convolutional layer with input and output volumes. Source: [Oreilly's library](#).

in 2.2. Typically, there are three main types of layers: **Convolutional layer**, **Pooling layer** and **Fully-Connected** or **Linear layer**.

- **Convolutional layer (Conv):** This is the core building block of CNNs's architecture which transform the input volume by using 2D convolutions. As Figure 2.3 illustrates, the 2D convolution is a simple operation: starting with an array of weights called *kernel (filter)* with a size smaller than the size of the input, this kernel slides over the input volume, performing dot product with the region of the input it is currently on to produces a two-dimensional output called a *feature map*. For example, let us assume the input is a  $(32 \times 32 \times 3)$  array of pixel values and the kernel size is  $(5 \times 5 \times 3)$ . Start with the top-left corner, we stride this kernel pixel by pixel, from left to right and top to bottom. When the kernel strides in a sliding window manner, it is convolved with the original pixel values of the corresponding region of the image and returns a single output number. As a result, every unique location on the input volume produces a number. After sliding the kernel over all the locations, we obtain an output  $(28 \times 28)$  feature map, corresponding to our kernel. In practice, more kernels are used. The feature maps for each kernel are stacked together to build the 3D output volume. *Conv* layers have parameters that include **Kernel size**, **Output depth**, **Stride** and **Padding**.

- **Kernel size:** In theory, an array of any size can be a kernel. However, in practice, the kernel size is often small spatially with respect to the size of the input to reduce the number of learning parameters. Because kernels are also applied for every depth of the input volume, the depth of the kernel must be equal to the depth of its input.
- **Output depth:** The depth of the output volume is the number of kernels we apply to the input volume, which specifies how many neurons in the convolutional layer that connect to the same region of the input volume. Each kernel acts as a filter to encapsulate different aspects of the input volume.
- **Stride:** Stride defines the spatial offset, in pixels, at one sliding time. A low stride will result in larger output volumes because it yields more overlapping receptive

fields. The higher the stride is, the smaller output volumes we get because we are limiting the overlap of two subsequent dot products in the convolution operation.

- **Padding:** Padding means adding specific values to the border of the input volume to adjust the spatial size of the output. Without padding, the input size will get reduced. Together with stride, padding has impacts on the input size and helps us control the output size to fit in some specific task.
- **Pooling layer (Pool):** *Pooling layer* performs a downsampling operation, such as the *max* operation (referred to as *Max Pooling*), along the spatial dimensions. A *Max Pooling* with a stride of 2 is often used in practice which means that if the spatial size of the input is  $(32 \times 32)$ , the output volume would be  $(16 \times 16)$  spatially. *Pooling layer* is usually placed between *Convolutional Layers* and this layer is responsible for reducing the spatial dimension of each input feature map. This layer helps the network increase the reception fields while going deeper. It also makes the model robust to small variations in the input and helps to control overfitting.
- **Fully Connected layer (FC):** In this layer, every neuron of the input volume is connected to the output. In image classification applications, a *Fully Connected layer* is often placed at the end of the network to compute class scores and predict the label of an image. It takes an input volume and outputs one  $N$  dimensional vector, where  $N$  is the number of output classes we are evaluating.
- **Rectified Linear Units (ReLU):** Besides the three main layers above, non-linear activation functions must also be applied to obtain non-linearity property. A typical CNN uses *ReLU* as the activation function. *ReLU* will apply an element-wise activation function equals to the  $\max(0, x)$  thresholding at zero which leaves the size of the input volume unchanged.

With these basic layers, a typical simple network may look like this:

*Input*  $\rightarrow$  *Conv*  $\rightarrow$  *ReLU*  $\rightarrow$  *Pool*  $\rightarrow$  *Conv*  $\rightarrow$  *ReLU*  $\rightarrow$  *Conv*  $\rightarrow$  *ReLU*  $\rightarrow$  *Pool*  $\rightarrow$  *FC*

### Properties of CNNs

CNNs have the following distinguishing properties:

- **Parameters sharing:** Imagine we have an image of a dog, and we want the network to recognize the dog. If the network learns about the dog in the top left corner or the dog in the right corner independently, then a load of work needed would be too heavy. This is a problem of shift variance, which is common in image data. CNNs use a parameter-sharing scheme to solve this problem. It allows the network to detect features/objects even if it does not look exactly like the images in its training set. It also helps the training process because fewer parameters are needed.





Figure 2.4 – Visualization of example filters learned by AlexNet ([Krizhevsky, Sutskever, and Hinton 2012](#)). Source [Oreilly's library](#).

- Features representation:** With the parameter-sharing scheme, CNNs can learn features which are invariant to spatial position. This allows the neural network to learn an overall representation which is not local to any particular set of features. Figure 2.4 presents several of the 96 filters of size  $(11 \times 11 \times 3)$  learned by Alexnet ([Krizhevsky, Sutskever, and Hinton 2012](#)). It can be seen that the network has different filters to detect different local structures of the image, such as a horizontal edge, color blob, etc. This means that we can learn the features in one place and not worry about learning it as a feature in all positions in the image. Another property of the features of CNNs is that in the very first layers, the network tries to learn some small structure such as edge or color blob... When we go deeper, the layers towards the end of the network have larger receptive field sizes and learn more extended features, and the network tends to learn more conceptual information about the image.

### Common CNN architectures

Since the breakthrough in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) ([Russakovsky et al. 2015](#)) achieved by AlexNet ([Krizhevsky, Sutskever, and Hinton 2012](#)), several other CNNs have been introduced to gain better performance. While in classic network architectures, *Convolutional layers*, *Pooling layers*, and *ReLU layers* are stacked merely together with a *Fully Connected layer* at the end, modern architectures explore new and innovative ways for constructing and arranging these layers in a way which allows for more efficient learning. Almost all of these architectures have a base unit, and then it is repeated multiple times throughout the network. In the beginning, these architectures are only designed for image classification purposes. However, since they can extract rich features, they can be adapted to solve various computer vision tasks such as object detection ([S. Ren et al. 2015](#)), image segmentation ([Long, Shelhamer, and Darrell 2015](#)), and many other more advanced tasks. In this section, we discuss two common architectures which will be used in our thesis,



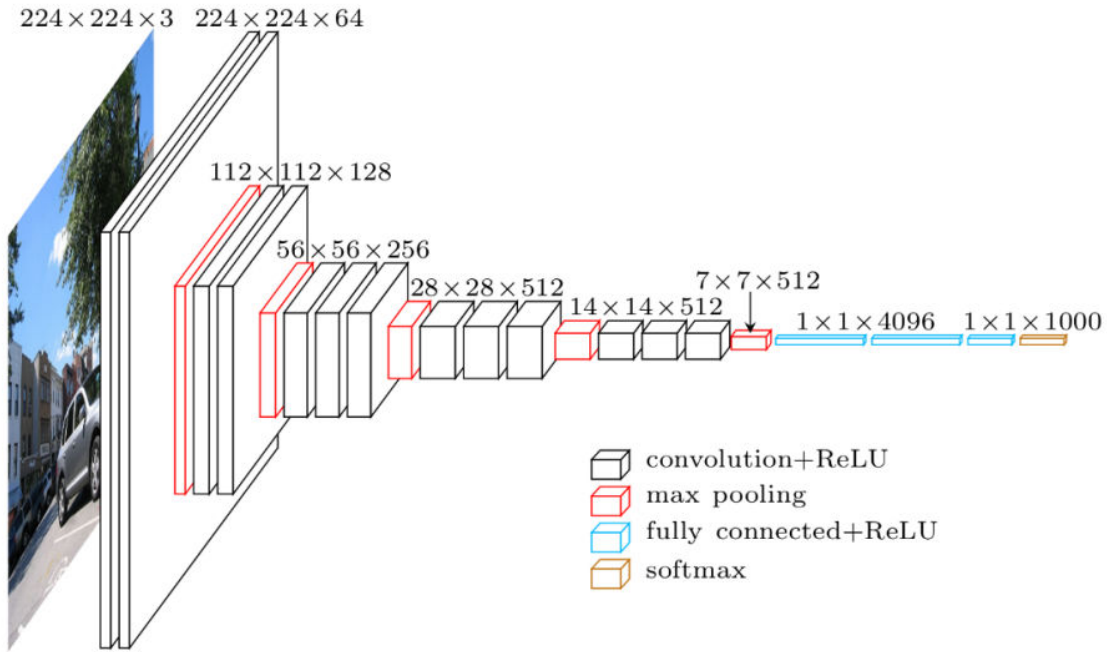


Figure 2.5 – The VGG Networks architecture. Image taken from (Simonyan and Zisserman 2014).

which are **VGG network** (Simonyan and Zisserman 2014) and **Residual network (ResNet)** (He, X. Zhang, et al. 2016).

- VGG network (Simonyan and Zisserman 2014):** The VGG network refers to a deep convolutional network for object recognition developed and trained by Oxford's renowned Visual Geometry Group (VGG) in 2014, which achieved excellent performance on the ImageNet dataset (Russakovsky et al. 2015). It consists of 16 convolutional layers with uniform architecture. It starts with a  $(7 \times 7)$  convolution layer at the beginning, then uses only  $(3 \times 3)$  convolutions, but lots of filters. At this time, it is the most preferred choice for extracting features from images. The weight configuration of the VGGNet is publicly available and has been widely used in many other applications as a baseline feature extractor. The architecture of this network is illustrated in Figure 2.5. One of the advantages of the VGG network is that instead of having many hyperparameters, it only has  $(3 \times 3)$  conv layers with stride 1 and same padding, all the max pooling layers being  $(2 \times 2)$  with a stride of 2. These conv layers are stacked to form 5 blocks of convolutional layers named accordingly conv1 to conv5, each having from 2 to 4 layers. This network has two versions: VGG-16 with a total of about 138 million parameters with top-5 errors of 8.8% and VGG-19 with 144 million parameters and top-5 errors rate of 9%.
- Residual Networks (He, X. Zhang, et al. 2016):** Among the VGG networks, VGG-19 has more layers, but the overall performance is worse than the one of VGG-16. This situation is quite unusual because in theory, deeper networks must have higher accuracy



Figure 2.6 – Residual Network architecture with residual connection (top row). Plain network architecture without residual connection (bottom row). Image taken from (He, X. Zhang, et al. 2016).

than shallow networks. However, experiments in practice reveal that deeper models do not always perform well. This problem is sometimes called a degradation problem. When deeper networks start converging, with the network depth increasing, accuracy gets saturated and then degrades rapidly. The reason is due to the *vanishing gradient problem* which implies that the model weights of the first layers cannot be updated correctly through the back propagation of the error gradient because when the chain rule multiplies error gradient values lower than one, the gradient error becomes zeros. In 2015, the Deep Residual learning framework (He, X. Zhang, et al. 2016) was proposed to solve this problem. Instead of learning a direct mapping of  $x \mapsto y$  with a function  $H(x)$ . The authors define the residual function using  $F(x) = H(x) - x$ , which can be re-framed into  $H(x) = F(x) + x$ , where  $F(x)$  and  $x$  represents the stacked non-linear layers and the identity function respectively. The "+x" operation at the end is the shortcut which allows the gradient to pass backward directly. By stacking these layers, the gradient could theoretically "skip" over all the intermediate layers and reach the bottom without being diminished. This breakthrough idea of ResNet (He, X. Zhang, et al. 2016) enabled the development of much deeper networks (hundreds of layers as opposed to tens of layers).

In its original form, *ResNet* includes four *Residual Blocks* and one prediction layer. Each block contains a different number of *Residual Units* and adds max-pooling operations at the end to reduce spatial dimensions. The *Residual Unit* is the core of *ResNet*. It performs a series of convolutions in a specific way. There are two types of *Residual Units*: the *baseline* and the *bottleneck unit*, which are illustrated in Figure 2.7. We will use the *bottleneck unit* in our segmentation network. It consists of three stacked operations:  $(1 \times 1)$ ,  $(3 \times 3)$  and  $(1 \times 1)$ . The two  $(1 \times 1)$  operations are designed for reducing and restoring dimensions. This leaves the  $(3 \times 3)$  convolution, in the middle, to operate on a less dense feature vector.

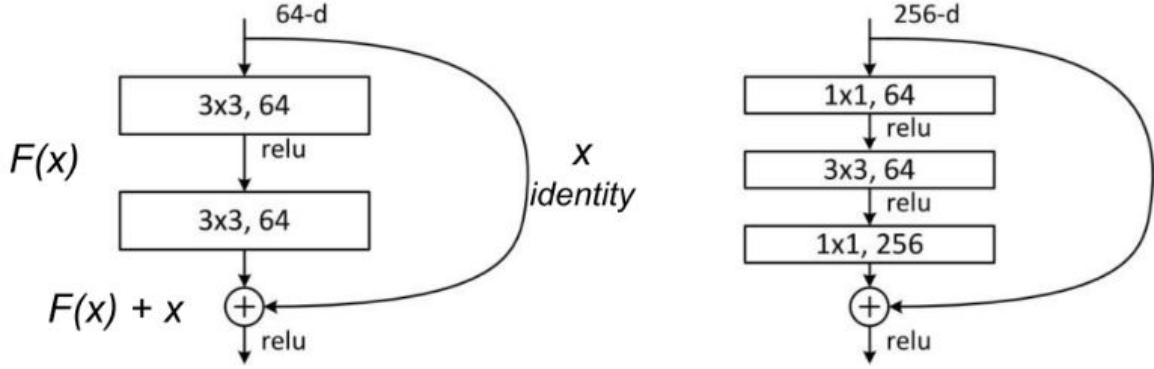


Figure 2.7 – Residual Unit. Left: the baseline unit; right: the bottleneck unit. Image taken from (He, X. Zhang, et al. 2016).

Also, *Batch Norm Layer* (Ioffe and Szegedy 2015) (which is explained later) is applied after each *Convolution layer* and before *ReLU*. This group of operations acts as applying a non-linear function  $F$  to its input  $x$ . After the non-linear transformations in  $F(x)$ , the unit combines the result of  $F(x)$  with the original input  $x$ . This combination is done by adding the two functions, an operation called skip connection. Skip connection is critical in the network architecture; without it, information from the previous block must follow a series of layers, which makes gradient vanishing if we have deep architecture. Rather than following the main path, with skip connection, the information can now follow a shortcut to go much deeper into the neural network. Figure 2.6 illustrates the ResNet architecture with skip connection, compared to plain network without skip connection. As a result, ResNet won the ILSVRC 2015 classification task (Russakovsky et al. 2015). Since the introduction of ResNet, the idea of residual learning has become essential in most of the modern network architectures.

## Learning

As in the traditional Neural Networks, CNNs are trained by minimizing a pre-defined **loss function** through **back propagation**. During the **forward pass**, a **batch** of training samples is passed through the whole network, and the output is compared with the ground truth to compute the loss. Then a back propagation process calculates the gradient of all the networks parameters with respect to this loss. These gradients are then used by an **optimizer** to update the network parameters. The process continues with the next batch of training samples until all training samples are visited. This procedure will finish one **epoch**. In general, more epochs are needed to increase the accuracy of the network. In this section, we will discuss some advanced techniques which are helpful in the learning process. We focus on discriminative tasks in which the network is designed to model conditional probabilities.

**Loss function:** Loss function measures the inconsistency between the predicted value and the actual value. There are different forms of loss functions, and they have different effects

on the model. In a classification problem, the output of a CNNs is often passed through a **softmax function** to model the conditional distribution  $p(y|x, \theta)$  of the target  $y$ , given the input  $x$  and the parameters  $\theta$ . From this distribution, a very common type of loss function to use is Cross Entropy loss:

$$H(p, q_\theta) = \mathbb{E}_p [q_\theta]$$

where  $p$  is the true distribution and  $q$  is the model distribution parameterized with  $\theta$ . Optimize this loss is equivalent to minimize the negative log-likelihood of the training data. Assume  $q_\theta = \{\hat{y}_j | j = 1..C\}$  with  $C$  the number of classes.

$$\mathcal{L}(\theta, X, Y) = - \sum_{j=1}^C y_j \log \hat{y}_j$$

In binary classification, where the number of classes  $C$  equals 2, cross-entropy can be calculated as:

$$\mathcal{L} = -(y \log(p) + (1 - y) \log(1 - p))$$

with  $y \in \{0, 1\}$  and  $p \in [0, 1]$ . The segmentation problem can be seen as a pixel-wise classification problem, therefore a reasonable loss is the **pixel-wise cross entropy** loss. This loss examines each pixel individually, comparing the class predictions (depth-wise pixel vector) to the ground truth one-hot encoded target vector.

Another popular loss function developed for binary image segmentation tasks is the Dice coefficient, which is essentially a measure of overlap between two samples.

$$Dice = \frac{2 |A \cap B|}{|A| + |B|}$$

**Optimization:** When optimizing the loss in CNNs, all optimization methods in practice find a local minimum instead of a global minimum because of the non-convexity of the problem. Among them, the gradient descent method is the classical optimization method. It initializes randomly a set of parameters  $\theta_0$ , then the update process is defined as:

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla_{\theta} \mathcal{L}(\theta, X, Y)$$

In this equation,  $\eta$  is the *learning rate* which defines the parameters update speed. This method requires a full pass on the data at each iteration which is not efficient in practice with large-scale problems. To address this issue, an alternative called *Stochastic Gradient Descent* (SGD) (Robbins and Monro 1951) is often adopted to approximate the gradient. In this case, a mini-batch of random training samples is used to estimate the gradient.

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla_{\theta} \mathcal{L}(\theta, x^{i,i+1\dots i+n}, y^{i,i+1\dots i+n})$$

where  $n$  is the size of mini-batch and  $n \ll d$ . To reduce the noise and improve the convergence speed of SGD, many different variants of this learning rule are proposed, of which the *momentum* (Qian 1999) method and *Adam* (Kingma and Ba 2014) are popular choices. Momentum (Qian 1999) is a method that helps accelerate SGD in the relevant direction and dampens oscillations. It does this by adding a fraction of  $\gamma$  of the updated vector of the past time step to the current updated vector:

$$\begin{aligned} v_{t+1} &= \gamma v_t + \eta \nabla_{\theta} \mathcal{L}(\theta) \\ \theta_{t+1} &= \theta_t - v_{t+1} \end{aligned}$$

**Regularization:** The complex architecture of a CNN makes it a powerful tool to model different data distributions. However, these complex models make CNNs face a common problem: *overfitting*. Overfitting happens when the model is able to achieve good performance on the training data but performs poorly on the test data. When overfitting, the model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. In CNNs design, there are classical techniques to reduce overfitting without increasing the training data. These are called regularization techniques. The most powerful and common techniques are **Weight Decay** (Krogh and Hertz 1992) and **Drop Out** (Srivastava et al. 2014).

- **Weight Decay:** Weight decay is used to penalize large weights using certain constraints on their values such as  $L_1$  or  $L_2$  norm. In  $L_2$  regularization, an extra term is added to the cost function that penalizes the square magnitude of all parameters.  $L_2$  regularization has an intuitive interpretation of heavily penalizing peaky weight vectors and preferring diffuse weight vectors. Another popular weight decay regularization is the  $L_1$  regularization, which poses sparse constraints on the weights. Neurons with  $L_1$  regularization end up using only a sparse subset of their most important inputs and become nearly invariant to the "noisy" inputs.
- **DropOut:** An extremely effective regularization technique for neural networks is *Dropout*. During training, *Dropout* is implemented by only keeping a neuron active with a specific probability  $p$  (a hyperparameter), or setting it to zero otherwise. It can be interpreted as sampling a neural network within the full neural network and only updating the parameters of the sampled network based on the input data. At test time, we would ideally like to find a sample average of all possible  $2^n$  dropped-out networks. Unfortunately, this is unfeasible for large values of  $n$ . However, we can find an approximation by using the full network with each node's output weighted by a factor of  $p$ , so the expected value of the output of any node is the same.

**Convergence:** In training Neural Networks, convergence describes a progression towards a network state where the network has learned to appropriately respond to a set of training

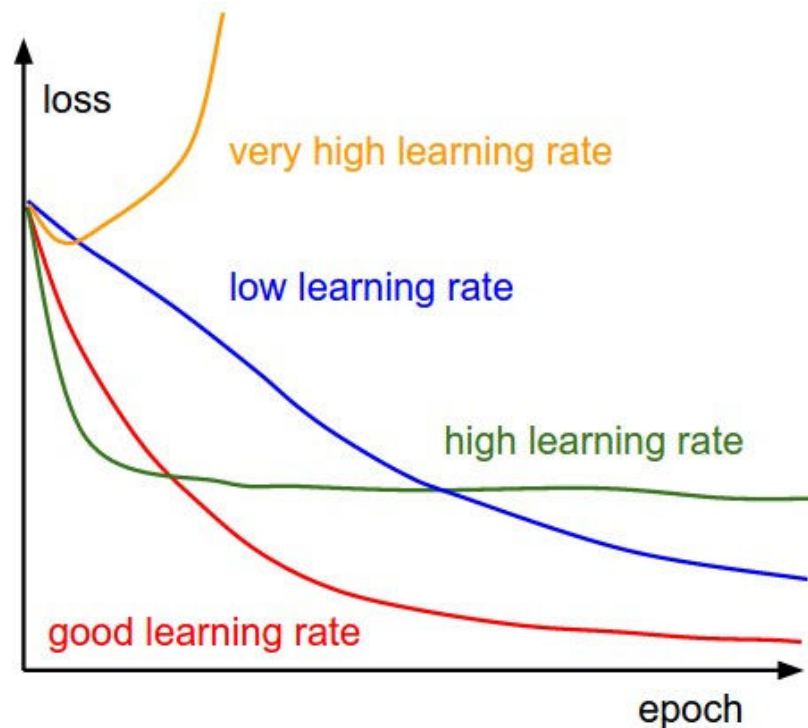


Figure 2.8 – Effect of various learning rates on convergence. Image credit: [Stanford CS231n class](#).

patterns within some margin of error. In practice, make CNNs convergence fast is a challenging task. We will discuss some advanced techniques and tricks to make CNNs convergence faster during training.

- **Learning rate:** The learning rate is a hyperparameter that controls how much we are adjusting the weights in the direction of the loss gradient of a mini-batch. It is one of the essential hyper-parameters to tune for training a deep CNN. The lower the value, the more reliable the process, but optimization will take a lot of time because steps towards the minimum of the loss function are tiny. Moreover, we can get stuck in a bad local optimum. In contrast, if the learning rate is high, then the system contains too much kinetic energy and the parameter vector bounces around chaotically. Therefore, training may not converge or even diverge. The effects of different learning rates on the performance of the model are illustrated in Figure 2.8.

Ideally, the training should follow an annealing scheme over time. Starting from a relatively large learning rate because, in the beginning, random weights are far from optimal, and then the learning rate can decrease during training to allow more fine-grained weight updates. However, deciding how large is the starting point and which strategies to anneal the learning rate is a tricky problem.

- *Learning rate initialization:* Typically, initializations of learning rates are configured naively at random by experiences of the CNNs practitioners. A naive approach is to try a few different values and see which one gives the best loss



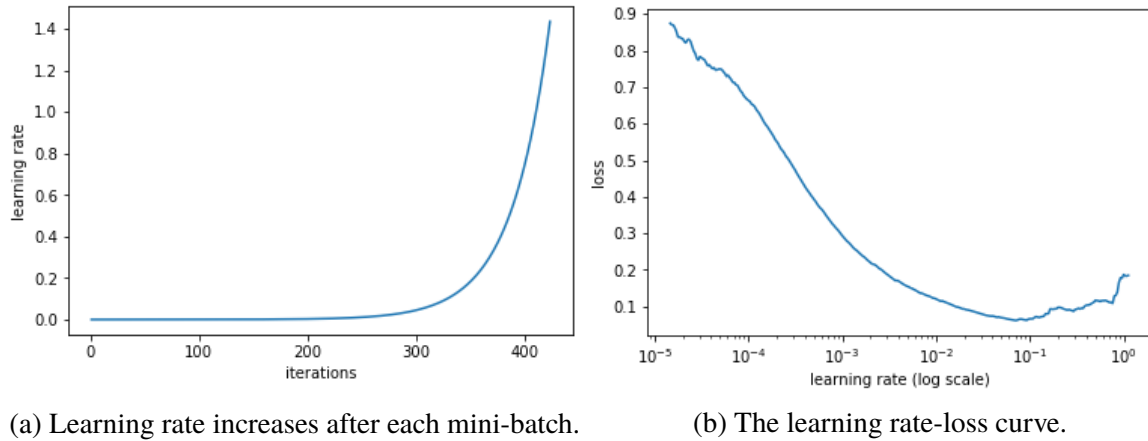


Figure 2.9 – A learning rate estimation method by training the model initially by increasing the learning rate after each iteration (a), and report a learning rate-loss curve (b).

without sacrificing the speed of training. However, it is not a good solution, and there exist several ways to select a proper initialization for the learning rate. In (Smith 2017), the authors argue that a good learning rate can be obtained by training the model initially with a very low value and increasing it exponentially at each iteration and output a learning rate-error curve. As the learning rate increase, there will be a point in this curve where the loss stops decreasing and starts to increase. The best learning rate should lie in the region which is close to that point where the learning rate is high when the loss still decreases. For example, in Figure 2.9(b), the best learning rate should be in the range of (0.001,0.01).

- *Learning rate annealing*: Selecting a learning rate once, before training, is not enough. The optimal learning rate decreases while training. There exist several common strategies of annealing the learning rate such as *Step decay* *Exponential decay* or *1/t decay*.
- *Cyclical Learning Rates*: Another strategy for learning rate is Cyclical Learning rates, where the learning rate is reset to a high value after several iterations or epoch, repetitively. The intuition behind that is that increasing the learning rate will force the model to jump to a different part of the weight space if the current local minima is not robust and make the model generalize better to unseen data. There are different variations of these strategies. In (Smith 2017), the authors propose a method where the learning rates are restarted after every few iterations (Figure 2.10 (a) and (b)). Another method that is also popular is called Stochastic Gradient Descent with Warm Restarts (Loshchilov and Hutter 2016) (Figure 2.10 (c)). This method uses the cosine function as the cyclic function and restarts the learning rate at the maximum at each cycle. All of these methods are shown in practice to have faster convergence and converge to a lower loss when compared to using a fix learning rate.

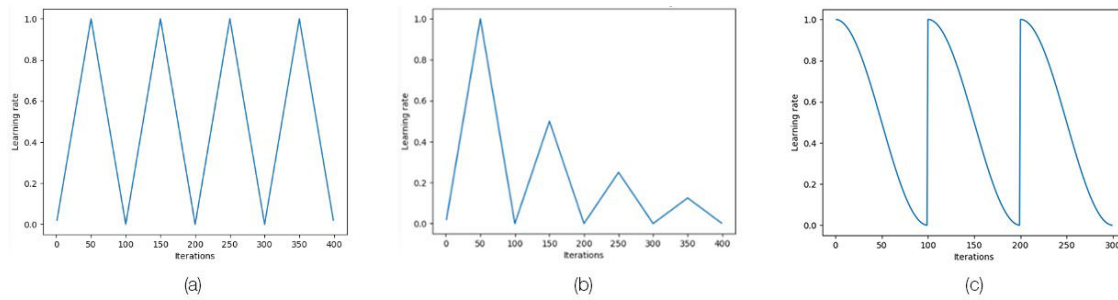


Figure 2.10 – Different methods in cyclical learning rates. (a) "Triangular" method proposed by (Smith 2017), min and max learning rate are kept the same. (b) "Triangular 2" method by (Smith 2017), the difference is cut in half after each cycle. (c) Cosine annealing method by (Loshchilov and Hutter 2016), the cosine function is used as the cyclic function.

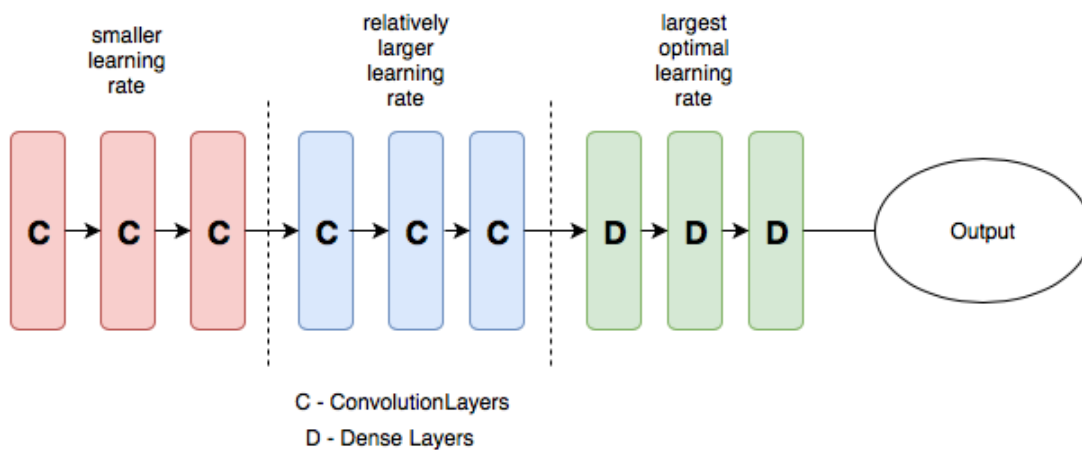


Figure 2.11 – Transfer learning using differential learning rates. Image credit: <https://towardsdatascience.com/transfer-learning-using-differential-learning-rates-638455797f00>

- *Discriminative fine tuning*: This is a method where different learning rates are set to different layers in the network during training rather than using the same rate throughout the network during training. For example, Figure 2.11 illustrates the concept of discriminative fine-tuning where a pre-trained model is split into 3 groups, each group would be configured with an increased learning rate value. The intuition behind this method of configuration is that the first few layers would typically contain very granular details of the data, such as the lines and the edges of which we usually do not want to change much. In contrast, in later layers, where we get detailed features of the data for some particular task, we might not necessarily need to keep them. This trick is especially useful for transfer learning.
- **Batch Normalization (BatchNorm)**: Another common technique for faster training in deep CNNs is Batch Normalization which normalizes the internal representation of the data. This method partially alleviates the problem of internal covariate shift, a problem



occurring when the distributions of intermediate layers of neural networks change during training. This is because the input of an intermediate layer is the output of the previous layer, so when the parameters of this previous layer get updated over time, its output will change too. This problem makes training deep networks difficult. In BatchNorm, during training, each activation of the mini-batch is centered to zero-mean and unit variance. The mean and variance are measured over the whole mini-batch, independently for each activation. Using BatchNorm will make the model more robust. Therefore it allows us to use much higher learning rates and be less careful about initialization.

## 2.1.2 Segmentation Networks

Regular image classification CNNs all have a similar structure. These models take images as input and output a single value representing the category of that image. Different from image classification, in segmentation we want to make decisions for every pixel in an image. Therefore, for each pixel, the model needs to classify it as one of the pre-determined classes. In another way, segmentation means understanding images at a pixel level. In this section, we will discuss transfer learning, a classical technique used in segmentation. After that, we review different segmentation network architectures that motivate the segmentation network architecture we will use later in this work. We finish by describing different evaluation metrics on image segmentation.

### Transfer learning

In practice, training a deep neural network from scratch by random initialization is difficult in many situations because it requires large-scale dataset, careful tuning of hyperparameters and hours of computations to converge. Therefore, transfer learning schemes are often adopted. Transfer learning is a technique where a model trained on one task is re-purposed on a second related task. This scheme enables us to utilize knowledge from previously learned tasks and apply them to newer, related ones. In the case of neural networks, transfer learning means that the pre-trained network weights for one particular task are used as initialization or as feature extractor for other tasks. It is proved in (Yosinski et al. 2014) that transferring features can be better than using random initialization. Nevertheless, finding a whole new architecture of a CNN is often not an easy task. Therefore, it is common to reuse already existing network architectures, thus enabling transfer learning.

In the case of computer vision problems, certain low-level features, such as edges, shapes, corners, and intensity, can be shared across tasks, and thus enable knowledge transfer among tasks. In particular, for our segmentation purpose, due to the difficulty of gathering and creating per-pixel labeled segmentation datasets, segmentation datasets are not as large as classification datasets such as ImageNet (Russakovsky et al. 2015). For that reason, transfer learning, and in particular fine-tuning from pre-trained classification networks is a common

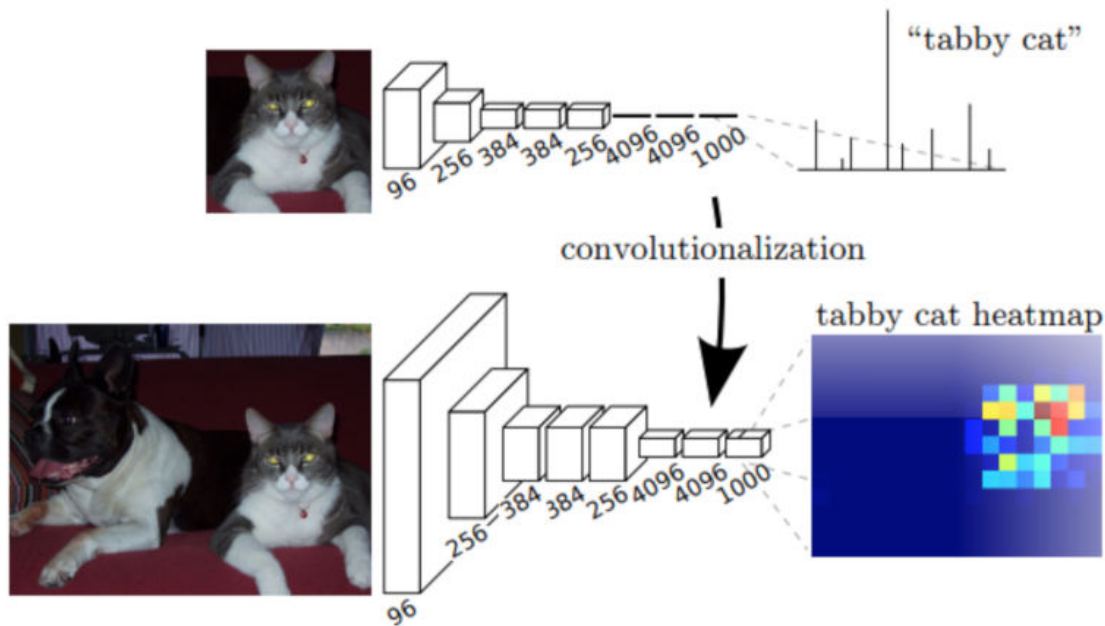


Figure 2.12 – Fully Convolutional Network. Image taken from (Long, Shelhamer, and Darrell 2015).

trend for segmentation networks and has been successfully applied in most of the methods.

During transfer learning, the training process differs slightly when fine-tuning instead of training from scratch. Because the early layers of the network have captured the low-level concepts of the image, usually only the higher-level part of the network should be fine-tuned. Moreover, an appropriate policy for the learning rate should be picked which is usually smaller since the pre-trained weights are expected to be relatively good so there is no need to change them drastically.

In the next section, we discuss common several Neural Network architectures with a focus on Deeplab - a model which is used in our implementation.

### Fully Convolutional Networks (FCNs)

The insight of the FCNs approach is to take advantage of existing CNNs as powerful visual models that are able to learn hierarchies of features. FCNs have two key features: Firstly, it upsamples the last layer using *fractionally strided convolutions* (also named *deconvolutions* or *transpose convolution*) to produce dense per-pixel labeled outputs. This makes the final output layer have the same height and width as the input image, with a number of channels equal to the number of classes. Secondly, all layers are convolutional layers. The last Fully connected layer is removed and replaced by convolutional layers to output spatial maps and classify each pixel in the image. Therefore, it can deal with arbitrary sizes. An overview of the architecture of a FCN is illustrated in Figure 2.12. This approach achieved a significant improvement in segmentation accuracy over traditional methods on standard datasets like PASCAL VOC (Everingham et al. 2015) and was a milestone in the image segmentation

problem.

### Encoder-Decoder architecture

Since the introduction of FCNs ([Long, Shelhamer, and Darrell 2015](#)), most of the successful state-of-the-art works for semantic segmentation have used similar approaches. They usually employ Encoder-Decoder architectures with three common components: convolutions, downsampling, and upsampling layers.

- **Encoder:** As in FCNs architecture, most of the methods in segmentation rely on a pre-trained network for the image classification task, remove its *FC* layers and use the resulting network as the encoder part. It produces low-resolution image representations or feature maps by either using convolution striding or regular pooling operations to reduce the spatial dimensions of given feature maps. In general, an encoder allows one to encapsulate the information of the image into a low dimensional space. After this part, we obtain a feature vector with shape  $[w, h, d]$  where  $w, h$  and  $d$  are the width, height, and depth of the output features respectively. The spatial dimensions of this compressed vector are smaller but denser than the original input.
- **Decoder** While the encoder remains quite similar for different segmentation networks, the main problem lies in learning how to decode those low-resolution representations of the encoder to pixel-wise predictions for segmentation. The decoder is the most important part of this kind of architecture. The goal is to increase the spatial resolution so that the output vector has the same dimensions as the input. The basic idea is feeding the encoded feature map to a series of upsampling layers to restore the original resolution. Upsampling can be obtained by using some simple operation such as bilinear interpolation or nearest neighbor. However, in practice, to have more control, upsampling layers are often obtained by strided transpose convolution (also called deconvolution) layers with learnable weights. These functions go from deep and narrow layers to wider and shallower ones.

Many network implementations are using these upsampling layers in different ways. For example, in FCN ([Long, Shelhamer, and Darrell 2015](#)), after converting regular CNNs networks into FCNs by replacing their final layers, the author combined upsampling from that final layer with upsampling from earlier activation features, to get more precise spatial information. The upsampling procedure use "deconvolutions" to upsample the intermediate activation maps so that they match the width and height of the original input image. This upsampling procedure can solve the problem of losing too much spatial information during the downsampling step in the encoder. Another typical example of a decoder is Unet([Ronneberger, Fischer, and Brox 2015](#)). It is a network where the decoder consists of upsampling and concatenation followed by regular convolution operations. A more detail review can be seen in Chapter 3.

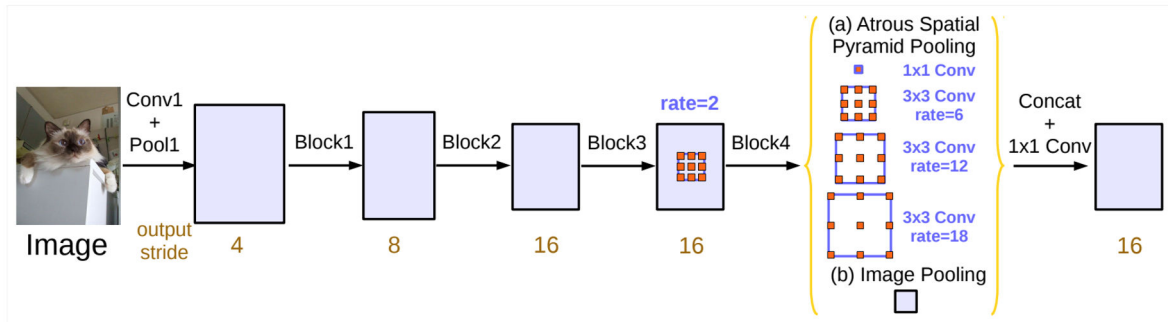


Figure 2.13 – Parallel modules with atrous convolution (ASPP), augmented with image-level features. Image taken from (L.-C. Chen, Papandreou, Kokkinos, et al. 2016).

- **Skip connection** One of the most crucial factors in improving the performance of a segmentation network is *skip connection*. As mentioned in the name, a skip connection bypasses some layer in the neural network and feeds the output of one layer as the input to the next layer and also possibly some other layers. In the encoder-decoder architecture, skip connections are often used to transfer local information by concatenating or summing feature maps from some early layer in the encoder with feature maps from the upsampling path in the decoder. By merging features from various resolution levels, it helps combining context information with spatial information and recovering information lost during down-sampling, so that correctly classifying small details becomes easier. Skip connections also facilitate the training of deeper networks. As we pass through many layers, the gradient can be lost, which, as we saw already, is known as the gradient vanishing problem. Skip connections create a path for gradient during the back propagation, therefore reducing the risks of gradient vanishing. Because of these advantages, skip connection became a common technique used in different segmentation networks (e.g., U-Net (Ronneberger, Fischer, and Brox 2015), V-Net (Milletari, Navab, and Ahmadi 2016), and The One Hundred Layers Tiramisu (DenseNet) (Jégou et al. 2017)).

## Deeplab architecture

In this section, we discuss Deeplab (L.-C. Chen, Papandreou, Kokkinos, et al. 2016; L.-C. Chen, Papandreou, Schroff, et al. 2017), a segmentation network which we will use in our method.

Deeplab was proposed in 2015 by L.-C. Chen, Papandreou, Kokkinos, et al. (2016). It presents an architecture for controlling signal decimation and learning multi-scale contextual features. In the following section, we will discuss this method in details as it is the backbone of our segmentation network.

Deeplab uses an ImageNet pre-trained network as its main feature extractor. However, it proposes a new *Residual block* for multi-scale feature learning. Instead of regular convolutions, the last block of the encoder uses *atrous convolutions*. Also, each convolution (within this

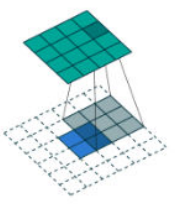
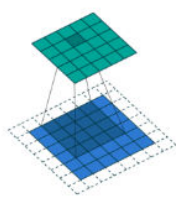
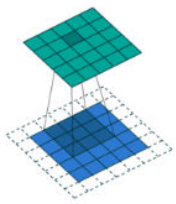
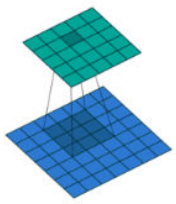
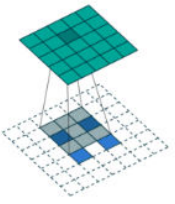
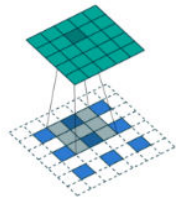
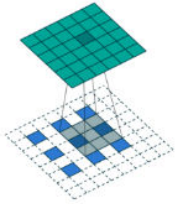
			
No padding, no strides, transposed	Arbitrary padding, no strides, transposed	Half padding, no strides, transposed	Full padding, no strides, transposed
			
No padding, strides, transposed	Padding, strides, transposed	Padding, strides, transposed (odd)	

Figure 2.14 – An example of atrous convolution with different strides and paddings. Image taken from (L.-C. Chen, Papandreou, Kokkinos, et al. 2016).

new block) uses different dilation rates to capture the multi-scale context. Additionally, on top of this new block, it uses *Atrous Spatial Pyramid Pooling* which uses dilated convolutions with different rates as an attempt of classifying regions with an arbitrary scale.

The new *Atrous Residual Block* contains three residual units. In total, the three units have three ( $3 \times 3$ ) convolutions. Motivated by multigrid methods, Deeplab proposes different dilation rates for each convolution. In summary, multigrid defines the dilation rates for each of the three convolutions. The overview of Deeplab network architecture is illustrated in Figure 2.13.

To understand the Deeplab architecture, we need to focus on two components: (i) *Atrous convolutions* and (ii) *Atrous Spatial Pyramid Pooling (ASPP)*.

- **Atrous convolutions:** *Atrous convolution* is a filter for upsampling which allows us to expand filter's receptive fields (field of view) without losing resolution. Beside the learnable weights, atrous convolutions introduce another parameter called the *dilation rate*,  $l$  to controls the upsampling factor. For instance, a ( $3 \times 3$ ) convolution filter with the dilation rate equal to 1 is equivalent to a standard convolution. However, if we set the dilation rate to 2, it has the effect of enlarging the convolution kernel and will have the same field of view as a ( $5 \times 5$ ) kernel, while only using 9 parameters.

The principle of atrous convolution function is simple. It performs a regular convolution using the dilated filter. The dilated filter is obtained by expanding the convolution filter according to the dilation rate and then filling holes with zeros, as illustrated in Figure 2.14. As a consequence, a ( $3 \times 3$ ) convolution with  $l = 2$  make the kernel can cover a ( $5 \times 5$ ) area. Similarly, setting  $l = 3$  enables a ( $3 \times 3$ ) kernel to get signals

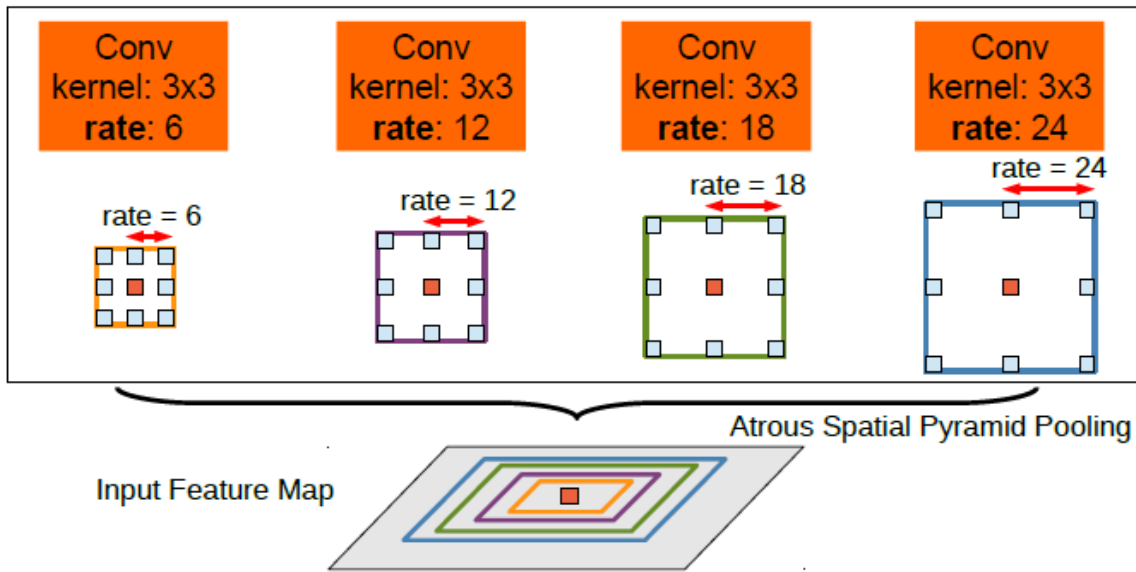


Figure 2.15 – Atrous Spatial Pyramid Pooling. Image taken from (L.-C. Chen, Papandreou, Kokkinos, et al. 2016).

from a  $(7 \times 7)$  corresponding area. This effect allows us to control the resolution of the output features. It helps adding larger context without increasing the number of parameters. Stacking several atrous convolutions layers makes the receptive fields grow exponentially. This allow efficient dense feature extraction. Also, atrous convolution does not require training any additional weights thanks to the dilated holes in convolution kernels, therefore, no additional computational cost required. However, by introducing a hyper parameter to the architecture, the parameter must be tuned according to the size of the features maps. The efficiency of atrous convolutions depends on a good choice of the dilation rate. To understand this, the authors introduce the concept of *output stride*.

*Output stride* is the ratio between the spatial size of the input image and the output feature map. For example, with an *output stride* of 16, an image size of  $(224 \times 224 \times 3)$  will outputs a feature map with size  $(14 \times 14)$  which is 16 times smaller in spatial dimensions. Different output strides can have effects on segmentation models. Excessive signal decimation (large output stride) is harmful to dense prediction tasks but the network can be trained faster. In contrast, models with smaller output stride can provide finer segmentation results with the cost of high training time and memory. In our architecture, we will choose an output stride equal to 16 for practical reasons.

- **Atrous Spatial Pyramid Pooling (ASPP):** The idea behind ASPP is to provide the model with multi-scale information by fusing the output of the atrous convolution with different rates. In details, to capture long-range context information at different scales of the object, ASPP applies a series of atrous convolutions with different dilation rates. For example, a ASPP in Figure 2.15 contains 4 parallel  $(3 \times 3)$  convolutions with dilation rates equal to (6, 12, 18, 24). Moreover, image-level features are incorporated to



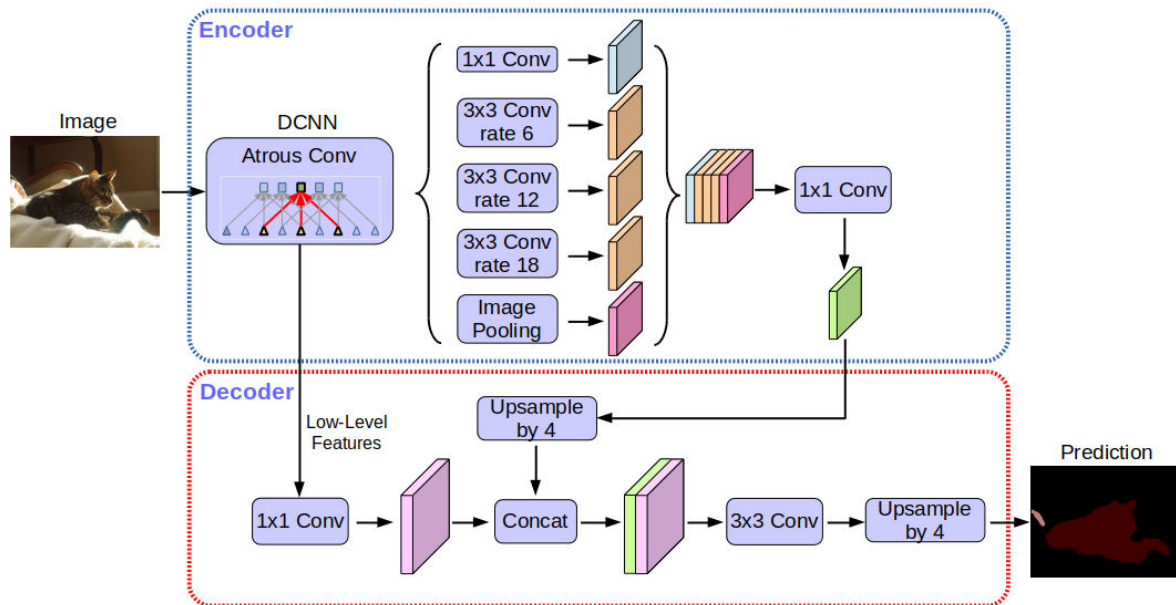


Figure 2.16 – Deeplabv3 plus. Image from (L.-C. Chen, Y. Zhu, et al. 2018).

add more global context information via *Global Average Pooling* (GAP). The output features from last atrous block are passed through a GAP layer to provide large receptive fields, and the resulting features are then fed to a  $(1 \times 1)$  convolution. Finally, the result is bilinearly upsampled to the desired dimensions. Output features from all the branches are combined into a single vector via concatenation. After ASPP, the result is fed to another  $(1 \times 1)$  convolution to produce the final segmentation result.

- **Deeplabv3 plus:** Since the introduction of the first DeepLab model (L.-C. Chen, Papandreou, Kokkinos, et al. 2016), many improvements have been made in Deeplabv2 (L.-C. Chen, Papandreou, Kokkinos, et al. 2018) and Deeplabv3 (L.-C. Chen, Papandreou, Schroff, et al. 2017): better object scale modeling, careful assimilation of contextual information, improved training procedures, and increasingly powerful hardware and software. Recently, the authors have extended Deeplabv3 by combining it with an Encoder-Decoder architecture and skip connections, resulting in the architecture named as *Deeplabv3 plus* (L.-C. Chen, Y. Zhu, et al. 2018). An overview of the network architecture is shown in Figure 2.16. In this architecture, the decoder side backbone is based on ResNet (He, X. Zhang, et al. 2016) and Xception (Chollet 2017) and it includes a dilated convolution with various sizes which learns a small feature map. Then the decoder effectively up-scales by using skip connections. This decoder module aims to refine the segmentation results, especially along object boundaries. Moreover, many tricks have been proposed to make the training process faster: batch normalization layers (Ioffe and Szegedy 2015) inside the atrous spatial pyramid pooling block, use of a pre-trained large multi-label dataset COCO (T.-Y. Lin et al. 2014) depthwise separable convolution to both atrous spatial pyramid pooling. This method yields the current state-of-the-art on the popular benchmark dataset PASCAL VOC (Everingham et al.

2015) with 89.0% *mIoU*.

### 2.1.3 Evaluation metrics

For a segmentation system to be useful and to produce a significant contribution to the field, its performance must be evaluated using standard and well-known metrics.

In this section, we review different ways to evaluate the efficiency and accuracy of segmentation systems with a focus on semi-supervised approach. Given a ground truth mask  $G$  on a particular frame and an output segmentation  $M$ , there are several important accuracy metrics to evaluate the performance of segmentation results: region similarity, average precision and recall.

Depending on the purpose or the context of the system, some metrics might be of more importance than others, i.e., accuracy may be expendable up to a certain point in favor of execution speed for a real-time application. In our video objects removal application, recall is more important than Jaccard, as we will explain later.

**Region Similarity  $\mathcal{J}$ :** This is the standard metric for segmentation purpose. It measures the region-based segmentation similarity by using the Jaccard index  $\mathcal{J}$  or *IoU* which is a ratio between the intersection and the union of the ground truth set and the predicted segmentation.

$$\mathcal{J} = IoU = \frac{|G \cap M|}{|G \cup M|}$$

This ratio calculates the number of true positives (intersection) over the sum of true positives, false negatives, and false positives (union). This index *IoU* is computed on a per-class basis and then averaged.

**Precision and Recall  $\mathcal{P}, \mathcal{R}$ :** They are the simplest metric. Precision is simply computing a ratio between the amount of properly classified pixels (true positive) and the total number of them (true).

$$\mathcal{P} = \frac{|G \cap M|}{|M|}$$

Recall is the ratio between the amount of classified pixels (true positive) and the number of ground truth pixel (positive).

$$\mathcal{R} = \frac{|G \cap M|}{|G|}$$

Precision and recall used to describe under/over segmentation phenomenon.

Beside these accuracy metrics, others important factors when evaluating a segmentation system are computational time and memory footprint.



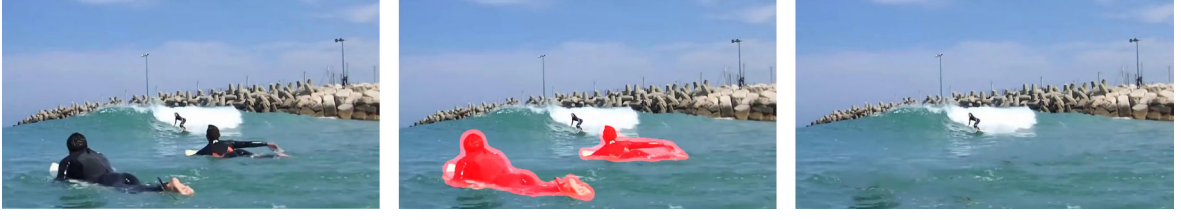


Figure 2.17 – The inpainting problem. Left: original image; middle: inpainting mask  $\mathcal{H}$ , in red; right: an inpainting result.

## 2.2 Inpainting

In the digital world, inpainting refers to the application of sophisticated algorithms to replace lost or corrupted parts of the image/video data.

In this section, we introduce the notations and the concepts necessary to the modeling of the problem of inpainting and different components which we use in our video inpainting method.

### 2.2.1 Definition and notation

**Image and video** We define an image or a video as a continuous function:

$$u : p \in \Omega \subset \mathbb{R}^m \mapsto u(p) \in \mathbb{R}^n$$

where  $m = 2$  for images and  $m = 3$  for videos,  $n$  is the number of channels in the image, e.g.,  $n = 1$  for a gray-scale image, and  $n = 3$  for an RGB image. Each channel  $k$  of  $u$  will be noted  $u_k$ . In our thesis, we focus on the video inpainting case. A video will be considered as a sequence of images of the same size that are placed one after the other to form a spatio-temporal volume  $\Omega$ . We denote a spatio-temporal position in the video as  $p = (x, y, t) \in \Omega$ , and  $u(p)$  is the value of the video at this position: e.g.  $u(p) \in \mathbb{R}^3$

**The inpainting problem** Give an image/video  $\Omega$ , let us denote a missing domain  $\mathcal{H}$  in  $\Omega$ ,  $\mathcal{H} \subset \Omega$ . This domain contains the unknown part of the volume  $\Omega$  that we wish to reconstruct. The inpainting problem consists in modifying the values of the pixels in  $\mathcal{H}$  so that this region look consistent with respect to its surroundings (Figure 2.17). The localization of  $\mathcal{H}$  is not part of the inpainting problem, it is generally provided by the user or by a segmentation algorithm in the form of a binary image whose pixels labeled with 1 (respectively 0) represent the unknown data (respectively known). Let  $\mathcal{D}$  be the unoccluded region (source region), together with  $\mathcal{H}$ , they form a partition of  $\Omega$ , which means  $\mathcal{H} \cup \mathcal{D} = \Omega$  and  $\mathcal{H} \cap \mathcal{D} = \emptyset$ . In our thesis, we aim at reconstructing the missing value in  $\mathcal{H}$  by using the information from region  $\mathcal{D}$ .

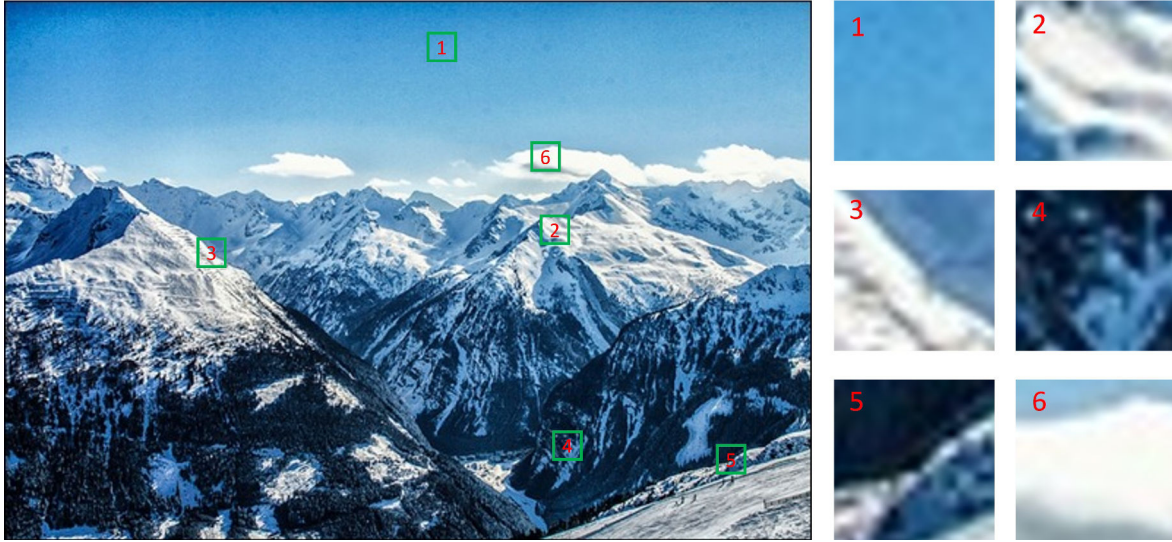


Figure 2.18 – Example of 2D patches extracted from an image.

**Patch definition** Since our video inpainting is based on patches, we explain the definition of a patch. We start with patches in the image and extend it to video cases.

In an image, a patch, also called a block, is defined as a rectangular sub-part of an image. It allows capturing the structures and textures present in the image in order to characterize it locally. By analyzing its content, for example, average, variance, histogram..., it is possible to extract macroscopic characteristics of the image. Examples of patches extracted from an image are given in Figure 2.18.

We first define a neighborhood for each pixel  $p \in \Omega$  as

$$\mathcal{N}_p = [p - n_s/2, p + n_s/2]^m,$$

where  $m = 2$  for images and  $m = 3$  for videos. Then, a patch centered at  $p$  is the collection of values defined as

$$\psi_p = \{\psi_p(q) : q \in \mathcal{N}_p\},$$

where  $\psi(q) = u(p + q)$ .

For color data, the patch centered at  $p$ ,  $\psi_p$  has size of  $(3 \times N)$  where the  $N$  pixels in  $\mathcal{N}_p$ ,  $q_1 \dots q_N$  are ordered in a pre-defined way.

Now let us define the source region. Let us denote  $\tilde{\mathcal{D}} = \{p \in \mathcal{D} : \mathcal{N}_p \subset \mathcal{D}\}$  the set of unoccluded pixels whose neighborhood is also unoccluded. Patches centered at pixels in this set are called source patches, and they are only composed of known color value pixels. These source patches are used to inpaint the occlusion.

**Nearest neighbor field** For patch-based inpainting applications, the goal is to copy patches from source regions and paste them into the occlusion. To specify which patches should be copied, a source patch searching algorithm must be applied.

However, if we search for the exact nearest neighbor for every point in the target region

$\mathcal{H}$ , the work will be too hard to complete. So in practice, the NNF is often approximated by some random-based approaches (Barnes, Shechtman, et al. 2009; Korman and Avidan 2011) in order to accelerate the calculation speed. The most common method for this purpose is PatchMatch (Barnes, Shechtman, et al. 2009) which proves to be a very efficient method in both accuracy and speed. The PatchMatch algorithm has three main components. Initially, the nearest-neighbor field is filled with either random offsets or some prior information. Next, an iterative update process is applied to the NNF, in which good patch offsets are propagated to adjacent pixels, followed by a random search in the neighborhood of the best offset found so far. More details will be presented when introducing our inpainting algorithm.

**Distance between two patches** For a patch  $\psi_p$  in the target region, the goal is to find  $q$  in the source region such that the similarity between  $\psi_p$  and  $\psi_q$  is small. The similarity is defined according to a distance function  $d(., .)$ . To compare two patches  $\psi_p$  and  $\psi_q$ , the simplest and most used distances is the *sum of squared differences (SSD)*. This distance has the advantage of detecting good candidate patches while being fast to calculate. If all pixels values of the target patch  $\psi_q$  are available, the SSD distance between  $\psi_p$  and  $\psi_q$  is defined as:

$$d_{SSD}(\psi_p, \psi_q) = \|\psi_p - \psi_q\|^2 = \sum_{i \in N_p} \|\psi_p(i) - \psi_q(i)\|^2$$

At the initialization stage of the algorithm, for several patches near the border of the occlusion, several pixels of the target patch may be occluded. Therefore,  $\psi_q$ , can be masked. In this case, the SSD taking counting hidden pixels in the target patch is defined as:

$$d_{SSD}(\psi_p, \psi_q) = \sum_{i \in N_p \cap \mathcal{D}} \|\psi_p(i) - \psi_q(i)\|^2$$

**Patch-based inpainting** We wish to complete an image/video in a missing part  $\mathcal{H}$ , knowing the values in a region  $\mathcal{D}$ .

The problem of inpainting by patch can be formulated in the following way: finding a *nearest neighbor field*  $\phi : p \in \mathcal{H} \mapsto \phi(p) \in \tilde{\mathcal{D}}$ . This function associates with each point  $p$  of the occlusion  $\mathcal{H}$  the position  $\phi(p)$  of a known pixel which will be copied on  $p$ . It is then defined as:

$$\phi(p) = \arg \min_{q \in \mathcal{H}} d(\psi_p, \psi_q)$$

This type of mapping is frequently used, not only in inpainting application (He and J. Sun 2012b; Newson, Almansa, Fradet, et al. 2014), but also in image retarget application (Pritch, Kav-Venaki, and Peleg 2009; Barnes, Shechtman, et al. 2009). An example of patch-based image inpainting is shown in Figure 2.19.



Figure 2.19 – Result of patch-based image inpainting method of (Newson, Almansa, Gousseau, et al. 2017). From left to right: Original image, the inpainting domain, result.

### 2.2.2 Motion estimation

Motion estimation is the process of determining motion vectors that describe the transformation from adjacent frames in a video sequence. In literature, the term motion estimation is often known as optical flow estimation. We use these two terms interchangeably. In our video inpainting method, optical flow is used at various points. It is key for both static (Chapter 4) and dynamic background (Chapter 5) cases.

We thus recall certain important notions of this domain. We start with some definitions, then review some classical approaches and finally focus on CNN-based optical flow estimation method which is used in our thesis.

**Introduction and formulation** In literature, optical flow is a 2D vector field which measures apparent motion between two consecutive frames caused by the movement of object or camera. Each vector is a displacement vector showing the movement of points from the first frame to the second (Figure 2.20). It is a classical problem, and it frequently appears in many domains with different applications such as object tracking, video compression, structure from motion, among others.

Optical flow is an ill-posed problem, and therefore, in several optical flow methods, different assumptions are made. The common assumption is the brightness constancy assumption which assumes that the pixel intensities of an object do not change between consecutive frames.

Let us consider the content of a pixel  $p = (x, y, t)$  in the first frame. It moves by distance  $(\Delta x, \Delta y)$  in next frame taken after  $\Delta t$  time. The brightness constancy assumption states that the content of the pixel  $p$  should be found at another position  $(x + \Delta x; y + \Delta y)$  at time  $t + \Delta t$ . This is formalized by the following equation:



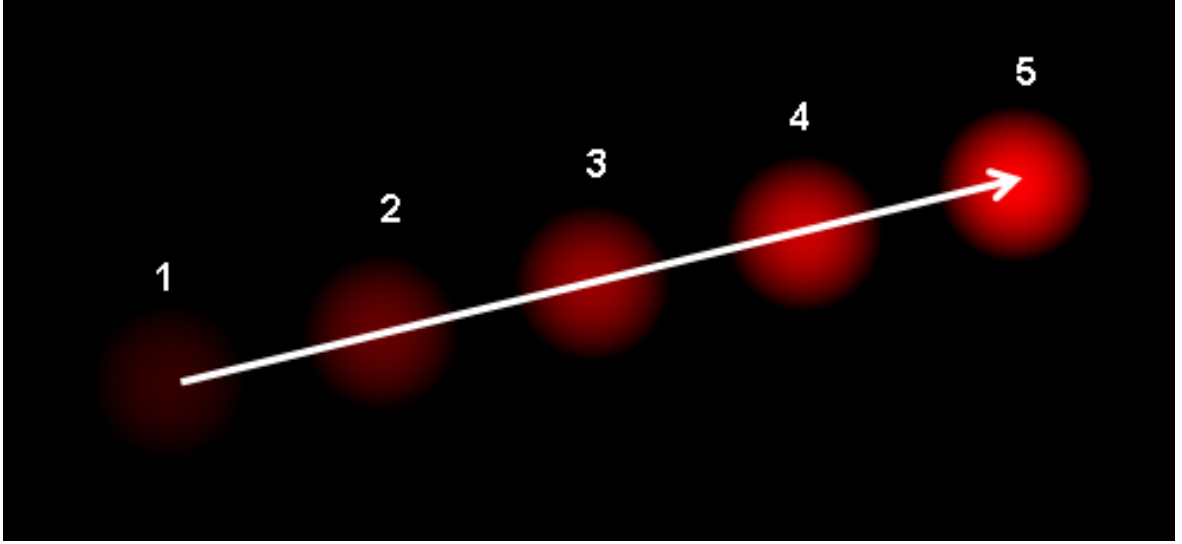


Figure 2.20 – The optical flow vector of a moving object in a video sequence. Image credit: [Wikipedia](#).

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t).$$

By taking the Taylor series approximation of the right-hand side, removing common terms and dividing by  $\Delta t$ , we get the following equation:

$$f_x u + f_y v + f_t = 0$$

where:

$$f_x = \frac{\partial f}{\partial x}; f_y = \frac{\partial f}{\partial y}$$

$$u = \frac{dx}{dt}; v = \frac{dy}{dt}$$

This equation is called the Optical Flow equation and  $u$  and  $v$  are the components of the optical flow vector, which we would like to estimate. Since this equation has two unknown variables, it is impossible to solve this. Therefore, many methods try to solve this equation by adding other assumptions. A classical and well-known method is Lucas-Kanade, which is described in the following paragraph.

**The Lucas-Kanade method (Lucas, Kanade, et al. 1981)** The Lucas-Kanade method (Lucas, Kanade, et al. 1981) adds one more assumption to the equation, which says that neighboring pixels have similar motion. They take a  $(3 \times 3)$  patch around the point  $p$  and find  $(f_x, f_y, f_t)$  for these 9 points. Since they all have the same motion, the problem becomes solving 9 equations with two unknown variables  $u$  and  $v$ , which is over-determined. A least-square fit method is applied to obtain the best solution.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i f_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix}$$

**Other classical approaches** The idea behind the Lucas-Kanade method ([Lucas, Kanade, et al. 1981](#)) is simple: give the algorithm some points to track and receive optical flow vectors of those points by solving the equation. It computes the optical flow for a sparse feature set. However, for some applications, a dense optical flow is needed. Therefore, in the literature, several works have tried to compute dense optical flows by introducing further assumptions. The most common hypothesis is spatial smoothness, which stipulates that the optical flow vector field should be as smooth as possible. This was carried out by ([Horn and Schunck 1981](#)).

These classical approaches usually fail under large motion. To deal with this problem, a multi-resolution scheme is often used in order to reduce the scale of the motion, and the optical flow equation is first minimized at a coarse pyramid level. The resulting solution is then upsampled, and the first of the two images at the finer level is warped to the second one. The same procedure is applied to the warped image, and the resulting motion vectors are simply the sum of the initial (coarse) vectors and the fine vectors.

We note that the literature in motion estimation is vast and providing the literature about optical flow is out of our scope. For a complete recent review of optical flow. We refer the readers to ([Fortun, Bouthemy, and Kervrann 2015](#)).

**CNN-based Optical flow estimation** Classical optical flow calculation approaches often suffer from accuracy-speed trade-off. An accurate optical flow estimation method often has high complexity cost, making it inapplicable in practice. Recently, inspired by the recent success of deep convolutional neural networks in various computer vision tasks, many researchers have used CNN to estimate optical flow. Compared with traditional methods, these methods achieved a significant improvement in quality while taking only milliseconds to perform. Therefore, it becomes a replacement in most of the modern techniques. Here, we focus on *FlowNet* and its variants, which is the method used in our work.

In the literature, several approaches to compute optical flow based on CNN are proposed. For example, in ([Güney and Geiger 2016; Zweig and Wolf 2017](#)), CNN is used to extract deep features of the input images. These features are then integrated into common optimization algorithms to calculate the optical flow. Estimating optical flow this way can achieve high accuracy, but the optimization process often has high complexity. Other methods try to learn the corresponding map between two images in an unsupervised manner, using unlabeled image pairs ([Z. Ren et al. 2017; L. Fan et al. 2018](#)). Although this approach does not require labeled data for training, the performance of the unsupervised method is inferior.

A powerful way to estimate the optical flow directly is by end-to-end supervised methods. In this way, the whole optical flow process is performed by the CNN (Figure 2.21). This

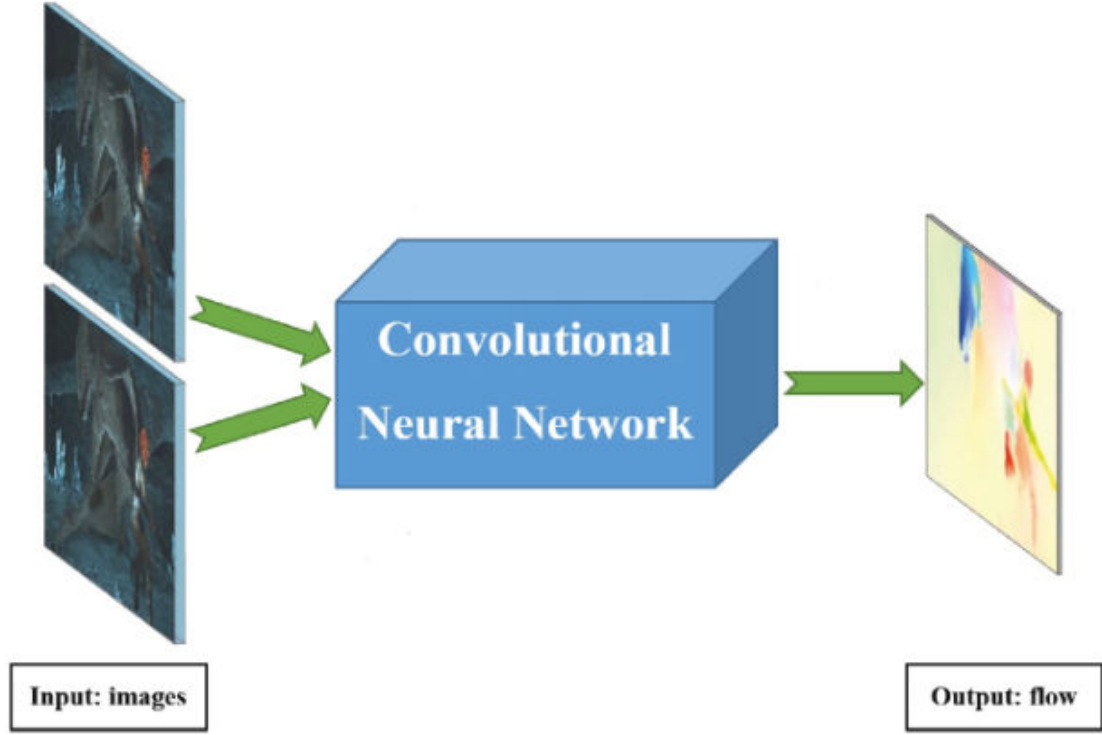


Figure 2.21 – End-to-end CNN learning model for optical flow estimation. Image taken from (Zhigang Tu et al. 2018).

method yield better performance, both in terms of accuracy and efficiency. The major problem of these methods is that they require a large amount of ground-truth optical flow to train. The method in (Dosovitskiy et al. 2015) overcomes this problem by introducing synthetic annotated data. They were pioneers in the use of end-to-end training scheme to learn the optical flow from this data. Their architecture called *FlowNet* outperforms classical approaches in term of accuracy while only taking milliseconds to compute.

In their original version, (Dosovitskiy et al. 2015) proposed two networks: *FlowNetSimple* (*FlowNetS*) and *FlowNetCorr* (*FlowNetC*). The architecture of these two networks can be seen in Figure 2.22. While in *FlowNetS*, two input images are simply stacked together and then fed to the network, in *FlowNetC*, two images are processed by two branch of the network separately to produce feature representations and are combined by a *correlation layer* to perform patch comparison. More specifically, given two feature maps  $f_1, f_2$ , with width  $w$ , height  $h$ , and  $c$  channels, the correlation of two patches  $x_1$  in the first map and  $x_2$  in the second map is defined as:

$$c(x_1, x_2) = \sum_{o \in [-k, k] \times [-k, k]} \left\langle f_1(x_1 + o), f_2(x_2 + o) \right\rangle$$

where  $x_1$  and  $x_2$  are the center of the first map and the second map respectively, and the square space patch of size  $K = 2k + 1$ . To deal with the problem of reducing the resolution, the authors propose an U-net like architecture (Ronneberger, Fischer, and Brox 2015). They

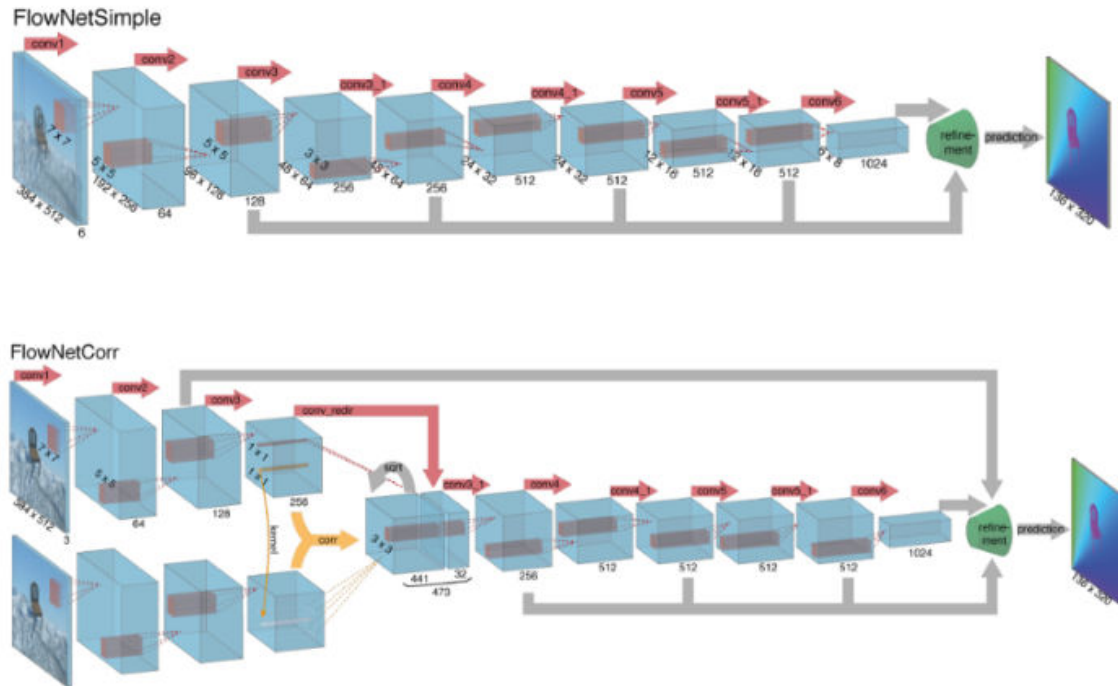


Figure 2.22 – Different network architectures of *FlowNet*. Top: *FlowNetSimple*, bottom: *FlowNetCorr*. Image taken from (Dosovitskiy et al. 2015).

refine the coarse pooled representation by "upconvolution" layers, consisting of unpooling and upconvolution. After upconvolutioning the feature maps, the authors concatenate it with corresponding feature maps and an upsampled coarse flow prediction.

This method is further extended by in *FlowNetv2* (Ilg et al. 2017). Compared with *FlowNet*, *FlowNetv2* has a large improvement in term of quality as well as speed. This success is achieved through different components (1) A better strategy of dataset schedules; (2) the stacking of multiple encoder-decoder networks; (3) the introduction of a sub-network focused on small, sub-pixel motion and (4) a new fusion architecture. More specifically, for training, the authors realize that training the network on Chairs and then fine-tuning on Things3D dataset yields the best performance. Next, the multiple encoder-decoder networks stacking scheme is introduced to deal with large displacement. They stacked *FlowNetS* and *FlowNetC* as is shown in Figure 2.23. The first *FlowNetC* gets the image  $I_1$  and  $I_2$  as input, and the second and the third *FlowNetS* takes image  $I_1$ , optical flow  $W_i$  computed by the first *FlowNetS*, image  $I_2$  warped by optical flow  $W_i$  and the brightness different error between image  $I_1$  and image  $I_2$  warped by the optical flow  $W_i$ . This is called *FlowNet-CSS*. To deal with small displacement, the authors introduced *flowNetSD*, by training *flowNetS* using a small dataset with small displacements. In *flowNetSD*,  $(7 \times 7)$  and  $(5 \times 5)$  kernels in the beginning are replaced by multiple  $(3 \times 3)$  kernels. Finally, a fusion network is used to fuse the output of *FlowNet-CSS* and *FlowNet-SD*.

*FlowNetv2* has very high performance in term of accuracy and speed. In Sintel (Butler et al. 2012) and Middlebury (Scharstein et al. 2014) dataset, *FlowNetv2* surpasses all the



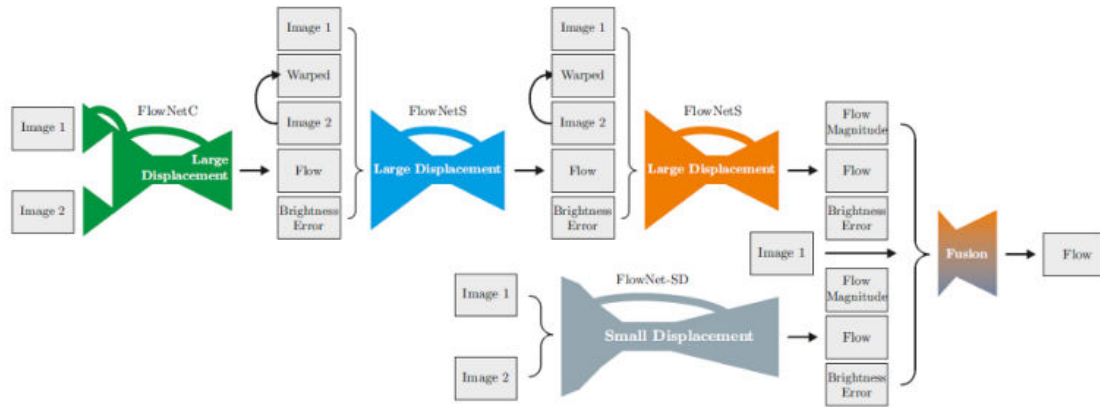


Figure 2.23 – *FlowNetv2* architecture. Image taken from (Ilg et al. 2017).

other reference methods in terms of accuracy, and it also performs well in other datasets with relatively high accuracy rate.



# 3

## Video Objects Segmentation

### Contents

---

<b>3.1</b>	<b>Introduction</b>	<b>40</b>
3.1.1	What is video objects segmentation?	40
3.1.2	Applications	42
3.1.3	Challenges	43
3.1.4	Video segmentation dataset	44
<b>3.2</b>	<b>Literature reviews</b>	<b>45</b>
3.2.1	Classical video segmentation	47
3.2.2	Deep learning-based objects segmentation.	54
	Semantic segmentation	54
	Video objects segmentation	61
<b>3.3</b>	<b>Method</b>	<b>69</b>
3.3.1	Choosing the best method for video objects segmentation	69
3.3.2	Overview	71
3.3.3	Segmentation networks	72
	Multi-OSVOS network	72
	Mask refining network	77
3.3.4	Multiple object masks tracking	79

3.3.5	Masks linking . . . . .	83
3.3.6	Post-processing . . . . .	86
<b>3.4</b>	<b>Experimental Results . . . . .</b>	<b>87</b>
3.4.1	Implementation details . . . . .	87
3.4.2	Evaluation metrics . . . . .	88
3.4.3	Evaluation . . . . .	89
	Single object segmentation . . . . .	90
<b>3.5</b>	<b>Concluding remarks . . . . .</b>	<b>99</b>
3.5.1	Limitations . . . . .	99
3.5.2	Conclusions and future works . . . . .	100

Inspired by recent advances of deep learning in object segmentation and tracking, in this chapter, we introduce the concept of using Convolutional Neural Networks (CNNs) to segment the objects in the video. We focus on using a video objects segmentation algorithm to create a mask for the video objects removal application. We demonstrate that highly accurate object segmentation in videos can be obtained by using a CNN trained with annotation on the first frame only. The key components of our approach are two-fold: (1) a combination of CNN-based segmentation networks and a classical tracking method, and (2) a classical graph-based data association strategies. Where the former produces good mask candidates for each object, the later helps in choosing the best masks among different candidates. We also introduce a way to adapt the video segmentation algorithm to our purpose, which is creating an input mask for the video inpainting algorithm. As a result, the proposed system is suitable for object removal applications with different requirements in terms of accuracy and efficiency.

## 3.1 Introduction

### 3.1.1 What is video objects segmentation?

Video segmentation is a fundamental task in computer vision which separates each frame in the video into two or more specific objects and a background. It comprises a variety of different tasks. These tasks can be categorized differently based on different aspects such as the level of supervision or the desired output of the algorithm.

Based on the level of supervision required by the user, video segmentation can be divided into 3 groups: **unsupervised approach**, **interactive approach** and **semi-supervised approach**.

- **Unsupervised approach:** This approach is fully automatic and no manual annotation is needed in this setting. It requires no prior knowledge about the location or shape of the objects. Unsupervised methods aim to group pixels consistent in both appearance and

motion and extract the most salient spatio-temporal object. These methods have their own advantages and disadvantages. A common advantage is that they do not need data labeling, resulting in decreased costs. Moreover, they do not depend on any particular dataset, leading to a more flexible and adaptable system, hence they can be used as auxiliary methods along with other supervised methods. However, the computational cost of these methods is often very large and the accuracy score is generally lower than supervised methods.

- **Interactive approach:** Unsupervised methods currently have difficulties in generating accurate segmentations and creating meaningful labels when dealing with difficult situations in videos such as multiple moving objects and cluttered background because they do not have any knowledge of the target object. To bring more information to the algorithm, some methods require user intervention, and manual annotations are needed. These annotations can have different forms such as manual pixel-level segmentation, bounding box around objects, rough strokes or clicks on the foreground objects and the background.
- **Semi-supervised approach:** Semi-supervised video object segmentation methods often assume pixel-accurate masks for objects at the first frame are available. They aim at using these annotated masks to calculate the masks for all objects in the rest of the video. In this chapter, we focus on this approach because it gives more accurate segmentation masks and enables users to select the objects they want in the objects removal application.

Based on the differences in the desired output between different tasks, we can distinguish four categories of video objects segmentation algorithms as shown in Figure 3.1

- **Foreground/ background separation.** In this approach, the objective is to distinguish the background and foreground. All foreground objects have the same label.
- **Instance-level segmentation.** In this family of works, the goal is to separate each object from the background and other foreground objects. In the case of multiple foreground objects, the algorithm is expected to create a separate segment per object (even if they belong to the same semantic category).
- **Semantic segmentation.** Semantic segmentation predicts per-pixel semantic labels given the input image. The frame is decomposed into multiple segments. Each segment is given a semantic label corresponding to the depicted object class.
- **Over-segmentation.** These works segment each frame to a significantly larger number of segments than the number of objects present in each frame. Each segment is relatively small and homogeneous in appearance and motion. Since the segments do not correspond to entire objects, the output is typically used as an intermediate step for other algorithms.

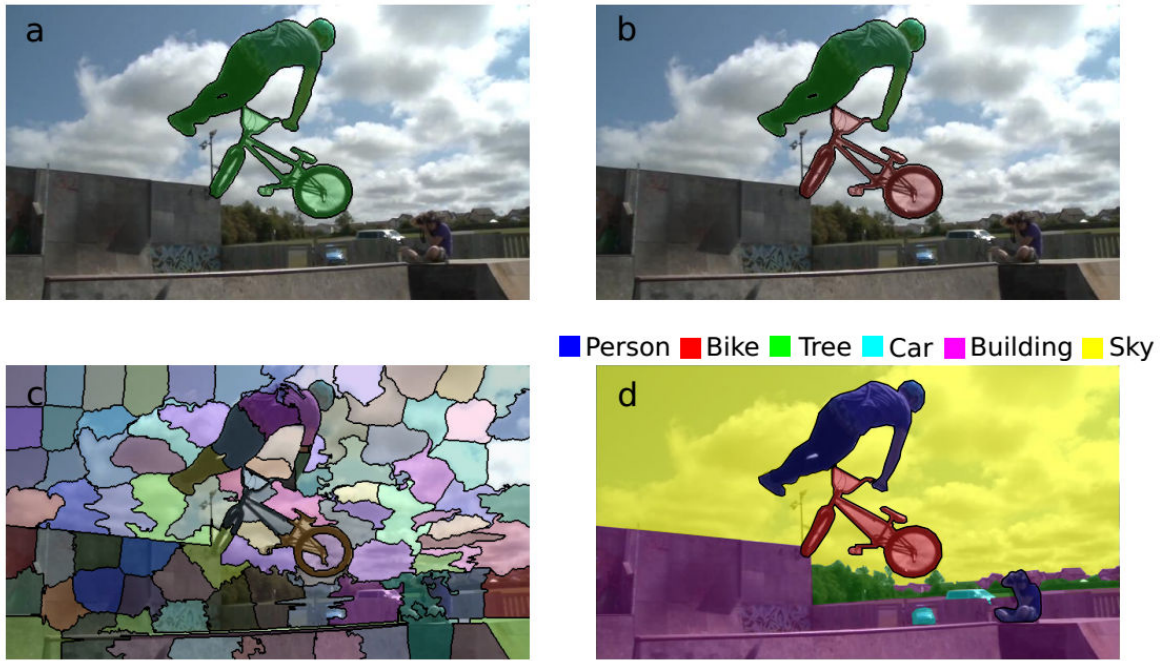


Figure 3.1 – Different categories of video segmentation algorithms based on expected output. (a) Foreground/background separation. (b) Instance-level segmentation: each foreground object is a separate segment. (c) Over-segmentation: the frame is decomposed into multiple segments. (d) Semantic segmentation: the frame is decomposed into multiple segment with a semantic label corresponding to the depicted object class. (Papazoglou 2016)

In this work we are interested in segmentation at instance level, which means we aim at separating different instances of the same object category.

### 3.1.2 Applications

A straightforward application of video objects segmentation is in media production. In this field, video segmentation is a beneficial and widely used technique. It is the basis of many real-world applications ranging from professional movie post-processing to popular mobile applications.

For instance, in the professional movie industry, since the cost of re-shooting new video footage is often prohibitive, movie post-production tasks are often used to correct errors and add visual effects. In this video post-processing task, video segmentation is key to cut out essential objects, remove unwanted objects in a scene, or apply different effects to different objects. It can save many editing times and promotes the reuse of materials, which leads to economic benefits. Because of its usefulness, many professional tools for visual effects are developed to perform this task such as Silhouette (SILHOUETTEFX 2014), Mocha (ImagineerSystems 2014) or RotoBrush in Adobe Aftereffect (Xue Bai et al. 2009).

Along with professional movie post-processing, video segmentation has been used in daily life by amateur video creators. For example, Google released Mobile Real-time Video Segmentation (Bazarevsky and Andrei Tkachenka 2018)- a new mobile application which

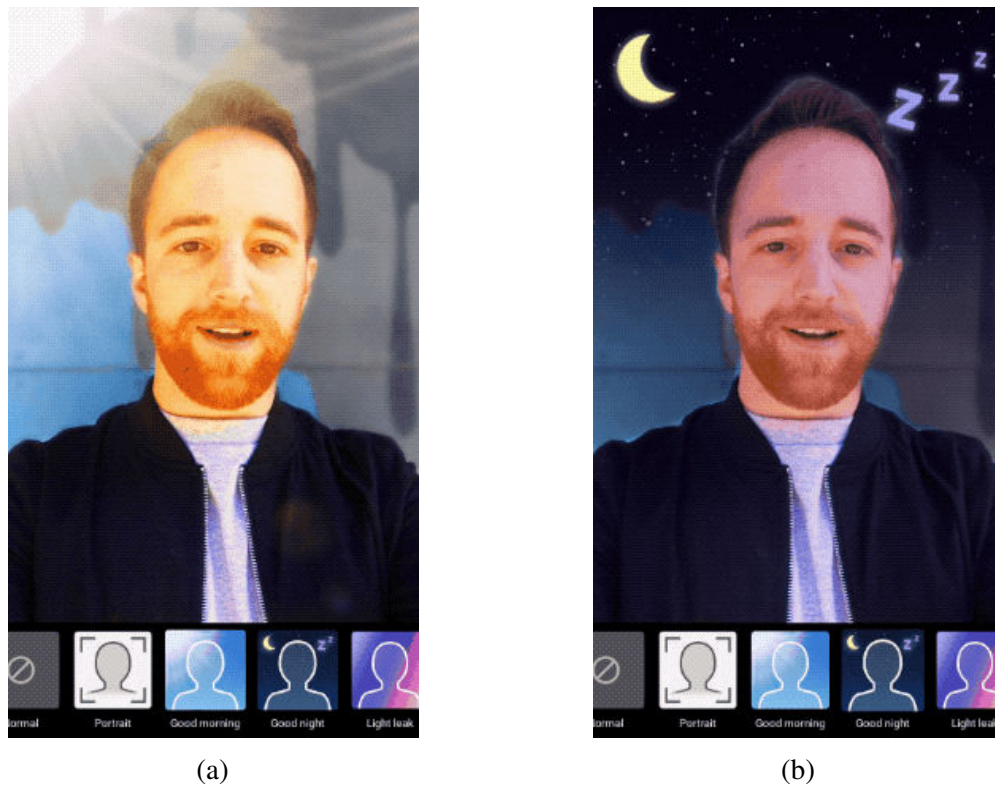


Figure 3.2 – Google’s background changing application. With this application, users can change the background easily. (a) Day background, (b) night background. Image taken from Google AI’s blog.

helps their YouTube video content creators to separate background and foreground of a video in real-time. With this application, users can replace video backgrounds with another fun location without using specialized equipment such as green screen studio, or resorting to the heavy commercial video processing production.

Beside media production, video segmentation became (thanks to its pixel-level accuracy) a very powerful tool in different practical applications such as surveillance systems, medical imaging, computer-guided surgery, object recognition, content-based browsing, self-driving cars, etc.

### 3.1.3 Challenges

Despite many years of progress both in industrial and academic research, many challenges still remain in the video segmentation field.

**Lack of large-scale annotated video datasets.** Recently, Convolutional Neural Networks (CNNs) have shown many significant improvements in the image and video segmentation field, especially supervised approaches. The performance of CNNs often depends on the availability of labeled data. Usually, CNNs require large-scale annotated datasets for training. However, video object annotation at pixel-level for segmentation is an arduous task, since it costs a significant amount of time and effort to create a sufficiently large-scale dataset

of annotated video for training. Annotating a video is more complicated than annotating an image because videos have a complicated data structure such as the codecs, temporal coherence, and high frame rate. For example, to annotate ten seconds of video, we need to annotate 300 images. Although several annotation tools have been developed to facilitate this task, it remains difficult.

**Changes of object appearance.** In unsupervised video segmentation, we may have no prior knowledge about the object appearance, size, scale or position. That makes objects segmentation very hard. In practice, the semi-supervised approach which gives the system a little information about the object (e.g. first frame object segmentation masks) is often used in video segmentation applications. However, during the video, these object appearances can change extensively in angle and direction, shape and scale, illumination, blurring and occlusion, objects overlap. Moreover, an unconstrained camera setting can result in a moving background, motion blur, large displacement or non-rigid deformations. These problems are easy to deal with for the human visual system, but they are real challenges for a computer algorithm since it is difficult to distinguish the unique discriminating parts of the target object.

**Capturing temporal information.** Videos often contain much more information than the static image. Each frame in a single video often encodes different sources of information such as views, shapes, deformations, and motion. These information helps to achieve a better separation of objects from the background. However, modelling temporal information in videos is a very challenging problem, both for classical graph partitioning techniques and modern CNNs based techniques. In graph-based techniques, the additional temporal information increases the time-complexity of the graph cut algorithm exponentially. In CNNs-based approaches, to capture temporal information, the number of parameters to be optimized increase drastically (and so does the need for a larger labelled dataset) since the network must process not just the image but multiple frames at a time. Besides, it is not clear if the temporal dimension and spatial information should be encoded similarly. Therefore, generally speaking, finding an efficient way to encode the temporal context remains an open question.

### 3.1.4 Video segmentation dataset

Data is always an essential part of any machine learning system, especially when many deep networks and complex architectures are recently developed. Thus, in a deep learning-based video segmentation system, collecting adequate data becomes crucial. Unlike video object detection or classification, constructing an appropriate dataset for video segmentation is a difficult problem. This dataset must have a scale large enough to represent the system use case and also must be pixel-wise accurate for the system to understand and learn. This task not only requires time, and an appropriate infrastructure but also domain expertise to select



the most relevant information. Nevertheless, with many efforts of researchers, experts and community, several standard datasets have been created. These standardized datasets bring many advantages to the community. They enable fair comparisons between systems and serve as a baseline for different learning applications.

Throughout the years, segmentation datasets mostly focused on images (Everingham et al. 2015; Hariharan, Arbeláez, Bourdev, et al. 2011; T.-Y. Lin et al. 2014). However, with the development of equipment, several video segmentation datasets have been created, both for semantic and instance segmentation. In this section, we describe some popular datasets for video segmentation, with different categories ranging from urban scenes to YouTube commons objects. A summary of these datasets is presented in table 3.1.

In the past, some remarkable video segmentation datasets have been created to evaluate classical video segmentation algorithms such as SegTrackv2 (F. Li et al. 2013), MoViCs (Chiu and Fritz 2013), ObMICs (Y. Yang, Sundaramoorthi, and Soatto 2015), YouTube Objects (Prest et al. 2012; S. D. Jain and Grauman 2014). These datasets are relatively simple. Moreover, all of them are in small scales which are insufficient for a deep learning-based system.

With the development of the self-driving car application, many datasets such as KITTI (Geiger et al. 2013), CamVid (Brostow et al. 2008), CityScape (Cordts et al. 2016), are constructed to understand common categories of urban scenes in a self-driving environment. These classes could be "pedestrians, vehicles, buildings, vegetation, sky..."

In 2016 F. Perazzi, J. Pont-Tuset, et al. (2016) released a new dataset called Densely-Annotated Video Segmentation (DAVIS) and it quickly became a standard benchmark in this domain, which served for yearly challenges and contests. Its 2016 version contains 50 high definition sequences with a single foreground object per video while the 2017 version has 90 videos with multiple objects per video. The video content is complicated with real-life video scenarios such as camera shake, multiple objects interactions, background clutter, occlusions, among other complexities. Frame resolution varies across sequences, and pixel-wise perfect ground truth annotations are provided for every frame.

Recently, a new large-scale video object segmentation dataset called YouTube Video Object Segmentation dataset (YouTube-VOS) (N. Xu, L. Yang, et al. 2018) was published. The dataset contains 4453 YouTube video clips and 94 object categories. This is by far the largest video object segmentation dataset that has been released to date. The purpose of this dataset is to solve the problem of data lacking and to enable the advancement of sequential learning algorithms such as Recurrent Neural Networks (RNNs) to explore spatio-temporal features for video segmentation.

## 3.2 Literature reviews

Video segmentation is an important step in many applications such as video summarisation (Y. J. Lee, Ghosh, and Grauman 2012; X. Zhu, Chen Change Loy, and Gong 2016), action

Name	Type	Object	Description
KITTI <sup>a</sup> , CamVid <sup>b</sup> , CityScape <sup>c</sup>	Semantic segmentation	Common objects in street scenes.	City street scenes videos are collected using the cameras mounted in a moving car and manually annotated.
SegTrackv2 <sup>d</sup>	Single & multiple objects	Common categories: car, bird, human...	14 videos with a single object or interacting objects presented in each video. All frames in each video are fully annotated.
MoVICS <sup>e</sup>	Multiple objects seg- mentation.	Animals in the wild.	4 video sets with 11 weakly labeled videos. 5 frames of each video are labeled with the pixel-level ground-truth.
ObMIC <sup>f</sup>	Multiple objects	Human, animal & cartoon character.	8 videos (2 videos in each group) including 2 objects in each video. Annotations are provided for all frames.
Youtube Objects <sup>g</sup>	Single & multiple objects	10 Youtube objects categories	126 challenging web videos from 10 object categories with more than 20,000 frame. Ground truth is manually segmented by (S. D. Jain and Grauman 2014) every 10 <sup>th</sup> frame.
DAVIS 2016 <sup>h</sup>	Single object	Prominent moving objects	50 high quality video sequences of diverse object categories with 3400 densely annotated, pixel-accurate frames. The videos are unconstrained in nature and contain challenges such as occlusions, motion blur, and appearance changes.
DAVIS 2017 <sup>i</sup>	Multiple objects	Diverse object categories	An extension of DAVIS 2016 with separated ground truth for target foreground objects which have more than one instance from the same class to allow instance segmentation.
Youtube-VOS 2018 <sup>j</sup>	Multiple objects	70+ common objects	Large-scale dataset for video object segmentation. It contains 4000+ YouTube videos and densely-sampled high-quality pixel-level annotations.

<sup>a</sup>(Geiger et al. 2013)<sup>b</sup>(Brostow et al. 2008)<sup>c</sup>(Cordts et al. 2016)<sup>d</sup>(F. Li et al. 2013)<sup>e</sup>(Chiu and Fritz 2013)<sup>f</sup>(M. Y. Yang et al. 2015)<sup>g</sup>(Prest et al. 2012; S. D. Jain and Grauman 2014; K. Tang et al. 2013)<sup>h</sup>(Federico Perazzi, Jordi Pont-Tuset, et al. 2016)<sup>i</sup>(Jordi Pont-Tuset, Federico Perazzi, et al. 2017)<sup>j</sup>(N. Xu, L. Yang, et al. 2018)

Table 3.1 – Common video segmentation dataset.

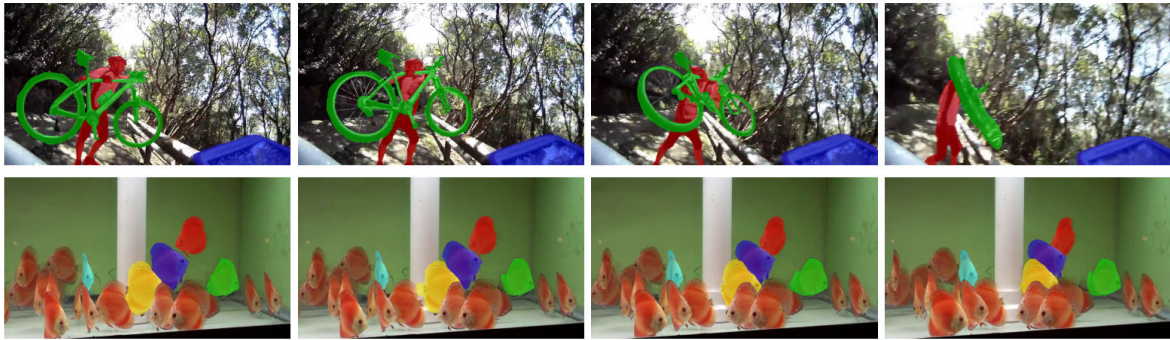


Figure 3.3 – Sample annotated frames of Youtube-VOS dataset (N. Xu, L. Yang, et al. 2018).

recognition (Kuehne, Gall, and Serre 2016) and learning object class models (Prest et al. 2012), where we need to localize the object in every frame in the video. By partitioning the video volume into groups of objects or regions which are coherent in appearance and motion, video segmentation delivers the first step to interpret the video content and thus has shown to be helpful in various computer vision tasks.

In recent years, video segmentation has received significant attention, with great progress on fully automatic methods (Ochs, Malik, and Brox 2014; Yi and Pavlovic 2015; F. Xiao and Jae Lee 2016; S. D. Jain, Xiong, and Grauman 2017; Tokmakov, Alahari, and Schmid 2016), human-guided mask propagation techniques (Tsai, M.-H. Yang, and Black 2016; Märki et al. 2016; Sergi Caelles et al. 2017), and interactive methods (S. D. Jain and Grauman 2016; L. Wang et al. 2015; Spina and Falcao 2016). In this section, we briefly review some common video segmentation techniques, ranging from classical video segmentation techniques using handcrafted features to more recent methods using deep learning.

### 3.2.1 Classical video segmentation

**Bounding box-level objects tracking.** Video objects segmentation often relates to visual tracking, where the objective is to localize the position of the objects in a video, but at bounding box level. Object tracking is a highly popular research topic in computer vision due to its crucial role in several real-world vision applications. Classic works on visual tracking perform at bounding box level for its simplicity and speed. Recently, many state-of-the-art methods mainly rely on updating across time a template model using hand-crafted (Breitenstein et al. 2009; Y. Wu, J. Lim, and M.-H. Yang 2013; LIRIS no date) or CNN-based features (Danelljan et al. 2015; C. Ma et al. 2015; L. Wang et al. 2015). Variations of correlation filters (Bolme et al. 2010; Henriques et al. 2015; M. Tang and Feng 2015), boosting based classifiers (Son et al. 2015; Kalal, Mikolajczyk, and Matas 2012) and convolutional neural networks architectures (Held, Thrun, and Savarese 2016; Bertinetto et al. 2016; Tao, Gavves, and Smeulders 2016; Nam, Baek, and Han 2016; Nam and Han 2016) are popular choices for learning and updating this template model.

In several applications such as crowd analysis or surveillance systems, it is necessary

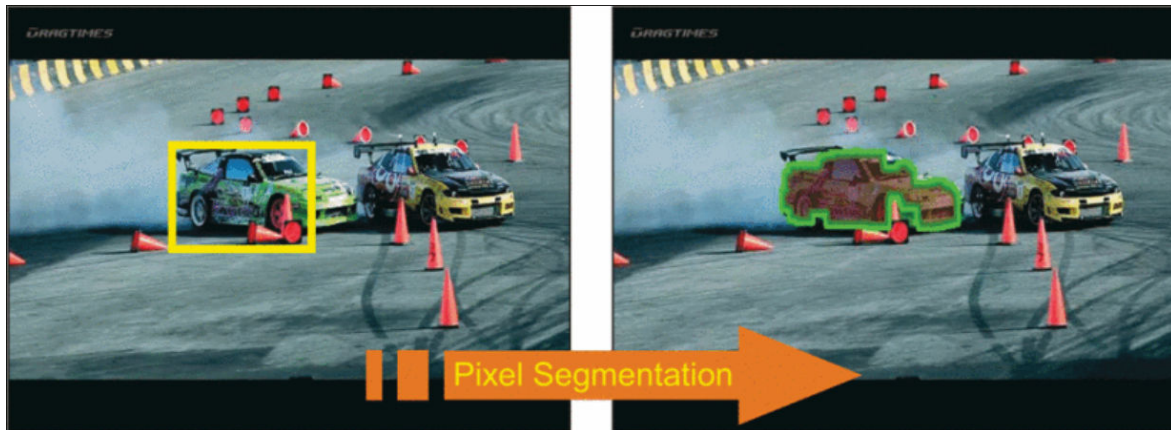


Figure 3.4 – From bounding box to pixel level: A bounding box proposal (left, yellow box) is fine-tuned and produces a pixel-level segmentation mask (right, green boundary). Image from (F. Xiao and Jae Lee 2016).

to track multiple objects across time. A common approach in multiple objects tracking is track-by-detection. In this framework, a set of detections is extracted on all images; then detections are linked together by a data association algorithm to produce smooth trajectories. The data association task often leads to graph optimization problem which can be solved by bipartite graph matching (Shu et al. 2012; Reid 1979; Pirsivash, Ramanan, and Fowlkes 2011), network flow (L. Zhang, Yuan Li, and Ramakant Nevatia 2008) conditional random field (B. Yang and Ram Nevatia 2012; Milan, S. Roth, and Schindler 2014), or generalized clique graphs (Roshan Zamir, Dehghan, and Shah 2012; Dehghan, Modiri Assari, and Shah 2015).

**Pixel-level objects tracking** Tracking at the bounding box level is insufficient for some higher-level vision tasks such as human activities recognition or content-based retrieval. These applications often require pixel-level accuracy. For this reason, many efforts have been made to track objects at the pixel-level by bringing several insights from bounding box level tracking into the field.

For example, in (Godec, P. M. Roth, and Bischof 2013; F. Xiao and Jae Lee 2016), the authors do pixel-level tracking by building a box-level tracker first, then apply a space-time graph-based algorithm such as GrabCut (Rother, Kolmogorov, and Blake 2004) to achieve accurate segments (Figure 3.4). Similarly, (Godec, P. M. Roth, and Bischof 2013) builds the bounding boxes by training the target at the first frame using Generalized Hough transforms, followed by the sparse pixels voting to segment the target using GrabCut (Rother, Kolmogorov, and Blake 2004). On the same track, (F. Xiao and Jae Lee 2016) discover and rank a set of clusters to discover easy instances of an object to build an initial model. This model is iteratively refined to discover harder instances in adjacent frames. Finally, the bounding boxes are used as a weak supervision for refining a pixel-level appearance model which is similar to GrabCut (Rother, Kolmogorov, and Blake 2004). GrabCut works by running a graph-cut based algorithm to optimize an energy based on Markov Random Field which prefers connected

regions having the same labels.

With a different approach, in (F. Li et al. 2013), objects are obtained by tracking many figure-ground segments. They compute multiple segment proposals per frame and link them across frames using appearance similarity. In (Wen et al. 2015), a joint online tracking and segmentation algorithm is proposed, which formulates the video segmentation task as online multi-part tracking and segmentation in a unified energy function. These methods only segment one object per frame. To track multiple objects at pixel-level, graph-based techniques are often adopted. For instance, (Milan, Leal-Taixé, et al. 2015) track superpixels by casting the tracking problem as a multi-labels optimization problem which is solved by conditional random fields. Similarly, (Babaee, Y. You, and Rigoll 2016) simultaneously segment, reconstruct, and track targets in a multi-view setup. They solve the graph by integer linear programming. Recently, (Seguin et al. 2016) extend the work in (F. Xiao and Jae Lee 2016) to multiple objects by using the bounding box for each target as a constraint in a graph-based optimization problem. Although many good results have been shown, these methods are far from the capability of extracting accurate instances.

**Fully automatic video segmentation.** Fully automatic (or unsupervised) video segmentation methods assume no human intervention. The algorithm usually detects and segments objects based on their saliency. Appearance information and motion cues are usually employed as a characteristic to distinguish between background and foreground. In the past decade, a variety of techniques have been proposed in the literature to perform this task.

One common approach in fully automatic video segmentation is graph-based partitioning. The process is as follows: In the first step, features are extracted to represent video information. Next step is the construction of a graph according to the spatio-temporal neighborhood and edge weights estimation base on the computed features. Finally, a spatio-temporal clustering method is adopted as a graph-based partitioning problem. There are different forms of basic data units to construct a graph: pixels, superpixels/supervoxels or point trajectories.

A typical graph-based partitioning method is the work of (Grundmann et al. 2010). In this method, the algorithm starts by over-segmenting the video using the appearance to obtain small space-time regions similar to superpixels but in spatio-temporal dimension, followed by the construction of a graph hierarchically. Dense optical flow is used to guide temporal connections in the initial graph which can improve the coherence. Finally, the authors employ a greedy clustering algorithm that merges two adjacent superpixels base on their color differences. This method can preserve long-term temporal coherence in the identities and boundaries. However, the accurate results for some complicated videos are not achieved since the method uses only motion information instead of the spatio-temporal structure of the video.

Another method using superpixels is the work of (Galasso et al. 2013). This method performs matching of superpixels frame by frame. They use optical flow to propagate labels from the source frame over time. However, optical flow-based propagation methods often





Figure 3.5 – Unsupervised video segmentation based on point trajectories clustering. Left: A sample frame input. Right: Long term motion analysis is used to separate the man, telephone receiver from the background. Image from (Brox and Malik 2010)

suffer from the accumulated errors when the distance from the source frame increases. Besides, this segmentation propagation approach cannot introduce new objects as the label set is fixed based on the source frame.

Another way of unsupervised video segmentation is based on point trajectories clustering (Brox and Malik 2010; Sundaram and Keutzer 2011; Fragkiadaki, G. Zhang, and Jianbo Shi 2012; Ochs, Malik, and Brox 2014; W. Wang and Bing 2017). In these methods, instead of modeling a video as a 3D volume, interest points are tracked throughout the video to form trajectories. These point trajectories are used to integrate motion information available in multiple frames; then they are analyzed to form separate moving objects (Figure 3.5).

In particular, Brox and Malik (2010) group pixels with coherent motion computed via long-range motions vector from the past and future frames to create point trajectories. An affinity matrix between pairs of trajectories is then built, and spectral clustering is applied using this matrix to obtain separated objects. The underlying assumption of this pairwise clustering is that all object points move according to a single translation. However, this assumption is not valid in the presence of non-rigid or articulated objects.

In (Fragkiadaki, G. Zhang, and Jianbo Shi 2012), dense point trajectories are obtained by linking optical flow fields of consecutive frames in both forward and backward direction. Similar to (Brox and Malik 2010), long-range similarities of point trajectories are employed. Moreover, discontinuities of embedding density between spatially neighboring trajectories are detected. Their metric of affinities incorporate large time intervals and can correctly delineate objects even if they move similarly for a subset of frames. Different from the method in (Brox and Malik 2010) which uses a pairwise model, Ochs, Malik, and Brox (2014) propose to incorporate higher order motion models which they call the triplet model. This model allows them to have a single similarity transformation instead of rigid motion.

For the graph-partitioning problem, most of these techniques use the well-known spectral

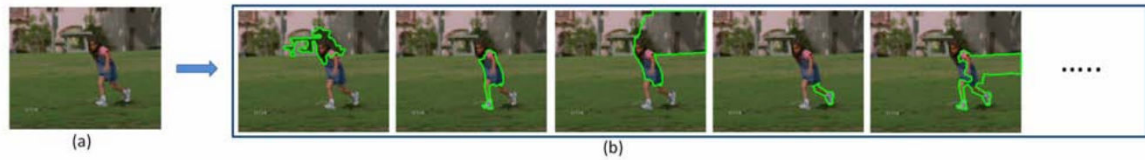


Figure 3.6 – Object proposals produced by (T. Ma and Latecki 2012). (a) A sample video frame. (b) Proposals ranked in order of "objectness". Image taken from (T. Ma and Latecki 2012)

clustering paradigm. Spectral clustering for segmentation become popular since the seminal work of (Jianbo Shi and Malik 2000) which use a normalized cut approach to do video segmentation. Since then, spectral methods have received much attention and proven to be suitable for video segmentation due to its ability to include long-range affinities (Fragkiadaki, G. Zhang, and Jianbo Shi 2012; Sundaram and Keutzer 2011; Ochs, Malik, and Brox 2014). However, the major limitation of spectral methods is the computational complexity, limiting their extension to high-quality video. Therefore, several attempts have tried to replace spectral clustering. For example, the work of (Keuper 2017) has shown the advantages of casting the motion trajectory segmentation as a minimum cost multi-cut problem. In general, since these approaches rely on point-trajectories, they suffer from many problems such as occlusion or large displacements.

The main limitation of graph partitioning approaches is the lack of an explicit notion of object appearance. By using only low-level information, they tend to over-segment videos. While this can be a useful intermediate step for some high-level video processing tasks, the extracted segments might not directly correspond to objects, making it non-trivial to obtain object masks.

Beside graph partitioning, another way of solving the fully automatic video segmentation problem is by using object proposals (Banica et al. 2013; Y. J. Lee, J. Kim, and Grauman 2011; T. Ma and Latecki 2012; D. Zhang, Javed, and Shah 2013; Oneata et al. 2014; M. Jain et al. 2014; Fragkiadaki, Arbelaez, et al. 2015; Z. Wu et al. 2015; F. Xiao and Jae Lee 2016; W. Wang, Jianbing Shen, and Porikli 2015). Under this approach, the "objectness" properties of the object are explored to create a set of considerable potential object proposals, then each proposal is ranked, and the most appropriate object is selected. (Figure 3.6).

In (Y. J. Lee, J. Kim, and Grauman 2011), the method first identifies object-like regions using both static and dynamic cues. A series of binary partitions among those candidates are then built. Finally, pixel-level object labeling across all frames is estimated through ranked hypotheses. Similarly, the authors in (Banica et al. 2013) compute multiple segment proposals per frame and link them across frames using appearance similarity. Oneata et al. (2014) produce multiple video segments by deleting image boundaries that do not exhibit high flow strength and is upper-bounded by the static boundary detector.

The method in (M. Jain et al. 2014) computes spatio-temporal region proposals from an

independent motion evidence map, which estimates for each pixel in each frame the likelihood that its motion is different from the dominant motion. In (Fragkiadaki, Arbelaez, et al. 2015) spatio-temporal segment proposals are created according to the "moving objectness". These segments are ranked with a Moving Objectness Detector trained on image and motion fields.

In the work of (Z. Wu et al. 2015), image segment proposals are generated and tracked using learned appearance models. Forward tracking and backward tracking schemes are used to track segments starting from every frame and through complete occlusions. Besides, some other segmentation methods have been proposed, those methods using non-local consensus voting (Faktor and Irani 2014) or geodesic distances (W. Wang, Jianbing Shen, and Porikli 2015) to localize the foreground regions.

These methods can, in general, have an overall notion of an object and can distinguish well between background and foreground. However, these proposals-based techniques have high computational complexity, and their dependency on a large number of proposals leads to considerable difficulty and complexity for the selection process.

To summarize, although unsupervised video segmentation has certain benefits, it often does not have an accurate boundary, and the algorithm is valid only if the objects have notably different characteristics.

**Semi-supervised video segmentation.** Semi-supervised video segmentation often requires user intervention to provide rough annotation for guiding the application to which objects should be segmented. These techniques are advantageous in video editing application, where users can manually annotate which objects they want to select at the first frame and then that annotation is propagated through the video to form a mask. Even though many encouraging results have been achieved in the past few years, this task remains challenging due to many problems such as object deformation, occlusion, background clutter, appearance change, among others. Given the annotations at first frame, the segmentation masks of the object can be obtained by either the online or offline approach.

In the online approach, the mask is propagated using the current frame and previous frames based on appearance similarities and motion smoothness across the temporal direction. One approach is to segment the video into superpixels and determine the labels for these superpixels based on tracking (Shu Wang et al. 2011) or using a graphical model (Papazoglou and Ferrari 2013; Tsai, M.-H. Yang, and Black 2016). In (Shu Wang et al. 2011), the authors adopt a tracker to distinguish the target and the background with mid-level cues which aims to handle changes in scale, motion, and shape with occlusion. After extracting the descriptor of the superpixels, they group them by mean-shift clustering. The tracking task is then formulated by computing a target-background confidence map, and the best candidate is obtained through posterior maximization.

Similarly, the label propagation method in (Badrinarayanan, Galasso, and Cipolla 2010) jointly models appearance and semantic information. This label propagation scheme is based



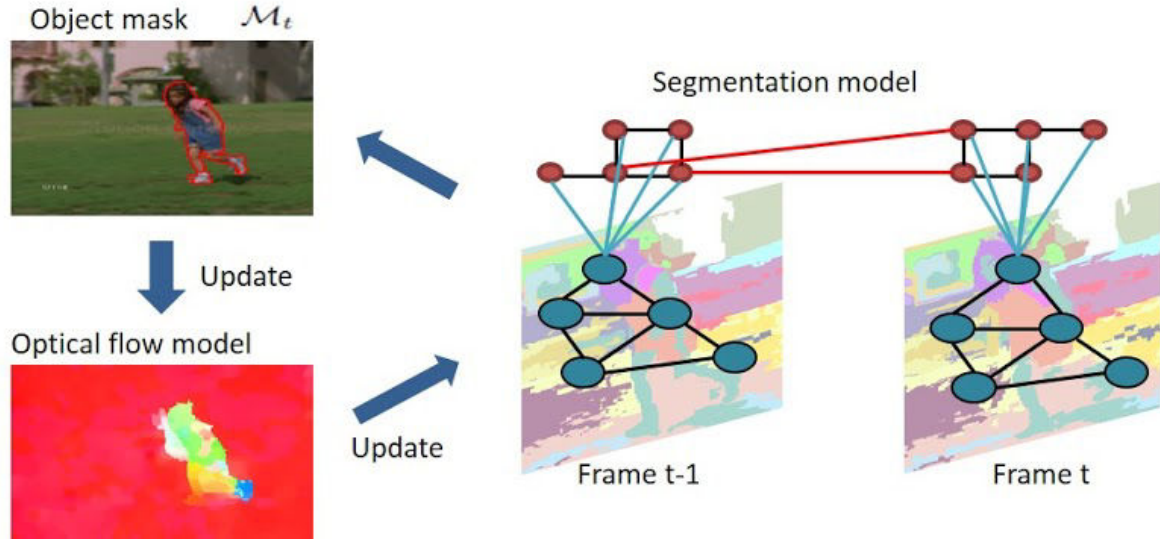


Figure 3.7 – Overview of ObjectFlow, which is proposed in (Tsai, M.-H. Yang, and Black 2016). The authors consider a multi-level spatio-temporal model for segmentation. Pixels (red circles) and superpixels (turquoise circles) are both used. The turquoise lines denote the relationships between the pixels and the superpixel. After obtaining the object mask, the optical flow is re-estimated to update both models iteratively. Image taken from (Tsai, M.-H. Yang, and Black 2016).

on pixel-wise correspondences obtained from motion estimation, patch-based similarities coupled with an Expectation Maximization-like optimization process.

Different from label propagation methods, there exist several methods using a GrabCut-like algorithm to propagate the mask. Typical work under this approach is that of (Papazoglou and Ferrari 2013) in which a superpixels labeling problem is solved by minimizing a pre-defined energy using a conditional random field model. This energy includes a unary term calculated by a Gaussian Mixture model and the pairwise term based on spatial and temporal consistency.

Recently, (Tsai, M.-H. Yang, and Black 2016) extend the work in (Papazoglou and Ferrari 2013) by combining both the superpixel-level model and the pixel-level model simultaneously. In particular, they build a graphical model that consists of both pixels and superpixels (Figure 3.7). The purpose of the superpixel-level is to distinguish between objects and background so that object boundary may not be accurate. At the pixel level, these boundary errors are corrected to produce more accurate details. By exploiting both statistics contained in the superpixel level and details in the pixel level, the object boundary can be better identified. They also adopt the middle layer of a CNN as features to train a classifier for distinguishing between background and foreground. These features are combined with optical flow information. Therefore, their segmentation results attain higher accuracy.

All of these methods as mentioned above often leverage optical flow to force the temporal coherence. However, the optical flow estimation algorithm is sometimes unreliable especially for texture-less regions, large displacements or non-rigid motion, and also has a high computational complexity. Many works replace optical flow by computing the nearest

neighbor field between frame using PatchMatch-liked algorithm ([Ramakanth and Babu 2014](#); [Q. Fan et al. 2015](#)). These methods have remarkable results in preserving spatial coherency while running in a reasonable time span. Another powerful method is found in ([Xue Bai et al. 2009](#)) which uses localized classifiers around the boundary to provide proper segmentation and prevent the changing of the annotation if the user adjusts a small region in the mask. This algorithm is integrated into a commercial product (Rotobrush tool in Adobe After Effect) and appears to be very useful. However, it requires considerable user effort to refine the mask.

In general, the main drawback of the online mask propagation method is that the error will be propagated through time. Thus, it is not robust to occlusion and not applicable if the objects cross each other.

Besides online mask propagation, another well-studied approach is the offline approach which treats the video as a tube. In this approach, graph-based models that connect all frames are often adopted. In ([S. D. Jain and Grauman 2014](#)), the authors use a Markov Random Field to cluster supervoxels - a 3D spatio-temporal region with uniform motion and similar appearance to form regions. Similarly, the authors in ([D. Sun et al. 2013](#); [Federico Perazzi, O. Wang, et al. 2015](#)) extend the local model in the temporal direction across all the frames by a fully connected graph. To balance local and global model, Zhong et al. ([2012](#)) combine multiple classifiers while Märki et al. ([2016](#)) encode color, position and time into a 6D feature before performing segmentation in this high dimensional space.

### 3.2.2 Deep learning-based objects segmentation.

In the last few years, deep learning added a massive boost to many different fields of computer vision. The field of image and video segmentation is not an exception. Recently, image and video segmentation have achieved significant progress thanks to deep learning. In this section, we briefly review the domination of deep learning in the image and video segmentation field. In particular, for image segmentation, we focus on two critical tasks: semantic segmentation and instance segmentation. These tasks are baselines for many state-of-the-art video segmentation algorithms. We then concentrate more on video objects segmentation, which is our primary interest.

#### Semantic segmentation

Semantic segmentation is a common type of segmentation where each pixel in the image is assigned to a specific semantic class. The goal of semantic segmentation is to recognize the object and provide dense pixel-wise predictions as an output. Semantic segmentation has many real-life applications. The most popular and straight forward application is autonomous driving ([Treml et al. 2016](#); [Ros et al. 2016](#)) where semantic segmentation helps us understand the urban environment by categorizing the scenes into different classes such as road, sky, tree, human, car. Beside autonomous driving, applications for semantic segmentation may include

industrial inspection (Bian, S. N. Lim, and N. Zhou 2016; Z. Xiao et al. 2018), classification of terrain visible in satellite imagery (Henry, Azimi, and Merkle 2018) or medical image analysis (Winkens et al. 2018).

Before deep learning dominates the field of computer vision, most of the successful semantic labeling methods relied on hand-crafted features combined with flat classifiers, such as random forests (Shotton, Johnson, and Cipolla 2008), boosting (Shotton, Winn, et al. 2009; Zhuowen Tu and Xiang Bai 2010) or support vector machines (Fulkerson, Vedaldi, and Soatto 2009). Despite several improvements by integrating richer information (Carreira et al. 2012; George 2015) or structure prediction (Krähenbühl and Koltun 2011; Gould, Fulton, and Koller 2009; Carreira et al. 2012), these shallow models have always been restricted by the limited power of the extracted features. To further boost the performance of the model, it is essential to extract deep features using deep architecture. Since the success of deep learning in image classification, many recent works on semantic segmentation employ the top convolutional layers on a ImageNet pre-trained CNNs architecture such as AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGGNet (Simonyan and Zisserman 2014), ResNet (He, X. Zhang, et al. 2016) or GoogLeNet (Szegedy et al. 2015) as the feature extraction module to obtain richer feature representations (Girshick et al. 2014; Caesar, Uijlings, and Ferrari 2015).

In 2015, the academic world witnessed a significant breakthrough in the image segmentation domain since the introduction of the Fully Convolutional Network (FCN). The insight of that approach is to transform existing classification models that are already able to learn hierarchies of features and re-purpose them to perform an image segmentation task. Well-known models such as AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG (Simonyan and Zisserman 2014) which are trained on ImageNet (Russakovsky et al. 2015), are transformed into Fully Convolutional Network by replacing the fully connected layers with convolutional ones to output a dense pixel-wise classification map. To overcome the problem of spatial reduction caused by the pooling layers, Long, Shelhamer, and Darrell (2015) extract features from intermediate layers via skip connections and up-sampling of the feature maps using fractionally-strided convolutions to produce dense per-pixel labeled outputs. This work has proven to be very useful for the image segmentation task. It is considered the milestone in this domain because it shows how the efficiency of the CNN-based approach outperform the traditional ones.

Since then, many improvements have been made, most of them focus on modifying the architecture and the upsampling method. In (L.-C. Chen, Papandreou, Kokkinos, et al. 2016; F. Yu and Koltun 2015), the authors use atrous convolution (L.-C. Chen, Papandreou, Kokkinos, et al. 2016), or dilated convolution (F. Yu and Koltun 2015) to reduce the pooling factor of the pre-trained network.

About the network architecture, in (Noh, Hong, and Han 2015; Hong, Noh, and Han 2015), encoder/decoder networks have been proposed. The encoder takes an input image and generates a high-dimensional vector to aggregate features. The decoder, stacked on top of

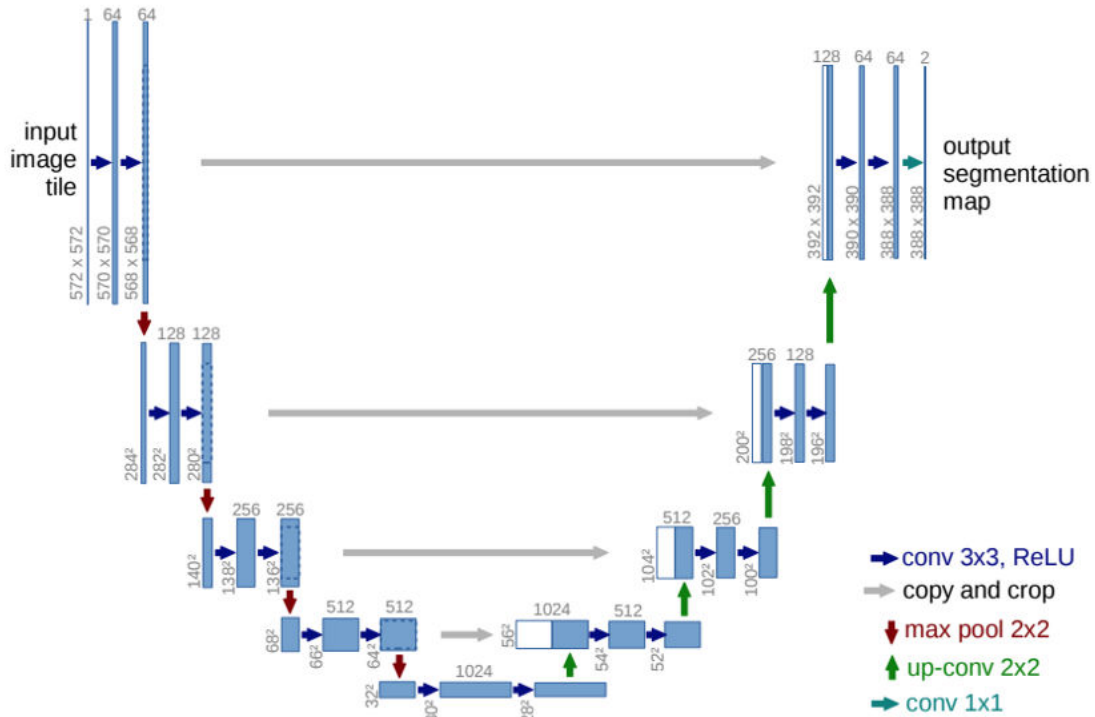


Figure 3.8 – U-net architecture. Multi-channel feature maps are described by blue boxes. White boxes represent copied feature maps. The arrows denote the different operations. Image taken from (Ronneberger, Fischer, and Brox 2015).

the encoder, then takes that high-dimensional feature vector and decodes features aggregated by the encoder at multiple levels to generate a semantic segmentation mask (Zeiler, Taylor, and Fergus 2011). Similarly, Badrinarayanan, Kendall, and Cipolla (2015) re-use the pooling indices from the encoder and learn extra convolutional layers to densify the feature responses. A notable modification is made by Ronneberger, Fischer, and Brox (2015) in which the skip connections between the encoder features path and the corresponding decoder activations are added by applying a concatenation operator instead of a sum (Figure 3.8). These skip connections intend to provide local information to the global information while upsampling which allows to transfer information from low-level features to the output activation directly.

To obtain the segmentation mask from the dense pixel-wise probability map, a simple max operation is often used. However, with this operation, the output is often not smooth and not precise in the boundary of the objects because the pooling operation limits the spatial accuracy of most of the network. An alternative solution is to apply post-processing smoothing operations to the output of a segmentation system in order to obtain more consistent predictions and fine-grained details (Krähenbühl and Koltun 2011; Barron and Poole 2016). Most commonly, Conditional Random Fields (CRFs) (Krähenbühl and Koltun 2011) are applied on the network output to capture long-range dependencies between pixels and usually improve the segmentation score (L.-C. Chen, Papandreou, Kokkinos, et al. 2016; Kokkinos 2015; G. Lin et al. 2016). CRFs are graphical models which "smooth" the segmentation

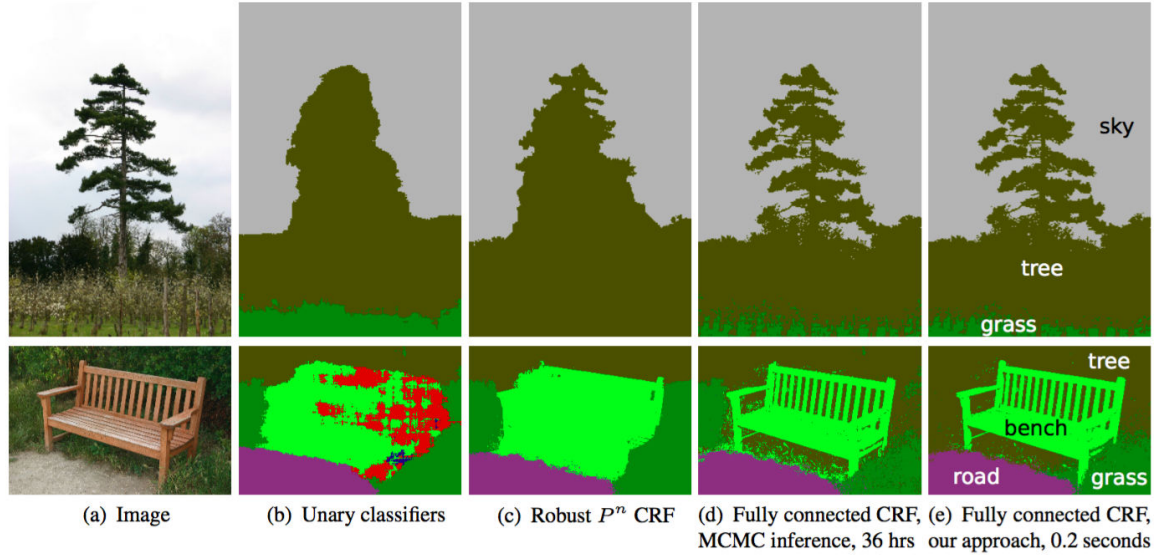


Figure 3.9 – CRF illustration. (b) Unary classifiers are the segmentation input to the CRF. (c, d, e) are common variants of CRF with (e) being the widely used one. Image taken from (Korman and Avidan 2011).

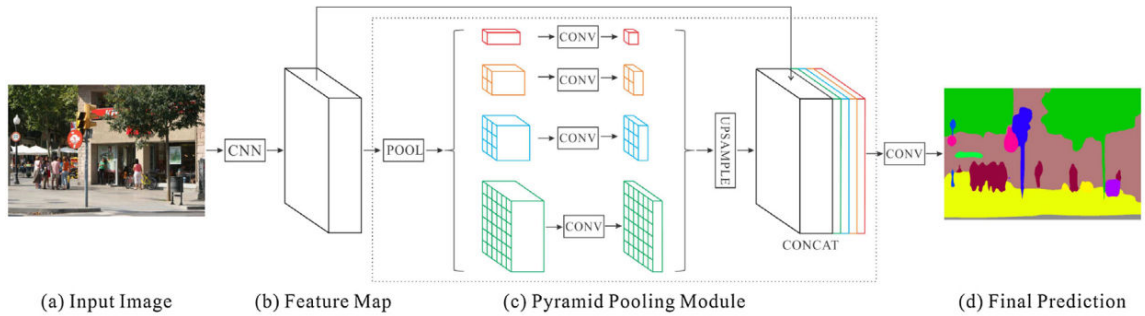


Figure 3.10 – PSPNet architecture. From input image (a), initial feature maps are extracted by a pre-trained network. The pyramid pooling module (c) covers different levels of the image to capture different levels of information. Finally, the initial feature maps are concatenated with the pooling module to generate a final predicted map. Figure extracted from (H. Zhao et al. 2017).

based on the combination of low-level image information such as the intensities and the interactions between pixels with the output probabilities map of multi-class prediction systems that produce per-pixel class. It works based on the observation that similar intensity pixels tend to be labeled as the same class. That combination is especially important to capture long-range dependencies and fine local details. CRFs can boost scores by 1 – 2%. More recently, some papers make further improvement by approximating the mean-field inference of CRFs using specialized network architectures (Zheng et al. 2015; A. G. Schwing and Urtasun 2015; Ziwei Liu et al. 2015).

Another way of improving the segmentation results is by the fusion of different context information from different image scales. A common way is using Spatial Pyramid Pooling (SPP) (H. Zhao et al. 2017; L.-C. Chen, Papandreou, Kokkinos, et al. 2016; Ziwei Liu et al.



2015). Spatial Pyramid Pooling empirically extracts global features by different region-based context aggregation. It applies the idea of spatial pyramid matching, which is a pre-existing technique in the classical computer vision method to the context of CNNs. The output feature maps are divided into a fixed number of bins with sizes proportional to the image size. Each bin captured different levels of granularity, thus capturing the context in different ranges.

This idea has given rise to numerous works. (Ghiasi and Fowlkes 2016) employ multi-scale predictions via a Laplacian pyramid reconstruction network to successively improve boundary adherence. The image-level features are exploited by Ziwei Liu et al. (2015) for global context information. H. Zhao et al. (2017) perform spatial pooling at several grid scales via a pyramid scene parsing network. Figure 3.10 shows Pyramid Scene Parsing Networks (PSPNets) (H. Zhao et al. 2017) which provide a pyramid parsing module focused into feature fusion at four different pyramid scales to embed global contexts from complex scenes.

Recently, L.-C. Chen, Papandreou, Kokkinos, et al. (2016) propose atrous spatial pyramid pooling (ASPP), where parallel atrous convolution layers with different rates capture multi-scale information. In their improved version (L.-C. Chen, Papandreou, Schroff, et al. 2017), they propose to augment the ASPP module with image-level features to encode global context and further boost performance. This approach achieves state-of-the-art performance on various datasets.

(G. Lin et al. 2016) propose a multi-path refinement network that exploits all the information available along the down-sampling process to enable high-resolution predictions using long-range residual connections.

**Instance segmentation.** As can be seen in Figure 3.11, while in semantic labeling, pixels are grouped by object class, in instance segmentation, they are categorized by object instances. The principal purpose of instance segmentation is to split objects of the same class into different instances. It is a challenging task because it requires the correct detection of all objects in an image while also precisely segmenting each instance. While semantic segmentation provides a notion for different objects, instance labeling gives us extra information about how many instances in the scenes as well as the information of any particular instance and occlusion situations. Hence, it is very useful in some applications such as detecting particular objects in robotics tasks, or counting products in a factory.

Despite its challenge, instance segmentation has been widely investigated in the past few years. Many of them employ the object proposals approach (Arbeláez et al. 2014; Jordi Pont-Tuset and Van Gool 2015; Hosang et al. 2016). For example, in (Hariharan, Arbeláez, Girshick, et al. 2014), the Simultaneous Detection and Segmentation (SDS) method has been proposed to perform instance segmentation. They start with an object candidate generation process called Multiscale Combinatorial Grouping (MCG) (Arbeláez et al. 2014) to obtain region proposals, followed by an adapted version of the Region-CNN (R-CNN) (Girshick et al. 2014) to obtain bounding boxes. These bounding boxes are used to select the appropriate

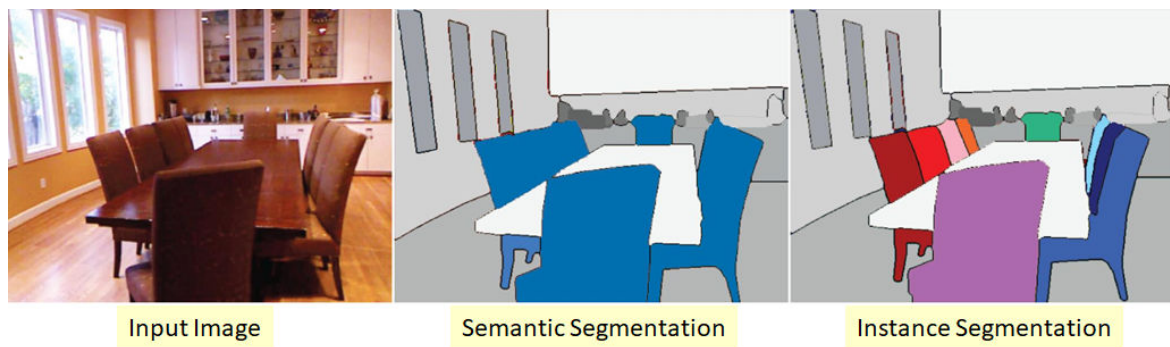


Figure 3.11 – Semantic segmentation vs instance segmentation. Figure extracted from (Pinheiro, Collobert, and Dollár 2015).

region provided by the MCG method.

Later, (Pinheiro, Collobert, and Dollár 2015) presented DeepMask - an object proposal model based on a single CNN. This model predicts how likely an image patch fully contains a centered object and also outputs an associated segmentation mask for the object. DeepMask can be seen as solving a large number of binary classification problems. For every patch, the algorithm specifies wherever this patch contains an object or not. If it contains the object then for every pixel, the algorithm classifies whether this pixel is part of the objects. A deep network is used to solve this binary classification problem, and the computation is shared for every patch and every pixel so that the object proposals are discovered and segmented quickly.

Using the DeepMask architecture as a starting point, novel architectures for object instance segmentation called SharpMask have been proposed by the same authors (Pinheiro, T.-Y. Lin, et al. 2016). SharpMask implements a top-down refinement process the output of DeepMask, generating higher-fidelity masks that more accurately near the object boundaries. The goal of this process is to efficiently merge low-level features with high-level semantic information from upper network layers. The process consists of different refinement modules stacked together (one module per pooling layer), with the purpose of inverting pooling effect by generating a new upsampled object encoding. While DeepMask predicts coarse masks in a feedforward pass through the network, SharpMask reverses the flow of information in a deep network and refines the predictions made by DeepMask by using features from progressively previous layers in the network. This way, it achieves a better performance in terms of accuracy and speed. Some example segmentation results are illustrated in Figure 3.12.

Also using the object proposals approach, Dai, He, and J. Sun (2016) propose a complex multiple-stage cascade to predict instance masks in the image. The model consists of three networks, respectively differentiating instances, estimating masks, and categorizing objects. These networks form a cascaded structure and are designed to share their convolutional features.

Another approach to instance segmentation task relies on detecting individual objects (Girshick et al. 2014; Dai, He, Yi Li, et al. 2016) to obtain bounding box, followed by a refining process to obtain pixel-level accuracy. Given a bounding box, several methods can be



Figure 3.12 – Some example outputs generated by DeepMask and refined by SharpMask. Figure extracted from [https://code.fb.com/ml-applications/segmenting-and-refining-images-with-sharpmask/?utm\\_campaign=Deep+Learning+Weekly&utm\\_medium=email&utm\\_source=Revue+newsletter](https://code.fb.com/ml-applications/segmenting-and-refining-images-with-sharpmask/?utm_campaign=Deep+Learning+Weekly&utm_medium=email&utm_source=Revue+newsletter).



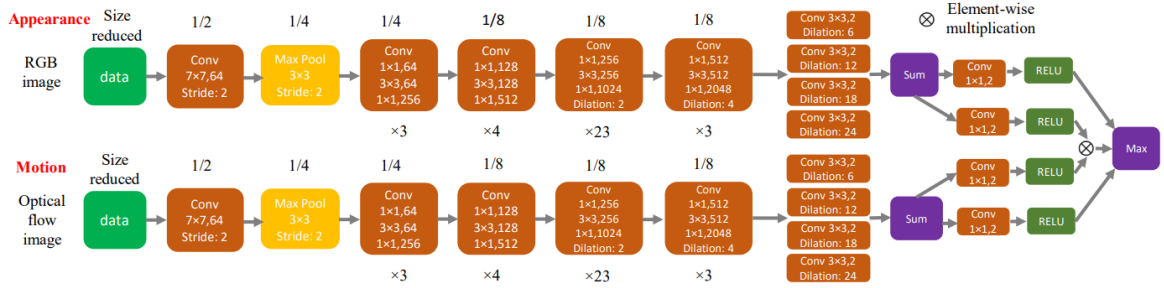


Figure 3.13 – Two-stream network which includes an appearance branch and a motion branch. Figure extracted from (S. D. Jain, Xiong, and Grauman 2017).

used to obtain object instances such as GrabCut (Rother, Kolmogorov, and Blake 2004) and its variants (Lempitsky et al. 2009; M.-M. Cheng et al. 2015; Tanai, Matsushita, and Naemura 2015; F. Yu and Koltun 2015; H. Yu et al. 2017; N. Xu, Price, Cohen, J. Yang, et al. 2017).

Several attempts have tried to combine the advantages of both object-proposals-based and object-detection-based approaches. For example, Zagoruyko et al. (2016) use a modified R-CNN model (Girshick et al. 2014) to propose instance bounding boxes as a starting point. After that, they use DeepMask object proposals instead of Selective Search to further refine the result to obtain instance level object masks. This combined system is called MultiPath Classifier. As its name implies, MultiPathNet allows information to flow along multiple paths through the net, enabling it to exploit information at multiple image scales and in surrounding image context. It supposed three modifications to Fast R-CNN: improving localization with an integral loss; providing context by using foveal regions; and finally skip connections to give multi-scale features to the network. For this reason, they improve performance over COCO dataset. Ultimately, these approaches suffer from the fact that they predict a binary mask within the bounding box proposals, making the system slower and less accurate. Similarly, Yi Li et al. (2017b) propose to combine the object detection approach of (Dai, Yi Li, et al. 2016) and the segment proposals of (Dai, He, Yi Li, et al. 2016) for fully convolutional instance segmentation (FCIS), predicting a set of position-sensitive output channels in a fully convolutional manner. These channels simultaneously address object boxes, masks, and semantic classes, making the system fast. However, this approach might experience errors and forced edges on overlapping instances.

Most recently, Mask-RCNN (He, Gkioxari, et al. 2017) extends Faster-RCNN (S. Ren et al. 2015) by adding a branch for predicting segmentation masks on each Region of Interest (RoI) in parallel with the existing branch for classification and bounding box regression. The mask branch is a small FCN applied to each RoI, predicting an object mask in a pixel-to-pixel manner. The parallel prediction makes the system simpler and more flexible.

### Video objects segmentation

Since the successful performance of convolutional neural networks in image segmentation tasks (Long, Shelhamer, and Darrell 2015; L.-C. Chen, Papandreou, Kokkinos, et al. 2016;

F. Yu and Koltun 2015), many researchers have brought them into video objects segmentation and obtained remarkable results in both semi-supervised and unsupervised tasks. These methods often process videos per-frame, treat the video object segmentation problem as a pixel classification problem. Most CNNs architecture are built upon the image semantic labelling network (Long, Shelhamer, and Darrell 2015; Ronneberger, Fischer, and Brox 2015; Bansal et al. 2017).

**Unsupervised video segmentation** In the unsupervised task, spotlight methods use different approaches to train the network such as optical flow (Tokmakov, Alahari, and Schmid 2016), the combination between optical flow and appearance model (S. D. Jain, Xiong, and Grauman 2017), or object saliency maps (W. Wang, Jianbing Shen, and Shao 2018).

More specifically, in (Tokmakov, Alahari, and Schmid 2016), moving objects' patterns are learned via an encoder-decoder style Convolutional Neural network. Their fully convolutional network is learned from synthetic video sequences with known optical flow and motion segmentation masks. At the first stage, a coarse representation of the optical flow field is learned, then in the next step, it is refined iteratively to produce more precise motion labels. Their encoder-decoder style network first learns a coarse representation of the optical flow field features and then refines it iteratively to produce motion labels at the original high-resolution. Finally, the output is further refined by a combination of conditional random field (CRF) (Krähenbühl and Koltun 2011) and an objectness to correct some errors caused by the unreliability of optical flow.

With the same purpose of segmenting generic objects in the video, S. D. Jain, Xiong, and Grauman (2017) propose a framework which composes two-stream CNNs with an appearance stream and a motion stream. While the appearance stream is used to encode general appearance from RGB image frames, the motion stream is used to capture the notable motion from its input optical flow. These two streams are fused in a unified framework. By this combination, their method can segment both static and dynamic objects.

Similarly, in (Tokmakov, Alahari, and Schmid 2017), the authors improve their previous method (Tokmakov, Alahari, and Schmid 2016) by introducing a two-stream network as in (S. D. Jain, Xiong, and Grauman 2017). Different from (S. D. Jain, Xiong, and Grauman 2017), these two streams are fused by a memory module based on a recurrent layer. More specifically, while the appearance stream and the motion stream are taken from (L.-C. Chen, Papandreou, Kokkinos, et al. 2016) and (Tokmakov, Alahari, and Schmid 2016) to describe appearance and motion features respectively, the memory module is a convolutional gated recurrent unit (GRU) that is used to encode the evolution of the object in the input video sequence.

Another unsupervised salient object detection method for videos via fully convolutional networks is proposed in (W. Wang, Jianbing Shen, and Shao 2018). In this method, they try to detect salient objects from videos by integrating static saliency and dynamic saliency networks

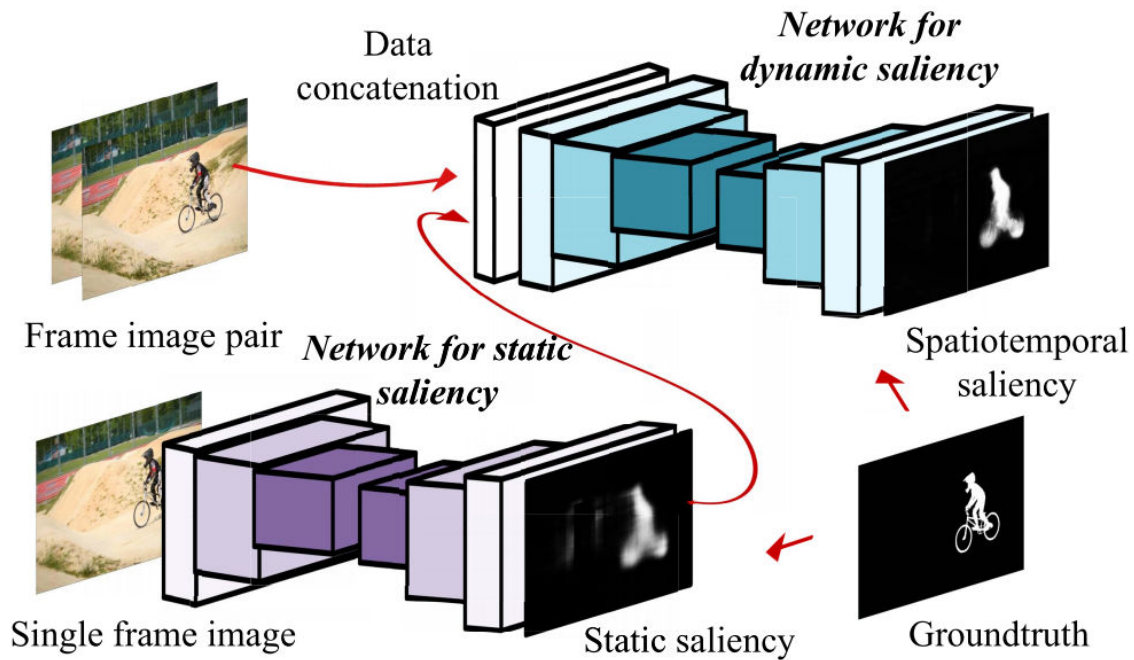


Figure 3.14 – The general pipeline of the unsupervised salient object detection in videos by (W. Wang, Jianbing Shen, and Shao 2018). Figure extracted from (W. Wang, Jianbing Shen, and Shao 2018)

simultaneously. The dynamic saliency model, explicitly incorporates saliency estimates from the static saliency model, directly produces spatio-temporal saliency inference without time-consuming optical flow computation. They also propose a novel data augmentation technique that simulates video training data from existing annotated image data sets, which enables our network to learn diverse saliency informations and prevents overfitting with the limited number of training videos.

In general, the advantage of unsupervised methods is that they do not need user intervention and any additional information rather than the video itself. However, because these methods ignore the guidance mask (such as the first frame annotation) and try to segment the most salient object, both in motion and appearance, they sometimes also identify wrong objects such as waves. Besides, they also fail to distinguish similar looking instances in videos where multiple salient objects move.

**Semi-supervised video segmentation.** Semi-supervised video segmentation refers to the task of segmenting the of objects given the annotation of several frames, typically first frame. In this task, different methods are proposed to take into account the guiding mask providing by first frame annotation. In DAVIS challenge (Federico Perazzi, Jordi Pont-Tuset, et al. 2016)- a popular video objects segmentation benchmark, two main leading approaches in the semi-supervised task is OSVOS (S. Caelles et al. 2017) and MaskTrack(F. Perazzi, A. Khoreva, et al. 2017).

*One Shot Video Object Segmentation (OSVOS).* This method was first introduced in (S.

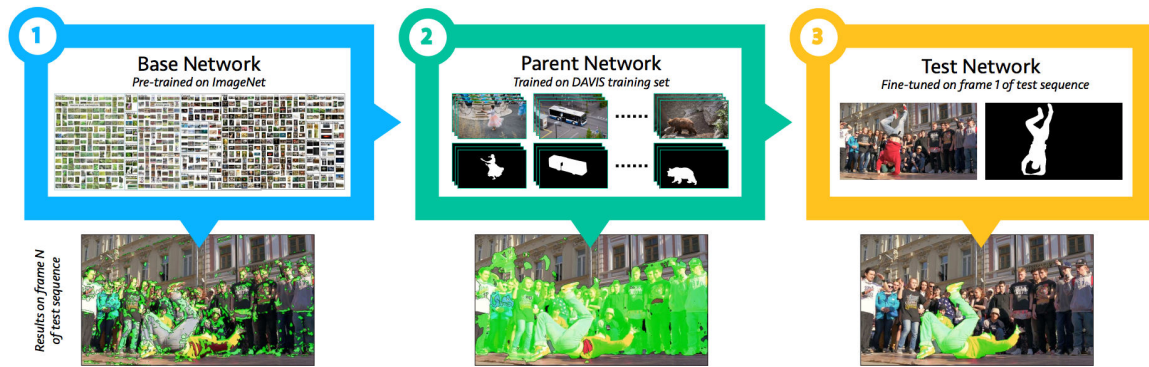


Figure 3.15 – General pipeline of OSVOS: a base network is repurposed by training on a video segmentation dataset, then finetune on first frame annotation to adapt to one particular video. Figure extracted from (S. Caelles et al. 2017)

Caelles et al. 2017). The principle is simple yet powerful: take a pre-trained FCN network, re-train it using a large video dataset and finally fine-tune it per-video using the annotation at first frame to focus on the object being segmented. More specifically, it composes four main steps:

- *Creating a base segmentation network.* In this step, a well-known network which is pre-trained on ImageNet for classification e.g. VGGNet (Simonyan and Zisserman 2014) or ResNet(He, X. Zhang, et al. 2016) is converted it to a Fully Convolutional Network by removing the Fully Connected Layer. This network is referred to as *base network*.
- *Inserting a new loss.* A new loss for image segmentation task is used, for example, pixel-wise sigmoid balanced cross-entropy as in (Xie and Zhuowen Tu 2017). Now each pixel is separately classified into foreground or background.
- *Domain transfer.* Train the new Fully Convolutional Network on the large-scale video training set such as DAVIS-2016 (Federico Perazzi, Jordi Pont-Tuset, et al. 2016). This is known as the *parent network*.
- *One-shot training.* At inference time, given a new input video for segmentation and a ground-truth annotation for the first frame, create a new model, initialized with the weights of the *parent network* and fine-tuned it using the first frame annotation.

The result of this process is a unique, one-time-use model for every new video that is overfitted for that specific video according to the first frame annotation. Because for most videos the appearance of the object and the background does not change drastically, this model produces good results. As their segmentation is not guided, and therefore it cannot distinguish multiple instances of the same object. Instead, they incorporate the notion of the object to be segmented based solely on the first frame annotation, which might result in performance decay over time,

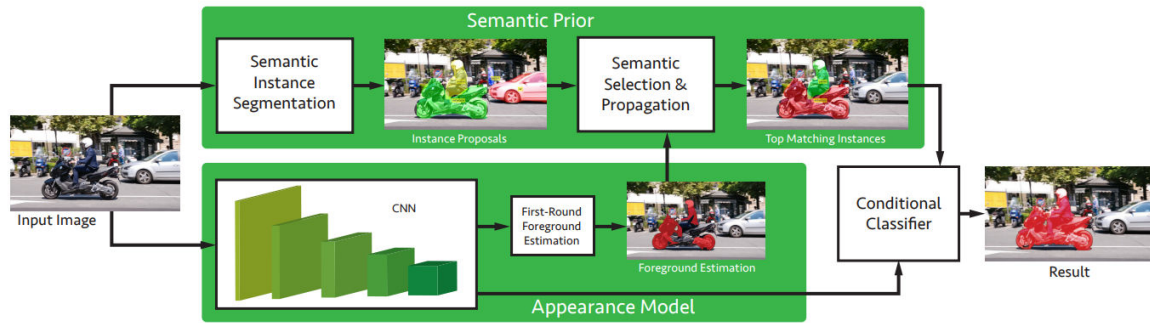


Figure 3.16 – Network architecture overview of OSVOS-S. It is composed of three major components: a base network as the feature extractor, and three classifiers built on top with shared features. Figure extracted from (Maninis et al. 2018)

as the object appearance diverges from the initial frame. This approach is later extended in various of works to improve the performance.

In (Voigtlaender and Leibe 2017), the authors propose online fine-tuning during the sequence to adapt to changes in appearance. They add an extra training step which updates the network using online training examples. These examples are selected based on the confidence of the network and the spatial configuration. More specifically, pixels belonging to the objects of interests with high confidence are selected as positive examples, while pixels which are far away from the last assumed pixel mask are selected as negative examples. These examples are mixed with the first frame to avoid drift and degraded performance. Additionally, a pre-training step based on objectness, which is learned on PASCAL (Everingham et al. 2015) is also proposed to boost the performance. As a result, they got a very high  $mIoU$  score of 86.7% over DAVIS 2016 dataset.

Recently, (Maninis et al. 2018) extend the work of (S. Caelles et al. 2017) by incorporating the semantic information of an instance segmentation method into the video object segmentation pipeline, which they call OSVOS-S. In particular, a list of object masks proposal, along with their categories is extracted in each frame using an instance-aware semantic segmentation tool such as MNC, (Dai, He, and J. Sun 2016), Mask R-CNN (He, Gkioxari, et al. 2017) or FCIS (Yi Li et al. 2017a). Given the first annotated frame, the authors perform two steps including "semantic selection" and "semantic propagation". In "semantic selection, the categories of the objects of interest are inferred by finding the best-overlapping mask. Next, in the "semantic propagation" step, extracted semantic information are used to segment the rest of the video. In particular, instances extracted from the semantic instance segmentation algorithm that best match the model of the object are used to combine with the appearance model of the object, using a conditional classifier. This helps the algorithm can align well with the same categories selected in the first frame. For example, if in the first frame, a person is selected, then throughout the rest of the video, this category should remain unchanged.

In (Ci, C. Wang, and Y. Wang 2018), the problem of segmenting the target object throughout a video given only a bounding box in the first frame is solved by leveraging



the location-sensitive embeddings which are capable of distinguishing the pixels of similar objects. The pipeline of the method is as follow: (1) based on the segmentation results of the previous frame; the method first zooms in to the probable foreground region by cropping and resizing in the current image. (2) The cropped image is fed to a network to predict the location-sensitive embeddings and initial foreground predictions. (3) The embeddings are fused with the foreground predictions to obtain final segmentation.

All of these approaches perform segmentation segments the frames independently without considering temporal consistency. By using per image segmentation, these approaches can deal with large displacement and occlusion. However, since their backbone is a network used for semantic segmentation, they can not distinguish between instances in the same class or between objects that resemble each other. Moreover, these methods only learn the target appearance in the initial frame via extensive training of deep neural networks with stochastic gradient descent, and therefore, their performance will decay over time.

Several approaches try to capture temporal dependency by incorporating motion information via optical flow (Bao, B. Wu, and W. Liu no date; P. Hu et al. 2018; J. Cheng, Tsai, Shengjin Wang, et al. 2017). In particular, (Bao, B. Wu, and W. Liu no date) performing temporal propagation via a spatio-temporal MRF in which temporal dependencies are modeled by optical flow, while spatial dependencies are expressed by a CNN. In (P. Hu et al. 2018), an optical flow based active contour model was applied to gain guidance for the cascaded segmentation refinement network, which significantly reduced the number of fine-tuning iterations. In (J. Cheng, Tsai, Shengjin Wang, et al. 2017), the authors propose SegFlow in which features extracted from object segmentation and optical flow estimation are concatenated at different scales for mutual boosting. Recently, (H. Xiao et al. 2018) intensely exploited optical flow to provide enhanced feature representation and motion prior.

*CNNs-based mask tracking:* In OSVOS-liked method, each frame is processed independently, which limits the ability to capture temporal dependency. In order to exploit consistency between video frames, several classical methods propagate the supervisory segmentation mask of the first frame to the temporally adjacent ones (Tsai, M.-H. Yang, and Black 2016; Wen et al. 2015; Märki et al. 2016). Following this trend, recently CNN-based mask tracking methods have been proposed.

For example, (Jampani, Gadde, and Gehler 2017) mix CNNs with ideas of bilateral filtering. They introduce a Video Propagation Network (VPN) that propagates information forward through video data. The VPN architecture is composed of two components. The first one is a bilateral temporal network that performs image adaptive spatio-temporal dense filtering. The bilateral network allows to connect all pixels densely from current and previous frames and to propagate associated pixel information to the current frame. This is then followed by a standard spatial CNN on the bilateral network output to refine and predict the mask for the present video frame.

(F. Perazzi, A. Khoreva, et al. 2017) use the previous mask layer as the signal to guide the

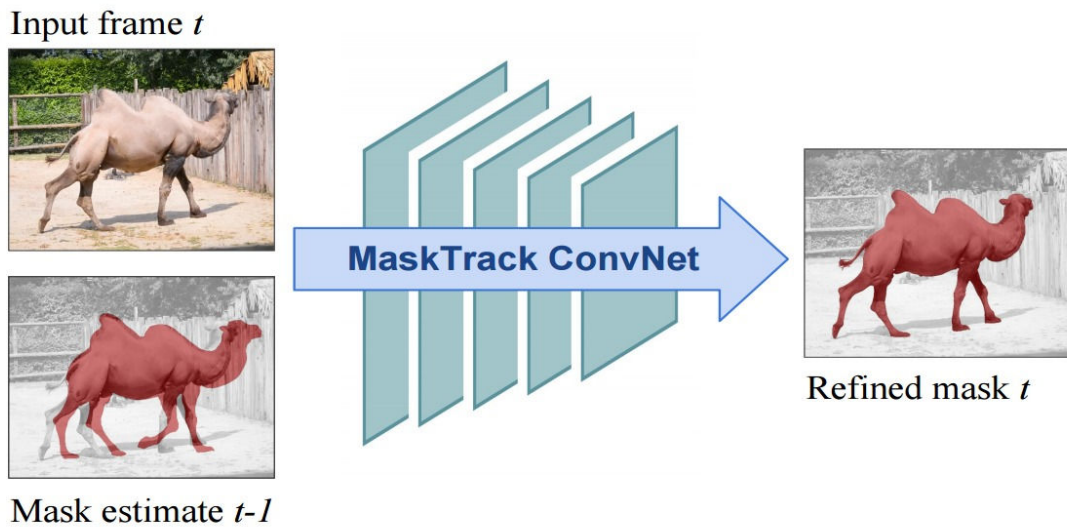


Figure 3.17 – Given a rough mask estimate from the previous frame, the MaskTrack method train a CNN to provide a refined mask output for the current frame. Figure extracted from (F. Perazzi, A. Khoreva, et al. 2017)

network. For each frame, the predicted mask of the previous frame is added as an additional input to the network: the input is now four channels (RGB+previous mask). This input for the previous mask channel is artificially synthesized by small transformations of the ground truth annotation of each still image. They also add an identical second stream network, based on optical flow input. The model weights are the same as in the RGB stream. The output of two streams is fused by averaging the results.

The work of (F. Perazzi, A. Khoreva, et al. 2017) is later extended in (Anna Khoreva et al. 2017) by complex data augmentation on the first frame. This strategy handle video challenges by estimating potential variation on the target object. Instead of using large training sets hoping to generalize across domains, they generate in-domain training data using the provided annotation on the first frame of each video to synthesize plausible future video frames. This approach allows reaching competitive results while using  $30 \times 100\times$  less annotated data than competing methods. Their results indicate that using a larger training set is not automatically better and that for the tracking task a smaller training set that is closer to the target domain is more effective.

Most recently, (X. Li et al. 2017) have proposed to employ an adaptive object re-identification module along with the mask propagation method of (F. Perazzi, A. Khoreva, et al. 2017) to retrieve missing instances. Specifically, when missing instances are re-identified with high confidence, they are assigned with a higher priority to be recovered during the mask propagation process. For each retrieved instance, its frame is taken as the starting point, and the mask propagation is applied bi-directionally. Both mask propagation and re-identification modules are iteratively applied to the whole video sequence until no more high confidence instances can be found. Besides, they adopt the much deeper ResNet network with atrous spatial pyramid pooling and multi-scale testing (L.-C. Chen, Papandreou, Schroff, et al. 2017)



to increase the model capacity and the resolution of prediction.

*Other recent methods.* With the rapid development of deep learning, the video objects segmentation domain is growing very fast. In 2018, a large number of works on video objects segmentation had been proposed since the release of YouTube-VOS - a large scale video segmentation dataset.

For example, to take advantage of long-term temporal structures of the video, (Y.-T. Hu, J.-B. Huang, and A. Schwing 2017) propose MaskRNN- a recurrent neural net approach which fuses in each frame the output of two deep nets for each object instance and a binary segmentation net providing a mask and a localization net providing a bounding box.

Specifically, the most notable work in 2018 is PreMVOS (Luiten, Voigtlaender, and Leibe 2018) which won the 2018 DAVIS Challenge on video object segmentation and Youtube-VOS challenge. In PreMVOS, the algorithm first generates a set of accurate object segmentation mask proposals for all of the objects in each frame of a video using a Mask R-CNN (He, Gkioxari, et al. 2017) like object detector to generate coarse object proposals, then a fully convolutional refinement network inspired by (N. Xu, Price, Cohen, J. Yang, et al. 2017) and based on the DeepLabv3+ (L.-C. Chen, Y. Zhu, et al. 2018) architecture to produce accurate pixel masks for each proposal. Secondly, these proposals are selected and merged into accurate and temporally consistent pixel-wise object tracks over the video sequence. They use a merging algorithm that takes into account an objectness score, the optical flow warping, a Re-ID feature embedding vector, and spatial constraints for each object proposal.

In contrast with PreMVOS which focus on the accuracy, many different method focus exchange accuracy for speed. Instead of heavily fine-tuning the segmentation network for multiple rounds, these methods attempt to solve the VOS task within a single forward pass, such that the runtime can be significantly reduced. Those methods take the first frame with its mask annotation either as guidance to slightly adjust parameters of the segmentation model (L. Yang et al. 2018) or as reference for segmenting the following frames without tuning the segmentation model (Y. Chen et al. 2018; J. Cheng, Tsai, Hung, et al. 2018; Oh et al. 2018). (J. Cheng, Tsai, Hung, et al. 2018) converted VOS into multiple parts tracking problems by tracking and segmenting the similar parts of the target object. (Oh et al. 2018) proposed a deep Siamese encoder-decoder to propagate the mask of the previous frame, avoiding the process of online fine-tuning. (Y. Chen et al. 2018) regarded VOS as a problem of metric learning and well tackled different kinds of annotation. In (L. Yang et al. 2018), the parameters of the segmentation network are affected by two kinds of modulators, which are similar to the meta-learner and induced from the appearance of the target object and the spatial prior of previous results. The feed-forward based approaches can achieve favorable runtime but are unable to fast and continuously adapt the segmentation network, which might be fragile to drastic changes in the video.

## 3.3 Method

### 3.3.1 Choosing the best method for video objects segmentation

Having now done an extensive review of both the classical and recent video segmentation approaches, we would like to recap the approaches to video segmentation which seem most promising and viable. We recall that our goal is to create a video segmentation which has these main objectives which are useful for video inpainting. (1) The resulting masks must cover all details of the objects, to make sure that there is no part of the objects left and create artifacts in the object removal algorithm. (2) The algorithm must have the capabilities of distinguishing between different instances of similar objects so that users can decide which instance they want to remove. (3) It must allow user interaction for correcting errors, but the amount of user intervention to be done must be little.

Let us first note that we set aside unsupervised video segmentation approach because it does not allow users to select the objects at the first frame and moreover, the results produced by this approach is far from our desire. The interactive video segmentation methods such as Rotobrush ([Xue Bai et al. 2009](#)) or Instance Cut ([Levinkov et al. 2016](#)) require too much user annotation, therefore, not applicable for our application. The best fit to our purpose is the semi-supervised video segmentation approach which allows users to select the objects at the first frame. In term of semi-supervised approaches, then, there are basically two option available:

- Frame-by-frame CNN-based segmentation. ([S. Caelles et al. 2017](#))
- Mask tracking ([Ramakanth and Babu 2014](#); [Tsai, M.-H. Yang, and Black 2016](#); [F. Perazzi, A. Khoreva, et al. 2017](#))

The first option has taken the advantage of CNNs in semantic image segmentation and extend it to video segmentation. A typical method under this approach is the OSVOS network of ([S. Caelles et al. 2017](#)). Specifically, this method is frame-by-frame segmentation in which object appearances are learned and the video segmentation problem is treated as a sequential image segmentation problem. The strength of CNN networks in feature representation is much better than other methods, which give them the capability of distinguish between background and foreground. This approach produces very accurate segmentation within a reasonable time. Furthermore, by treating each image independently, it does not need to process frames sequentially and it is able to overcome several challenges such as occlusions, complex motion, and errors propagation. In addition, the implementation of this method is straight forward and the pre-trained models are also available, which is clearly an advantage.

However, current CNN-based semantic segmentation algorithms are still essentially image-based, and do not take global motion information sufficiently into account. As a consequence, semantic segmentation algorithms cannot deal with sequences where: (a) several instances of



Figure 3.18 – OSVOS method can distinguish well between background and foreground, but it has difficulties when separating similar instance of the same semantic class. Top: Results when we want to segment both persons. Bottom: results when we only want to segment the person on the right.

similar objects need to be distinguished; and (b) these objects may eventually cross each other. Examples of such sequences which are extremely challenging for video inpainting are *Les Loulous*<sup>1</sup> introduced in (Newson, Almansa, Fradet, et al. 2014) and the *Museum*<sup>2</sup> introduced in (Granados, Tompkin, et al. 2012). Figure 3.18 demonstrates a typical example with LES LOULOUS sequence. On top of the figure, we show the segmentation result of OSVOS when trying to separate between background and foreground. In this case, we want to segment all the persons. We therefore select both of them at the first frame and annotate them by a single label. As we can see, OSVOS can extract the appearance features of these two persons, and since these features are different from the background, OSVOS can distinguish very well between background and foreground. However, if we only want to extract one particular object, e.g., the man on the right (as shown in the bottom of Figure 3.18), this method fails because the background, in this case, contains the remaining person, which is very similar with the object we want to segment. These similar appearances confuse the network because it does not take into account any knowledge about the location and the motion of objects.

Moreover, another disadvantage of this type of approach is that it does not allow users to interfere during the segmentation process. If users do not satisfy with the result, and they want to produce better results by correct the error, they must retrain the network again, which is a very time-consuming process.

On the other hand, in the second option, more classical mask tracking techniques like optical-flow based propagation or global graph-based optimization do take global motion information into account (H. Yang et al. 2011). Therefore, it can track the mask selected by users in the first frame and can guarantee temporal coherence. Moreover, these mask tracking methods allow users to revise the mask quality at some particular frames, the mask

<sup>1</sup>[https://perso.telecom-paristech.fr/gousseau/video\\_inpainting/](https://perso.telecom-paristech.fr/gousseau/video_inpainting/)

<sup>2</sup><http://gvv.mpi-inf.mpg.de/projects/vidinp/>



Figure 3.19 – Segmentation results of different mask track-based methods. Top: Seam-seg(Ramakanth and Babu 2014), bottom: ObjectFlow(Tsai, M.-H. Yang, and Black 2016).

propagation process can continue from that frame without retraining the network as in OSVOS. Nevertheless, these methods are most often based on bounding boxes or rough descriptors and do not provide a precise delineation of objects' contours. Besides that, they suffer from the error propagation problem so that they are susceptible to occlusion and objects interaction. Two recent attempts to adapt video-tracking concepts to provide a precise multi-object segmentation (Ramakanth and Babu 2014; Tsai, M.-H. Yang, and Black 2016) fail completely when objects cross each other like in the MUSEUM or LES LOULOUS sequences. This is illustrated in Figure 3.19. When another object occludes the object we want to segment, the mask propagation procedure fails, and the correct mask is never discovered again.

Therefore, we approach the video object segmentation problem by introducing a novel hybrid technique which combines the benefits of classical video tracking with those of CNN-based semantic segmentation. While the CNN-based segmentation helps us in separating the objects from the background, classical mask tracking combine with a CNN *mask refining* network enable us capturing the temporal dependency across frames.

### 3.3.2 Overview

The main idea underpinning our method is to extract object candidates for each frame and to link them together across frames to form target identities. For each new frame we wish to label pixels as objects of interest/background, for this we build upon the architecture of the existing pixel labeling CNNs and train it to generate per-frame semantic segments. The challenge is then: how to separate different instance from the same object? We solve this by using two complementary strategies. First, we use the mask tracking scheme coping with a *mask refining* network to track the object across frames and generate object mask proposals. Second, we choose the best mask by graph-based data association algorithm. The structure of our hybrid technique is shown in Figure 3.20. CNN-based modules are depicted in green and red, and their inner structure is described in Section 3.3.3. Modules that are inspired by



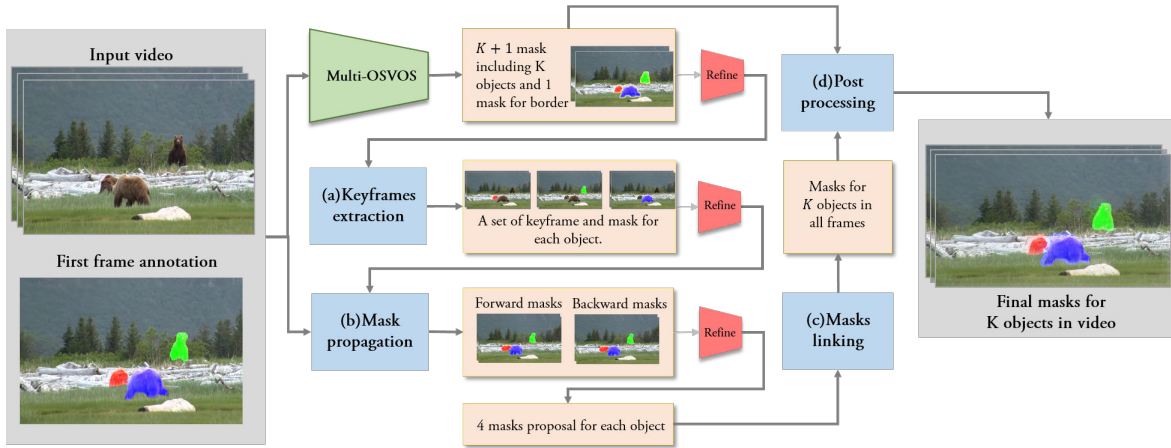


Figure 3.20 – General pipeline of our object segmentation method. Given the input and the annotation in the first frame, our algorithm alternates two CNN-based semantic segmentation steps (*multi-OSVOS* network in green and Refining network in red) with 4 video-tracking steps (depicted as blue blocks): (a) keyframe selection, (b) mask propagation, (c) mask linking and (d) post processing.

video-tracking concepts are depicted in blue and are detailed in Section 3.3.4. In the next section, we first introduce the CNN-based segmentation networks which are adopted in our scheme, and we then explain how to use them to achieve accurate masks for multiple objects segmentation task.

### 3.3.3 Segmentation networks

Our system uses two different segmentation networks: a *multi-OSVOS* network and a *mask refining* network. Both operate on a frame by frame basis. Assume we have  $K$  objects of interest (including the background), the *mask refining* network provides  $K$  segmentation masks, whereas the *multi-OSVOS* network provides  $K + 1$  segmentation masks. The additional mask corresponds to a *smart dilation* boundary layer common to all objects (which we explain later). The main difference between both networks is that the *multi-OSVOS* network yields the first prediction, whereas the *mask refining* network takes mask predictions as an additional input and improves those predictions based on image content.

Training these networks is a challenging task, because the only labeled example we can rely on (for supervised training) is the first annotated frame and the corresponding  $K$  masks. The next paragraphs focus on our network architectures and on semi-supervised training techniques that we use to circumvent the training difficulty.

#### Multi-OSVOS network

One Shot Video Object Segmentation (OSVOS) was first introduced in (S. Caelles et al. 2017) and become a breakthrough in video segmentation which achieved the best performance in DAVIS challenge 2016 (Federico Perazzi, Jordi Pont-Tuset, et al. 2016). The training technique

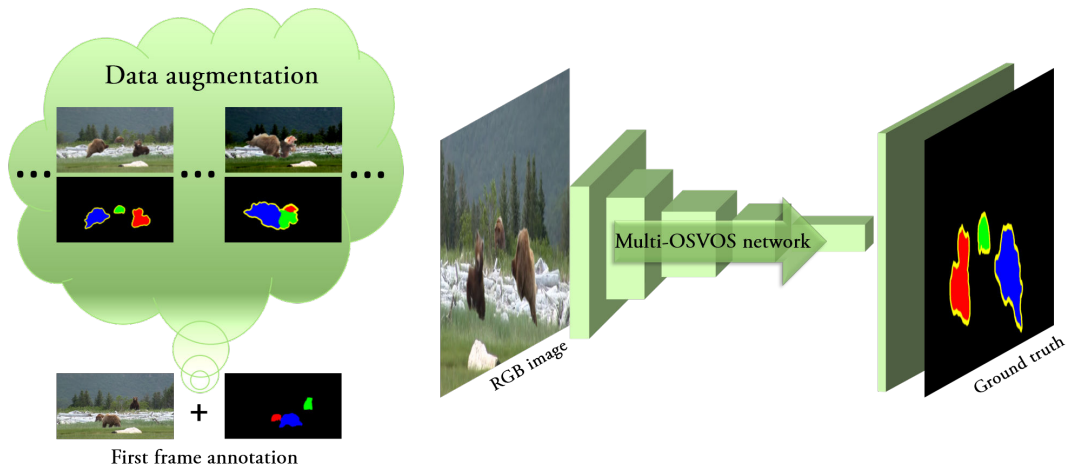


Figure 3.21 – The *multi-OSVOS* network helps us separating background and objects.

of this network composes two stages of training: offline training and online finetuning. In the offline training stage, the network is trained on a large variety of objects, to construct a model that is able to discriminate the general notion of a foreground. Then at the fine-tuning step, the network focus on the particular instance that we aim to segment. Our training scheme is mainly inspired by this procedure, which we will present in more details.

**Offline training** At this stage, the objective is creating a network that can perform object segmentation in a general way. Although inspired by OSVOS, the architecture of our network is slightly different. While OSVOS uses Fully Convolutional Network of (Long, Shelhamer, and Darrell 2015), we use Deeplab model as the backbone. The architecture of Deeplab is presented in Section 2.1.2 of Chapter 2. To recap, FCN works by replacing the fully connected layers of a pre-trained model (e.g. VGG (Simonyan and Zisserman 2014)) by convolution layers with a kernel size of  $1 \times 1$ , followed by unpooling layers to recover the spatial resolution of the feature maps. As a consequence, output maps can achieve the same resolution as the input image of the models. In order to reduce the noise in output maps, FCN introduces skip connections between pooling layers and unpooling layers. Deeplab, on the other hand, uses atrous convolution (L.-C. Chen, Papandreou, Kokkinos, et al. 2016), or dilated convolution (F. Yu and Koltun 2015) to reduce the pooling factor of the pre-trained network.

A major difference between FCN and Deeplab is that in FCN, the networks enlarge the receptive field by inserting a lot of down-sample layers which reduce the size of features and may lose the details. This problem is solved in Deeplab model by the introduction of the dilated convolution. By inserting zeros between filter elements, dilated convolution can expand the network's receptive field size and let the network cover more relevant information. Thus, it outperforms FCN in some common dataset such as PASCAL VOC 2012 (Everingham et al. 2015). For a more details about the Deeplab network, we refer the reader to Section 2.1.2 of Chapter 2. We use this as the backbone of our *base network*. We note that the *base network* is already pre-trained on ImageNet (Russakovsky et al. 2015).

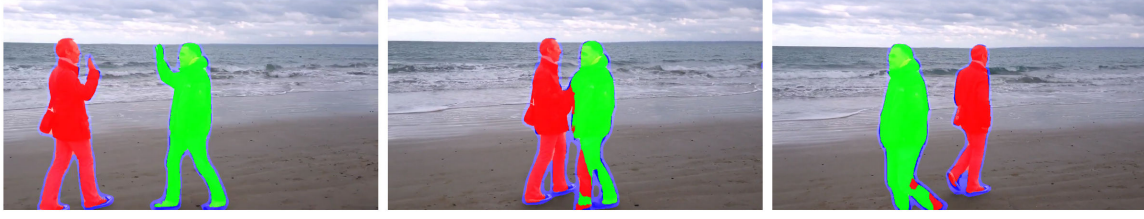


Figure 3.22 – The output of the *multi-OSVOS* network. The red and green color corresponding to different objects, the blue region is the learned dilated mask.

For the offline training, our *multi-OSVOS* network uses a transfer learning technique for image segmentation: The *base network* is retrained using a large database of labeled videos. After training, this so-called *parent network* can roughly separate all foreground objects from the background.

While the original OSVOS only separates background and foreground, to make it work with multiple objects segmentation, in the *multi-OSVOS*, we make some additional changes. More specifically, at the end of the network, instead of OSVOS' two-layer output (for background and foreground pixel probabilities), we use  $K + 1$  layers, including  $K$  objects of interest, and one additional layer for the border outside all objects. This border layer is specially designed for our video object removal algorithm since we need the mask to cover the whole object. We call this *Smart dilation*. The output of this CNN is a probability map with  $K + 1$  channels, and each channel represents whether a pixel belongs to the object of interest or not. This network is trained to minimize the standard cross entropy loss, which takes into account the fact that different classes occur with different frequencies in a dataset. Using the notation as in (Xie and Zhuowen Tu 2017), we denote the training dataset by  $S = \{(X_n, Y_n), n = 1, \dots, N\}$ , with  $X_n$  being the input image and  $Y_n = \{y_j^{(n)}, j = 1, \dots, |X_n|\}$ ,  $y_j^{(n)} = (y_j^{(n, L_1)}, \dots, y_j^{(n, L_{K+1})})$ ,  $y_j^{(n, L_i)} \in 0, 1$  the predicted pixel-wise labels for  $K + 1$  classes. For simplicity, we drop the subscript  $n$ . The training loss is:

$$L = \sum_{j \in Y} \omega_{y_j} C(y_j, \hat{y}_j), \quad j \in 1, \dots, |Y|$$

where  $\omega_{y_j}$  depends on the label  $y_j$  of pixel  $j$  as the inverse normalized frequency of labels inside the mini batch.  $C(\cdot)$  indicates the standard cross-entropy loss between the label and the prediction  $\hat{y}_j$ . The balanced loss has proven to perform very well in boundary detection (Xie and Zhuowen Tu 2017), where the majority of the samples belong to the background class.

We note that we do not use contour snapping branch as in OSVOS because our purpose is that the mask should cover all details of the object rather than having exact object contours.

For the training details, we train the network on the binary masks of the training set of DAVIS 2016 to learn a generic notion of how to segment objects from their background. We then use this network weights as initialization and retrain on DAVIS 2017 *train-val* set (exclude the validation sequences of DAVIS 2016) to segment multiple objects. We use the technique described in Section 2.1.1 of Chapter 2 for faster convergence. More specifically, for the





Figure 3.23 – Some example training images generated by our data augmentation framework.

parameter updating scheme, we use Stochastic Gradient Descent (SGD) with momentum 0.9 for 100 epochs. We augment the data by mirroring and zooming in. The base learning rate is found via the learning rate search scheme and is changed follow learning rate cyclically which is described in Section 2.1.1 of Chapter 2. We also add a weight decay of 0.0025. The learning rate and the weight decay is set to its base value at the end of the network and they are decreased by 10 times if we go to the preceding block of the network.

**Online fine-tuning.** After the *parent network* is trained using large labeled video dataset, at test time, we fine-tune this network using the first frame annotation (annotation mask and image) in order to improve the segmentation of a particular object of interest and further boosting the video segmentation quality. This idea of online training is originally proposed for object tracking techniques (Danelljan et al. 2015; Nam, Baek, and Han 2016). The idea is to use, at test time, the segment annotation of the first video frame as additional training data. Using augmented versions of this single frame annotation, we proceed to fine-tune the model to become more specialized for the specific object instance at hand.

Since we only have one pair of data for training (that is why it is called one-shot training), to enrich the data for training, we adopt a data augmentation technique in the spirit of Lucid Tracker (Anna Khoreva et al. 2017). This technique generates in-domain training data using the provided annotation on the first frame of each video to synthesize plausible future video frames. It indicates that, for the tracking task, a training set that is closer to the target domain is more effective than using a large training with general knowledge. To ensure our training data is in-domain, we synthesize samples from the provided annotated frame (first frame) in each target video. The outcome of this process is a large set of frame pairs. To capture a large range of possible future frames, such as illumination change, deformed or translated objects, objects occluded by another, etc. , a heavy augmentation and task-specific strategy is needed. We achieve this by cutting-out the foreground object, in-painting the background, perturbing both foreground and background, and finally recomposing the scene. More specifically, we do the following augmentation step:

- *Separating the background/foreground:* We extract all objects from the first frame and reconstruct the background using an image inpainting algorithm (Newson, Almansa, Gousseau, et al. 2017).

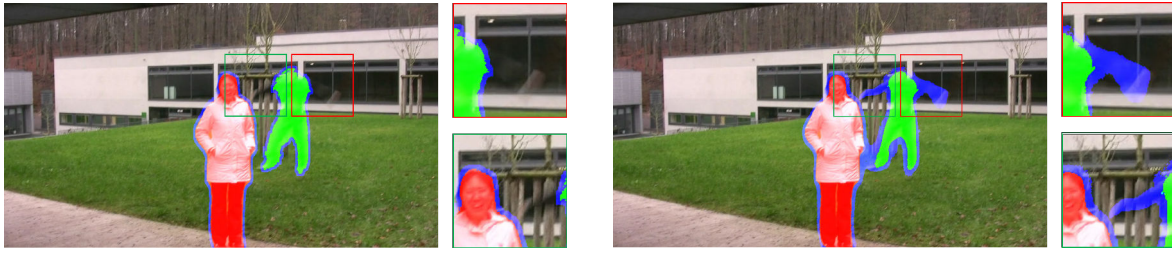


Figure 3.24 – Advantages of using the smart dilation mask, *i.e.* a smart border layer in the output map of our *multi-OSVOS* network. (a) The border is obtained by simply dilating the output map of the network: some parts of the objects are not covered. (b) The border layer is learned by the network: the transition region is covered.

- *Processing the background:* We globally modify the image by randomly altering saturation  $S$  and value  $V$  (from HSV color space) to synthesize illumination changes. To simulate camera view changes, we additionally transform the background using affine deformations by applying random translations, rotations, and scalings.
- *Processing the foreground:* Object motion: we simulate motion and shape deformations by applying global translation as well as affine and non-rigid deformations to the foreground object. The object is placed at any location within the image with a uniform distribution. We apply random rotation  $\pm 30^\circ$ , scaling  $\pm 15\%$  and thin-plate splines deformations.
- *Merging the background/foreground:* finally perturbed objects and perturbed background are composed by Poisson blending (Pérez, Gangnet, and Blake 2003a).

This is a sensible way of generating large amounts of labeled training data with an appearance similar to what the network might observe in the following frames.

We generate 100 training samples from this single annotation and proceed to fine-tune the model previously trained offline. With online fine-tuning, the network weights partially capture the appearance of the specific object being tracked. The model aims to strike a balance between general instance segmentation (so as to generalize with respect to object changes), and specific instance segmentation (so as to leverage the common appearance across video frames). The network is trained with the same learning parameters as for offline training, except the learning rate since we are only fine-tuning, the learning rate is 100 times smaller than the one we used for offline training.

**Smart dilation** In video objects removal application, one crucial property of the mask is that it does not miss any part of the object; otherwise, a very unpleasant artifact appears because the information of the remaining part will be propagated into the occlusion and damage the reconstruction result. A very basic way to handle this would be to dilate the segmentation mask by a fixed sized structural element. However, this is not practical because the output segmentation mask is not homogeneous around the contour of the object. That means for

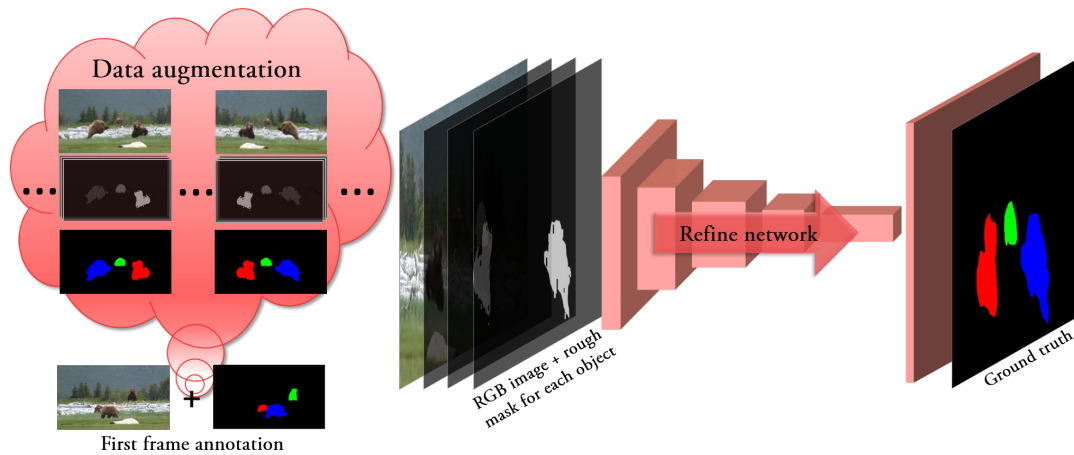


Figure 3.25 – The *mask refining* network is used to fine-tune a rough input mask to output an accurate mask of the object.

some regions, a dilation with a small size of the structural element is enough to cover the details, but other regions require a larger size, typically in the presence of motion blur.

To tackle this problem, we introduce *smart dilation* which aims at learning the dilation. The dilated mask is learned as an additional input layer to the network. This technique is similar to the work of N. Xu, Price, Cohen, and T. Huang (no date) where the authors predict alpha matting of an image. However, in contrast to (N. Xu, Price, Cohen, and T. Huang no date) where the trimap and the alpha matting mask are available for training, in our method, ground truth for this layer is not available. Therefore, we create the ground truth by simply dilating the original ground truth mask using a  $7 \times 7$  structuring element. It can be seen as a boundary region between the background and the foreground. If there are several pixels which are confused by the network, such as the motion blur region, this boundary region will capture them. This way, the network has a more flexible way to cover the details of the objects. The smart dilation mask is particularly essential in the presence of motion blur, as explained in the next paragraph.

A typical example can be seen in figure 3.24 where some parts of the hands and legs of the man cannot be captured by simply dilating the output mask. The explanation for this case is that motion blur leads to partially transparent zones which are not recognized by the network as part of the man’s body. With the smart dilation mask, the missing parts are properly captured, no pixel is left over.

### Mask refining network

The *multi-OSVOS* network can separate objects and background precisely, but it relies exclusively on how they appear in the annotated frame without consideration of their position, shape or motion cues across frames. Therefore, when objects have similar appearances, *multi-OSVOS* fails to separate between individual object instances. In order to take such cues into account, we propagate and compare the prediction of *multi-OSVOS* across frames using

video tracking techniques (Section 3.3.4) and then we double-check and improve the result after each tracking step using the *mask refining* network described below.

To take the position and shape information into account, we introduce a mask refining network - a modification of *multi-OSVOS* by expanding the network input from RGB to RGB+rough masks channels. This network takes the estimated version of the object masks- which we call rough masks and output the accurate version of these masks. Each estimated mask for each object is provided in a separate channel, expanding the network to accept  $3 + K$  input channels (RGB + K object masks).

These extra mask channels are meant to provide an estimation of the visible area of the objects in the current frame, their approximate location, and shape. This information can be seen as the guidance signal in order to guide the pixel labeling network to output the accurate masks of the objects of interest. Therefore, the goal of the *mask refining* network, as its name implies, is to refine a prediction of the object masks and output an accurate segmentation mask for each object. There are two key observations that make this approach practical. First, very rough input masks are enough for our trained network to provide sensible output segments. The primary role of the input mask is to point the CNN towards the correct object instance. This ideal of guided instance segmentation is similar to networks like DeepMask (Pinheiro, Collobert, and Dollár 2015), SharpMask (Pinheiro, T.-Y. Lin, et al. 2016), and Hypercolumns (Hariharan, Arbeláez, Girshick, et al. 2014), but instead of taking a bounding box as guidance, we can use an arbitrary input mask. The architecture of the network is similar to the one used in the *multi-OSVOS* network. More specifically, the Deeplab architecture is reused with the initialization weights from a pre-trained model on ImageNet. Zeros initialization is used for the additional mask channels in the input.

The training step of this network is similar to our *multi-OSVOS* network, but because of the additional rough masks, it has a few differences.

**Offline training.** Offline training is performed in exactly the same way as for *multi-OSVOS* network, except that the training set has to be augmented with inaccurate input mask predictions. These should not be exactly the same as the output masks, otherwise, the network would learn to perform a trivial operation ignoring the RGB information. Such inaccurate input mask predictions are created by applying relevant random degradations to ground truth masks, e.g., small translations, affine and thin-plate spline deformations, followed by a coarsening step (morphological contour smoothing and dilation) to remove details of the object contour; finally, some random tiny square blocks are added as noise. These steps have their own purposes. The affine transformations and non-rigid deformations aim at modeling the expected motion of an object between two frames. The coarsening permits us to generate training samples that resemble the test time data, simulating the blobby shape of the output mask given from the previous frame by the CNN. The square noise simulates the errors caused by our mask propagation step. These three ingredients make the estimation more robust to noisy



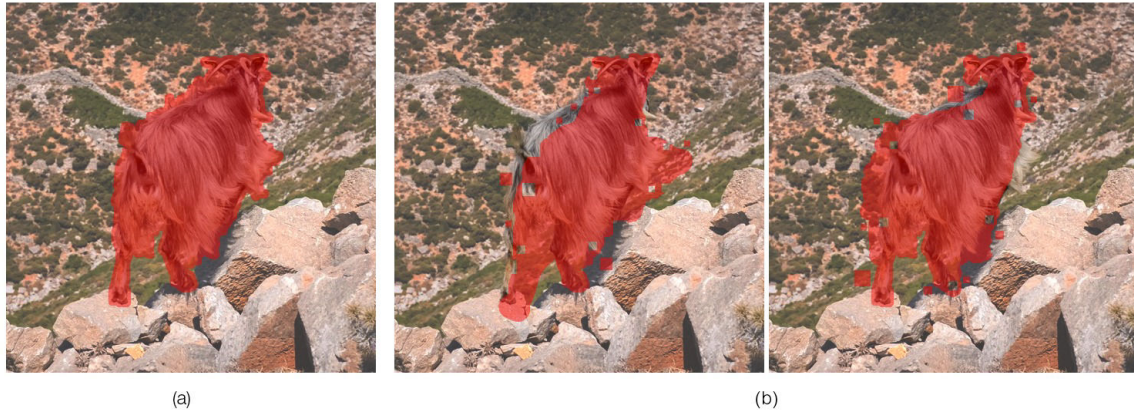


Figure 3.26 – The *mask refining network* is used to fine-tune a rough input mask. (a) Annotated mask. (b) Example training masks.

segmentation estimates while helping to avoid accumulation of errors from the preceding frames.

The ground truth output masks in the training dataset are also dilated by a structuring element of size  $7 \times 7$  pixels in order to have a safety margin which ensures that the mask does not miss any part of the object, which is similar to *smart dilation*.

**Online training** For further boosting the video segmentation quality, online training as in *multi-OSVOS* network is also added. With online fine-tuning, the network weights partially capture the appearance of the specific object being tracked. The model aims to strike a balance between general instance segmentation (so as to generalize to the object changes), and specific instance segmentation (so as to leverage the common appearance across video frames).

### 3.3.4 Multiple object masks tracking

As a complement to CNN-based semantic segmentation, we use more classical video tracking techniques in order to take global motion and position information into account. The main ingredient of our object tracking subsystem is a motion-based *mask propagation* technique described below (see block (b) in Figure 3.20). This module alone provides results similar to other object tracking methods like ObjectFlow (Tsai, M.-H. Yang, and Black 2016). In particular, it can distinguish between different instances of similar objects, based on motion and position. However, it loses track of the objects when they cross each other, even if they are not similar. In order to prevent this from happening we complement the mask propagation module with different kinds of checks:

**multi-OSVOS refinement:** The *mask refining network* (Section 3.3.3) is applied to the output of *multi-OSVOS* network.

**Keyframes extraction:** Mask propagation is effective only when starting from reliable masks, for example when it does not cross another object. Frames where this is detected to be

true, are labeled as *keyframes*, and mask propagation is performed only between pairs of successive keyframes.

**Mask propagation:** the *mask refining* network (Section 3.3.3) is applied to the output of each mask propagation step in order to avoid errors to accumulate from one frame to the next.

**Mask linking:** When the mask propagation step is not sure about which decision to make, it will provide not one, but several mask candidates for each object. A graph-based technique allows linking together all these mask candidates. This way the decision on which mask candidate is the best for a given object on a given frame is taken based on global motion and appearance information.

**Post-processing:** After mask linking a series of post-processing steps is performed that uses the original *multi-OSVOS* result to expand labeling to unlabelled regions.

**Interactive correction:** In some situations where errors appear, the user can manually correct them at one frame and this correction is propagated to the remaining frames by the propagation module.

The following paragraphs describe in detail the internal workings of the five main modules of the multiple object tracking subsystem: (a) multi-OSVOS refinement, (b) Keyframe extraction, (c) Mask propagation, (d) Mask linking and (e) Post-processing.

**multi-OSVOS refinement.** To refine the results of the *multi-OSVOS* network, we feed the output of this network to the *mask refining* network to further boost the performance. This online refinement technique is inspired by (Voigtlaender and Leibe 2017) where the output of the network is used as a guidance signal to improve accuracy. We note that our *mask refining* network is used multiple times as depicted in Figure 3.20.

**Key frames extraction** A frame  $t$  is a keyframe for an object  $i \in \{1, \dots, K\}$  if the mask of this particular object is known or can be computed with high accuracy. Masks from these frames are used to propagate to other frames. Keyframes are used to mitigate the error propagation problem. Due to large displacement, motion blur, appearance changes or occlusion, and because we used the previous mask to predict the current mask, if an error occurs, it will be propagated across time. Keyframes mitigates this problem because they provide high accurate mask and can act as the checkpoints to reset the error. The first frame of the video is a keyframe because the object masks in this frame are provided by the user.

However, using only the first frame to compute masks for other frames is not enough since the object appearances change across time. Therefore, we need to find more keyframes in the video. That means instead of propagating only from the first frame, we propagate from keyframes.

Keyframes are also important in our video objects removal application because they allow the user to correct errors during mask propagation. If the user is not satisfied with the results in the current frame, he can manually correct it. This way, the corrected frame is treated as a keyframe and the mask at this frame is used to propagated to the next frame. This enables as simple and easy way to correct the segmentation mask.

However, with only the first frame annotation available, finding keyframes is not an easy task. In the literature, there are several works which follow a similar approach. For example, in (X. Li et al. 2017), to extract keyframes, the authors rely on an objects detection algorithm at bounding box level, namely Faster-RCNN (S. Ren et al. 2015) to detect object position in the future frame. After that, a semantic segmentation network is applied to obtain pixel-level accuracy inside the bounding box. However, this technique is only applicable if the objects we want to segment belong to the common categories which are detected by the Faster R-CNN. Otherwise, we must retrain this detector using first frame annotation which is very complicated and may lead to performance degradation. Other replacement for Faster-R-CNN is the Mask R-CNN (He, Gkioxari, et al. 2017) or MNC (Dai, He, and J. Sun 2016). These techniques estimate instance segmentation directly at pixel level without the need of a semantic segmentation network. However, they do not provide precise segmentation in the boundary.

In our method, we define keyframes in a much simpler way, which fits to our video object removal application. In most of video objects removal dataset such as (Granados, Tompkin, et al. 2012; Granados, K. I. Kim, et al. 2012; Newson, Almansa, Fradet, et al. 2014), a common situation is an isolated moving object crosses another moving or static object and remains isolated after that. Therefore we define the keyframe as follows: The remaining frames are considered keyframes for a particular object when it is clearly isolated from other objects and the mask for this object can be computed easily. To this end, we rely on the *multi-OSVOS* network which returns  $K + 1$  masks  $O_i$  (which we call object mask) for each frame  $t$  and  $i \in \{1, \dots, K + 1\}$ . This allows to compute the global foreground mask  $F = \bigcup_{i=1}^{K+1} O_i$ . In these two masks, object mask are often inaccurate when objects have similar appearance, but it provides a good approximation of these objects mask. Global foreground mask, on the other hand, produces an accurate separation between background and objects, but it does not have the capability of separating each object instance. For this reason, a combination of these two maps is exploited to find the keyframes for each object. To verify if this frame is a keyframe for object  $i \in \{1, \dots, K\}$  we proceed as follows:

1. Compute the connected components of  $O_i$ . Let  $O'_i$  represent the largest connected component.
2. Compute the set of connected components of the global foreground mask  $F$  and call it  $\mathcal{F}$ .
3. For each connected component  $O' \in \mathcal{F}$  compute the overlap ratio with the current object  $r_i(O') = \frac{|O'_i \cap O'|}{|O'|}$ . If  $r_i(O') > 80\%$  and both  $O'_i$  and  $O'$  are isolated from the remaining



objects<sup>3</sup> then this is a keyframe for object  $i$ .

**Mask propagation.** Between each pair of keyframes masks are propagated forwards and backwards to ensure temporal coherence. More specifically, the forward propagation proceeds as follows: Given the mask  $M_t$  at frame  $t$ , the propagated mask  $M_{t+1}$  is constructed with the help of a patch-based nearest neighbor shift map  $\phi_t$  from frame  $t + 1$  to frame  $t$ . Typically, in the literature, the shift-map is often defined as

$$\phi_t(p) := \arg \min_{\delta} \underbrace{\sum_{q \in N_p} \|u_{t+1}(q) - u_t(q + \delta)\|^2}_{d(D_{t+1}(p), D_t(p + \delta))}$$

*i.e.* the shift  $\delta$  that minimizes the euclidean distance between a descriptor of the neighborhood of pixel  $p$  at frame  $t + 1$  to the neighborhood of pixel  $p + \delta$  at frame  $t$ . In this expression,  $N_p$  indicates the neighbor region around  $p$  which has the same size as the patch. To improve robustness and speed, this approximate nearest neighbour field is often computed using an approximate nearest neighbor field such as Coherency Sensitive Hashing (CSH) (Korman and Avidan 2011), or FeatureMatch. To reduce the dimension of the patches and speed up the search, every patch in image  $I_{t+1}$  and  $I_{t-1}$  is represented with lower dimension features. The lower dimension feature representation for each patch, helps in using fast nearest-neighbour algorithms that are obtained by using the main components in the Walsh-Hadamard space. In video, there exists connectivity of patches across frames, therefore, to capture this relation, (Ramakanth and Babu 2014) added two terms to the distance between patches. The first term is the spatial consistency term based on how far is the matching patch in frame  $t + 1$  from the current location in frame  $t$ . The second term seeks to ensure that adjacent patches flow coherently. It is required that adjacent pixels move coherently, while propagating labels, hence this term is introduced to minimise in-coherency of mapping, to accurately capture object motion. We follow this energy of (Ramakanth and Babu 2014) and use their implementation to calculate the shiftmap.

Once the shift map has been computed we propagate the mask as follows: Let  $u_t(p)$  be the RGB value of the pixel  $p$  in frame  $t$ , then the similarity between a patch  $D_{t+1}(p)$  in frame  $t + 1$  and its nearest neighbour  $D_t(p + \phi_t(p))$  in frame  $t$  is measured as

$$s_p = \exp(-d(D_{t+1}(p), D_t(p + \phi_t(p)))) .$$

Using this similarity measure the mask  $M_{t+1}$  is propagated from  $M_t$  using the following

---

<sup>3</sup>*i.e.* if  $O'_i \cap O'_j = O' \cap O'_j = \emptyset$  for all  $j \in \{1, \dots, K\}$  such that  $j \neq i$

weighted and thresholded average:

$$\tilde{M}_{t+1}(p) = \begin{cases} 1 & \text{if } \sum_{q \in N_p} s_q M_t(q) > \frac{1}{2} \sum_{q \in N_p} s_q \\ 0 & \text{otherwise} \end{cases}$$

Then, a series of morphological operations including opening and hole filling are applied on  $\tilde{M}_{t+1}$ . This propagation scheme provides a good approximation of the mask at the current frame, but sometimes contains errors. To obtain a more accurate mask, we feed it to the *mask refining* network. A final mask  $M_{t+1}$  is obtained by fusing the results with the segmentation mask provided by our *multi-OSVOS* network.

Then  $M_{t+1}$  is iteratively propagated to the next frame  $t + 2$  using the same procedure until we reach the next keyframe.

Although this mask propagation approach is useful, several artifacts may occur when objects cross each other: the propagation algorithm may lose track of an occluded object or it could mistake one object for the other.

To avoid these two kinds of errors mask propagation is performed in both forwards and backwards directions between keyframes. This gives for each object two candidate masks at each frame  $t$ :  $M_t^1 = M_t^{FW}$ , *i.e.* the one that has been forward-propagated from a previous keyframe  $t' < t$  and  $M_t^2 = M_t^{BW}$ , *i.e.* the one that has been backward-propagated from an upcoming keyframe  $t' > t$ . In order to circumvent both lost and mistaken objects we consider for each object two additional candidate masks namely

$$M_t^3 = M_t^{FW} \cap M_t^{BW} \quad \text{and} \quad M_t^4 = M_t^{FW} \cup M_t^{BW}.$$

The decision between these four mask candidates for each frame and each object is deferred to the next step, which makes that decision based on a global optimization.

### 3.3.5 Masks linking

After the backward and forward propagation, each object has 4 mask proposals (except for keyframes where it has a single mask proposal). The next step is to define a way to chose the best mask among them. This section describes a global data association method to perform this task.

**Graph-based data association** In order to decide which mask to pick for each object at each frame, we use a graph-based data association technique (GMMCP) (Dehghan, Modiri Assari, and Shah 2015) that is specially well-suited for video tracking problems. This technique does not only allow to select among the 4 candidates for a given object on a given frame. It is also capable of correcting erroneous object-mask assignments on a given frame, based on global similarity computations between mask proposals along the whole sequence. The

underlying generalized maximum multi-cliques problem is clearly NP-hard, but the problem itself is of sufficiently small size to be handled effectively by a fast Binary-Integer Program as in (Dehghan, Modiri Assari, and Shah 2015).

More specifically, given a segment containing  $N$  frames, each frame has at least  $K$  regions where  $K$  is the number of target objects. Our goal is to merge these regions together to form  $K$  different object trajectories across this segment. Formally, we define a complete undirected graph  $G = (V, E)$  where  $V$  is a set of vertexes, each vertex corresponding to a mask proposal. Vertexes in the same frame are grouped together to form a cluster.  $E$  is the set of edges connecting any two different vertexes. Each edge  $e \in E$  is weighted by a score measuring the similarity between the two masks it connects. This score will be detailed in the next paragraph. All the vertexes in different clusters are connected together. The objective is to pick a set of  $K$  cliques<sup>4</sup> that maximize the total similarity score, with the restriction that each clique contains exactly one vertex from each cluster. Each selected clique represents the most coherent tracking of an object across all frames. This problem can be solved by binary linear programming (BIP) as in (Dehghan, Modiri Assari, and Shah 2015). Since this problem is NP-hard, to reduce the computational complexity, the segment is divided into several small segments of  $M$  frames and find all tracklets in each segment by GMMCP. These tracklets are then merged together to form trajectories by the same technique. At this time, vertexes correspond to tracklets, clusters correspond to segments and the cost between two tracklets is the sum of all their elements pairwise scores.

In conclusion, our region linking technique takes as input a set  $S$  of regions of  $K$  target objects in a segment containing  $N$  frames and provides as an output  $K$  object trajectories across these frames.

In our system, this data association technique is applied to serve different purposes:

The first one is to verify the key frames selection step in Section 3.3.4. Since key frames are very important because the masks in these frame are propagated to other frames, we want to make sure that the resulting key frame obtained in the key frame selection process is correct, and remove any existing error. To this end, given a specific object, the computed key frame for this object, and a set  $S'$  of all the connected component regions in these key frames, we perform the linking technique to obtain  $K$  different trajectories. We then compare the resulting trajectories with the object trajectories obtained in key frame selection step. The target trajectory is the one with highest overlap ratio. In the target trajectory, all frames that cause these two trajectories to be not identical are removed from the key frames set. The rest are confirmed as key frames.

The second and main usage of this region linking technique is at the end of the mask propagation step when masks from all frames in the video are extracted. After the backward and forward propagation, each object has 1 to 4 object proposals, (1 in key frame and 4 in other frames). To form object masks across frames, we must decide which region should be

---

<sup>4</sup>A clique is a subgraph where all pairs of vertices are connected to each other by an edge.

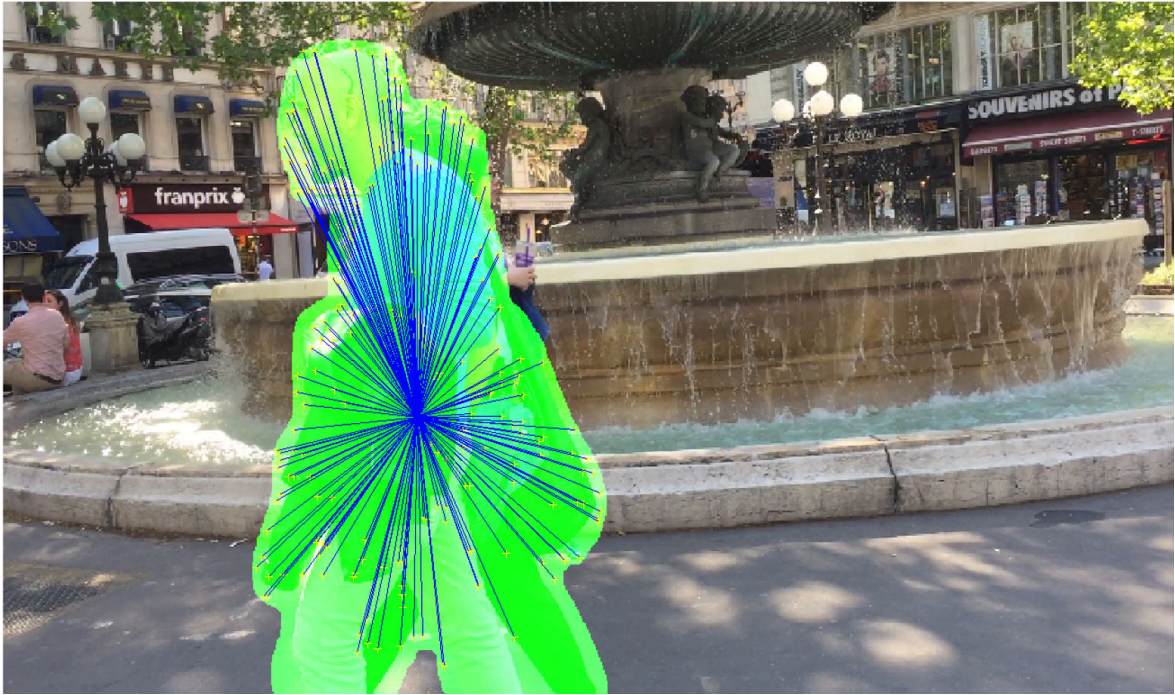


Figure 3.27 – Region descriptor: each region is described by a global histogram, a set of SURF keypoints (yellow points), and a set of vectors which connect each keypoint and the centroid of the region.

picked for a specific object. To this end, we perform this technique again. In this case, we input to the algorithm a set of region proposals from  $K$  objects in all the frames in the video, and output  $K$  final trajectories.

**Region similarity for mask linking** In our graph-based data association technique, a score needs to be specified to measure the similarity between the two masks, and the associated image data. This similarity must be robust to illumination changes, shape deformation and occlusion. Many previous approaches in multiple object tracking ([Roshan Zamir, Dehghan, and Shah 2012](#); [Dehghan, Modiri Assari, and Shah 2015](#)) have focused on global information of the appearance model, typically the global histogram, or motion information (given by the optical flow or a simple constant velocity assumption). However, when dealing with large displacements and with an unstable camera, the constant velocity assumption is invalid and optical flow estimation is hard to apply. Furthermore, using only global information is not sufficient since our object regions already resemble in global appearance. Another way to compute the similarity is using the ReID network, such as ([Wei Li, X. Zhu, and Gong 2017](#)). The idea is to train a Siamese network with triplet loss to learn the transformation from image to a vector such that two similar images would result in two similar vector in the embedded space. This idea has been used widely on person re identification task to recognize persons or faces. However, this is not a suitable approach for our problem, because our object labels are not limited to person (we do not know the label of the object in general). Moreover, we want to select the best region at the pixel level, not bounding box, so that inference to the ReID

network may cause problem.

In our work, we define a new similarity score as the combination between global and local features. More precisely, each region  $R$  is described by the corresponding mask  $M$  its global HSV histograms  $H$ , a set  $P$  of SURF keypoints (Bay et al. 2008) in it and a set  $E$  of vectors which connect each keypoint with the centroid of the mask. An example of our mask descriptor is shown in Figure 3.27. Each region is determined by four elements:

$$R := (M, H, P, E)$$

$$P := \{p_1, p_2, \dots, p_N | p_i \in M\} \text{ where } p_i \text{ is the } i\text{-th keypoint}$$

$$E = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N | \mathbf{e}_i = p_i - C\} \text{ where } C \text{ is the barycenter of } M.$$

Then the similarity between two regions is defined as:

$$S(R_1, R_2) = S_H(R_1, R_2) + \alpha S_P(R_1, R_2)$$

In this expression,  $S_H(R_1, R_2) = \exp(-d(H_1, H_2))$  where  $d$  is the cosine distance between two HSV histograms which encode global color information,  $S_P$  is the local similarity computed based on keypoint matching, and  $\alpha$  is the balance coefficient to specify the contribution of each component.  $S_P$  is computed by

$$S_P(R_1, R_2) = \sum_{p_i \in P_1} \sum_{p_j \in P_2} \gamma_{ij} \cdot w_{ij}$$

where  $\gamma_{ij}$  is the indicator function which is set to 1 if two keypoints  $p_i$  and  $p_j$  match and zero otherwise. This function is weighted by  $w_{ij}$  based on the position of the matching keypoints with respect to the centroid of the region:

$$w_{ij} = \exp\left(\frac{-d(\mathbf{e}_i, \mathbf{e}_j)}{2\sigma}\right)$$

where  $d$  is the cosine distance between two vectors and  $\sigma$  is a constant.

### 3.3.6 Post-processing

At this time, we already have  $K$  masks for  $K$  objects for all the frames in the video. Now we perform a post-processing step to make sure our final mask covers all the details of the objects. This is very important in video object removal since any missing detail can cause perceptually annoying artifacts in the object removal result. This post-processing includes two main steps:

The first step is to give a label for each region in the global foreground mask  $F_t = \bigcup_{i=1}^K O_t^i$  (the union of all the object masks produced by multi OSVOS for frame  $t$ ) which does not have any label yet. To this end, we proceed as follows: First, we compute the connected components



$C$  of all masks  $O_t^i$  and try to assign a label to all pixels in each connected component. To this end we consider the masks  $M_t^j$  that were obtained for the same frame  $t$  (and possibly another object class  $j$  by the mask linking method). A connected component is considered as isolated if  $C \cap M_t^j$  is empty for all  $j$ . For non-isolated components a label will be assigned by a voting scheme based on the ratio  $r_j(C) = \frac{|C \cap M_t^j|}{|C|}$ , i.e. the assigned label for region  $C$  will be  $\hat{j} = \arg \max_j r_j(C)$ , the one with the highest ratio. If  $r_j(C) > 80\%$  then region  $C$  is also assigned label  $j$  regardless of the voting result, which leads to possibly multiple labels per pixel.

In the second step, we do a series of morphological operations, namely opening and hole filling. Finally we dilate each object mask again with size  $9 \times 9$  this time allowing overlap between objects.

## 3.4 Experimental Results

In this section we specify our implementation details (Section 3.4.1), describe our evaluation metrics (Section 3.4.2) and report results comparing to state-of-the-art techniques over different datasets (Section 3.4.3). We evaluate our system on both single and multiple objects segmentation cases.

### 3.4.1 Implementation details

For the segmentation network architecture, we use Deeplabv3+ (L.-C. Chen, Y. Zhu, et al. 2018), both for the *multi-OSVOS* and *mask refining* networks. Details of the implementation are as follows.

**Deeplabv3+ implementation details.** Using the *ResNet101* (He, X. Zhang, et al. 2016) as feature extractor, our implementation of Deeplabv3+ employs the following network configuration: We use *output stride*  $OS = 16$ . Although setting  $OS = 8$  can produce slightly better results, setting the output stride to 16 gives us the advantage of substantially faster training. Compared to an output stride of 8, a stride of 16 makes the *Atrous Residual* block deal with 4 times smaller feature maps than its counterpart. For the multi-grid atrous convolution rates, we fix it to *Multi-grid* = (1, 2, 4). The multi-grid dilation rates are applied to the 3 convolutions inside the Atrous Residual block. With  $OS = 16$  and *Multi-grid* = (1, 2, 4), the three convolutions will have *rates* =  $2 \cdots (1, 2, 4) = (2, 4, 8)$  respectively. After the last Atrous Residual block, we put an ASSP with different dilation rates of (6, 12, 18).

**Training** The offline and online training strategies follow the procedure described in Section 3.3.3. For the data augmentation, we generate 100 pairs of images and ground truth from the first frame annotation following the protocol described in Section 3.3.3. More specifically, to reconstruct the background after extracting the objects, we use the image inpainting algorithm



of Newson, Almansa, Fradet, et al. (2014) with its default parameters. After extracting the object, we apply random flipping and with probability 0.8. Then, we apply affine transform with random rotation  $\pm 30^\circ$ , scaling  $\pm 15\%$ , and shearing 30% of the object size. We also add thin-plates spline deformation (Bookstein 1989) by choosing five control points and randomly shifting the points in x and y directions within  $\pm 10\%$  margin of the original segmentation mask width and height. We then warp the old image using these control points via thin-plates spline algorithm (Bookstein 1989). For the illumination change, we globally modify the image by randomly altering saturation  $S$  and value  $V$  (from HSV color space) with a value of  $\pm 30\%$ . After modifying the background and foreground, we place the foreground at a random position and paste it to the background via Poisson blending technique.

**Other implementation details** For the patch-based mask propagation and mask linking method, we evolved from the implementation of (Ramakanth and Babu 2014) and (Dehghan, Modiri Assari, and Shah 2015), respectively.

**Hardware and software** We implement our system using PyTorch and Matlab on a PC with an Intel i7 CPU, 32 Gb of RAM and an nVidia GTX 1080 GPU.

### 3.4.2 Evaluation metrics

For the proposed object removal system, the most crucial point is that the segmentation masks must completely cover the considered objects, including motion and transition blur. Otherwise, unacceptable artifacts remain after the full object removal procedure. In terms of performance evaluation, this means that we favor *recall* over *precision*.

The *recall* metric is defined as the ratio  $\frac{|SM \cap GT|}{|GT|}$  where SM denotes the segmentation mask, GT denotes the ground truth, and  $|A|$  denotes the area of a region  $A$ . The *precision* is the ratio  $\frac{|SM \cap GT|}{|SM|}$  between the area of the intersection and the area of the SM.

However, for video objects removal application, the ground truth mask cannot be used directly as an inpainting mask, because it does not include transition zones induced by, e.g., motion blur. This also means that the ground truth provided with classical datasets may not be fully adequate to evaluate segmentation in the context of object removal. For this reason, recent video inpainting methods that make use of these databases to avoid the tedious manual selection of objects, usually start from a *dilation* of the ground truth. For example, in (J.-B. Huang et al. 2016; Le et al. 2017; Bokov and Vatolin 2018), the ground truth mask is dilated by  $15 \times 15$  structural element. In our case, a dilation is learned by our architecture (smart dilation) at the segmentation step, as explained above. For these reasons, we compare our method with state-of-the-art object segmentation methods, after various dilations and on the dilated versions of the ground truth.

Besides, to compare with other state-of-the-art methods, we also measure the Jaccard index, or *IoU* (intersection over union), which is defined as the ratio between intersection and

union of the segmentation mask and the ground truth..

We note that the boundary precision metric in DAVIS-2016 (Federico Perazzi, Jordi Pont-Tuset, et al. 2016) is not used since we do not aim at providing the mask with an accurate contour.

### 3.4.3 Evaluation

**Datasets.** We evaluate the proposed approach on various video objects segmentation datasets. We use sequences from the DAVIS-2016 (Federico Perazzi, Jordi Pont-Tuset, et al. 2016) challenge, from the MOViC (Chiu and Fritz 2013) and from the ObMIC (M. Y. Yang et al. 2015) datasets. For the description details of these datasets, we refer the reader to Section 3.1.4. Besides, we also consider classical sequences from the papers (Granados, K. I. Kim, et al. 2012) and (Newson, Almansa, Fradet, et al. 2014), which are used for video inpainting purpose. The masks of these videos are manually segmented by the authors. Eventually, we provide several new challenging sequences containing strong appearance changes, motion blur, objects with similar appearance and possibly crossing, as well as complex dynamic textures.

**Evaluation of the Deeplabv3+** Firstly, we evaluate the performance of our implementation of Deeplabv3+, which is the backbone of our network. For this purpose, we use 3 different datasets: PASCAL VOC 2012 (Levin, Lischinski, and Weiss 2008), DAVIS-2016 (Federico Perazzi, Jordi Pont-Tuset, et al. 2016), and GyGO (Friedman et al. 2017).

For the PASCAL VOC 2012 dataset, the purpose is to evaluate the performance of our implementation of Deeplabv3+ model on a semantic image segmentation task. With this dataset, we use the augmented PASCAL VOC 2012 training set provided by (Hariharan, Arbeláez, Bourdev, et al. 2011) to train the network. The training data is composed of 8252 images, 5623 from the training set and 2299 from the validation set. The model is tested using the original PASCAL VOC 2012 val dataset. Note that we did not pre-train in the COCO dataset. Despite that, the model was able to achieve decent results on the PASCAL VOC validation set with mean Intersection over Union ( $mIoU$ ) 76.4%, this results is competitive with other results reported in other implementations on the same task.

For the DAVIS-2016 dataset (F. Perazzi, J. Pont-Tuset, et al. 2016), we train on the training set and test on the validation set without fine-tuning at the first frame. For this dataset, we achieve the  $mIoU$  73.6%, which is very competitive to other state-of-the-art methods in unsupervised tracking (the best method in DAVIS-2016 yields  $mIoU$  77.2%). This result specifies that using only the appearance information is enough in this dataset.

We also test our Deeplabv3+ on GyGO (Friedman et al. 2017) - a new video object segmentation dataset focused on e-commerce. This dataset is composed of 131 training and 24 validation sequences with high quality ground truth annotations. The sequences in this dataset are quite simple in that they are virtually devoid of occlusions, fast motions and many other usual complexities. On the other hand, the objects in these videos vary wildly in their

semantic categories: toys, models and office fluff. For this dataset, we yield the mean of Jaccard Index of 91.5%.

Several segmentation results on these three datasets are shown in Figure 3.28.

### Single object segmentation

For single video object segmentation, we evaluate our method on the DAVIS-2016 (Federico Perazzi, Jordi Pont-Tuset, et al. 2016) validation set. This dataset includes different challenges such as appearance changes, occlusions, motion blur and shape deformations. We compare our approach to recent semi-supervised state-of-the-art techniques including SeamSeg (Ramakanth and Babu 2014), ObjectFlow (Tsai, M.-H. Yang, and Black 2016), MSK (F. Perazzi, J. Pont-Tuset, et al. 2016), OSVOS (S. Caelles et al. 2017) and onAVOS (Voigtlaender and Leibe 2017). Among these methods, SeamSeg (Ramakanth and Babu 2014) is a tracking method based on patch matching between two frames, ObjectFlow (Tsai, M.-H. Yang, and Black 2016) is also a tracking method using based on optical flow combined with features extracted from a CNN. OSVOS (S. Caelles et al. 2017), onAVOS (Voigtlaender and Leibe 2017) and MSK (F. Perazzi, J. Pont-Tuset, et al. 2016) are CNN-based methods with an end-to-end learning scheme. While OSVOS (S. Caelles et al. 2017) and onAVOS (Voigtlaender and Leibe 2017) rely on static images only, MSK (F. Perazzi, J. Pont-Tuset, et al. 2016) uses the previous mask as an additional input (similar to our *mask refining network*). To ensure a consistent comparison between methods, we re-computed scores from the pre-computed segmentation masks provided by the authors.

As explained above, we consider a dilated version of the ground truth (we use a dilation by a  $15 \times 15$  structuring element, as in (J.-B. Huang et al. 2016; Le et al. 2017)). Therefore, we apply a dilation of the same size as the masks from all the concurrent methods. In our case, this dilation has both been learned (size  $7 \times 7$ ) and applied as a post-processing step (size  $9 \times 9$ ). Since the composition of two dilations with such sizes yields a dilation with size  $15 \times 15$ , the comparison is fair.

Table 3.2 reports the comparisons using the three above-mentioned metrics. We show that our model achieves significantly better *recall* than the previous state-of-the-art semi-supervised algorithms, therefore achieving its objective. The precision score remains very competitive. Besides, our method outperforms OSVOS (S. Caelles et al. 2017) and MSK (F. Perazzi, J. Pont-Tuset, et al. 2016), those having a similar approach with ours. The precision and *mIoU* scores compare favorably with onAVOS (Voigtlaender and Leibe 2017) which uses an online fine-tuning scheme.

Qualitative comparison on several sequences of DAVIS-2016 dataset are shown in Figure 3.29. We can see that the traditional mask propagation methods have a problem when the pose of the object changes drastically, such as the sequences DRIFT-CHICANE or PARKOUR. Due to the online fine-tuning on the first frame annotation of a new video, our system is able to capture the appearance of the specific object of interest. This allows it to



Figure 3.28 – Several segmentation results of our implementation of Deeplabv3+ model on three different datasets. In each dataset, top row: ground truth, bottom row: our segmentation result.



	Metric		
	Recall (%)	Precision (%)	IoU (%)
SeamSeg	59.31	73.08	50.20
ObjectFlow	70.63	90.97	67.78
MSK	82.83	95.00	79.94
OSVOS	86.78	92.38	80.58
onAVOS	87.64	<b>96.67</b>	<b>85.17</b>
Ours	<b>89.63</b>	94.31	84.70

Table 3.2 – Quantitative evaluation of our object segmentation method compared to other state-of-the-art methods, on the single object DAVIS-2016(Federico Perazzi, Jordi Pont-Tuset, et al. 2016) validation set. As explained in the text, the main objective when performing object removal is to achieve high Recall scores.

better recover from occlusions, out-of-view scenarios and appearance changes, which usually affect methods that strongly rely on propagating segmentations on a per-frame basis such as SeamSeg or ObjectFlow. Compared with CNN-based methods, our system gives better results than OSVOS or MSK and compares favorably with onAVOS. We note that OSVOS and onAVOS methods cannot guarantee temporal coherence since they only use an appearance cue.

Moreover, we observe on these examples that our approach yields full object coverage, even with complex motion and motion blur. This is particularly noticeable on the sequences KITE-SURF and PARAGLIDING-LAUNCH where our method is able to capture small details such as the parachute rope. All of the other methods (the dilated version) fail to do that. This property is beneficial for our objects removal application.

As a further experiment, we investigate the ability of dilations with various sizes to improve the recall without degrading the precision too much. For this, we plot precision-recall curves as a function of the structuring element size (ranging from 1 to 30). To include our method on this graph, we start from our original method (highlighted with a green square) and apply to it either erosions with a radius ranging from 1 to 15, or dilation with a radius ranging from 1 to 15. Again this makes sense since our method has learned a dilation whose equivalent radius is 15. Results are displayed in Figure 3.30. As can be seen from this figure, our method is the best in terms of recall, and the recall is increasing significantly with respect to the dilation size. With onAVOS method, on the other hand, the recall increases slowly, and the precision drops drastically as the dilation size increases. Basically, these experiments show that the performances achieved by our system for the full coverage of a single object (that is, with as little missed pixels as possible) cannot be obtained from state-of-the-art object segmentation methods by using simple dilation techniques.

Figure 3.31 provides qualitative results of our system on the *val* set of DAVIS-2016. The video results prove that our system is robust to challenging situations such as occlusions, fast motion, small objects, object shape deformation, camera view change and motion blur.

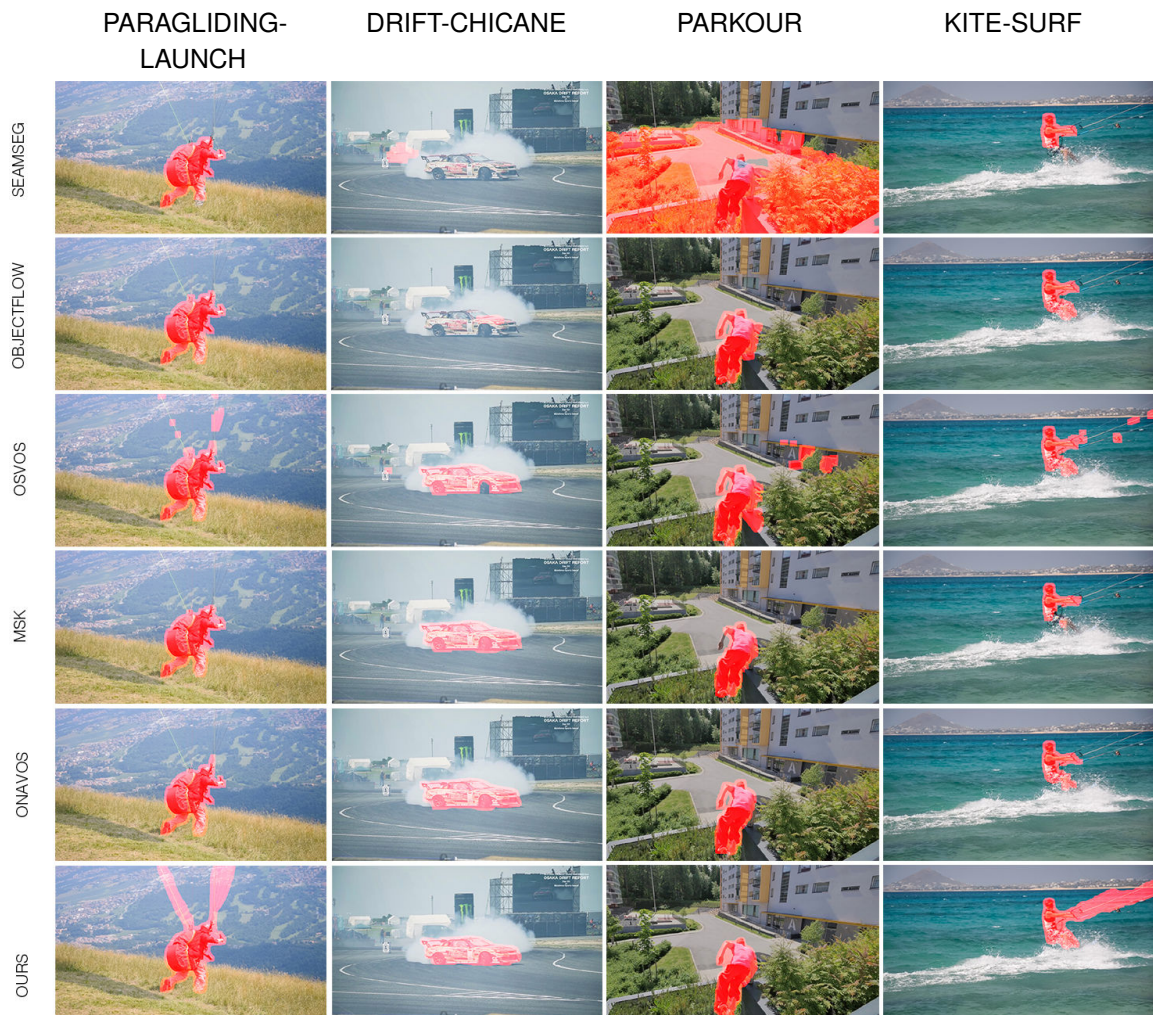


Figure 3.29 – Visual comparison of different single objects video segmentation approaches on DAVIS-2016 dataset. From top to bottom: SeamSeg ([Ramakanth and Babu 2014](#)), ObjectFlow ([Tsai, M.-H. Yang, and Black 2016](#)), OSVOS ([S. Caelles et al. 2017](#)), MSK ([F. Perazzi, J. Pont-Tuset, et al. 2016](#)), onAVOS ([Voigtlaender and Leibe 2017](#)), ours.



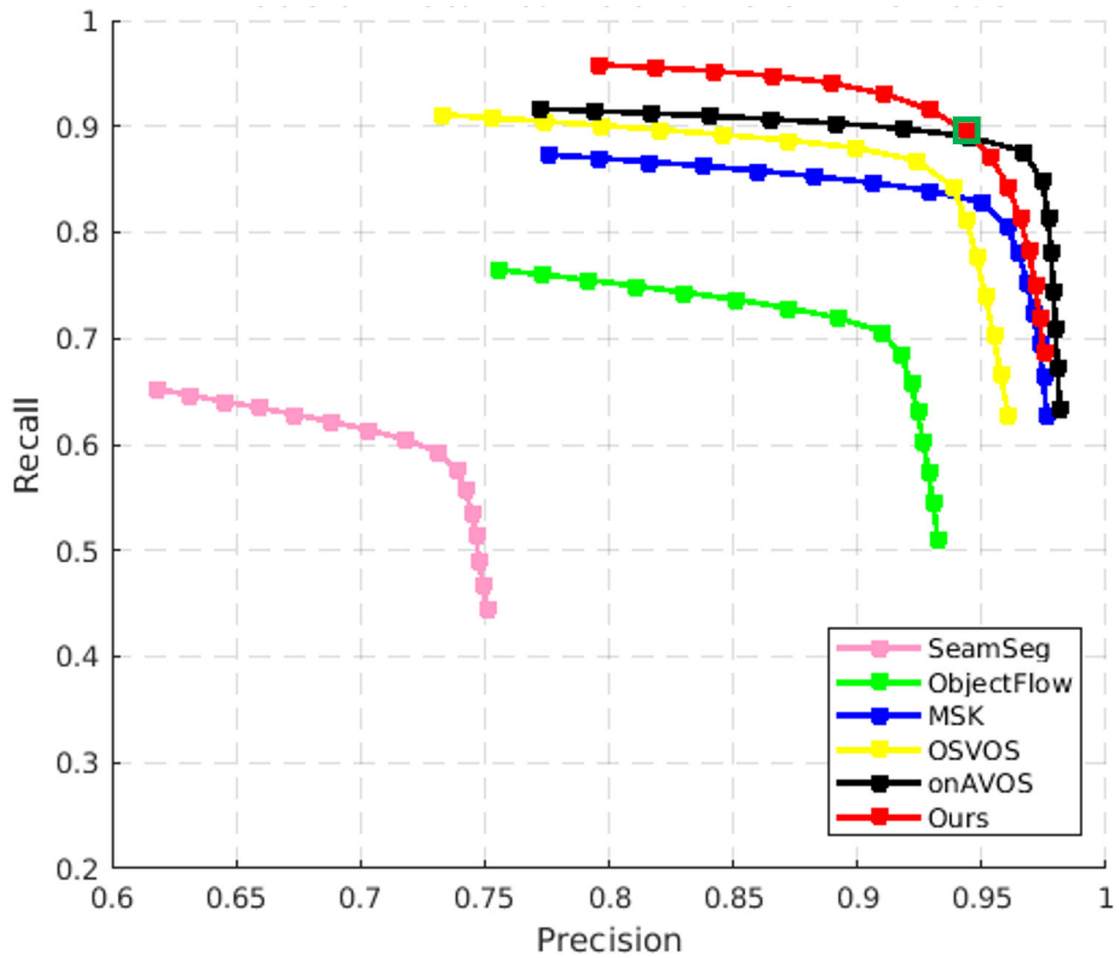


Figure 3.30 – Precision-recall curves for different methods with different dilation sizes.

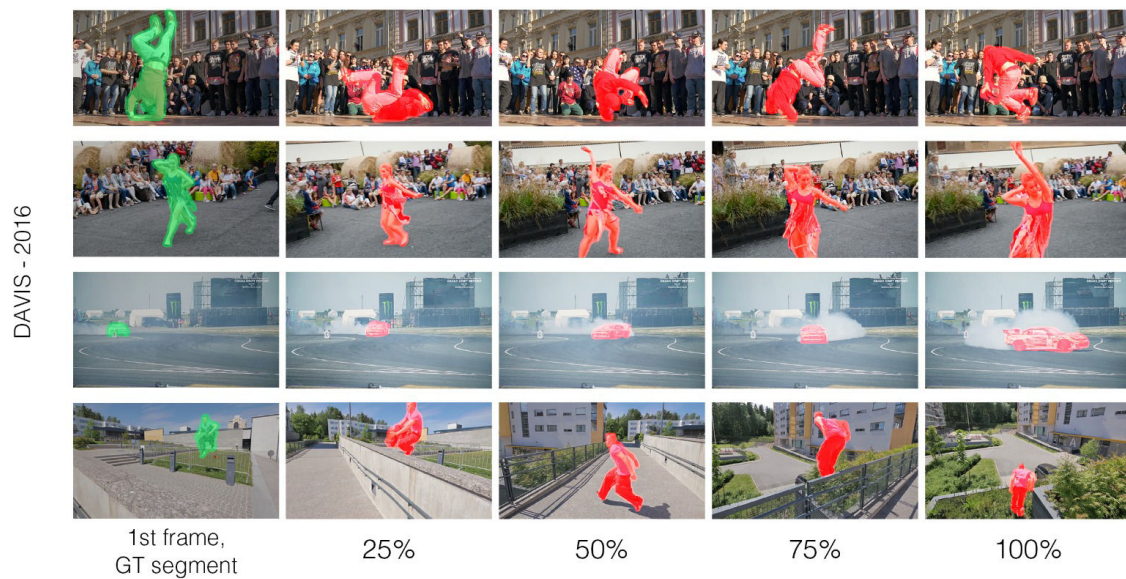


Figure 3.31 – Our single object segmentation qualitative results. Frames sampled along the video duration. Our model is robust to various challenges, such as view changes, fast motion, shape deformations, and out-of-view scenarios.

	Metric		
	Recall (%)	Precision (%)	IoU (%)
<b>MOVICs</b>			
SeamSeg	78.63	74.06	65.96
ObjectFlow	59.50	77.01	52.33
OSVOS	85.48	83.87	76.63
Ours	<b>89.28</b>	<b>87.09</b>	<b>81.58</b>
<b>ObMIC</b>			
SeamSeg	91.00	80.30	75.33
ObjectFlow	53.14	83.00	43.64
OSVOS	85.89	84.08	74.55
Ours	<b>94.42</b>	<b>88.48</b>	<b>83.81</b>

Table 3.3 – Quantitative evaluation of our object segmentation method compared to other state-of-the-art methods, on two multiple objects datasets (MOVICs (Chiu and Fritz 2013) and ObMIC (M. Y. Yang et al. 2015))

**Multiple objects segmentation.** Next, we perform the same experiments for datasets containing videos with multiple objects. For this task, we consider various datasets: MOVICs (Chiu and Fritz 2013) and ObMIC (M. Y. Yang et al. 2015), DAVIS-2017(Jordi Pont-Tuset, Federico Perazzi, et al. 2017). The MOVICs and ObMIC datasets include multiple objects, but only have one label per sequence. To evaluate the multiple object situations, we only kept sequences containing more than one object, and then manually re-annotated the ground truth giving different labels for different instances. Observe that these datasets contain several major difficulties such as large camera displacement, motion blur, similar appearances, and crossing objects. For DAVIS-2017 dataset, since the *test* ground truth was not yet available (at the time of this writing), and since our network was trained on the *train-val* set of this dataset, and the private *test-dev* set is not publicly available, we cannot compute Precision and Recall indexes as before; We can only evaluate our performance in terms of the Jaccard index that is reported by the DAVIS challenge system after having uploaded our results to the DAVIS challenge server.

Results on MoVICs and ObMIC dataset are summarized in Table 3.3. From this table, roughly the same conclusions as in the single object can be drawn, namely the superiority of our method in term of recall score, without sacrificing much the precision score.

For DAVIS-2017 (Jordi Pont-Tuset, Federico Perazzi, et al. 2017), we evaluate on the *test-set* of the challenge. Since the ground truth of this set is private, we can not perform the same evaluation as before. In this case, we only measure the *mIoU* score by uploading our result to the evaluation server and receive the returning score. We note that before uploading the results, our segmentation masks are eroded by size  $15 \times 15$ , to compensate for the *smart dilation* layer. Doing this way somehow decreases the performance of our method. The *mIoU* score of our method, together with the scores of the top performance methods are reported in Table 3.4. We can clearly see that even our system cannot beat the bleeding edge state-of-the-art

	Methods						
	PReMVOS	CINM	OSVOS-S	onAVOS	OSVOS	LucidTracker	Own
<i>mIoU</i>	<b>67.5</b>	64.5	52.9	49.9	47.0	63.4	54.2

Table 3.4 – Comparison with the top performance methods on DAVIS-2017 *test-dev* set.

	Metric		
	Recall (%)	Precision (%)	IoU (%)
<b>GRANADOS-S1</b>			
OSVOS	62.04	59.17	52.15
Ours	80.12	86.31	67.53
<b>GRANADOS-S3</b>			
OSVOS	74.42	87.00	63.02
Ours	80.12	86.31	67.53

Table 3.5 – Quantitative evaluation of our object segmentation method compared to the OSVOS segmentation method (S. Caelles et al. 2017), on two sequences manually segmented for inpainting purposes (Granados, Tompkin, et al. 2012)

results (PReMVOS, CINM and LucidTracker), but it achieves a better score than OSVOS (S. Caelles et al. 2017), OSVOS-S (Sergi Caelles et al. 2017), and onAVOS (Voigtlaender and Leibe 2017). The top scores of other methods like PReMVOS (Luiten, Voigtlaender, and Leibe 2018), CINM (Bao, B. Wu, and W. Liu no date) and LucidTracker (Anna Khoreva et al. 2017) are achieved by the combination between different networks for appearance, optical flow, location, etc. which make these systems very sophisticated.

Since our final objective is creating masks for objects removal application, we test our method with the sequences of (Newson, Almansa, Fradet, et al. 2014) and (Granados, K. I. Kim, et al. 2012), which are used for video inpainting purpose. The ground truth segmentation mask for these sequences are manually annotated by the authors. Table 3.5 provides a comparison between OSVOS (S. Caelles et al. 2017) and our approach on two sequences from (Granados, Tompkin, et al. 2012). These sequences have been manually segmented by the authors of (Granados, Tompkin, et al. 2012) for video inpainting purposes. On such extremely conservative segmentation masks (in the sense that they over-detect the object), the advantage of our method is particularly strong.

Figure 3.32 illustrates the segmentation results of other segmentation methods, namely SeamSeg, ObjectFlow and OSVOS. The segmentation masks for these methods are obtained using their publicly available implementation with the default parameters. It shows multiple objects segmentation results on MOVICs (Chiu and Fritz 2013), the LES LOULOUS sequences of (Newson, Almansa, Fradet, et al. 2014) and the GRANADOS-S1 sequence (Granados, K. I. Kim, et al. 2012) respectively. In the multiple objects cases, these examples illustrate the capacity of our method to deal with complex occlusions. This cannot be achieved



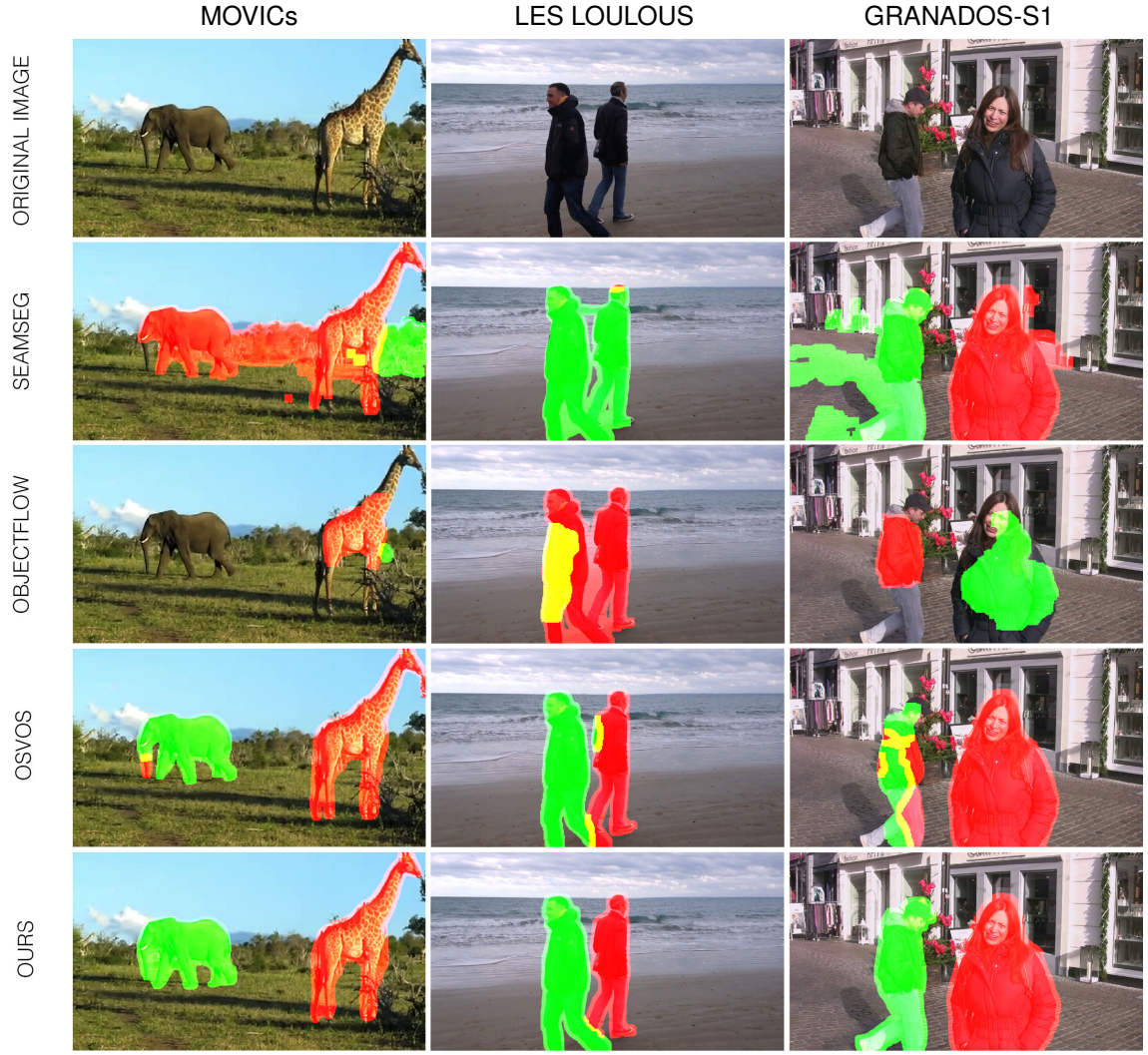


Figure 3.32 – Visual comparison of different method on multiple objects segmentation case. From top to bottom: Original image, SeamSeg (Ramakanth and Babu 2014), ObjectFlow (Tsai, M.-H. Yang, and Black 2016), OSVOS (S. Caelles et al. 2017), ours.

with mask tracking methods such as ObjectFlow (Tsai, M.-H. Yang, and Black 2016) or SeamSeg (Ramakanth and Babu 2014). The OSVOS method (S. Caelles et al. 2017) yields some confusion between objects, probably because the temporal continuity is not taken into account by this approach.

In Figure 3.33, we show our segmentation results on three different datasets: DAVIS-2017, MoVIC- ObMIC, and sequences from (Granados, Tompkin, et al. 2012) and (Newson, Almansa, Fradet, et al. 2014) which are used for video inpainting purpose. Qualitative results of these different datasets show the success of our system handling with multiple objects, full and partial occlusions, distractors, multiple instances of the same semantic class, and out-of-view scenarios.



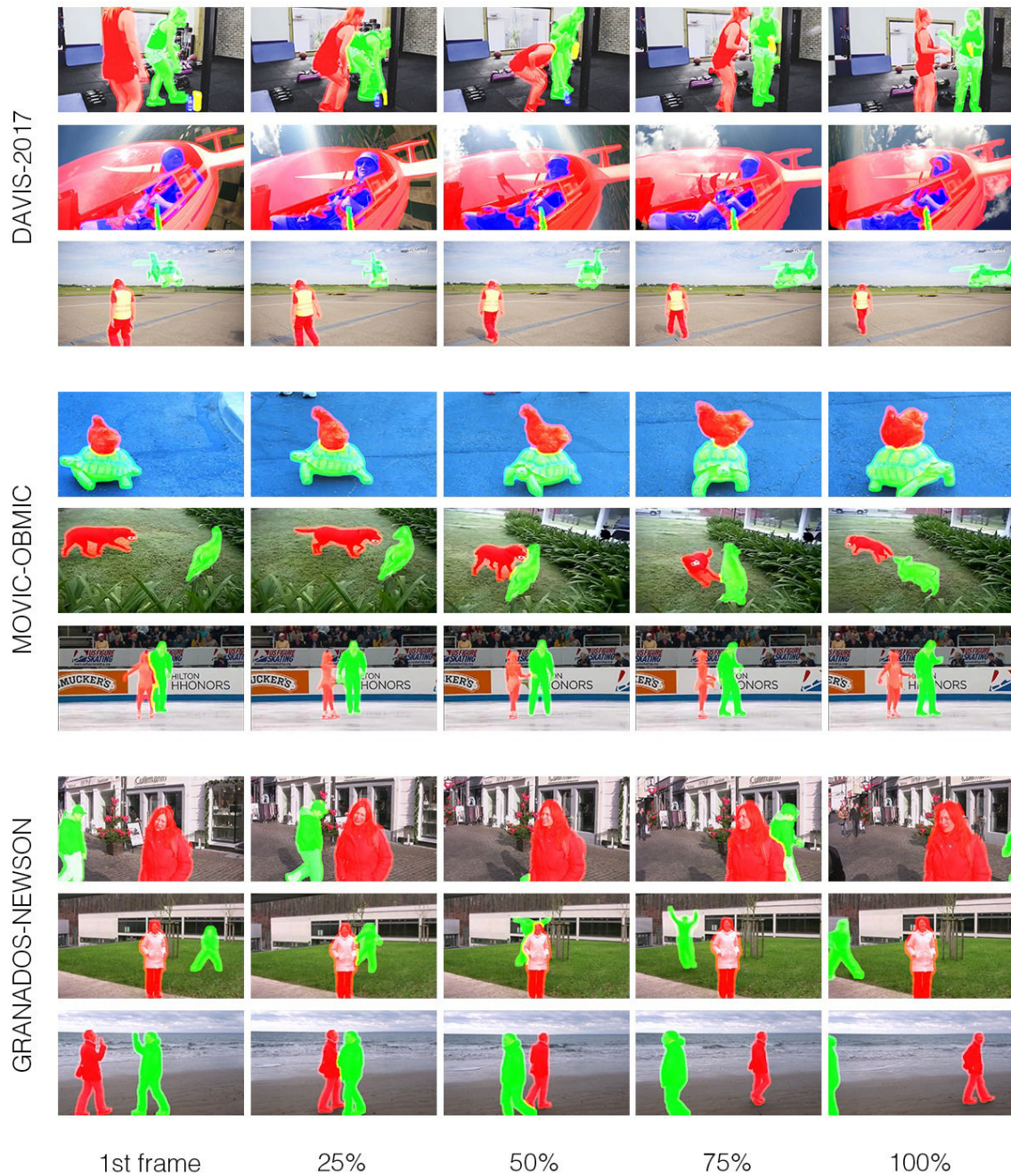


Figure 3.33 – Qualitative results of our segmentation system on 3 different multiple objects segmentation datasets. Frames sampled along the video duration. Our model is robust to various challenges, such as full and partial occlusions, distractors, multiple instances of the same semantic class. From top to bottom: DAVIS-2017 dataset, MoVIC and ObMIC datasets, sequences from (Granados, Tompkin, et al. 2012) and (Newson, Almansa, Fradet, et al. 2014) which are used for video inpainting purpose.



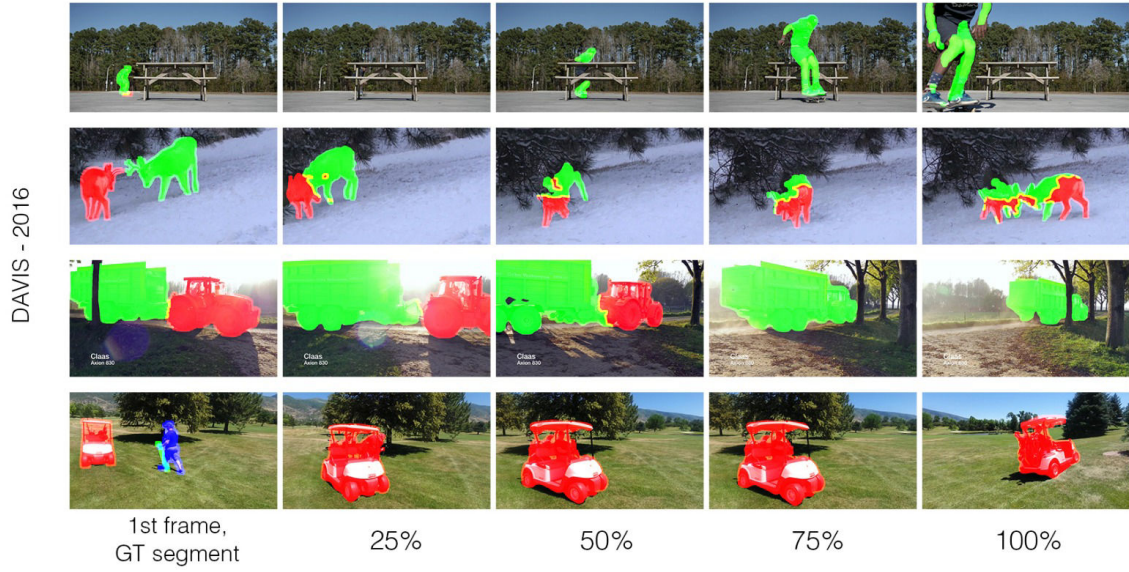


Figure 3.34 – Several failure cases of our system of DAVIS-2017 dataset.

## 3.5 Concluding remarks

### 3.5.1 Limitations

Although our video objects segmentation system is quite robust, there are several cases where our method fails. A few representatives of failure cases are visualized in Figure 3.34. Since we are using only the annotation of the first frame for training the network, a clear failure case is caused by dramatic view point changes of the object from its first frame appearance, as in the first row of Figure 3.34. By annotating multiple frames, we not only have more examples to train but also have more key frames, which means we can reduce the error propagation. This also enables users to correct the error during the propagation process.

The proposed approach also under-performs when objects are stuck together during the whole video. In this case, we do not have any keyframe excepts the first frame. Propagation only from the first frame will cause error propagation, as shown in the second row of Figure 3.34.

Another problem is the wrong label assignment. This problem appears when two objects stick together, for example in row 3-4 of Figure 3.34, when two objects are close to each, the system merges them together to form a unique object. This is a common problem with mask tracking approaches, and it remains challenging for any video object segmentation system.

We also observe that our system struggles to track the fine structures or details of the object, e.g. the skate in in the first row and the golf stick in the fourth row of Figure 3.34. This is the issue of the convnet architecture, due to the several pooling layers the spatial resolution is lost and hence the fine details of the object are missing. Another choice could be to crop a rectangle a round the object, as done in (Yi Li et al. 2017a). They report to have better segmentation with small objects.

### 3.5.2 Conclusions and future works

We have presented a novel approach to video object segmentation. By treating video object segmentation as a guided instance segmentation problem, we have proposed to use a pixel labelling convnet for frame-by-frame segmentation. By exploiting both offline and online training with image annotations only, the network is able to produce highly accurate video object segmentations. Besides, we propose a mask tracking module to ensure temporal continuity and a mask linking module to ensure the identity coherence across frames. Moreover, we introduce a simple way to learn the dilation layer in the mask, which helps us creating a good mask for video objects removal application. The proposed system is generic and reaches competitive performance on different video segmentation datasets. The method can handle different types of input annotations and our results are competitive. Future work should consider exploring more efficient way to detect key frames, a better way to preserve object identity (using a ReID network could be a promising way). Moreover, a motion cue (e.g. optical flow) should be added to the network to improve the overall score. Finally, more users interactions should be allowed to create a flexible mask selection tool for a video objects removal system.

# 4

## Objects removal with a static background

### Contents

---

<b>4.1</b>	<b>Introduction to image/video inpainting</b>	<b>102</b>
4.1.1	Overview	102
4.1.2	Applications	103
4.1.3	Challenges	104
4.1.4	Video objects removal with static background	105
<b>4.2</b>	<b>Related works</b>	<b>106</b>
<b>4.3</b>	<b>Proposed method</b>	<b>108</b>
4.3.1	Motion field reconstruction	109
4.3.2	Motion-guided pixel reconstruction	110
4.3.3	Poisson blending	111
<b>4.4</b>	<b>Results</b>	<b>113</b>
4.4.1	Implementation details	113
4.4.2	Evaluation	114
	Visual results	114
	Visual comparison with other state-of-the-art methods	114
<b>4.5</b>	<b>Conclusion</b>	<b>120</b>

---

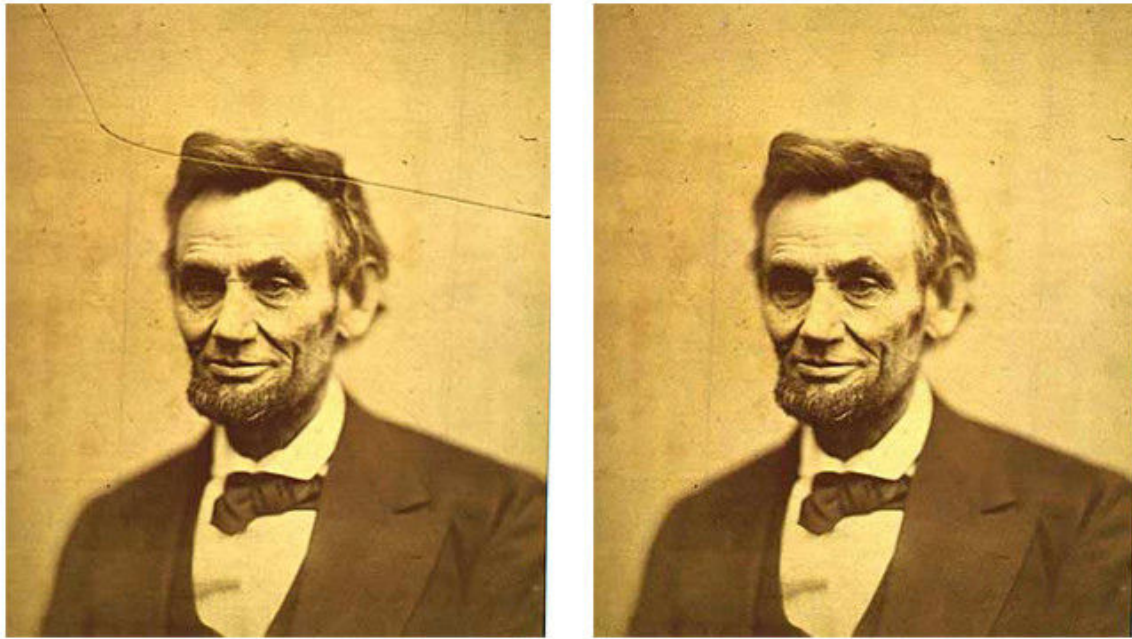


Figure 4.1 – An example of photo restoration. The scratches in the photo can be removed by inpainting. Left: An 1865 Photograph of Abraham Lincoln taken by Alexander Gardner which contains scratches. Right: Image restored with an inpainting algorithm. Image credit: <https://sites.tufts.edu/kienle/inpainting/>.

In the previous chapter, we discussed how to create dynamic masks for tracking a given object (or a given collection of objects) in a video. Our goal now is to remove an object from the video given its dynamic mask, which is nothing else than the video inpainting problem. In this chapter, we first consider the simpler situation where the background of the video is static. We show that this simpler inpainting problem can be solved very efficiently using a rather naive motion-based technique. A more complicated algorithm adapted to dynamic textures and complex motions will be presented in Chapter 5. Let us first introduce briefly the video inpainting problem. Next we will describe our simple yet effective approach to remove objects on static background. Last, we will present some results, do comparisons, and discuss the limitations of this approach.

## 4.1 Introduction to image/video inpainting

### 4.1.1 Overview

In fine art restoration, the word *inpainting* refers to the hand restoration of paintings that have undergone physical alterations such as cracks or scratches. In the digital world, inpainting is generally defined as the process of either restoring missing pixels and damaged regions in images or videos, or removing unwanted objects, in both cases in the most plausible way. This problem has many applications (see also below): editing, denoising, superresolution,

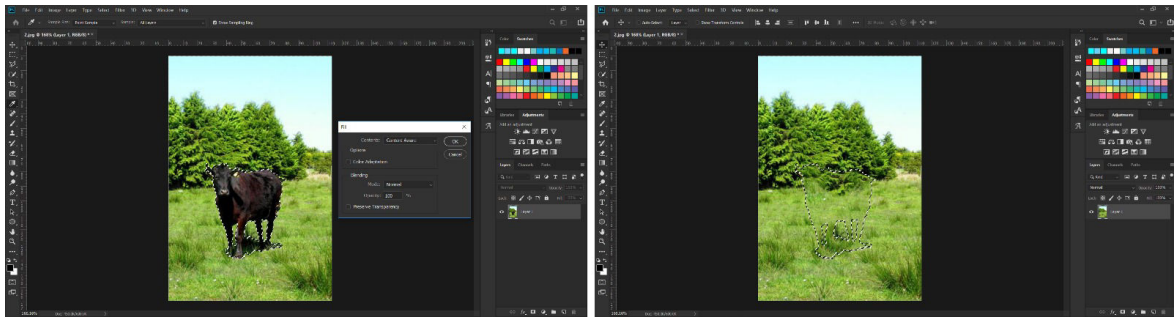


Figure 4.2 – Using Content Aware Fill tool to remove an object in Photoshop.

demosaiicing, interpolation, compression, etc. (see below). Figure 4.1 shows a typical example of an old photograph where scratches have been removed using inpainting. This problem has been studied a lot in the past twenty years both in academia and industry. The most supervised approaches are based on classical editing tools, for example the *cloning buffer* which allows users to inpaint a region by simply copying the pixel values picked somewhere else in the image. This repetitive work requires a lot of precision and several iterations to obtain a good result.

Many contributions have tried to make this editing task easier, either semi-supervised in a way which requires a minimal human intervention, or even fully automatic. The major difficulty is always that an inpainted region must look as much plausible as possible with respect to the rest of the image.

A popular tool for image inpainting is Content-Aware Fill in Photoshop which helps users remove some objects in images automatically. Figure 4.2 illustrates the use of Content-Aware Fill to remove an object. The user only needs to draw a contour to select what he/she wants to remove, and then the tool removes it automatically. Inpainting a region may require only a few seconds.

While several semi-supervised commercial tools are available and rather effective for still image inpainting, it is not the case for videos. This is certainly due to the fact that the additional time dimension makes the problem significantly more difficult because our visual system performs extremely well in detecting even tiny changes along time.

### 4.1.2 Applications

Applications of image/video inpainting are numerous, we already mention a few of them. We outline below the most common and practical ones.

- **Removing unwanted objects:** Undesired objects can be removed from an image/video using inpainting techniques. It is often used in post-media processing, for instance to remove an unwanted microphone in a movie scene (Figure 4.3).
- **Restoration:** Inpainting is also used to restore the content of an image/video, for instance typically to remove scratches in vintage film/photo (N. C. Tang et al. 2011).





Figure 4.3 – Removing a microphone from a movie scene using inpainting (image from [\(Bertalmio, Sapiro, et al. 2000b\)](#)).

Another typical application is the correction of data loss in satellite imaging. ([Ebdelli, Le Meur, and Guillemot 2015](#)).

### 4.1.3 Challenges

Inpainting is a typical example of an ill-posed inverse problem: given an image/video and a mask, there are many possible ways to reconstruct the information behind the mask, but not all ways are plausible. In other words, the content of the reconstructed region must be consistent with the structure, texture, color and brightness image information around it. This is a very challenging problem because there is so far no clear quality measure which could be used to evaluate the "plausibility" of the reconstruction.

The temporal dimension in video inpainting is both an advantage and a challenge. It is an advantage because due to motion some objects which are hidden by the inpainting domain may become visible at some earlier or later moments. But it yields also several challenges. As already said, the first challenge is that the consistency must be ensured across time because our visual system is very sensitive to temporal inconsistencies. In particular, it is not a good solution to inpaint each frame individually and independently from the others. The second challenge is how to deal with moving objects and dynamic textures. The moving objects which are occluded must be reconstructed consistently, and the spatial and temporal structure of the dynamic background must be preserved. Even when the background is static, the camera



Figure 4.4 – Sample frames of the PET2009-S2L1 sequence (Ferryman and Shahrokni 2009). This sequence contains several pedestrians who are moving in front of a static background. The objective is to remove the pedestrians marked in red.

movement and the illumination changes may generate changes between consecutive frames and the time-space neighborhood of the inpainting domain may be more difficult to use. As a last challenge, the amount of data to process may be huge in a video. This computational issue is a real obstacle which makes most video objects removal tools impractical.

#### 4.1.4 Video objects removal with static background

Let us consider a typical video completion scenario such as sequence PET2009-S2L1 (Ferryman and Shahrokni 2009) which contains a few moving pedestrians. Some representative frames are shown in Figure 4.4. Our purpose is to remove some moving pedestrians in the video and reconstruct the background. In this sequence, the camera and the background are static, and there is no illumination change. In this simple scenario, if we want to reveal a static background behind a moving object, all that we have to do is to wait that the object has moved enough so that we may recover the real information. Indeed, because the object is moving, the current background appears in some of the previous or some of the future frames. So we basically can copy those pixels from previous/ future frames into the current frame. Doing so, the background can be reconstructed extremely quickly, which is a huge advantage in real-life applications.

However, such approach works only for the particularly simple situation where the expected background is revealed sooner or later. There are of course much more difficult scenarios. For example, what if we want to remove a static occlusion such as the tripod in Figure 4.4, or what if the camera is moving and the illumination changes over time? In these cases, does any fast and straightforward solution still exist?

In this chapter, we will explore briefly a fast and straightforward video restoration algorithm which works well in many situations, but more specifically when the background we want to reconstruct is static without any dynamic texture. The basic concept of our technique is that if the occlusion or the camera moves, the previously occluded information appears in other frames. Therefore it can be found and copied back to its appropriate location. Motion information is used to track the missing pixels and establish a trajectory from the missing

region toward the source region. This concept is the key component of many video inpainting methods which will be reviewed in the next section.

In the upcoming sections, several problems associated with object removal under static background are addressed. First, we make a quick detour of the methods which are related to our simple approach in Section 4.2. For a complete review of different methods for video inpainting, we refer the reader to Chapter 5. Next, we describe the method we propose in Section 4.3. We provide visual results of our method and compare them to previous works in Section 4.4.

## 4.2 Related works

In the video inpainting literature, motion-guided pixel reconstruction is a well-known approach. For example, ([Grossauer 2006](#)) proposes a method for removing blotches and scratches in old movies using optical flow. The blotches are detected using the optical flow itself, by comparing forward and backward vectors. The method then copies and pastes uncorrupted pixels from the directly previous and next frames. If no uncorrupted pixels are available, an image inpainting method is applied for restoration. In this approach, the optical flow is computed without any modification inside the occlusion, although the presence of the film defects influences it.

Other methods use a similar idea of motion transfer to reconstruct more substantial occlusions. They usually try to restore the motion field in the occlusion domain either by gradually propagating motion vectors ([Matsushita, Ofek, Ge, et al. 2006](#)), or by sampling space-time motion patches ([Shiratori et al. 2006](#); [N. C. Tang et al. 2011](#)), or by interpolating the missing motion ([S. You et al. 2013](#); [Bokov and Vatolin 2018](#)). With the complete motion field, pixel values from outside the occlusion are propagated into the occlusion to complete the missing region.

Having a complete motion field is very useful for video stabilization ([Matsushita, Ofek, Ge, et al. 2006](#)), which improves the quality of the inpainting as we will see later. With a different purpose, ([Shiratori et al. 2006](#)) completes a damaged video by transferring sampled motion fields from the available portions of the video to predict motion in the hole. The transferring scheme is done by sampling space-time patches. ([N. C. Tang et al. 2011](#)) also uses this principle but adopt a weighted priority of patches to select the best match. As discussed in ([Shiratori et al. 2006](#)), comparing with sampling color directly, this indirect approach is useful in cases where periodic motion is absent. However, it works only on stationary video and may easily cause over-smoothing artifacts due to the pixel values interpolation during color propagation. Similarly, ([S. You et al. 2013](#)) interpolates the missing optical flow field from surrounding regions with the assumption of spatial continuity of the motion. The interpolated motion can be traced to fill in the missing color information. With the same principle, ([Bokov and Vatolin 2018](#)) estimates the optical flow for the missing region in each



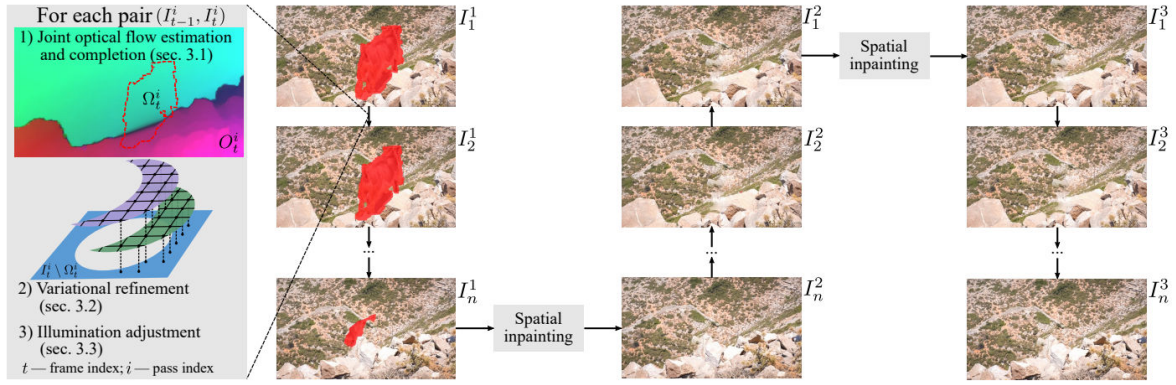


Figure 4.5 – The video-completion pipeline proposed in (Bokov and Vatolin 2018). The authors use temporally consistent propagation of known input-video to the missing region (denoted in red) using optical flow. Image taken from (Bokov and Vatolin 2018).

frame independently. The results are accumulated over the input sequence to construct a map from the missing regions to the known regions. Then the authors improve the resulting accumulation of flow and color error by applying frame-by-frame variational refinement and illumination adjustment.

A major problem of these approaches is that their performances strictly rely on the reliability of the optical flow. In general, approaches that use optical flow are less efficient in case of large or complex pixel displacements. Therefore, they do not work well for the completion of videos with complex motions. Despite that, the advantage of this approach is that the temporal coherence is maintained and the computational complexity is small which makes the method applicable to real-world applications in some domains where fast processing is required like in augmented reality (Herling and Broll 2012).

As discussed in (B. Xu et al. 2017), interpolating the optical flow and using it to update color information in the occluded region is a sub-optimal solution. To overcome this problem and reduce the dependence of the method on the precision of the optical flow, several attempts have been proposed to estimate jointly the optical flow and the color information inside the occlusion. For example, (J.-B. Huang et al. 2016) proposes an optimization framework which performs alternate minimization between optical flow and pixel values. The authors use two-dimensional patches to reconstruct color values as in (Pritch, Kav-Venaki, and Peleg 2009). To ensure temporal consistency, they rely on the estimated flow field which is refined after pixel values are obtained in the missing region. Doing so they can handle camera movement and large occlusion. This method, however, has very high computational complexity, it requires hours to inpaint a 100 frames 480p video sequence.

Another method which uses the same principles and adapts them to spherical image sequences is introduced in (B. Xu et al. 2017). The authors try to reconstruct a static background by an iterative estimation of motion and color field. The geometric properties of spherical images are considered to deal with spherical videos. Since a common optical flow estimator can hardly deal with large pixel displacements in spherical videos, the authors first estimate

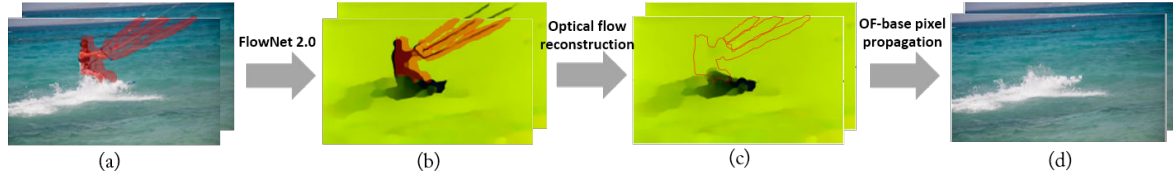


Figure 4.6 – The global pipeline of the optical flow-based propagation approach for reconstructing a static background: From input video (a), forward/backward optical flow fields are estimated by FlowNetv2 (Ilg et al. 2017) (b), they are inpainted by an image inpainting algorithm (c). From these optical flow fields, pixels from the source region are propagated into the missing region (d).

the rotational motions and then pre-process the video by derotating it. After that, optical flow fields are estimated from this derotated video and then initialized via interpolation. Following this, an iterative optimization method inpaints motion and color information alternatively in a coarse-to-fine way. Motion is inpainted and refined to enforce spatial and temporal coherence in occluded regions. Color information is inpainted by iteratively propagating corresponding pixels from other frames based on inpainted motion, in a temporally coherent manner. The main problem of this kind of approach is that the computational complexity is enormous while the performance depends on the type of background. This type of approach can actually only work if the background is static without dynamic texture or moving objects to reconstruct.

Recently, a simple yet efficient greedy method has been proposed in (Bokov and Vatolin 2018): first the optical flow is estimated and then used to compute color values in the missing region. The results obtained with this simple method are comparable (but obtained 100× faster) than those produced by the more complicated method of (J.-B. Huang et al. 2016). In the same spirit as (Bokov and Vatolin 2018), both for simplicity and speed, we propose a simple approach based on motion transfer to deal with the static background situation.

### 4.3 Proposed method

Our motion-guided pixel reconstruction approach is composed of three main steps which are illustrated in Figure 4.6. After stabilizing the video to compensate the camera movements, we use FlowNetv2 (Ilg et al. 2017) to estimate forward and backward optical flow fields. These optical flow fields are then inpainted using a classical image inpainting method to fill in the missing information. Next, these inpainted motion fields are concatenated to create a correspondence map between pixels in the inpainting region and known pixels. Lastly, missing pixels are reconstructed by a copy-paste scheme followed by a blending technique to reduce artifacts. We will present these steps in more details.





Figure 4.7 – Motion field reconstruction by inpainting the optical flow field. From two consecutive images (top row), we compute the optical flow field by FlowNetv2 (bottom left), then we use an image inpainting method to reconstruct the optical flow field inside the occlusion (bottom right).

### 4.3.1 Motion field reconstruction

Since our method relies on the motion transfer principle, the first step is to reconstruct the optical flow inside the occlusion. Let us recall that an optical flow is a dense correspondence between two images, which supposedly respects the intensity consistency assumption. A more detailed description of optical flow and different ways to compute it are presented in Section 2.2.2 of Chapter 2. To recap, let  $(u, v)$  represent the optical flow vector, with  $u$  corresponding to the horizontal component and  $v$  corresponding to the vertical component. The optical flow is often defined using the following generic minimization problem.

$$(u, v) = \min_{u, v} \int_{\Omega} E_{data}(I, u, v) + E_{smooth}(u, v)$$

In this equation,  $\Omega$  is the occlusion domain,  $E_{data}$  corresponds to the intensity consistency assumption and  $E_{smooth}$  is the smoothness prior which is used to alleviate the ill-posedness of the problem.

In a video inpainting situation, it is necessary to overcome the problem of the presence of the occlusion to obtain a reliable optical flow. A possible approach to reconstruct the optical flow inside the damage region is a smooth interpolation, for instance, in the framework of a variational approach, by ignoring the data term and using only the smoothness term in the missing regions. More formally, using only the smoothness term and setting  $E_{data}$  to 0 in the occluded areas, we have the following energy:

$$E(I, u, v) = \int_{\Omega} \begin{cases} E_{smooth}(u, v) & \text{if } (x, y) \text{ is occluded,} \\ E_{data}(I, u, v) + E_{smooth}(u, v) & \text{otherwise.} \end{cases}$$

This approach has been used in several methods for video inpainting purposes, such as (S. You et al. 2013; Bokov and Vatolin 2018). For example, (Bokov and Vatolin 2018) computes the optical flow without the data term inside the occlusion, then uses the computed optical flow to transfer pixel values. The optical flow is obtained by the method of (Kroeger et al. 2016), which is computed by alternating patch-based gradient descent and variational refinements.

However, this approach leads to an over-smoothed and unreliable optical flow. Therefore, we choose to reconstruct the optical flow using more sophisticated image inpainting techniques. We treat the optical flow field as an image and reconstruct the optical flow in an inpainting-like manner. More specifically we first compute outside the missing region the forward/backward optical flow fields between two consecutive frames using the FlowNetv2 approach from (Ilg et al. 2017). We then rely on the image inpainting method from (Newson, Almansa, Gousseau, et al. 2017) to interpolate these motion fields. Although this method has higher computational complexity, it can reconstruct more sophisticated optical flow and give a better result than simply interpolating the optical flow. An example of optical flow reconstruction is shown in Figure 4.7.

### 4.3.2 Motion-guided pixel reconstruction

Once the motion field inside the occlusion is filled, it can be used to propagate the pixel in the source toward the occlusion. This process can be done in several ways.

A typical way would be to propagate the pixel values frame by frame as in (Herling and Broll 2012). In this approach, the sub-pixel accuracy of optical flow requires an interpolation scheme. As it has been observed by (Kokaram, Collis, and Robinson 2005), using the optical flow along with a bi-linear interpolation scheme to propagate the information presents a significant blur in the results after just a few frames. Therefore, the completion of large time intervals is not straightforward. In (Kokaram, Collis, and Robinson 2005), the problem is alleviated by using a higher order interpolation scheme. However, though higher order interpolation schemes behave better than the bilinear one, the blur artifact still persists. For that reason, these approaches allow completion only over a relatively small number of frames.

This blurry problem is then solved in (Facciolo et al. 2011) and later in (Sadek et al. 2013) by a numerical scheme which is the deblurring scheme for the convective derivative, which makes the propagation possible through a large number of frames without the blurring artifacts. However, the complexity is very high so that it loses the advantage of fast video completion.

We adopt a different approach to reconstruct the pixel values inside the occlusion. We create a map which associates each pixel in the occlusion with their correspondent in the

source region. This map is obtained by concatenating the optical flows from frame to frame. The concatenation of optical flow vectors is done with bilinear interpolation. We do both forward and backward optical flow, which leads us to the forward map and the backward map. From this map, pixels value can be obtained by a copy-paste method. The intuition behind this method is that if the pixel is not covered by the occlusion at some points in the map, then it is possible to use its color values at that spatio-temporal position for restoration. We do copy-paste pixel values rather than patches because patch-based synthesis typically requires to compute the average of several similar patches which leads to blurry results.

We do three passes: first a forward pass using the forward map to reconstruct the occlusion. Then a backward pass using the backward map. After these two passes, the remaining hole is the region where information behind it is never revealed in the video. To fill this hole, we adopt the image inpainting method of (Newson, Almansa, Gousseau, et al. 2017) to complete this region in one key frame, usually the middle frame of the video, and then propagate information from this frame to other frames in the video. In several rare cases where some occlusions are still missing (they usually appear in the border of the video), we inpaint them by the image inpainting technique of (Newson, Almansa, Fradet, et al. 2014).

A major problem of this method is that the greater is the number of optical flow vectors which are concatenated, the more errors are accumulated, which leads to very poor reconstruction. Therefore, we limit the temporal length of the map to 20 frames. This means we only allow to copy from a source region which is not further than 20 frames to the current frame. Using this scheme, to complete one pass (backward or forward), we need to repeat several sub-passes. In each sub-pass, information within 20 frames is used to reconstruct the occlusion of a target frame. After finishing one sub-pass, all the reconstructed pixels become sources for the next sub-pass. This approach can be seen as a compromise between a frame-by-frame propagation and a propagation which uses optical flow concatenation.

### 4.3.3 Poisson blending

Real-life videos often contain illumination changes, especially outdoor scenes. Since our method is based on copying and pasting pixel values from the source, it maintains color value consistency. Therefore, when the illumination of the sources is different from the current restoration frame, a visible artifact across the border of the occlusion appears. A common way to resolve this is by applying a blending technique, e.g Poisson blending (Pérez, Gangnet, and Blake 2003b). The objective of Poisson blending algorithm is to fuse a source image and a target image in the gradient domain. Because the human visual system is more sensitive to contrast than intensity values, Poisson blending achieves a more realistic composition than naively pasting two similarly colored images together. The principle behind Poisson blending is that solving the Poisson equation with Dirichlet boundary conditions removes potential lighting differences. However, applying Poisson blending frame-by-frame may impact negatively the temporal consistency. To maintain it, we adopt the recent method of

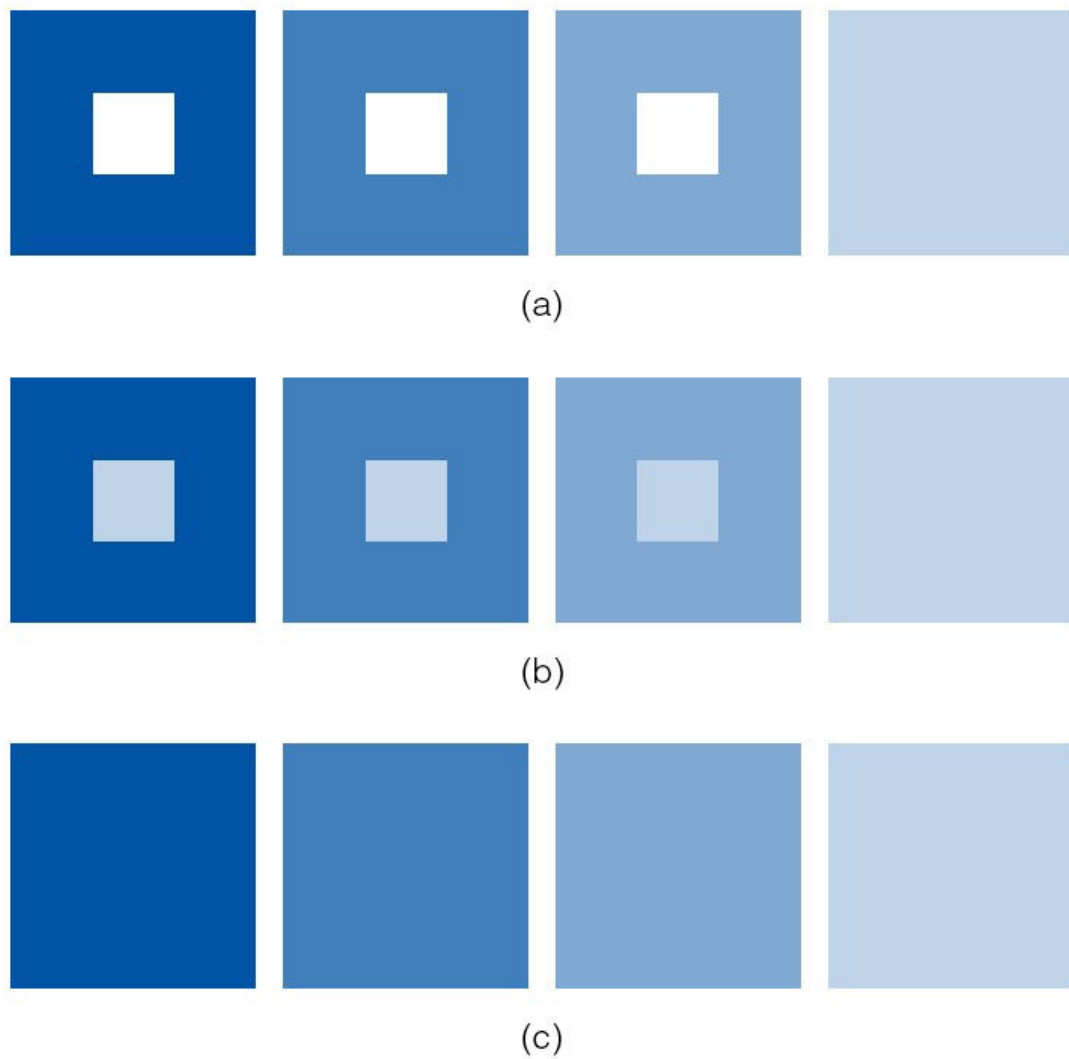


Figure 4.8 – The effect of Poisson blending. (a) A synthetic example of a video with illumination changes. All frames except the last one have the same missing region. (b) Result without Poisson blending, the result contains spatial inconsistency around the border of the occlusion (c) result with Poisson blending, the artifact problem is fixed.

(Bokov and Vatolin 2018) which takes into account the discrepancies between the reconstructed colors and their corresponding colors in the optical-flow-aligned previous frame. Given the colors of the current and previous inpainted frames  $I_t(p)$ ,  $I_{t_0}(p)$ , respectively, the refined Poisson-blended image  $I(p)$  can be obtained by minimizing the discrete energy functional (Bokov and Vatolin 2018):

$$B(I) = \sum_{p \in \Omega_t} \|\nabla I(p) - G_t(p)\|^2 + \sum_{p \in \sigma\Omega_t} w_p^{PB} \|I(p) - I_t(p)\|^2 \\ + \sum_{p \in \sigma\Omega_t} (1 - w_p^{PB}) \|I(p) - I_{t-1}(p + O_t(p))\|^2$$

Here,  $\sigma\Omega_t$  denotes the set of outer-boundary pixels of the missing region  $\Omega_t$  and  $G_t(p)$  is the target gradient field.  $w_p^{PB}$  denotes the adaptive weights with  $w_p^{PB} = (1 + \sigma^{PB} \|\nabla I^{PB}(p) - G_t(p)\|^2)^{-1}$  to weight the reconstruction results from the previous frame  $I_{t_0}$  to formulate boundary conditions. This adaptive weight is used to enforce temporal consistency more in regions where the gradient field of the base Poisson-blending result  $I^{PB}$  deviates the most from the target gradient field  $G_t(p)$ . In this adaptive weight,  $\sigma^{PB}$  is a constant that modulates the level of temporal-consistency enforcement. This approach enables better processing of scenes with global and uniformly changing brightness while enforcing temporal stability around local inconsistencies.

To justify the need for Poisson blending, we synthesize a simple sequence which contains four frames, each frame having its own green color, and our purpose is to understand the effect of illumination changes. All frames except the last frame have the same missing region, which is a rectangle in the center. Figure 4.8 shows the result without/with Poisson blending. As we can see, if we do not use Poisson blending, the result contains artifact along the border because pixels from the last frame are used to reconstruct the occlusion. This artifact disappears when we apply Poisson blending. An example with real video can be viewed on the supplementary website.

## 4.4 Results

### 4.4.1 Implementation details

We implement our simple video completion algorithm in Python. For optical flow computation, we use the FlowNetv2 (Ilg et al. 2017) with Pytorch implementation. The image inpainting is done using the method of (Newson, Almansa, Gousseau, et al. 2017) with the author's default parameters.



### 4.4.2 Evaluation

#### Visual results

Since video completion is an ill-posed problem, it usually has more than one unique optimal solution. Therefore, a clear metric for evaluating the performance of a video completion algorithm almost does not exist. Traditional quality metrics like PSNR and SSIM are a good fit when the damaged region is small either temporally or spatially, but they become decreasingly robust in cases where the hole is large in both spatial and temporal dimensions, as complete adherence to ground truth can no longer be expected. Prior works have not explicitly addressed the problem of video-completion quality assessment for large spatio-temporal holes. In most state-of-the-art methods in this domain, the authors often evaluate the performance of their method based on the satisfaction of the human visual system when playing a video at a sequence. Therefore, in this section, we only provide visual results.

We evaluate our algorithm on a variety of challenging sequences, including sequences which are used in (J.-B. Huang et al. 2016). These sequences are taken from DAVIS-2016 - a recent benchmark dataset for image segmentation. The major challenges in these sequences include dynamic background, motion blur, camera shake, background clutter, and complicated hole shapes. The ground truth pixel-wise annotation is dilated using a  $15 \times 15$  structuring element. We note that we do not aim to segment the shadow in our experiments.

Figure 4.9 shows sample frames from four input image sequences with mask overlay, our completion results, and our completion results with border. With the first two sequences BMX-TREES and TENNIS, we demonstrate that our algorithm can seamlessly fill the missing dynamic background for videos captured with freely moving camera, even when the camera contains large displacements, and the sequences are sampled with low frame rate. The last two sequences RHINO and COWS highlight the advantage of our motion-guided pixel propagation for ensuring the temporal consistency when the occlusion is large, and there is part of the occlusion where pixels in these regions are never seen elsewhere in the temporal stack.

Figure 4.10 shows some representative frames with other challenging sequences CHEETAH, BICYCLE-RACE, and WIRE, which we extracted from VOS dataset (CHEETAH and BICYCLE-RACE), and video completion dataset (WIRE). It can be seen in these cases that when the background is static, our method works well and produce a very plausible sequences in a reasonable time. It is actually able to reconstruct both high and low frequency components.

#### Visual comparison with other state-of-the-art methods

We show now some visual comparisons of the restoration results produced by several different algorithms (including our video inpainting algorithm). We choose the algorithm of the optical flow based algorithm which we explored in Section 2.2.2 of Chapter 2 and the video completion by (J.-B. Huang et al. 2016). We also compare with our patch-based video completion method (Le et al. 2017) described in Chapter 5.

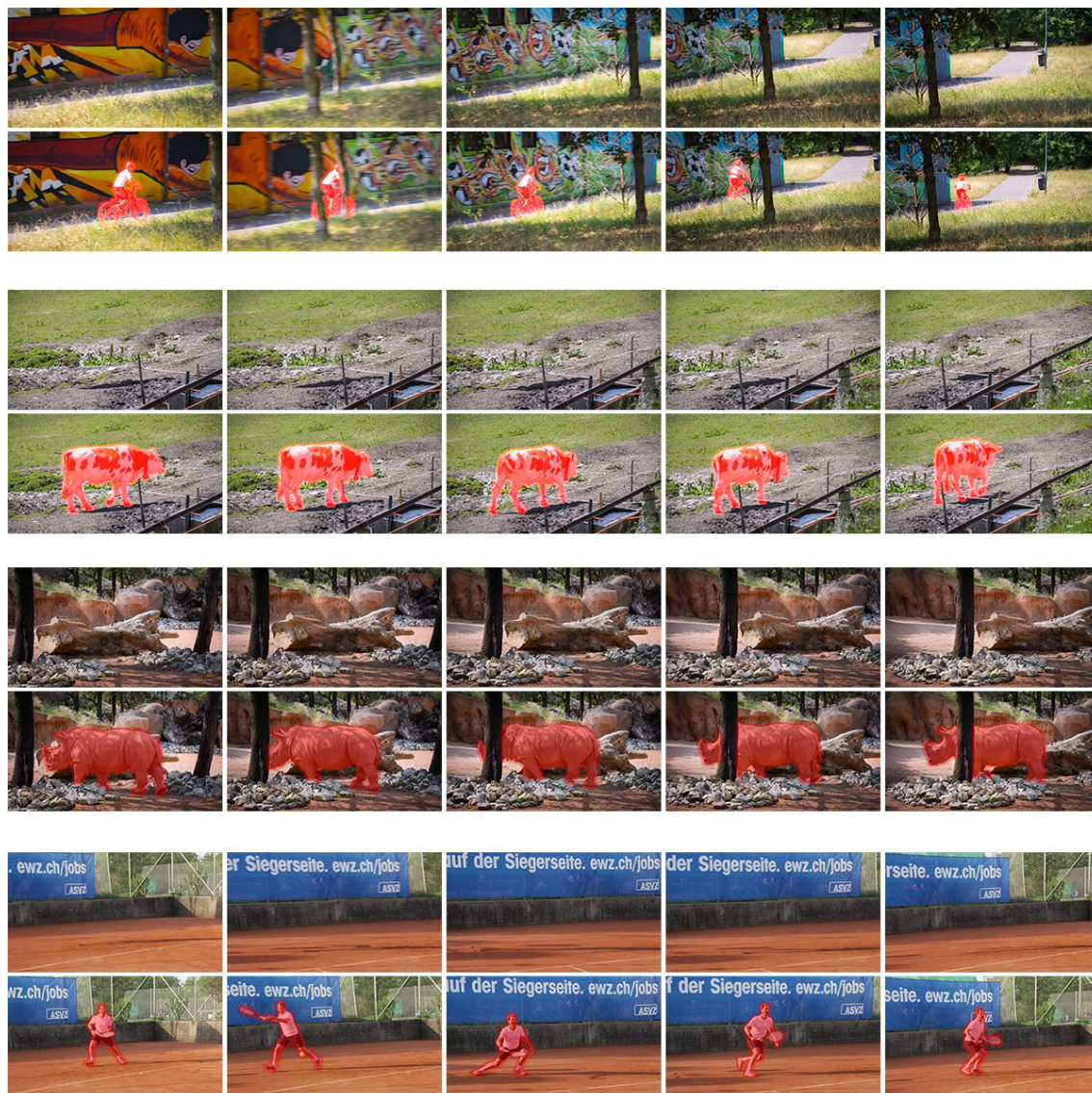


Figure 4.9 – Object removal from several sequences in DAVIS-2016 dataset: BMX-TREES, TENNIS, COWS, and RHINO. For each input sequence (odd row), we show representative frames with mask overlay. We show the completed results in even rows.



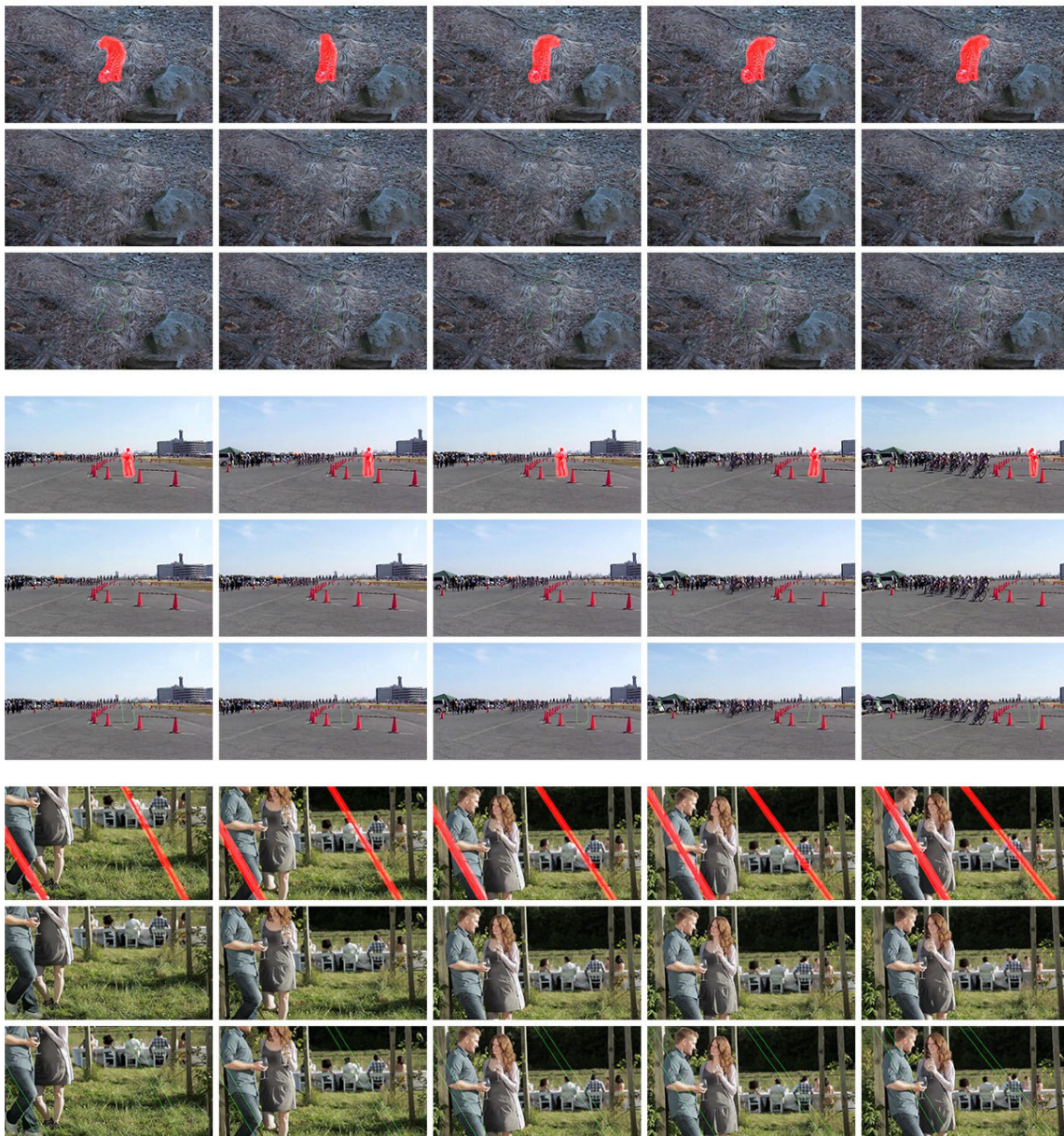


Figure 4.10 – Object removal from other sequences CHEETAH, BICYCLE-RACE, WIRE. In each video, we show the input with mask overlay (first row), inpainted result (second row), inpainted result with the occlusion border.



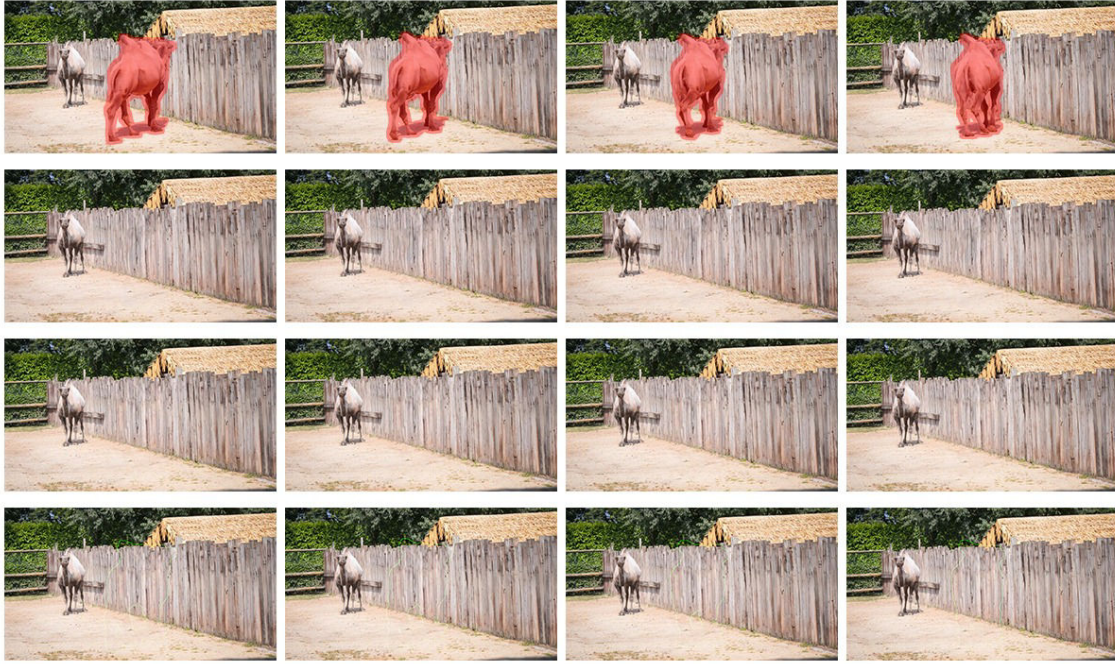


Figure 4.11 – Visual comparison with (J.-B. Huang et al. 2016) on CAMEL sequence. This sequence is challenging due to the fast camera motion and the geometric structure of the scene. From top to bottom: Image + mask, result of (J.-B. Huang et al. 2016), our result, our result with border of the occlusion. In this sequence, our simple greedy optical-flow guided pixel propagation method seamlessly removes the dynamic object and produce results with similar quality as in (J.-B. Huang et al. 2016).

**Comparison with (J.-B. Huang et al. 2016)** We compare qualitatively our method with a recent state-of-the-art motion-guided video completion algorithm. In this experiment, we use two sequences of DAVIS-2016 dataset, namely CAMEL and HORSE-JUMP-HIGH. For a better comparison with (J.-B. Huang et al. 2016), we also segment the shadow in our method. Figure 4.10 shows the representative frames from these two sequences and the completion results. (J.-B. Huang et al. 2016) fills the hole by alternatively minimizing between optical flow and the pixel values under a high complexity optimization framework. Two-dimensional patches are used to reconstruct color values as in (Pritch, Kav-Venaki, and Peleg 2009). To ensure temporal consistency, the authors rely on the estimated flow field which is refined after pixel values are obtained in the missing region. Our method, on the other hand, interpolates the motion once and uses it to propagate the pixel, which results in a simple and faster greedy approach. As we can see from Figure 4.11, both method have the same level of quality. This is to say that, in the object removal task under static background, a simpler greedy approach is enough.

**Comparison with the patch-based method of (Le et al. 2017)** In Figure 4.12 we compare with the patch-based technique of (Le et al. 2017) for inpainting the background and foreground simultaneously in a global framework. We use sequences RHINO (DAVIS-2016) and BIRD

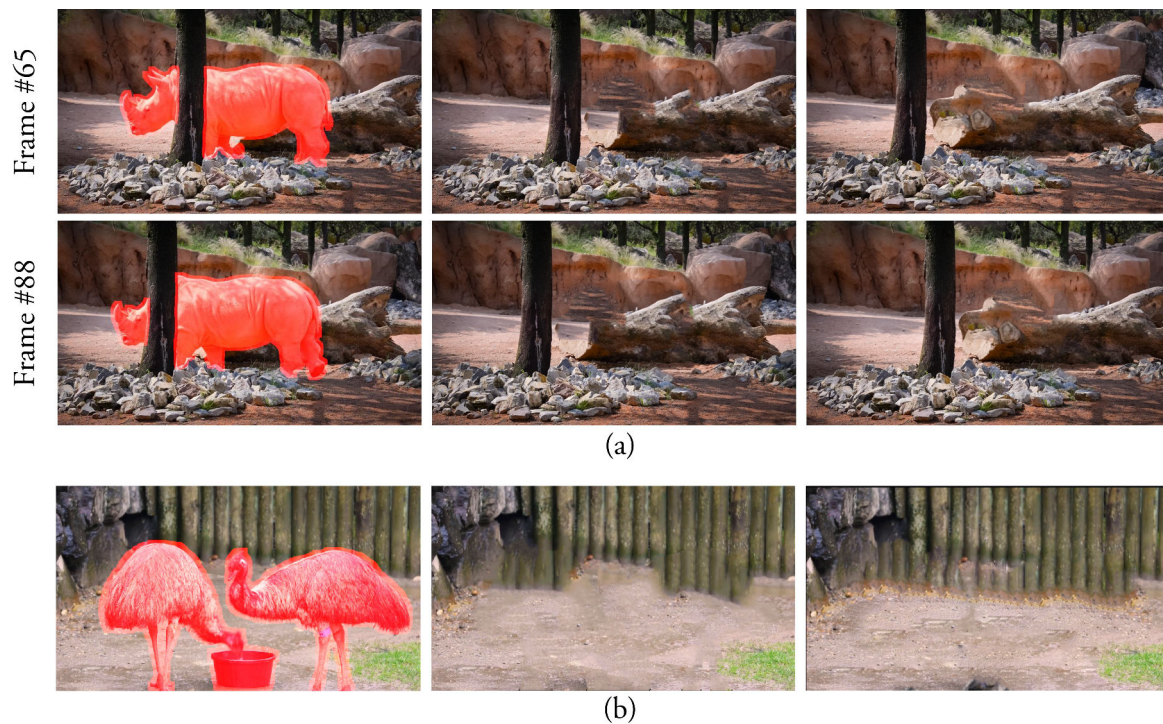


Figure 4.12 – Visual comparison with (Le et al. 2017) on reconstructing static background with sequence RHINO (a) and BIRD (b). From left to right: image + mask, result of the patch-based method of (Le et al. 2017), our result. In RHINO sequence, the motion-based method is more appropriate because it has more temporal coherence, patch-based method, on the other hand, contains oscillation effect because the temporal coherence is weak.





Figure 4.13 – Visual comparison with (Le et al. 2017) on reconstructing dynamic textures. Top: damaged video input, middle: result of (Le et al. 2017), bottom: our result. It is clear that the simple motion-based method cannot recreate dynamic texture.

(extracted from YouTube) sequences, we show the results of the two method on video completion under static background. These two sequences have very large occlusion and there is a large region which is occluded permanently. We note that generally speaking, in RHINO sequence, patch-based methods produce good results when viewed on a frame-by-frame basis. However, the lacks of the temporal coherence is clearly visible when the video is seen at normal speed. The video shows several "oscillation" effect. This is due to the fact that the occlusion stays in the same position in all frames, which makes the temporal consistency harder to obtain. On the other hand, the motion-transfer method propagates the information from one fixed key frame, therefore can ensure the temporal consistency and produce better reconstruction result.

In BIRD sequence, patch-based method contains blurry artifact due to the average of similar patches. This do not happen in motion-guided pixel propagation method.

However, our motion-guided pixel propagation cannot deal with dynamic texture or moving objects. This is mainly because the optical flows under dynamic texture are often unreliable and reconstructing the motion of the moving objects is a difficult problem. Moreover, the repetition in 3D of the spatial-temporal texture cannot be captured by a simple propagation technique. Figure 4.13 demonstrates this situation with sequences "FOUNTAIN-BARCELONA". Patch-based approaches can produce plausible results with these sequences. It is quite clear that the optical flow-based restoration approach is inadequate in this situation. It fails to reconstruct the dynamic textures of the scene and create a very unpleasant artifact due to the unreliable of optical flow estimation. This motivate us to another video completion method based on

patches and which is described in Chapter 5.

**Limitations** Our simple video completion has several limitations. First, when a large area is occluded throughout the entire image sequence, the performance of our algorithm is limited by how well an image completion algorithm can fill the hole, which lead to an image inpainting problem.

Second, our method relies on the optical flow interpolation scheme in the beginning. Therefore, if the scene has complex structures, e.g. multiple planes or non-rigid camera movement the optical flow reconstruction algorithm will produce bad optical flow field, which can lead to poor performance.

The third and most serious limitation is the difficulty to deal with the dynamic textures and moving objects. As mentioned above, our algorithm relies on dense flow fields to guide the completion, it often fails to generate convincing completion of dynamic textures, e.g., waves, due to the unreliable flows. It also fails to reconstruct the moving objects which are occluded. This problem can be solved using patch-based video propagation as described in the next chapter.

## 4.5 Conclusion

In this chapter, we presented a simple video completion algorithm that is capable of handling static background scenario. We formulate the filling process as a pixel propagation approach based on the interpolated motion field. Experiments show that our approach produces results which are comparable to those obtained with a more sophisticated approach.

# 5

## Video inpainting with a complex background

### Contents

---

<b>5.1</b>	<b>Introduction</b>	<b>122</b>
<b>5.2</b>	<b>State-of-the-art in inpainting</b>	<b>123</b>
5.2.1	Image inpainting	124
	Diffusion-based approach	124
	Texture synthesis	126
	Exemplar-based method	127
	Inpainting using deep learning	134
5.2.2	Video inpainting	137
	Object-based approach	137
	Patch-based video inpainting	140
<b>5.3</b>	<b>Proposed method</b>	<b>143</b>
5.3.1	Pre-processing	144
5.3.2	Energy	146
5.3.3	Coarse initialization	150
5.3.4	Source patch searching	151

5.3.5	Pixel reconstruction . . . . .	155
<b>5.4</b>	<b>Result . . . . .</b>	<b>156</b>
5.4.1	Implementation details . . . . .	156
5.4.2	Experimental results . . . . .	157
	Removing dynamically moving objects under static background . . . . .	158
	Dynamic background reconstruction . . . . .	161
	Moving objects reconstruction . . . . .	163
5.4.3	Contribution of each component . . . . .	170
	Video stabilization as a pre-processing step . . . . .	170
	Texture in videos . . . . .	173
	Optical flow-driven initialization versus onion peel initialization . . . . .	173
	The importance of the optical flow term in the distance metric . . . . .	174
<b>5.5</b>	<b>Limitations . . . . .</b>	<b>176</b>
<b>5.6</b>	<b>Conclusion and Future Works . . . . .</b>	<b>177</b>

In this chapter, we propose a motion-consistent video inpainting method that can deal with complex scenes containing dynamic background and moving objects. Our algorithm is not limited to objects removal but extends to a simultaneous reconstruction of background and foreground objects even when these objects are occluded for a long period. We build upon the idea that the spatio-temporal occlusion can be filled with data available on other regions of the video (source region). The intuition is to copy spatio-temporal patches from the source region and to paste them into the occlusion. This idea is formulated in a global optimization framework base on patches as in (Wexler, Shechtman, and Irani 2007; Newson, Almansa, Fradet, et al. 2014). In our method, we extend this approach in several ways to boost the performance both in terms of speed and accuracy. In particular, we make a significant improvement to the method of (Newson, Almansa, Fradet, et al. 2014) by the introduction of optical flow terms: a novel initialization scheme, a modified patch distance, an optical flow-guided patch searching strategy, and a separation map are proposed. We also attain the goal of reducing the computation time by parallelizing the algorithm. By experimenting and comparing the result with other state-of-the-art results, we show that our method has the capability of preserving the spatio-temporal coherency under dynamic background as well as reconstructing moving objects within a long temporal occlusion. It compares favorably with previous works while radically reducing the computation time.

## 5.1 Introduction

In the previous chapter, we learned how to remove objects in a video with a simple static background. We also conclude that in this particular case, the video inpainting problem can

be solved efficiently by a simple motion-guided pixel propagation method. However, videos in the wild often contain dynamic textures and moving objects, which cause difficulty for the propagation-based method. In general, removing dynamic objects from videos containing dynamic background and moving objects is an extremely challenging problem.

Most state-of-the-art methods address this problem by solving a global patch-based optimization problem based on spatio-temporal patches (Wexler, Shechtman, and Irani 2007; Newson, Almansa, Fradet, et al. 2014; Granados, Tompkin, et al. 2012). This appears to be an appropriate approach as it can guarantee global coherence and enforce spatial and temporal consistency. However, these methods have some drawbacks such as large computation times (Granados, Tompkin, et al. 2012) or the inability to deal with motion within large occlusion (Newson, Almansa, Fradet, et al. 2014). Inspired by these methods, we propose a fast video inpainting technique which maintains temporal coherence through both spatio-temporal patches and motion information. Our method has four novel major contributions:

The first and most significant advancement is a systematic use of the optical flow in most steps of the algorithm. This term is incorporated in several stages: it is inserted in the patch distance, controls the patch shape, supports the nearest neighbor search, and serves as a guide in coarse initialization. All these stages enable us to ensure the temporal coherency and the reconstruction of objects with complex motions occluded for long time periods. The second improvement is the integration of a confidence map and a separation map in the pixel reconstruction step to reduce artifacts in the border and between the background/foreground. The third contribution is an accurate pre-processing stabilization step to compensate for the patch distortion caused by a moving camera. The last contribution is a significant reduction in computation cost achieved by parallelizing the algorithm and modifying the patch matching process. This makes our technique more practical and shortens the way to its industrial application.

We evaluate our method on a variety of videos and compare it with some other state-of-the-art approaches. The result can be found in the supplemental materials.

This chapter is organized as follows:

Section 5.2 reviews prior works in video inpainting. Although focusing on patch-based method, we also mention some other relevant approaches to have a holistic overview of the problem. Our core algorithm will be presented in section 5.3. In section 5.4.2, we perform some experiments and compare the result with previous works. The conclusion will be presented at the end of this chapter.

## 5.2 State-of-the-art in inpainting

In this section, we review different inpainting methods in the literature. Since video inpainting is closely related to image inpainting, we first take a brief detour on image inpainting by describing different families of algorithms that exist, then we go into more details in the video



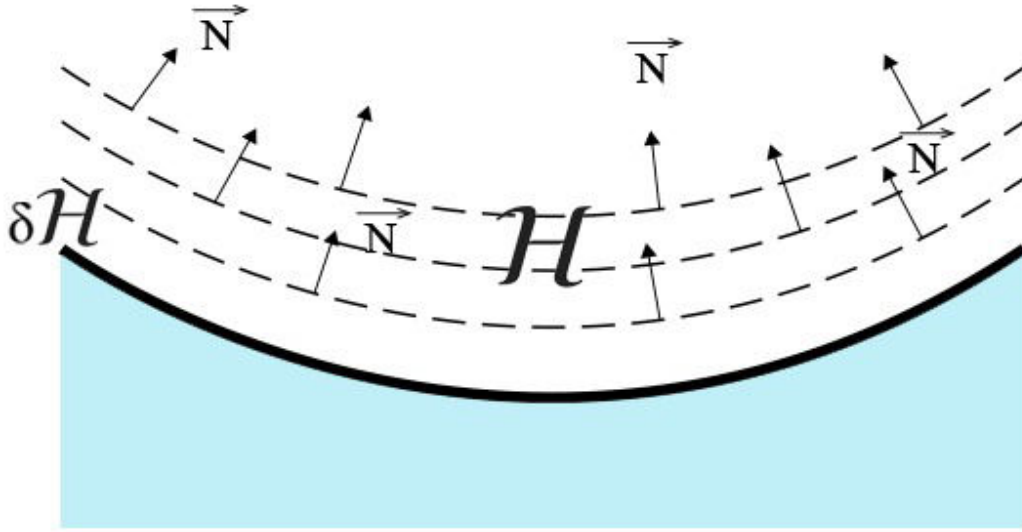


Figure 5.1 – Propagation direction using PDE. Image recreated from (Bertalmio, Sapiro, et al. 2000b).

inpainting methods that have particularly drawn our attention in this thesis. We propose a review of these methods according to their particularities and discuss their advantages and disadvantages respectively. We also note that the emphasis will be put upon patch-based inpainting methods since it relates to our work. Throughout our exploration, we use several definition and notation which we explain in Section ?? of Chapter ???.

### 5.2.1 Image inpainting

We start here by describing common image inpainting approaches before exploring video inpainting methods.

#### Diffusion-based approach

From a mathematical point of view, inpainting can be categorized as an interpolation problem, which is to determine unknown values from the known values of an image. However, images do not behave smoothly like some mathematical functions used in classical interpolation methods. Nevertheless, the idea of diffusion could be used, which aims at propagating local information with smoothness constraints, from outside into the occlusion. In these approaches, two issues need to be carefully considered: (1) how to describe the local image structure; and (2) along which direction the local image structure should be propagated.

In the past, pioneers in diffusion-based image inpainting addressed these issues using geometric structures to guide the propagation process. For example, (Masnou and Morel 1998) propose to interpolate the missing part of the image by connecting the *level lines* that are broken by the inpainting mask. This corresponds to minimizing the following functional:

$$\min_u \int_{\Omega} |\nabla u| (1 + |\text{curv}(u)|^p) dx,$$

with  $u|_{\mathcal{D}} = u^*|_{\mathcal{D}}$ , where  $\text{curv } u = \text{div } \frac{\nabla u}{|\nabla u|}$ , and  $u^*$  is the known content of the image in the source region  $\mathcal{D}$ . This formula specifies that the length of the level lines and the total angle variation along the lines should be small when the boundary conditions of the solution are satisfied.

In (Bertalmio, Sapiro, et al. 2000b), the authors propose to solve a partial differential equation (PDE) for extending *isophotes* (similar with *level lines* in (Masnou 2002)) within the inpainting mask. This is done by using an iterative algorithm which progressively propagates image information along the isophotes into the occlusion, this process alternates with the resolution of an anisotropic diffusion equation.

In a subsequent work by (Ballester et al. 2001), the authors adopt the ideas of (Bertalmio, Sapiro, et al. 2000b) about the simultaneous gray level and gradient continuation to define a formal variational approach to the inpainting problem. Their variational approach is solved via steepest descent, which leads to a set of two coupled second-order PDEs, one for the gray levels and one for the gradient orientation.

Another PDE based approach is a family of works by Chan *et.al* (Jianhong Shen and T. F. Chan 2002; T. F. Chan and Jianhong Shen 2005; T. F. Chan, Jianhong Shen, and H.-M. Zhou 2006). The motivation is to create a scheme which is motivated by existing denoising/segmentation methods. Their approach is based on the most famous model in image processing, the *Total Variation* (TV) model. In particular, they minimize the *Total Variation* in the masked area of the image to be reconstructed, with boundary conditions around the mask.

(Tschumperle and Deriche 2005) proposed two different PDE-based frameworks able to design specific regularization processes from a given underlying local smoothing geometry. These methods have two main interests: on the one hand, they unify many previously proposed equations into generic diffusion PDEs and provide a local geometric interpretation of the corresponding regularization. On the other hand, they clearly separate the design of the smoothing geometry from the smoothing process itself.

Other diffusion-based methods have been proposed in the literature for image inpainting. For example, in (Levin, Zomet, and Weiss 2003; S. Roth and Black 2005) the minimization of statistical characteristics learned from natural images are applied to the task of image inpainting. A transport method based on coherence flow directions is used in (Bornemann and März 2007). These techniques use a priori on the regularity and propagate the local structures from the outside to the inside of the inpainting mask.

In conclusion, diffusion-based inpainting algorithms have achieved quite good results in terms of overall geometric structure coherence. They are suitable for filling non-textured or small missing regions. However, they tend to strongly smooth the textured regions or to produce artifacts in large missing region. Therefore, they are not suitable for the reconstruction of complex textures.

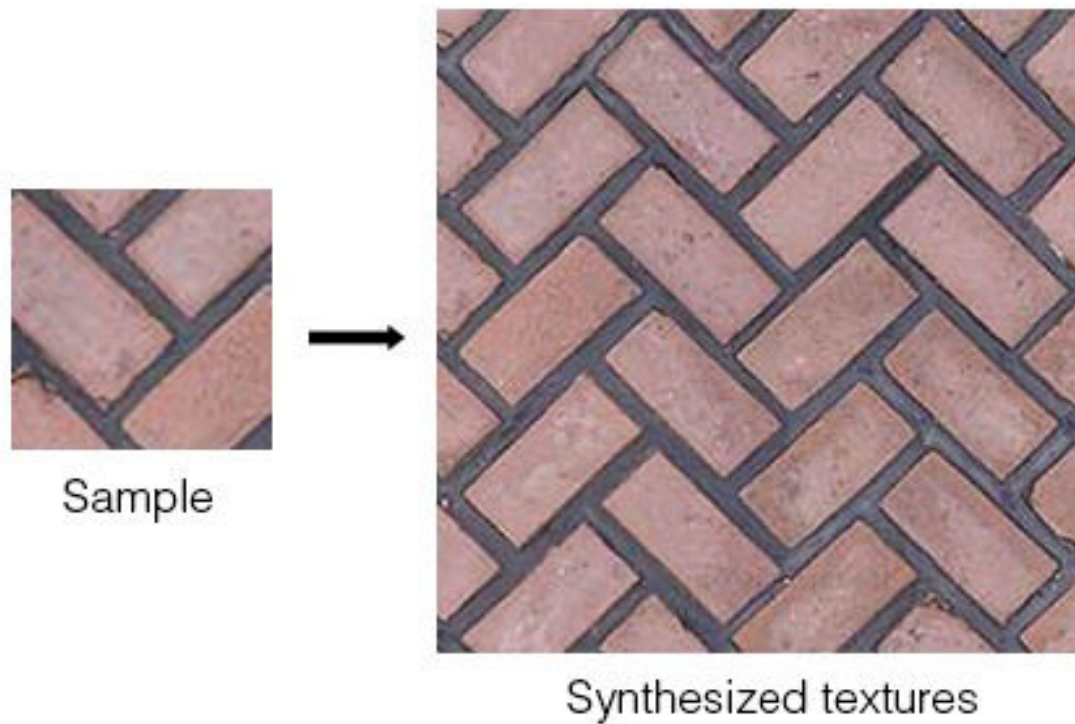


Figure 5.2 – A large texture is produced from the input sample

### Texture synthesis

A popular choice to reconstruct textured regions is inspired by texture synthesis approaches. The purpose of texture synthesis is to create a new, possibly large, image from an initial sample in a way which preserves the visual appearance of the sample. An example of texture synthesis is illustrated in Figure 5.2. In early works on texture synthesis, textures are modeled stochastically e.g. using Random Markov Fields (Cross and A. K. Jain 1983). That is to say that each pixel of texture may be considered as a random sample of a certain distribution to estimate. While the results for stochastic textures were excellent, this class of algorithms was unable to deal well with structured textures.

In (Efros and Leung 1999), the authors try to address this shortcoming by proposing a different and straightforward way of synthesizing textures locally using patches. They were the first to use the notion of patches for texture synthesis. The core of the method is the concept of self-similarity. They synthesize a texture by estimating the value of a pixel  $p$  as a function of the patch  $\mathcal{N}_p$  centered in it. This value is found by looking for the  $k$ -nearest neighbors of  $\mathcal{N}_p$  according to a metric  $d = d_{SSD} * G$  where  $d_{SSD}$  is the squared difference vector and  $G$  is a 2D Gaussian kernel. Shortly after, (Wei and Levoy 2000) proposed a faster approach that uses multi-resolution and a tree-structured Vector Quantization structure to speed up the searching process.

Many ideas introduced for texture synthesis have largely influenced recent advances in image inpainting (Yamauchi, Seidel, et al. 2003). Among them, the idea of patches is the most

important concept, Which led to a very successful branch of image inpainting, sometimes called exemplar-based inpainting. In the following section, we propose to classify them and to make a review of these methods.

### Exemplar-based method

Applying texture synthesis techniques to solve the inpainting problem in images, as proposed in (Yamauchi, Seidel, et al. 2003), performs well in the case of images with regular textures. However, texture synthesis-based inpainting approach is not suitable for images with a lot of structures. In this case, combining textures and structures data in the image helps to provide better results. This idea is adopted in exemplar-based inpainting techniques. Under this approach, image patches are used both to analyze their content and find the best way to reconstruct the missing parts. Exemplar-based methods can be categorized into three categories:

- *Greedy approaches* (Bornard et al. 2002; Drori, Cohen-Or, and Yeshurun 2003; Criminisi, Pérez, and Toyama 2004; Perez, Gangnet, and Blake 2004; Le Meur, Gautier, and Guillemot 2011; Z. Xu and J. Sun 2010; Martínez-Noriega, Roumy, and Blanchard 2012) complete the missing area in one pass by copying pieces or complete patches using a greedy algorithm
- *Hybrid approaches* (Bertalmio, Vese, et al. 2003; Starck, Elad, and Donoho 2005; Jia and C.-K. Tang 2004; Cao et al. 2011) incorporate elements of diffusion-based methods to complete the structures in the first place. The textures are completed later using a pattern based approach
- *Global approaches* (Wexler, Shechtman, and Irani 2007; Komodakis and Tziritas 2007; Pritch, Kav-Venaki, and Peleg 2009; Mansfield et al. 2011; Arias, Caselles, and Facciolo 2012; He and J. Sun 2012b; Newson, Almansa, Fradet, et al. 2014; Newson, Almansa, Gousseau, et al. 2017) solve the inpainting problem by minimizing a global energy, and often require multiple iterations for convergence. These methods reconstruct all pixels in the missing part at the same time.

Next, we explain these approaches in more details.

**Greedy approaches** The principle of greedy algorithms is to fill the occlusion  $\mathcal{H}$  pixel by pixel until  $\mathcal{H} = \emptyset$ . Each pixel of  $\mathcal{H}$  is filled only once during the process. This type of algorithm is often made of 4 main steps:

1. Select a pixel  $p$  in the inner border of the occlusion  $p \in \delta\mathcal{H}$
2. Search in the source region  $\mathcal{D}$  a patch  $\psi_{\hat{p}}$  that is similar to  $\psi_p$
3. Copy valid data from  $\psi_{\hat{p}}$ .

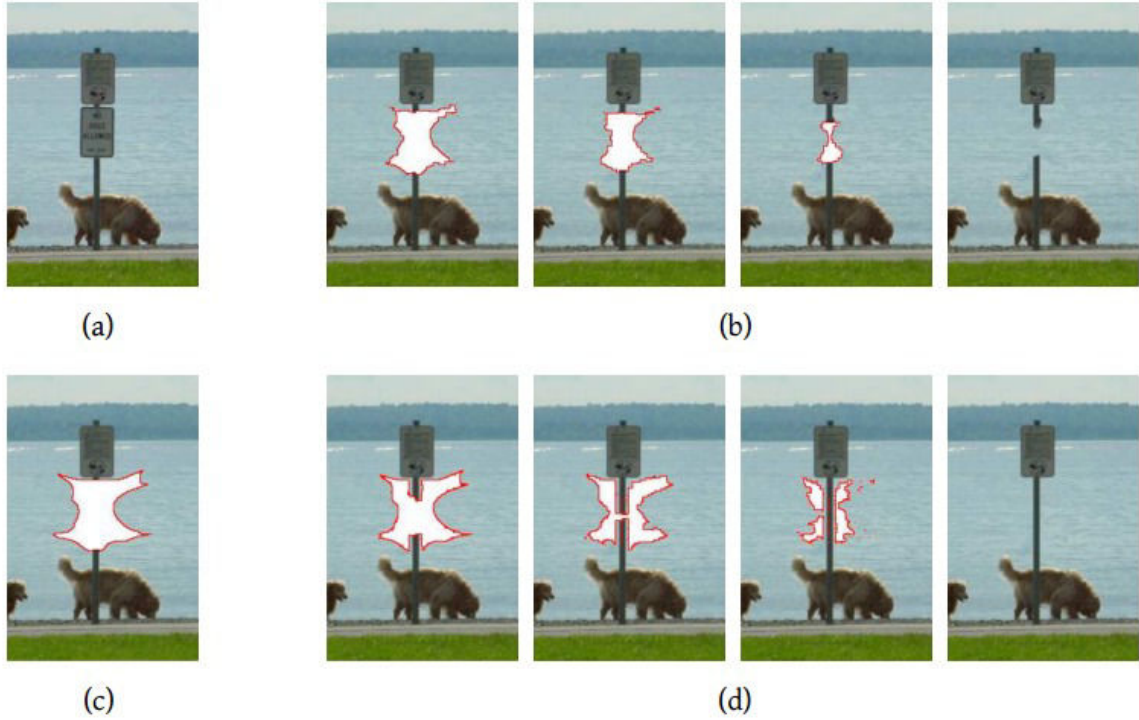


Figure 5.3 – Results of filling the damaged region following onion peel order and filling using the data term  $D(p)$  (image from [(Criminisi, Pérez, and Toyama 2004)]). (a) original image, (b) filling following onion peel order, (c) image + damaged region, (d) filling using the term  $D(p)$ .

4. If  $\mathcal{H} \neq \emptyset$  return to step 1.

The first two steps are the heart of the algorithm and have critical importance: different choices of priority for the pixel selection will lead to completely different results. This effect is due to the greedy nature of this algorithm that uses reconstructed patches at a previous iteration to find a good patch for the current iteration. Naive implementations of this type of algorithm often lead to unsatisfactory results, as demonstrated in Figure 5.3 (b).

A typical example of inpainting with a greedy approach is the well-known image inpainting method of (Criminisi, Pérez, and Toyama 2004). We will explain this method in details:

*The greedy algorithm of (Criminisi, Pérez, and Toyama 2004)*

Figure 5.4 provides an overview of the algorithm of (Criminisi, Pérez, and Toyama 2004). As in a typical greedy approach, it consists of 4 steps as mentioned above. We focus here in more details on the two steps which are essential parts of this method: (1) *reconstruction priorities* and (2) *searching for the best patch candidate*.

- *Reconstruction priorities:* In (Criminisi, Pérez, and Toyama 2004), instead of filling the hole layer-by-layer from the outside to the inside as in (Bornard et al. 2002) (Figure 5.3 (b)), the authors have proposed a filling order based on known local structures around the hole. More specifically, a priority term  $P(p)$  is calculated for all pixels in the occlusion,  $p \in \mathcal{H}$  as:

$$P(p) = C(p) \cdot D(p)$$



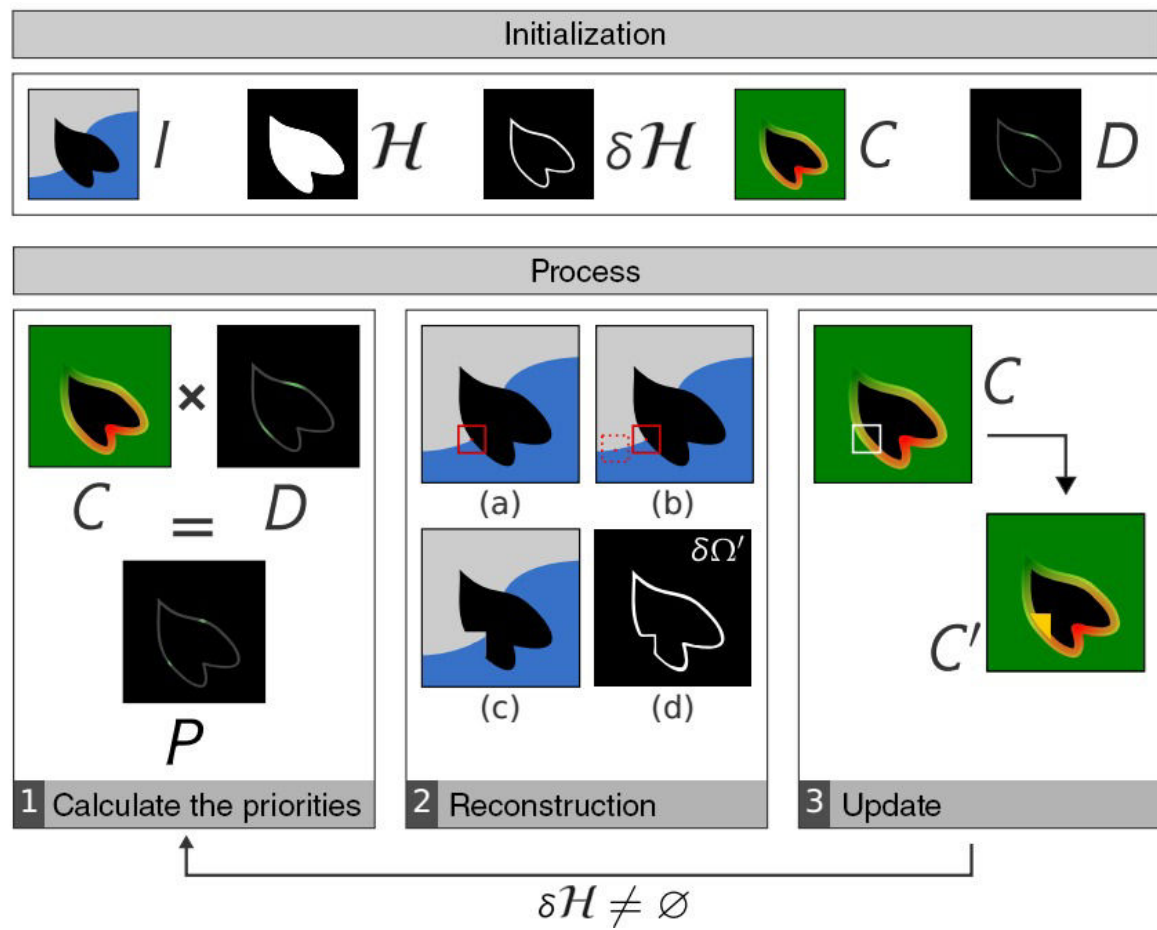


Figure 5.4 – Overview of the greedy image inpainting method of (Criminisi, Pérez, and Toyama 2004). Image taken from (Daisy 2015)

where  $C(p)$  is a confidence measure, and  $D(p)$  is defined as a data term that takes into account the presence of structures in  $\psi_p$ . The term of confidence  $C(p)$  is a measure of the amount of information known in patch  $\psi_p$ . It is defined in (Criminisi, Pérez, and Toyama 2004) as:

$$C(p) = \frac{\sum_{q \in (\mathcal{N}_p \cap \overline{\mathcal{H}})} C(q)}{|\psi_p|}$$

where  $|\cdot|$  is the area of patch  $\psi_p$ . This term has value elevated close to the edge of the mask and decreases to the center of  $\mathcal{H}$ . This favors the rebuilding of the first pixels which is the most effective. This is especially useful where the structured information has to be reconstructed first.

The data term  $D(p)$  accounts for the presence of local structures in the image at the edges of  $\mathcal{H}$ . It plays a critical role in calculating the priority and promotes the continuation of the structures entering  $\mathcal{H}$ . Figure 5.3 (bottom) shows the process of inpainting at different iterations using (Criminisi, Pérez, and Toyama 2004) and shows that by using an adapted reconstruction priority term, it is possible to reconstruct the linear structures correctly. The term data is defined in (Criminisi, Pérez, and Toyama 2004) as:

$$D_p = \frac{|\overrightarrow{\nabla I_p}^\perp \cdot n_p|}{\alpha}$$

where  $n_p$  is the normal to  $\mathcal{H}$  in  $p$ ,  $\overrightarrow{\nabla I_p}^\perp$  is the direction of the level lines (Masnou 2002) defined as:

$$\overrightarrow{\nabla I_p}^\perp = \left\{ \overrightarrow{\nabla I_q}^\perp \mid \arg \max_{q \in (\overline{\mathcal{H}} \cap \mathcal{N}_p)} \|\overrightarrow{\nabla I_q}\| \right\}$$

and  $\alpha$  is a normalization term which in fact can be ignored, being the same for every  $p$ .

- *Searching for a best patch and reconstruction:* At each iteration of the algorithm, the patch  $\psi_{\hat{p}}$  most similar with  $\psi_p$  is searched for in the image. Often, the search area is limited to a window  $\mathcal{W}$  of size  $w_s \times w_s$  centered in  $p$ . This makes it possible to reduce the search space and thus the execution time compared to an exhaustive search throughout the image.

*Some improvements of the greedy method of (Criminisi, Pérez, and Toyama 2004):*

As the process of inpainting is greedy, a minor shift in priority can change the nature of the results obtained. As a result, several studies have been conducted in the literature to improve or make more robust the effect of the priority term  $P(p)$

- The authors of (Le Meur, Gautier, and Guillemot 2011) proposed a heuristic approach

to improve the priority term thanks to data-based tensors:

$$D(p) = \alpha + (1 - \alpha) \exp\left(\frac{\eta}{(\lambda_1 - \lambda_2)^2}\right)$$

where  $\lambda_1$  and  $\lambda_2$  are the eigenvalues of the structure tensor evaluated at  $p$ , and  $h, \alpha$  are constant hyperparameters set at  $h = 8$  and  $\alpha = 0.01$ . The idea is to use the possible anisotropy of the structure tensor at  $p$  to determine if there is a significant color variation locally. The use of such tensors makes it possible to obtain a more robust estimate of the overall geometry than the gradient on color images.

- In (Z. Xu and J. Sun 2010), the authors proposed a parsimonious data term, which measures for each pixel  $p \in \delta\mathcal{H}$ , the overall similarity of a patch centered in  $p$  with its neighboring patches. This was done to begin the reconstruction where we have, in theory, more chance to find a patch consistent with the neighborhood of the desired patch.
- (Guillemot et al. 2013) proposed changing the term of priority  $P(p)$  by introducing a term  $E(p)$  - a specific contour-based term. This term  $E(p)$  favors the priority reconstruction of patches containing pixels belonging to outlines.

**Hybrid approach** Diffusion-based methods have proved that they can reconstruct global structures correctly, but do not work when dealing with complex textures (Masnou 2002; Ballester et al. 2001; Tschumperle and Deriche 2005). The greedy methods can correctly reconstruct the textures, but sometimes struggle to reconstruct a plausible global geometry. This is particularly the case when the missing part of a structure is not present anywhere else in the rest of the image, e.g., when the mask obscures a curved line. From Figure 5.5, we can observe clearly these properties. Diffusion-based methods permit to obtain smooth contours and structures but are not able to reconstruct the textures. When looking at the patch-based approach in the fourth column, we observe the opposite, textures are pretty well reconstructed but edges are not continuous.

Since natural images contain both structures and textures, hybrid approaches combining diffusion-based and patch-based methods have naturally been explored in the literature. For example, in (Bertalmio, Vese, et al. 2003; Starck, Elad, and Donoho 2005), the authors break the image to be filled into a structure part and a texture part and fill them independently with a diffusion-based approach for the structured regions, and texture synthesis for the textured areas. These images are then recombined to produce the final result. With a different approach, (Jia and C.-K. Tang 2004; Cao et al. 2011) first tries to reconstruct the strong contours that enter the mask. The result is then used to define the search areas and then fill in the missing pixels with an exemplar-based approach. More specifically, while the method proposed in (Jia and C.-K. Tang 2004) uses a tensor vote to infer the missing part of the curves, level lines

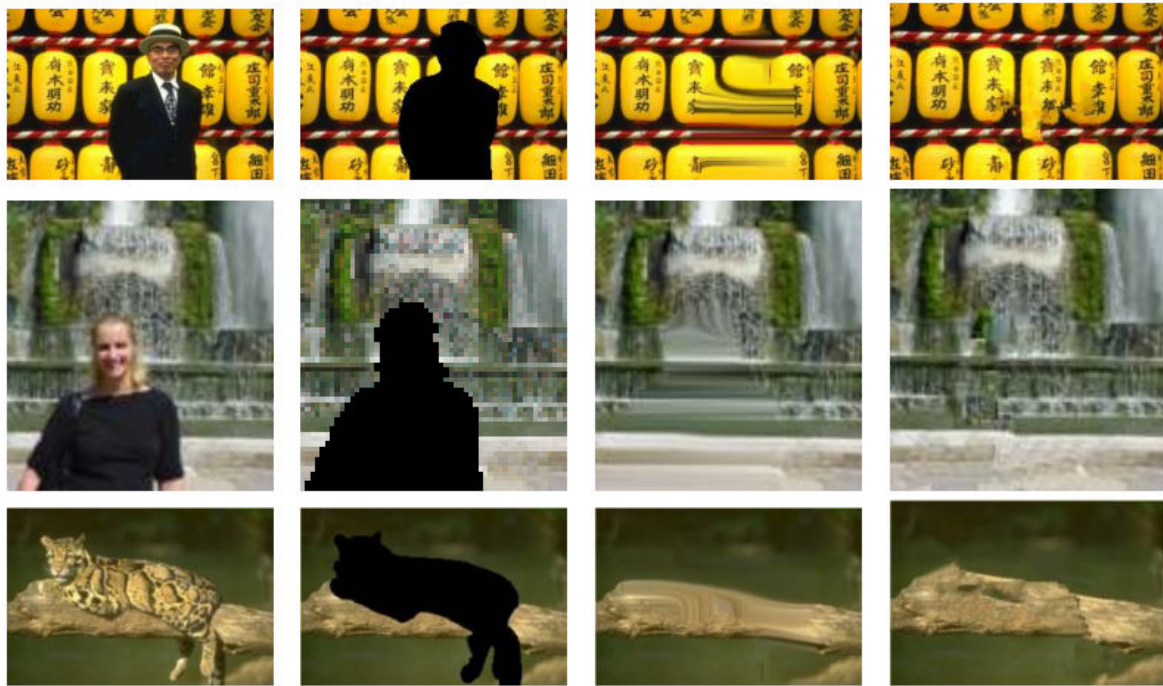


Figure 5.5 – The respective advantages of patch-based methods and diffusion-based methods. Patch-based methods can reconstruct textures while diffusion-based methods fail. On the other hand, diffusion-based methods can preserve strong geometric structures. From left to right: Original images, damaged regions, results produced by a diffusion-based method (Tschumperle and Deriche 2005), results produced by a patch-based method (Criminisi, Pérez, and Toyama 2004)

with Euler spirals are used in (Cao et al. 2011) to complete this type of curves. Once the sketches have been obtained, the masked pixels are filled with a texture synthesis method that is constrained by previously completed curves.

In conclusion, the hybrid approaches combine the advantages of diffusion-based and exemplar-based methods, so that they can perform better. However, they are difficult to apply in practice since the separation of structures and textures from an image is a difficult problem. Besides, the high execution time of these methods reduces their potential for practical use.

**Energy-based methods** Greedy approaches incrementally fill-in the occlusion pixel by pixel, the inpainting order being established by calculating a priority term. Using this term, each time, the optimization is performed locally. For this reason, the result of these methods does not ensure a global image coherence. To maximize the coherence over all the pixels in the occlusion, many works have tried to synthesize the missing part pixel by pixel using multiple patches at the same time. These methods realize the reconstruction of the missing part via the minimization of a global energy relative to the coherence of reconstruction.

The first energy modeling for patch-based inpainting is proposed in (Demanet, Song, and T. Chan 2003) where the authors introduce an energy term which is optimized over the shift map  $\phi$  and the image value  $u$ . To ensure overall consistency, they propose to minimize the following consistency measure, which measures for each missing pixel, the similarity between



Figure 5.6 – By allowing different transformations for the patches, the method of (Mansfield et al. 2011) can reconstruct complex structure such as a circle. From left to right: Image with mask, result produced by (Wexler, Shechtman, and Irani 2007), result produced by (Mansfield et al. 2011). Figure taken from (Mansfield et al. 2011).

the patches that cover it and the patches in the known part of the image:

$$E(\phi|u) = \sum_{p \in \mathcal{H}} d^2 \left( \psi_p^u, W_{p+\phi(p)}^u \right)$$

where  $d^2(\cdot, \cdot)$  is a measure of distance. This equation is highly non-linear and non-convex and there is no explicit solution or algorithm to find a global minimum. Therefore, (Demanet, Song, and T. Chan 2003) proposed to approximate the solution by repeating until convergence the basic texture synthesis approach with an "onion peel" strategy. The algorithm updates, for each pixel  $p$ ,  $u(p)$  and  $\phi(p)$  conjointly.

A similar energy was also proposed in (Wexler, Shechtman, and Irani 2007) for inpainting. To minimize the energy, the algorithms repeat two steps until convergence: (1) a nearest neighbor search updating  $\phi$  for all pixels; (2) a reconstruction step updating  $u$ . The reconstruction is done through patch fusion by calculating a weighted average of pixels from several overlapping patches. They reconstruct the image with a multi-resolution iterative process. A reconstructed image at a given iteration (respectively resolution) serves as initialization for the next iteration (respectively resolution). Used together with the PatchMatch algorithm (Barnes, Shechtman, et al. 2009), it is a main component of the famous "Content-aware Fill" in Photoshop.

(Mansfield et al. 2011) improves the synthesis of (Wexler, Shechtman, and Irani 2007) by allowing different transformations for the patches: translation, rotation, scaling, or change of brightness. These transforms are necessary in cases when shifting is not sufficient, e.g., completing circles or reflection-symmetric objects, see Figure 5.6). However, these transformations further increase the dimensionality of the search space which makes the nearest neighbor searching algorithm slower and potentially less efficient.

With the principle of (Wexler, Shechtman, and Irani 2007), (Kawai, Sato, and Yokoya 2009) also takes into account the change of brightness in the search space, thus increasing the number of candidate patches. On the other hand, the spatial locality of the textured patterns is considered an implicit constraint.



With a different approach, (Komodakis and Tziritas 2007) sees the inpainting problem as a Markov Random Field Labeling problem. An objective function is defined and optimized through a Priority Belief Propagation (or Priority-BP) that improves the standard BP to handle a large number of markers. Inpainting with this method seems natural but the calculation time (up to 2 min for a  $256 \times 127$  image) makes this method difficult to use in practice.

The method from (Pritch, Kav-Venaki, and Peleg 2009) considers inpainting as a graphing problem. Here, the weights associated with the edges of the graph are the relative offset of each pixel of the output image, and the offset map is calculated via the minimization of an energy composed of two terms: a data term indicating constraints on pixels, and a regularity term that minimizes discontinuities between objects caused by discontinuities in the shift map. In this method, the calculation of the offset (pre-inpainting) takes up to 30 seconds for  $1600 \times 1600$  images.

(He and J. Sun 2012b) have the idea of using the offset map (or shift map) in a way similar to (Pritch, Kav-Venaki, and Peleg 2009). However, they propose to calculate the most frequent offsets between a patch and its nearest neighbor. The input image is shifted several times according to the main offsets, the images produced are then stacked, and finally, the best cuts between these images are calculated using a graph-cut algorithm. A Poisson equation merging technique is then used to minimize discontinuities between reconstructed image pieces. This method provides a new way of analyzing the image for inpainting. Nevertheless, the number of shifts extracted initially, the parameter  $K$  in (He and J. Sun 2012b), depends very much on the topology of the image and can lead to texture repetition or local inconsistencies.

(Arias, Caselles, and Facciolo 2012) propose a general variational framework managing both a local and a non-local aspect of the inpainting problem. Several schemes are derived via the selection of an appropriate patch similarity criterion: mean/ median/ Poisson of the nonlocal gradient of patches, corresponding to criteria of similarity based on standards  $L_2$  and  $L_1$  distances between patches or their gradients. The objective function adds to the term of similarity a term measuring the entropy of similarities seen as probabilities. A coordinate descent algorithm is used to minimize the objective function, alternately between the weights of similarity and the updates of the images.

### Inpainting using deep learning

Traditional image inpainting approaches such as diffusion-based techniques that propagate local structures into the unknown parts, or exemplar-based method that construct the missing part while maintaining the consistency with the neighborhood pixels, both fail when the content of the missing region does not appear elsewhere in the image. Figure 5.7 illustrate this situation where the eyes of the girl are occluded. With a traditional method such as Content-Aware-Fill, it is impossible to reconstruct the eyes. In this example, hallucinations by machine are needed.

To overcome this problem, deep learning-based methods have also been proposed for

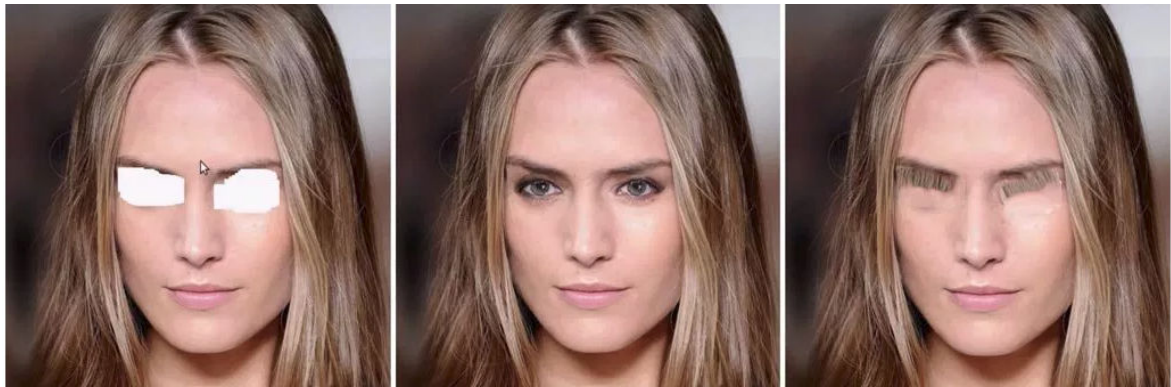


Figure 5.7 – The advantage of image inpainting using deep learning. Deep learning methods can reconstruct information which does not appear in the source region, while traditional patch-based methods cannot do that. From left to right: Image + mask, results produced by deep learning-based method ([G. Liu et al. 2018](#)), results produced by the patch-based method (Content-Aware-Fill).



Figure 5.8 – Result of Context Encoder ([Pathak et al. 2016](#)) with different losses. (a) input context, (b) result with  $L_2$  loss, result with  $L_2$  + adversarial loss. Image from ([Pathak et al. 2016](#)).

inpainting. Authors from (Pathak et al. 2016) were pioneers in deep learning-based image inpainting. They propose a pipeline called Context Encoders. An encoder-decoder architecture is used to encode the context information and predict the missing content with respect to a learned loss function. The encoder is a Convolutional Neural Network (CNN) trained on a large dataset to learn features extracted from images in which random regions are automatically removed. From these feature maps, in the decoder side, missing pixels are generated using another CNN. As the traditional  $L_2$  loss tends to blur the result because it averages the multiple modes in predictions (Figure 5.8(b)), the authors propose to combine it with an adversarial loss, which is based on a Generative Adversarial Network (GAN) (Goodfellow et al. 2014). GAN jointly learns a generative model of the data distribution together with a discriminative model which provides loss gradients to the generative models. The adversarial loss has the effect of picking a particular mode from the data distribution, resulting in less blurry results (Figure 5.8(c)). This method produces impressive reconstruction results when the structure of holes is fixed but is less effective for arbitrarily structured regions. Moreover, the results from this method often lack texture details, which leads to blurry effects at the border of the missing area.

Authors of (Iizuka, Simo-Serra, and Ishikawa 2017) further improve the method of (Pathak et al. 2016) by using two discriminators as adversarial losses. The global discriminator is used to enforce the global appearance (whole image) while the local discriminator focus on local appearance (a small area centered at the generated region) to enforce the local consistency. Also, they use dilated convolutions to replace channel-wise fully connected layer adopted in Context Encoders, which the purpose of increasing the receptive fields of output neurons and adapt to any arbitrary input size.

(C. Yang et al. 2017) proposed another pipeline of combination between Context Encoder (Pathak et al. 2016) and Patch-based synthesis. They use the output of the Context Encoder as an initial solution to solve a discrete model based on joint optimization of image content and texture content between generated patches and the matched patches in known regions. This helps them preserves contextual structures and also produces high-frequency details by matching and adapting patches with the most similar mid-layer feature correlations of a deep classification network. The first part of their method is similar to Context Encoder, and is made of two networks. The first network reconstructs an image  $u$  from the input image  $u_0$  while the second is the pre-trained network which is used to compute, from an image  $u$ , features  $f$  at a predetermined layer. While the results are still blurry, it shows promising visual results and it is a first step for combining patch-based and geometric models together with convolutional neural networks. The major drawback of this method is that is very slow due to the optimization process. Another limitation of these methods is that the shaped holes are often assumed to be center in the image. These limitations may lead to overfitting to the rectangular holes, and ultimately limit the utility of these models in applications.

To overcome these limitations, Nvidia Research uses partial convolutions with a step of

automatic mask update (G. Liu et al. 2018), which removes any masking where the partial convolution was able to operate on an unmasked value. Besides, they also demonstrate the efficacy of training image inpainting models on irregularly shaped holes. As a result, their model achieves state-of-the-art results in image inpainting and can robustly handle holes of any shape, size location, or distance from the image borders.

To compare the performance of deep learning-based approach versus traditional patch-based approach, we test 3 methods: Context Encoder (Pathak et al. 2016), the hybrid method from (C. Yang et al. 2017) and a patch-based method, respectively. Visual comparison is shown in Figure 5.9 From this figure, we can see that deep learning methods achieve good performance in some cases, but traditional patch-based algorithms maintain a strong position and outperform the deep-learning alternatives on numerous tests.

### 5.2.2 Video inpainting

While the image inpainting problem has been extensively studied for several decades, the literature devoted to video inpainting is much smaller. This can be explained by the fact that when dealing with inpainting problem in video, major challenges arise: (1) the high sensitivity of our visual system to temporal inconsistencies, which requires not only the spatial coherence but also the temporal continuity among video frames; (2) the difficulty of dealing with complex motions; (3) the exponentially increased computational complexity.

In the previous section, we already took a tour of the image inpainting literature. Some key concepts were introduced and these are also very helpful for the video inpainting problem. In general, video inpainting methods are the extensions of image completion algorithms with some additional constraints that aim to provide both spatial and temporal consistency. In the following subsections, we describe the widely cited approaches in the literature, broadly classified into object-based and patch-based approaches. A brief survey of this topic can be found in (Ilan and Shamir 2015).

#### Object-based approach

The object-based approach is usually based on video segmentation as a pre-processing step to split the video into moving foreground objects and background. Each part is then reconstructed independently using separate algorithms, and the results are merged at the end.

A pioneer work under this approach was that of (Jia, Tai, et al. 2006), which inpaint the missing holes of the occluded objects by aligning the object's visible trajectory. (Figure 5.10. In this algorithm, the background and moving foreground are segmented. For the moving foreground, they sample cyclic motion objects and identify their periodicity and align them with the corresponding part in the hole which allows the moving objects to be best fitted into the occlusion. The actual alignment is done by warping the moving objects using a homographic transformation. For the background, they adopt a layered mosaic approach



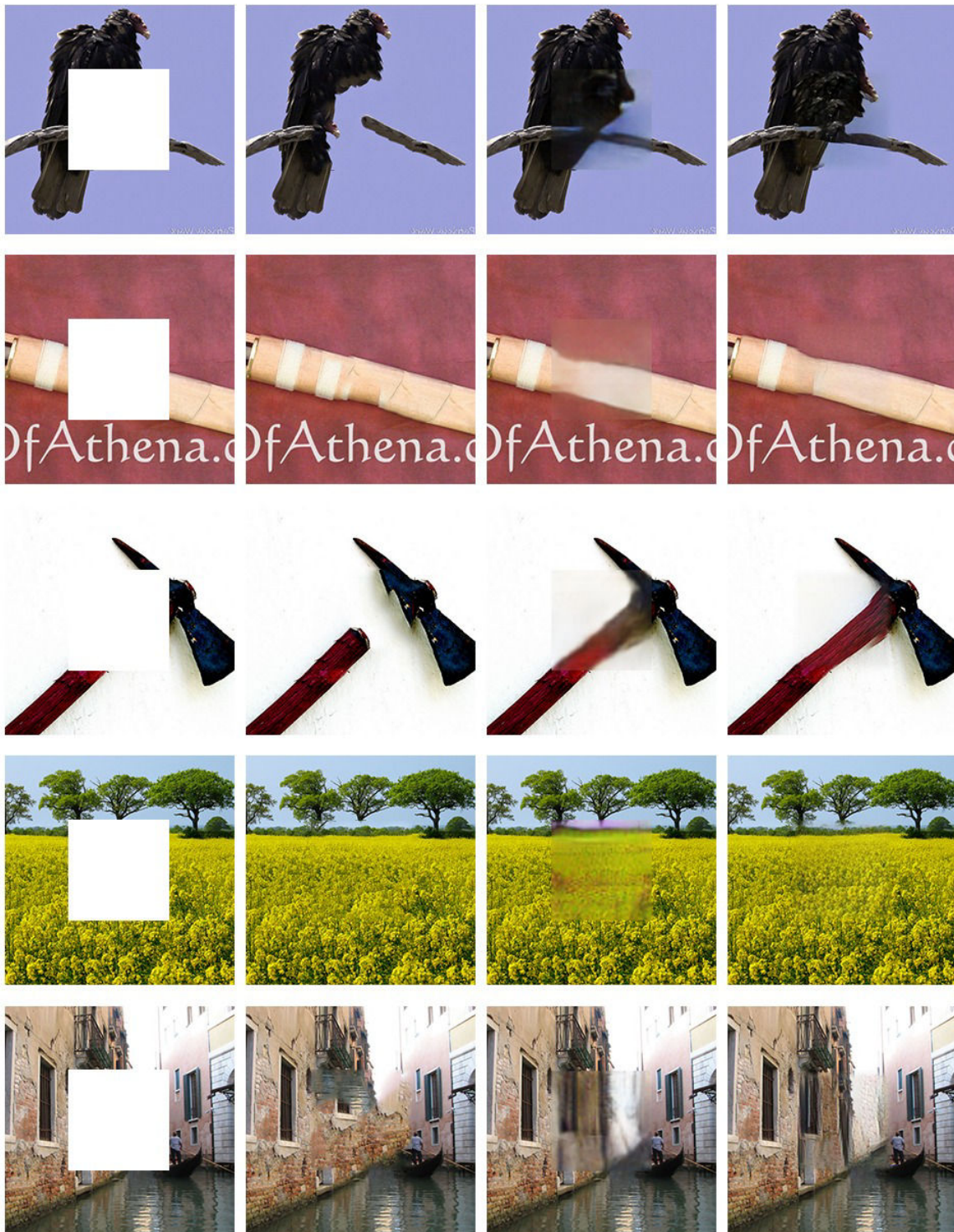


Figure 5.9 – Results of different image inpainting methods. From left to right: Input context, patch-based method ([Newson, Almansa, Gousseau, et al. 2017](#)), Context Encoder ([Pathak et al. 2016](#)), hybrid method ([C. Yang et al. 2017](#)).



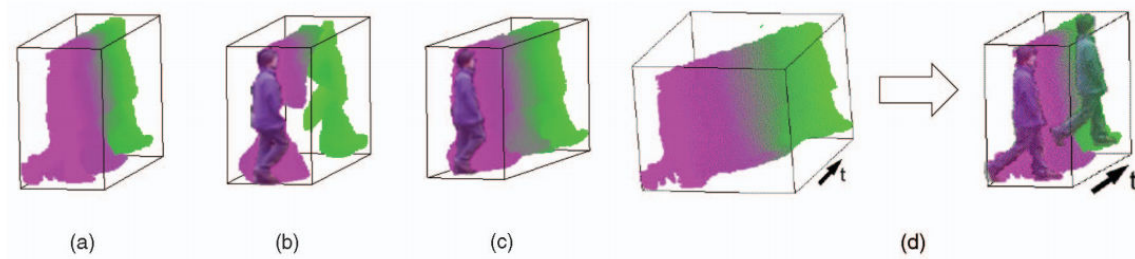


Figure 5.10 – (Jia, Tai, et al. 2006) perform video inpainting by repairing cyclic motions. (a) Sample move, (b) damaged move, (c) repaired move, (d) sample move after wrapping and stabilization. Image taken from (Jia, Tai, et al. 2006).

followed by a homography blending to avoid artifacts. This approach could guarantee spatial and temporal structure consistency under variable illumination. However, they supposed a strict periodicity of motion. Furthermore, they also require the user's manual intervention to segment the video into different layers of coherent motion.

Authors from (Venkatesh, Cheung, and J. Zhao 2009) propose a simplified version of (Jia, Tai, et al. 2006). Their object-based hole completion method tracks the occluded moving object to build a set of segmented frames where the moving object is fully unoccluded. The missing regions are then inpainted by realigning the best segment in the set and pasted into the occlusion. As other methods in object-based video inpainting, the background is completed using the patch-based method of (Criminisi, Pérez, and Toyama 2004). This method is later extended by (Ling et al. 2009) using the contour of the occluded object to retrieve the object frames from the set.

Another approach which has some common points with object-based video inpainting was that of (Shih et al. 2008). They propose a method to falsify the video, which means to modify the motion of people. In this method, after separating the background and foreground, the moving foreground is divided into different moving objects. A 2D skeleton model of each tracked object in the video is built to reduce the complexity of the patches matching search. Similar to different methods in this family, they require the cyclic motion to be able to provide correct results.

With a different approach, (Granados, K. I. Kim, et al. 2012) proposed a semi-automatic algorithm using graph-cut to solve the homography between two consecutive frames. Then they completed the background by warping and image inpainting techniques. This method was designed to handle a free moving camera, but it requires user's assistance to keep the desired motion in the background.

Another approach using Gaussian mixture models (GMMs) to distinguish the background and foreground of the entire video was proposed by (Xia et al. 2011). This permits to save time for calculating the optical flow mosaic. After that, the video is reconstructed by copying as much as possible the information from other frames pixel by pixel, and the remaining hole is filled by example-based image inpainting techniques. The main drawback of this technique is that the GMMs model have difficulty to handle large camera movement and the simple pixel

copy/paste framework cannot deal with complex scenes.

Typically, object-based systems are fast and can provide some reasonable results for a specific object, especially human, in simple scenes. However, they work under some strict conditions such as periodic motion or user-assistance to obtain accurate segmentations of moving objects and static background. Furthermore, the foreground and background completion procedures are performed independently so blending one with the other may cause artifacts.

### Patch-based video inpainting

Different from the object-based approach which consider the temporal information using the segmentation of video objects, patch-based approach use spatio-temporal patches to ensure the global coherency in the space-time domain. In this approach, patches from source regions are used to fill-in the occlusion. It is therefore important to find the best exemplar patches from the source first. After that, the occluded pixels will be reconstructed using these exemplary sources. Similar to patch-based image inpainting, the filling of the hole can be performed in a greedy or global manner.

**Greedy approach** The idea of using greedy methods for video inpainting has been introduced in (Patwardhan, Sapiro, and Bertalmio 2005). Their video inpainting method was considered as an extension of the well-known exemplar-based image inpainting approach introduced in (Criminisi, Pérez, and Toyama 2004). The inpainting algorithm starts with moving objects. During this process, the priority term in (Criminisi, Pérez, and Toyama 2004) is modified: it is computed based on the local amount of undamaged pixels and the motion direction. Occlusion is reconstructed greedily. Starting with the highest priority, the method copies those patches that best match the target patch with respect to the known pixels. The similarity between patches is obtained by calculating the distance between the patches textures and motion vectors. The whole process is shown in Figure 5.11.

For the remaining static background, they exploit temporal redundancy, using similarity between frames, which means temporal information is copied where possible if the background becomes visible at a particular frame. For the remaining hole corresponding to positions which are never revealed, they using again the method of (Criminisi, Pérez, and Toyama 2004). This method can handle translational camera motions and is later extended to handle more general camera motions in (Patwardhan, Sapiro, and Bertalmio 2007) by using foreground mosaics.

With a similar approach, (Daisy et al. 2015) employed a tensor voting term when calculating the priority. On the other hand, they focused on constructing a geometry-guided blending technique to reduce space-time artifacts caused by reconstructed patches.

In conclusion, the greedy approach is often one-pass greedily propagating information from outside the hole into the missing region. These approaches are reasonably fast and

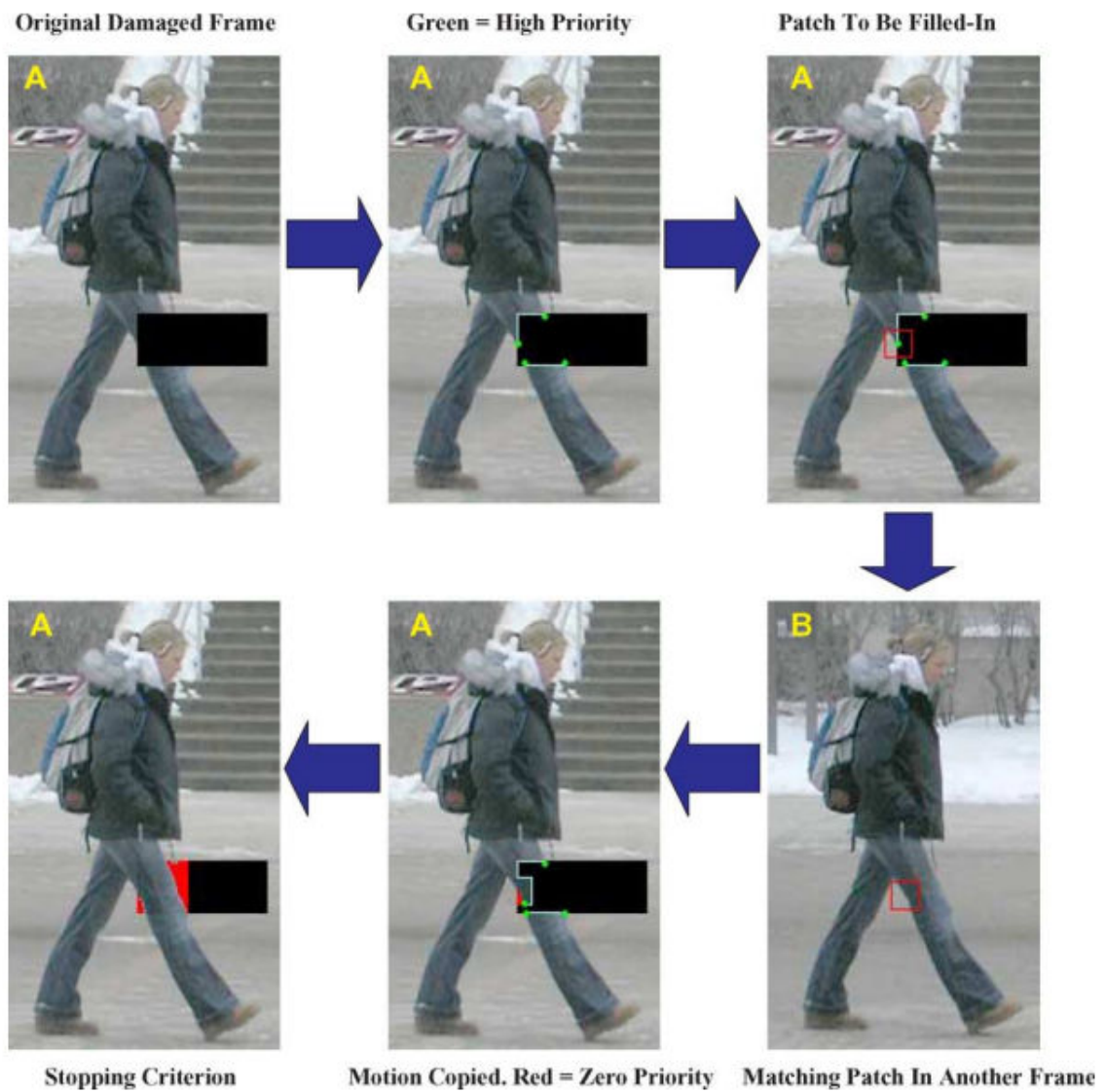


Figure 5.11 – Motion inpainting scheme in the greedy method of (Patwardhan, Sapiro, and Bertalmio 2005). Image from (Patwardhan, Sapiro, and Bertalmio 2005).

enable plausible results when the hole is small. However, they often cannot preserve the spatio-temporal coherence, especially with respect to moving objects. In addition, the greedy patch filling process inevitably propagates errors at early steps to the subsequent steps, often yielding globally inconsistent results.

**Globally optimized approaches** To increase the spatial coherency, globally optimized approaches have been proposed. In such approaches, the inpainting problem is solved through optimizing a single energy patch-based function to provides globally consistent results.

A pioneer work in this category was the method of (Wexler, Shechtman, and Irani 2007). Under this method, the video inpainting problem is formulated as a problem of optimizing a global energy function based on 3D spatio-temporal patches which ensure global coherence.

$$E_{\omega}(u) = \sum_{p \in \mathcal{H}} \min_{q \in \mathcal{D}} d(\psi_p, \psi_q)$$

where  $\mathcal{H}$  and  $\mathcal{D}$  are the occlusion and the source region, respectively.  $\psi_p$  is the spatio-temporal patch centered at  $p$ . This eenterenergy specifies that we want each spatio-temporal patch inside the occlusion to resemble its nearest neighbor as much as possible. The minimization process is achieved thanks to an expectation maximization (E-M)-like procedure. This procedure requires repeated uses of 2 main steps: nearest neighbor search and reconstruction of pixel values inside the occlusion. In the first step, space-time source patches inside the occlusion are sampled to find their best nearest neighbor candidate in the source region. Then, in the second step, the missing region is synthesized as a field of overlapping patches copied from the known region.

In the first step, since the exact nearest neighbor search is way too costly for for videos, (Wexler, Shechtman, and Irani 2007) accelerate the search by adopting a  $\epsilon - ANNs$  scheme. For images, (Barnes, Shechtman, et al. 2009) imprves the approximate search using a random search and propagation scheme, resulting in the PatchMatch algorithm. The combination between (Wexler, Shechtman, and Irani 2007) (image version) and (Barnes, Shechtman, et al. 2009) is the core of the famous Content-Aware-Fill tool of Photoshop. However, this is limited to image completion.

Later, (Newson, Almansa, Fradet, et al. 2014) made a great amendment to the method of (Wexler, Shechtman, and Irani 2007) by using the 3D PatchMatch- an extension of (Barnes, Shechtman, et al. 2009) to video. This replacement not only speed up the patch searching step but also strengthen the coherence. They also compensate dominant camera motion with global affine transformation and introduce a texture term in the patch distance which allows them to reconstruct textured regions. This approach can handle both structured and textured scenes and provide high quality inpainting results without any segmentation. They finally came to a very impressive result with high definition videos while dramatically reducing the computation time.

In (Y. Shen et al. 2006), the authors proposed a video inpainting method base on motion

manifold. The idea is to inpaint pixels along the manifolds of each moving object through a set of patches from the source region. To reduce the search space, each moving object is tracked with a mean-shift tracker. Therefore, the searching window for each missing pixel is reduced from 3D to 2D.

Another global approach is the method of (Granados, Tompkin, et al. 2012). They solve the problem in an original way which focuses on solving approximate nearest neighbor shift map in 3D space using graph-cut. They obtained very good results with dynamic scenes and moving objects. However, the computation time was huge compared to that of (Newson, Almansa, Fradet, et al. 2014).

Recently, (J.-B. Huang et al. 2016) modify the energy of (Wexler, Shechtman, and Irani 2007) by adding an optical flow term to enforce temporal coherence of the synthesized contents as well as propagating the known content into the unknown regions. They used 2D patches instead of 3D patches. To the best of our knowledge, this is arguably the method which yields the best result and is compatible with many scenarios, as long as the content is not too complex, and in particular does not contain dynamic textures.

**Optical flow-based video inpainting** Many works in video inpainting field use 3D spatio-temporal patches as a simple solution for preserving spatial and temporal coherency while some others use motion information, such as optical flow, instead. For example, in (Strobel, Diebold, and Cremers 2014), the authors inpaint the optical flow field first and use the result to guide the patch searching algorithm. The idea of using optical flow was also reported in (J.-B. Huang et al. 2016). They not only rely on optical flow to find the best patches but also use it to reconstruct the pixels. As a result, they can achieve more temporal consistency than others. However, their method use only 2D patches, and optical flow is therefore the only ingredient for the preservation of temporal coherence, which at time may generate artifacts. They also require re-computing an optical flow field at each iteration which make the computation heavier. To benefit from the advantages of these two solutions, in our method, we use both 3D spatio-temporal patches and optical flow fields.

## 5.3 Proposed method

Figure 5.12 demonstrate the principle of our algorithm. In general, our aim is to use source patches to fill-in the occlusion in a global manner. The way to achieve this is to optimize a non-local patch-based energy, in the spirit of (Wexler, Shechtman, and Irani 2007; Newson, Almansa, Fradet, et al. 2014). This energy minimization serves both for selecting the source patches and for reconstructing pixel intensity values. The energy is minimized thanks to an iterative procedure embedded in a coarse-to-fine pyramidal scheme. Our algorithm involves two core steps: (i) the computation of a nearest neighbor field which finds the best approximation in the source region  $\mathcal{D}$  for each patch in the occlusion  $\mathcal{H}$ ; and (ii) a



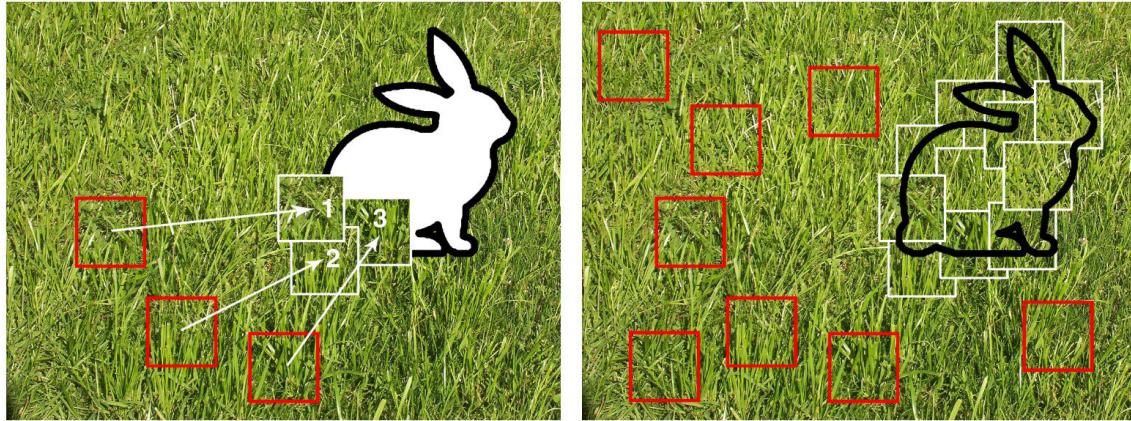


Figure 5.12 – Different from greedy approaches which copy and paste one patch at a time (left), our global approach copy patches from the source region and paste them into the occlusion all at the same time (right).

reconstruction step using this field to determine the values of all occluded pixels.

Within this framework, it is essential to address many problems. The first one is the deformation of patches caused by the camera motion. This deformation may make our spatio-temporal patches unaligned and lead to a wrong decision when finding the best nearest neighbor candidate. Next, we must deal with the coherent preservation of motion, which is usually difficult to maintain across a long occlusion. After that, the searching strategy to find the appropriate patches must be modified when dealing with complex motion. Another important problem is the border artifacts which make the inpainting results unsatisfying. Last but not least, the computational time must be reduced to have a practical application.

These problems are addressed in our method by five mechanisms, namely:

- adopting a stabilization pre-processing step,
- modifying patch shapes and patch metric,
- integrating a novel optical flow-driven initialization scheme,
- speeding up the nearest neighbor search and
- enhancing the reconstruction step and parallelizing the algorithm.

These techniques will be presented in the following subsections.

### 5.3.1 Pre-processing

Over the years, different approaches have been proposed to manage the deformation caused by camera motion, in the context of image inpainting. For instance, (Granados, Tompkin, et al. 2012) uses homography, (Ebdelli, Le Meur, and Guillemot 2015) utilizes registration technique in a temporal slide window. Another approach is that of (J.-B. Huang et al. 2016)

which predicts the optical flow and exploits the generalized version of PatchMatch ([Barnes, Shechtman, et al. 2009](#)) which treats the geometry transformation as an additional variable. However, this approach made the searching algorithm convergence slower and could lead to a wrong selection of patches. By experimenting with different kinds of camera motions, we found out that employing a video stabilization technique as a preprocessing step is a simple but efficient solution. This video stabilization step aims at compensating the patch distortion, at obtaining repetitive spatio-temporal patches and at supporting the patch searching algorithm. The fundamental properties behind patch-based methods is that contents are redundant and repetitive. To maintain these properties after the addition of the temporal dimension, we need an accurate stabilization algorithm.

In ([Newson, Almansa, Fradet, et al. 2014](#)), the authors used the method of Odobez and Bouthemy ([1995](#)) to calculate the motion model between two consecutive frames. This can handle some simple camera behaviors: translation, rotation, divergence, pan-tilt-zoom, etc. But it doesn't work with large displacement camera movement. Moreover, the complex motion of background makes this method does inoperative in several difficult cases.

Most efficient (non-learning) techniques in video stabilization calculate the transformation between two frames using a feature-based approach. These techniques include the computation and matching of some robust features such as SIFT, SURF, then use RANSAC-like algorithm to remove outliers ([K.-Y. Lee et al. 2009](#); [Matsushita, Ofek, Xiaoou Tang, et al. 2005](#)). These methods are robust to various conditions such as outliers, illumination changes, and large camera displacements. However, the quality of the estimation depends strictly on the feature points detection and matching, and will fail if an insufficient number of feature points is detected, which is usually the case in the presence of motion blur. Moreover, if the background is dynamic, e.g. video of waves or fountain, then the transformation matrix estimated by RANSAC algorithm is often not accurate. The reason behind that is that the displacement of features points not only caused by camera movement but also by dynamic texture. This often leads to a flickering effect which degrades the inpainting quality. In the literature, this is often solved by introducing a smoothness term to ensure a smooth camera path ([Sánchez 2017](#)). However, the performance still may not be good enough for our application.

To deal with this situation, we apply a refinement step using the pixel-based parametric image alignment method of ([Evangelidis and Psarakis 2008](#)). The principle of this method is that the geometric transformation between the input and the template image can be estimated using an enhanced correlation coefficient (ECC) maximization scheme. In particular, the expected transformation is the one that maximizes the correlation coefficient between the template and the warped input image. This is a non-convex optimization problem and it is solved by an iterative scheme. Therefore, it needs a good initialization to be able to converge to a global optimum. In our stabilization scheme, this initialization solution is obtained using a feature-based method. More specifically, after calculating the transformation by the feature-based method, we use the output matrix as an initialization to the ECC-based method

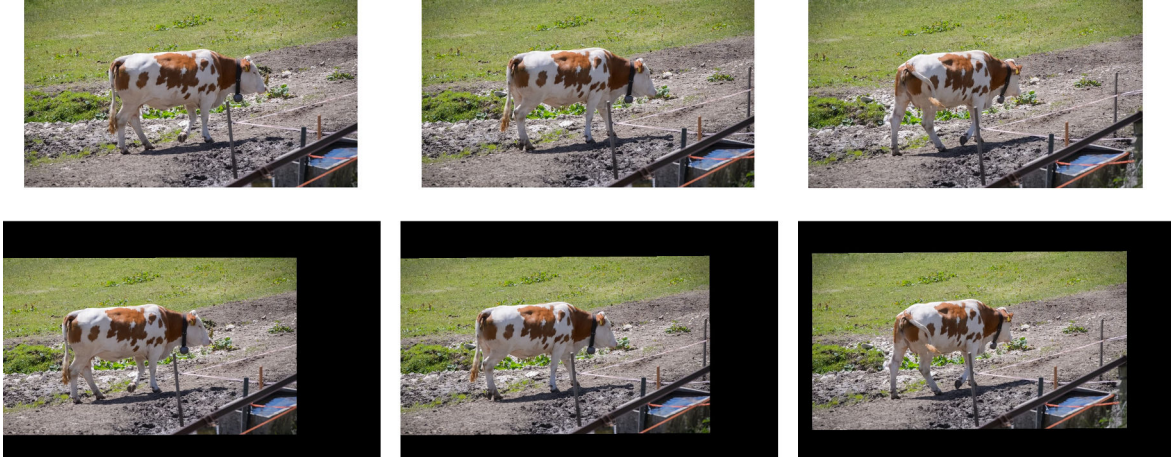


Figure 5.13 – Representative frames of our video stabilization method. Top (original video), bottom (stabilized video).

(Evangelidis and Psarakis 2008). By using pixel-based template refinement, this method is robust for dynamic scenes and produces accurate results. Moreover, it can avoid the case when no key points are detected.

We use the homography as a geometric transformation matrix which has the highest degree of freedom. As indicated in (Sánchez 2017), the type of model has a strong effect on the final results, and it depends on the input video. Systematically choosing which model is best is out of the scope of the thesis.

After computing the homography, each frame in the video is aligned to a reference frame which is chosen to be the middle frame of the video. We also add zero padding to avoid losing information in the border while doing frame alignment. Note that the size of the video will increase due to zero-padding but this does not affect the speed of the algorithm because the computation time depends only on the occlusion size. Some representation frames of our stabilization method are shown in Figure 5.13.

Once the video is stabilized, we apply the same transformation to the occlusion domain. This new video together with the new occlusion is given as an input to the patch-based video inpainting technique. When the inpainting step finishes, we apply an inverse transform to the inpainted video and merge it with the original video to obtain the final result.

### 5.3.2 Energy

After stabilization, we minimize a Wexler-like energy  $E(u, \phi)$  to find the inpainted sequence  $u$  and the corresponding patch correspondence (or shift map)  $\phi$ . Denoting  $\psi_p(u)$  the patch centered at a pixel  $p$  in the given occlusion domain  $\mathcal{H}$ , the shift  $\phi(p)$  at  $p$  is defined as the spatial offset  $q - p$  where  $q$  is a minimizer in  $\mathcal{H}^c$  of the patch distance  $d(\psi_p^u, \psi_q^u)$  (see below for the definition of  $d$ ). The energy  $E$  associated with an image  $u$  (known outside a given

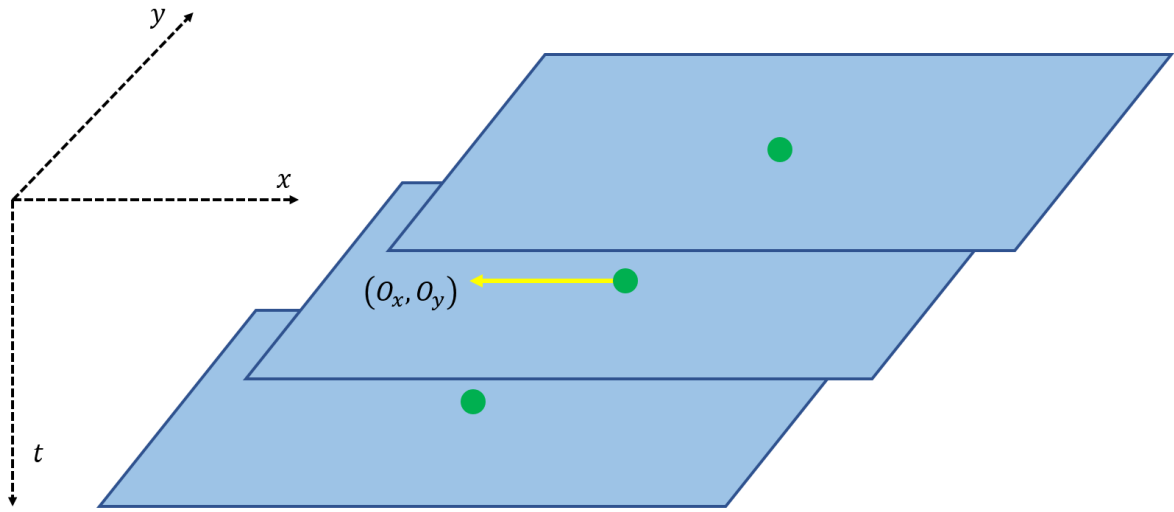
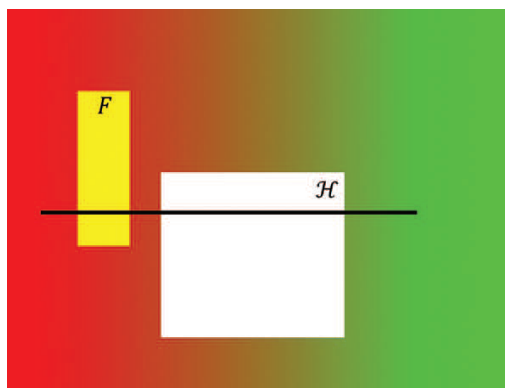
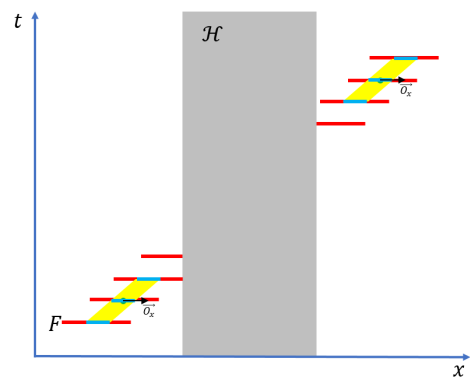


Figure 5.14 – The shape of the patch is a parallelepiped with skew in  $x$  and  $y$  directions controlled by its optical flow vector.



(a)



(b)

Figure 5.15 – (a) A synthetic video where the foreground (yellow rectangle) crosses the occlusion. (b)  $x$ - $t$  profile along the black line slice.



occlusion domain  $\mathcal{H}$ ) and a shift map  $\phi$  is defined as

$$E(u, \phi) = \sum_{p \in \mathcal{H}} d^2(\psi_p^u, \psi_{p+\phi(p)}^u)$$

Minimizing this energy ensures that each patch  $\psi_p^u$  centered around a pixel  $p$  in the occlusion domain  $\mathcal{H}$  is as close as possible to its nearest neighbor  $\psi_{p+\phi(p)}^u$  outside the occlusion (in the sense that  $p + \phi(p) \notin \mathcal{H}$ ). We use a metric  $d$  between patches defined by:

$$d^2(\psi_p^u, \psi_q^u) = \frac{1}{|N_p|} \sum_{\substack{r \in N_p \\ r-p+q \notin \mathcal{H}}} \left[ \alpha \left( \|u(r) - u(r-p+q)\|_2^2 \right) + \right. \\ \left. \beta \left( \|T(r) - T(r-p+q)\|_2^2 \right) + \gamma \left( \|O(r) - O(r-p+q)\|_2^2 \right) \right]$$

where the terms  $T$  and  $O$  will be defined shortly. In this expression,  $N_p$  indicates a spatio-temporal neighborhood of  $p$ . Instead of using a rectangular cuboid, we use a parallelepiped as a neighbor region. The skew in the  $x$  and  $y$  direction of this parallelepiped is controlled by its own optical flow vector as indicated in Figure 5.14. Each patch has its own shape. We can see this step as a coordinate normalization following the optical flow vector. This adaptive shape helps us to capture more information into one single patch and to avoid the case where both background and foreground are present in one patch. In a situation where an object moves in front of different backgrounds if a patch contains both this object and background, it is arduous to find the appropriate match for this patch because the background parts of all the patches with the same foreground part are disparate. On the other hand, by using a parallelepiped with optical-flow driven shape, the patch now contains only the information of that moving object and its best nearest neighbor candidate can more easily be found. This is illustrated in Figure 5.14 (b) where a synthetic video with a simple moving foreground cross a fixed occlusion and the background is a color gradient. If we use normal cuboid patches, the patch will contain a part of the yellow foreground and a part of the background. We denote this patch as  $P_A$ . Since the background is different, there will be no exact match for  $P_A$ . However, if we use the parallelepiped shape,  $P_A$  now contains only the information of the foreground and its nearest neighbor can be found easily.

Following (Newson, Almansa, Fradet, et al. 2014), our distance incorporates texture features  $T = (|\frac{\partial u}{\partial x}|, |\frac{\partial u}{\partial y}|)$ , made of the norm of the spatial gradient in  $x$  and  $y$ . The motion information is represented by  $O = (|O_x|, |O_y|)$ , the norm of each component of the optical flow vector. The terms  $\alpha, \beta, \gamma$  are weighting coefficients. Together with the color value of the pixel, texture and optical flow features play a very important role in guiding the nearest neighbor searching step. While texture features are extremely helpful (for example, they help distinguish between a flat region and an oscillating region with the same color mean value), optical flow features help the algorithm in finding the right patches for moving objects (for example, they



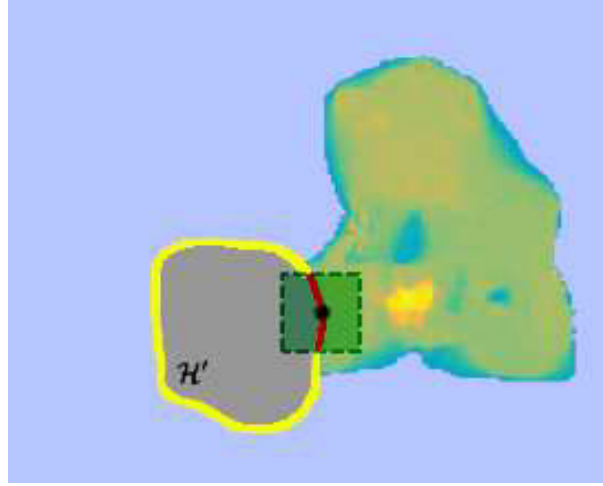


Figure 5.16 – Initialization driven by the optical flow. After selecting the patch which has the highest optical flow, we inpaint the region which has highest priority (in this case the red region, which is the intersection between that patch and the border).

can distinguish between two objects with the same appearance but with different motions). Textures feature and motion features are used at all pyramid levels and they are reconstructed simultaneously with the pixel intensity in the reconstruction step (to be described thereafter). Weights for each component are set to 1, 50, 50 in our implementation. Notice that the optimal weights may depends on the input; their automatic setting is a challenging problem and is the purpose of ongoing work.

Our energy  $E$  is high dimensional and highly non-convex, but as observed in (Arias, Caselles, and Facciolo 2012), a good local minimum can be obtained by alternate minimization *w.r.t*  $u$  and  $\phi$ , coupled with a good initialization and a coarse-to-fine multi-scale scheme. Texture and motion features in the similarity metric are key to guiding the algorithm towards a good local minimum from the coarsest scale. Three volume pyramids including the RGB images volume, spatial gradient volume, and optical flow volume are built up by subsampling their versions at the finest level. These three volume pyramids play an important role in assisting the algorithm to search for appropriate patches. They will be reconstructed simultaneously during the reconstruction step. Starting with the coarsest level, we proceed to the finer level by interpolating the nearest neighbor field followed by a reconstruction step. The general structure of our algorithm is as follows:

- Build multiscale pyramids for color  $u$ , occlusion domain  $\mathcal{H}$ , texture features  $T$  and motion features  $O$ .
- Initialization at coarsest scale (see section 5.3.3).
- From coarsest to finest scale do:
  - Iterate until convergence:
    - \*  $\min_{\phi}$  (source patch searching, section 5.3.4).

- \*  $\min_u$  (pixels reconstruction, section 5.3.5).
- \* features reconstruction.
- If not finest scale: Upsample  $\phi$ ,  $u$ , and features.

### 5.3.3 Coarse initialization

Due to the non-convexity of the functionals which are typically used in global patch-based methods, having a reliable initialization is crucial for the local minimization. Nevertheless, this step is often left unspecified in the literature. In several state-of-the-art methods, the coarse initialization is obtained using the median filter or spatial interpolation. However, the median filter requires occluded pixels in the temporal stack to be available at least in one frame. The spatial interpolation cannot provide solutions with good spatial-temporal continuity. Recently, (Newson, Almansa, Fradet, et al. 2014) proposed a greedy inpainting technique in which the order of regions to inpaint was determined by the onion peel ordering. In their method, they inpainted the occlusion layer by layer from the border toward the center. This scheme can produce a good initialization for small occlusion. However, in the case when moving objects are occluded for a long period, onion peel scheme tends to wipe out the moving object, so foreground objects consequently could not be reconstructed. To handle this situation, we propose an optical flow-driven inpainting technique inspired by (Criminisi, Pérez, and Toyama 2004) and (Nishihara et al. 2011), in which the order of the patch to inpaint is determined by an optical flow-based priority term. More precisely, the priority term of the pixel  $i$  is defined as follow:

$$Pr_i = C_i \cdot D_i$$

where  $D_i$  is the average of the optical flow magnitude in the patch centered at  $i$ , and  $C_i \propto \exp\{-d^2(i, \mathcal{H}_{\text{coarse}})\}$  measures how close the pixel  $i$  is to the border of the original occlusion  $\mathcal{H}_{\text{coarse}}$  at coarsest scale).

After thresholding the optical flow field by an adaptive method (Otsu 1979) to determine background/foreground label for each pixel, we repeat this procedure until all the foreground pixels are reconstructed. This is demonstrated in Figure 5.16.

- Let  $\mathcal{B}' = \mathcal{H}' \setminus (\mathcal{H}' \ominus B(0, 1))$ , i.e.  $\mathcal{B}'$  is the one-pixel wide outer boundary of  $\mathcal{H}'$ . Calculate  $Pr_i$  for  $i \in \mathcal{B}'$ .
- Select patch  $P_i$  with highest priority term  $Pr_i$ , and define the region to inpaint  $R_i = P_i \cap \mathcal{B}'$ .
- Inpaint  $R_i$  and get new occlusion region  $\mathcal{H}' \rightarrow \mathcal{H}' \setminus R_i$ .

Thereafter, the rest of the occlusion which is the background (determined by Otsu thresholding on the optical flow, as explained above), will be inpainted following onion peel order. Unlike (Newson, Almansa, Fradet, et al. 2014) who used erosion in both spatial and temporal dimensions to get one onion layer of the occlusion (onion peel 3D), we only erode in

temporal dimension (onion temporal) to get one layer of the occlusion. This is to enforce the temporal coherence along the temporal direction and therefore reducing the oscillation effect. After these step, a vertical hole may appear since some occluded pixels do not reveal in the temporal stack. In that case, we use onion peel scheme to complete one frame, then repeat our onion peel temporal step.

In conclusion, this initialization scheme driven by the optical flow helps us at reconstructing moving object at the coarsest level and providing a good start for the algorithm at the beginning. By this scheme, we can keep the motion in a long lasting occlusion. The moving objects on finer level will be maintained through integrating the optical flow in the distance metric. Effect of different components will be analyzed in Section 5.4.

### 5.3.4 Source patch searching

To minimize the energy mentioned in Section 5.3.2 w.r.t  $u$ , we need to find the best candidate for each patch in the occlusion region given the pixel intensity of all the occluded pixels. This is equivalent to find the nearest neighbor field for all patches inside the occlusion. Finding an exact nearest neighbor search has huge computational complexity and is often not applicable in practice, therefore, it is common to replace it by an approximate nearest neighbor search.

Since its introduction, PatchMatch (Barnes, Shechtman, et al. 2009) has become a very popular method for searching similar patches. It is a common choice in image and video inpainting, not only for its computation speed but also for the spatial consistency it implies. The core part of the algorithm includes a propagation step to spread out good matches and a random search step to jump out of the local optima. These two steps are repeated in several iterations. Recently, many methods in the literature are proposed to improve PatchMatch in both accuracy and computational complexity such as Coherency Sensitivity Hashing (Korman and Avidan 2011), Propagation-Assisted KdTrees (He and J. Sun 2012a) or PatchTable (Barnes, F.-L. Zhang, et al. 2015). All of these algorithms transform the patches onto another space e.g. Walsh–Hadamard space using a set functions. These methods work very well, and outperform PatchMatch in the case of images. However, for the case of videos, the idea of projecting 3D patches on another space to reduce the dimension is less straightforward because this requires to store all main coefficients of each patch into instance memory, which is very large and often not practical. For example, if we only represent each patch by a 20-dimensional vector with 32-bit floating number, for a video with size  $854 \times 480 \times 100$ , we need  $854 \times 480 \times 100 \times 20 \times 4 \times 3 \approx 9$  Gb just to store the representation of patches while PatchMatch only need  $854 \times 480 \times 100 \times 3 \approx 0.12$  Gb, because it works directly on pixel intensity. Therefore, these method have not yet been extended to video.

In our video inpainting context, the spatio-temporal extension version of PatchMatch by (Newson, Almansa, Fradet, et al. 2014) is adopted with several modifications to improve its efficiency. We first explain quickly the core details of the algorithm and we propose our modifications:

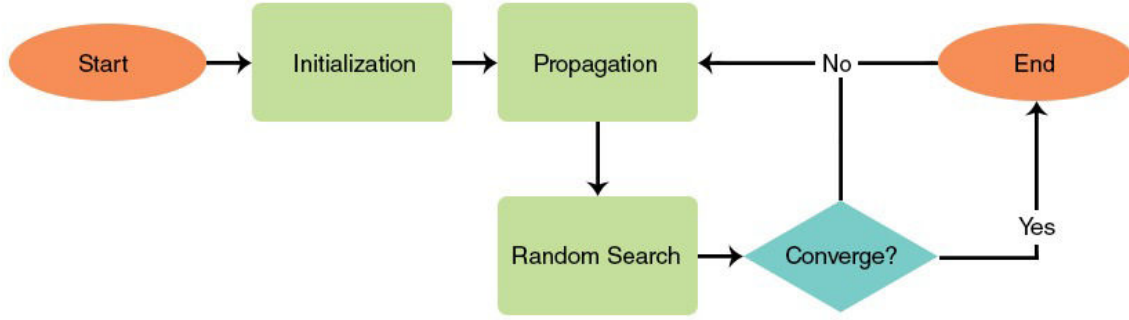


Figure 5.17 – The work flow of 3D-PatchMatch algorithm.

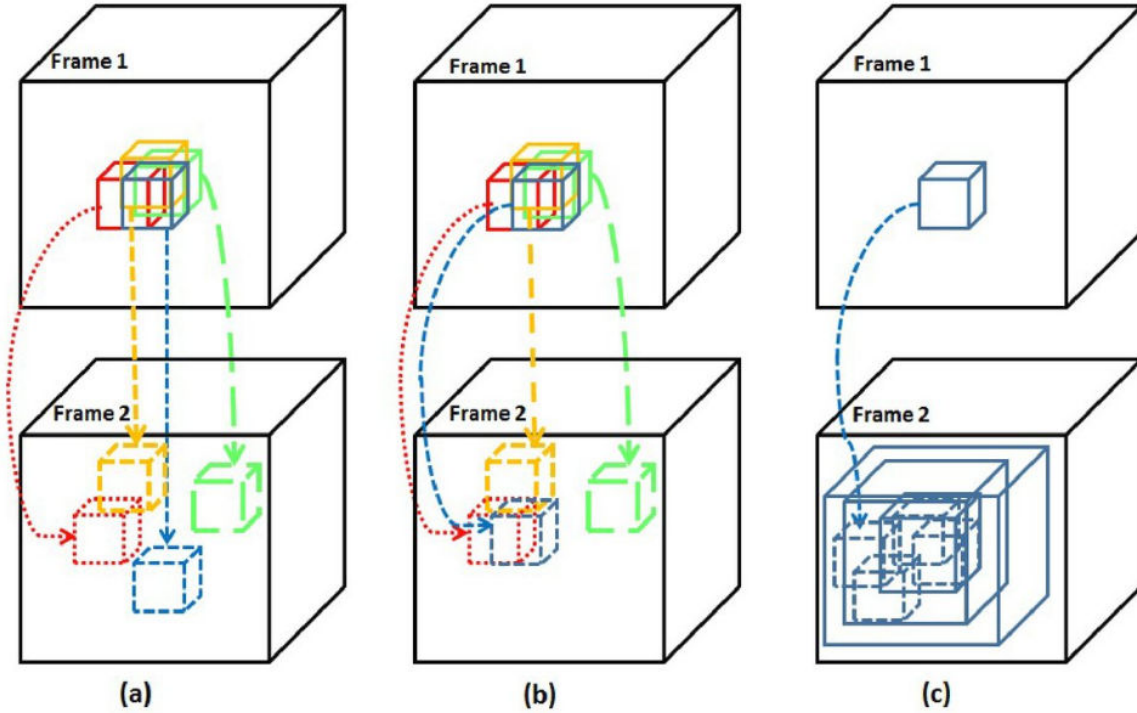


Figure 5.18 – The 3D PatchMatch search consists of 3 main step: (a) initialization, (b) propagation and (c) random search. Image from (Zontak and O'Donnell 2016).

**3D PatchMatch search** The intuition behind PatchMatch search is simple: neighboring patches should have neighboring nearest neighbor candidate. This means the correspondences map defined by the spatial offsets tend to be smooth in natural images. This idea is also true for natural video and there is even more coherence in the temporal direction.

Figure 5.17 presents the work flow of the 3D-PatchMatch algorithm. It is made of three main steps: *initialization*, *propagation* and *random search*. After initialization, the propagation and random search steps are repeated until convergence. These steps are illustrated in Figure 5.18. We detail these steps below:

- *Initialization:* Initially, if no information is available, the shift map is initialized at random by assigning each patch in the occlusion  $\mathcal{H}$  with a random candidate in the source region  $\mathcal{D}$  with independent uniform sample. After that, since our inpainting

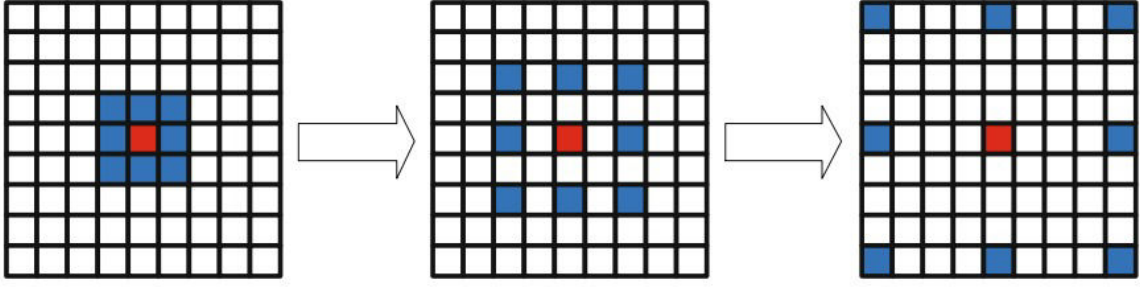


Figure 5.19 – The jump flood technique in the propagation step. In each round, each patch, e.g., the red patch, investigates different blue neighbors with different offsets. The offset  $l$  is doubled after each round. From left to right,  $l = 1, l = 2, l = 4$ . For clarity, we demonstrate here the jump flood technique for 2D PatchMatch, the same principle is also applied for the 3D version.

method is iterative, the shift map from the previous iteration is used as an initialization for the PatchMatch algorithm at current iteration.

- *Propagation:* This step iteratively propagates good matches to nearby area. The algorithm uses the offset of nearby pixels as alternative estimates of the current offset, and selects the best one. If a good offset is available for a given pixel of a region with constant offset, this will very quickly propagate to the whole region. More specifically, In this step, the video volume is lexicographically scanned in two rounds. In the first round, pixels are scanned from top-left-front corner of the volume and good match will be propagated from left to right, top to bottom, front to end. For a given patch  $\psi_p$  at location  $p = (x; y; t)$ , the algorithm considers the following three candidates :  $\psi_{p+\phi(x-1;y;t)}$ ,  $\psi_{p+\phi(x;y-1;t)}$  and  $\psi_{p+\phi(x;y;t-1)}$ . It then select the best patch among these three candidates. In the next round, The scanning order is reversed, and the algorithm consider 3 remaining patches:  $\psi_{p+\phi(x+1;y;t)}$ ,  $\psi_{p+\phi(x;y+1;t)}$  and  $\psi_{p+\phi(x;y;t+1)}$
- *Random search:* To avoid getting trapped in bad local minima after each propagation step, a random search step follows, based on a random sampling of the current offset field. The position of random candidate  $q_i, i = 1, \dots, L$  are chosen as

$$q_i = p + \phi(p) + R_i$$

where  $R_i$  is a bi-dimensional random variable, uniform over a square grid of radius  $2^{i-1}$  excluding the origin. This formula is derived from the fact that in practice, most of these new candidates are pretty close to the current candidate  $p + \phi(p)$ , but large differences are also allowed, with small probability.

**Our improvements of 3D PatchMatch search** Next, we discuss our modification of the 3D PatchMatch search to improve it speed and efficiency.



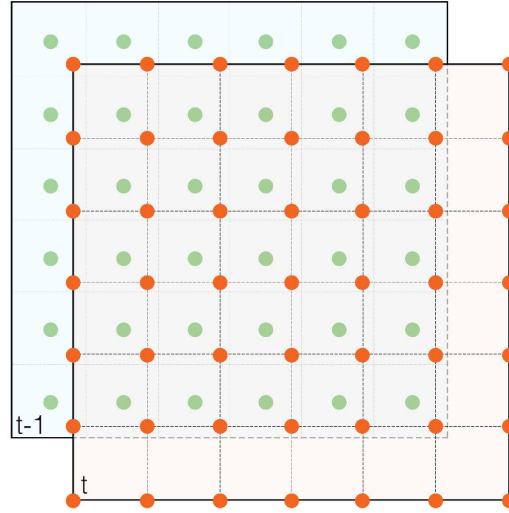


Figure 5.20 – The random search step is only applied in the spare grid to reduce the computational time. Grid at time  $t$  and time  $t - 1$  are interleaved.

1. The first modification is to speed up PatchMatch by parallelizing the algorithm. Notice that the initialization and the random step itself are trivially parallelizable, so the main challenge is to parallelize the propagation operation. For this step, we adopt the jump flood scheme of (Rong and Tan 2006). More specifically, the propagation steps contains several round. For a  $n \times n \times n$  volume, there are  $\log n + 1$  rounds with doubled offset of  $1, 2, \dots, n/2, n$ . Figure 5.19 illustrates how the information is propagated. In each round with offset  $l$ , for each patch  $\psi_p$  at location  $p = (x, y, t)$  in the occlusion, we exhaustively try each patch  $\psi_{p+\phi(x+w, y+h, t+d)}$ ,  $w \in \{-l, 0, l\}$ ,  $h \in \{-l, 0, l\}$ ,  $d \in \{-l, 0, l\}$  as a new candidate patch in the source region. The best patch among these three candidates is then selected. We note that in the serialized version, the raster scan operation is capable of propagating information all the way across a scan line. But in practice, long propagations are not needed, and a maximum jump distance of  $n = 32$  suffices. With this scheme, each thread of the kernel of the GPU computes a nearest neighbor to one patch in the occlusion. Therefore, the propagation operation is fully parallelized.

Another amendment to save the computation time is to use a spare grid during the random search step (the most costly step of the algorithm). We realized that it is not necessary to perform a random search step for every occluded pixel. Instead, we can apply this step only for pixels on a sparse grid without losing the efficiency. Spatial grid at time  $t$  and  $t - 1$  are interleaved as illustrated in Figure 5.20. For occluded pixels which do not lie in this grid, their nearest neighbors candidate are updated via the propagation step. The grid size is set to 1 at the coarsest level and it doubles when we upsample the pyramid. To boost the speed even more, we use foreground/background patch clustering to reduce the search space. When looking for the nearest neighbor candidate for foreground/background pixel, we only look in the foreground/background

region.

As a result, combining these three factors enables our patch searching algorithm to run additionally 5-7 times faster than the traditional 3D PatchMatch of (Newson, Almansa, Fradet, et al. 2014) with the same level of accuracy.

2. The second modification is the use of an optical flow guide for the propagation. From the assumption that adjacent patches are more likely to have similar nearest neighbor candidates, PatchMatch can preserve well the spatial coherency through the propagation step. For temporal coherency, this assumption is still true only if the background is static or in periodic motion. Otherwise, this can not hold. To enforce the temporal consistency, instead of propagating offsets to a fixed temporal neighbor, we propagate it following the optical flow direction. To be more formal, in the propagation step, the temporal neighbor for the patch centered at pixel  $(x, y, t)$ ,  $P_{(x,y,t)}$ , is  $P_{(x+O_x, y+O_y, t+1)}$  rather than  $P_{(x,y,t+1)}$  where  $O_x$  and  $O_y$  are the optical flow components in the  $x$  and  $y$  directions.

### 5.3.5 Pixel reconstruction

After finding the nearest neighbor for each patch, we minimize the energy function w.r.t  $\phi$ . All pixels in the occlusion  $\mathcal{H}$  are reconstructed using the following weighted average:

$$u(p) = \frac{\sum_{q \in N_p} s_q^p u(p + \phi(q))}{\sum_{q \in N_p} s_q^p},$$

In this expression,  $s_p^q$  is a weight defined as:

$$s_p^q = \exp\left(\frac{d^2(\psi_q, \psi_{q+\phi(q)})}{2\sigma_p^2}\right) \psi_p^q \varphi_p^q$$

where the first term is the original term in (Wexler, Shechtman, and Irani 2007) and Newson (Newson, Almansa, Fradet, et al. 2014), which is calculated using the similarity between two patches. The parameter  $\sigma_p$  is defined as the 75th percentile of all distances  $d(\psi_q; \psi_{q+\phi(q)}); q \in N_p$  as in (Wexler, Shechtman, and Irani 2007).

In our method, this weighting function is modified by adding 2 terms  $\psi_p^q$  and  $\varphi_p^q$ . The first term is a confidence map inspired by (Fedorov, Facciolo, and Arias 2015).

$$\psi_p^q = \begin{cases} (1 - C_0) \exp\left(-\frac{d(q, \sigma\mathcal{H})}{\sigma^2}\right) + C_0 & \text{if } q \in \mathcal{H} \\ 1 & \text{otherwise} \end{cases}$$

where  $d(q, \Omega)$  is the distance between pixel  $q$  and the border  $\sigma\mathcal{H}$  of the occlusion.  $C_0$  and  $\sigma^2$  are two tuning parameters. Together they control the “width” and “strength” of the confidence

map. Setting  $C_0 = 1$  will lead to a constant map. By this map, we put more confidence to the patches near the border of the occlusion. This map is used to guide the information from the border toward the center and therefore can help us eliminate some border artifacts.

The second term is a separation map defined as:

$$\varphi_p^q = \begin{cases} 1 & \text{if } p \text{ and } q \text{ are same type} \\ 0 & \text{if } p \text{ and } q \text{ are different type} \end{cases}$$

The two possible *types* for each pixel are background and foreground. As before, they are determined from the optical flow field using a simple adaptive thresholding method (Otsu 1979). The implication of this term is that when we reconstruct background (resp. foreground) pixels, we use only background (resp. foreground) patches. This is a simple way to avoid blending effects between background and foreground in the result.

## 5.4 Result

### 5.4.1 Implementation details

In this section, we present implementation details which are essential for achieving good results.

The first parameter we need to consider is the patch size. We use relatively small  $5 \times 5 \times 5$  patches in our synthesis. Different from some image completion algorithms which use larger patches e.g.  $11 \times 11$  in (He and J. Sun 2012b), we observe that in video inpainting, a larger patch size increases the run-time significantly while the visual quality does not substantially improve.

Another critical parameter is the weight of color, texture and motion information in the patch distance. As discussed in section 5.3.2, these weights have a strong impact on the final results, and the best values may depend on the input video. Automatic setting these parameters is a very challenge problem and is the purpose of ongoing work. As a base setting, the weights for color, textures, and motion are fixed to 1, 50, 50 respectively in our implementation. We may adjust these weight according to the video and our purpose. For instance, when we want to reconstruct moving objects, we should put more weights on the motion information. On the other hand, when dealing with complex texture, more weights for texture information should be applied.

Since our objective function is optimized in a multi-resolution pyramidal framework, a natural question arises: Should we use temporal sub-sampling and treat the temporal direction and the spatial direction similarly? Through experiments, we have realized that temporal sub-sampling would be helpful if the video is recorded with a high frame rate. In this case, the difference between frames is small, and we do not lose too much information at the coarsest level. However, if the video has a low frame rate, as mentioned in (Wexler, Shechtman, and

Irani 2007; Granados, Tompkin, et al. 2012; Newson, Almansa, Fradet, et al. 2014), temporal subsampling can be detrimental to inpainting results due to the difficulty of representing motion at coarsest levels. Since many videos in our experiment have a low frame rate (for memory efficiency and runtime complexity), we decided not to use temporal sub-sampling.

Another crucial choice when using a multi-resolution framework is the number of pyramid levels to use. As mentioned in (Newson, Almansa, Fradet, et al. 2014), the number of pyramid levels should be chosen such that at the coarsest level, the maximum width and height of the occlusion is less than two times the patch size. This strategy is not absolutely required in our case because we use priority initialization instead of onion peel. Nevertheless, we still use this rule of thumb since it is useful in many cases. More specifically, with a video of size  $854 \times 480$ , we use 4 levels of the pyramid, and with a video of size  $1280 \times 720$ , we use 5 levels.

To optimize our energy, we repeat 2 steps: parallel spatio-temporal PatchMatch and pixel reconstruction at a specific pyramid level. As a stopping criterion, we use the average color difference in each channel per pixel between iterations and set it to 0.1. Another stopping criterion is the number of repetition. We impose a maximum number of 10 iterations to avoid iterating for too long.

Considering the parallel spatio-temporal PatchMatch, as in the paper of (Barnes, Shechtman, et al. 2009), we use 10 iterations of propagation and random search during the ANN search. The window size reduction factor is set to 0.5. The grid size in the random search step is set to 1 at the coarsest level and is doubled when we go to the next level.

In the pixel reconstruction step, the "thickness" and "strength" of the confident map are other parameters to tune. We set the "thickness" of the confident map to 1 at the coarsest level and double it each time going to the next pyramid level. We fix the "strength" of the confident map to 0.5 for all our experiments.

For the optical flow computation, to obtain maximum speed, we apply FlowNetv2 (Ilg et al. 2017) - a state-of-the-art Convolutional Neural Network model for computing optical flow. This method can compute optical flow between two  $1280 \times 720$  images in less than 1 seconds while the accuracy outperforms other state-of-the-art methods.

### 5.4.2 Experimental results

We evaluate the performance of our approach under different forms of experiments: objects removal under static background, dynamic background restoration, and moving object reconstruction. In all experiments, there is no prior knowledge about the areas that should be inpainted. Hence, we only visually assess the performance of the method. In these 3 types of experiments, the dataset contains a wide variety of conditions, including occluded moving object, static and moving camera, dynamic background, large occlusion.

We compare with other state-of-the-art methods (Granados, Tompkin, et al. 2012; Newson, Almansa, Fradet, et al. 2014; J.-B. Huang et al. 2016; Ebdelli, Le Meur, and Guillemot 2015) using their publicly available datasets or code. We also propose several new challenging sequences



Figure 5.21 – A comparison of our inpainting result with the method of (Granados, Tompkin, et al. 2012) on GWAREdAS-S6 sequence. ON are able to achieve comparable results with (Granados, Tompkin, et al. 2012) without requiring the mask of the remaining foreground object.

and encourage other authors to evaluate their methods on our sequences. These videos are available at the supplemental website: <https://object-removal.telecom-paristech.fr>.

### Removing dynamically moving objects under static background

**Comparison with (Granados, K. I. Kim, et al. 2012)** In this first experiment, we test our method with the sequences provided by (Granados, K. I. Kim, et al. 2012) to remove a dynamic object from videos captured by a freely-moving camera. In each sequence, two or more people are moving in front of a static background. The mask of the object to be removed and the mask of the remaining foreground objects are provided by the authors. Our method does not require the mask of the object which remains in the scene. There are small viewpoint variations and narrow baselines around the object of interest. Figure 5.21 illustrates the visual results of the two algorithms. Since these examples are designed to fit best with the method of (Granados, Tompkin, et al. 2012), their visual results are almost perfect. In these cases, we obtain similar high quality with them. In particular, we notice that our method can well reconstruct high structural zone such as the stone in the ground (GRANADOS-S6 sequence) because our video stabilization step can compensate the camera movement very well in that case. One advantage of our method is that we do not require foreground/background segmentation to produce satisfactory results.

**Comparison with (J.-B. Huang et al. 2016)** The next experiment is to remove a dynamic object in a video recorded in the wild with a freely moving camera. The purpose is to test the method under challenging situations as well as to compare our results with those of (J.-B. Huang et al. 2016). The same dataset in (J.-B. Huang et al. 2016) called DAVIS-2016 - a recent benchmark dataset in object segmentation (Federico Perazzi, Jordi Pont-Tuset, et al. 2016) is used. It is made of very challenging sequences due to the dynamic scenes, the complex camera movements, the motion blur effects, and the large occlusions. The occlusion masks are constructed by dilating the ground truth using a  $15 \times 15$  element structure. We note that in this experiment, we do not aim to remove the shadow while in (J.-B. Huang et al.



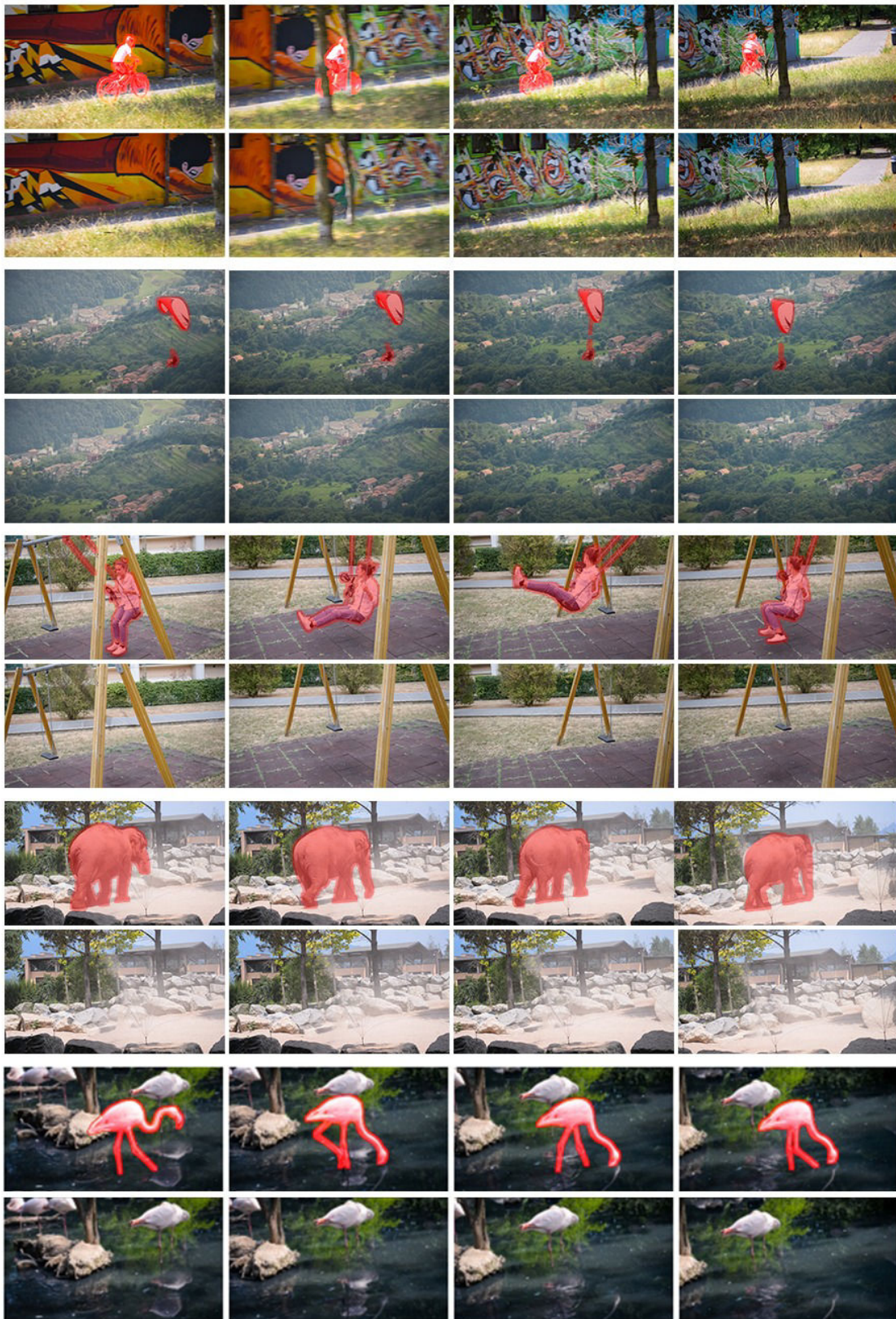


Figure 5.22 – Our results on DAVIS-2016 ([Federico Perazzi, Jordi Pont-Tuset, et al. 2016](#)) sequences. From top to bottom: BMX-TREES, PARAGLIDING, SWING, ELEPHANT, FLAMINGO. In each sequence, we provide the image + mask (top row) and our results (bottom row).



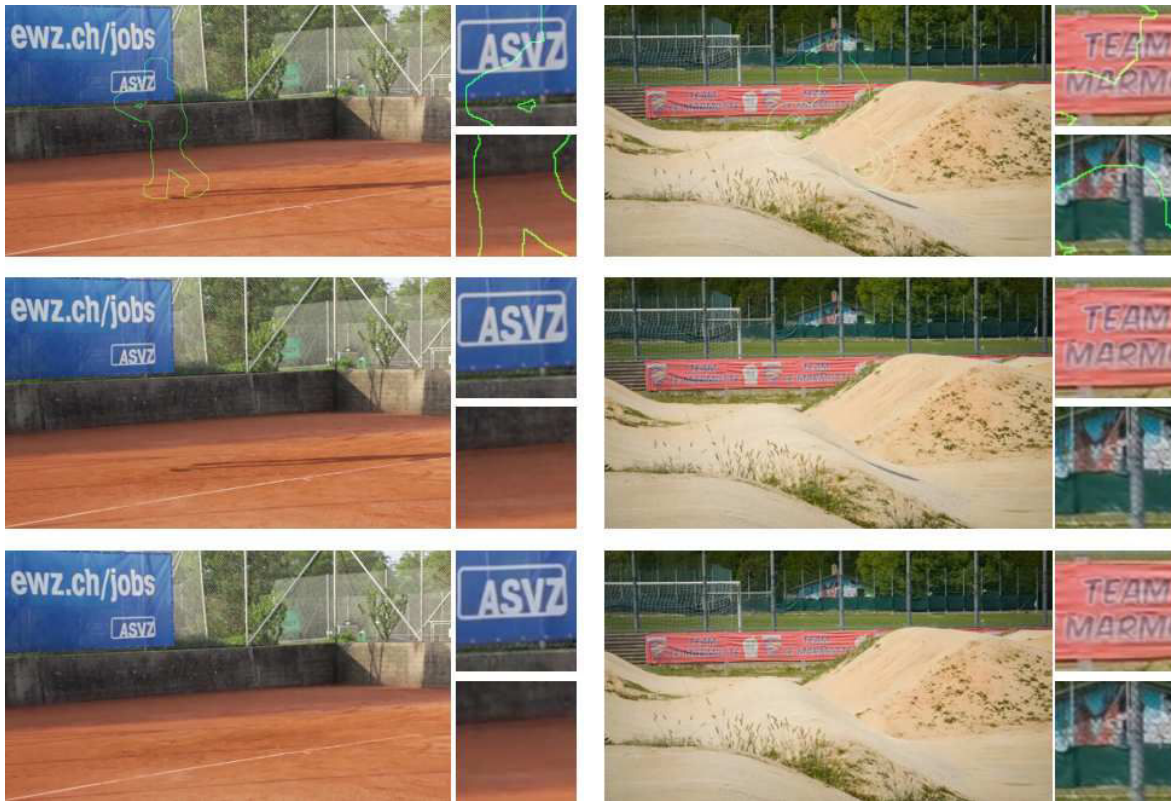


Figure 5.23 – Some representative frames of the results with sequence *TENNIS* (left) and *BMX-BUMP* (right). From top to bottom: our result with occlusion mask, our result without occlusion mask, result of (J.-B. Huang et al. 2016).

2016), the authors include the shadow in the mask by select it manually using the Rotobrush tool of AfterEffect.

Figure 5.22 shows sample frames from five input sequences. In each sequence, we show mask overlay (top row), and our completion results (bottom row). Results of the first three sequences *BMX-TREES*, *PARAGLIDING* and *SWING* demonstrate that our algorithm can deal with large camera movement. In these sequences, the displacements between frames are very large, and the scene is complicated with different types of textures (soil, trees) and structure (chain, swing). Our results can fill the occlusion regions seamlessly and give very plausible results. They can reconstruct the background with different planar planes (*SWING*), or under motion blur caused by a fast moving camera (*BMX-TREES*). In the *PARAGLIDING* sequence, even though there is a small ghost artifact in the border in the beginning of the video, the overall structure of the scene is preserved and the sequence looks plausible. In the next sequence *ELEPHANT*, the occlusion covers a big part of the video, and there is still some part of the background which is never revealed in the entire sequence. In this case, our algorithm can reconstruct some complicated textures such as soil, rock, and it can also reconstruct the geometric structures such as the tree, fence. The last sequence *FLAMINGO* is a very successful case where the geometric structure such as the flamingo’s leg and the textured zones (water, grass) can both be well reconstructed.

Next, we compare with (J.-B. Huang et al. 2016) visually. Figure 5.23 shows some

representative frames of the result. The whole video of (J.-B. Huang et al. 2016) method can be seen at the following address: <https://filebox.ece.vt.edu/~jbhuang/project/vidcomp/supp/index.html>. In these two sequences TENNIS and BMX-BUMPS, we achieve the same quality as (J.-B. Huang et al. 2016). In both methods, the spatial structure is well-preserved under very large camera displacement.

In general, in DAVIS-2016 dataset (Federico Perazzi, Jordi Pont-Tuset, et al. 2016) where the frame rate of the sequence is low and the displacement of the camera is huge, our results lack temporal coherence compare to that of (J.-B. Huang et al. 2016). The reason is that the spatio-temporal patches in our method have difficulties to capture information of the background with large displacement. On the other hand, (J.-B. Huang et al. 2016) only use 2D patches, and use optical flow propagation which is recalculated after each iteration to maintain temporal coherence. Therefore, they can attain strong temporal consistency if the background is static. However, by using only 2D patches, the quality of the temporal coherency depends absolutely on the accuracy of the optical flow computation which can not be achieved in some complicated sequences. Besides, since the optical flow field must be recomputed in each iteration, their method is slower than ours. While (J.-B. Huang et al. 2016) took about 3 hours to complete one video of size  $854 \times 480 \times 90$  with  $\approx 20\%$  of occlusion, our method took about 50 minutes to finish.

Although the method of (J.-B. Huang et al. 2016) can achieve good performance with a static background, as shown in Chapter 5, similar results can be attained more rapidly by using optical flow more directly without recalculating optical flow at each iteration. On the other hand, we will show in the following experiments that the method from (J.-B. Huang et al. 2016) fails when dealing with dynamic background or moving objects. In such cases, our method yields much better results.

### Dynamic background reconstruction

In this experiment, the purpose is to reconstruct the background with dynamic textures. Beside some sequences in DAVIS 2016 dataset such as BREAKDANCE, DRIFT-CHICANE or BLACKSWAN, we also introduce some additional video recorded by ourselves. These sequences capture several natural dynamic textures such as waves, water, crowd, fire, fountain.

We qualitatively compare our method with (J.-B. Huang et al. 2016). We use the code released by the author and tested it on the image sequences using the default parameters provided by the authors. In general, (J.-B. Huang et al. 2016) fails to generate convincing dynamic textures. This is because their algorithm relies on dense flow fields to guide the completion. In the case of dynamic textures, this optical flow is unreliable. Moreover, they fill the hole by sampling only 2D patches from source regions; therefore, the periodic repetition of the background is not captured. Our method, on the other hand, using the combination of spatio-temporal patches and optical flow to capture the dynamic information, can fill the missing dynamic textures regions with convincing contents.



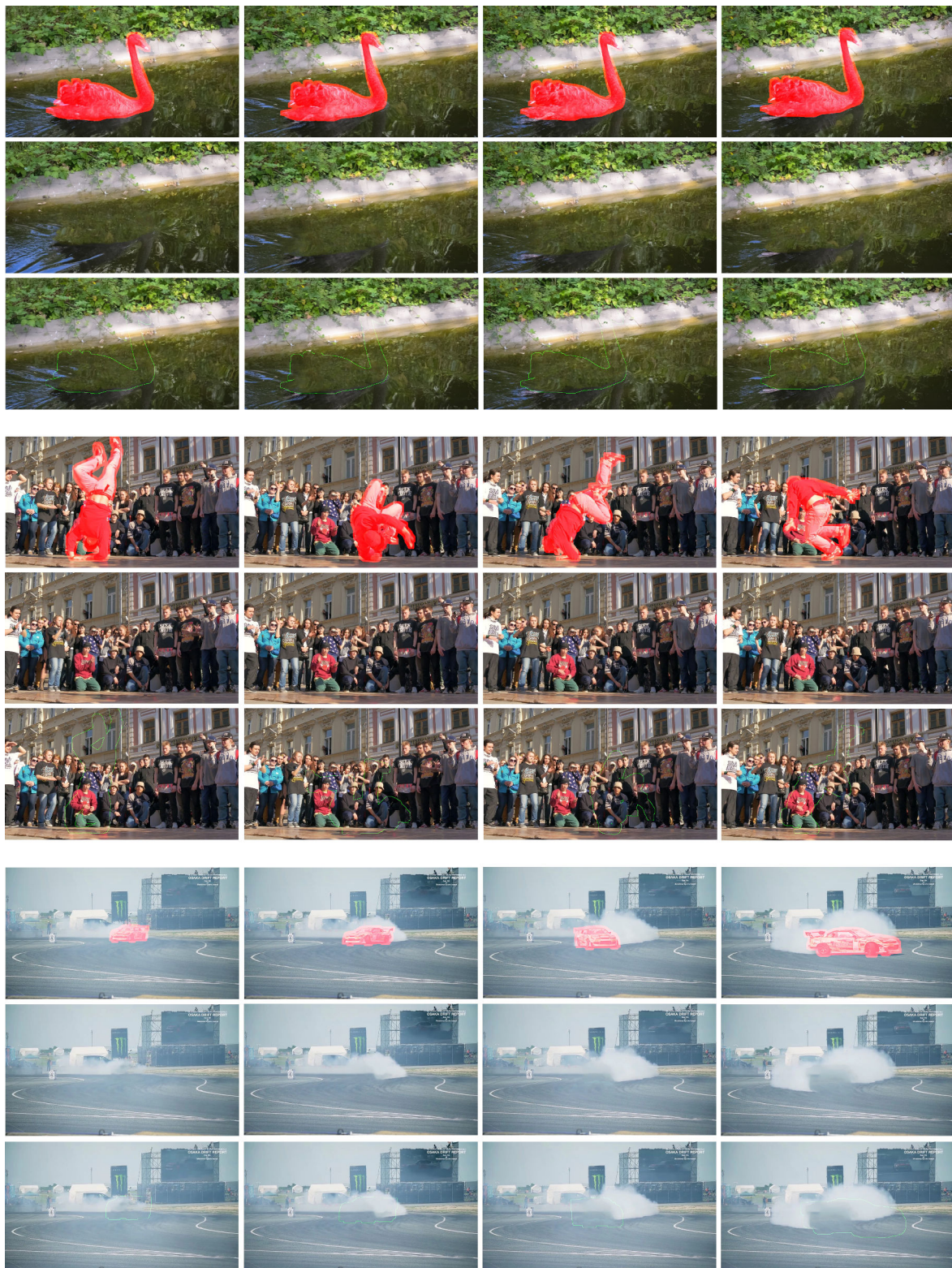


Figure 5.24 – Our visual results on dynamic background completion using 3 sequences in DAVIS-2016 dataset. From top to bottom: BLACKSWAN, BREACKDANCE, DRIFT-CHICANE. In each sequence, we provide the image + mask (top row), our video completion result (middle row), our result with the border of the occlusion in gr (botteonom row).

Figure 5.24 shows the representative frames from 3 sequences in DAVIS-2016 dataset and our completion results. The full video of our method and the results of (J.-B. Huang et al. 2016) can be viewed in the supplemental website. In these sequences, while (J.-B. Huang et al. 2016) provides a very displeasing artifact, e.g. the hand of the red man is "stick", the water is "glued". In other words, the repetition pattern of the dynamic background is not captured. Our method, on the other hand, can reproduce these dynamic textures in a very plausible way.

Next, we introduce two challenging sequences for dynamic texture: FOUNTAIN-BARCELONA (taken from Youtube) and FIRE-TEDDY-BEAR (recorded by our elves). We show visual comparison of our result and the one of (J.-B. Huang et al. 2016) on Figure 5.25. In the first FOUNTAIN-BARCELONA sequence, a fountain is recorded at night time. In this sequence, the background has a strong dynamic texture, and the camera is moving. The occlusion has a fixed position for all frames. In that sequence, as we can see, the results provided by our method are almost perfect while (J.-B. Huang et al. 2016) has some difficulty in reconstructing the fountain at the end of the sequence. The later FIRE-TEDDY-BEAR sequence is recorded indoor with the purposed of removing a teddy bear which is passing in front of a fireplace. Although the video is almost stable, this sequence is still challenging because of the illumination changes and the presence of both dynamic and static textures. Using the method from (J.-B. Huang et al. 2016), some artifacts are produced. In contrast, our method can reconstruct the fire perfectly. This sequence highlights the advantage of our method in recreating dynamic textures.

### Moving objects reconstruction

This experiment evaluates performances in the context of the reconstruction of moving objects. In this experiment, we aim at reconstructing moving objects which are occluded for a long time. We consider videos in which a moving object crosses a fixed or a moving occlusion. Such an object can be partly or even completely occluded (sequence jumping girl). Classical videos in (Newson, Almansa, Fradet, et al. 2014; Granados, Tompkin, et al. 2012) are tested with our method. We also introduce some new challenging videos and encourage others to try them. We compare our results with different state-of-the-art methods: (Wexler, Shechtman, and Irani 2007), (Granados, Tompkin, et al. 2012), (Newson, Almansa, Fradet, et al. 2014), and (J.-B. Huang et al. 2016).

**Comparison with (Wexler, Shechtman, and Irani 2007) and (Granados, Tompkin, et al. 2012).** In the beginning, we compare our method with some classical video inpainting methods from (Wexler, Shechtman, and Irani 2007) and (Granados, Tompkin, et al. 2012). Since the implementation provided by (Wexler, Shechtman, and Irani 2007) is obsolete and the code of (Granados, Tompkin, et al. 2012) is not publicly available, we use their publicly available sequences to provide visual comparisons. We note that the previous method is almost perfect and there is little room for improvements. We select several typical videos for



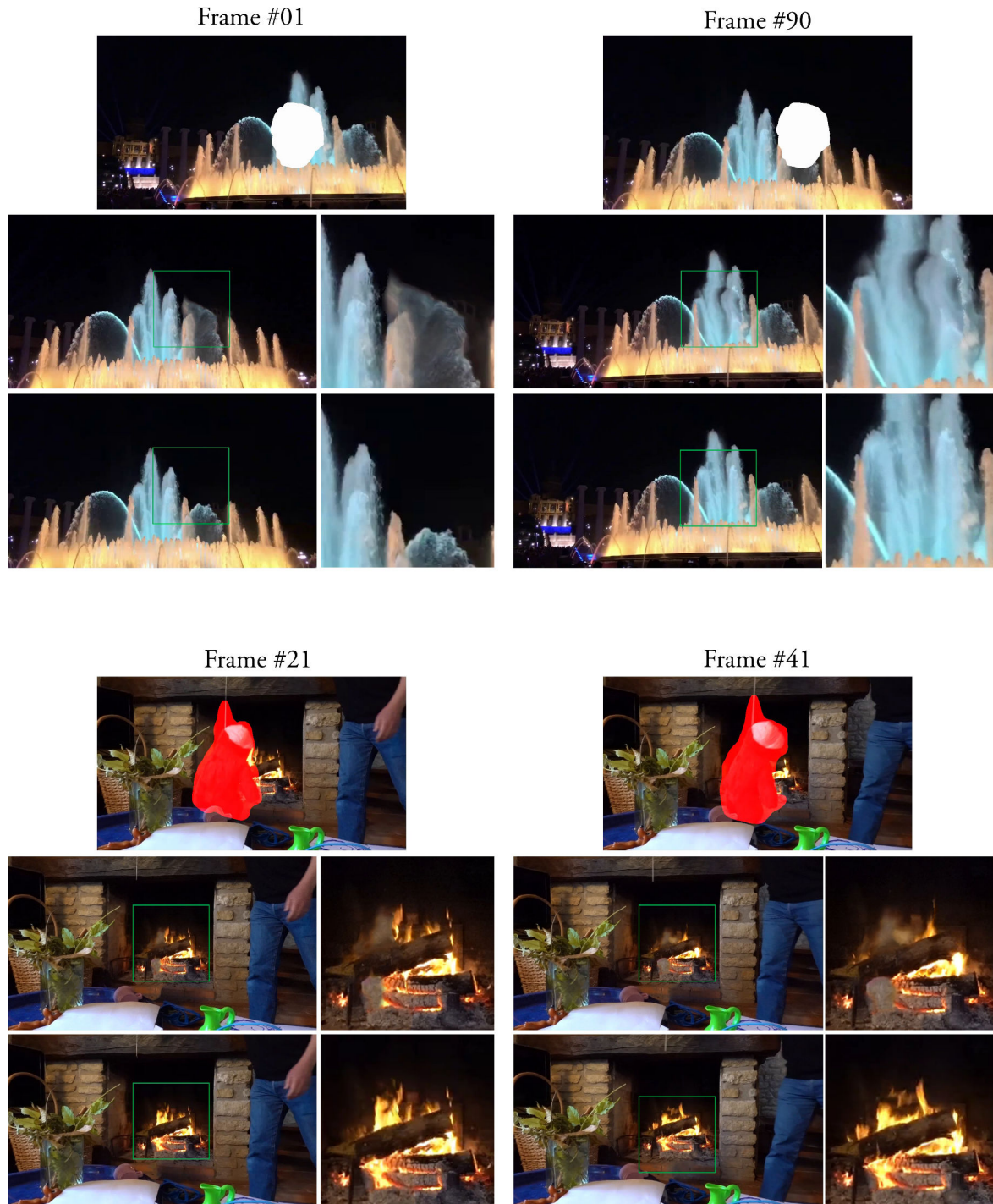


Figure 5.25 – Visual comparison of our method and the method of (J.-B. Huang et al. 2016) on two sequences: FOUNTAIN-BARCELONA (top) and FIRE-TEDDY-BEAR (bottom). In each sequence, we show results on several representatives frame including the image + mask (top row), result provided by (J.-B. Huang et al. 2016) (middle row), our result (bottom row). We also zoom into the occlusion (green box) to see the differences.



Figure 5.26 – Visual comparison with (Wexler, Shechtman, and Irani 2007) using their 2 publicly available sequences: BEACH-UMBRELLA (top row), JUMPING GIRL (bottom row). In each sequence, we show the original frame, result of (Wexler, Shechtman, and Irani 2007), and our result respectively from left to right.

each method to test our algorithm. More specially, we use two sequences JUMPING-GIRL and BEACH-UMBRELLA for comparing with (Wexler, Shechtman, and Irani 2007); and two sequences DUO, MUSEUM for comparing with (Granados, Tompkin, et al. 2012).

In the sequences from (Wexler, Shechtman, and Irani 2007), all sequences have low resolution, and the background is quite simple (beach, grass). The scenes consist of a static (umbrella) or nearly static object (waving girl) to be removed and a moving object with a periodic motion to be reconstructed. Figure 5.26 shows the visual comparison of our method with the one of 5.26. It can be seen that our results have similar quality compared to (Wexler, Shechtman, and Irani 2007). Both methods can plausibly reconstruct the moving object.

To compare with (Granados, Tompkin, et al. 2012), we use two sequences: DUO, MUSEUM from the dataset of (Granados, Tompkin, et al. 2012). This dataset is made of different videos recorded indoor or outdoor. In all the sequences, the camera is still, and the background is dynamic with several moving objects to be reconstructed. The purpose is to remove an undesired pedestrian out of the video. The occlusion is manually selected by the author and is publicly available. Some representative frames of the result are exposed in Figure 5.27. In the sequence DUO, two pedestrians that pass in front of two music performers standing in front of a reflective surface are removed. This is challenging because the guitarist exhibit repetitive hand movements which must be reconstructed. In the background, there is also the reflections of other moving objects which are occluded by the two pedestrians. As shown in Figure 5.27 (left), our method produces a plausible result and successfully completes the dynamic foreground and background scene elements.

The MUSEUM sequence (Figure 5.27 right) is made of a pedestrian walking in front of a crowd. It is another challenging sequence because people have different types of motions and their appearance are very similar. With this sequence, although we can notice some small defaults such as the distortion of the man’s head at the end, the overall quality still is very good. Our method produces satisfactory completions for the large occlusions. Furthermore, the algorithm can complete the reflections successfully on the floor. These results are comparable





Figure 5.27 – Visual comparison with (Granados, Tompkin, et al. 2012) using their publicly available sequences. Left: DUO, right: MUSEUM

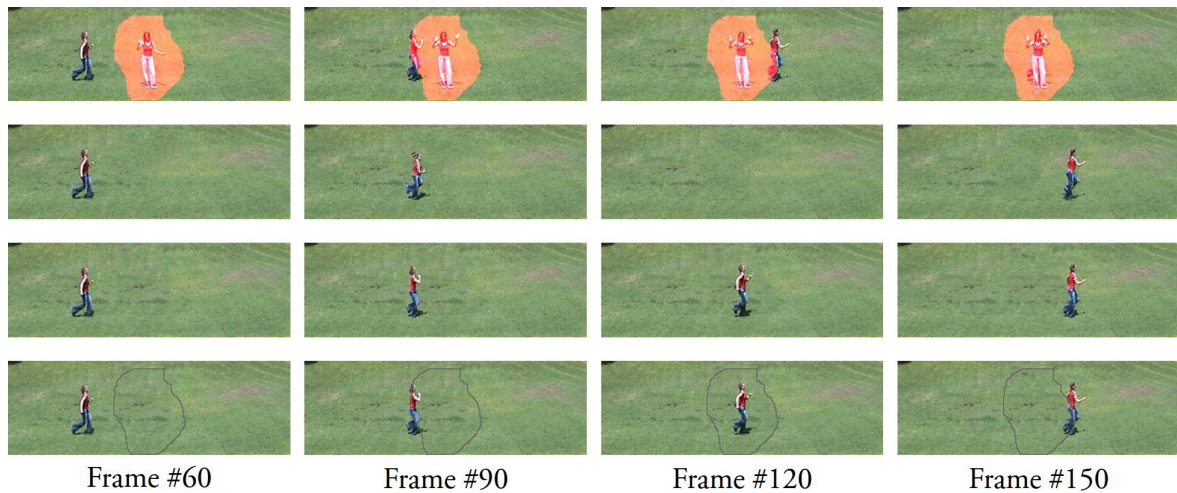


Figure 5.28 – Visual comparison with (Newson, Almansa, Fradet, et al. 2014) on JUMPING-GIRL sequence from (Wexler, Shechtman, and Irani 2007). We modify the occlusion to make it more difficult. From top to bottom: original images + masks, result by (Newson, Almansa, Fradet, et al. 2014), our result, our result with border.

with (Granados, Tompkin, et al. 2012) while our method is several orders of magnitude faster than (Granados, Tompkin, et al. 2012), which is a significant advantage of the algorithm.

**Comparison with (Newson, Almansa, Fradet, et al. 2014)** Next, we compare our result with the one of (Newson, Almansa, Fradet, et al. 2014) which was the starting point of our algorithm. To do that, we use the publicly available code provided by the authors and test it on different sequences. These sequences include both classical videos and new challenging videos.

For classical videos, we take the sequence JUMPING-GIRL of (Wexler, Shechtman, and Irani 2007). We modify the occlusion by enlarging it to make it more difficult. Figure 5.28 shows the occlusion and the inpainted video for both methods. It can be seen from this figure that the method of (Newson, Almansa, Fradet, et al. 2014) fails in that case. In their result, the jumping girl is faded, and the algorithm only reconstructs the grass. The main reason is that the occlusion is too large, thus at coarsest level, the onion peel cannot rebuild the motion and therefore it cannot be reconstructed further. Our method, on the other hand, uses the optical flow-driven initialization at the coarsest level. This type of initialization allows the moving object (jumping girl) to be reconstructed at the coarsest scale. When going to the next finer scale, this moving object is kept because we add an optical flow-based term on the distance. This term enables the searching algorithm to find the moving patches in the source region. As a result, the jumping girl can be reconstructed in a very plausible way.

Another classical sequence to compare our method with the one of (Newson, Almansa, Fradet, et al. 2014) is PARK-COMPLEX, taken from (Granados, Tompkin, et al. 2012). This video contains a person occluding several other people, each displaying different behaviors. We modify this video by focusing on the moment where the man we want to remove occludes





Figure 5.29 – Visual comparison with (Newson, Almansa, Fradet, et al. 2014) on moving objects reconstruction using 2 sequences: MUSEUM (left) and PARK-COMPLEX (right). In each sequence, we provide original image + mask, result of (Newson, Almansa, Fradet, et al. 2014), our result, our result with green border, from top to bottom respectively. We also zoom into the occlusion to see the differences.



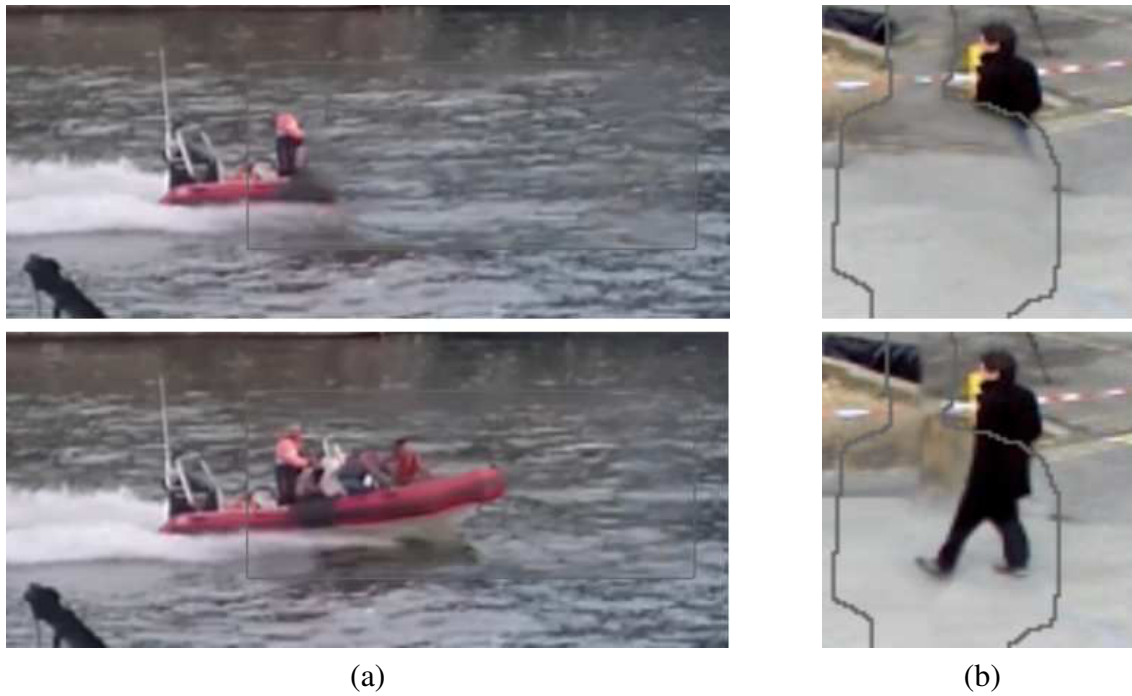


Figure 5.30 – Some sample cropped frames of results in some video sequences: (a) BOAT, (b) S2L1, (c) LES LOULOUS. Top: result of (Newson, Almansa, Fradet, et al. 2014), bottom: our result.

another man walking in the opposite direction. The later man is being occluded by a static obstacle (a bench) at the same time. In this example, (Newson, Almansa, Fradet, et al. 2014) cannot reconstruct the moving man being occluded because the background behind this object is too complicated. It changes over time (from tree to wall). Because the method of (Newson, Almansa, Fradet, et al. 2014) treats the background and the foreground similarly, it can not rebuild the situation "man in front of the wall" because it never sees this situation in the entire video. Thanks to the separation map, and the uses of optical flow, our method can reconstruct the "man" and the "wall" independently. Thus, it can reproduce the man. This example again highlight the importance of the optical flow in our method.

A similar situation can also be seen in the MUSEUM sequence (Granados, Tompkin, et al. 2012). In this sequences, the method of (Newson, Almansa, Fradet, et al. 2014) can not reconstruct the head of the man when it is occluded by the moving occlusion while our method is capable of doing it.

We then introduce several new challenging examples. To focus on the capacity of reconstructing moving objects, we utilize videos recorded with a static camera. These videos contain common objects such as boats, pedestrians, cycles, cars... Then we synthesize the occlusion in different forms: static rectangle, moving rectangle and arbitrary static shape. For example, in the sequences BOAT, we employ a moving box as the moving occlusion. This rectangle occluded both the moving object (red boat) and the dynamic texture (water). Another example is the video S2L1, taken from the MOT tracking challenge. For this sequence, we manually annotate the occlusion (the tripod and the light), and we want to reconstruct moving

pedestrians which are occluded by this occlusion.

These two sequences are very challenging. In BOAT sequence, we must reconstruct both the moving object and the dynamic textures under moving occlusion in a coherence manner. In contrast, even S2L1 has a static background, the occlusion in this sequence is fixed (no information behind the occlusion is available), and there are many moving objects with similar appearances crossing the occlusion.

Figure 5.30 highlight the capacity of our method to reconstruct dynamic background. From this figure, we observe that our result outperforms the one in (Newson, Almansa, Fradet, et al. 2014). Figure (a) 5.30 shows that (Newson, Almansa, Fradet, et al. 2014) cannot reconstruct the foreground object (the boat) even though in this sequence, the foreground are not completely occluded and some of its parts are still available. This can be explained by the fact that onion peel initialization will propagate the information of the background from the border toward the center. Since background and foreground are treated similarly, some part of the moving object cannot be recovered. Our method, on the other hand, can completely reconstruct the background and foreground in a plausible manner. This proves the importance of the additional optical flow term in our method. In Figure 5.30 (b), as explained before, the method of (Newson, Almansa, Fradet, et al. 2014) can not reconstruct the moving objects while our can. In this case, the optical flow also helps us to pick the right source to copy from since there are many objects with similar appearances in this video.

**Compare with (J.-B. Huang et al. 2016)** Lastly, we compare our results with (J.-B. Huang et al. 2016). We pick the sequence LES LOULOUS from (Newson, Almansa, Fradet, et al. 2014) and sequence "CAR-RACE" introduced by us. Some representative frames of the result of LES LOULOUS and CAR-RACE sequences are shown in Figure 5.31 and Figure 5.32 respectively. As noted in (J.-B. Huang et al. 2016), the incorrect synthesis of optical flow may lead to several displeasing artifacts. This is reported in (J.-B. Huang et al. 2016) with LES LOULOUS sequence while our method provides a plausible result with this sequence. A similar situation happens in the CAR-RACE sequence where two cars can not be reconstructed when they enter the occlusion.

### 5.4.3 Contribution of each component

In this section, we provide ablation study and investigate the contribution of each component in our method.

#### Video stabilization as a pre-processing step

Video stabilization is a essential step to compensate for the patch deformation caused by camera movements. It guarantees that the searched for spatio-temporal patches in the source region will have the same structure and motion. On the other hand, this steps also reduces

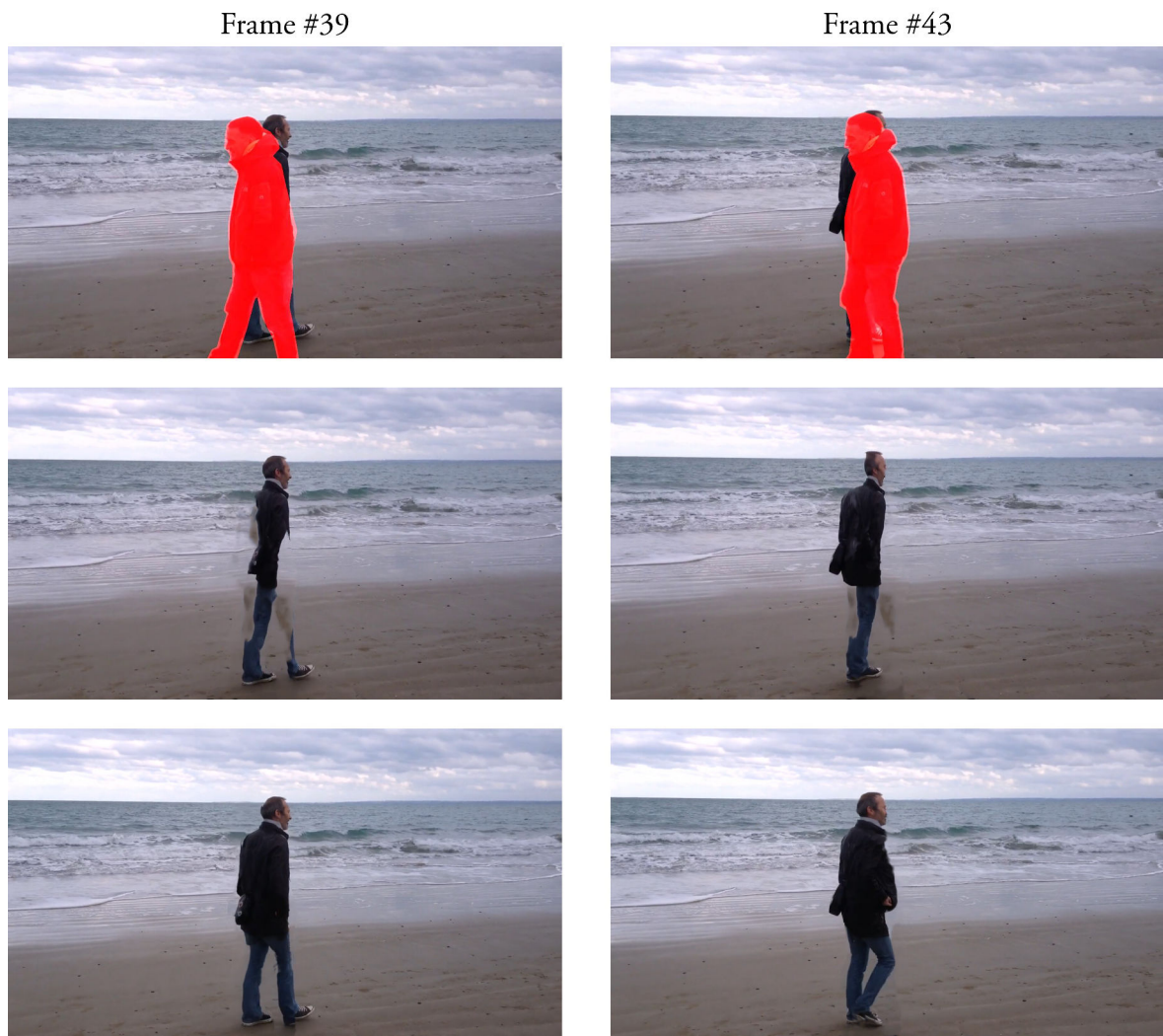


Figure 5.31 – Visual comparison with (J.-B. Huang et al. 2016) on reconstructing moving object with LES LOULOUS sequence. From top to bottom: original images + masks, result of (J.-B. Huang et al. 2016), our result.

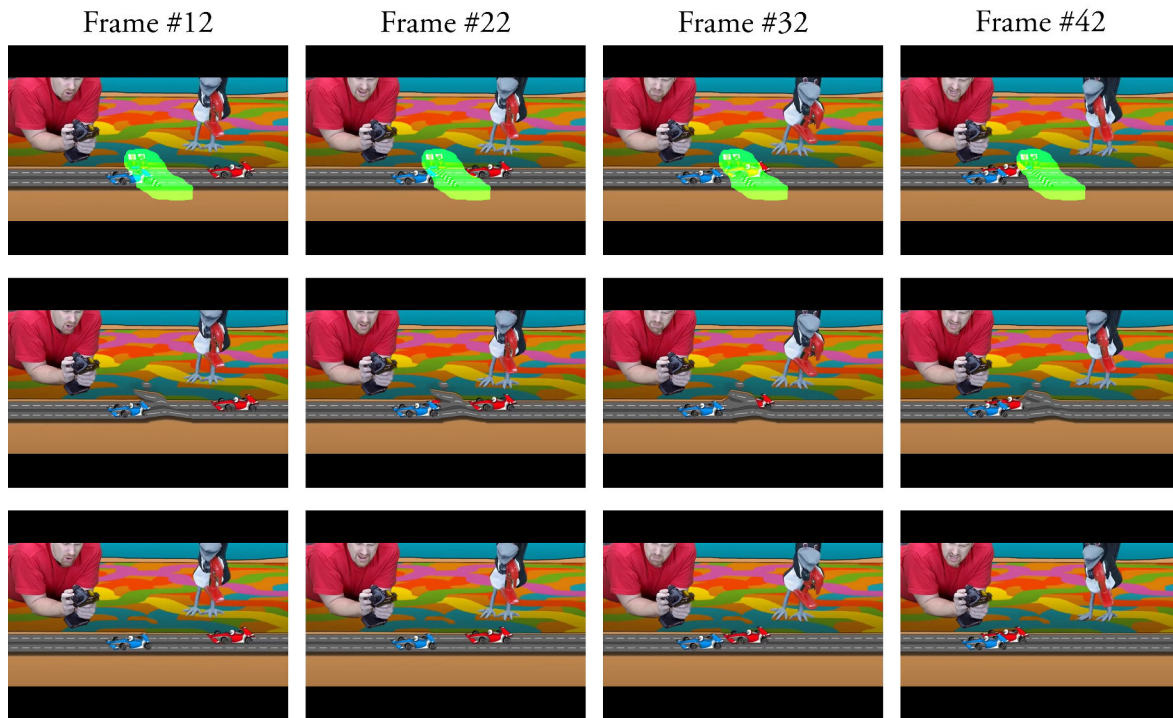


Figure 5.32 – Visual comparison with (J.-B. Huang et al. 2016) on reconstructing moving object with the CAR-RACE sequence. From top to bottom: original images + masks, result of (J.-B. Huang et al. 2016), our result.

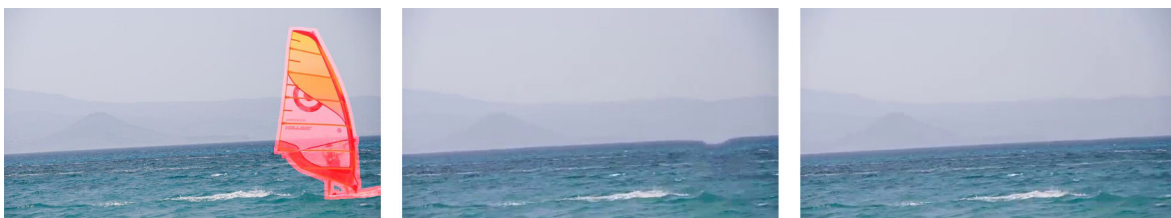


Figure 5.33 – A comparison of inpainting results with and without video stabilization. The reconstructed video have less coherence in both spatial and temporal direction. From left to right: image + mask, result without video stabilization, result with video stabilization.



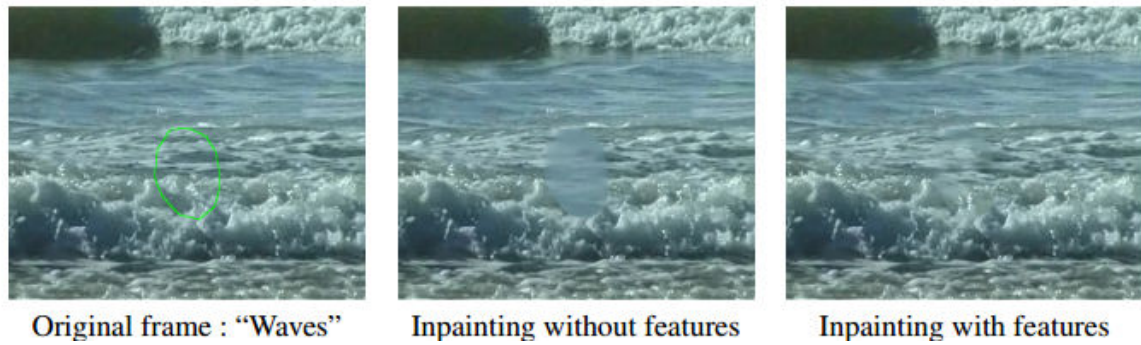


Figure 5.34 – Comparison of inpainting results with and without texture features. Without the features, the algorithm fails to recreate the waves correctly. Figure taken from (Newson, Almansa, Fradet, et al. 2014).

large displacements and make the optical flow estimation more reliable. Figure 5.33 compares the results with and without this pre-processing step. Without it, it is not possible to find coherent patches which respect the border conditions.

### Texture in videos

As indicated in (Newson, Almansa, Fradet, et al. 2014), reconstructing textured regions in video is challenging for several reasons. Firstly, the multiresolution pyramids can make patch comparisons in textured regions ambiguous, because texture information does not exist at coarser levels, leading to ambiguities when comparing textured and smooth patches. Secondly, the use of  $L_2$  SSD patch distance is not appropriate for comparing two textured patches. Very oscillating patches can easily be matched with very smooth patches. Thirdly, the PatchMatch algorithm itself can contribute to the identification of incorrect patches, because their principle of constant nearest neighbor offset  $\phi$  is less appropriate in the case of stochastic textures.

To solve this problem, in (Newson, Almansa, Fradet, et al. 2014), the authors add an additional term to the patch distance, which is the magnitude of the gradient in  $x$  and  $y$  direction. We realize that this is a simple and efficient solution to deal with textured regions. An example of the impact of the texture features in video inpainting may be seen in Figure 5.34

### Optical flow-driven initialization versus onion peel initialization

Since our method rely on the optimization of a non-convex and highly dimensional energy function in a multi-pyramidal approach, a good initialization at coarsest level is necessary to obtain a good result. As already explained, the method of (Newson, Almansa, Fradet, et al. 2014) inpaints the occlusion following onion peel order at coarsest level. This type of initialization have difficulty reconstructing moving object because it treats background and foreground similarly. Our method, on the other hand, use optical flow driven initialization scheme and put the more priority in recreating the moving object. As a result, the moving



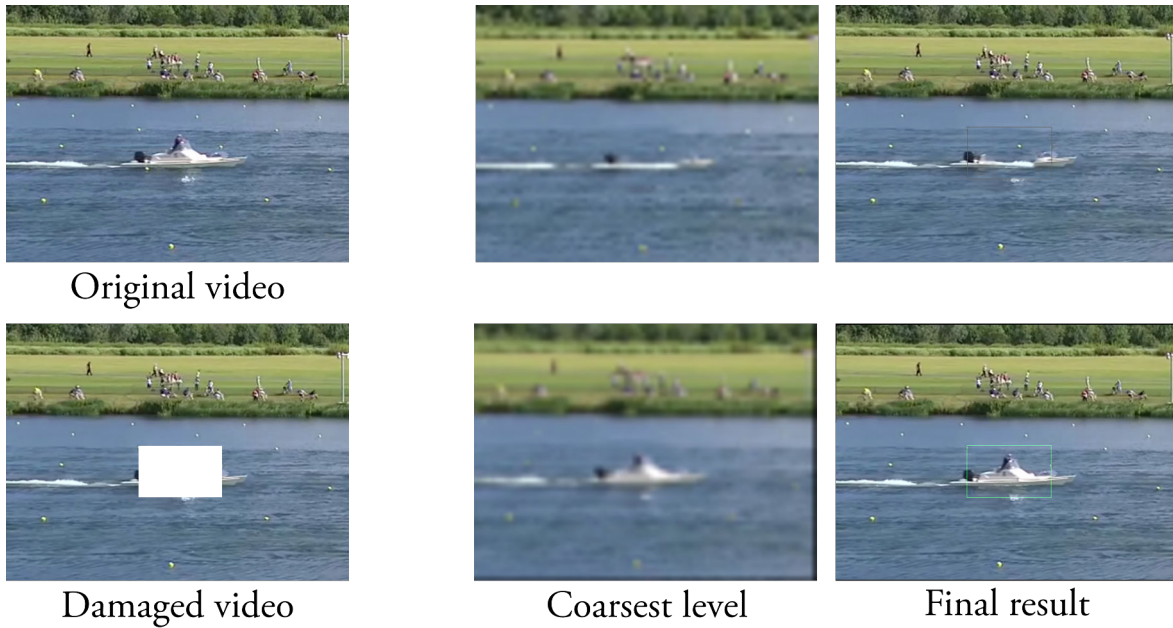


Figure 5.35 – Onion peel initialization versus optical flow-driven initialization. Using onion peel (top), we can not reconstruct the boat at the coarsest level. Hence, the final result lacks accuracy because the boat is not reconstructed. With optical flow-driven initialization (bottom row), we can reconstruct the motion at the coarsest level, so the algorithm can fully reconstruct the boat in the final result.

object is reconstructed as the coarsest level. Figure 5.35 shows some evidence to support our careful initialization scheme. As can be seen from this figure, when the moving object (the boat) enter the occlusion, while onion peel initialization is unable to reconstruct the object, our method can fully recreate it in a plausible way.

### The importance of the optical flow term in the distance metric

Inspire by the texture term introduce by (Newson, Almansa, Fradet, et al. 2014), we also add a new term to the patch distance, which is the magnitude of optical flow in  $x$  and  $y$  direction. The purpose of this term is to support the nearest neighbor search algorithm in searching for the appropriate patches. In Figure 5.36, we show the results with and without using the optical flow term in the distance. We start with the same optical-flow driven initialization solution where the moving object (the man) is reconstructed. When we do not use the motion term in the distance, after a few iterations, this moving object fade away and the algorithm prefers to reconstruct the background since it creates a more stable solution. This situation does not happen when we add the optical flow term to the patch distance. In that way, we can keep this motion on finer pyramid levels.

**Confidence map for pixel reconstruction** Next, we evaluate the advantage of using confidence map for pixel reconstruction. Figure 5.37 shows the comparison between the results with and without using the confidence map. As can be seen, border artifact is reduced

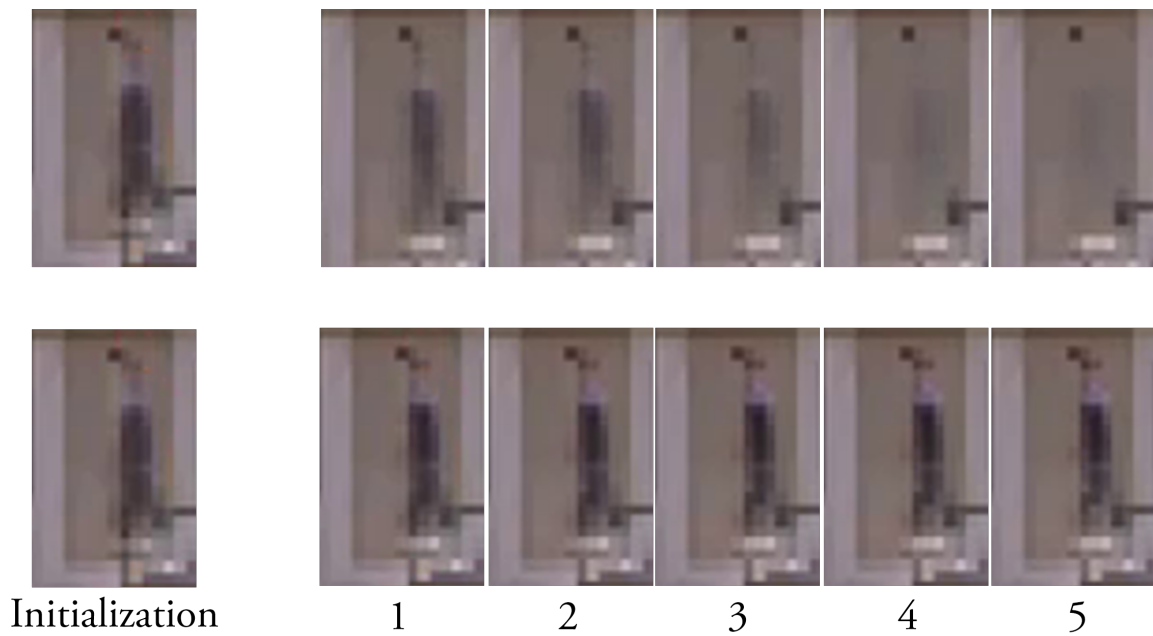


Figure 5.36 – The effect of adding the optical flow term in the distance. In the initialization solution, the man is reconstructed, but without using the optical term, the man faded after a few iterations (top row). This situation does not happen when we add the optical flow term in the distance (bottom row).

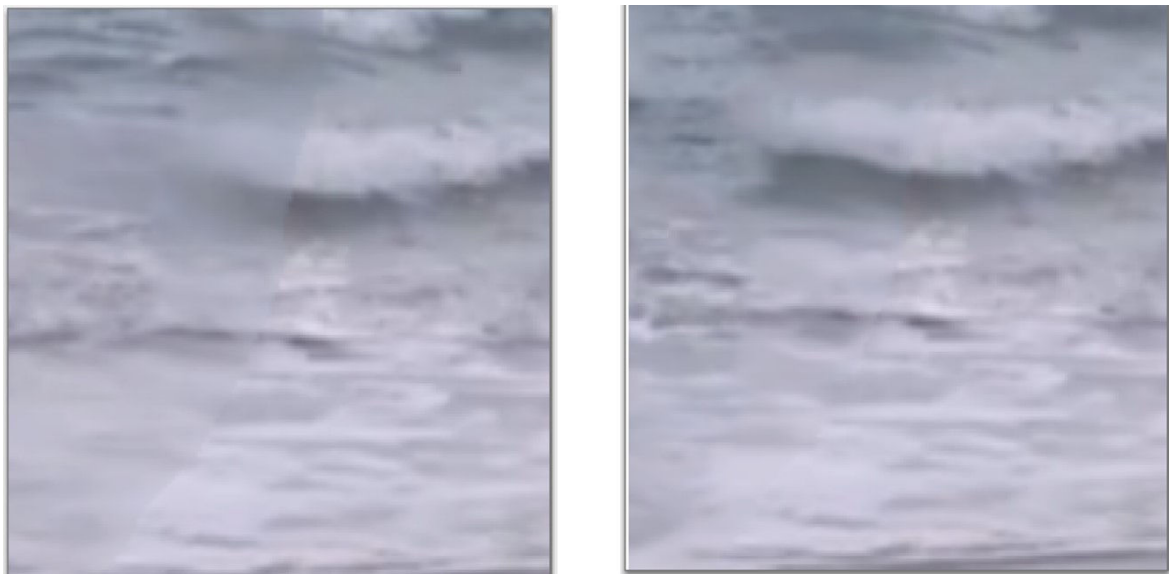


Figure 5.37 – Comparison of results with and without confidence map for the the pixel reconstruction step. With the confidence map, the border artifact is less visible.



Figure 5.38 – Our method has difficulties to deal with regions which are occluded over the entire sequences. Spatial and temporal coherences in these regions are weak.

when we apply the confidence map. This is straightforward because by putting more weight to patches near the border of the occlusion, we encourage the algorithm to produce results which have more spatial coherence near the border.

## 5.5 Limitations

While our patch-based video inpainting approach is efficient and flexible, several problems still persist. Firstly, the computational complexity and the memory requirement is high. Since we use 3D spatio-temporal patches, the complexity of the algorithm is high and it increases rapidly when we enlarge the patch size. Even with the parallel version and the other improvements that have been made, it still takes us  $\approx 50$  minutes to inpaint a  $854 \times 480 \times 90$  sequences with  $\approx 20\%$  of occlusion. The memory requirement of our method is also high because we treat the hole video sequences as a tube, and perform patch comparison globally.

Secondly, the spatial and temporal coherence is weak in the region where the damage regions occlude the whole sequences. In this case, the algorithm must invent the information in these regions, which may lead to spatial incoherence. Since our process a global, each frame may invent different solution and the temporal consistency is short because it is only kept by the 3D patches. This may lead to some oscillation effects. A typical example is the RHINO sequence, which is illustrated in Figure 5.38. One solution could be to increase the patch size in the temporal direction, but as discussed previously, this increases the computational complexity drastically. Moreover, increasing the patch size in the temporal direction will make it harder to find the best patch. Another solution could be that when looking for the nearest neighbor for a patch  $\psi_p$  at  $(x, y, t)$ , we put more weights to patches near the spatial coordinate  $(x, y)$  of  $p$ . In fact we tried this method and found that a strong blur artifact appear.

Thirdly, our method performs poorly with video having low frame rate and large pixel displacements. This is natural because the 3D patches cannot capture the large displacements of the pixel. This is usually solved by the video stabilization step. However, if the stabilization



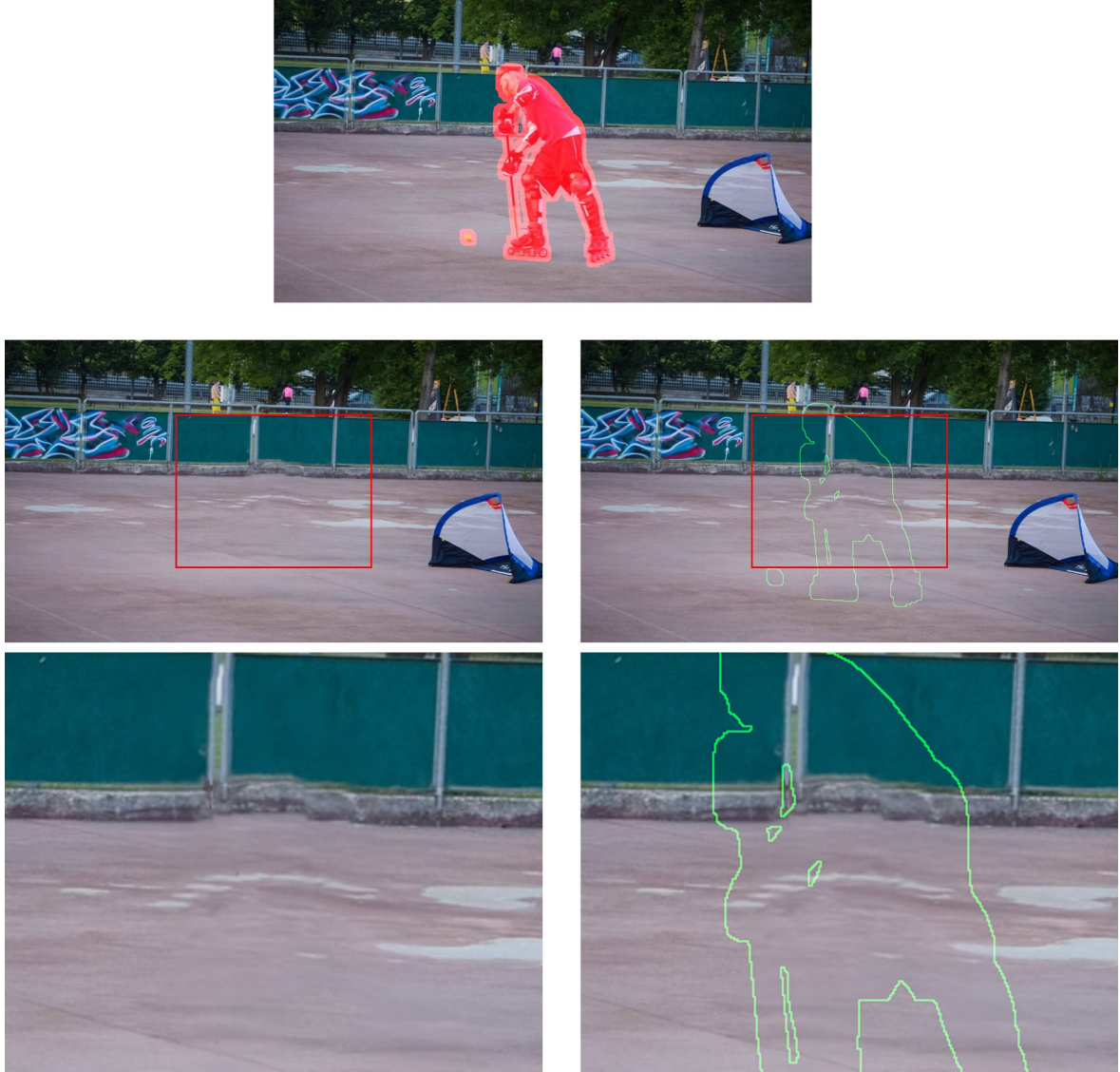


Figure 5.39 – Our method sometime failed to reconstruct geometric structure due to errors in video stabilization. Top: image + mask, middle inpainting result + border, bottom: zoomed image of region in the red box.

is not perfect, e.g. when the video contains different planar plane, our method provides poor result (Figure 5.39).

Lastly, we still can encounter some undesired artifact such as ghost effects and blur due to the spatial inconsistency and the weighted average of patches. This happens at times for video containing highly textured regions (see Figure 5.40).

## 5.6 Conclusion and Future Works

In this chapter, we have presented a robust video inpainting method that can handle challenging scenarios such as dynamic background or moving objects. We solve the video inpainting problem by optimizing a global patch-based energy function. Our main contribution is the improvement of the previous work of (Newson, Almansa, Fradet, et al. 2014) in terms of both

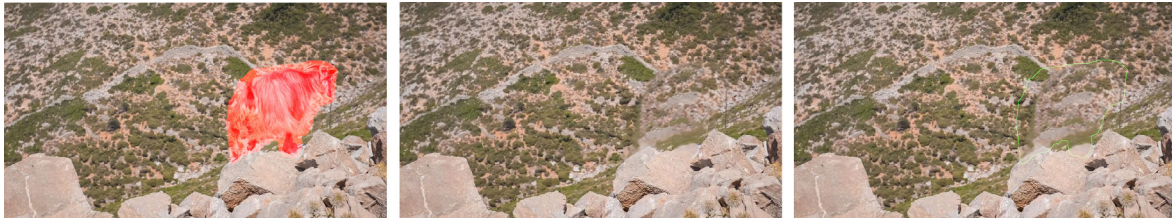


Figure 5.40 – In some sequences, ghost and blur artifacts still exist. From left to right: image + mask, inpainting result, inpainting result + border.

accuracy and computation time, which is achieved thanks to the integration of optical flow and the parallelization of the algorithm. Firstly, we observed that reconstructing moving objects damaged by the occlusion poses a significant problem due to the incorrect identification of source patches and the lack of motion at the coarsest level. We solve this problem by systematically incorporating the optical flow in the overall process: in the patch distance, patch shape, initialization, and nearest neighbor search. As a result, our method can reconstruct moving objects as well as reproducing dynamic textures with high temporal consistency. Secondly, we addressed the problem of speed in the ANN search by proposing a parallel extension of the 3D PatchMatch algorithm with several modifications. By this way, we can speed up the source patch searching process by 5-7 times while preserving the same accuracy level. Experimental results on various sequences show that our approach significantly extends the capacity of the previous video completion algorithm on videos containing multiple moving objects and with dynamic scenes. This indicates that the performance of the proposed method is comparable with state-of-the-art methods in the case of objects removal with a static background and significantly better in the case of dynamic background.

In the future, several extensions of our methods should be considered. The first point is the automatic selection of the weights for spatial gradient and optical flow term. While it is finetuned per-video in the current version, a further study on how to use adaptive weights for these components can improve the robustness of the algorithm. The second point which is difficult to handle is the case where a static background region is occluded for the entire sequence. Since the problem is solved easily with a simple motion-guided propagation as mentioned in Chapter 4, the natural question arising is: how to combine these two methods in a unified manner. One possibility could be to use the result of motion-guided propagation method as the initialization for the patch-based algorithm. However, this would slow down the reconstruction in uncomplicated cases. Combining these two methods efficiently is an open question and should be investigated further.



# 6

## A complete system for objects removal in videos

### Contents

---

<b>6.1</b>	<b>Introduction</b>	<b>180</b>
<b>6.2</b>	<b>Proposed method</b>	<b>181</b>
6.2.1	Overview	181
6.2.2	First frame annotation	182
6.2.3	Objects segmentation	184
6.2.4	Objects removal	184
<b>6.3</b>	<b>Experimental results</b>	<b>186</b>
6.3.1	Comparison with state-of-the-arts objects removal methods	186
	Results with sequences of (Granados, K. I. Kim, et al. 2012)	187
	Results with sequences of (Newson, Almansa, Fradet, et al. 2014)	188
	Results with sequences from (J.-B. Huang et al. 2016)	189
6.3.2	Results with our new challenging sequences	192
6.3.3	Application in real-life situations	194
6.3.4	Impact of the segmentation masks on the inpainting performances	196
6.3.5	Some failure cases	197
<b>6.4</b>	<b>Conclusion</b>	<b>199</b>

After introducing the objects segmentation method (Chapter 3) as well as the video inpainting algorithm (Chapter 4 and Chapter 5), in this chapter, we combine these two blocks in a unified framework. We aim at building a system to remove one or several objects from a video, starting with only a few user annotations. More precisely, the user only needs to approximately delimit in the first frame the objects to be edited. Then, these annotations are refined and propagated through the video. One or several objects can then be removed automatically. This results in a flexible computational video editing tool, with numerous potential applications.

## 6.1 Introduction

Recently, advances in both the analysis and the processing of videos have permitted advances in the emerging field of computational video editing. Examples include, among others, tools for the automatic, dialogue-driven selection of scenes ([Leake et al. 2017](#)), time slice video synthesis ([Cui et al. 2017](#)), or methods for the separate editing of reflectance and illumination components ([Bonneel et al. 2014](#)), etc. Among many video editing tasks, removing unwanted objects (such as a boom microphone) or people (such as an unwanted wanderer) is a common task in video post-production. Such tasks are critical given the time constraints of movie production and the prohibitive costs of reshooting complex scenes. They are usually achieved through extremely tedious and time-consuming frame-by-frame processes, for instance using the Rotobrush tool from Adobe After Effects ([Xue Bai et al. 2009](#)) or a professional visual effects software such as Silhouette ([SILHOUETTEFX 2014](#)) Mocha ([ImagineerSystems 2014](#)).

To facilitate this task, we present in this chapter a system for the removal of objects from videos. As an input, the system only needs a user to draw a few strokes on the first frame, roughly delimiting the objects to be removed. To the best of our knowledge, this is the first such system proposed in the literature. The key steps of our system are the following: after initialization, the masks are first refined and then automatically propagated through the video. The missing regions are then synthesized using video inpainting techniques.

Two main challenges arise in developing such a system. First, not a single part of the objects to be edited shall be left over in the analysis part of the algorithm; otherwise, they are propagated and enlarged by the completion step, resulting in unpleasant artifacts. Second, our visual system is good at spotting temporal discontinuities and aberrations, making the completion step a tough one. These challenges will be addressed in both the video segmentation and the video inpainting parts of our system. As a results, our system can deal with difficult scenarios e.g. multiple objects crossing each others in front of a dynamic background. This result opens up a potential for a computational tool that can alleviate tedious manual operations for editing high-quality videos.

The first step of our system consists of transforming a rough user annotation into a mask that accurately represents the object to be edited. For this, we use a classical strategy relying on a CNN-based edge detector, followed by a watershed transform yielding super-pixels, which are eventually selected by the user to refine the segmentation mask. After this step, a label is then given to each object. The second step is the temporal propagation of the labels which is described in Chapter 3. There we make use of state-of-the-art advances in CNN-based multiple objects segmentation. Besides, our approach includes an original and crucial algorithmic brick which consists in learning the transition zones between objects and the background, in such a way that the propagated masks will fully cover the objects. Our last step is then to remove some or all of the objects from the video, depending on the user's choice. For this, we employ two strategies: a motion-based pixel propagation for static background (Chapter ??) and a patch-based video completion for dynamic background (Chapter ??). Both methods rely heavily on the knowledge of segmented objects. This interplay between objects segmentation and the completion scheme improves the method in many ways: it allows for better video stabilization, for a faster and more accurate search for similar patches, and more accurate foreground/background separation. These improvements yield completion results with very little or no temporal incoherence.

We illustrate the effectiveness of our system through several challenging cases including severe camera shake, complex and fast object motions, crossing objects, and dynamic textures. Moreover, we show on several examples that our system yields comparable or better results than the state-of-the-art video completion methods applied on manually segmented masks.

## 6.2 Proposed method

### 6.2.1 Overview

The general steps of our method are as follows (Figure 6.1):

- (a) First, the user draws a rough outline of each object of interest in one or several frames, for instance in the first one
- (b) These approximate outlines are refined by the system, then propagated to all remaining frames using different labels for different objects
- (c) If some errors are detected, the user may manually correct them in one or several frames (using step (a)) and propagate these edits to the other frames (using step (b));
- (d) Finally, the user selects which of the selected objects he/she wants to remove, and the system removes the corresponding regions in the whole video, reconstructing the missing parts in a plausible way. For this last step two options are available : a fast one for static background and a more involved one for dynamic backgrounds.



Figure 6.1 – We introduce a complete system for segmenting and removing objects in videos. Given a video and hand-drawn approximate outlines (in one frame) of objects chosen by the user, the system computes accurate segmentation masks of these objects in the whole video (Figure (a)) and then removes all or some of them according to user's choices. Figures (b) and (c) show the results of removing one particular object (characters in red), or of removing all objects, respectively.

In the first step most methods only select the object to be removed. There are, however, several advantages to tracking multiple objects with different labels:

1. It gives more freedom to the user for the inpainting step with the possibility to produce various results depending on which objects are removed; in addition, objects which are labeled but not removed are considered as important by the system and therefore better preserved during the inpainting of other objects.
2. It may produce better segmentation results than tracking a single object, in particular when several objects have similar appearance.
3. It facilitates video stabilization and therefore increases the temporal coherence during the inpainting step, as shown in the results
4. It is of interest for other applications, e.g., action recognition or scene analysis.

The illustration of these steps can be found in the supplementary website.

Next, these steps are explained in details.

### 6.2.2 First frame annotation

A classical method to cut out an object in a frame involves commercial tools such as the Magic Wand of Adobe Photoshop which is fast and convenient. However, this classical method requires many refinement steps and is not accurate with complex objects. To increase

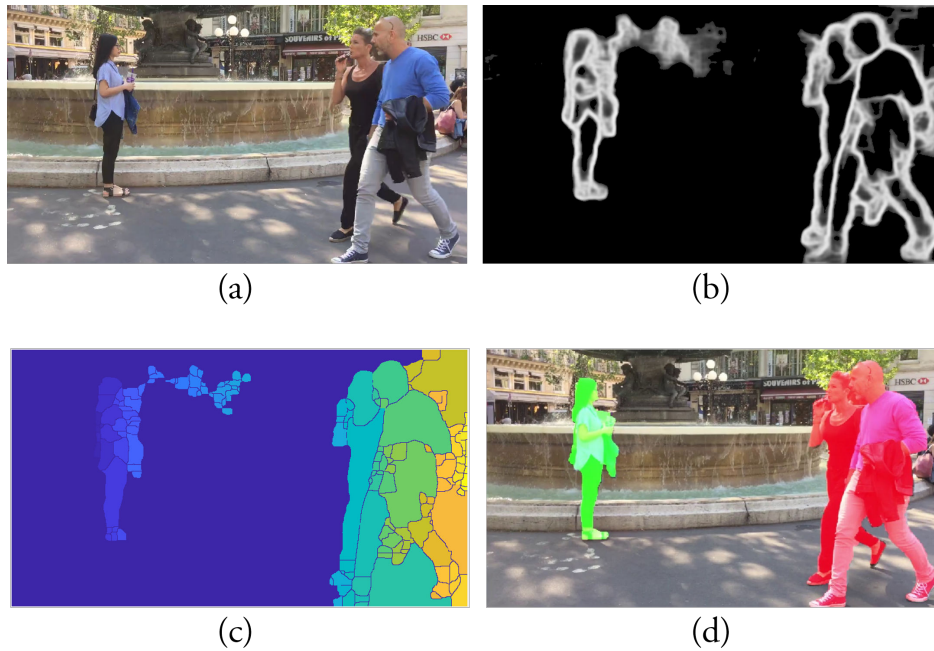


Figure 6.2 – Giving the first frame (Figure (a)), we compute an edge map for objects using a CNN-based edges detector (Xie and Zhuowen Tu 2017) (Figure (b)), then run watershed transform on this edge map to obtain superpixels (Figure (c)). Users can annotate the objects by selecting these superpixels, result in a correct masks for multiple objects (Figure (d)).

the precision and reduce the user’s intervention, many methods have been proposed where interactive image segmentation is performed using scribbles, point clicks, superpixels, etc. Among them, some state-of-the-arts annotators achieve a high degree of precision by using edge detectors to find the contour map and create a set of object proposals from this map (S. D. Jain and Grauman 2016); the appropriate regions are then selected by the user using point clicks. A major drawback of these approaches is the large computation time and the lack of interaction between human and computer.

To balance between human effort and accuracy, we adopt a fast and efficient but straightforward algorithm. The overview of our first frame annotation is shown in Figure 6.2. Our system first generates a set of superpixels from the first image, and then users can select suitable superpixels by simply drawing a rough contour around each object. The set of superpixels is created using an edge-based approach. More precisely, an FCN-based edge detector network introduced in (Xie and Zhuowen Tu 2017) is applied to the first image, and its output is a probability map of edges. Superpixels are extracted from this map by the well-known watershed transform (Meyer 1994), which runs directly on this edge map. There are two main advantages of using this CNN-based method to compute the edge map:

1. It has shown superior performances over traditional boundary detection methods that use local features such as colors and depths. In particular, this CNN-based method focuses on the edges of the objects rather than edges in the background, therefore the edge map provided by this CNN-based method is much more accurate and suitable for



the objects removal application.

2. It is extremely fast: one forward pass of the network takes about 2 ms. Hence the annotation step is performed in real time and very interactively.

After computing all superpixels, users can select a suitable object by drawing a contour around each target object to get rough masks. Superpixels which overlap these masks by more than 80 percent are selected. The user can also refine the mask by adding or removing superpixels using mouse clicks. As a result, accurate masks for all objects of interest are extracted in a frame within a few seconds of interactive annotation. A demonstration video of this process can be viewed in the supplemental website: <https://object-removal.telecom-paristech.fr/contribution.html>.

### 6.2.3 Objects segmentation

In this step, we start from the object masks computed on the first frame using the method described in the previous section, and we aim at inferring a full spatiotemporal segmentation of each object of interest in the whole video. We want our segmentation to be as accurate as possible, in particular without false negatives.

To achieve this goal, the video segmentation method described in Chapter 3 is used. In this method, the combination of smart dilation, the CNN-based segmentation method, and the object tracking method allows us to have accurate masks for video objects segmentation purpose. Besides, to add more flexibility to the system, we also allow users to correct errors. If they do not satisfy with the result in one particular frame, they can correct the mistake manually and provide an accurate mask; then this correction is used to propagate to other adjacency frames.

Figure 6.3 illustrates this situation where the book is not covered by the segmentation mask (since it does not appear at the first frame of the video). In this video, if we want the book to be covered, we can add this book to the mask in one frame, and the correction results will be propagated to the next frames automatically.

### 6.2.4 Objects removal

Following the method from the previous section, all selected objects have been segmented along the complete video sequence. From the corresponding masks, the user can then decide the objects to be removed. This last step is performed thanks to video inpainting techniques that we detailed in Chapter 4 and Chapter 5. To recap, for the case where the background is static (or can be stabilized), a fast and straightforward inpainting method relies on the reconstruction of a motion field are adopted. Then for the case where the background is moving with some complex motions and dynamic textures, a more involved method based on the optimization of spatiotemporal patch-based energy is used.

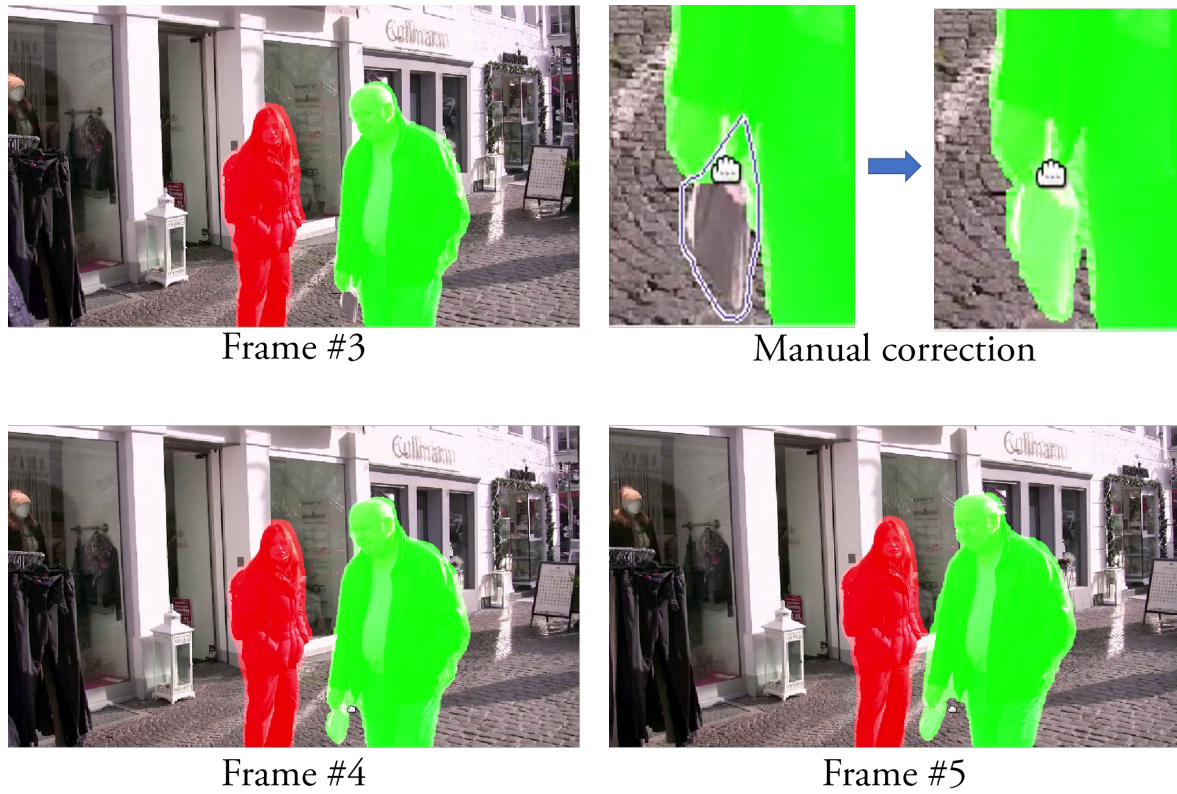


Figure 6.3 – If the users are not satisfy with the segmentation result, they can correct the error manually at one frame and the algorithm will propagation this modification to the next frames.

With multiple objects segmentation mask obtained at the previous step, users have choices to remove only a part or all of them, which provides our system with a nice interactive mode. Moreover, it is a well-known fact that the performances of inpainting algorithms may depend significantly on the quality of the inpainting masks. In our case, the accuracy of our multiple objects video segmentation method improves the performance of the video inpainting method in three ways: (1) it guarantees the quality of the masks both in space and in time; (2) it helps in creating better stabilization step and (3) it produces better background/foreground separation.

**Mask quality** It is a well-known fact that the performances of inpainting algorithms may depend significantly on the quality of the inpainting masks. In our case, the accuracy of our multiple objects video segmentation method guarantees the quality of the masks both in space and in time. The supervised segmentation step provides masks which are accurate in time and in space and which have different labels. In many previous contributions on video inpainting, masks are segmented manually frame by frame or using commercial tools as RotoBrush (Xue Bai et al. 2009; W. Li et al. 2016) which is very painful and time-consuming for the user, and which gives less guarantees for the masks to be continuous in time. Time continuity and differentiation of the masks provided by our method have a huge impact on the quality of the inpainting.

**Video stabilization** In general, patch-based video inpainting techniques require a good video stabilization as a pre-processing step to compensate patch deformations due to camera motions (Odobez and Bouthemy 1995; Sánchez 2017). This video stabilization is usually done by calculating a homography between two consecutive frames using keypoints matching followed by a RANSAC algorithm to remove outliers (Choi, T. Kim, and W. Yu 2009). However, large moving objects appearing in the video may reduce the performances of such an approach because too many keypoints may be selected on these objects and prevent the homography to be estimated accurately from the background. This problem can be solved by simply neglecting all objects for computing the homography. This is easy to do: since we already have the masks of the selected objects, we just have to remove all keypoints which are covered by masks. This is an advantage of our approach where both segmentation and inpainting are addressed.

**Background/Foreground inpainting** In addition to stabilization improvement, multiple segmentation masks are also helpful for inpainting separately the background and the foreground. More precisely, we first inpaint the background neglecting all pixels contained in segmented objects. After that, we inpaint in priority the segmented objects that we want to keep and which are partially occluded. This increases the quality of the reconstruction, both for the background and for the objects. Furthermore, it reduces the risk of blending segmented objects which are partially occluded because segmented objects have separate labels. In particular, it is extremely helpful when several objects overlap.

## 6.3 Experimental results

In the previous chapters, we evaluated our segmentation and video objects removal methods using various datasets. In this chapter, we assess the whole pipeline in a real application task. In particular, from first frame annotation, we calculate the objects segmentation masks then perform objects removal based on these resulting masks. We again using the classical sequences from previous works on video objects removal (Granados, K. I. Kim, et al. 2012; Newson, Almansa, Fradet, et al. 2014; J.-B. Huang et al. 2016) for comparison purpose, and we also introduce new challenging sequences recorded by our selves.

### 6.3.1 Comparison with state-of-the-arts objects removal methods

In this first experiment, we re-perform our video objects removal technique on classical sequences from (Granados, K. I. Kim, et al. 2012; Newson, Almansa, Fradet, et al. 2014; J.-B. Huang et al. 2016) using our automatic segmentation masks.

### Results with sequences of (Granados, K. I. Kim, et al. 2012)

The dataset of (Granados, K. I. Kim, et al. 2012) contains seven real-world sequences captured in the outdoor environment at four different scenes. They are publicly available at <http://gvv.mpi-inf.mpg.de/projects/vidbginp/>. We focus here on 4 sequences: GRANADOS-S1, GRANADOS-S2, GRANADOS-S3, and GRANADOS-S7 since GRANADOS-S4, GRANADOS-S5, GRANADOS-S6 have similar contents with GRANADOS-S4. These sequences are captured with a hand-held digital camcorder in Full HD resolution at 25fps. Each sequence features two or more people moving in front of a static background. We focus on the occluded moment where a moving person walks in front or behind another immobile person. The dimension of the video is also reduced to  $854 \times 480$  for memory efficient.

While in (Granados, K. I. Kim, et al. 2012), the mask of the object to be removed and the mask of the remaining foreground objects were created manually using Rotobrush tool available in Adobe After Effects CS5, in our method, these masks are provided automatically by our segmentation algorithm.

Three main challenges arise while creating these masks: the motion blur, the shadow, and the occlusion. The blur regions caused by the fast movement of the person are difficult to handle since the traditional segmentation networks will confuse to specify these regions as background or foreground. If the segmentation masks do not cover any blur part of the objects, the video looks very unpleasant. Another challenge is the shadow in these sequences are extremely hard to capture by any segmentation method since the shadow boundary is difficult to mask correctly. Therefore, we do not segment the shadow in these sequences. To the best of our knowledge, shadow extraction is challenging and there not much method working on this kind of task. Another challenge of these sequences is the occlusion, where a moving person is fully occluded by another person. This occlusion situation not only causes difficulty to any mask-tracking based segmentation but also is a real challenge for video inpainting algorithm since both the background and the objects need to be reconstructed.

For this experiment, we use our patch-based method rather than motion-guided pixel propagation method to reconstruct the video because of two main reasons. The first reason is that even though the background is static, we need to rebuild an object which is occluded by another moving object. The object to be reconstructed have a small movement which is different from the movement caused by the camera instability. Thus, using simple motion-guided pixel reconstruction, in this case, is not a good choice. The second reason is that the mask of our method is moving, and therefore the information behind the mask is always revealed at some particular times in the video, which means that there is no region where the data behind it is never visible during the video (this is the main problem of our patch-based synthesis approach). Therefore, the patch-based approach produces a temporal consistency result.

In Figure 6.4, we show several representative images of our segmentation and objects



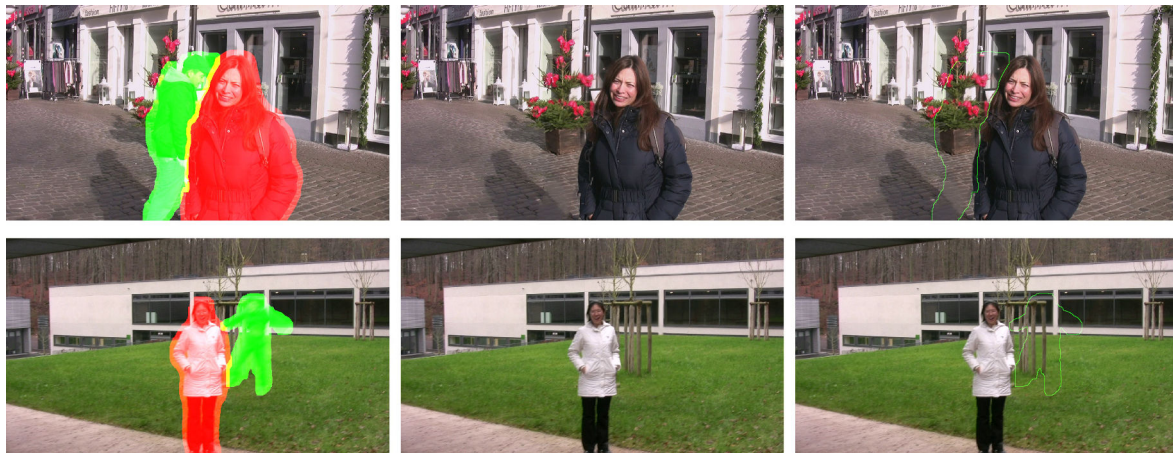


Figure 6.4 – Our object removal results on GRANADOS-S1 (top row) and GRANADOS-S3 (bottom row). In each sequence, we provide our segmentation result, our result of removing the moving object (green mask), our result with the border of the occlusion, respectively.

removal results on GRANADOS-S1 and GRANADOS-S3 sequences. The full video results can be found in the supplementary website. From these videos, we can see that the quality of inpainting results using our automatic segmentation masks is comparable with the one of (Granados, K. I. Kim, et al. 2012) performed on manually segmentation masks. More specifically, the motion blur is almost covered by our smart dilation layer so that the background can be well reconstructed. In these videos, we do not aim at segment the shadow; hence some visible artifact still exists. However, the video, in general, look natural with temporal consistency. We also notice that geometric structures (e.g., windows in GRANADOS-S3 sequence and door in GRANADOS-S1 sequence) are preserved. Nevertheless, several artifacts persist. For example, in GRANADOS-S1 sequence, there is small distortion in the reconstructed girl when the man occludes the standing girl, such as in the hair. The reason for this is because our mask also covers some parts of the girl so that when the motion of the girl different from the background, it causes distortion.

To go further, we perform a more ambitious experiment while still using the same video as (Granados, K. I. Kim, et al. 2012), which is removing all the object in the scenes, including the moving objects and the static objects. This task is difficult because the information behind the static object is never revealed so that the algorithm must invent it. We try with the sequence GRANADOS-S1 and GRANADOS-S3. Figure 6.5 demonstrates some representative frames of the results. As can be seen, the algorithm can maintain the temporal consistency for the background, even behind the static object. Although several geometric structures or some details of the background are not reconstructed perfectly, the results still look plausible when playing as a sequence.

### Results with sequences of (Newson, Almansa, Fradet, et al. 2014)

For the comparison with (Newson, Almansa, Fradet, et al. 2014), we pick two sequences: FOUNTAIN-CHATELET and LES-LOULOUS and perform our whole pipeline on those





Figure 6.5 – We reuse the sequences of (Granados, Tompkin, et al. 2012), in particular, GRANADOS-S1 (top row) and GRANADOS-S3 (bottom row) and try to remove all static and dynamic objects. Our method provides plausible results with these sequences. These results highlight the capacity of our algorithm to perform well in a wide range of inpainting situations.

sequences. While the FOUNTAIN-CHATELET sequence is simple which contains a man moving slightly in front of a dynamic fountain, LES-LOULOUS sequence is more challenge because it includes two moving people with similar appearances cross each other under a dynamic background and camera instability. For these sequences, we pick the patch-based synthesis approach since the motion-guided method cannot deal with dynamic textures.

Figure 6.6 shows several frames of the results. We obtain similar high-quality results with (Newson, Almansa, Fradet, et al. 2014) while using automatic segmentation masks. The masks created by our method somehow have high temporal consistency than the manually segmented mask by (Newson, Almansa, Fradet, et al. 2014). For the objects removal task, the results on the FOUNTAIN-CHATELET sequence is almost perfect without any artifact. By observation under the human visual system, we cannot spot the differences between the original and the reconstructed region. In LES-LOULOUS sequence, the segmentation masks are quite satisfying but the reconstructed video still contains visible artifacts, especially the ghost effect around the border of the reconstructed region. These problems are also reported in (Newson, Almansa, Fradet, et al. 2014) and this sequence remains challenging for any video objects removal algorithm.

### Results with sequences from (J.-B. Huang et al. 2016)

For comparison with (J.-B. Huang et al. 2016), we use 3 sequences: PARAGLIDING-LAUNCH, COWS and CAMEL from DAVIS-2016 dataset (Federico Perazzi, Jordi Pont-Tuset, et al. 2016). The major challenges of these sequences are the large camera movement, small details to segment (e.g., the rope of the parachute on PARAGLIDING-LAUNCH sequence), large occlusion (COWS) or multiple objects of the same category (CAMEL). For these examples, motion-guided pixel propagation method will perform better since the background

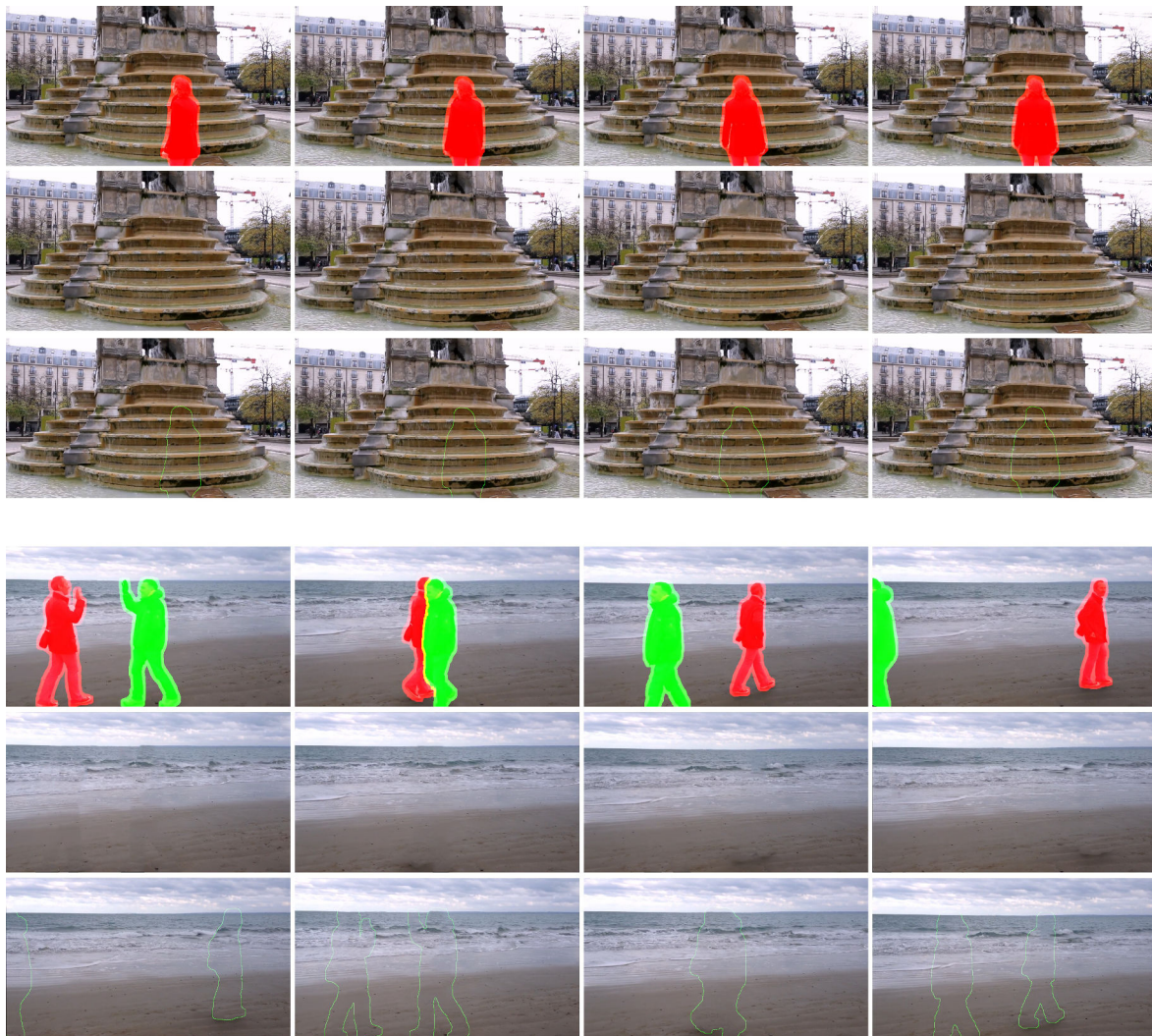


Figure 6.6 – Our objects removal results with 2 sequences from (Newson, Almansa, Fradet, et al. 2014). Top: FOUNTAIN-CHATELET, bottom: LES-LOULOUS. In each sequence, we provide our segmentation results, our result of removing all objects, our result with the border of the occlusion from top to bottom respectively.



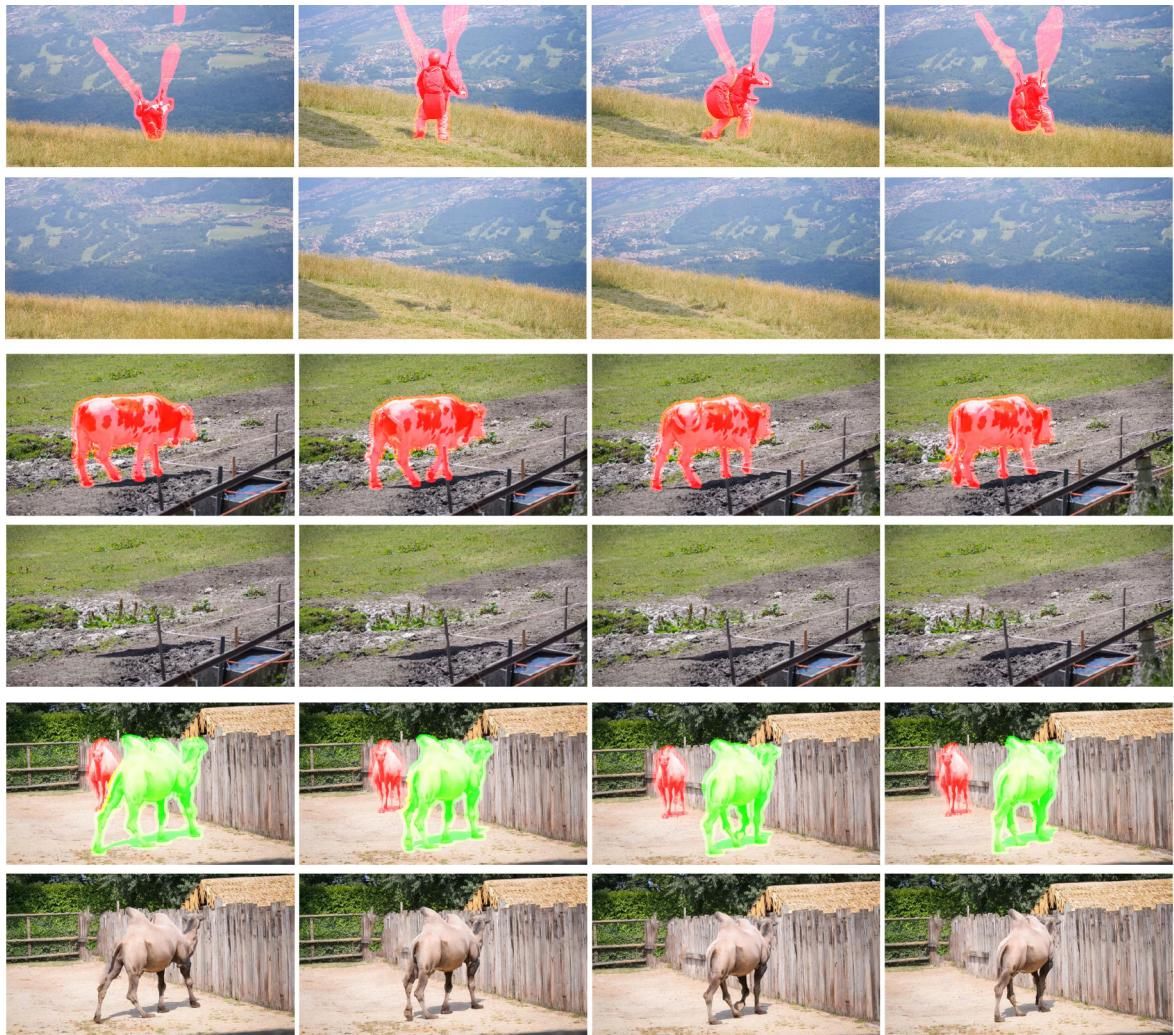


Figure 6.7 – Our object removal results on 3 sequences of DAVIS-2016 dataset: PARAGLIDING-LAUNCH (top), COWS (middle) and CAMEL (bottom). In each sequence, we provide our segmentation results and our result of remove all objects respectively.

is static without any dynamic texture or moving objects. As illustrated in Figure 6.7, we obtain high quality results, which are similar to (J.-B. Huang et al. 2016). These results conclude that a simple motion-guided pixel propagation method is enough to reconstruct the static background and a global and general framework like (J.-B. Huang et al. 2016) is not needed. This fact is also reported in (Bokov and Vatolin 2018).

### 6.3.2 Results with our new challenging sequences

We also introduce 4 new challenging sequences which are: BEAR, BIRD, KART, and FOUNTAIN-OPERA. These sequences contain multiple objects in static or dynamic scenes. These sequences raise many challenges to both video segmentation and video objects removal algorithm, mostly because of the similar of object appearances, the occlusion and the presence of both static and moving objects. The objective is to remove any of these objects (one or all of them) base on the user's choice. Results of our methods with these 4 sequences: BEAR, BIRD, KART and FOUNTAIN-OPERA are shown in Figure 6.8, 6.9, 6.10, 6.11 respectively.

The first BEAR sequence is extracted from a YouTube video (under Creative Common Licence). It comprises 3 similar bears, one static and two others cross each other. The background has high geometric details and the camera has a small movement. As illustrated in Figure 6.8, our objects removal results for this video is awe-inspiring. For video segmentation results, although objects have similar appearances, we are able to distinguish each object instance. For the video objects removal results, our result when we remove all objects and reconstruct the background have high consistency in both spatial and temporal direction. Especially, when we want to remove only one object (e.g., the bear on the left), the algorithm can reproduce the moving objects even it is occluded, which is very impressive.

The second BIRD sequence is also extracted from Youtube. In this sequence, two similar birds are drinking water in front of a static background containing high geometric structures. The major challenge in this sequence is the presence of regions which are never seen anywhere in the video. This sequence is used to examine the ability of the motion-guided pixel propagation algorithm. Similar to BEAR sequence, the result, in this case, also has high temporal coherence. However, we note that there are small artifacts caused by the remaining details (water).

The next KART sequence, taken from DAVIS-2017 dataset (Jordi Pont-Tuset, Federico Perazzi, et al. 2017), is a sequence about two similar karts are racing under static background. This sequence is challenging because the occlusion is very large and the camera is unstable, which causes difficulties to the video stabilization algorithm. As explained in Section 6.2.4, the segmentation masks provided by the segmentation step help us obtain more stable video and lead to better reconstruction. As a result, we attain a good quality video in this case.

Finally, we present our most challenging sequence: FOUNTAIN-OPERA. This sequence contains a static object occluded by two moving objects in front of a dynamic background. The objective is to remove any of these objects: the moving objects, the static object or both. We



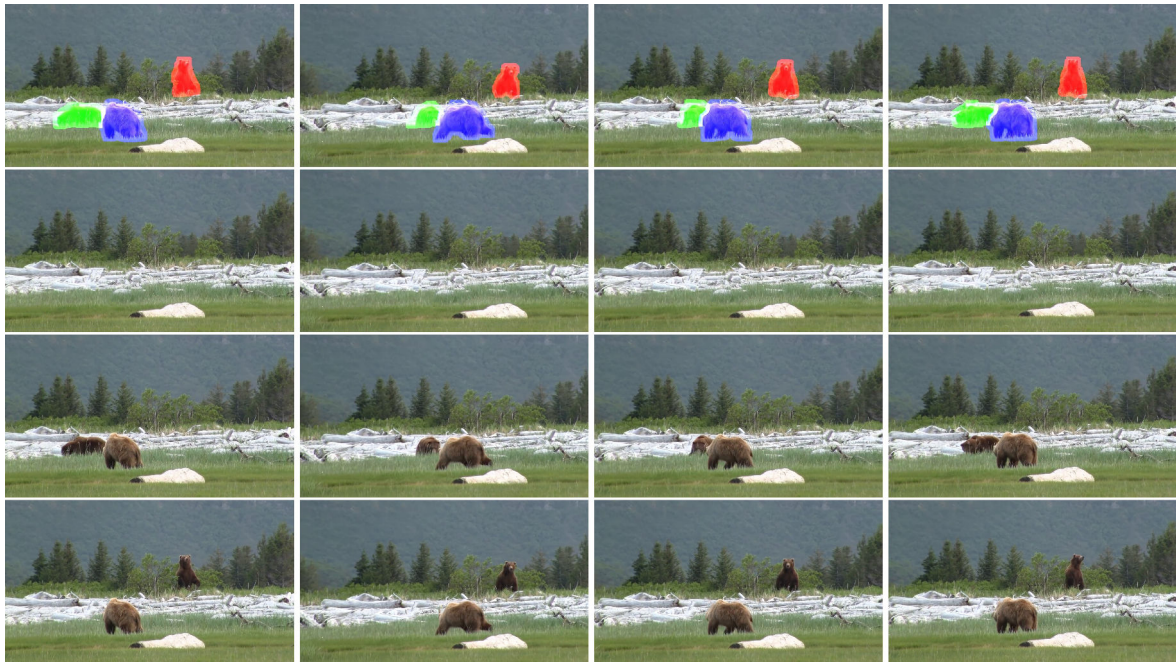


Figure 6.8 – Our object removal results on BEAR sequence. From top to bottom: our multiple objects segmentation results, the result of removing all objects, the result of removing the object depicted in red, the result of removing the object depicted in green.

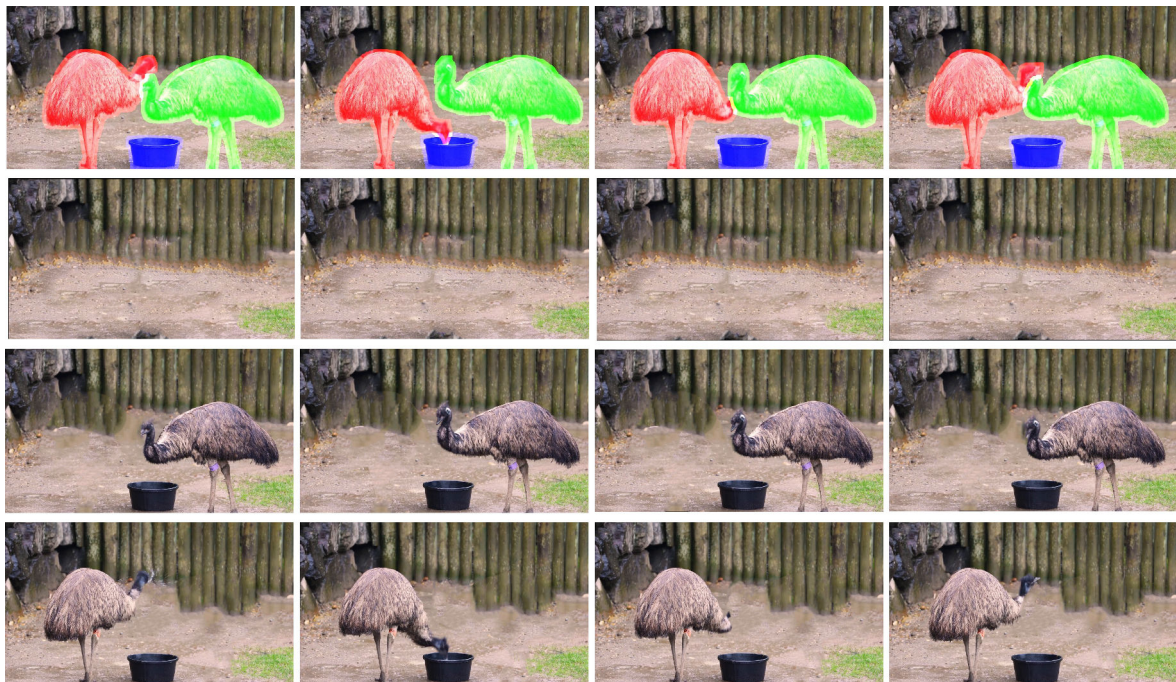


Figure 6.9 – Our object removal results on BIRD sequence. From top to bottom: our multiple objects segmentation results, the result of removing all objects, the result of removing the object depicted in red, the result of removing the object depicted in green.



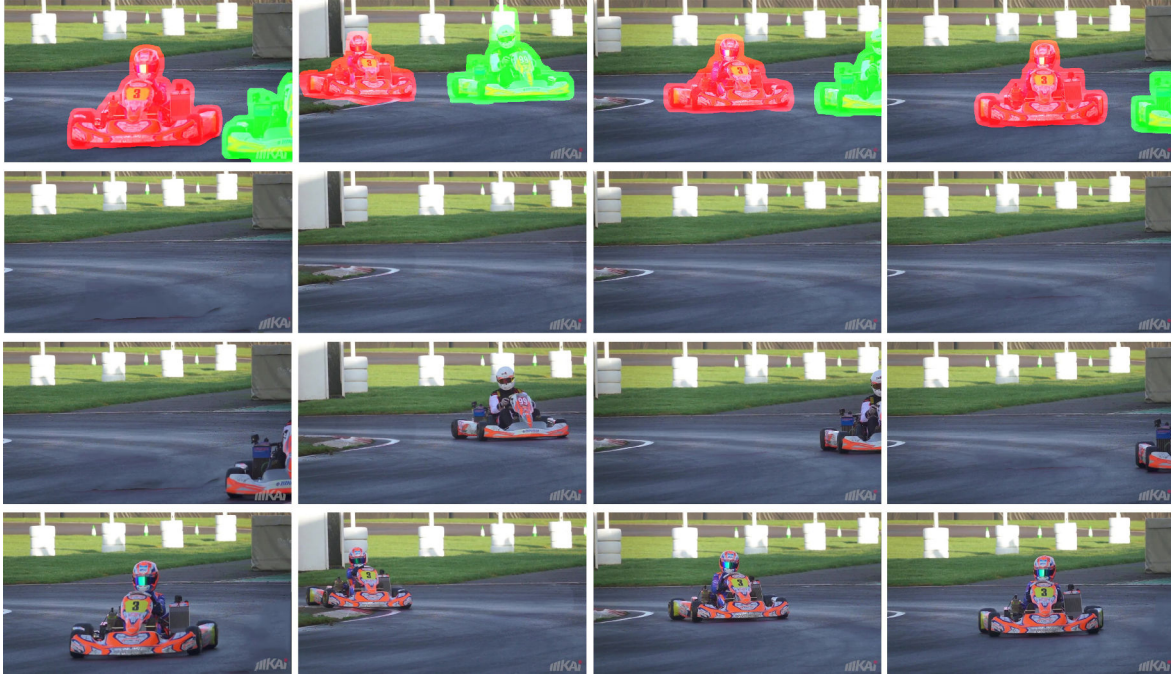


Figure 6.10 – Our object removal results on KART sequence. From top to bottom: our multiple objects segmentation results, the result of removing all objects, the result of removing the object depicted in red, the result of removing the object depicted in green.

apply our video objects removal pipeline and report the result in Figure 6.11. In this example, we use our patch-based synthesis approach to deal with this dynamic background case. The results for this video of our method is very pleasant, in both three cases: (1) removing the static object, (2) removing the moving objects, or removing both and reconstructing the background.

### 6.3.3 Application in real-life situations

In this experiment, we investigate some scenarios in real-life where objects removal application is needed. In particular, we choose four videos: WIRE, BICYCLE-RACE, STICK, and SURF. Some representative frames of these video with the segmentation masks and the corresponding objects removal results are illustrated in Figure 6.12.

The WIRE sequence is taken from (Bokov and Vatolin 2018) where a beautiful scene is impeded by two cable wire while filming. The objective is then to remove these wires. After our video objects removal algorithm, the results are almost perfect, and the scene looks satisfaction.

The next BICYCLE-RACE sequence is taken from Youtube-VOS dataset (N. Xu, L. Yang, et al. 2018) which records a fantastic moment at the finish of the race. Unfortunately, this moment is ruined by a cameraman. Since this moment is unique and there is no way to re-shoot the video, all we can do is to remove the cameraman out of the video. This can be done easily with our objects removal algorithm. After removing the cameraman, the scene looks perfect.

The third SURF sequence is also taken from Youtube-VOS dataset (N. Xu, L. Yang, et al.





Figure 6.11 – Our object removal results on FOUNTAIN-OPERA sequence. From top to bottom: our multiple objects segmentation results, the result of removing all objects, the result of removing the object depicted in red, the result of removing the object depicted in green.

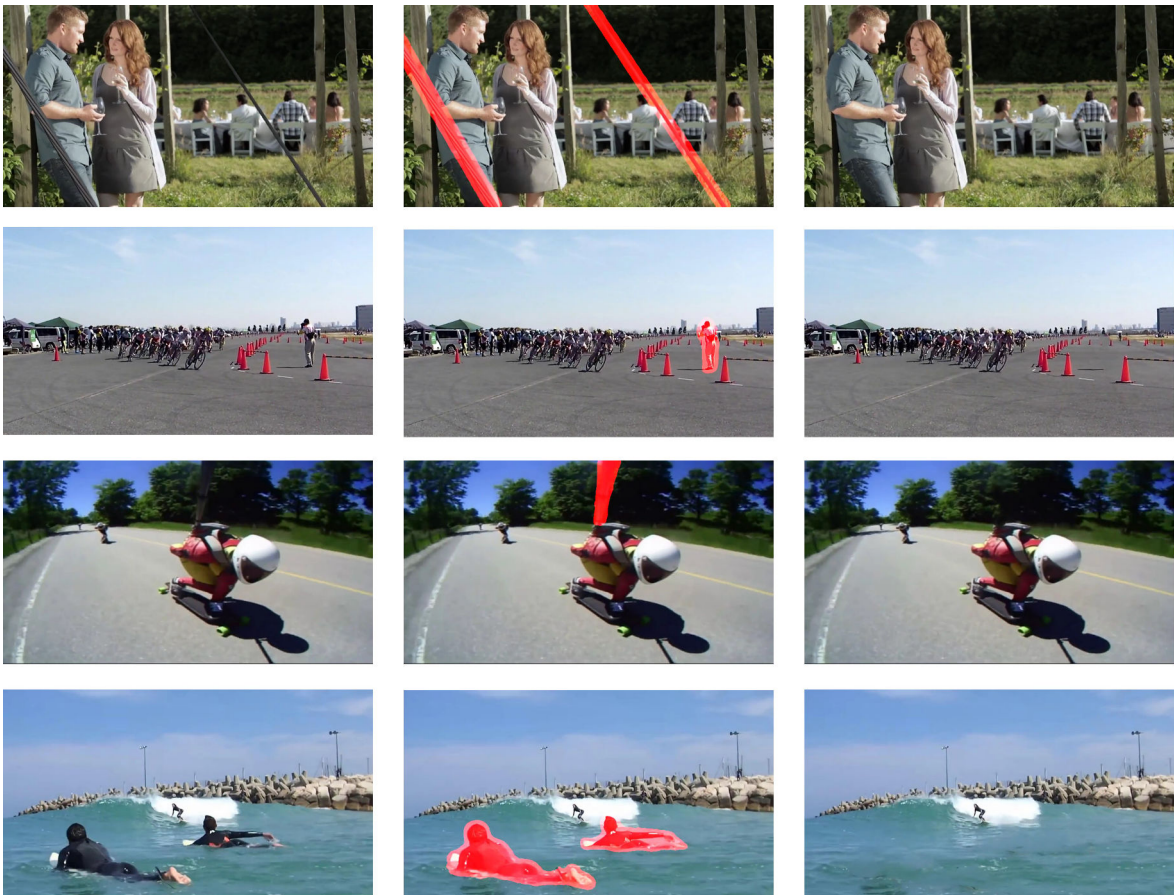


Figure 6.12 – Some real-life situations where objects removal is needed. From top to bottom: WIRE, BICYCLE-RACE, STICK, SURF. In each video, we provide the original image, our segmentation results and our object removal results from left to right respectively.

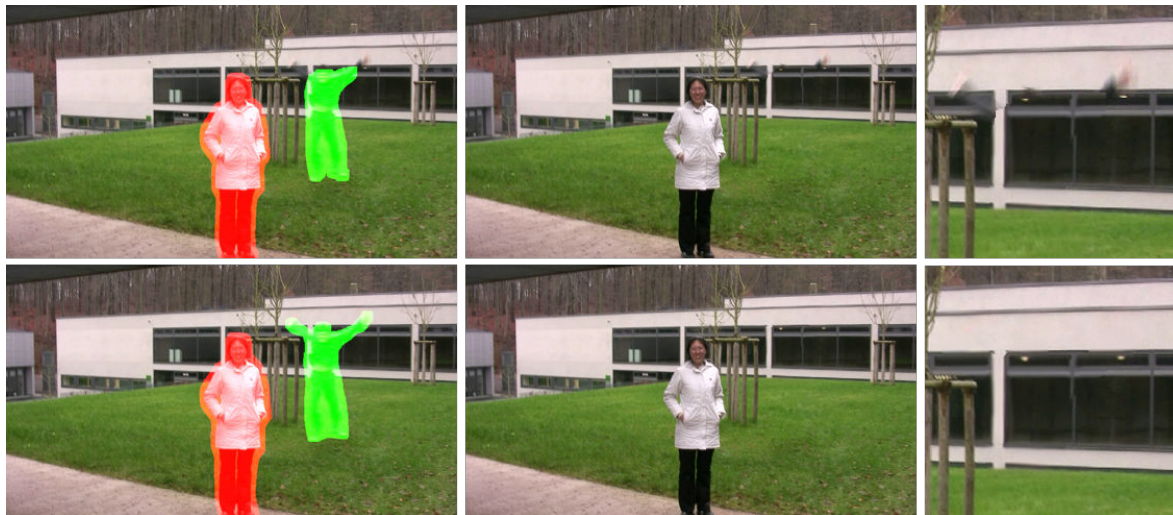


Figure 6.13 – Results of object removal using masks computed by OSVOS (top) and ours (bottom). From left to right: Segmentation mask, the resulting object removal on one frame, zooms. We can see that when the segmentation masks do not fully cover the object (OSVOS), the resulting video contain visible artifacts (the hand of the man remains after object removal).

2018). In this video, a wonderful moment of a surfer is damage by two random persons. After removing these random people, we can obtain a stunning video.

Another application of objects removal algorithm is to removing a selfie stick as in STICK sequence. In this sequence, a scene of a rolling skater is recorded by a camera mounted in a stick, so that some parts of the stick still left in the video. After our object removal algorithm, the video looks more natural.

Beside these situations, we strongly believe that there are more scenarios in real-life where objects removal is helpful.

### 6.3.4 Impact of the segmentation masks on the inpainting performances

In these experiments, we highlight the advantages of using the segmentation masks of multiple objects to improve the video inpainting results.

First, we emphasize the need for masks which fully cover the objects to be removed. Figure 6.13 (top) demonstrate the situation where some object details (the waving hand in this case) are not covered by the mask (here using the state-of-the-art OSVOS method) (S. Caelles et al. 2017). This situation leads to a very unpleasant artifact when video inpainting is performed. Thanks to the smart dilation, introduced in the previous sections, our segmentation mask fully cover the object to be removed, yielding a more plausible video after the inpainting step.

Object segmentation masks can also be helpful for the video stabilization step. Indeed, in case of large foregrounds, these can have a strong effect on the stabilization procedure, yielding a bad stabilization of the background, which in turn yields bad inpainting results. In contrast, if the stabilization is applied only to the background, the final object removal results





Figure 6.14 – The advantage of using the segmentation masks to separate background and foreground. Left: without separating background/foreground, the result have many artifacts. Right: the background and foreground are well reconstructed when being reconstructed independently.

are much better. This situation is illustrated in the supplementary material.

To further investigate the advantage of using multiple segmentation masks to separate background/foreground in the video completion algorithm, we compare our method with the direct application of the inpainting method from (Le et al. 2017), without separating objects and background. Representative frames of both approaches are shown in Figure 6.14. Clearly, (Le et al. 2017) produce artifacts when the moving objects (the two characters) overlap the occlusion, due to patches from these moving objects being propagated within the occlusion in the nearest neighbor search step. Our method, on the other hand, does not suffer from this problem because we reconstruct background and moving objects separately. This way, the background is more stable, and the moving objects are well reconstructed.

### 6.3.5 Some failure cases

In the previous section, we present our superior results for different videos. However, since video objects removal is a very challenging problem, there are many situations where our method fails to perform. We present here some failure cases of our approach. The visual illustration is shown in Figure 6.15. The video results can be found in the supplementary website.

The first sequence is LES-LOULOUS from (Newson, Almansa, Fradet, et al. 2014). In this video, the purpose is to remove the man on the left (depicted in red). In this video, we obtain encouraging video result, in particular, we can reconstruct the remaining moving object (the man depicted in green). However, ghost artifacts around the border of the occlusion and the deformation of the remaining object when it enter the occlusion make the video look unnaturally. These artifacts are also reported in different methods (Newson, Almansa, Fradet, et al. 2014). Therefore, dealing with this video is a huge challenge for any video objects removal application.

The next sequence is GRANADOS-S1 from (Granados, K. I. Kim, et al. 2012). In this video, we perform 3 task: removing the moving object (as in ()) remove all objects (which is

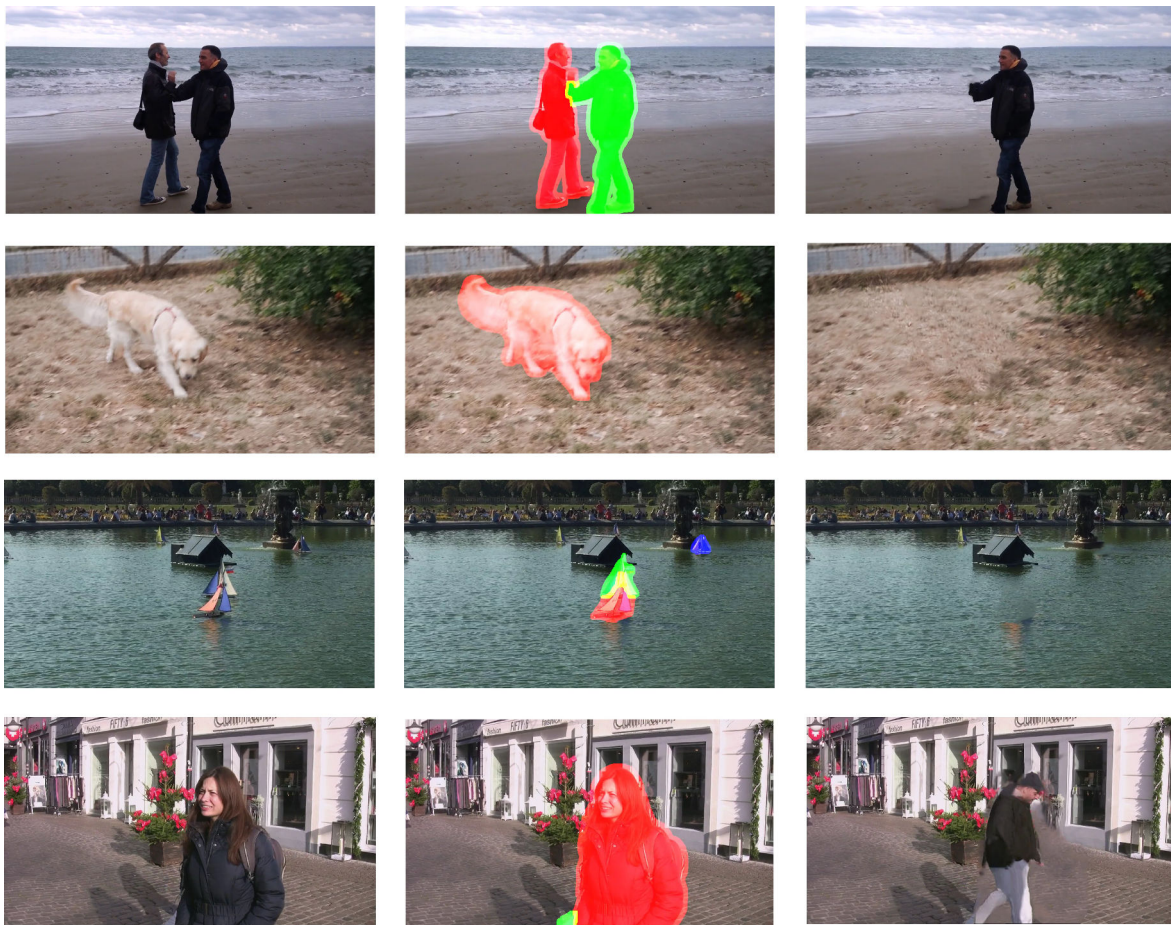


Figure 6.15 – Several failure cases of our object removals algorithm. From top to bottom: LES-LOULOUS, DOG, BOAT, GRANADOS-S1. In each video, we provide the original image, our segmentation results and our object removal results from left to right respectively.



harder because information behind static objects is never revealed) and remove static object only (which is an arduous task because we need to reconstruct both the moving object (which is occluded) and the background at the same time). For the results, while the result of removing the moving objects is quite good, when we try to remove all objects, a geometric shape of the scenes is not preserved (mostly because the information behind the static objects is not available). Still, we can obtain a stable background and temporal coherence. The most failure case is when try to remove only the static object. In this case, we see that the moving object is not well reconstructed because it is occluded for a long period and the movement of the moving objects is very complex.

Another failure of our system is when dealing with very complex texture, low frame rate and large camera displacement, such as in the DOG sequence, taken from DAVIS-2016 dataset. In this sequence, the video stabilization algorithm fails and the result contains lots of artifacts.

Finally, our system has limitations when dealing with complex shadows, such as in BOAT sequences. In these sequences, the shadows of the boats are reflected in the water, and can not segment by the segmentation algorithm. This shadow makes the video look very unnaturally.

We also notice that our segmentation method cannot deal with GRANADOS-MUSEUM sequence since the scene is extremely complex with many different objects with similar appearance cross each other.

## 6.4 Conclusion

In this chapter, we have provided a full system performing object removal in videos. The input of the system is made of a few strokes provided by the user to indicate the objects to be removed. To the best of our knowledge, this is the first system of this kind, even though the Adobe company has recently announced to be developing such a tool, under the name *Cloak*. The approach can deal with multiple, possibly crossing objects, and can reproduce complex motions and dynamic textures.

Although our method achieves good visual results on different datasets, it still suffers from a few limitations. First, parts of the objects to be edited may be ignored by the segmentation masks. In such cases, as already emphasized, the inpainting step of the algorithm will amplify the remaining parts, creating strong artifacts. This is an intrinsic problem of the semi-supervised object removal task and room remains for further improvement. Further, the system is still relatively slow, and in any case far from realtime. Accelerating the system could allow for interactive scenarios where the user can gradually correct the segmentation-inpainting loop.

The segmentation of shadows is still not flawlessly performed by our system, especially when the shadows are not strongly contrasted. It is a desirable property of the system to be able to deal with such cases. This problem can be seen in several examples provided in the supplementary material.

Another limitation occurs in some cases where the background is not revealed, specifically when some semantic information should be used. Such difficult cases are gradually being solved for single images by using CNN-based inpainting schemes ([Iizuka, Simo-Serra, and Ishikawa 2017](#)). While the training step of such methods is still out of reach for videos as of today, developing an object removal scheme fully relying on neural networks is an exciting research direction.

# 7

## Conclusion, limitations and future works

### Contents

<a href="#">7.1 Conclusion</a>	201
<a href="#">7.2 Limitations and perspectives</a>	203

### 7.1 Conclusion

In this dissertation, we have studied different components for creating an object removal application in video, which includes two main steps: (1) video object segmentation and (2) video inpainting.

First, in Chapter 3, we proposed a video object segmentation algorithm with the purpose of creating the masks for object removal. The proposed algorithm can separate multiple objects in difficult situations such as camera movement, occlusion, similar objects of the same semantic class and object crossing. This robustness is obtained by a combination of CNN-based segmentation networks, a classical tracking method and a graph-based data association strategy. Moreover, we adapt our method to the goal of creating a mask which covers all details of the objects as much as possible through the introduction of a new layer called *smart dilation*. We then evaluate our algorithm visually and quantitatively in comparison with previous work, and find out that the our method achieves high recall metric (which is our main goal) while the precision and the *mIoU* are comparable with respect to previous works.

Second, we solve the problem of video completion (or inpainting) by separating it into two

sub-problems: one for the static background case and one for the complex background case.

Chapter 4 dealt with the problem of video inpainting with static background. We have shown that the missing information in the static background case can be filled in by pixels in the source region which are found by tracing via the optical flow field. To solve the problem of missing optical flow information inside the occluded regions, we proposed a simple image inpainting technique which treats the optical flow field as an image. Finally, the problem of illumination changes are solved by the extension of Poisson blending in 3D. By experimenting with different sequences and comparing the results with other state-of-the-art methods, we showed that our proposed approach is not only conceptually simple, but also provides result that are comparable with existing approaches in the same category.

To deal with more complex videos with dynamic textures and moving objects, in Chapter 5, we present a motion-consistent video inpainting method by optimizing a global patch-based function. In particular, we adopt the similar idea of globally copying and pasting spatio-temporal patches, as introduced in (Wexler, Shechtman, and Irani 2007; Newson, Almansa, Fradet, et al. 2014). However, we improve these method both in speed and accuracy. To increase the speed of the algorithm, we proposed a parallel extension of the 3D PatchMatch algorithm with several modifications in the random search. This way, we can speed up the source patch searching process by a factor of 5-7 while preserving the same accuracy level. We also observed that previous methods have difficulty in reconstruct moving objects. To solve that problem, we systematically incorporated the optical flow in the overall process : in the patch distance, patch shape, initialization and nearest neighbor search. We then evaluate visually in comparison with previous work, and found a significant improvement in reconstructing moving objects as well as in reproducing dynamic textures with high temporal consistency.

Finally, in Chapter 6, the segmentation and video inpainting steps are combined in a unified system for objects removal in video. The input of the system is made of a few strokes provided by the user to indicate the objects to be removed. To the best of our knowledge, this is the first system of this kind. In our system, the user only needs to approximately delimit in the first frame the objects to be edited. These annotation is very easy with the help of superpixels. Then, these annotations are refined and propagated through the video. One or several objects can then be removed automatically. This results in a flexible objects removal tool with numerous potential applications. More generally, the proposed system paves the way to sophisticated movie editing tasks, ranging from crowd suppression to unphysical scenes correction, and has potential applications for multi-layered video editing.

We hope that the thesis work not only contributes to the academic world in advanced video editing domain, but also opens an opportunity for creating a real application which can be commercialized.

## 7.2 Limitations and perspectives

Indeed, the proposed system still has some limitations. We mention below some of the most important academic and industrial challenges that remain to be solved.

**Speed.** Our system have high computational complexity both in video segmentation and video inpainting steps which is a real obstacle for a practical application. One way to solve that problem is by choosing a faster video segmentation method, such as (L. Yang et al. 2018), and enable more users interaction to compensate for the scarification in accuracy.

**Unified video inpainting system.** Our motion-guided pixels propagation method can deal with static background cases while our global patch-based method work well with dynamic background cases. A natural question arises: how to combine these two methods in a unified manner. One possibility could be to use the result of motion-guided propagation method as the initialization for the patch-based algorithm. However, this would slow down the reconstruction in uncomplicated cases. Combining these two methods efficiently is an open question and should be investigated further.

**Shadows and reflections.** Our system excels in inpainting dynamic textures, but it is confused when it has to inpaint shadows or reflections. To overcome this difficulty we need to integrate new tools that allow both to reliably segment shadows and to paste shadowed patches into better illuminated areas and vice-versa. A similar shadow-related problem was recently solved for images via *osmosis filtering* by (Calatroni et al. 2018) among others. This provides a nice start to generalize our Poisson editing scheme to better deal with shadows. Reflections pose and even more difficult challenge.

**Semantic Inpainting.** Both example-based and tracking-based video inpainting algorithms work well when the occluded area's textural, shape and motion characteristics resemble the information that is available in the visible are of the same sequence. However when the occluded area is expected to show *e.g.* a human face, and no other face is visible in the sequence, with the right context the system will never produce a satisfying result. This kind of semantic inpainting task can only be solved by extensive learning on very large external databases, and some results are starting to emerge under extremely constrained settings, for static images only, and after several months of learning on high-performance GPUs (Iizuka, Simo-Serra, and Ishikawa 2017). Generalizing such CNN-based approaches to videos in less constrained situations is a huge challenge that remains to be addressed.

**Invariances.** In some situations simple example-based inpainting algorithms fail, not because of the absence of good examples to copy from, but because those examples are not at the right scale, rotation or perspective. The importance of affine invariance in example-based



image inpainting was demonstrated by Fedorov (2016, Chapter 10), see also (Fedorov, Facciolo, and Arias 2015).

**Interactivity and complex layering.** In extremely complex situations where many plausible inpainting solutions are admissible, it would be useful to have a much higher-level degree of interactivity and let the user decide of the general appearance of the inpainted sequence. This is the case for instance when two or more moving objects cross each other inside the occlusion. In this case we have first to decide on the layering order of the sequence (which object should be visible in the foreground and which one should be partly occluded behind it) before we let the inpainting algorithm look for a solution which is compatible with those layering constraints, much in the same way as it was done by (Cao et al. 2011) to constrain an image inpainting algorithm via semantic labeling. This decision could be based on user-interaction, or based on additional cues such as depth, which can be reliably estimated even from a single image thanks to recent advances in CNN research (Carvalho et al. 2018; Kendall and Gal 2017).

# Bibliography

- Arbeláez, Pablo et al. (2014). “Multiscale combinatorial grouping.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 328–335.
- Arias, Pablo, Vicent Caselles, and Gabriele Facciolo (2012). “Analysis of a variational framework for exemplar-based image inpainting.” *Multiscale Modeling & Simulation* 10.2, Pages 473–514.
- Babaei, Mohammadreza, Yue You, and Gerhard Rigoll (2016). “Pixel Level Tracking of Multiple Targets in Crowded Environments.” In: *European Conference on Computer Vision*. Springer, Pages 692–708.
- Badrinarayanan, Vijay, Fabio Galasso, and Roberto Cipolla (2010). “Label propagation in video sequences.” In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, Pages 3265–3272.
- Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2015). “Segnet: A deep convolutional encoder-decoder architecture for image segmentation.” *arXiv preprint arXiv:1511.00561*.
- Bai, Xue et al. (2009). “Video snapcut: robust video object cutout using localized classifiers.” In: *ACM Transactions on Graphics (ToG)*. Volume 28. 3. ACM, Page 70.
- Ballester, Coloma et al. (2001). “Filling-in by joint interpolation of vector fields and gray levels.” *IEEE transactions on image processing* 10.8, Pages 1200–1211.
- Banica, Dan et al. (2013). “Video object segmentation by salient segment chain composition.” In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*, Pages 283–290.
- Bansal, Aayush et al. (2017). “Pixelnet: Representation of the pixels, by the pixels, and for the pixels.” *arXiv preprint arXiv:1702.06506*.
- Bao, Linchao, Baoyuan Wu, and Wei Liu (no date). “CNN in MRF: Video Object Segmentation via Inference in A CNN-Based Higher-Order Spatio-Temporal MRF.” In:
- Barnes, Connelly, Eli Shechtman, et al. (2009). “PatchMatch: A randomized correspondence algorithm for structural image editing.” *ACM Transactions on Graphics (ToG)* 28.3, Page 24.
- Barnes, Connelly, Fang-Lue Zhang, et al. (2015). “Patchtable: Efficient patch queries for large datasets and applications.” *ACM Transactions on Graphics (TOG)* 34.4, Page 97.
- Barron, Jonathan T and Ben Poole (2016). “The fast bilateral solver.” In: *European Conference on Computer Vision*. Springer, Pages 617–632.
- Bay, Herbert et al. (2008). “Speeded-Up Robust Features (SURF).” *Computer Vision and Image Understanding* 110.3, Pages 346–359. ISSN: 1077-3142. DOI: [10.1016/J.CVIU.2007.09.014](https://doi.org/10.1016/J.CVIU.2007.09.014).
- Bazarevsky, Valentin and LLC Andrei Tkachenka (2018). *Mobile Real-time Video Segmentation*. URL: <https://ai.googleblog.com/2018/03/mobile-real-time-video-segmentation.html> (visited on 02/07/2019).

- Bertalmio, Marcelo, Guillermo Sapiro, et al. (2000a). “Image Inpainting.” In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00, Pages 417–424.
- (2000b). “Image inpainting.” In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., Pages 417–424.
- Bertalmio, Marcelo, Luminita Vese, et al. (2003). “Simultaneous structure and texture image inpainting.” *IEEE transactions on image processing* 12.8, Pages 882–889.
- Bertinetto, Luca et al. (2016). “Fully-convolutional siamese networks for object tracking.” In: *European Conference on Computer Vision*. Springer, Pages 850–865.
- Bian, Xiao, Ser Nam Lim, and Ning Zhou (2016). “Multiscale fully convolutional network with application to industrial inspection.” In: *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE, Pages 1–8.
- Bokov, Alexander and Dmitriy Vatolin (2018). “100+ TIMES FASTER VIDEO COMPLETION BY OPTICAL-FLOW-GUIDED VARIATIONAL REFINEMENT.”
- Bolme, David S et al. (2010). “Visual object tracking using adaptive correlation filters.” In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, Pages 2544–2550.
- Bonneel, Nicolas et al. (2014). “Interactive intrinsic video editing.” *ACM Transactions on Graphics (TOG)* 33.6, Page 197.
- Bookstein, Fred L. (1989). “Principal warps: Thin-plate splines and the decomposition of deformations.” *IEEE Transactions on pattern analysis and machine intelligence* 11.6, Pages 567–585.
- Bornard, Raphaël et al. (2002). “Missing data correction in still images and image sequences.” In: *Proceedings of the tenth ACM international conference on Multimedia*. ACM, Pages 355–361.
- Bornemann, Folkmar and Tom März (2007). “Fast image inpainting based on coherence transport.” *Journal of Mathematical Imaging and Vision* 28.3, Pages 259–278.
- Breitenstein, Michael D et al. (2009). “Robust tracking-by-detection using a detector confidence particle filter.” In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, Pages 1515–1522.
- Brostow, Gabriel J et al. (2008). “Segmentation and recognition using structure from motion point clouds.” In: *European conference on computer vision*. Springer, Pages 44–57.
- Brox, Thomas and Jitendra Malik (2010). “Object segmentation by long term analysis of point trajectories.” *Computer Vision—ECCV 2010*, Pages 282–295.
- Butler, Daniel J et al. (2012). “A naturalistic open source movie for optical flow evaluation.” In: *European Conference on Computer Vision*. Springer, Pages 611–625.
- Caelles, S. et al. (2017). “One-Shot Video Object Segmentation.” In: *Computer Vision and Pattern Recognition (CVPR)*.
- Caelles, Sergi et al. (2017). “Semantically-Guided Video Object Segmentation.” *arXiv preprint arXiv:1704.01926*.
- Caesar, Holger, Jasper Uijlings, and Vittorio Ferrari (2015). “Joint calibration for semantic segmentation.” *arXiv preprint arXiv:1507.01581*.
- Calatroni, Luca et al. (2018). “Anisotropic osmosis filtering for shadow removal in images.”
- Cao, Frédéric et al. (2011). “Geometrically guided exemplar-based inpainting.” *SIAM Journal on Imaging Sciences* 4.4, Pages 1143–1179.

- Carreira, Joao et al. (2012). “Semantic segmentation with second-order pooling.” In: *European Conference on Computer Vision*. Springer, Pages 430–443.
- Carvalho, Marcela et al. (2018). “Deep Depth from Defocus: How Can Defocus Blur Improve 3D Estimation Using Dense Neural Networks?” In: *ECCV Workshop - 3D Reconstruction in the Wild*, Pages 307–323. DOI: [10.1007/978-3-030-11009-3\\_18](https://doi.org/10.1007/978-3-030-11009-3_18).
- Chan, Tony F and Jianhong Shen (2005). “Variational image inpainting.” *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences* 58.5, Pages 579–619.
- Chan, Tony F, Jianhong Shen, and Hao-Min Zhou (2006). “Total variation wavelet inpainting.” *Journal of Mathematical imaging and Vision* 25.1, Pages 107–125.
- Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, et al. (2016). “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.” *arXiv preprint arXiv:1606.00915*.
- (2018). “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.” *IEEE transactions on pattern analysis and machine intelligence* 40.4, Pages 834–848.
- Chen, Liang-Chieh, George Papandreou, Florian Schroff, et al. (2017). “Rethinking atrous convolution for semantic image segmentation.” *arXiv preprint arXiv:1706.05587*.
- Chen, Liang-Chieh, Yukun Zhu, et al. (2018). “Encoder-decoder with atrous separable convolution for semantic image segmentation.” *arXiv preprint arXiv:1802.02611*.
- Chen, Yuhua et al. (2018). “Blazingly Fast Video Object Segmentation with Pixel-Wise Metric Learning.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 1189–1198.
- Cheng, Jingchun, Yi-Hsuan Tsai, Wei-Chih Hung, et al. (2018). “Fast and Accurate Online Video Object Segmentation via Tracking Parts.” *arXiv preprint arXiv:1806.02323*.
- Cheng, Jingchun, Yi-Hsuan Tsai, Shengjin Wang, et al. (2017). “Segflow: Joint learning for video object segmentation and optical flow.” In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, Pages 686–695.
- Cheng, Ming-Ming et al. (2015). “Densecut: Densely connected crfs for realtime grabcut.” In: *Computer Graphics Forum*. Volume 34. 7. Wiley Online Library, Pages 193–201.
- Chiu, Wei-Chen and Mario Fritz (2013). “Multi-class video co-segmentation with a generative multi-video model.” In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, Pages 321–328.
- Choi, Sunglok, Taemin Kim, and Wonpil Yu (2009). “Robust video stabilization to outlier motion using adaptive RANSAC.” In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, Pages 1897–1902.
- Chollet, François (2017). “Xception: Deep learning with depthwise separable convolutions.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 1251–1258.
- Ci, Hai, Chunyu Wang, and Yizhou Wang (2018). “Video object segmentation by learning location-sensitive embeddings.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*, Pages 501–516.

- Cordts, Marius et al. (2016). "The cityscapes dataset for semantic urban scene understanding." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 3213–3223.
- Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama (2004). "Region filling and object removal by exemplar-based image inpainting." *IEEE Transactions on image processing* 13.9, Pages 1200–1212.
- Cross, George R and Anil K Jain (1983). "Markov random field texture models." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1, Pages 25–39.
- Cui, Zhaopeng et al. (2017). "Time slice video synthesis by robust video alignment." *ACM Transactions on Graphics (TOG)* 36.4, Page 131.
- Dai, Jifeng, Kaiming He, Yi Li, et al. (2016). "Instance-sensitive fully convolutional networks." In: *European Conference on Computer Vision*. Springer, Pages 534–549.
- Dai, Jifeng, Kaiming He, and Jian Sun (2016). "Instance-Aware Semantic Segmentation via Multi-task Network Cascades." In: *(CVPR) IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Pages 3150–3158. ISBN: 978-1-4673-8851-1. DOI: [10.1109/CVPR.2016.343](https://doi.org/10.1109/CVPR.2016.343).
- Dai, Jifeng, Yi Li, et al. (2016). "R-fcn: Object detection via region-based fully convolutional networks." In: *Advances in neural information processing systems*, Pages 379–387.
- Daisy, Maxime (2015). "pattern based image and video inpainting applied to stereoscopic data with depth map." Theses. Université de Caen Normandie.
- Daisy, Maxime et al. (2015). "Exemplar-based video completion with geometry-guided space-time patch blending." In: *SIGGRAPH Asia 2015 Technical Briefs*. ACM, Page 3.
- Danelljan, Martin et al. (2015). "Convolutional features for correlation filter based visual tracking." In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*, Pages 58–66.
- Dehghan, Afshin, Shayan Modiri Assari, and Mubarak Shah (2015). "Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 4091–4099.
- Demanet, Laurent, Bing Song, and Tony Chan (2003). "Image inpainting by correspondence maps: a deterministic approach." *Applied and Computational Mathematics* 1100.217-50, Page 99.
- Dosovitskiy, Alexey et al. (2015). "Flownet: Learning optical flow with convolutional networks." In: *Proceedings of the IEEE international conference on computer vision*, Pages 2758–2766.
- Drori, Iddo, Daniel Cohen-Or, and Hezy Yeshurun (2003). "Fragment-based image completion." In: *ACM Transactions on graphics (TOG)*. Volume 22. 3. ACM, Pages 303–312.
- Ebdelli, Mounira, Olivier Le Meur, and Christine Guillemot (2015). "Video inpainting with short-term windows: application to object removal and error concealment." *IEEE Transactions on Image Processing* 24.10, Pages 3034–3047.
- Efros, Alexei A and Thomas K Leung (1999). "Texture synthesis by non-parametric sampling." In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Volume 2. IEEE, Pages 1033–1038.
- Evangelidis, Georgios D and Emmanouil Z Psarakis (2008). "Parametric image alignment using enhanced correlation coefficient maximization." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.10, Pages 1858–1865.
- Everingham, Mark et al. (2015). "The pascal visual object classes challenge: A retrospective." *International journal of computer vision* 111.1, Pages 98–136.



- Facciolo, Gabriele et al. (2011). “Temporally consistent gradient domain video editing.” In: *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, Pages 59–73.
- Faktor, Alon and Michal Irani (2014). “Video Segmentation by Non-Local Consensus voting.” In: *BMVC*. Volume 2. 7, Page 8.
- Fan, Lijie et al. (2018). “End-to-end learning of motion representation for video understanding.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 6016–6025.
- Fan, Qingnan et al. (2015). “JumpCut: non-successive mask transfer and interpolation for video cutout.” *ACM Trans. Graph.* 34.6, Pages 195–1.
- Fedorov, Vadim (2016). “Affine Invariant Image Comparison and Its Applications.” Doctoral dissertation. UPF (Universitat Pompeu Fabra).
- Fedorov, Vadim, Gabriele Facciolo, and Pablo Arias (2015). “Variational framework for non-local inpainting.” *Image Processing On Line* 5, Pages 362–386.
- Ferryman, James and Ali Shahrokni (2009). “Pets2009: Dataset and challenge.” In: *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*. IEEE, Pages 1–6.
- Fortun, Denis, Patrick Bouthemy, and Charles Kervrann (2015). “Optical flow modeling and computation: a survey.” *Computer Vision and Image Understanding* 134, Pages 1–21.
- Fragkiadaki, Katerina, Pablo Arbelaez, et al. (2015). “Learning to segment moving objects in videos.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 4083–4090.
- Fragkiadaki, Katerina, Geng Zhang, and Jianbo Shi (2012). “Video segmentation by tracing discontinuities in a trajectory embedding.” In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, Pages 1846–1853.
- Friedman, Itamar et al. (2017). *GyGO: an E-commerce Video Object Segmentation Dataset by Visualead*.
- Fulkerson, Brian, Andrea Vedaldi, and Stefano Soatto (2009). “Class segmentation and object localization with superpixel neighborhoods.” In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, Pages 670–677.
- Galasso, Fabio et al. (2013). “A unified video segmentation benchmark: Annotation, metrics and analysis.” In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 3527–3534.
- Geiger, Andreas et al. (2013). “Vision meets robotics: The KITTI dataset.” *The International Journal of Robotics Research* 32.11, Pages 1231–1237.
- George, Marian (2015). “Image parsing with a wide range of classes and scene-level context.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 3622–3630.
- Ghiasi, Golnaz and Charless C Fowlkes (2016). “Laplacian reconstruction and refinement for semantic segmentation.” *arXiv preprint arXiv:1605.02264*.
- Girshick, Ross et al. (2014). “Rich feature hierarchies for accurate object detection and semantic segmentation.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 580–587.

- Godec, Martin, Peter M Roth, and Horst Bischof (2013). “Hough-based tracking of non-rigid objects.” *Computer Vision and Image Understanding* 117.10, Pages 1245–1256.
- Goodfellow, Ian et al. (2014). “Generative adversarial nets.” In: *Advances in neural information processing systems*, Pages 2672–2680.
- Gould, Stephen, Richard Fulton, and Daphne Koller (2009). “Decomposing a scene into geometric and semantically consistent regions.” In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, Pages 1–8.
- Granados, Miguel, Kwang In Kim, et al. (2012). “Background inpainting for videos with dynamic objects and a free-moving camera.” In: *European Conference on Computer Vision*. Springer, Pages 682–695.
- Granados, Miguel, James Tompkin, et al. (2012). “How not to be seen—object removal from videos of crowded scenes.” In: *Computer Graphics Forum*. Volume 31. 2pt1. Wiley Online Library, Pages 219–228.
- Grossauer, Harald (2006). “Inpainting of movies using optical flow.” In: *Mathematical Models for Registration and Applications to Medical Imaging*. Springer, Pages 151–162.
- Grundmann, Matthias et al. (2010). “Efficient hierarchical graph-based video segmentation.” In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, Pages 2141–2148.
- Guillemot, Christine et al. (2013). “Object removal and loss concealment using neighbor embedding.” *Eurasip Journal on Signal Processing: Image Communication*.
- Güney, Fatma and Andreas Geiger (2016). “Deep discrete flow.” In: *Asian Conference on Computer Vision*. Springer, Pages 207–224.
- Hariharan, Bharath, Pablo Arbeláez, Lubomir Bourdev, et al. (2011). “Semantic contours from inverse detectors.”
- Hariharan, Bharath, Pablo Arbeláez, Ross Girshick, et al. (2014). “Simultaneous detection and segmentation.” In: *European Conference on Computer Vision*. Springer, Pages 297–312.
- He, Kaiming, Georgia Gkioxari, et al. (2017). “Mask r-cnn.” In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, Pages 2980–2988.
- He, Kaiming and Jian Sun (2012a). “Computing nearest-neighbor fields via propagation-assisted kd-trees.” In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Pages 111–118.
- (2012b). “Statistics of patch offsets for image completion.” In: *Computer Vision—ECCV 2012*. Springer, Pages 16–29.
- He, Kaiming, Xiangyu Zhang, et al. (2016). “Deep residual learning for image recognition.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 770–778.
- Held, David, Sebastian Thrun, and Silvio Savarese (2016). “Learning to track at 100 fps with deep regression networks.” In: *European Conference on Computer Vision*. Springer, Pages 749–765.
- Henriques, João F et al. (2015). “High-speed tracking with kernelized correlation filters.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.3, Pages 583–596.
- Henry, Corentin, Seyed Majid Azimi, and Nina Merkle (2018). “Road Segmentation in SAR Satellite Images with Deep Fully-Convolutional Neural Networks.” *arXiv preprint arXiv:1802.01445*.

- Herling, Jan and Wolfgang Broll (2012). “Pixmix: A real-time approach to high-quality diminished reality.” In: *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*. IEEE, Pages 141–150.
- Hong, Seunghoon, Hyeonwoo Noh, and Bohyung Han (2015). “Decoupled deep neural network for semi-supervised semantic segmentation.” In: *Advances in neural information processing systems*, Pages 1495–1503.
- Horn, Berthold KP and Brian G Schunck (1981). “Determining optical flow.” *Artificial intelligence* 17.1-3, Pages 185–203.
- Hosang, Jan et al. (2016). “What makes for effective detection proposals?” *IEEE transactions on pattern analysis and machine intelligence* 38.4, Pages 814–830.
- Hu, Ping et al. (2018). “Motion-Guided Cascaded Refinement Network for Video Object Segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 1400–1409.
- Hu, Yuan-Ting, Jia-Bin Huang, and Alexander Schwing (2017). “MaskRNN: Instance Level Video Object Segmentation.” In: *Advances in Neural Information Processing Systems*, Pages 324–333.
- Huang, Jia-Bin et al. (2016). “Temporally coherent completion of dynamic video.” *ACM Transactions on Graphics (TOG)* 35.6, Page 196.
- Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa (2017). “Globally and locally consistent image completion.” *ACM Transactions on Graphics (TOG)* 36.4, Page 107.
- Ilan, Shachar and Ariel Shamir (2015). “A Survey on Data-Driven Video Completion.” In: *Computer Graphics Forum*. Volume 34. 6. Wiley Online Library, Pages 60–85.
- Ilg, Eddy et al. (2017). “FlowNet 2.0: Evolution of optical flow estimation with deep networks.” In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, Pages 1647–1655.
- ImagineerSystems, Ltd (2014). *Mocha Pro v3.1 software*.
- Ioffe, Sergey and Christian Szegedy (2015). “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” *arXiv preprint arXiv:1502.03167*.
- Jain, Mihir et al. (2014). “Action localization with tubelets from motion.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 740–747.
- Jain, Suyog Dutt and Kristen Grauman (2014). “Supervoxel-consistent foreground propagation in video.” In: *European Conference on Computer Vision*. Springer, Pages 656–671.
- (2016). “Click carving: Segmenting objects in video with point clicks.” *arXiv preprint arXiv:1607.01115*.
- Jain, Suyog Dutt, Bo Xiong, and Kristen Grauman (2017). “Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos.” *arXiv preprint arXiv:1701.05384*.
- Jampani, Varun, Raghudeep Gadde, and Peter V Gehler (2017). “Video propagation networks.” In: *Proc. CVPR*. Volume 6, Page 7.
- Jégou, Simon et al. (2017). “The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Pages 11–19.
- Jia, Jiaya, Yu-Wing Tai, et al. (2006). “Video repairing under variable illumination using cyclic motions.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.5, Pages 832–839.

- Jia, Jiaya and Chi-Keung Tang (2004). "Inference of segmented color and texture description by tensor voting." *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6, Pages 771–786.
- Kalal, Zdenek, Krystian Mikolajczyk, and Jiri Matas (2012). "Tracking-learning-detection." *IEEE transactions on pattern analysis and machine intelligence* 34.7, Pages 1409–1422.
- Kawai, Norihiko, Tomokazu Sato, and Naokazu Yokoya (2009). "Image inpainting considering brightness change and spatial locality of textures and its evaluation." In: *Pacific-Rim Symposium on Image and Video Technology*. Springer, Pages 271–282.
- Kendall, Alex and Yarin Gal (2017). "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" *NIPS*, Pages 5574–5584.
- Keuper, Margret (2017). "Higher-order minimum cost lifted multicuts for motion segmentation." In: *Proc. ICCV*. Volume 2.
- Khoreva, Anna et al. (2017). "Lucid Data Dreaming for Object Tracking." *arXiv preprint arXiv:1703.09554*.
- Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*.
- Kokaram, Anil C, Bill Collis, and Simon Robinson (2005). "Automated rig removal with bayesian motion interpolation." *IEE Proceedings-Vision, Image and Signal Processing* 152.4, Pages 407–414.
- Kokkinos, Iasonas (2015). "Pushing the boundaries of boundary detection using deep learning." *arXiv preprint arXiv:1511.07386*.
- Komodakis, Nikos and Georgios Tziritas (2007). "Image completion using efficient belief propagation via priority scheduling and dynamic pruning." *IEEE Transactions on Image Processing* 16.11, Pages 2649–2661.
- Korman, Simon and Shai Avidan (2011). "Coherency sensitive hashing." In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, Pages 1607–1614.
- Krähenbühl, Philipp and Vladlen Koltun (2011). "Efficient inference in fully connected crfs with gaussian edge potentials." In: *Advances in neural information processing systems*, Pages 109–117.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*, Pages 1097–1105.
- Kroeger, Till et al. (2016). "Fast optical flow using dense inverse search." In: *European Conference on Computer Vision*. Springer, Pages 471–488.
- Krogh, Anders and John A Hertz (1992). "A simple weight decay can improve generalization." In: *Advances in neural information processing systems*, Pages 950–957.
- Kuehne, Hilde, Juergen Gall, and Thomas Serre (2016). "An end-to-end generative framework for video segmentation and recognition." In: *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE, Pages 1–8.
- Le Meur, Olivier, Josselin Gautier, and Christine Guillemot (2011). "Exemplar-based inpainting based on local geometry." In: *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, Pages 3401–3404.
- Le, Thuc et al. (2017). "Motion-consistent video inpainting." In: *ICIP 2017: IEEE International Conference on Image Processing*.



- Leake, Mackenzie et al. (2017). “Computational video editing for dialogue-driven scenes.” *ACM Transactions on Graphics (TOG)* 36.130.
- LeCun, Yann et al. (1998). “Gradient-based learning applied to document recognition.” *Proceedings of the IEEE* 86.11, Pages 2278–2324.
- Lee, Ken-Yi et al. (2009). “Video stabilization using robust feature trajectories.” In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, Pages 1397–1404.
- Lee, Yong Jae, Joydeep Ghosh, and Kristen Grauman (2012). “Discovering important people and objects for egocentric video summarization.” In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, Pages 1346–1353.
- Lee, Yong Jae, Jaechul Kim, and Kristen Grauman (2011). “Key-segments for video object segmentation.” In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, Pages 1995–2002.
- Lempitsky, Victor S et al. (2009). “Image segmentation with a bounding box prior.” In: *ICCV*. Citeseer, Pages 277–284.
- Levin, Anat, Dani Lischinski, and Yair Weiss (2008). “A closed-form solution to natural image matting.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2, Pages 228–242.
- Levin, Anat, Assaf Zomet, and Yair Weiss (2003). “Learning how to inpaint from global image statistics.” In: *null*. IEEE, Page 305.
- Levinkov, Evgeny et al. (2016). “Interactive Multicut Video Segmentation.” In: *Pacific Graphics*.
- Li, Fuxin et al. (2013). “Video segmentation by tracking many figure-ground segments.” In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 2192–2199.
- Li, W. et al. (2016). “Roto++: Accelerating Professional Rotoscoping using Shape Manifolds.” *ACM Transactions on Graphics* 35.4, Pages 1–14. ISSN: 15577368. DOI: [10.1145/2897824.2925973](https://doi.org/10.1145/2897824.2925973).
- Li, Wei, Xiatian Zhu, and Shaogang Gong (2017). “Person re-identification by deep joint learning of multi-loss classification.” *arXiv preprint arXiv:1705.04724*.
- Li, X et al. (2017). “Video Object Segmentation with Re-identification.” In: *The 2017 DAVIS Challenge on Video Object Segmentation-CVPR Workshops*.
- Li, Yi et al. (2017a). “Fully Convolutional Instance-Aware Semantic Segmentation.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- (2017b). “Fully convolutional instance-aware semantic segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 2359–2367.
- Lin, Guosheng et al. (2016). “Efficient piecewise training of deep structured models for semantic segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 3194–3203.
- Lin, Tsung-Yi et al. (2014). “Microsoft coco: Common objects in context.” In: *European conference on computer vision*. Springer, Pages 740–755.
- Ling, Chih-Hung et al. (2009). “Video object inpainting using posture mapping.” In: *Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE, Pages 2785–2788.
- LIRIS, France (no date). “The Visual Object Tracking VOT2014 challenge results” ().
- Liu, Guilin et al. (2018). “Image inpainting for irregular holes using partial convolutions.” *arXiv preprint arXiv:1804.07723*.

- Liu, Ziwei et al. (2015). “Semantic image segmentation via deep parsing network.” In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 1377–1385.
- Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). “Fully convolutional networks for semantic segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 3431–3440.
- Loshchilov, Ilya and Frank Hutter (2016). “Sgdr: Stochastic gradient descent with warm restarts.” *arXiv preprint arXiv:1608.03983*.
- Lucas, Bruce D, Takeo Kanade, et al. (1981). “An iterative image registration technique with an application to stereo vision.”
- Luiten, Jonathon, Paul Voigtlaender, and Bastian Leibe (2018). “PReMVOS: Proposal-generation, Refinement and Merging for Video Object Segmentation.” *arXiv preprint arXiv:1807.09190*.
- Ma, Chao et al. (2015). “Hierarchical convolutional features for visual tracking.” In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 3074–3082.
- Ma, Tianyang and Longin Jan Latecki (2012). “Maximum weight cliques with mutex constraints for video object segmentation.” In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, Pages 670–677.
- Maninis, K.-K. et al. (2018). “Video Object Segmentation Without Temporal Information.” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.
- Mansfield, Alex et al. (2011). “Transforming Image Completion.” In: *BMVC*, Pages 1–11.
- Märki, Nicolas et al. (2016). “Bilateral space video segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 743–751.
- Martínez-Noriega, Raúl, Aline Roumy, and Gilles Blanchard (2012). “Exemplar-based image inpainting: Fast priority and coherent nearest neighbor search.” In: *Machine Learning for Signal Processing (MLSP), 2012 IEEE International Workshop on*. IEEE, Pages 1–6.
- Masnou, Simon (2002). “Disocclusion: a variational approach using level lines.” *IEEE Transactions on Image Processing* 11.2, Pages 68–76.
- Masnou, Simon and J-M Morel (1998). “Level lines based disocclusion.” In: *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*. IEEE, Pages 259–263.
- Matsushita, Yasuyuki, Eyal Ofek, Weina Ge, et al. (2006). “Full-frame video stabilization with motion inpainting.” *IEEE Transactions on pattern analysis and Machine Intelligence* 28.7, Pages 1150–1163.
- Matsushita, Yasuyuki, Eyal Ofek, Xiaoou Tang, et al. (2005). “Full-frame video stabilization.” In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Volume 1. IEEE, Pages 50–57.
- Meyer, Fernand (1994). “Topographic distance and watershed lines.” *Signal Processing* 38.1, Pages 113–125. DOI: [10.1016/0165-1684\(94\)90060-4](https://doi.org/10.1016/0165-1684(94)90060-4).
- Milan, Anton, Laura Leal-Taixé, et al. (2015). “Joint tracking and segmentation of multiple targets.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 5397–5406.
- Milan, Anton, Stefan Roth, and Konrad Schindler (2014). “Continuous energy minimization for multitarget tracking.” *IEEE transactions on pattern analysis and machine intelligence* 36.1, Pages 58–72.

- Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi (2016). “V-net: Fully convolutional neural networks for volumetric medical image segmentation.” In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, Pages 565–571.
- Nam, Hyeonseob, Mooyeol Baek, and Bohyung Han (2016). “Modeling and propagating cnns in a tree structure for visual tracking.” *arXiv preprint arXiv:1608.07242*.
- Nam, Hyeonseob and Bohyung Han (2016). “Learning multi-domain convolutional neural networks for visual tracking.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 4293–4302.
- Newson, Alasdair, Andrés Almansa, Matthieu Fradet, et al. (2014). “Video inpainting of complex scenes.” *SIAM Journal on Imaging Sciences* 7.4, Pages 1993–2019.
- Newson, Alasdair, Andrés Almansa, Yann Gousseau, et al. (2017). “Non-local patch-based image inpainting.” *Image Processing On Line* 7, Pages 373–385.
- Nishihara, Akinori et al. (2011). “Iterative gradient-driven patch-based inpainting.” In: *Pacific-Rim Symposium on Image and Video Technology*. Springer, Pages 71–81.
- Noh, Hyeonwoo, Seunghoon Hong, and Bohyung Han (2015). “Learning deconvolution network for semantic segmentation.” In: *Proceedings of the IEEE international conference on computer vision*, Pages 1520–1528.
- Ochs, Peter, Jitendra Malik, and Thomas Brox (2014). “Segmentation of moving objects by long term video analysis.” *IEEE transactions on pattern analysis and machine intelligence* 36.6, Pages 1187–1200.
- Odobez, Jean-Marc and Patrick Bouthemy (1995). “Robust multiresolution estimation of parametric motion models.” *Journal of visual communication and image representation* 6.4, Pages 348–365.
- Oh, Seoung Wug et al. (2018). “Fast video object segmentation by reference-guided mask propagation.” In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Pages 7376–7385.
- Oneata, Dan et al. (2014). “Spatio-temporal object detection proposals.” In: *European conference on computer vision*. Springer, Pages 737–752.
- Otsu, Nobuyuki (1979). “A threshold selection method from gray-level histograms.” *IEEE transactions on systems, man, and cybernetics* 9.1, Pages 62–66.
- Papazoglou, Anestis (2016). “Video object segmentation and applications in temporal alignment and aspect learning.”
- Papazoglou, Anestis and Vittorio Ferrari (2013). “Fast object segmentation in unconstrained video.” In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 1777–1784.
- Pathak, Deepak et al. (2016). “Context encoders: Feature learning by inpainting.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 2536–2544.
- Patwardhan, Kedar A, Guillermo Sapiro, and Marcelo Bertalmio (2005). “Video inpainting of occluding and occluded objects.” In: *Image Processing, 2005. IICIP 2005. IEEE International Conference on*. Volume 2. IEEE, Pages II–69.
- (2007). “Video inpainting under constrained camera motion.” *IEEE Transactions on Image Processing* 16.2, Pages 545–553.
- Perazzi, F., A. Khoreva, et al. (2017). “Learning Video Object Segmentation from Static Images.” In: *Computer Vision and Pattern Recognition*.

- Perazzi, F., J. Pont-Tuset, et al. (2016). “A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation.” In: *Computer Vision and Pattern Recognition*.
- Perazzi, Federico, Jordi Pont-Tuset, et al. (2016). “A benchmark dataset and evaluation methodology for video object segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 724–732.
- Perazzi, Federico, Oliver Wang, et al. (2015). “Fully connected object proposals for video segmentation.” In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 3227–3234.
- Perez, Patrick, Michel Gangnet, and Andrew Blake (2004). “Patchworks: Example-based region tiling for image editing.” *Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-2004-04*, Pages 1–8.
- Pérez, Patrick, Michel Gangnet, and Andrew Blake (2003a). “Poisson image editing.” *ACM Transactions on Graphics* 22.3, Page 313. ISSN: 07300301. DOI: [10.1145/882262.882269](https://doi.org/10.1145/882262.882269).
- (2003b). “Poisson image editing.” *ACM Transactions on graphics (TOG)* 22.3, Pages 313–318.
- Pinheiro, Pedro O, Ronan Collobert, and Piotr Dollár (2015). “Learning to segment object candidates.” In: *Advances in Neural Information Processing Systems*, Pages 1990–1998.
- Pinheiro, Pedro O, Tsung-Yi Lin, et al. (2016). “Learning to refine object segments.” In: *European Conference on Computer Vision*. Springer, Pages 75–91.
- Pirsiavash, Hamed, Deva Ramanan, and Charless C Fowlkes (2011). “Globally-optimal greedy algorithms for tracking a variable number of objects.” In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, Pages 1201–1208.
- Pont-Tuset, Jordi, Federico Perazzi, et al. (2017). “The 2017 DAVIS Challenge on Video Object Segmentation.” *arXiv:1704.00675*.
- Pont-Tuset, Jordi and Luc Van Gool (2015). “Boosting object proposals: From Pascal to COCO.” In: *Proceedings of the IEEE international conference on computer vision*, Pages 1546–1554.
- Prest, Alessandro et al. (2012). “Learning object class detectors from weakly annotated video.” In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, Pages 3282–3289.
- Pritch, Yael, Eitam Kav-Venaki, and Shmuel Peleg (2009). “Shift-map image editing.” In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, Pages 151–158.
- Qian, Ning (1999). “On the momentum term in gradient descent learning algorithms.” *Neural networks* 12.1, Pages 145–151.
- Ramakanth, S. Avinash and R. Venkatesh Babu (2014). “Seamseg: Video object segmentation using patch seams.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 376–383.
- Reid, Donald (1979). “An algorithm for tracking multiple targets.” *IEEE transactions on Automatic Control* 24.6, Pages 843–854.
- Ren, Shaoqing et al. (2015). “Faster r-cnn: Towards real-time object detection with region proposal networks.” In: *Advances in neural information processing systems*, Pages 91–99.
- Ren, Zhe et al. (2017). “Unsupervised deep learning for optical flow estimation.” In: *Thirty-First AAAI Conference on Artificial Intelligence*.
- Robbins, Herbert and Sutton Monro (1951). “A stochastic approximation method.” *The annals of mathematical statistics*, Pages 400–407.



- Rong, Guodong and Tiow-Seng Tan (2006). "Jump flooding in GPU with applications to Voronoi diagram and distance transform." In: *Proceedings of the 2006 symposium on Interactive 3D graphics and games*. ACM, Pages 109–116.
- Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-net: Convolutional networks for biomedical image segmentation." In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, Pages 234–241.
- Ros, German et al. (2016). "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 3234–3243.
- Roshan Zamir, Amir, Afshin Dehghan, and Mubarak Shah (2012). "Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs." *Computer Vision–ECCV 2012*, Pages 343–356.
- Roth, Stefan and Michael J Black (2005). "Fields of experts: A framework for learning image priors." In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Volume 2. IEEE, Pages 860–867.
- Rother, Carsten, Vladimir Kolmogorov, and Andrew Blake (2004). "Grabcut: Interactive foreground extraction using iterated graph cuts." In: *ACM transactions on graphics (TOG)*. Volume 23. 3. ACM, Pages 309–314.
- Russakovsky, Olga et al. (2015). "Imagenet large scale visual recognition challenge." *International Journal of Computer Vision* 115.3, Pages 211–252.
- Sadek, Rida et al. (2013). "A variational model for gradient-based video editing." *International journal of computer vision* 103.1, Pages 127–162.
- Sánchez, Javier (2017). "Comparison of Motion Smoothing Strategies for Video Stabilization using Parametric Models." *Image Processing On Line* 7, Pages 309–346.
- Scharstein, Daniel et al. (2014). "High-resolution stereo datasets with subpixel-accurate ground truth." In: *German conference on pattern recognition*. Springer, Pages 31–42.
- Schwing, Alexander G and Raquel Urtasun (2015). "Fully connected deep structured networks." *arXiv preprint arXiv:1503.02351*.
- Seguin, Guillaume et al. (2016). "Instance-level video segmentation from object tracks." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 3678–3687.
- Shen, Jianhong and Tony F Chan (2002). "Mathematical models for local nontexture inpaintings." *SIAM Journal on Applied Mathematics* 62.3, Pages 1019–1043.
- Shen, Yuping et al. (2006). "Video completion for perspective camera under constrained motion." In: *null*. IEEE, Pages 63–66.
- Shi, Jianbo and Jitendra Malik (2000). "Normalized cuts and image segmentation." *IEEE Transactions on pattern analysis and machine intelligence* 22.8, Pages 888–905.
- Shih, Timothy K et al. (2008). "Video falsifying by motion interpolation and inpainting."
- Shiratori, Takaaki et al. (2006). "Video completion by motion field transfer." In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Volume 1. IEEE, Pages 411–418.
- Shotton, Jamie, Matthew Johnson, and Roberto Cipolla (2008). "Semantic texton forests for image categorization and segmentation." In: *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, Pages 1–8.

- Shotton, Jamie, John Winn, et al. (2009). "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context." *International Journal of Computer Vision* 81.1, Pages 2–23.
- Shu, Guang et al. (2012). "Part-based multiple-person tracking with partial occlusion handling." In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, Pages 1815–1821.
- SILHOUETTEFX, LLC. (2014). *Silhouette v5 software*.
- Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556*.
- Smith, Leslie N (2017). "Cyclical learning rates for training neural networks." In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Pages 464–472.
- Son, Jeany et al. (2015). "Tracking-by-segmentation with online gradient boosting decision tree." In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 3056–3064.
- Spina, Thiago Vallin and Alexandre Xavier Falcao (2016). "Fomtrace: Interactive video segmentation by image graphs and fuzzy object models." *arXiv preprint arXiv:1606.03369*.
- Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1, Pages 1929–1958.
- Starck, J-L, Michael Elad, and David L Donoho (2005). "Image decomposition via the combination of sparse representations and a variational approach." *IEEE transactions on image processing* 14.10, Pages 1570–1582.
- Strobel, Michael, Julia Diebold, and Daniel Cremers (2014). "Flow and Color Inpainting for Video Completion." In: *German Conference on Pattern Recognition*. Springer, Pages 293–304.
- Sun, Deqing et al. (2013). "A fully-connected layered model of foreground and background flow." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 2451–2458.
- Sundaram, Narayanan and Kurt Keutzer (2011). "Long term video segmentation through pixel level spectral clustering on gpus." In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, Pages 475–482.
- Szegedy, Christian et al. (2015). "Going deeper with convolutions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 1–9.
- Tang, Kevin et al. (2013). "Discriminative segment annotation in weakly labeled video." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 2483–2490.
- Tang, Ming and Jiayi Feng (2015). "Multi-kernel correlation filter for visual tracking." In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 3038–3046.
- Tang, Nick C et al. (2011). "Video inpainting on digitized vintage films via maintaining spatiotemporal continuity." *IEEE Trans. Multimedia* 13.4, Pages 602–614.
- Taniai, Tatsunori, Yasuyuki Matsushita, and Takeshi Naemura (2015). "Superdifferential cuts for binary energies." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 2030–2038.
- Tao, Ran, Efstratios Gavves, and Arnold WM Smeulders (2016). "Siamese instance search for tracking." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 1420–1429.

- Tokmakov, Pavel, Karteek Alahari, and Cordelia Schmid (2016). “Learning motion patterns in videos.” *arXiv preprint arXiv:1612.07217*.
- (2017). “Learning video object segmentation with visual memory.”
- Treml, Michael et al. (2016). “Speeding up semantic segmentation for autonomous driving.” In: *MLITS, NIPS Workshop*.
- Tsai, Yi-Hsuan, Ming-Hsuan Yang, and Michael J Black (2016). “Video segmentation via object flow.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 3899–3908.
- Tschumperle, David and Rachid Deriche (2005). “Vector-valued image regularization with PDEs: A common framework for different applications.” *IEEE transactions on pattern analysis and machine intelligence* 27.4, Pages 506–517.
- Tu, Zhigang et al. (2018). “A survey of variational and CNN-based optical flow techniques.” *Signal Processing: Image Communication* 72. DOI: [10.1016/j.image.2018.12.002](https://doi.org/10.1016/j.image.2018.12.002).
- Tu, Zhuowen and Xiang Bai (2010). “Auto-context and its application to high-level vision tasks and 3d brain image segmentation.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.10, Pages 1744–1757.
- Venkatesh, M Vijay, Sen-ching Samson Cheung, and Jian Zhao (2009). “Efficient object-based video inpainting.” *Pattern Recognition Letters* 30.2, Pages 168–179.
- Voigtlaender, Paul and Bastian Leibe (2017). *Online adaptation of convolutional neural networks for the 2017 davis challenge on video object segmentation*.
- Wang, Lijun et al. (2015). “Visual tracking with fully convolutional networks.” In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 3119–3127.
- Wang, Shu et al. (2011). “Superpixel tracking.” In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, Pages 1323–1330.
- Wang, Wenguan and Shenjian Bing (2017). “Super-trajectory for video segmentation.” *arXiv preprint arXiv:1702.08634*.
- Wang, Wenguan, Jianbing Shen, and Fatih Porikli (2015). “Saliency-aware geodesic video object segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 3395–3402.
- Wang, Wenguan, Jianbing Shen, and Ling Shao (2018). “Video salient object detection via fully convolutional networks.” *IEEE Transactions on Image Processing* 27.1, Pages 38–49.
- Wei, Li-Yi and Marc Levoy (2000). “Fast texture synthesis using tree-structured vector quantization.” In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., Pages 479–488.
- Wen, Longyin et al. (2015). “Jots: Joint online tracking and segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 2226–2234.
- Wexler, Yonatan, Eli Shechtman, and Michal Irani (2007). “Space-time completion of video.” *IEEE Transactions on pattern analysis and machine intelligence* 29.3.
- Winkens, Jim et al. (2018). “Improved Semantic Segmentation for Histopathology using Rotation Equivariant Convolutional Networks.”
- Wu, Yi, Jongwoo Lim, and Ming-Hsuan Yang (2013). “Online object tracking: A benchmark.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 2411–2418.

- Wu, Zhengyang et al. (2015). "Robust video segment proposals with painless occlusion handling." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 4194–4203.
- Xia, Aijuan et al. (2011). "Exemplar-based object removal in video using GMM." In: *Multimedia and Signal Processing (CMSP), 2011 International Conference on*. Volume 1. IEEE, Pages 366–370.
- Xiao, Fanyi and Yong Jae Lee (2016). "Track and segment: An iterative unsupervised approach for video object proposals." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pages 933–942.
- Xiao, Huaxin et al. (2018). "MoNet: Deep Motion Exploitation for Video Object Segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 1140–1148.
- Xiao, Zhitao et al. (2018). "Defect detection and classification of galvanized stamping parts based on fully convolution neural network." In: *Ninth International Conference on Graphic and Image Processing (ICGIP 2017)*. Volume 10615. International Society for Optics and Photonics, 106150K.
- Xie, Saining and Zhuowen Tu (2017). "Holistically-nested edge detection." *International Journal of Computer Vision*, Pages 1–16.
- Xu, Binbin et al. (2017). "Spatio-temporal video completion in spherical image sequences." *IEEE Robotics and Automation Letters* 2.4, Pages 2032–2039.
- Xu, Ning, Brian Price, Scott Cohen, and Thomas Huang (no date). "Deep image matting." In: Xu, Ning, Brian Price, Scott Cohen, Jimei Yang, et al. (2017). "Deep grabcut for object selection." *arXiv preprint arXiv:1707.00243*.
- Xu, Ning, Linjie Yang, et al. (2018). "YouTube-VOS: A Large-Scale Video Object Segmentation Benchmark." *arXiv preprint arXiv:1809.03327*.
- Xu, Zongben and Jian Sun (2010). "Image inpainting by patch propagation using patch sparsity." *IEEE transactions on image processing* 19.5, Pages 1153–1165.
- Yamauchi, Hitoshi, Hans-Peter Seidel, et al. (2003). "Image restoration using multiresolution texture synthesis and image inpainting." In: *null*. IEEE, Page 120.
- Yang, Bo and Ram Nevatia (2012). "An online learned CRF model for multi-target tracking." In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, Pages 2034–2041.
- Yang, Chao et al. (2017). "High-resolution image inpainting using multi-scale neural patch synthesis." In: *null*.
- Yang, Hanxuan et al. (2011). "Recent advances and trends in visual tracking: A review." *Neurocomputing* 74.18, Pages 3823–3831. ISSN: 09252312. DOI: [10.1016/j.neucom.2011.07.024](https://doi.org/10.1016/j.neucom.2011.07.024).
- Yang, Linjie et al. (2018). "Efficient video object segmentation via network modulation." *algorithms* 29, Page 15.
- Yang, Michael Ying et al. (2015). "Temporally object-based video co-segmentation." In: *International Symposium on Visual Computing*. Springer, Pages 198–209.
- Yang, Yanchao, Ganesh Sundaramoorthi, and Stefano Soatto (2015). "Self-occlusions and disocclusions in causal video object segmentation." In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 4408–4416.



- Yi, Saehoon and Vladimir Pavlovic (2015). "Multi-cue structure preserving mrf for unconstrained video segmentation." In: *Proceedings of the IEEE International Conference on Computer Vision*, Pages 3262–3270.
- Yosinski, Jason et al. (2014). "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*, Pages 3320–3328.
- You, Shaodi et al. (2013). "Robust and Fast Motion Estimation for Video Completion." In: *MVA*, Pages 181–184.
- Yu, Fisher and Vladlen Koltun (2015). "Multi-scale context aggregation by dilated convolutions." *arXiv preprint arXiv:1511.07122*.
- Yu, Hongkai et al. (2017). "Loosecut: interactive image segmentation with loosely bounded boxes." In: *Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE, Pages 3335–3339.
- Zagoruyko, Sergey et al. (2016). "A multipath network for object detection." *arXiv preprint arXiv:1604.02135*.
- Zeiler, Matthew D, Graham W Taylor, and Rob Fergus (2011). "Adaptive deconvolutional networks for mid and high level feature learning." In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, Pages 2018–2025.
- Zhang, Dong, Omar Javed, and Mubarak Shah (2013). "Video object segmentation through spatially accurate and temporally dense extraction of primary object regions." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 628–635.
- Zhang, Li, Yuan Li, and Ramakant Nevatia (2008). "Global data association for multi-object tracking using network flows." In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, Pages 1–8.
- Zhao, Hengshuang et al. (2017). "Pyramid scene parsing network." In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Pages 2881–2890.
- Zheng, Shuai et al. (2015). "Conditional random fields as recurrent neural networks." In: *Proceedings of the IEEE international conference on computer vision*, Pages 1529–1537.
- Zhong, Fan et al. (2012). "Discontinuity-aware video object cutout." *ACM Transactions on Graphics (TOG)* 31.6, Page 175.
- Zhu, Xiatian, Chen Change Loy, and Shaogang Gong (2016). "Learning from multiple sources for video summarisation." *International Journal of Computer Vision* 117.3, Pages 247–268.
- Zontak, Maria and Matthew O'Donnell (2016). "Speeding up 3D speckle tracking using PatchMatch." In: *Medical Imaging 2016: Image Processing*. Volume 9784. International Society for Optics and Photonics, 97843W.
- Zweig, Shay and Lior Wolf (2017). "InterpoNet, A brain inspired neural network for optical flow dense interpolation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pages 4563–4572.

