Hierarchical motion-based video analysis with applications to video post-production.

Juan-Manuel Perez-Rua

ANNÉE 2017

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du Signal et Télécommunications

Ecole doctorale MathSTIC

présentée par

Juan Manuel PÉREZ RÚA

préparée au
Centre Inria Rennes - Bretagne Atlantique

Hierarchical motion-based video analysis with applications to video post-production.

Thèse soutenue à Rennes
le 4 décembre 2017
devant le jury composé de :

Luce MORIN
Professor, INSA Rennes / président de jury

Christian WOLF
Associate Professor, INSA Lyon / rapporteur

Anil KOKARAM
Prof. / Head of Department, Trinity College Dublin / rapporteur

Jean-Marc ODOBEZ
Senior Researcher, EPFL / examinateur

Ivan LAPTEV
Research Director, INRIA Paris / examinateur

Patrick PÉREZ
Distinguished Scientist, TECHNICOLOR / examinateur

Patrick BOUTHEMY
Research Director, INRIA Rennes / directeur de thèse

Tomas CRIVELLI
Senior Scientist, TECHNICOLOR / co-directeur de thèse

I like the scientific spirit - the holding off, the being sure but not too sure, the willingness to surrender ideas when the evidence is against them: this is ultimately fine - it always keeps the way beyond open - always gives life, thought, affection, the whole man, a chance to try over again after a mistake - after a wrong guess.

Walt Whitman

Acknowledgements

To my family and friends, and Asia.
P.P., P.B., and Tomas.
Martin, Dmitry, Oriel, Jean, and Charlotte.
It was an adventure.
Thank you all for your support.

Contents

RÉSUMÉ EN FRANÇAIS

Nous présentons dans ce manuscrit les méthodes développées et les résultats obtenus dans notre travail de thèse sur l'analyse du contenu dynamique de scène visuelle. Nous avons considéré la configuration la plus fréquente de vision par ordinateur, à savoir caméra monoculaire et vidéos naturelles de scène extérieure. Nous nous concentrons sur des problèmes importants généraux pour la vision par ordinateur et d'un intérêt particulier pour l'industrie cinématographique, dans le cadre de la post-production vidéo. Les problèmes abordés peuvent être regroupés en deux catégories principales, en fonction d'une interaction ou non avec les utilisateurs :

- **Analyse interactive du contenu vidéo**.
- **Analyse vidéo entièrement automatique**

Cette division est un peu schématique, mais elle est en fait liée aux façons dont les méthodes proposées sont utilisées en post-production vidéo. Ces deux grandes approches correspondent aux deux parties principales qui forment ce manuscrit, qui sont ensuite subdivisées en chapitres présentant les différentes méthodes que nous avons proposées. Néanmoins, un fil conducteur fort relie toutes nos contributions. Il s'agit d'une analyse hiérarchique compositionnelle du mouvement dans les scènes dynamiques. Nous motivons et expliquons nos travaux selon l'organisation du manuscrit résumée ci-dessous.

Introduction

Nous partons de l'hypothèse fondamentale de la présence d'une structure hiérarchique de mouvement dans la scène observée, avec un objectif de compréhension de la scène dynamique. Cette hypothèse s'inspire d'un grand nombre de recherches scientifiques sur la vision biologique et cognitive. Plus précisément, nous nous référons à la recherche sur la vision biologique qui a établi la présence d'unités sensorielles liées au mouvement dans le cortex visuel [Joh70, Gib77, GTJ15]. La découverte de ces unités cérébrales spécialisées a motivé les chercheurs en vision cognitive à étudier comment la locomotion des animaux (évitement des obstacles, planification des

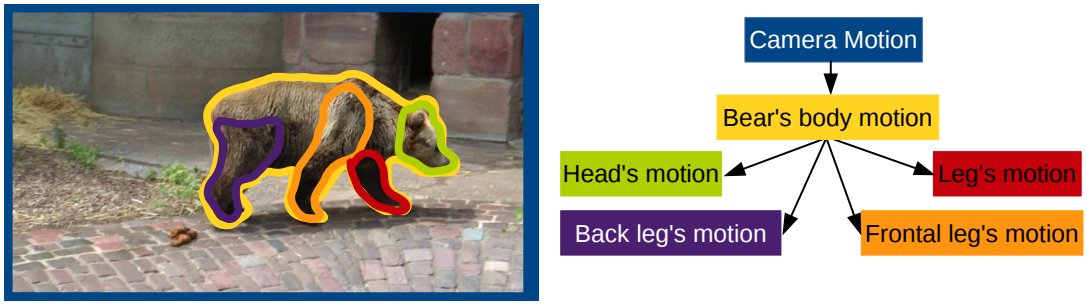


Figure 0.1: **Hiérarchie de mouvement.** Illustration d'une entité mobile avec ses parties mobiles, et de la structure hiérarchique de la composition du mouvement.

chemins, localisation automatique) et d'autres tâches de niveau supérieur sont directement influencées par les perceptions liées aux mouvements. Fait intéressant, les réponses perceptuelles qui se déroulent dans le cortex visuel sont activées non seulement par le mouvement lui-même, mais par des occlusions, des désocclusions, une composition des mouvements et des contours mobiles. En outre, la vision cognitive a relié la capacité du cerveau à appréhender la nature compositionnelle du mouvement dans l'information visuelle à une compréhension de la scène de haut niveau, comme la segmentation et la reconnaissance d'objets [KS83, LHP80].

En raison de ce qui précède, nous nous sommes investis dans l'analyse hiérarchique des éléments en mouvement pour capter des aspects essentiels du mouvement dans des scènes naturelles. En effet, le mouvement dans les vidéos est trés souvent structuré de manière hiérarchique, selon les actions que les éléments de la scène effectuent. En effet, ces éléments partagent des modèles de mouvement de groupe, avec des différences se situant à un niveau d'instance plus fin. En outre, les parties d'un objet peuvent hériter des motifs de mouvement de leur racine commune, mais peuvent se différencier entre elles. L'ensemble du comportement dynamique d'une scène peut ainsi être compris comme relevant d'une nature compositionnelle du mouvement, et cette nature compositionnelle perdure dans le mouvement perçu dans la séquence vidéo. Cette structure compositionnelle du mouvement est illustrée à la figure 0.1 pour une entité animée d'un mouvement articulé.

Dans cette thèse, nous commençons par effectuer des expériences initiales autour de cette idée, puis nous proposons des algorithmes qui exploitent explicitement la nature compositionnelle du mouvement des scènes dynamiques. De plus, nous nous plaçons dans le contexte des tâches liées aux mouvements qui sont utiles en postproduction vidéo. Plus exactement, dans la première partie de ce manuscrit qui concerne le traitement interactif, nous abordons les problèmes de vision numérique suivants :

- **Flot d'objet**. Estimation du mouvement dense au sein d'un objet.

- **Suivi des contours d'objet**. Ou rotoscopie, comme dénommé dans le domaine du post-traitement vidéo.

Ces deux modules fonctionnent dans un mode supervisé par l'utilisateur, tout en partageant une architecture de mouvement hiérarchique, néanmoins peu profonde. Les deux problèmes forment chacun un chapitre.

D'autre part, dans la partie de cette thèse qui considère les méthodes entièrement automatiques, nous abordons les problèmes suivants :

- **Détection d'occlusion**. Estimation d'une carte binaire indiquant quels pixels sont occultés dans une séquence d'images.
- **Segmentation du mouvement**. Segmentation de mouvement hiérarchique dense et regroupement hiérarchique de trajectoires de mouvement.
- **Estimation de modèles de mouvement paramétriques**. Estimation d'un modèle de mouvement paramétrique expliquant le mouvement global de la scène entre deux images successives.

Ces trois sujets constituent trois chapitres.

Analyse vidéo interactive

Nous abordons deux problèmes qui intéressent au plus près le post-traitement vidéo. Il s'agit de tâches liées à l'estimation de mouvement, à la segmentation des objets et au suivi des silhouettes d'objets. Dans le pipeline de post-traitement vidéo actuellement opérationnel, la plupart des tâches sont menées essentiellement à la main, nécessitant de longues heures de travail intensif. Sans aucun doute, il est nécessaire de créer des outils qui puissent faciliter les procédés de production cinématographique.

Les tâches de traitement vidéo de haut niveau qui se déroulent dans le pipeline de post-production vont bien au-delà de l'assemblage général de plans vidéo. L'application d'effets spéciaux visuels (VFX en anglais), l'élimination d'objets indésirables tels que les plates-formes, les fils ou tout autre objet parasite dans la scène, et les modifications ou éditions ciblées, représentent les tâches les plus courantes de haut niveau qui peuvent être exécutées. Au cœur de celles-ci, la segmentation vidéo et la rotoscopie jouent sans doute le rôle le plus important.

Flot d'objet

Dans le pipeline de post-traitement, il est très courant d'avoir des tâches d'édition vidéo spécifiquement appliquées à des objets ou à des surfaces indépendantes. Ces

tâches comprennent une simple transformation des couleurs, un remplissage, ou un remplacement du contenu (par exemple, pour insérer un logo). Si l'objet cible possède un mouvement rigide au cours de la séquence, un suivi simple de l'objet pourrait suffire à propager l'édition d'une image clé vers le reste de la séquence. Cependant, lorsque la surface subit un mouvement non rigide compliqué, un suivi de points denses peut devenir nécessaire. La solution la plus commune consiste alors à exploiter des cartes de flots optique denses. Néanmoins, l'estimation complète du flot optique est très inefficace dans ce contexte. L'objet cible est généralement de petite taille comparé à l'ensemble de l'image. Un algorithme de suivi de points denses restreint aux limites de l'objet cible pourrait non seulement être plus efficace, mais le contexte de l'objet accessible de par l'initialisation de l'utilisateur effectuée dans une image clé peut contribuer à augmenter la précision du suivi. Dans le chapitre 3, nous montrons expérimentalement que c'est en effet le cas en adjoignant à un algorithme de flot optique un suivi d'objet générique.

Flot d'objet simple. Après avoir montré que l'information contextuelle apportée par une technique de suivi d'objet améliore la précision de l'estimation du mouvement, un algorithme de calcul de flot dense à l'intérieur du support de l'objet est proposé. Ce premier algorithme intègre un suivi de points épars et un schéma d'interpolation original pour générer des cartes de flot pour les objets cibles.

Flot d'objet profond. Nous franchissons une étape en proposant d'adapter une méthode de flot optique performante de l'état de l'art, à savoir Deep Flow, et de la combiner avec une méthode de segmentation vidéo originale, ce qui permet d'obtenir très précisément les vecteurs de flot optique sur le support de l'objet cible.

Les deux méthodes présentées sont plus précises pour un suivi de points sur un temps long, comme le démontrent des expérimentations approfondies sur des vidéos réelles. Nos expériences montrent que tenir compte d'une composition hiérarchique de mouvement peu profonde est avantageux pour obtenir des estimateurs précis de mouvement dense et de trajectoires de points.

Rotoscopie

La rotoscopie consiste à délimiter avec précision les contours d'objets le long d'une séquence vidéo. C'est une tâche qui se déroule interactivement dans le pipeline de post-production vidéo. Ces contours sont utilisés pour plusieurs tâches, la composition vidéo étant l'une des plus critiques. La composition vidéo se réfère à la tâche de combiner de manière transparente deux prises de vues différentes.

Outre les frontières d'objet obtenus en utilisant la technique de rotoscopie, les algorithmes de composition vidéo utilisent une carte d'étiquettes connue sous le nom

de trimap, indiquant les régions qui sont clairement en arrière-plan ou au premier plan, et quelles régions sont un mélange flou des deux. Les tâches de rotoscopie et de définition des trimaps nécessitent beaucoup de temps. Nous proposons deux modèles traitant de ces problèmes importants, modèles qui seront développés dans le chapitre 4.

ROAM: un modèle d'apparence enrichie. Nous abordons la rotoscopie comme un problème de suivi de contours d'objet, où l'utilisateur spécifie le détourage initial de l'objet qui doit être suivi. Afin de résoudre ce problème de suivi, nous proposons un modèle englobant des informations globales et locales sur les couleurs et les mouvements, et nous le dénommons ROAM.

Plus précisément, notre modèle définit un contour comme une polyligne fermée intégrant les éléments suivants :

- informations globales sur les couleurs intégrées en exploitant le théorème de Green;
- classifieurs locaux sur l'apparence (couleurs) aux contours de l'objet;
- amers à l'intérieur de l'objet, formant un graphe en étoile.

L'inférence dans ce modèle graphique se fait en deux étapes à l'aide de la programmation dynamique. Entre les deux étapes, nous ajoutons un mécanisme de recalage basé sur des descripteurs temporels liés au mouvement, qui correspondent pour l'essentiel à une structure peu profonde de composition de mouvement. Nous validons notre méthode sur des séquences d'images,servant de benchmarks en segmentation vidéo et en rotoscopie.

ROAM+ avec suivi de trimap intégré. Comme mentionné précédemment, le problème de l'annotation des régions identifiant le premier plan, le fond et la région intermédiaire, c'est-à-dire l'obtention du trimap, est très lié à la rotoscopie. Nous présentons un modèle étendu que nous avons dénommé ROAM+.

ROAM+ exploite les classifieurs locaux de ROAM . En effet notre nouveau modèle utilise la confiance du classifieur pour construire le terme principal d'attache aux données servant à définir la région intermédiaire du trimap. Nous montrons des résultats expérimentaux de notre approche, puis nous dérivons une méthode s'appuyant sur le trimap estimé pour améliorer la précision du contour de l'objet sous-jacent.

Analyse vidéo entièrement automatique

Comme nous l'avons déjà souligné, l'estimation du mouvement, la détection des occultations et la prise en compte de la nature hiérarchique du mouvement dans

les scènes dynamiques observées, sont des leviers importants dans de nombreux problèmes de compréhension de scène. Dans les trois chapitres de cette partie de la thèse, nous étudions des approches entièrement automatiques pour ces questions. Nous commençons par aborder le problème de la détection d'occultations (ou occlusions), puis nous tirons parti du caractère compositionnel du mouvement pour construire une segmentation hiérarchique des scènes vidéo. Enfin, nous introduisons un réseau neuronal convolutif hiérarchique pour l'estimation du mouvement paramétrique. Tous les problèmes abordés présentent un intérêt réel pour le pipeline de post-production vidéo.

Détection d'occultations

La détection d'occultations est plutôt un problème dit de bas niveau en vision par ordinateur. Dans l'état de l'art, le problème en tant que tel est assez négligé. En outre, la détection des occlusions est surtout reconnue comme une question annexe en analyse vidéo. Généralement liée à l'estimation du flot optique ou de cartes de disparité en stéréovision, la détection d'occlusion n'a pas été considérée comme un problème en soi le plus souvent.

En fait, les occlusions sont porteuses d'informations très utiles par elles-mêmes, et elles peuvent apporter des indications précieuses pour les problèmes de segmentation et de suivi temporel dans des vidéos. En outre, ce sont des ingrédients clés dans la propagation réaliste d'éditions à travers une séquence vidéo. Nous proposons de traiter les occlusions aussi indépendamment que possible de l'estimation précise du mouvement. Nous avons élaboré un algorithme, décrit au chapitre 5, qui formule ce problème comme une reconstruction locale spatio-temporelle des couleurs en chaque point entre deux images consécutives. Brièvement, un point est déclaré occulté si sa reconstruction spatio-temporelle ne peut égaler une reconstruction locale purement spatiale dans la première image. Notre nouveau critère de détection d'occlusion s'avère performant sur des ensembles de données synthétiques et réelles. Il est incorporé à un schéma de régularisation pour l'obtention de cartes d'occultation.

Segmentation de mouvement hiérarchique

Dans la même ligne de méthodes entièrement automatiques, nous considérons dans le chapitre 6 l'analyse de mouvement hiérarchique dans des vidéos, tant au niveau de la trame qu'au niveau du plan vidéo. Le mouvement dans une séquence vidéo est naturellement organisé de manière hiérarchique, du mouvement global dû au déplacement de la caméra aux mouvements des objets, puis de leurs parties et sous-parties. Le contenu dynamique au sein d'une vidéo peut ainsi être mieux appréhendé par une analyse compositionnelle du mouvement. Nous nous inspirons de travaux en vision biologique pour construire, à partir de l'analyse de mouvement hiérar-

chique, deux nouveaux algorithmes correspondant à deux tâches importantes en compréhension automatique de vidéo : la segmentation du mouvement dans l'image et le regroupement de trajectoires dans une séquence.

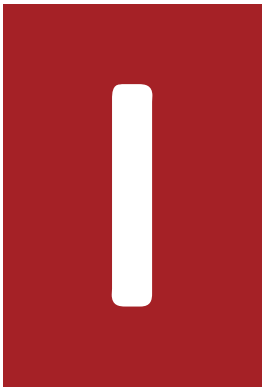
Segmentation du mouvement au niveau de la trame Nous commençons notre analyse hiérarchique en posant le problème de segmentation du mouvement hiérarchique comme l'inférence d'un modèle de champ aléatoire conditionnel. Nous proposons une procédure alternée pour minimiser la fonction d'énergie associée à ce modèle, et nous obtenons une segmentation image par image de la vidéo. Notre méthode met en lumière l'importance d'une analyse hiérarchique du mouvement, qui se traduit dans notre méthode de segmentation comme un problème d'inférence d'étiquettes au sein d'un arbre, et plus précisément comme l'identification de la branche adéquate de l'arbre initial de propositions.

Segmentation du mouvement au niveau du plan vidéo Pour dériver une forme plus utile de segmentation du mouvement et plus stable au cours du temps, nous avons exploré une approche originale s'attachant au regroupement des trajectoires au sein d'un plan vidéo. Les trajectoires de mouvement dense le long d'un plan vidéo nous permettent de gérer ce plan globalement et non plus image par image. Pour atteindre cet objectif, nous avons défini une méthode d'apprentissage de dictionnaire structuré qui convient parfaitement à nos besoins. Nous interprétons le résultat factorisé comme un code qui peut être exploité par la suite pour la segmentation via une procédure de k-moyennes hiérarchique. Nous démontrons expérimentalement que notre méthode fonctionne mieux que plusieurs méthodes existantes, et atteint des performances équivalentes à celles d'algorithmes sophistiqués de clustering de trajectoires vidéos.

Estimation de modèle de mouvement paramétrique

Dans plusieurs modules de cette thèse, nous utilisons une méthode populaire [OB95] pour l'estimation des modèles de mouvement paramétriques. En effet, l'estimation du mouvement est au cœur de plusieurs de nos contributions. Cependant, l'estimation actuelle du modèle de mouvement paramétrique présente quelques limitations lorsqu'il s'agit de déplacements de grande amplitude, de scènes floues et d'apparence ambigüe. Nous proposons un nouveau modèle supervisé basé sur l'apprentissage profond qui pose le problème de l'estimation du modèle de mouvement en tant que problème d'apprentissage de bout en bout. Ainsi, nous espérons que notre modèle apprendra à surmonter les limites des méthodes classiques. Nous fournissons des résultats partiels encourageants sur des données synthétiques, et nous esquissons des travaux futurs dans cette direction.

PART



PRELIMINARIES

CHAPTER 1

GENERAL INTRODUCTION

The fact that images, video, and in general, multimedia are key players in our lives is an undeniable fact. From entertainment and gaming, to social networks, and life-easing applications, the user visual experience has increased dramatically, in quantity, quality and diversity. Interestingly, the number of professionally produced titles has been presenting an immense growth in recent years. In fact, as of September of 2017, the Internet Movie Database (IMDb) has created approximately **4.5 million entries**, including television show episodes, films, documentaries, and video games[1]. Perhaps even more impressively, **300 hours of video are uploaded every single minute to YouTube**, with a total of **3.25 billion hours of video watched per-month**[2]. Furthermore, an increasing number of creators on the online video platform, seeking to increase the quality of the content, and thus, the number of viewers, make use of professional methods for video production [Jar08].

Meanwhile, technology has to accompany the insatiable need of automatic data processing that comes along with the explosion of professionally produced multimedia. Depending on the specific task at hand, which can range from simple editing, to application of visual effects (VfX), video processing techniques might need to be user-guided, or fully automatic. Even in modern times, professional video production that takes place in the film and television industries, and more recently, in the making of high quality on-line video content, requires a large amount of intense manual labor. The process, which is complex and very rarely linear, requires a number of stacked building blocks which solve individual sub-tasks. The exact nature of the building

[1] `http://www.imdb.com/stats`
[2] `https://www.youtube.com/yt/press/statistics.html`

blocks, depends, again, on the desired final product. We simply denominate such workflow as **the post-production pipeline**.

Undoubtedly, there is a clear need of video processing tools that are well adjusted to the particular issues of video data. In the light of the aforementioned explosion of multimedia content, and the amount of manual labor that an individual piece of production requires, new computer vision algorithms ought to be invented in order to alleviate such demand. At the bottom end of the post-production pipeline, automatic video analysis techniques specially related to motion-based descriptors generate the feeds that subsequent stages encompassing higher levels of scene understanding will integrate (e.g., video style transfer, 2D to 3D film conversion, content detection, recognition and classification). We focus on three important items related to dynamic scene analysis in video, namely, motion estimation, occlusion detection, and clustering of motion trajectories. All in all, the analysis of scene dynamics is essential for any system that wants to achieve the level of video processing that is necessary to tackle the complicated tasks that are derived from the post-production pipeline.

1.1 Goals and motivations

This dissertation investigates several methods to tackle fundamental and applied problems in video processing from two fronts: **interactive or user-aided**, and **fully-automatic**.

On the interactive front, we tackle two different but closely related problems: **video object segmentation**, and **rotoscoping**. The first problem consists in labeling all the pixels that belong to the same user-defined object in a video-shot. Similarly, rotoscoping consists in tracking the user-defined outline of a specific object across a video-shot. Interestingly, the set of applications which can be tackled by both methods might sometimes overlap, but one is preferred above the other for different niche applications.

On the other hand, on the fully-automatic front we address three important problems in computer vision that are specially related to video processing and its applications: **detection of occluded regions**, **motion segmentation**, and **parametric motion estimation**. A special emphasis on motion segmentation is made, entering into a deeper analysis of the hierarchical and compositional structure of the motion in a scene.

1.1.1 Hierarchical structure of natural scene motion

Through this manuscript we take a special interest in hierarchical analysis, whether deep or shallow, of moving objects for capturing essential aspects of motion in natural scenes. In fact, motion in natural videos is often nicely structured in a hierarchical

fashion, according to the actions that the scene entities[1] perform during the video sequence. In effect, groups of objects share group motion patterns, with some differentiation at a finer per-instance level. Furthermore, object parts might as well share motion patterns with its parent object, but can differentiate from other parts of the same object. This scene dynamic behavior results in what can be interpreted as a compositional nature of motion in video scenes. One question that we try to answer in the present thesis regarding this matter is then:

1. Can the hierarchical composition characteristic of natural videos be exploited to facilitate video analysis?

To answer this question, we propose a general strategy to analyze video data. As we will introduce in later chapters, this strategy is outlined by the simple observation of hierarchical motion composition. We hypothesize that this concept must serve as lever for computational algorithms to achieve better and deeper understanding of visual data.

Interestingly, ideas that motivated this hypothesis come from a rather varied range of scientific fields: biology, computer vision, and machine learning. From biology, we were inspired by the multitude of research oriented towards understanding how the visual cortex works. In particular, how the brain of numerous animals is able to utilize motion as means to perform foreground separation and to drive visual attention. In fact, it seems that the visual cortex has regions specifically dedicated to perform some form of hierarchical decomposition of motion: from the visual scene, to objects, to object parts [GTJ15]. Furthermore it is currently understood that the brain uses these basic compositional cues to achieve higher level understanding of dynamic visual information [Joh73]. Curiously, these cues empower the brain with the ability to perform very high level tasks like action recognition and people identification. This is the case even when the actors of the actions are out of focus. This seems to be, at least, one the reasons why humans are able to partially recognize people when looking at them only "out of the corner of their eyes" when perceiving certain motion patterns.

From the machine learning and computer vision communities, motivation to our work comes from the recent popularity of Deep Learning with neural networks (stacks of several layers of linear classifiers with non-linear activation functions). More specifically, we consider convolutional neural networks (CNN), deep learning models that were made popular by Krizhevsky [KSH12], when they won the ImageNet image classification challenge in 2012[2] by a considerable margin with respect to any other competitor. Since then, CNNs have been exploited to revisit with great success

[1] We call the entities that take part in a video shot with the generic term "object", although they might be more arbitrary regions of an image that could encompass animated or non-animated things and/or its parts.

[2] `http://www.image-net.org/challenges/LSVRC/`

many other computer vision problems. These successes are motivating. However, more interestingly for this manuscript, learning the parameters of stacked layers of convolutional filters might be interpreted as the learning of hierarchical features from data. Recently, several works [FCNL13, MHYY15] point to this direction, explicitly understanding the hidden convolutional layers in deep models as features that are hierarchically composed from a semantic point of view. The deeper one goes in a CNN architecture, the higher the level of the features. We are driven by the success of such hierarchical models for motion related tasks [MHYY15].

On the whole, this hierarchical view of dynamic content structures this manuscript and the addressed computer vision tasks, which are outlined in the following section.

1.1.2 Interactive and fully automatic methods for video analysis

This manuscript is organized in two main parts. We start by tackling and discussing interactive methods in computer vision for video analysis in the first part, and continue with fully automatic methods in the second part.

Interactive methods We start the discussion on interactive methods with a view on a frequently overlooked problem in computer vision: *object flow*, the dense estimation of motion inside an object region. At the heart of motion analysis, estimation of optical flow (instantaneous apparent motion for each pixel of an image) and visual tracking (estimation of the motion of a certain scene region over time) are classical computer vision problems. Major progresses have been recently achieved on both fronts in the state-of-the-art, with robust and accurate techniques available for each problem. Towards understanding the interaction between pixel- and object-level motion estimation, we introduced a novel per-object motion estimator called *dense object flow*. Long term tracking and segmentation of a moving object are leveraged to improve the precision of dense motion estimation at the object level. This is done by explicitly equipping dense motion estimators with an object tracker. The experiments we present while introducing our object flow method are a first view on the importance of a hierarchical analysis of motion composition.

Second, we take a look at a problem that is closely related to *video object segmentation* and object flow. The tracking of object outlines across a video sequence, or as it is known in the post-processing industry, *rotoscoping*, is a complex problem that requires understanding of motion and appearance information at several levels of abstraction. From a global object track, to the precise tracking of object contours, rotoscoping fits nicely in the framework that encompasses this manuscript, the *hierarchy of motion*. We do this by proposing a model that encompasses global and local motion and appearance features in an unified manner. In other words, the relationship between whole-object and contour motions is exploited in similar way than for the proposed object flow. This is, a shallow hierarchical composition of motion.

Both methods, object flow and rotoscoping, require an initial definition of what constitutes the object of interest. The user is expected to provide the algorithms with *concise initialization*, which can be in the form of strokes or bounding boxes, or exact delineation of the object region. Moreover, both algorithms should allow further interactivity so the users are able to correct the output. These requirements are in line with a number of naturally interactive frameworks, like the post-production pipeline.

Fully automatic methods Similarly to object flow for interactive methods, *detection of occlusions* is very often let aside as a secondary issue in previous state of the art. In this manuscript, we study the problem of estimating occlusions, a problem that traditionally have been inseparably linked to optical flow estimation. The focus of our research in this area was modeling occlusions in such a way that accuracy of occlusion estimation could be, at least partially, separated from accuracy of optical flow estimation. Towards future applications, a twofold advantage of occlusion reasoning is key: while it will feed higher levels of analysis, it is as well a fundamental feature for better and consistent tracking, and object detection. Occlusions are also important in applications like propagation of visual edits in the post-production pipeline.

On the other hand, as an intermediate step towards video related applications, we explore the intrinsic characteristic of scene motion that forms the spirit of this thesis: *motion compositionality*. The dynamic content of physical scenes is largely compositional, that is, the movements of the objects and of their parts are hierarchically organized and relate through composition along this hierarchy. This structure also prevails in the apparent 2D motion that a video captures. Accessing this *visual motion hierarchy* is important to get a better understanding of dynamic scenes and is useful for video manipulation. We explore two settings of the same underlying problem: by modeling *per-pixel* motion composition of parametric motion models, and by studying shot-based composition through *point trajectories*. For the latter, we propose to capture compositionality through a novel structured learning algorithm on point trajectories. The output of our method is a new mid-level motion representation, which we embed within an unsupervised clustering scheme to partition hierarchically the trajectories into meaningful groups. These groups normally, though not necessarily always, keep semantic meaning at various scales: object groups, object instances and/or object parts. In this sense, we leverage intrinsic nature of scene motion to discover moving objects and parts.

Finally, we tackle the problem of *parametric motion estimation*, which begets a large amount of applications in computer vision and robotics. We take advantage of the intrinsic hierarchical processing of deep learning architectures for the problem of motion estimation. Interestingly, parametric motion modeling finds an important place in several chapters of this thesis.

The fully automatic methods that are presented in this thesis can be used as mod-

ular building blocks for more complex video processing tasks. Clusters of trajectories and motion and occlusion maps are valuable for the computer vision community in a general sense. Of course, interactive and automatic methods can both be used together towards solving complex tasks. This is the spirit of the thesis, *a collection of fundamental building blocks for a variety of video processing tasks.*

1.1.3 Motion analysis and video processing for the cinema industry

As mentioned earlier, real-life applications of our research lie at the heart of the post-production pipeline in high-end film and television industries. With a diverse set of objectives (color grading, video editing, video compositing, etc.), film post-production encounters large amounts of tedious and repetitive manual work. All of the proposed methods of this thesis can find their applications in the post-production pipeline. Indeed, the large amounts of manual labor could be assisted by the individual contributions of this manuscript.

1.2 Organization of this thesis

In order to present the findings that came along the preparation of this manuscript in an organized manner, we group them in three major parts. Besides the preliminaries of this thesis, which encompass this introduction, and an overview chapter with discussions of the state of the art, the core of our contributions are presented in Parts II, and III. The two main parts fit well the two types of explored methods: interactive and fully automatic, as introduced earlier in this chapter. Each part is composed of a few chapters corresponding to each one of the tackled problems. More explicitly, the synthetic division of the core of this thesis ends up like this:

- Object-based video analysis with interactive methods: Part II
  - Object flow: Chapter 3. [PRCP15, PRCP16].
  - Rotoscoping and trimap tracking: Chapter 4. [PRMTP17, PRCBP17].
- Fully automatic video analysis: Part III
  - Occlusion detection: Chapter 5. [PRCBP16].
  - Motion segmentation: Chapter 6. [PRCPB16a, PRCPB16b].
  - Parametric motion modeling via deep neural networks: Chapter 7.

The mentioned references for each listed chapter correspond to the publications that came along the preparation of this manuscript.

We discuss the impact of our research and future and ongoing work in Part IV. We provide further support of our contributions when needed, in the annex chapters.

Hopefully the distinction of parts and chapters will not distract the reader from the global message that wants to be delivered: the analysis of videos with a particular interest on motion-related issues, exploiting its natural hierarchical organization.

CHAPTER 2

AN OVERVIEW OF IMAGE MOTION ANALYSIS AND ITS APPLICATIONS TO THE POST-PRODUCTION PIPELINE

In this chapter, we provide an overview of the main tendencies in video processing techniques, with a special emphasis on motion-based methods. We approach this analysis from a view that motivates our thesis work. We first start by discussing the importance of motion analysis from the point of view of a very complex vision system: the human one. This discussion serves both as context and inspiration for understanding the role of motion in computer-based video analysis. We then continue the discussion by introducing current mechanisms for tackling motion in computer systems, while also providing a critique of the current state of the art. The points that we discuss further form the core of this thesis in posterior chapters. However, focused related work and state-of-the-art analysis is further explored in the individual chapters that follow this overview. Finally, we continue by providing an analysis of a real world application of motion analysis techniques: post-production in the film industry. We describe several tasks from this pipeline, explain how computer vision is applied to tackle them, and discuss how current methods could be improved. Interestingly, we will show how the issues studied by human vision experts are relevant for such applications.

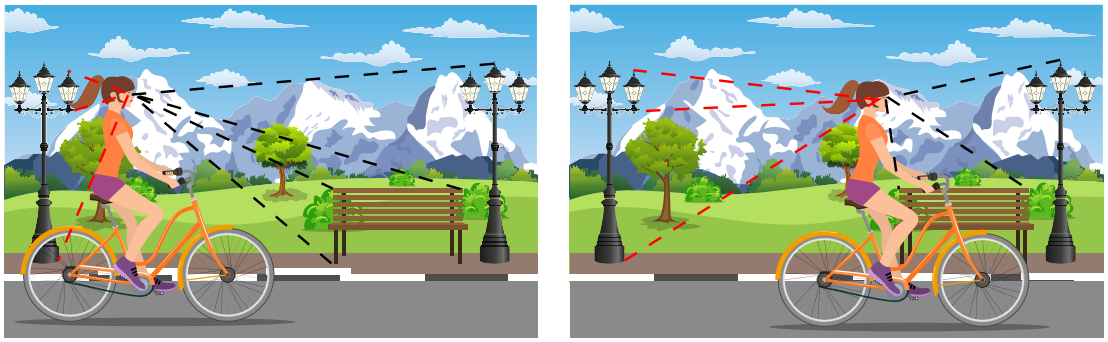


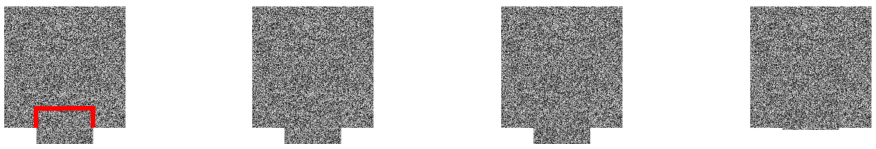
Figure 2.1: An illustration of the static optic array for a biker at two different time frames. A few of the optic rays (dashed lines) are shown at each instant converging to one point. All the possible lines are part of the array, even the ones that cannot stimulate the perception system due to occlusions or size of field of view (in red). The change between left and right frames due to the motion of the biker generates a change in the optic array. The perceptual cues generated from the changing optic array directly influence the behavior of the bicycle rider.

2.1 Motion analysis: a psychological context

The optic array, optical flow and occlusions. Computer-aided motion analysis has deep roots in early psychological vision research. For instance, J. Gibson argued in his 1958 paper [Gib58] that the visual perception system leverages visual motion stimuli for locomotion control. He postulated a theory encompassing the physical array of light rays, and how they converge to a given point in space (the observer) forming what he denominated the *optic array*.

In practical terms, the optic array forms the inputs to the mechanisms for image formation, involving *static visual patterns*. The related term *optical flow* was also introduced by this particular line of work in vision research [1]. It describes the visual stimuli generated by a changing optic array, either by a transforming scene or by a change of the observer point-of-view, producing *changing visual patterns*. In other words, optical flow is correlated with changes of the 3D scene, as apparent from the point of view of the observer. Clearly, it encodes valuable information describing the spatial arrangement of scene elements. An illustrative example of the optic array is shown in Fig. 2.1. These ideas encouraged perception researchers to start moving on from initial interest on stereopsis, which was derived from the discovery of specialized neural units that are related to retinal disparity in the optical cortex (See [BBP67]). In particular, Nakayama and Loomis hypothesized that optical flow could be used by

[1] Although it could be argued that the concept had been previously introduced in the 1940s by a report to the British Air Ministry about landing a plane based solely on velocity cues. As it happened, Calvert [Cal50] published derived ideas in 1950.



(a) (b) (c) (d)

Figure 2.2: Three frames of a video sequence displaying a moving scene with two indistinguishable objects, one moves towards the other causing occlusion (b-d). The occlusion edges are not visible in any of the displayed frames. The edges between the objects, and the depth layers are only visible when watching the moving scene. The occlusion edge is shown in red for the first frame for clarity (a).

sensitive cells with functional properties to outline edges of surfaces at distinct depth layers [NL74].

Ecological psychology. In the meantime, Gibson's constructs led to the formation of what is known as "ecological psychology", which describes the *direct* relationship of human (and animal) behaviors and his/her environment [Gib50]. One of the main theorems from this theory, as eloquently summarized by G. Johansson [Joh70] is: "The visual system receives optically no specific information about the environment when it is in a static state but a moving organism obtains such information due to its motion"[2].

Gibson defended the idea that pure geometrical information about the environment could not be obtained without analyzing motion through what he called *accretion* and *deletion* [GKRW69]. The latter term, deletion, is related to the gradual occlusion of a moving object as it passes behind another. On the other hand, accretion conveys somewhat the opposite meaning, the gradual appearing of an object emerging from behind another one. Gibson *et al.* [GKRW69] showed with simple experiments that one does not need to see a line to *perceive* a line. When observing two distinct objects, one camouflaged with the other, it can become rather difficult to distinguish edges between them at isolated time instants. For some scenes it might be not possible at all, as it is illustrated in Fig. 2.2. However, if one of the objects moves, occluding or dis-occluding the other object, an edge becomes apparent. Occluding edges, then, offer information about the relative ordering of objects, and gives an idea of depth, even under monocular settings. Gibson considered this effect as an essential aspect of human visual perception.

[2]Ecological psychology was fairly controversial since its foundation. A public debate between Gibson and Johansson through publications dedicated to each other made the field very lively in the 1970's, *e.g.*, [Joh70, Gib70, Gib77].

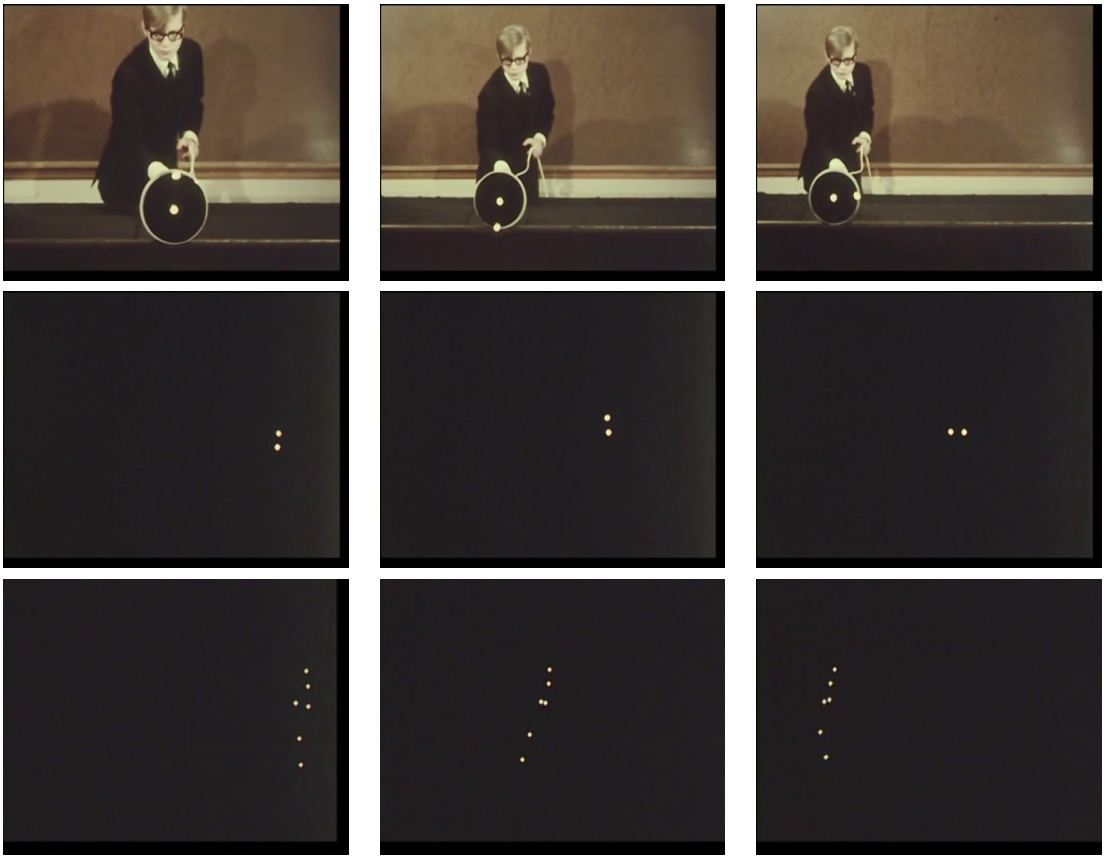


Figure 2.3: Experimental setting of the work developed by G. Johansson. First row: the light bulb setting used to generate percepts of motion organization; in this case the bulbs are located in the center and extreme of a wheel. Second row: room light is turned off, producing a scene composed only of moving dots; observers of the scene perceive a main translation motion in composition with a secondary vertical oscillation motion. Third row: the same setting is used on human motion. Interestingly, the percept of biological locomotion is only vivid when observing the moving scene, and less so for the displayed individual pictures.

Through time, multiple authors contributed to the field by showing the various ways humans leverage visual motion information to improve their understanding of the surrounding environment. For example, Burkel showed [Bur52] that even under total occlusions the human visual system can track targets with smooth trajectories. In another example, as described by Kellman and Spelke [KS83], children exploit motion cues for object localization and differentiation under contradictory textural information.

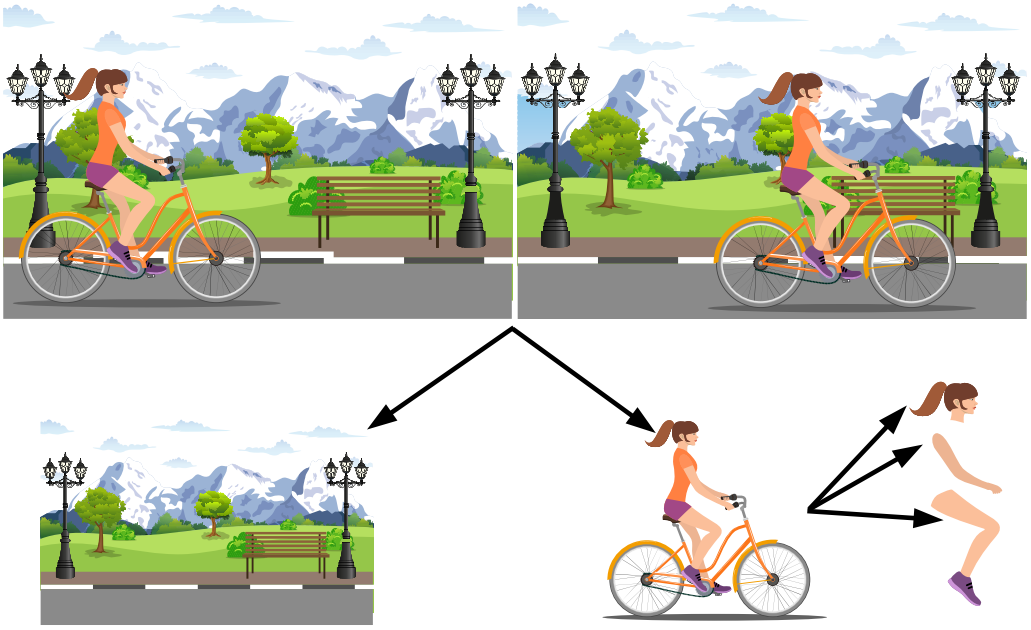


Figure 2.4: An observer of a moving scene composed of biker in park leverages motion cues to understand its hierarchical organization. Background and main moving objects are separated in a first layer of the hierarchy. Object parts are further separated in a second layer.

Natural motion structure. An interesting line of work in psychological vision aiming at understanding the details of human perception was started in the 1950's by G. Johansson with the publication of his PhD dissertation [Joh50] [3]. Johansson showed experimentally how complex percepts can be activated by simple punctual motion patterns. He devised experiments to demonstrate the perceptual organization of simple motions, which can be observed in the first two rows of Fig. 2.3.

Departing from his own initial analysis, Johansson studied visual motion patterns derived from living organisms locomotion [Joh73]. The most revolutionary of his experiments consisted in putting small light bulbs along the articulations of human actors in a dark room. The filming conditions allowed to create videos composed only of a very sparse set of moving white spots. Observers of the video that were not involved in its creation were able to recognize human figures and fully identify the performed actions (See Fig. 2.3). Furthermore, observers reported the vivid sensation of observing human actions with as few as five points located in the hips and legs,

[3] Johansson's work focused on understanding how the vision system is able to isolate useful patterns from a chaotic world full of irrelevant data. Thus, his work is an example of the Gestalt psychology movement that originated in the Berlin School of Experimental Pshycology around 1922.

irrespective of walking direction. This means that strong percepts of human actions can be activated when its motion is observed, even when there is no other visual information available. Interestingly, this is related to the posterior discovery of the fact that visual motion patterns alone are used to separate objects and object parts from the rest of the scene. It can be argued that motion itself can lead to understanding of a hierarchical layer-based organization of the visual scene. For instance, a group of people running in the same direction can be understood as a hierarchy of nested motions, departing from the global group motion, to individuals, to their parts: an arm, a leg (See Fig. 2.4). Current works on psychological vision further confirm the utility of hierarchical and compositional analysis of motion for the human perception system in the processes of parsing scenes into objects [GTJ15].

Computer vision. Without any doubt, motion analysis plays an important role in living organisms with the capacity to see. It is interesting how low level visual motion cues end up building a very complex understanding of the surrounding world. As Gibson and his theory of ecological psychological predict, complex behaviors are derived from these basic cues. It is fascinating to observe that computer vision started in a rather similar way to our notion of how the human visual system works. First with a deep interest in stereo vision to understand the structure of the world from a single point of view [MP76, LK81], to optical flow [HS81] for understanding of the geometry of a changing world. Explicit use of optical flow for achieving higher level of understanding in computer vision came a bit later. Where computer vision evolution differs from psychological vision is how long it took to understand the importance of occlusions as informative descriptors of events in a visual scene.

Furthermore, there have not been many attempts to exploit the hierarchical and compositional nature of visual motion for improving the understanding of visual scenes. It could be the case that, from a computational point of view, motion hierarchies might not be very useful. Indeed, the fact that the brain works in a specific way does not mean necessarily that it is also the best way for computers to approach a given problem. Nonetheless, a form of hierarchical analysis has already been visited in previous computer vision literature. In particular, optimization methods for optical flow have benefited of what can be interpreted as shallow architectures of motion composition. This means that motion composition have indeed collaborated to discover better methods for motion estimation, but, can it provide deeper insights about video?

In the next section we discuss more deeply these concerns about the current state-of-art in computer-based video motion analysis.

2.2 Computer-based motion analysis

In this section, as in the rest of this manuscript, we study visual motion analysis from the perspective of a monocular camera, moving or static. Of course, vision is not limited to this particular set-up, with options spanning many types of technologies: multi-view stereo [HZ03], light-field [Lev06], and many other types of non-conventional vision devices [MRFS12]. The use of those technologies in television and film, nonetheless sporadic, has been demonstrated in the past. For example, Kitahara *et al.* [KSA+01], exploited a large set of calibrated cameras to perform camera sweeps during replays of the XXXV Super Bowl and the largely acclaimed 1999 science fiction film "Matrix", by the Wachowskis. However, with the ubiquity of mobile cameras, and the fact that non-conventional imaging has not yet enjoyed a strong success in professional video production, it is safe to say that monocular video is the most common computer vision set-up in this context.

There are many imaginable tasks for a computer to perform from video. So-called video analysis is a broad term that refers to extracting valuable information from a video-shot. In our field, we usually refer to the extracted information elements that carry a lot of semantic value as "higher level". For example, video object segmentation is considered to be a higher level vision problem if the extracted segments are required to be tagged with semantic labels (*e.g.*, pedestrian, car, road, sky, tree, etc.). On the contrary, if the task is to segment a video into background and moving objects, without specifying anything else about the nature of the objects, the problem is considered to be of lower level than the former example. More simply, the higher the complexity of the semantic concept, the higher the level of the video task. On the other hand, lower level or early vision is closer to the physical properties of the scene elements (edges, shape, structure, etc.). This denomination does not imply that one task might be more "important" or "easier" than another one. However, it is usually related to the way systems are assembled to solve those tasks. Higher level tasks might build, although not always explicitly, upon lower level ones.

In this manuscript, our main focus is on motion-based video analysis. Being naturally related to the physics of the 3D world, motion analysis can be considered a low level vision task. However, it could be used, and it usually is, to build complex systems performing all kind of high-level vision tasks. We start by briefly discussing the role of the most common ways to characterize motion in the state of the art, while giving a glimpse on current issues and open questions.

2.2.1 About optical flow

As in psychological optics, one of the lowest level or most fundamental tasks in computer-based video analysis is the computation of optical flow. Drawing inspiration from psychologists, computer scientists identified very early the importance of such

a problem. In computer vision, optical flow is defined as the field of displacement vectors for all the pixels in one image with respect to another image of the same video shot. The first attempts to solve this problem go back as far as 1981, with the methods proposed by Horn and Schunck [HS81], and almost in parallel by Lucas and Kanade [LK81][4].

Optical flow is by definition an ill-posed problem [PTK85]. In 2D, it requires to solve for two variables per-pixel with a single constraint. The most common hypothesis that constrains the optical flow problem is that color or intensity structure is preserved in time. This hypothesis is commonly denominated optical flow constraint (OFC), or "brightness constancy assumption" when only intensities are used in the formulation[5]. In addition to this hypothesis some form of regularization is needed to relax the problem, and it is often made explicit by assuming that neighbor pixels move together with similar direction and magnitude. The optical flow constraint and the smooth motion assumption lie at the hearth of most optical flow methods in the form of an energy functional with two terms, a data-driven one, and a smoothness one. Flow fields are extracted given a pair of frames by minimizing such energy.

Naturally, first attempts to solve optical flow focused on recovering motion fields of small magnitude (a couple of pixels). Indeed, the optical flow constraint and simple regularization are not well suited for large displacements. Example optical flow maps are show in Fig. 2.5[6].

In order to allow optical flow to deal with large displacements, many methods adopted a hierarchical approach on the image scale space [Ana89, BAHH92, HPB94]. In the image scale space, coarser images (lower resolution) are used to estimate longer range motions. Estimated motion at a coarse scale is fed to subsequent finer scales as initialization for optimization. At the finest scale, only motion details are estimated. In a sense, the hierarchical approach becomes an incremental refinement method. This approach, nonetheless, still suffers of limited effectiveness for long displacements of image regions that are blurred out at coarse scales (small moving objects and motion details). Moreover, with the strong regularization and loss of motion detail, motion discontinuities are commonly blurred out by previous algorithms. The first attempt to solve optical flow for both long displacements and motion discontinuities was done by Heitz and Bouthemy [HB93], by incorporating intensity and edge-based local

[4] Initially conceived as an image registration method for stereo, the popular algorithm is mostly recognized today as an early solution to optical flow.

[5] There is an implicit problem with the hypothesis: intensity or color information may vary for many reasons for corresponding points of a pair of image frames. Optical flow does not necessarily correspond with the projection of 3D motion onto the image plane according to this definition.

[6] Observe that the used images in Fig. 2.5 are computer-generated. Synthetic datasets have played a major role in the motion estimation state-of-the-art. This is due to the fact that it is very hard to annotate ground-truth optical flow, making evaluation and learning difficult. CGI datasets allow exact annotation of ground-truth optical flow by projecting the 3D motion vectors onto the image plane. Other annotation methods using expensive set-ups have also been proposed (*e.g.*, [SS03, GLSU13]).

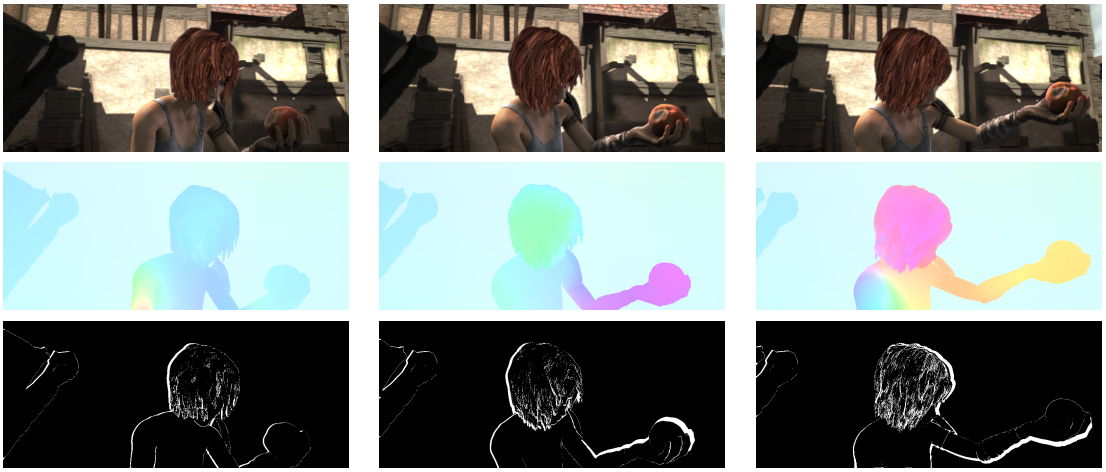


Figure 2.5: Optical flow (second row) and occlusion (third row) maps corresponding to an animated movie video shot (first row). The portrayed film, *Sintel*, is an open source movie produced by the Blender team[7], and used by computer science researches of the *Max Planck Institute for Intelligent Systems* [BWSB12b] for extracting ground-truth annotation and creating a benchmark for optical flow evaluation. The optical flow maps in the second row are color-coded by magnitude and angle of the flows vector in the *HSV* color space, and correspond to the projection of ground-truth 3D motion onto the image plane. The occluded pixels are white-colored, while the visible ones are black.

measurements. Other authors have proposed higher order priors to account from optical flow modeling under specific conditions. For example, Allain *et al.*, [ACC09] jointly estimate velocity and so-called disturbance potential to model crowd dynamics.

More recently, optical flow methods have relied on descriptor matching to account for long range displacements [BBM09, WRHS13]. In these methods, optical flow computation is divided in two steps: matching features and integration of the matches with the optical flow energy functional, and minimization of new energy. Multi-scale analysis can be used in both matching and minimization steps, leading to a mixed domain hierarchy of motion.

Although not explicitly, the different hierarchical approaches for optical flow are related to the hierarchical motion structure described in psychological vision research [GTJ15]. However, the coarse-to-fine method misses the available higher level information that can be extracted from such hierarchy, and it is only used to improve motion estimation itself.

Other lines of work have helped optical flow methods gain accuracy over the years. The use of robust estimators [BA93, OB95, SRB10], relying on locally computed

[7] `https://durian.blender.org/`

parametric motions and piece-wise inference [BA96, YL15] are examples of this. Furthermore, combining dense with piece-wise parametric motion estimation, which can be seen as shallow hierarchies, also played an important role [JBJ96, MP98]. Advances in combinatorial [LRRB10] and variational optimization [BBPW04] are other key ingredients in this progress. More insights into this evolution can be found in a recent survey [FBK15b]. Moreover, optical flow estimation is one of the tasks that have been successfully tackled by deep end-to-end trainable architectures. Very recently, *FlowNet 2.0* [IMS+17], which builds on *FlowNet* [DFI+15], has achieved results that are almost state-of-the-art and runs in real time, granted the availability of specialized hardware (GPUs). Finally, deep learning approaches in combination with classical elements of non-learnable optical flow algorithms have also been proposed. In particular, Thewlis *et al.* [TZTV16] proposed a trainable architecture which replaces Deep Matching[8] with a convolutional neural network. Furthermore, unsupervised deep learning with the OFC used as loss have also been proposed by [JHD16]. A network designed by Bailer *et al.* [BVS17] works as a CNN-based patch matching algorithm achieving state-of-the-art results across multiple benchmarks. The architecture comprises hierarchical analysis by learning features at different image scales.

2.2.2 Occlusions: issue or tool?

Given a pair of images of the same scene taken at two distinct instants and/or viewpoints, an occluded region is a set of pixels that are visible only in one of them. Considering the progress of the video sequence in time, one talks of occluded or dis-occluded pixels. Effectively, occlusions are caused by moving objects passing in front of other scene elements with respect to the point of view of the observer (See Fig. 2.5 for example occlusion maps). Regions of the image that go out of frame are also considered occluded since they are not visible in subsequent frames. As mentioned in previous section, occlusions themselves are informative. For example, motion direction of occluding edges hints at the direction of moving objects [LHP80]. Occlusions also help distinguishing between different similarly-patterned objects, and provide information about scene layers and depth ordering [GKRW69].

In computer vision, occlusions are mostly understood as an obstacle towards accurate estimation of optical flow [FBK15b], or stereo disparity [SCD+06]. Indeed, occlusions are always analyzed in the light of optical flow or disparity estimation. This is why most algorithms deal with occlusions in the same way as outliers. For simplicity, methods equate occlusions regions to regions where motion estimation confidence is low. Due to this, a large number of methods utilize one of two simple criteria: an occlusion must exist if *the appearances of two matched points in the image*

[8] *DeepFlow* [WRHS13], contrary to what its name might convey, does not make use of deep learning. However, a (non-learned) layered stack of patch-based convolutional responses lies at the core of its matching algorithm.

plane differ strongly; or if *the difference between the forward and backward optical flows at corresponding points* is large.

The main issue with these criteria is that it already assumes that accurate flow fields are readily available even for the occluded regions. Naturally, this leads to an alternative chicken and egg problem. Furthermore, by considering occlusions only as a by-product or residue of optical flow estimation, while measuring occlusion accuracy based only on the accuracy of the corresponding flow field, occlusions are tied to the estimation issues of optical flow. An error in optical flow estimation will likely lead to an error on occlusion detection, and vice versa.

It is worthwhile to take a deeper look into the occlusion problem and deal with the aforementioned issues, considering its direct applications to post-production. One of such applications is video editing, where a scene element might be wanted to be replaced, covered or inpainted. The replacement or editing has to be automatically inferred for a set of frames given annotation in only a few of them. In this case, methods have to deal with occlusions accurately so the editing is propagated through frames in a credible way [CCR+12].

2.2.3 Other ways to characterize motion: object tracking and motion trajectories

Optical flow and occlusions are not the only ways to characterize scene motion. Object tracks and motion trajectories are two other common ways to approach motion analysis in video. We first discuss briefly on object tracking.

Object tracking. The problem of tracking an object through a video sequence can be defined as finding its position at every frame given initialization by means of automatic object detection or interactive user input. In the usual set-up, the user selects the object that he/she wants to be tracked by proving a bounding box, or a delineation of the object, on the first frame of the video sequence. Notoriously, video object segmentation is a form of object tracking. This is particularly true under the interactive framework for video segmentation, where the user selects the object or region that he/she wants to be segmented out from a video sequence. The relationship between background and foreground modeling for background subtraction, a general form of video segmentation, and object tracking has been discussed in the literature, for example, by [Ziv04]. Object tracking is a very interesting problem on its own, and the state of the art has evolved in many ways since the first papers started appearing in the first half of the 1980's [GGF+80, SM82]. For the sake of consistency, in this manuscript we focus on a particular issue of object tracking methods, while the reader is encouraged to refer to recent surveys [YJS06, LHS+13, SCC+14] and benchmarks

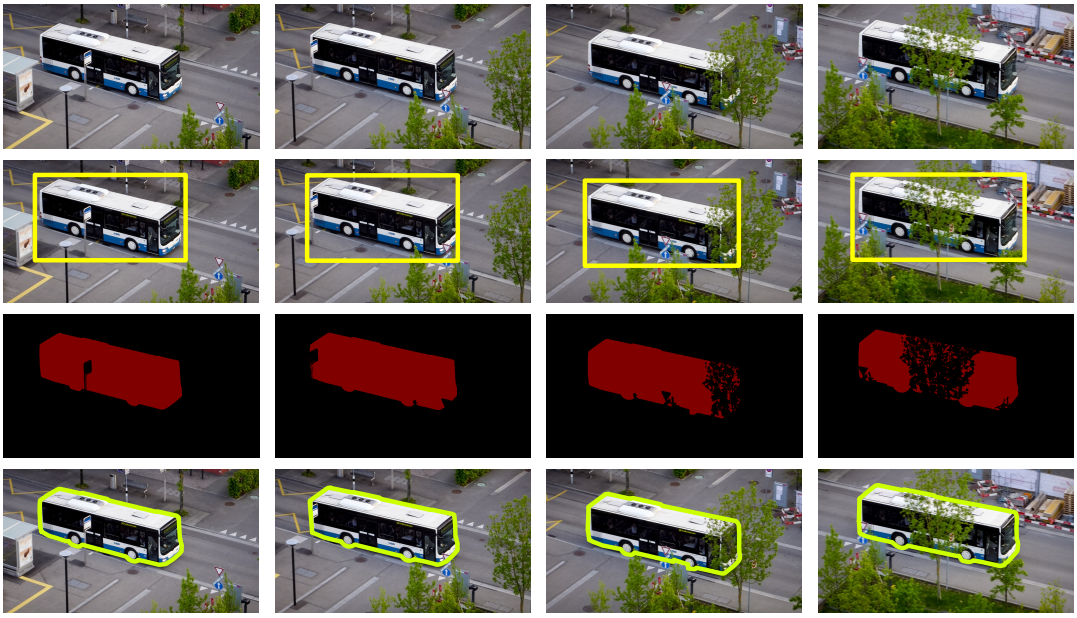


Figure 2.6: Several faces of tracking related tasks in computer vision. First row: a video sequence depicting a moving vehicle. Second row: object tracking by bounding box. Third row: video object segmentation. Red color indicates that a pixel belong to the target object. Fourth row: closed contour tracking (yellow curves), known as rotoscoping in the post-processing pipeline.

[WLY13] for deeper details on object tracking. Figure 2.6 exemplifies the different computer vision tasks that are similar to object tracking.

Although object tracking and optical flow are deeply related problems, there has been very little research on their relationship in the state of the art. A few exceptions of this observation come from works aiming at guiding object tracks, either as bounding boxes [YMSM95] or as evolving contours [MKT98], by leveraging precomputed optical flow maps. Surprisingly, the opposite direction relationship has not been explored. However, recent works in optical flow do find the benefit in partitioning the scene in various regions so that motion is more easily estimated, as in [YL15] for arbitrarily-shaped regions, or as in [FBK16b] for patches. For the RGBD case (RGB colour image plus a depth channel) D. Sun *et al.* propose to explicitly segment the scene into motion layers [SSP15] in alternation with motion estimation. In all the optical flow methods previously mentioned the authors compute parametric motion models for the image regions, with further refinement to account for local distortions.

If a global motion estimation of arbitrary regions is helpful for optical flow estimation at the pixel level, then an object track should be able to serve in a similar way. This should be even more since the initial partition of the region that defines the object is

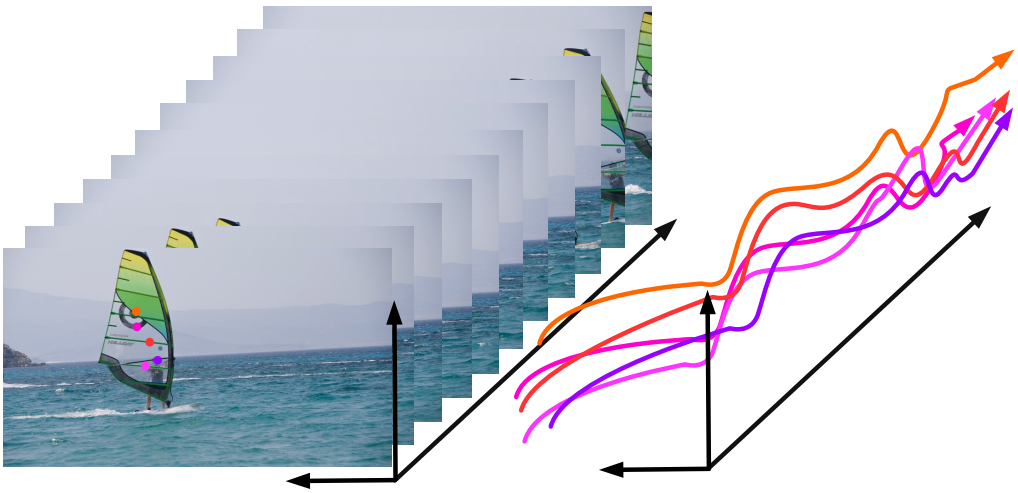


Figure 2.7: Illustration of motion trajectories. Left: A moving scene displayed as an image volume for clarity of the presentation. A few colored points are indicated. Right: Trajectories along the time dimension for each one of the colored points.

already provided by the user or detection algorithm.

Indeed, a bounding-box model for object tracking already assumes that the global object motion can be described by a parametric model, while optical flow inside object boundaries can be seen as a deformations of the global parametric motion. The problem with previous works in this line [SLP14, YL15] is that the arbitrary appearance-based segments do not necessarily correspond with object boundaries. It is the same case for motion layers [SSP15]. We believe that there is a need to explore the relationship between the global object track, and the local object deformations.

Motion trajectories. So far, with the motion characterization tools that we have discussed it is still difficult to analyze motion in a way that encapsulates the whole video and the whole image space altogether. Optical flow is defined in a frame-to-frame basis, while object tracks would require multiple instances to be able to globally describe the visual scene.

With the objective of analyzing scene motion in a holistic manner, several authors have proposed the use of long-term motion trajectories. Basically, motion trajectories are point tracks across a video sequence (See Fig. 2.7). Ideally, the tracks are as dense as possible, and with a large time-span. They were introduced by M. Shah *et al.* [SRT93] in 1993, and later became powerful tools for video analysis.

Trajectories are often computed by integrating optical flow fields over time[9]. How-

[9]This method has become increasingly popular in recent years. It offers denser and longer term

ever, due to discrete nature of the image arrays, occlusions, and inaccuracies of optical flow, simple integration leads to drifting and tracking errors [CFC+15]. Several methods have been proposed to deal with these issues, including filtering techniques [SRT93], trajectory completion [RLF12], forward-backward optical flow checking [SBK10a], and multi-step flow field combination [CCRP12, CCR+12].

Since trajectories capture global motion patterns of a video shot, they have been used for a great deal of high-level computer vision applications: automatic video segmentation [OMB14], anomalous behavior detection [WMS10], action recognition [WKSL11]. However, the question of how they are involved in the compositional and hierarchical nature of motion is still unexplored. As shown by Morris and Mohan [MT09], clusters of trajectories contain rich information about the scene. Distinct trajectory clusters extracted from visual scenes usually correspond with independently moving objects. Thus, we know that trajectories are able to encode at the very least a shallow hierarchical structure of the scene. It is fair to ask then if it is possible to go further and obtain deeper structures composed not only by objects but object parts and finer scene elements.

2.3 Applications to post-production

YouTube®, Netflix®, Amazon Prime®, and many other platforms offering different types of high-end multimedia services have been growing steadily in the last years. The relatively new services, together with the traditional multimedia distribution channels produce a gigantic amount of content. All this product is expected to keep up with the expectations of high visual quality that final users have grown used to.

Post-production plays an important role in the cycle of high quality multimedia production. It operates *after* the video and audio recordings are performed. The process is usually very involved, requiring even more time and budget that previous steps in multimedia production. The reasons why a film might be edited after shooting are plentiful. Apart from choosing and assembling shots and scenes into a story, the reasons include promoting a certain product, preserving identity, applying visual effects, correcting errors, and many others. Fig. 2.8 shows examples of post-production video processing for application of visual effects on the film "Iron Man"[10] and the TV show "Game of Thrones". In this section, we provide insights about the *video* post-production pipeline, and how computer-vision techniques play a role in it.

tracks than standard feature detection [TK91] and tracking [S+94].

[10] This movie has everything. Great rock n' roll guitar riffs, outstanding visual effects, the thoughtful story of Tony Stark, a genius millionaire philanthropist that mastered artificial intelligence, robotics, and quantum physics in a cave, while being held prisoner of a terrorist group. In the humble opinion of the author of this manuscript, this movie might be the greatest of all time. Although this should be taken with a grain of salt, the author also enjoyed "Twilight", which does not speak too positively of his value as film critic.

Figure 2.8: Visual effects in film and television industries make strong use of computer vision techniques. Top: Background-foreground separation for blending recorded video with CGI in the acclaimed TV show "Game of Thrones", by D. Benioff and D. Weiss. Bottom: Motion capture for realistic blending of an animated armor with a real human actor in the film "Iron Man" (2008), by J. Favreu.

The post-production pipeline. Interactive tools for video editing revolutionized the film industry. Before the invention of the Moviola[11], the first machine for film editing, in 1924, simple post-production tasks like cutting a shot could not be done while parallelly watching the film. This made the process very time intensive, requiring careful observation of the film strips. Being able to watch the film while processing it allowed editors to select cut-points more precisely. Moviola machines were popular until mid 1970's, when they were replaced by more advanced versions of it [Cas13]. However, not much changed in the way film post-production was done. For many years, post-production tasks were related to selection of shot intervals, shot transitioning, cropping, and general assembling of shots. If actual changes had to be made to the content of the film, editing was made directly into the strips by hand.

In terms of technical innovation, the biggest breakthrough in the last decades was probably the one that came along with the introduction and popularization of digital film technologies. Digital technologies, initially introduced merely as a change in the medium for multimedia delivery, opened the door for computers to improve the post-production experience. With the appearance of new software packages and the maturing of computer graphics and vision technologies, post-processing of films was

[11] Iwan Serrurier, inventor of the Moviola, originally intended his creation to be used as a home film viewer. Due to its high price, low sales, and recommendations from people involved in the film industry, Serrurier modified the design so that the machine could be used for editing.

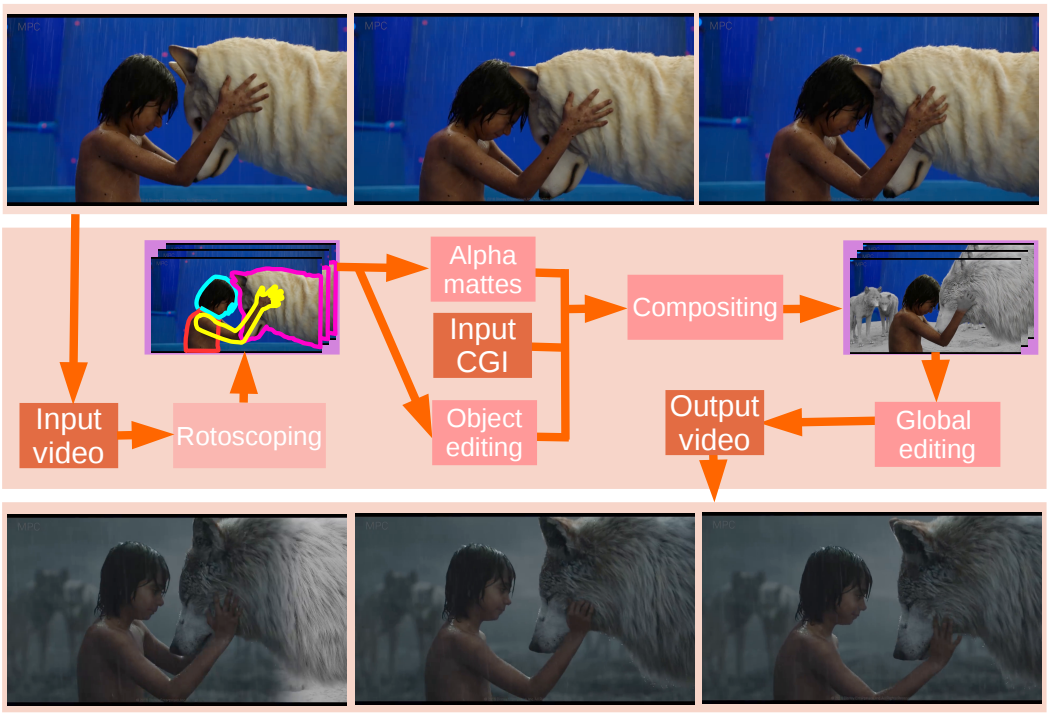


Figure 2.9: An illustration of the non-linear post-production pipeline for the 2016 Disney film "The jungle book", by J. Favreu (for which large part of post-production was conducted by Technicolor/MPC). Most of the scenes were filmed in studio with a blue-screen and stuffed animals. Computer generated graphics (CGI) were used for most of the background and animals. The CGI was composited with the filmed shots, and further image and video processing techniques were applied for simulating weather conditions.

afforded with higher complexity tasks. For instance:

- Color grading.
- Video inpainting (removal of rigs, wires or general scene elements across the video).
- Object editing: replacement or modification of scene objects.
- Video compositing: blending together elements from two different video shots.
- 2D to 3D film conversion.

Another important advantage of computer aided post-production was the introduction of a non-linear paradigm, which proved to be most efficient in complex post-production projects. Having to perform more than one task for a single video

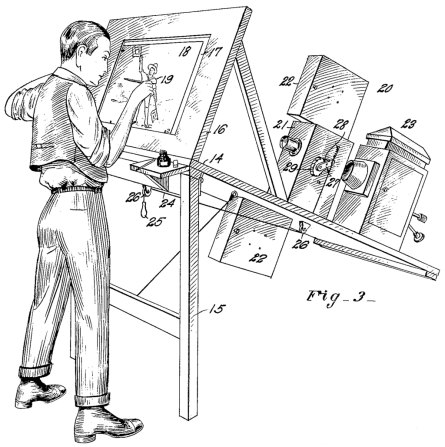


Figure 2.10: The rotoscoping procedure for animated films. Image taken from the 1917 US patent [Fle17] by Max Fleischer, the inventor of the rotoscope.

shot, editing artists benefit from block-based non-linear processing tools, enabling the re-use of important blocks like optical flow and occlusion maps for different tasks on the same video shot. An illustration of this paradigm is shown in Fig. 2.9.

About rotoscoping. Taking into consideration the nature of the tasks that are usually requested from an editing professional, the most important building block in video post-production is arguably rotoscoping. It consists of the delineation of a scene object across all the frames of a video shot. The resulting outlines are known as roto-curves. This operation is at the hearth of many of the previously mentioned post-processing tasks. For example, the roto-curves are input for video inpainting, object editing, video compositing, and 2D to 3D film conversion. In general, any object-centered operation benefits from the accurate track of the object of interest.

The term rotoscoping has historical connotations in the animated film industry. It first referred to the process of drawing animated characters by delineating over real footage of human or animal actors. The purpose of this was to facilitate the arduous process of animation. It was involved in motion capture, where real human or animal silhouettes would guide the per-frame drawing process[12]. An schematic drawing of the first machine used for rotoscoping in animation is shown in Fig. 2.10. Besides being used as motion capture technique, rotoscoping was sometimes also used to guide the design of animated characters. Sometimes, the animated characters were stylized versions of the real actors. Because of this, some controversy surrounded the use of rotoscoping in the animated film industry, some part of it even undermining the artistic value of such process. However, it later came to be an accepted form of

[12]Wikipedia has a comprehensive list of films that were produced using the rotoscoping technique `https://en.wikipedia.org/wiki/List_of_rotoscoped_works`

artistic and professional animation. Eventually, film producers discovered that the same technique could be used for extraction of *mattes*, a cut-out of the objects from a video shot, to use them with a different background (compositing).

Departing from such primitive methods for compositing, the introduction of chroma keying facilitated the extraction of mattes for realistic blending of scene elements. Chroma keying consists on filming with a solid background color, usually green or blue, decreasing the complexity of matte extraction by simple color-based segmentation of the scene. However, even when chroma keying is used, rotoscoping is still necessary for precise delineation of the objects of interest [Bra11]. The process is performed in two steps. First, *garbage* and *holdout* mattes are extracted. They define loose regions of the scene that are to be removed or kept, respectively, from the final matte. Second, a more accurate delineation of the indicated objects under the guidance of initial mattes is performed.

Professional rotoscoping artists rely on very basic tools for extracting the final mattes. In fact, most of the work is done by hand, even in current days. It usually starts by selecting one or more keyframes of a video shot. The roto-curves are then delineated carefully by hand for those keyframes. Then, the produced roto-curves are transferred or interpolated from keyframes to subsequent frames, and modified interactively to the likeness of the artist. The artist then iterates over all the frames of the video shot until the desired level of temporal consistency and contour accuracy is achieved. The whole process is excruciating, taking hours for short video shots, and several days for longer ones.

If the tasks that follows rotoscoping is compositing, the required mattes are not binary maps as in “inside of the roto-curve” and “outside of the roto-curve”. Due to motion blur, the discrete nature of digital images, among other reasons, mattes must also contain information about the appearance mixture of foreground and background for a given pixel. To do so, artists take the roto-curve, and use it to guide the generation of another curve indicating where are the problematic regions, and where are the pure foreground and background regions. The mixture coefficient of background/foreground appearance inside the curves is blindly interpolated. Computer vision algorithms for alpha-matting are available, but not widely used. These algorithms leverage a similar object, usually called *trimap*, that indicates the regions where alpha values must be computed.

The computer vision approach. From the computer vision point of view, rotoscoping is an interesting problem. It requires accurate tracking of an object along a video sequence, while also allowing some level of interaction or online supervision. A computer-assisted rotoscoping tool must still allow the user to select the objects to be cutout from the scene, and must permit online correction of the roto-curves. In principle, one could be tempted to tackle the rotoscoping problem by using video

segmentation methods of the state of the art.

A first promising family of approaches are the hierarchical over-segmentation methods that provide a partition of the video at several scales, *e.g.*, [GKHE10]. However, since the segments obtained by such methods rely on appearance-based clustering, there is no guarantee that the actual borders of the object of interest coincide with the computed video segments at any scale. Nonetheless, this kind of algorithms could still be used to track the region that contains the object of interest by leveraging user interaction at the finer segmentation scale. Even with these considerations, the problem of allowing the user to correct the final segmentation still remains. Even with the finer segments, the borders are not guaranteed to correspond to the desired contours.

Closer to the objective of computer-assisted rotoscoping, interactive video segmentation algorithms that allow the user to make scribbles indicating foreground and background regions can be found in the literature [BS07, MPWSH16a]. These methods integrate user scribbles into a graph-based energy functional that when minimized, results in an optimal partition of the graph structure into foreground and background regions. Interestingly, these methods, not being guided by optical flow, suffer from temporal inconsistencies. To remedy this behaviour, X. Bai *et al.*, proposed a method that leverages object-based optical flow maps, together with local contour classifiers to take a better hold on the object edges while also maintaining certain temporal consistency [BWSS09]. Alternatively, Baugh and Kokaram empowers the pixel classification cost with a temporal prior stemming from long term motion trajectories [BK10]. However, there is an important issue that all video segmentation methods share. Being based on a global energy for labeling pixels, interactive video segmentation methods are not convenient for professional scenarios. The behavior of such global functions is not easy to predict, and some scribbles that intend to correct some labeling issue may cause some new problems on a different region of the scene. Similarly, snake-based tracking methods like [PSK01] still lack on flexibility for temporal interactivity, and long object displacements.

For all the reasons previously mentioned, rotoscoping artists still rely on simple interpolation of curves for the frames in between keyframes. Of course, interpolation of contour points does not provide with any real tracking capabilities. With this in mind, Agarwala *et al.* [AHSS04], proposed a new energy functional encompassing both the shape of the objects defined by the user in the keyframes, and object tracking. However, probably as a cause of the energy formulation, the tracking capabilities still let much work to the user, in particular for long range motions. Similarly, the recent work of [LVS+16] focuses on providing a comfortable interpolation framework for intermediate frames while tracking with a strong shape prior. In other words, there is still a lot of room for improvement of computer-assisted rotoscoping methods. Current methods make a strong emphasis in facilitating shape interpolation of the

roto-curves, but lack on accuracy of roto-curves tracking.

Propagation of edits along the video shot. Although rotoscoping is a core tool for post-production, not all of the tasks require such precise and time consuming delineation of object contours. For some tasks, the artists might decide to simply perform edits in a keyframe basis, and propagate them with an available dense point tracker. The edits that often recur in post-production are changing the color of an object, inserting a logo or any other graphic object replacement. For very simple cases, consisting of planar objects that do not exhibit strong non-rigid deformations, robust region-based tracking techniques like [BM04b] might suffice. Furthermore, slightly more complex cases can be handled with constrained non-rigid warping methods as in [LY05]. For most of the cases, surface deformations are too complex to be handled by such parametric models. In that scenario, optical flow might be a better solution. More exactly, dense motion trajectories, which can encompass the extended temporal domain that optical flow lacks. However, as mentioned in previous section, temporal integration of optical flow is often error-prone, quickly accumulating drift error along a sequence. Crivelli *et al.* [CCR+12] proposes a method to accurately integrate optical flow along a video sequence, by considering multiple flow fields for intermediate frames. In their work, they manage to handle occlusions and drift, very important issues for video editing[13].

Considering the nature of the simple edits, usually spanning a small area with respect to the image size, and the large temporal extension of the video-shots, which can span from a couple of seconds to a few tens, the robust optical flow integration methods seem to be excessively complex in time and memory. Regrettably, dense motion estimators for a user-specified region (as opposed to full frame motion estimation) are hardly explored in the current state-of-the-art. Furthermore, taking into consideration that most of the common edits are constrained to the spatial region that contains a scene object, integrating region tracking with dense motion estimation could deliver a useful tool for the post-production pipeline.

Keyframe detection. In the post-processing pipeline, keyframes are chosen from the video-shot to facilitate rotoscoping or edit propagation through the rest of the sequence. In computer vision, automatic keyframe detection algorithms have been exploited for simultaneous localization and mapping [ZLY10], tracking [PLW11], and cross-linking of TV news [ZNCT07]. In the context of computer-aided video post-processing, keyframe detection might be a worthwhile effort towards fully automatic rotoscoping. However, based on the characteristics of what corresponds to be a "good keyframe", it can be argued that given a correct understanding of the hierarchical

[13] Handling occlusions while propagating edits along motion flows is important in order to avoid propagation errors. Specifically, when the edited object is occluded, the edit must not be visible.

nature of scene motion, detecting keyframes might render to be unnecessary. Indeed, what an artists strives for when choosing keyframes is minimizing the amount of work on delineating new object parts or new object instances. For that reason, a good keyframe is one where most of the objects of interest are (clearly) visible. If that is the case, keyframe detection can be replaced by a partition of the video shot in moving objects and object parts. Naturally, a video processing step that loosely localizes all the moving objects and parts in the scene would bring a large performance boost on the rotoscoping task than simply detecting keyframes.

2.4 Discussion

Motion analysis in computer vision has evolved in similar ways to our understanding of the role of vision in human psychology. Motion can be used by computers, as it is by humans, in direct ways to guide complex tasks. In the context of motion-based computer aided analysis of video, we have found that there is still a lack of understanding on the role of occlusions. They can be used directly in important applications in the post-production pipeline, like propagation of edits and improved trajectory tracks. Moreover, a very important process of the human vision, the compositional analysis of motion for understanding the intrinsic scene hierarchy has not yet been tackled by the computer vision literature. We believe that this analysis, as it happens in biological vision system, might help computers to better understand the elements that compose a scene. Furthermore, motion compositionality might also have a role in the post-production pipeline. The rest of this manuscript focuses on the exploration of these ideas.

PART

INTERACTIVE VIDEO ANALYSIS

CHAPTER 3

OBJECT-AWARE DENSE MOTION ESTIMATION: OBJECT FLOW

Motion estimation in image sequences is classically addressed under one of the two following forms: estimation of optical flow (instantaneous apparent motion over the whole image) and visual tracking (locating a certain scene region over time). Major progresses have been recently achieved on both fronts, with robust and accurate techniques available for each problem. However, these problems are mostly studied as if they were independent, while they address in fact two faces of the same reality. This chapter analyzes the benefits and consequences of combining tracking methods for estimation of per-object dense motion trajectories. We show experimentally that studying the global motion of an object will benefit the motion estimation accuracy of sample points inside the larger structure.

3.1 Introduction

Object tracking and optical flow estimation are two important components of any computer vision toolbox, and have been focus of great research efforts, leading to significant progress in the last years (e.g. [BSL+11, WLY13]). The visual tracking problem consists in estimating the position or pose of a target object or scene fragment in every frame of a video sequence given its initial position. On the other hand, the optical flow estimation between a pair of frames consists in finding a displacement vector for each pixel of the first image, hence a *dense motion* or *displacement field*. Even though for several applications a complete (*i.e.*, for every pixel) motion-field

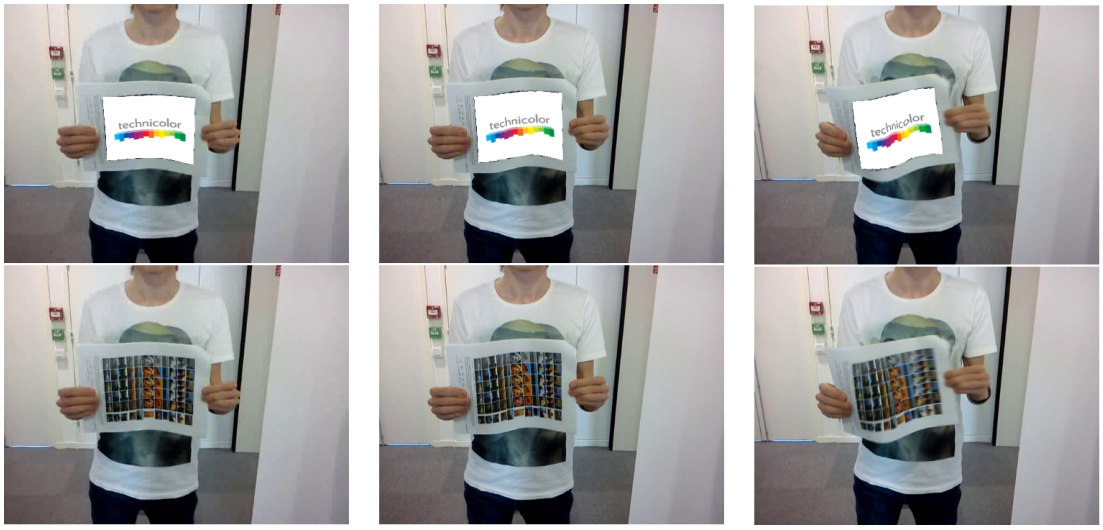


Figure 3.1: An input video sequence (Bottom) is to be modified by replacing a specific object with a different texture (Top).

is needed, other applications, like human action recognition [AS10], video editing [CCR+12] or structure-from-motion [FZ08], may only focus on an object of interest and thus, only a subset of motion vectors is required.

In such scenarios, combining optical flow and object tracking would be useful, especially to improve the precision of the motion description at the object level. To illustrate, we present an application to video manipulation in Figure 3.1, where an user-selected region of the image is replaced by a virtual graphical element. The per-pixel tracking is necessary to mimic real deformations on the original surface across the video. It is known that even with modern optical flow approaches, long-term dense motion estimation remains a challenge[CCR+12, BM11]. A high-level comparison of both problems is given in Table 3.1, which defines the scenario. While optical flow is usually considered for frame-by-frame per-pixel motion estimation, object tracking aims at localizing an object of interest along a video sequence. Our goal in this chapter is to investigate the results and benefits of combining both scenarios within an object-aware flow computation pipeline.

At large, object trackers provide a more robust, longer term motion estimation featuring an holistic description of the target, specially after recent works based on tracking-by-detection approaches [WLY13, BYB09, HST11]. On the other side, they lack the (sub) pixel precision and spatial reasoning of dense optical flow estimators, as well as a deeper use of contextual cues for motion vector estimation. Nevertheless, these two techniques are rarely studied in conjunction. Although optical flow has been widely used as a motion cue for object tracking, e.g., [S+94, OGP06], feeding a dense motion estimator with tracking information, to the best of our knowledge, has

Table 3.1: High-level characterization of object tracking, dense optical flow estimation and proposed object-aware flow estimation.

	Tracking	Object-aware flow	Optical flow
Spatial Span	Object-level	Pixel level in a ROI	Pixel-level (dense)
Computation	Real-time	Possibly real-time	Rarely real-time
Temporal Span	Long term	Long term	Frame pair
Type of problem	Detection problem	Inverse problem	Inverse problem

not yet been investigated. In particular, this study may be of value for applications like object-centric video editing and reconstruction.

In order to show the value of this symbiosis of techniques, we first devise and present experiments that assess the accuracy of simple optical flow estimators when equipped with object motion, the latter obtained through object tracking. Furthermore, we show that such a simple idea is worth taking into account while reasoning on scene dynamics and thus justifies a deep analysis. In a practical setting, we show the different benefits of having bounding box and foreground trackers in a pipeline for point tracking and object-based optical flow estimation. This set-up shows promising results for the problem of dense long-term point tracking that undergoes long-range motions and possibly occlusions.

We call Object Flow the set of flow vectors for every pixel inside the support of an object, or a semantically meaningful part of an object. This carries obvious similitude with the full-frame motion estimation counterpart, *i.e.*, optical flow (See Table 3.1).

3.2 Related work

This section is not intended to provide a complete and exhaustive overview of the current state of the art and historical evolution of optical flow or object tracking methods. We encourage the reader to refer to more specialized studies and surveys that reflect this evolution and propose deeper analysis of the current state of advance, including [BSL+11, FBK15b, SRB10] for optical flow and [WLY13, YJS06, LHS+13] for object tracking. We think that a proper analysis on combining these two motion estimation methods have yet to be done.

Exploiting context for optical flow estimation. We can find in the state of the art some hints on the importance of combining the displacement (in parametric form) of semantically meaningful entities with a low-level pixel-based motion estimation. Some examples include works that study the relationship between scene layers and motion estimation. In [HAP94] the authors propose a framework for layered motion

estimation that requires the pre-computing of an affine-model-based motion segmentation [WA93] and layer separation, with subsequent residual motion estimation (the initial affine motion models are intended to minimize the amount of residual motion calculation). Similar approaches include the early work of [BJ96] which uses color information to fit parametric motion models to every segment of the image. In the same line, but putting emphasis on regularized fitting with a Total Variation approach, we can find [XCJ08].

More recent approaches on layered motion estimation include [SSB12], which simultaneously solves for layer segmentation and optical flow estimation, achieving good results on both tasks. It is interesting to see that modern techniques of layered motion estimation such as [SSB12] also choose to assign parametrized motion models to the computed layers. This is evidence of the fact that the semantically meaningful structures of the image can be used to improve the overall motion estimation, even when they are assigned with simpler motion models. In fact, very recently, the concept of piecewise parametric models is revisited and used to successfully solve motion estimation [YL15].

The layered and segmentation related approaches to computing optical flow are often motivated by the weakness of classical methods with respect to the preservation of motion discontinuities or occlusion handling. For example, layering at the superpixel level for occlusion handling is explained in [SLP14]. However, it is known that these are not the only concerns in optical flow computation. Long displacements are major obstacles for motion reasoning between pairs of frames that still need to be addressed. Usually, multi-scale approaches are the preferred way of improving optical flow algorithms with respect to this matter [WM95, BBPW04], allowing the optimization schemes to escape local extrema with flow solutions of more than a few pixels of displacement magnitude. More recently, using rich features for dense point matching [LYT+08] has opened an interesting way of handling long range motions, through feature matching. The work presented in [BM11] exploits descriptors matching in a variational approach allowing it to handle large displacements. Furthermore, [WRHS13] proposes a matching scheme that is tailored for the optical flow problem, generating very good results in very challenging scenes. In general, feature matching and motion layers can be seen as different ways of utilizing more global information of the area that surrounds a given point to compute its motion. These algorithms, however, are not well adapted for single object computations where only the motion of points belonging to one or more objects are needed. A recent and insightful work on such scenarios is [MG15], which takes some assumptions on the types of motion that are found in automotive applications to compute scene flow (from a stereo setup) with emphasis on objects of interest (other vehicles).

Optical flow for object trackers. On the other hand, it is well known that using optical flow can boost more global motion estimations. For instance, in the literature we can find that pixel-wise motion estimation can be used to guide object trackers [BJ98, HN99, MB94], to track deformable surfaces [DM00] or to segment objects in video [OMB14]. Oddly enough and to the best of our knowledge, the use of object tracking to guide point trajectories inside said object, has not been exploited.

Finally, as standard tools for point tracking we can recall the work of Shi, Tomasi and Kanade [S$^+$94, TK91] (and its descendants), which basically consists in finding well conditioned image corners and computing their locally affine motions for every pair of frames. As a tracked feature eventually disappear (or its tracking becomes unreliable) after a few frames, the corner detection has to be restarted. An automatic scheme to reject outliers during the feature selection was proposed as an extension to the initial method in [TFTR98]. These methods, however, do not solve the problem of having very short-term tracks for each point. Furthermore, the tracked points are processed independently, without taking into account spatial context constraints. Inspired by the latest and exploiting a combination of Lucas-Kanade [LK81] and Horn-Schunck [HS81], Birchfield and Pundlik [BP08] propose a more global method to track features and edges.

Constructing short-term point trajectories is indeed useful for applications like action recognition [UIM08], but it may not be sufficient under other constraints and applications. In order to allow longer term feature tracking, Rubistein *et al.* [RLF12] apply a graph-based method for matching previously generated tracks. This approach is useful for off-line applications where more common feature tracking approaches can be used initially, and under a post-processing stage the method unifies tracks that were separated by occlusions or a target loss. However, an online method that can obtain a dense tracking for points that belong to a specific region of an image, while also being reasonably robust to occlusions, is yet to be proposed.

3.3 The compositional nature of dense motion

There are essential aspects in the way motion naturally manifests in video, some of these are essentially overlooked in tracking applications. For instance, the fact that the motion of the scene is composed by a set of separable and usually independent moving entities could be used as prior for motion estimation algorithms. Furthermore, even though some problems in computer vision require the estimation of motion and trajectories of individual points in an image, these motions are affected by larger scale entities and its estimation could benefit from this knowledge. Scale here is to be understood not as spatial scale but as different hierarchical levels in the scene composition.

In order to pin down this idea, let us consider the motion vector $\mathbf{f}_\mathbf{x} = (u_\mathbf{x}, v_\mathbf{x})$

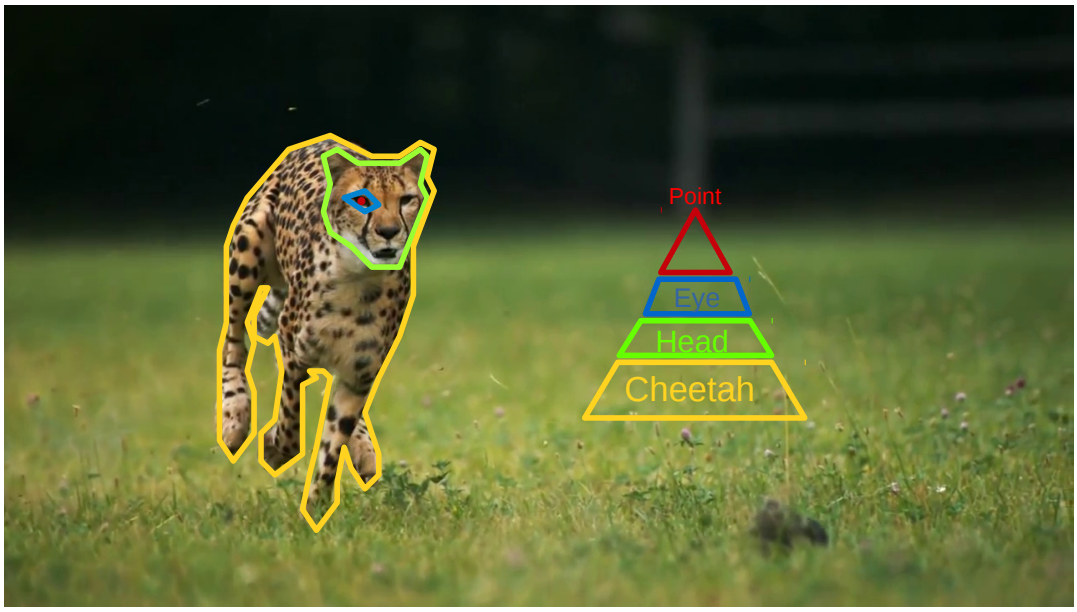


Figure 3.2: An example of a hierarchical representation of an image object, from semantic concepts to the pixel level, in relation with motion information.

attached to a pixel $\mathbf{x}$, given a pair of images $\mathscr{I} = (I_t, I_{t+1})$. Let us assume a hierarchy of memberships of M motion supports to which a pixel belongs to. That is, $\mathbf{x} \in S_i$ with $i = 1..M$ and $S_1 \subset S_2 \subset \cdots S_{M-1} \subset S_M$. Each support S_i is related to a coherent region that usually captures some level of semantics and can be seen as an entity contained by a more "complete" body in the image. For instance, a pixel that belongs to an eye, which is part of a human face, which is at the top of a human body. An example of the hierarchical composition of an object that appears in an image is illustrated in Figure 3.2. If we consider that each one of these supports has an assigned 2D motion vector $\mathbf{w}_i^{i+1}$ at location $\mathbf{x}$ with respect to its parent in the hierarchy, the total apparent motion of the pixel $\mathbf{x}$ can be expressed by:

$$\mathbf{f}_\mathbf{x} = \epsilon_\mathbf{x} + \sum_{m=1}^{M-1} \mathbf{w}_m^{m+1} + \mathbf{w}_M, \tag{3.1}$$

where $\epsilon_\mathbf{x}$ is the residual motion of the point $\mathbf{x}$ with respect to the motion of the entities the point belongs to, and $\mathbf{w}_M$ is the motion of the entity at the base of the hierarchy. In this scheme, the motion computation of each layer is supported by the calculation of the predecessor layer, leading to an instance of the "divide and conquer" idea. It should be noted that this approach of nested layers is different from layered motion estimation techniques such as [SSB12] where the image is spatially partitioned into independent components. Furthermore, it is reasonable to assume that for practical

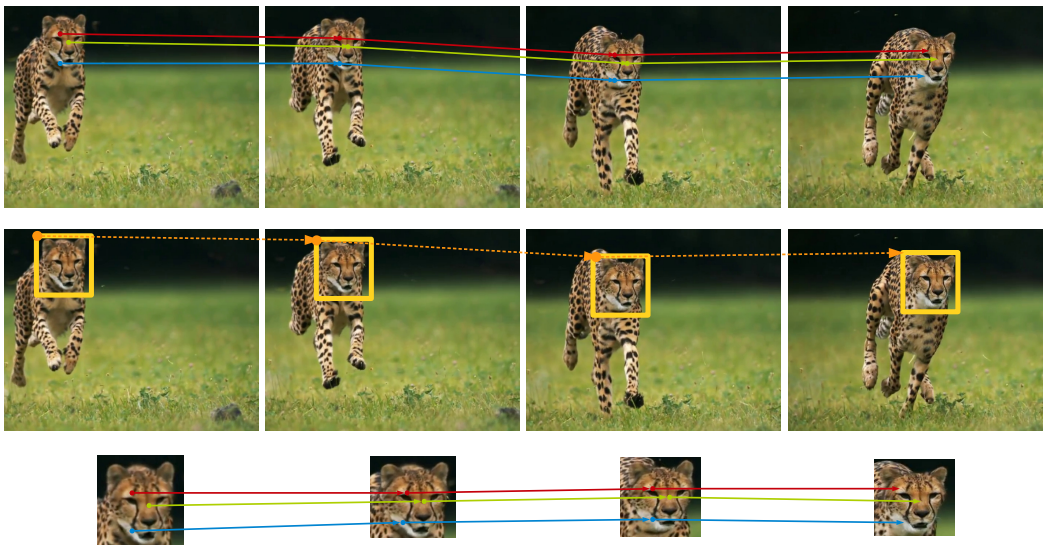


Figure 3.3: **A toy example of point tracking**. *Top*: Four frames of a video sequence showing a running cheetah. For this sequence, 3 points are marked for tracking and the manually constructed ground-truths for every point are shown in red, green and blue. *Middle*: A tracker provides a "global" motion estimation, marked in orange, of the object that contains the points of interest, *i.e.*, the head of the cheetah. *Bottom*: By taking the global motion as reference, the problem of point matching is simplified as the motion amplitude and size of the search space are greatly reduced.

reasons it may not be necessary to go very deep in the hierarchy to improve the point based motion estimation results. For most of the cases, a simple scheme with two layers will be sufficient to get accurate point tracking. Thus, simplifying the equation 3.1 to:

$$\mathbf{f}_{\mathbf{x}} = \epsilon_{\mathbf{x}} + \mathbf{w}_O, \tag{3.2}$$

where $\mathbf{w}_O$ is the motion vector of the region or object containing the point $\mathbf{x}$. This simple two-layered approach is illustrated in Figure 3.3, where the global motion is considered to be the one of the head of the cheetah. This motion can be obtained, for instance, with an object tracker initialized in the first frame of the sequence. As seen in the lower part of the figure, as the motion of points that belong to the head of the animal is bounded to the box of the tracker, its estimation intuitively becomes simpler.

In order to motivate this work, we perform an initial experiment that consists in utilizing a "perfect tracker" to guide a simple optical flow algorithm for the task of point tracking. For this initial experiment we make use of the "Bicycle" sequence, which is borrowed from the VOT challenge for object tracking [KPL+14]. It contains

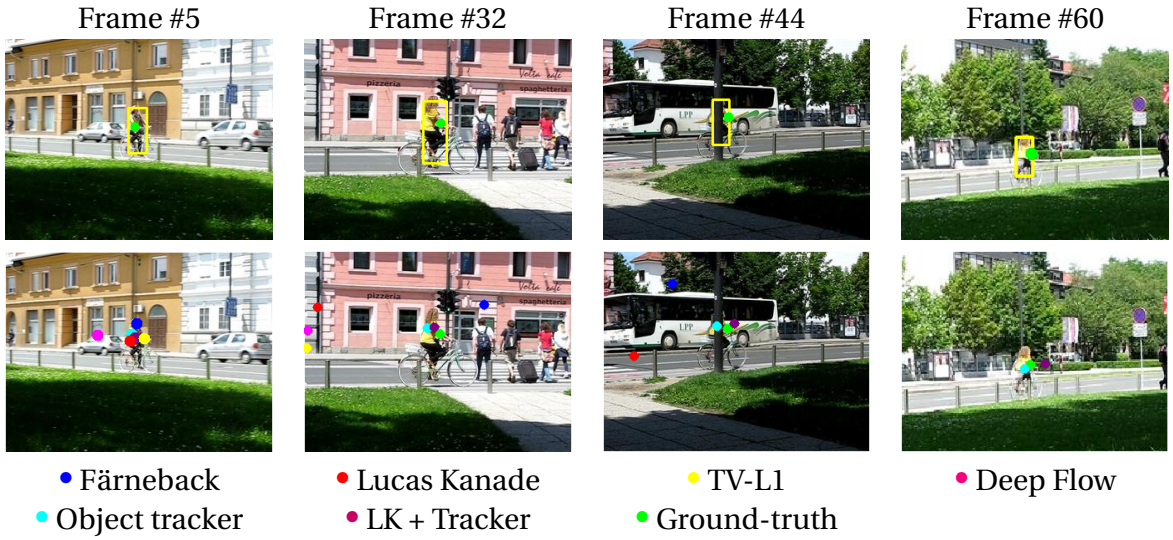


Figure 3.4: Point tracking on the Bicycle sequence. *Top:* The object tracker bounding box is shown in yellow, while the ground truth position of a point in the woman's elbow is displayed in green. *Bottom*: Tracks for the woman's elbow: Färneback's algorithm is shown in blue, Lucas-Kanade in red, TV-L1 in yellow, Deep flow in magenta, the motion of the point given by the object tracker displacement in cyan, and the combination of Lucas-Kanade with the tracker motion in purple.

annotated ground-truth for the object bounding box at each frame, *i.e.*, a woman riding a bicycle. This sequence is particularly challenging for point tracking since it contains camera-motion and occlusions, together with low resolution images that do not preserve sufficient texture detail throughout the sequence. We decided to subsample in time the sequence by a factor of four, to account for long displacements as well. Four frames of the resulting sequence and its tracking ground-truth are shown in Figure 3.4.

As in the example provided in Figure 3.3, the object tracking for this set-up effectively becomes the base of the hierarchical scheme. We compare the quality of the trajectories obtained by temporal Euler integration of optical flow methods [CFC+15] computed with and without tracker-guidance. To do so, we annotated by hand the position of 3 points through all the sequence, the elbow, head and a leg of the woman. As the points of interest are selected by hand, and do not necessarily correspond to well textured points, the usual feature tracking methods are not suitable. The evaluated optical flow algorithms are Färneback's polynomial expansion [Far03], which has demonstrated its value for trajectories estimation in [WKSL11], the pyramidal Lucas-Kanade optical flow estimator [Bou01], and the L1-norm total variation approach by [ZPB07]. We also include in the evaluation a more robust and specialized optical flow

method that is intended to solve for long displacements, DeepFlow [WRHS13]. Finally, the trajectory of the point is also computed by using the position of the bounding box to register every pair of images, and then computing the respective bounded optical flow estimations[1] over the registered images to extract the differential motion $\epsilon_{\mathbf{x}}$. In this way, the total motion of a point inside the bounding box is obtained by applying equation 3.2. The results of the experiments are summarized in Table 3.2. It is interesting to see the poor performance of most of the optical flow methods. In fact, in average, the worst performance is the one obtained with the DeepFlow [WRHS13] which is usually more robust for long displacements. The fact that the motion of the camera is by far the most important motion, together with the fact that most of the discernible corners in the picture are actually part of the background, DeepFlow cannot manage to capture the motion of the woman or the bicycle. Similarly, the rest of the optical flow methods loose the track of the point of interest very early in the sequence, with the exception of the Färneback's algorithm, which gets lost just before the 32th frame. On the other hand, just by using the global translational motion of the tracker the average error is reduced drastically (by an order of magnitude with respect to the baseline optical flow methods). The overall smallest error is achieved by combining the motion of the tracker with the Färneback's optical flow estimation (Tra.+FAR in the table). Interestingly, the poor performance of this method at the full-frame level is notably improved by using the motion of the tracker. In general the results of the point tracking with any of the optical flow methods is improved by including the object tracker as a carrier. This approach does not handle directly the occlusions of the object of interest; it rather relies on the tracker to workaround the estimation of the flow of the occluded point. This issue can be better addressed by taking as reference a frame of the sequence more in the past instead of the immediately previous one. By doing this, once the object is disoccluded the correct matches can be obtained. We will review this issue later on in this chapter.

3.4 Object flow

The experiment shown in the previous section used the motion of an object tracker to guide the estimation of the new position of a point inside the tracker's bounding box. However, although using the global motion of the object tracker in such a simple way indeed improves overall accuracy for point tracking, this method does not make use of the extra information that the object tracker provides. In fact, the tracker is only used to reduce the range of motion to compute for a given point, since this is compensated by the displacement of its bounding box. Going beyond, more information can be extracted and a deeper analysis can improve the accuracy of the estimated point tracks.

[1] In Fig. 3.4, the only optical flow method that is shown in combination with the object tracker is Lucas-Kanade. However, in Table 3.2 all the possible combinations are reported.

Table 3.2: Point tracking results on the Bicycle sequence for various methods for manually selected points in the elbow, head and legs of a woman riding a bicycle. Best results are shown in bold face and second best underlined. The error is computed as the average L2-norm of the difference between the ground truth position and the position obtained by integrating a given optical flow field through all the frames of the sequence.

Avg.Error	Tracker	LK[Bou01]	Tra.+LK	FAR[Far03]	Tra.+FAR
Elbow	12.42	128.04	7.19	85.57	**5.02**
Head	<u>6.74</u>	112.74	7.90	50.81	**6.11**
Legs	9.75	94.91	<u>8.15</u>	61.95	**5.21**
Avg.	9.64	111.90	<u>7.75</u>	66.11	**5.44**

Avg.Error	TV [ZPB07]	Tra.+TV	DEEP[WRHS13]	Tra.+DEEP
Elbow	142.84	<u>7.01</u>	154.72	19.87
Head	129.47	11.20	150.43	26.74
Legs	19.28	10.88	144.56	14.33
Avg.	97.20	9.69	149.91	20.31

In the introductory experiment of this chapter, a few sparse set of points needed to be tracked along a sequence. However, optical flow along the whole object support might be of interest as well for several applications. With those cases in mind, we propose an algorithm to compute object-aware motion maps which consists in two steps: object tracking and support extraction, followed by dense motion estimation, as illustrated in Figure 3.5. Furthermore, as represented in the same figure with dashed lines, computing a per-object dense motion map can be useful as feedback to improve the tracker itself in following frames. For the sake of conciseness, however, this analysis is left out of this chapter[2] and we focus on the computation of the dense motion map within the object boundaries. The reader can refer to the Annex A for more details on the details of the interactive video segmentation method that lies at the core of the algorithm proposed in this chapter.

In Sections 3.4.1 and 3.4.2, we present two implementations of the idea previously described. In both of them we rely on the segmentation approach which is explained in detail in Annex A. The algorithm leverages a graph-based object segmentation that is usually used for 2D images, through a robust scheme for propagating segmentation labels along the video sequence. The idea exploited there to perform the background-foreground separation is that, often the appearance and motion of an object of interest in a video sequence is different from those of the background. Besides that, it is usual to find points that were initially and confidently labeled as background inside

[2] Using object flow results to perform or improve object tracking is considered in Annex A.

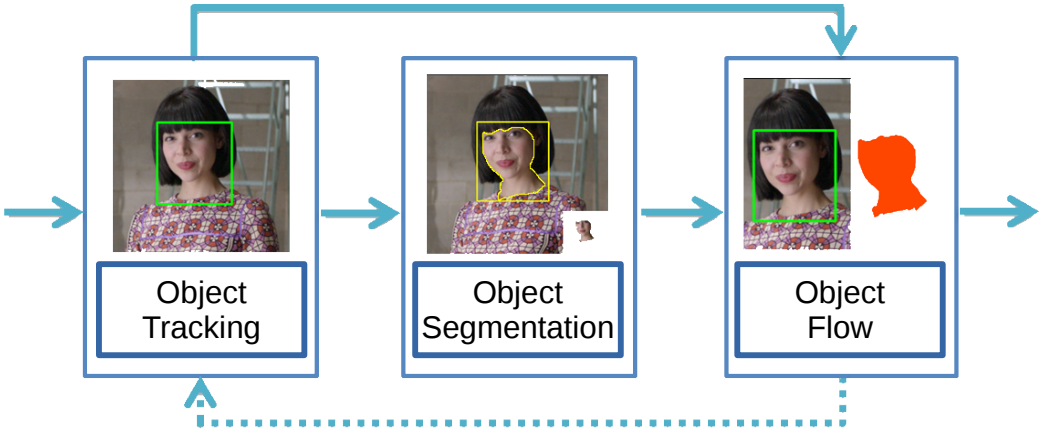


Figure 3.5: Representation of the pipeline for object-aware motion estimation.

the current bounding box of the object of interest. These ideas, together with a scheme that reasons on the strength of points tracks for label propagation, within a *Maximum a Posteriori* (MAP) estimation framework, produce good results for video object segmentation in challenging scenes as it can be observed in Figure 3.6. These results are obtained by running the algorithm with a hand-annotated initialization on the first frame.

3.4.1 A simple Object Flow implementation

We now describe a first interpolation-based implementation of the Object Flow. In general, obtaining the support of the object that contains the set of points of interest is of high value for the task of tracking because the motion of texture-less points can be more easily inferred by analyzing the motion of the points that are successfully tracked. In this way, the computation is improved in two ways. The first one is that, as already mentioned, by first solving the global motion of the object, the trajectories of the points can be computed more accurately. Furthermore, by relying on high confidence trajectories between a given pair of frames, motion can be propagated to less confident areas constrained to the object support. This is valid as points that belong to a single object usually undergo a smooth motion. In fact, it can be expected that for a significant number of cases, the motion inside an object can be very well approximated by a low dimensional parametric model. This is not necessarily the case for complex objects in a scene. For instance, an articulated motion similar to the one that a person or an animal can perform is not smooth inside the support of said object. In cases like the latter, it is more appropriate to incorporate more levels to the decomposition of the object following the hierarchical structure of the image

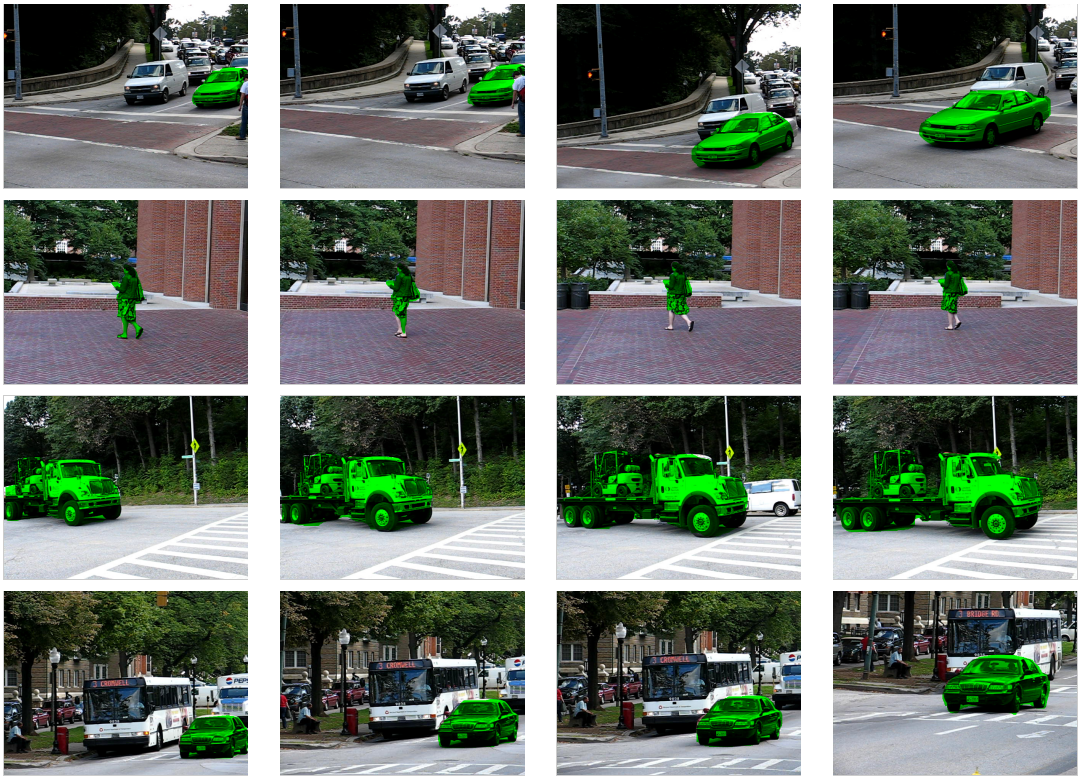


Figure 3.6: Results for three different sequences of the Hopkins dataset [TV07] obtained by our video segmentation method [PRCP15] (See Annex A). Resulting segmentation masks are shown with a green overlay.

(Fig. 3.2).

For our approach, we maintain the idea of strong motion smoothness within the object boundaries, but we prefer not to rely on parametric models in order to achieve better generalization for a broader range of applications. Our first algorithm to compute bounded motion maps, named Simple Object Flow, can be summarized in simple steps as shown in Algorithm 1.

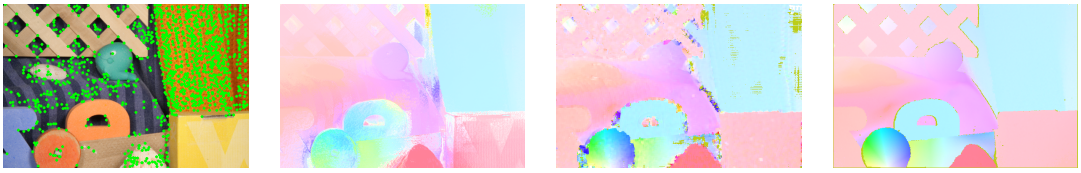
The first step of our approach consists in extracting the support of the object of interest for the given pair of frames. An image with the compensated motion given by the support window is computed to facilitate subsequent extraction of Lucas-Kanade matches. By filtering out the points that fall outside the object support in the second frame, the algorithm can safely assume a smoothness constraint during the last step. The optical flow inside the object boundaries is finally given by the sum of a weighted interpolation at every pixel and the global motion of the object, and it is given by the following equation:

Algorithm 1 Simple object flow.

1: **procedure** OBJECTFLOW
Inputs: Images $\mathscr{I} = (I_t, I_{t+1})$
Outputs: Object supports $\mathscr{S}_O(t)$, $\mathscr{S}_O(t+1)$ and Flow Vectors $f_{\mathbf{x}}\ \forall \mathbf{x} \in \mathscr{S}_O(t)$
2: $\mathscr{S}_O(t) \leftarrow$ `getSupportOfTheObject in` I_t ▷ Using [PRCP15]
3: $\mathscr{S}_O(t+1) \leftarrow$ `getSupportOfTheObject in` I_{t+1} ▷ Using [PRCP15]
4: $\{M_O, I_c\} \leftarrow$ `getCompensatedImage` I_{t+1} ▷ Object's motion compensated target image and global motion of the object
5: $\mathscr{C}(t) \leftarrow$ `getFeaturePoints in` $\mathscr{S}_O(t)$ ▷ Feature points within support
6: $\mathscr{V} = ()$ ▷ Initialize vector of matches
7: **for** `all` $\mathbf{x}$ `in` $\mathscr{C}(t)$ **do**
8: $\mathbf{y} \leftarrow$ `computeLucas-KanadeMatch of` $\mathbf{x}$ `from` I_t `and` I_c
9: **if** $\mathbf{y} \notin \mathscr{S}_O(t+1)$ **then** reject $\mathbf{y}$
10: **else** $\mathscr{V} \leftarrow (\mathscr{V},\ (\mathbf{x}, \mathbf{y}))$
11: **end if**
12: **end for**
13: **for** `all` $\mathbf{x}$ `in` $\mathscr{S}_O(t)$ **do**
14: $f_{\mathbf{x}} \leftarrow$ `interpolateFlow from` $\mathscr{V}$
15: **end for**
16: **end procedure**

$$\mathbf{f}_{\mathbf{x}} = \mathbf{w}_O + \frac{1}{W(\mathbf{x})} \sum_{\mathbf{y} \in C(t)} \mathbf{f}_{\mathbf{y}} K^s\left(||\mathbf{x}-\mathbf{y}||\right) K^c\left(||I_t(\mathbf{x}) - I_t(\mathbf{y})||\right) K^f\left(||I_t(\mathbf{x}) - I_{t+1}(\mathbf{x}+\mathbf{f}_{\mathbf{y}})||\right), \forall \mathbf{x} \in S_O(t), \quad (3.3)$$

where $C(t)$ is the set of detected feature points that belong to the object support $S_O(t)$ in the first frame. The flow vectors $\mathbf{f}_{\mathbf{y}}$ are obtained from the KLT tracker [TK91], while K^s and K^c are respectively range and spatial Gaussian kernels favoring pixels that are closer and more similar in color to the reference pixel $\mathbf{x}$. Furthermore, K^f is a Gaussian kernel giving more weight to motion vectors that preserve color information between frames. More exactly, $K^f(x) = \exp(-\min(x^2, \lambda)/(2 \cdot \sigma_f^2))$, making use of the well known truncated-quadratic penalty which has been used successfully for stereo matching [FH06]. Similarly shaped robust penalties are used for optical flow, but they are usually motivated by derivability constraints for tractable optimization, which is



(a) (b) (c) (d)

Figure 3.7: Weighted interpolation to densify a sparse set of point matches obtained with Lucas-Kanade method [LK81] in a sequence of the Middlebury. (a) Detected feature points of the first frame. (b) Our weighted interpolation output using equation 3.3. (c) Optical flow map computed with Lucas-Kanade only. (d) Ground truth flow. For this example, the flow at each point is computed by using the 10 nearest matched neighbors.

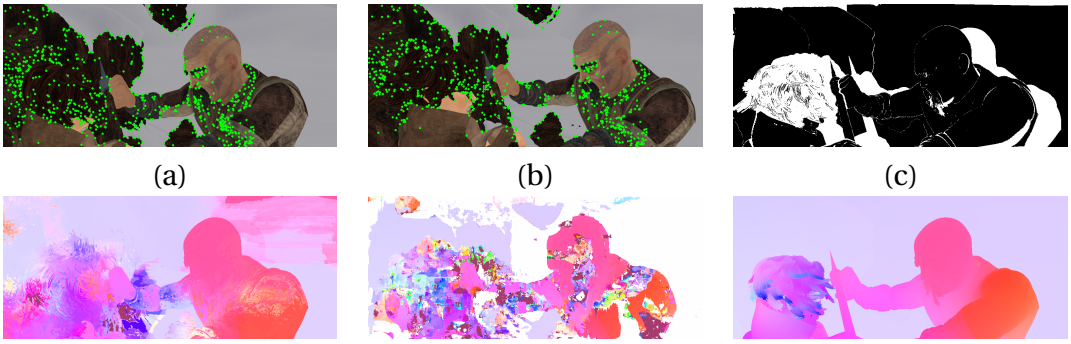
not a necessity in our algorithm. Finally, the weighting factor $W(\mathbf{x})$ normalizes the output flow values:

$$W(\mathbf{x}) = \sum_{\mathbf{y}\in C(t)} K^s\left(||\mathbf{x}-\mathbf{y}||\right) \cdot K^c\left(||I_t(\mathbf{x}) - I_{t+1}(\mathbf{y})||\right) \cdot K^f\left(||I_t(\mathbf{x}) - I_{t+1}(\mathbf{x}+\mathbf{f}_\mathbf{y})||\right). \quad (3.4)$$

This approach for extrapolating values for the unknown matches (*i.e.* pixels that do not belong to $C(t)$) is inspired by the use of joint bilateral filter [KCLU07] for computing pixel-wise energy terms for optical flow estimation in [TBKP12], and for performing upsampling of depth maps in [GMO+10].

For most of the tested sequences, the final number of matches that belong to $C(t)$ is around 20 to 30 at most. This means that equation 3.3 can be computed very rapidly. However, for high-resolution applications, with a high number of matches per-object, the set of feature points to iterate on should be reduced by using the K-nearest neighbors in the vector space formed by aggregation of color and spatial position. In this sense, one could imagine using this method to compute full optical flow maps from sparse point matches, however, due to the hypothesis of smooth motion, it is not expected to provide a good motion estimation for complex scenes. An example of this is presented in Figure 3.7. Although the estimated motion map (b) is relevant in comparison to the ground truth optical flow (d), the more so knowing that it was generated by a highly efficient and simple algorithm. Perhaps even more surprising, is the example of the Figure 3.8, where our algorithm manages to produce a correct optical flow map for a sequence depicting long displacements from the MPI Sintel dataset [BWSB12b]. These results, however, are still far from the ones that can be obtained by state-of-the-art optical flow methods.

Going further, let us suppose that for this sequence, only the motion map associated to the person on the right is needed, and that a bounding box is provided for



(a) (b) (c)

(d) (e) (f)

Figure 3.8: Weighted interpolation to densify a sparse set of point matches obtained with Lucas-Kanade method [LK81] in a sequence of the Sintel dataset. (a) Detected feature points of the first frame (Frame #37). (b) Corresponding feature points in the second frame (Frame #38). (c) Occlusion ground-truth. (d) Output of our weighted interpolation method (eq. 3.3) (e) Optical flow map computed with Lucas-Kanade only. (f) Ground truth flow. For this example, the flow at each point is computed by using the 10 nearest matched neighbors in equation 3.3.

the first frame of the sequence. As seen in Figure 3.9, when equipped with the object tracker, the estimation of the flow for the region of interest is slightly better (Fig. 3.9.e) than the estimation that does not take it into account (Fig. 3.9.d). However, when we go further and compute the motion extrapolation by only taking into account the point matches inside the object's computed segmentation masks, we can appreciate a visible improvement in the estimation of the flow field for this region (Fig. 3.9.f).

3.4.2 Deep object flow

Taking an off-the-shelf state-of-the-art optical flow method [WRHS13], we can easily combine it with our framework to convert it to an Object Flow method. In this way, we will be able to show in the experimental results the benefits of explicitly including global object-level motion information to feed an optical flow solver. As explained in Section 3.2, the Deep Flow method starts from an original matching algorithm termed Deep Matching which is well tailored for the optical flow problem. An energy functional within the variational framework proposed by [BM11] with a data, a smoothness and a deep matching preservation terms is proposed. More details are found in [WRHS13].

In order to make this algorithm object aware, only a few steps are required. Similarly to the simple implementation of the last section, the first step is to compute the object support in the input images. Secondly, the input images are aligned according to the global object tracker motion. Then, the deep matches are computed from the

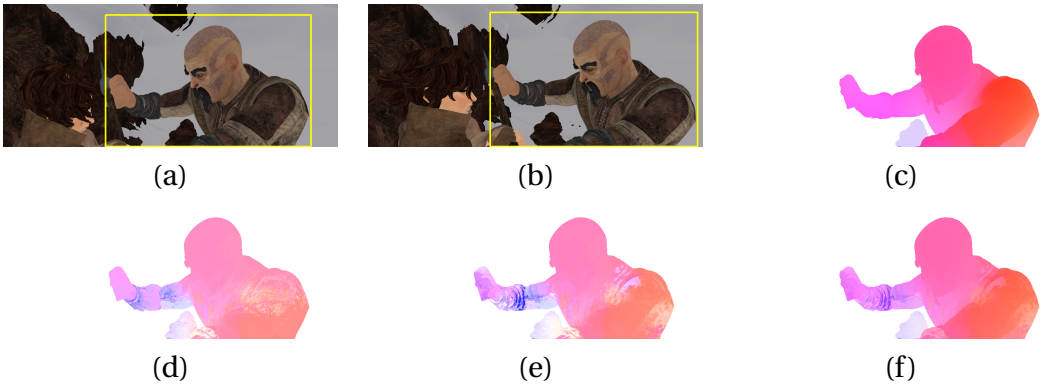


Figure 3.9: Flow estimation by using object-guided motion weighted extrapolation. (a-b) Frames with marked tracker. (c) Optical flow ground truth within the object support. (d) Results of the motion extrapolation without tracker (the mask of the object of interest is applied for clarity). (e) Masked results with tracker compensation. (f) Results using full tracker motion compensation and object boundaries.

aligned images (making the method robust for complex object motions). Finally, the variational solver is run for all the points within the object support in the first image. We summarize results of the two presented algorithms in the following section.

3.5 Experimental results

We evaluate our algorithms on the Hopkins 155 dataset [TV07], which was initially proposed to evaluate feature-based motion segmentation algorithms. This dataset, however, provides valuable information for assessing point tracking, since it provides ground-truth trajectories of sparse sets of points with regions' membership labels. More specifically, we take the sequences that contain scenes with well-defined objects and challenging obstacles for the task of point tracking as shown in Figure 3.10, where the objects of interest and the set of points that are to be tracked are explicitly marked. As some sequences contain more than one object of interest, we display the results specifying the label of the object which was taken into account. In total, the results were evaluated on 19 different objects from 16 sequences that contain between 20 and 61 frames.

Trajectories were computed for the set of points with available ground truth for 5 different methods: our Simple Object Flow as well as the Deep Object Flow; the KLT tracker [TK91], which is taken as baseline of the Simple Object Flow since we use the KLT matches between frames to generate a per-object dense motion field; Euler's integration of DeepFlow [WRHS13]; and Euler's integration of Färneback's algorithm [Far03].

We consider the position mean absolute error (MAE) as evaluation criteria. We also present figures for the median absolute error (MED) and median absolute deviation (MAD), whose importance for assessing point tracking robustly has been discussed in [CFC+15].

For this experiment, we can see in Table 3.3 that our Simple Object Flow clearly improves the accuracy of the baseline KLT tracker. In fact, Simple Object Flow is competitive with respect to more robust approaches for high accuracy optical flow methods like [WRHS13], and a method which has shown its robustness for several applications that require estimating point trajectories [Far03]. Unsurprisingly, the most effective method for point tracking is the Deep Object Flow, which overcomes by a good margin to its baseline, the Deep Flow algorithm. Moreover, with respect to a key measure, the position mean absolute error, our methods stand the most accurate for 14 of the 19 sequences. These results are very motivating as they allow us to conclude that even by using a very simplistic point motion estimation, equipping it with an object-level tracker displacement estimation and the knowledge of the support of the object provides highly pertinent information for accurate point tracking.

The Hopkins dataset is very challenging from several points of view. However, it lacks of ground-truth information on temporally occluded points for evaluating our method behavior in such situations. With this in mind, we propose to include artificial occlusions in some of the original sequences. In such a way, we utilize the same ground-truth for the point to track during and after the fake occlusion occurs in the video. As discussed in Section 3.3, our method does not handle occlusions[3] by itself at the pixel level, it actually relies on the object tracker to maintain an adequate position for the occluded points. This, however, can be improved by using as reference frame the initial frame of the video (or any other past frame with assessable tracking confidence). This means that the point matches are computed in general between the current object position and a past position of the object, not necessarily restricted to the immediate previous one. Such an approach is not reliable for common optical flow methods, since the displacement of the object with respect to the initial frame tends to grow with time, leading to motion vector computation errors. On the contrary, our method is much less sensitive to this due to the counterbalancing effect of the object tracker.

Having said this, we evaluate how robust is our method taking into account both choices of reference frame. That is, estimating motion between the current frame and either the initial frame (method REF0) or the previous one (method REFP). Table 3.4 shows a comparison of the results obtained with both versions of our method and by using a high accuracy optical flow algorithm to solve the point tracking problem under occlusions. It can be appreciated that the version of our method that takes

[3] Explicit detection method as proposed in Chapter 5 might be helpful in this framework. We left, however, this consideration for future work.

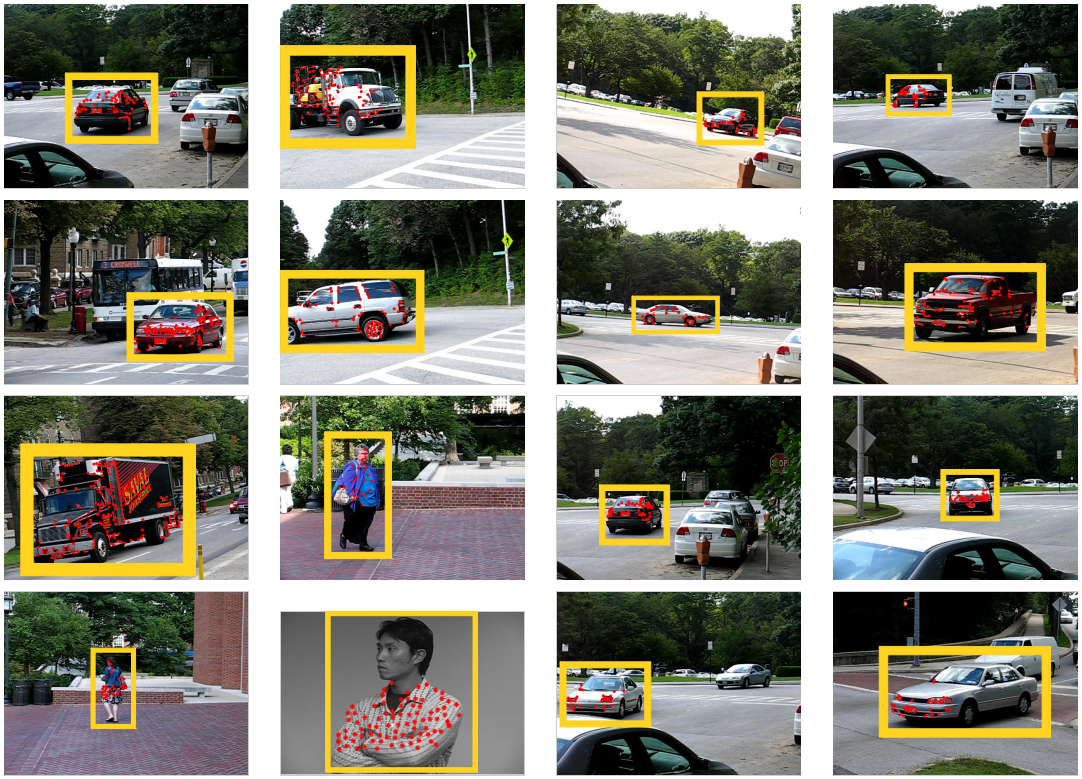


Figure 3.10: Sample frames from the used sequences of the Hopkins 155 dataset [TV07]. From left to bottom: *Cars2*, *Truck1*, *Cars7*, *Cars3*, *Cars10*, *Cars1*, *Cars6*, *Cars8*, *Truck2*, *people2*, *Cars2B*, *Cars4*, *people1*, *head*, *Cars5*, and *Cars9*. The object of interest is enclosed in a yellow bounding box for each sequence, and the points marked to track are shown in red.

Figure 3.11: Samples of the sequences of Hopkins 155 dataset that were artificially occluded. The occlusion is shown in yellow.

the first frame as reference remains the most precise for almost all of the evaluated sequences. In fact, for all the *Cars* sequences, all the other methods present high mean absolute errors while our referenced-to-the-first-frame algorithm still manages to perform a reasonably accurate point tracking. Furthermore, although the state-of-the-art optical flow algorithm used for comparison is claimed to be very accurate for long displacements [WRHS13], when it is referenced to the first frame (REF0), it still

Table 3.3: Comparative experimental results on the traffic and articulated scenes from the Hopkins 155 dataset. For each method we display the position mean absolute error (MAE), the median absolute error (MED), and the median absolute deviation (MAD). The best results according to the MAE are shown in bold face.

	Simple Object Flow			Deep Object Flow			Deep flow			KLT			Färneback		
Sequence	MAE	MED	MAD	MAE	MED	MAD	MAE	MED	MAD	MAE	MED	MAD	MAE	MED	MAD
TRUCK2-1	3.393	1.877	1.374	**2.684**	1.201	1.025	3.510	1.943	1.551	22.824	1.762	1.647	4.887	2.813	2.579
TRUCK1-1	3.796	1.737	1.223	3.478	1.335	1.541	4.037	1.759	1.234	2.624	1.062	0.792	**1.378**	0.836	0.575
CARS10-1	3.327	1.241	1.223	**1.892**	1.025	0.555	2.423	1.416	0.993	10.931	1.207	0.920	10.493	2.765	1.540
CARS9-1	2.264	0.565	0.453	2.959	1.421	0.787	4.037	2.409	1.418	**0.824**	0.523	0.335	3.504	2.358	2.210
CARS8-2	**2.513**	1.413	1.060	2.588	1.361	0.999	2.962	1.449	1.023	13.567	1.171	0.948	3.346	1.694	1.493
CARS7-1	2.429	1.195	0.957	3.002	1.074	0.603	4.569	1.072	0.745	1.127	0.625	0.418	**0.914**	0.677	0.482
CARS6-1	**2.426**	0.964	0.649	3.482	1.777	0.652	0.975	3.566	2.341	3.174	1.062	0.743	2.43	1.243	0.379
CARS5-1	3.175	1.781	1.425	2.012	0.445	0.235	1.299	0.990	0.667	3.525	1.695	0.520	**0.808**	0.531	0.379
CARS5-2	2.014	0.572	0.381	**1.466**	1.001	0.822	2.016	1.329	0.901	2.168	0.639	0.445	2.695	1.350	1.235
CARS4-1	**1.320**	1.179	0.954	1.513	1.014	0.436	1.620	1.185	0.754	3.109	1.094	0.778	1.515	0.881	0.752
CARS3-1	3.097	1.161	1.052	2.125	1.256	1.015	2.949	1.893	1.463	24.09	1.157	0.863	**0.862**	0.692	0.367
CARS3-2	3.484	1.181	0.866	**1.225**	0.878	0.465	1.660	1.006	0.671	59.97	21.38	20.94	1.905	0.937	0.702
CARS2B-1	1.792	1.151	0.776	**1.512**	1.214	0.678	1.699	1.364	0.866	10.49	1.220	1.034	1.856	1.741	1.474
CARS2B-2	3.945	0.764	0.588	**0.312**	1.154	0.625	0.456	1.378	0.777	2.250	0.539	0.380	0.563	0.434	0.301
CARS2-1	2.005	1.146	0.740	**1.878**	1.198	0.788	2.214	1.386	0.858	11.621	1.149	0.918	2.050	1.70	0.460
CARS1-1	4.889	2.361	1.775	**2.524**	1.521	1.040	5.524	1.800	1.148	135.3	83.33	82.083	6.329	0.713	0.538
PEOPLE1-1	**2.457**	1.619	1.066	2.513	1.815	1.028	7.500	1.769	1.393	3.576	1.203	0.868	3.113	1.866	1.515
PEOPLE2-1	7.572	6.196	4.855	**2.228**	1.246	1.133	3.258	1.589	1.287	19.87	9.912	9.562	2.459	0.855	0.636
HEAD-1	**0.324**	0.338	0.355	0.724	0.328	0.218	1.017	0.500	0.352	0.848	0.392	0.262	0.866	0.354	0.261

produces large tracking errors. In fact, for this optical flow method it is difficult to determine precisely which reference is better in general, since for the tested sequences, the two versions produce comparable results. We also present visual results on real datasets with strong occlusions in Figure 3.12. It is possible to see that these hard occlusions cannot possibly be recovered by integrating any optical flow method. The point can only be recovered through correct (global) object tracking. For this experiment we rely on the Simple Object Flow referenced to the first frame (REF0). This shows the strength of our general approach even under difficult cases as temporal occlusions of the object of interest.

It is worth mentioning that our method shares some of the limitations the object tracking methods. For example, it can get "lost" under certain conditions like prolonged and strong illumination changes or abrupt disappearing and reappearing. For some of the applications of the Object Flow, this is in fact not very critical. Usually the

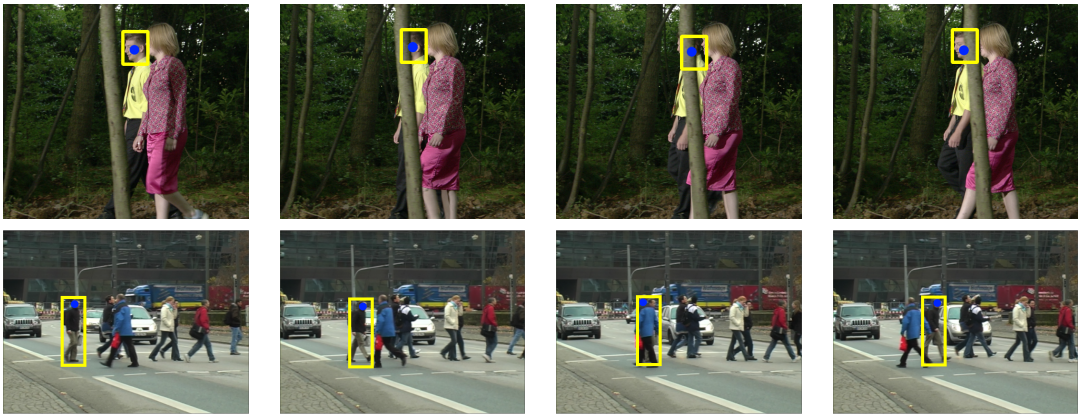


Figure 3.12: Visual point tracking results using our Simple Object Flow referenced to the first frame (REF0). The blue points are tracked along video sequences with total occlusions.

Table 3.4: Comparative experimental results on artificially occluded sequences of the Hopkins dataset. For each method we display the position mean absolute error (MAE), the median absolute error (MED), and the median absolute deviation (MAD). The best results according to the MAE are shown in bold face.

	Simple Obj. Flow REF0			Simple Obj. Flow REFP			Deep flow REF0			Deep flow REFP		
Sequence	MAE	MED	MAD	MAE	MED	MAD	MAE	MED	MAD	MAE	MED	MAD
CARS1-1	**15.215**	5.212	4.441	21.472	3.738	3.165	28.983	1.622	1.502	35.824	2.910	2.758
CARS3-1	**8.485**	2.842	1.255	21.172	5.830	5.579	16.641	1.166	1.024	11.231	1.005	0.922
CARS6-1	**5.689**	1.187	1.022	5.708	1.163	0.890	17.066	0.726	0.571	15.244	3.665	1.300
PEOPLE1-1	5.707	1.175	0.900	**5.528**	1.553	1.107	7.447	1.511	0.955	9.886	1.435	0.916
PEOPLE2-1	**4.848**	2.964	1.541	8.708	5.464	4.125	4.891	2.153	0.441	5.646	1.019	0.958

video editing pipelines require some level of human interaction. A quick stroke on the image can re-localize the object tracker and correct possible errors.

Regarding execution times, for the Simple Object Flow our interpolation algorithm from sparse matches is very fast. It can be computed in real time as fast as 4 ms for a pair of frames in a computer with a CPU Intel Core i5-3210 @ 2.50GHz. The segmentation part is slower, it takes an average of 0.8s per frame to process in the same machine for a picture of 640×480 pixels. These times compare favorably with modern optical flow methods computation times which take from a few seconds to a pair of minutes on a CPU for a pair of frames[4].

[4]Very recent optical flow methods might actually offer better performance given the availability of modern GPUs [DFI+15].

3.6 Conclusions

We have presented a simple algorithm to compute dense motion maps for given support regions of objects of interest in video scenes. We showed that the algorithm can be used for reliable point tracking in challenging video sequences, by testing our method and presenting quantitative comparisons with state-of-the-art and off-the-shelf optical flow methods. The experiments imply that our approach is reliable for occlusion handling and long-range motions, as long as the underlying tracking and segmentation of the object of interest is reliable. This work can be of high value during the exploration of new semantically-aware motion-based inference algorithms for video sequences. Even more, we believe that our findings on how the object contours can improve motion estimation for objects can lead to new algorithms for motion estimation that rely on the intrinsic hierarchy of the objects in the scene, rather than its motion layers. Furthermore, our experiments have shown the importance of considering motion composition for video analysis tasks.

CHAPTER 4

RICH OBJECT APPEARANCE MODELS FOR ROTOSCOPING AND TRIMAP TRACKING: ROAM AND ROAM+

Rotoscoping, the detailed delineation of scene elements through a video shot, is a painstaking task of tremendous importance at the heart of high-end film post-production. While pixel-wise segmentation techniques can help for this task, professional rotoscoping tools rely on parametric curves that offer the artists a much better interactive control on the definition, editing and manipulation of the segments of interest. Sticking to this prevalent rotoscoping paradigm, we propose a novel framework to capture and track the visual aspect of an arbitrary object in a scene, given a first closed outline of this object.

The related, yet different task of alpha-matting consists in extracting soft foreground masks accounting for transparency, translucency and motion blur. Rotoscoping curves, *a.k.a.* roto-curves, and alpha-mattes are used together in the post-production pipeline for several tasks, compositing in particular –the arduous task of combining elements from two different video shots in a seamless manner. However, an important intermediate step is missing between curve-based rotoscoping and matting: The extraction of a so-called *trimap*, which is the input of the alpha-matting algorithm. These trimaps define "background" and "foreground" regions to train a model that permits finding a mixture (alpha) coefficient for each pixel in a third and intermediate zone. An overview of all the involved techniques for a video sequence can be observed in Fig. 4.1. The problem of automatically determining the trimaps

that feed the alpha-matting algorithms along the video sequence, *i.e.*, trimap tracking, has not been sufficiently studied in previous literature. In fact, most current works have focused solely on the extraction of the alpha-mattes assuming the input trimaps are already available.

In this chapter we do not only present a model for rotoscoping, but also an extension of it which delivers such trimaps and further refines the quality of the initial roto-curves. All in all, we propose a framework for computing and propagating roto-curves and trimaps. We leverage local classifiers attached to the roto-curves to define a confidence measure that is well suited for the trimap definition. This trimap remains user-editable for subsequent manipulation by the artist. Qualitative and quantitative results on several relevant datasets show the merit of our methods.

4.1 Introduction

Modern high-end visual effects (vfx) and post-production rely on complex workflows whereby each shot undergoes a succession of artistic operations. Among those, rotoscoping is probably the most ubiquitous and demanding one [Bra11, LVS+16]. Rotoscoping amounts to outlining accurately one or several scene elements in each frame of a shot. This is a key operation for compositing [Wri06] (insertion of a different background, whether natural or synthetic), where it serves as an input to subsequent operations such as matting and motion blur removal[1]. Rotoscoping is also a prerequisite for other important operations, such as object colour grading, rig removal and new view synthesis, with large amounts of elements to be handled in the latter case.

Creating such binary masks is a painstaking task accomplished by trained artists. It can take up to several days of work for a complex shot of only a few seconds, using dedicated tools within video editing softwares like Silhouettefx, Adobe After Effect, Autodesk Flame or The Foundry Nuke. As discussed in [LVS+16], professional roto artists use mostly tools based on *roto-curves*, *i.e.*, parametric closed curves that can be easily defined, moved and edited throughout shots. By contrast, these artists hardly use brush-based tools[2], even if empowered by local appearance modelling, graph-based regularization and optic flow-based tracking as in After Effect's ROTOBRUSH.

[1] The use of blue or green screens on set can ease compositing but remains a contrived set-up. Even if accessible, such screens lead to chroma-keying and de-spilling operations that are not trivial and are not suited to all foreground elements [AAPS16], thus rotoscoping remains crucial.

[2] By "brush-based tools" we refer to the interactive video segmentation methods that allow inputs from the user in the form of strokes, bounding boxes, or similar pixel-oriented actions. Often, those cues are used by the underlying algorithm to initialize per-pixel costs inside a global energy-minimization framework for pixel labeling. Consequently, the user cues often affect the output segmentation in non-predictable ways.

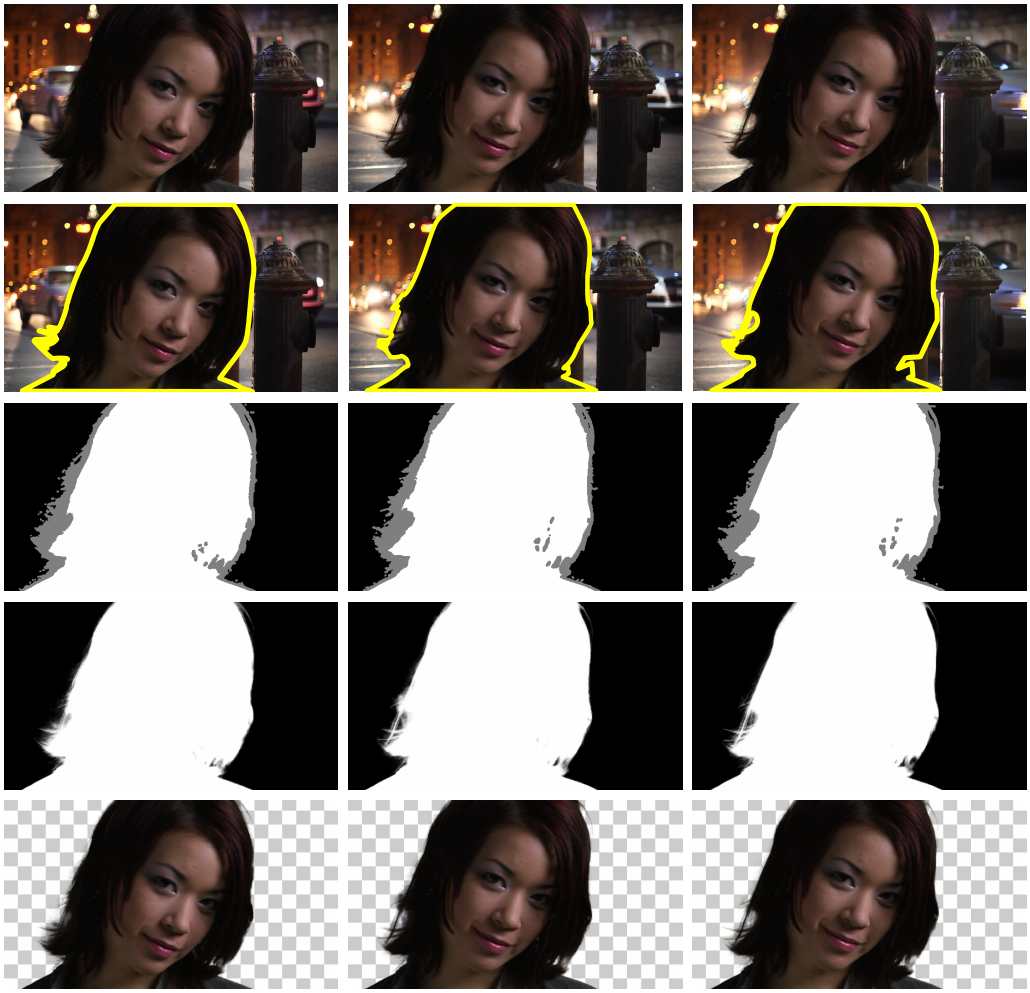


Figure 4.1: **Operations in the video post-production pipeline**. First row: an image sequence from a video matting benchmark [EGV+15]. Second row: Roto-curves drawn by an artist. Third row: Annotated trimaps along the sequence. Fourth row: alpha-mattes computed by [XPCH17]. Fifth row: The foreground is composited with a pattern background.

Due to its massive prevalence in professional workflows, we address here roto-scoping in its closed contour form, which we aim to facilitate. Roto-curves being interactively placed in selected keyframes, automation can be sought either at the key-frame level (reducing the number of user's inputs) or at the tracking level (reducing the number of required key-frames). In their recent work, Li *et al.* [LVS+16] offer with ROTO++, a tool that helps on both fronts, thanks to an elegant online shape modelling.

In this chapter, we explore a complementary route that focuses on the automatic

roto-curve tracking from a given keyframe. In essence, we propose to equip the roto-curve with a rich, adaptive modelling of the appearance of the enclosed object. This model, coined ROAM for Rich Online Appearance Model, combines in a flexible way various appearance modelling ingredients: (i) local foreground/background colour modelling, in the spirit of VIDEO SNAPCUT [BWSS09] but attached here to the roto-curve; (ii) fragment-based modelling to handle large displacements and deformations, and (iii) global appearance modelling, which has proved very powerful in binary segmentation with graph cuts, *e.g.*, in [BJ01].

We would like to emphasize that our model is the first that combines local appearance models along the closed contour with global appearance model of the enclosed object using the discrete Green theorem, and pictorial structure to capture locally rigid deformations, in a principled structured prediction framework. As demonstrated on recent benchmarks, ROAM outperforms state-of-art approaches when a single initial roto-curve is provided. It is in particular less prone to spurious changes of topology that lead to eventual losses than After Effects ROTOBRUSH, and more robust than ROTO++ [LVS$^+$16] in the absence of additional user inputs. This robustness makes it appealing to facilitate rotoscoping, either as a standalone tool, or combined with existing curve-based tools such as ROTO++.

In the post-production pipeline, besides the roto-curves, and due to transparency, translucency and motion blur, an artist needs also to manually extract alpha-mattes, a set of coefficient maps indicating the foreground/background mixing ratios for all pixels of each frame of the input video shot. In order to produce these coefficient maps, alpha-matting algorithms require two inputs from the user: an image, and a corresponding **trimap**. A trimap is defined as a per-pixel label map determining which pixels are surely background or foreground, and which pixels fall into an intermediate region whose appearance is better explained as mixture of both background and foreground elements. Automatically obtaining alpha-mattes from still images and a corresponding input trimap has been a topic of interest for a few years in the computer graphics and computer vision literature [GSAW05, RRG08, GO10, HRR$^+$11, CCSS01, LLW08]. Extensions from still image matting to video matting have also been previously tackled [CAC$^+$02, LCT13, WC$^+$08]. However, to the best of our knowledge, the intermediate task of generating trimaps for each one of the frames in a shot is still mostly unexplored and critical terrain. This is all the more true if we consider the task under the parameterized closed curve paradigm, as opposed to pixel-wise labeling frameworks, which lies at the core of main professional tools in compositing pipelines.

Under the realization of the importance of trimaps in professional settings, we propose an extension to ROAM, coined ROAM+, which complements the roto-curves produced by ROAM, and equips them with slack regions that can be interpreted as trimaps. Furthermore, we demonstrate that such trimaps can be used to refine the roto-curve tracking results, improving the overall segmentation performance.

4.2 Related work and motivation

Rotoscoping is a form of interactive "video object"[3] segmentation. As such, the relevant literature is vast. For sake of brevity, we focus mostly our discussion on works that explicitly target rotoscoping or very similar scenarios.

4.2.1 Rotoscoping and curve-based approaches

Li *et al.* [LVS+16] recently released a very detailed study of professional rotoscoping workflows. They first establish that trained artists mostly use parametric curves such as Bezier splines to delineate objects of interest in key-frames, "track" them from one frame to the next, edit them at any stage of the pipeline and, last but not least, pass them in a compact and manipulable format to the next stage of the vfx pipeline, *e.g.*, to the compo-artists. Professional rotoscoping tools such as Silhouettefx, Blender, Nuke or Flame are thus based on parametric curves, which can be either interpolated between key-frames or tracked with a homographic "planar tracker" when suitable. Sticking to this ubiquitous workflow, the authors propose ROTO++ to speed it up. Bezier roto-curves defined by the artist in the selected key-frames allow the real-time learning of a non-linear low-dimensional shape space based on a Gaussian process latent variable model. Shape tracking between key-frames, as well as subsequent edits, are then constrained within this smooth manifold (up to planar transforms), with substantial gains in work time. Our work is fully complementary to ROTO++: while ROAM does not use a strong shape prior in its current form, it captures the dynamic appearance of the video object, something that ROTO++ does not address.

In their seminal rotoscoping work, Agarwala *et al.* [AHSS04] proposed a complete interactive system to track and edit Bézier roto-curves. It relies on the popular active contour framework [BI00, KWT88]: a curve, parametrized by control points, finely discretized and equipped with a second-order smoothness prior is encouraged to evolve smoothly and to snap to strong image edges. Their energy-based approach also uses local optical flow along each side of the shape's border. In contrast to this work, our approach offers a richer local appearance modelling along the roto-shape as well as additional intra-object appearance modelling.

Similarly to [AHSS04], Lu *et al.* [LBSW16] recently introduced an interactive object segmentation system called "coherence parametric contours" (CPC), which combines planar tracking with active contours. Our system includes similar ingredients, with the difference that the planar tracker is subsumed by a fragment-based tracker and that the appearance of the object and of its close surrounding is also captured and modeled. We demonstrate the benefits of these additional features on the evaluation dataset introduced by Lu *et al.* [LBSW16].

[3] Throughout, "video object", or simply "object", is a generic term to designate a scene element of interest and the associated image region in the video.

4.2.2 Masks and region-based approaches

Other notable approaches to interactive video segmentation address directly the problem of extracting binary masks, *i.e.*, labelling pixels of non-keyframes as foreground or background. As discussed in [LVS+16, LBSW16], a region-based approach is less compatible with professional rotoscoping, yet provides powerful tools. Bai *et al.* [BWSS09] introduced VIDEO SNAPCUT, which lies at the heart of After Effect's ROTOBRUSH. Interaction in VIDEO SNAPCUT is based on foreground/background brushes, following the popular scribble paradigm of Boykov and Jolly [BJ01]. The mask available in a given frame is tracked to the next frame through the propagation of local windows that straddle its border. Each window is equipped with a local foreground/background colour model and a local shape template, both updated through time. After propagation along an object-centric optical-flow, these windows provide suitable pixel-wise unary data-driven energy terms that are fed to a classic graph-cut. This approach provides a powerful way to capture on-the-fly local colour models and to combine them adaptively with some shape persistence. However, being based on graph-cut (pixel-wise labelling), ROTOBRUSH can be penalized by its extreme topology flexibility: as will be showed in the experiments, rapid movements of the object, for instance, can cause large spurious deformations of the mask that can eventually lead to complete losses in the absence of user intervention. In ROAM, we take inspiration from the local colour modelling at the object's border, and revisit it in a curve-based segmentation framework that allows tighter shape control and easier subsequent interaction.

More recently, Fan *et al.* introduced JUMPCUT [FZL+15], another mask-based approach where frame-to-frame propagation is replaced by mask transfer from the key-frame(s) to distant frames. This long-range transfer leverages dense patch correspondences computed over the inside and outside of the known mask, respectively. The transfered mask is subsequently refined using a standard level set segmentation (region encoded via a spatial map). A salient edge classifier is trained online to locate likely fragments of object's new silhouette and drive the level set accordingly. They reported impressive results with complex deformable objects going through rapid changes in scene foreground. However, similarly to ROTOBRUSH, this agility might also become a drawback in real rotoscoping scenarios, as is the lack of shape parametrization. Also, the underlying figure/ground assumption (the object is moving distinctly in front of a background) is not met in many cases, *e.g.*, rotoscoping of a static scene element or of an object in a dynamic surrounding.

Finally, very recent mask-based methods with deep learning are worth mentioning. It must be noted first that, the computer vision task of semantic segmentation is not fully compatible with the rotoscoping framework. More exactly, roto-curves might be defined for an arbitrary region that do not necessarily encompass a semantic concept at the object level, as it is required by standard semantic segmentation methods [KVK16, FEF+17]. The video segmentation method proposed by Khoreva *et*

al., [KVK16] is closer to the rotoscoping task as it allows arbitrary input masks. However, such methods do not allow further user intervention along the video sequence. This characteristic is invaluable in the video post-production pipeline.

4.2.3 Previous works on trimaps

Trimap tracking as a standalone problem is not very common in previous state-of-the-art. The reason is that most of the work has focused on extracting the alpha mattes. To the best of our knowledge, the only directly related work is [BWS11], which aims at providing temporally coherent trimaps to be fed to alpha matting algorithms. Most common approaches for extending alpha-matting methods to videos make use of forward and backward optical flow fields for interpolation of trimaps [CAC+02]. However, these approaches share the same disadvantages that video segmentation has for rotoscoping: Per-pixel labeling methods are not convenient for the compositing artist in a task that is heavily supervised and requires strong interactivity. Starting with the ubiquitous roto-curves, an artist expects to be able to correct the trimaps in the same way he/she corrects the curves, *i.e.*, a couple of dragging operations on a parametric object. Moreover, trimap interpolation drastically fails when background moves very differently from foreground, or when the optical flow is generally unreliable (scene with large motions and occlusions).

Pérez *et al.* [PBG01] proposed a probabilistic (particle-based) framework for interactively extracting contours in static images. They show how to extend their approach to consider wider contour regions (for extracting roads from satellite images). Their model is conceptually similar to ours. However, they do not devise ways to handle video sequences and consider only thin elongated objects. Another issue of their framework is that randomized methods are less preferred in rotoscoping and related tasks.

On a tangentially related task, [FSW12] improves object tracking by performing a rough trimap estimation that allows advanced model updating schemes. As the final task is the accurate bounding box tracking, the quality of the trimaps is not truly important in this work, ignoring issues like labeling noise and temporal consistency. Similarly, [AB06] uses trimap estimation as an intermediate step for human silhouette extraction, resulting in what is basically an iterative procedure for static image segmentation.

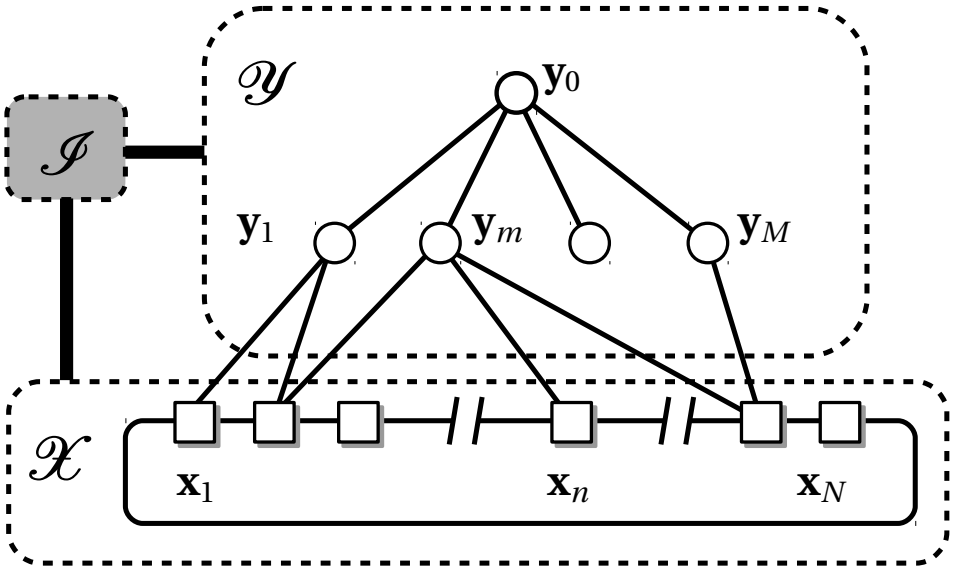


Figure 4.2: **Graphical model of ROAM**. In joint model defined by energy $E(\mathscr{X},\mathscr{Y};\mathscr{I})$ in (4.1), contour node variables (white squares) form a closed 1-st order chain conditioned on image data (grey box) and landmark variables (white circles), the latter variables forming a shallow tree conditioned on all others.

4.3 Introducing ROAM

Our model consists of a graphical model with the following components: (i) a closed curve that defines an object and a collection of local foreground/background[4] appearance models along it; (ii) a global appearance model of the enclosed object; and (iii) a set of distinctive object's landmarks. While the global appearance model captures image statistics as in graph-cut approaches [BJ01, RKB04], it is the set of local fg/bg appearance models placed along the boundary that enables accurate object delineation. The distinctive object's landmarks organized in a star-shaped model (Fig. 4.3, left) help to prevent the contour from sliding along itself and to control the level of non-rigid deformations. The landmarks are also used to robustly estimate a rigid transformation between the frames to "pre-warp" the contour, which significantly speeds-up the inference. In addition, the control points of the roto-curve, as well as the local fg/bg models and the landmarks are maintained through time, which provides us with different types of temporal correspondences.

[4] "Foreground/background" terminology, "fg/bg" in short, merely refers here to inside and outside of the roto-curve; it does not imply that the object stands at the forefront of the 3D scene with a background behind it.

Given a colour image $\mathscr{I} = \{\mathbf{I_p}\}_{\mathbf{p}\in\Omega}$, a conditional graphical model (Fig. 4.2) is defined through the energy function

$$E(\mathscr{X},\mathscr{Y};\mathscr{I}) := E^C(\mathscr{X};\mathscr{I}) + E^L(\mathscr{Y};\mathscr{I}) + E^J(\mathscr{X},\mathscr{Y}), \tag{4.1}$$

where E^C and E^L depend only on the roto-curve configuration $\mathscr{X}$ and the landmarks configuration $\mathscr{Y}$ respectively, and E^J links the two together (independently of the image). In the following, we describe these three energy terms in detail.

4.3.1 Curve-based modelling: E^C

While Bézier splines are a popular representation for rotoscoping [AHSS04, LVS+16], we simply consider polygonal shapes here: roto-curve $\mathscr{X}$ is a polyline with N vertices $\mathbf{x}_1 \dots \mathbf{x}_N \in \mathbb{Z}^2$ and N non-intersecting edges $\mathbf{e}_n = (\mathbf{x}_n, \mathbf{x}_{n+1})$, where $\mathbf{x}_{N+1}$ stands for $\mathbf{x}_1$, *i.e.*, the curve is closed. Given an orientation convention (*e.g.*, clockwise), the interior of this curve defines a connected subset $R(\mathscr{X}) \subset \Omega$ of the image pixel grid (Fig. 4.3, left), which will be denoted R in short when allowed by the context.

Energy E^C is composed of two types of edge potentials ψ_n^{loc} and ψ_n^{glob} that relate to local and global appearance respectively:

$$E^C(\mathscr{X};\mathscr{I}) := \sum_{n=1}^{N} [\psi_n^{\text{loc}}(\mathbf{e}_n) + \psi_n^{\text{glob}}(\mathbf{e}_n)]. \tag{4.2}$$

As with classic active contours [KWT88], the first type of potential will encapsulate both a simple ℓ_2-regularizer that penalizes stretching and acts as a curve prior (we are not using second-order smoothing in the current model), and a data term that encourages the shape to snap to strong edges. It will in addition capture colour likelihood of pixels on each side of each edge via local appearance models. The second set of potentials results from the transformation of object-wise colour statistics (discrete surface integral) into edge-based costs (discrete line integrals).

Note that, since we do not impose any constraint on the various potentials, the one specified below could be replaced by more sophisticated ones, *e.g.*, using semantic edges [DZ13] instead of intensity gradients, or using statistics of convolutional features [GDDM14] rather than colour for local and global appearance modelling.

Local appearance model. Each edge $\mathbf{e}_n$ is equipped with a local appearance model $p_n = (p_n^{\text{f}}, p_n^{\text{b}})$ composed of a fg/bg colour distribution and of a rectangular support R_n, with the edge as medial axis and a fixed width in the perpendicular direction (Fig. 4.3, right). Denoting R_n^{in} and R_n^{out} the two equal-sized parts of R_n that are respectively inside and outside R, we construct a simple edge-based energy term (the smaller, the better) that rewards edge-configurations such that colours in R_n^{in} (resp. R_n^{out}) are well

explained by model p_n^{f} (resp. p_n^{g}), and edge $\mathbf{e}_n$ is short and goes through high intensity gradients:

$$\begin{aligned}\psi_n^{\mathrm{loc}}(\mathbf{e}_n) := &- \sum_{\mathbf{p}\in R_n^{\mathrm{in}}} \ln p_n^{\mathrm{f}}(\mathbf{I_p}) - \sum_{\mathbf{p}\in R_n^{\mathrm{out}}} \ln p_n^{\mathrm{b}}(\mathbf{I_p}) \\ &+ \mu\|\mathbf{x}_{n+1}-\mathbf{x}_n\|^2 - \sum_{\mathbf{p}\in\mathbf{e}_n} \lambda\|\nabla\mathscr{I}(\mathbf{p}))\|^2,\end{aligned} \tag{4.3}$$

with μ and λ two positive parameters.

Global appearance model. A global appearance model captures image statistics over the object's interior. As such, it also helps pushing the roto-curve closer to the object's boundary, especially when local boundary terms are not able to explain foreground and background reliably. Defining $p_0 = (p_0^{\mathrm{f}}, p_0^{\mathrm{b}})$ the global fg/bg colour distribution, the bag-of-pixel assumption allows us to define region energy term

$$\sum_{\mathbf{p}\in R} \ln \frac{p_0^{\mathrm{b}}(\mathbf{I_p})}{p_0^{\mathrm{f}}(\mathbf{I_p})}. \tag{4.4}$$

This discrete region integral can be turned into a discrete contour integral using one form of the discrete Green theorem [Tan82]. Using horizontal line integrals for instance, we get

$$\sum_{\mathbf{p}\in R} \ln \frac{p_0^{\mathrm{b}}(\mathbf{I_p})}{p_0^{\mathrm{f}}(\mathbf{I_p})} = \sum_{n=1}^{N} \underbrace{\sum_{\mathbf{p}\in\mathbf{e}_n} \alpha_n(\mathbf{p}) Q(\mathbf{p})}_{:=\psi_n^{\mathrm{glob}}(\mathbf{e}_n)}, \tag{4.5}$$

where $Q(\mathbf{p}) = \sum_{\mathbf{q}\preccurlyeq\mathbf{p}} \ln(p_0^{\mathrm{b}}(\mathbf{I_q})/p_0^{\mathrm{f}}(\mathbf{I_q}))$ is the discrete line integral over pixels $\mathbf{q}$ to the left of $\mathbf{p}$ on the same row, and $\alpha_n(\mathbf{p}) \in \{-1,+1\}$ depends on the direction and orientation, relative to curve's interior, of the oriented edge $\mathbf{e}_n$. In (4.5), the second sum in r.h.s. is taken over the pixel chain resulting from the discretization of the line segment $[\mathbf{x}_n, \mathbf{x}_{n+1}]$ with the final vertex excluded to avoid double-counting.

4.3.2 Landmark-based modelling: E^L

Our model also makes use of a set $\mathscr{Y}$ of M distinctive landmarks $\mathbf{y}_1 \ldots \mathbf{y}_M \in R(\mathscr{X})$ detected inside the object of interest. Similarly to pictorial structures [FGMR10], these landmarks form the leaves of a star-shaped graphical model[5] with a virtual root-node $\mathbf{y}_0$. This part of the model is defined by leaf potentials $\phi_m(\mathbf{y}_m)$ and leaf to root potentials $\varphi_m(\mathbf{y}_0, \mathbf{y}_m)$:

[5] The star shape is used for its simplicity but could be replaced by another tree-shaped structure.

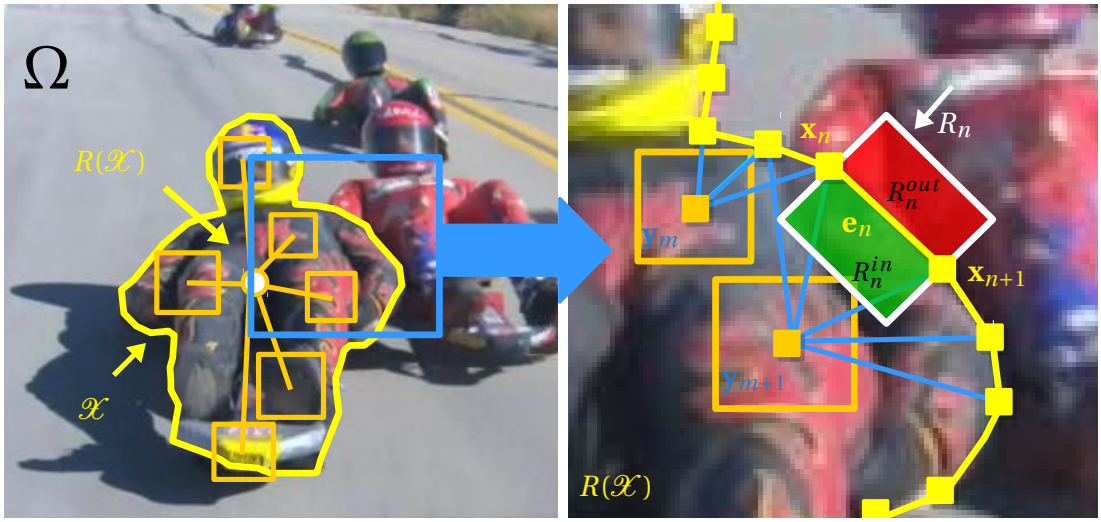


Figure 4.3: **Structure and notations of proposed model**. (Left) A simple closed curve $\mathscr{X}$ outlines the object region $R(\mathscr{X})$ in the image grid Ω. Several landmarks, forming a star-shaped graphical model, are defined in this region. (Right) Each edge $\mathbf{e}_n$ of the closed polyline defines a region R_n that staddles $R(\mathscr{X})$; each node $\mathbf{x}_n$ of the polyline is possibly connected to one or several landmarks.

$$E^L(\mathscr{Y};\mathscr{I}) := \sum_{m=1}^{M} \phi_m(\mathbf{y}_m) + \sum_{m=1}^{M} \varphi_m(\mathbf{y}_0,\mathbf{y}_m). \tag{4.6}$$

Each landmark is associated with a model, *e.g.*, a template or a filter, that allows the computation of a matching cost at any location in the image. The leaf potential $\phi_m(\mathbf{y}_m)$ corresponds to the negative matching cost for m-th landmark. The pairwise potentials φ_m penalize the difference in ℓ_2-norm between the current configuration and the one, $\hat{\mathscr{Y}}$, estimated in previous frame:

$$\varphi_m(\mathbf{y}_0,\mathbf{y}_m) = \frac{1}{2}\|\mathbf{y}_m - \mathbf{y}_0 - \hat{\mathbf{y}}_m + \hat{\mathbf{y}}_0\|^2. \tag{4.7}$$

4.3.3 Curve-landmarks interaction: E^J

The joint energy $E^J(\mathscr{X},\mathscr{Y})$ captures correlation between object's outline and object's landmarks. Based on proximity, shape vertices and landmarks can be associated. Let denote $n \sim m$ the pairing of vertex $\mathbf{x}_n$ with landmark $\mathbf{y}_m$. Energy term E^J decomposes over all such pairs as:

$$E^J(\mathscr{X},\mathscr{Y}) = \sum_{n\sim m} \xi_{nm}(\mathbf{x}_n,\mathbf{y}_m). \tag{4.8}$$

For each pair $n \sim m$, the interaction potential is defined as:

$$\xi_{mn}(\mathbf{x}_n,\mathbf{y}_m) = \frac{1}{2}\|\mathbf{x}_n - \mathbf{y}_m - \boldsymbol{\mu}_{mn}\|^2, \tag{4.9}$$

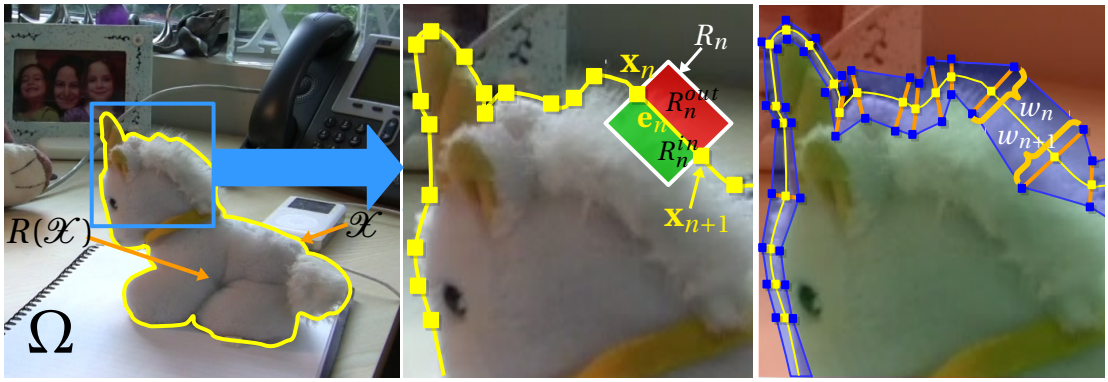


Figure 4.4: **Notations and structure of ROAM+.** (Left) Roto-curve of a scene object in yellow. (Center) Detail of the contour with yellow squares depicting the control points, and an example of a rectangular support that encompasses the local foreground/background classifiers along an edge. (Right) ROAM+ provides the contour tracker with a confidence region of variable width, defining the intermediate region of a trimap, shown in transparent navy blue. Observe that the fuzzy hair of the stuffed toy generates the widest intermediate region, while regions of sharper edges are thiner.

where $\boldsymbol{\mu}_{mn}$ is the landmark-to-vertex shift vector in the first image.

4.4 Introducing ROAM+

A standard trimap is obtained through manual pixel-wise labeling of the image. It is then fed to an alpha-matting algorithm as a means to provide samples for foreground/background color characterization and to define the region where alpha values (or mixing coefficients) must be computed [AB06]. However, in current production pipelines, artists feel more naturally inclined to define the trimaps with auxiliary roto-curves that are easy and intuitive to manipulate. In this sense, a trimap is better understood as an intermediate or *slack* ribbon-like region that separates what is definitely background from what is definitely foreground.

As for rotoscoping, artists prefer to adhere to the prevalent roto-curve paradigm rather than to scribble or pixel-labeling approaches. A logic step is to devise the trimap modeling as an extension of ROAM. The idea is to allow the user to initialize and correct the roto-curves, and automatically estimate the slack regions from them. Furthermore, foreground/background classifiers localized on the edges of ROAM's roto-curve readily provide the elements that determine what are the "uncertain" or "intermediate" regions.

In that order of ideas, we define the problem of estimating the slack region (or indirectly, the trimap) as determining the width of the region of uncertainty at each

control point of the roto-curve (See Fig. 4.4 right). Thus, we parameterize the trimap with a width vector $\mathbf{w} = (w_1 \dots w_N)$ such that the slack region width is $w_n \geq 0$ at control point $\mathbf{x}_n$.

Specifically for ROAM, every edge element $\mathbf{e}_n$ is equipped with a local foreground model p_n^{f} and a corresponding background model p_n^{b}. From these, we can define a local color model confidence measure at control point $\mathbf{x}_n$, function of slack region width w_n:

$$C_n(w_n; \mathscr{X}) = 1 - \frac{\int_{\mathscr{S}^+} p_n(\mathbf{I_s})\, d\mathbf{s} + \int_{\mathscr{S}^-} \big(1 - p_n(\mathbf{I_s})\big)\, d\mathbf{s}}{2\, w_n}, \tag{4.10}$$

where $\mathscr{S}^+, \mathscr{S}^-$ are the line segments $[\mathbf{x}_n, \mathbf{x}_n + w_n \hat{\mathbf{n}}_{\mathbf{x}_n}]$ and $[\mathbf{x}_n, \mathbf{x}_n - w_n \hat{\mathbf{n}}_{\mathbf{x}_n}]$, with $\hat{\mathbf{n}}_{\mathbf{x}_n}$ the unit vector normal to the curve at $\mathbf{x}_n$, and foreground likelihood p_n at node n is defined as:

$$p_n(x) = \frac{p_n^{\mathrm{f}}(x)}{p_n^{\mathrm{f}}(x) + p_n^{\mathrm{b}}(x)}. \tag{4.11}$$

Intuitively, our trimap model should favor width sections enclosing regions with low classifier confidence, while clear boundary parts (high model confidence) should coincide with very thin slack sections. Other priors should be encouraged too. For example, smaller trimap regions should be preferred. We also take into account the role of intensity gradients: the borders of the slack region, as opposed to the actual object's border captured by the roto-curve, are more likely to exhibit low gradients. Finally, we encourage the spatial regularity of the slack region by discouraging abrupt width changes in neighboring control points. All these ideas result in the energy-based model definition:

$$\begin{aligned} E^W(\mathbf{w}; \mathscr{I}, \mathscr{X}) := \sum_{n=1}^{N} \big(C_n(w_n) + \kappa \|\nabla \mathscr{I}(\mathbf{x}_n + w_n \hat{\mathbf{n}}_{\mathbf{x}_n})\| + \kappa \|\nabla \mathscr{I}(\mathbf{x}_n - w_n \hat{\mathbf{n}}_{\mathbf{x}_n})\| \big) + \\ \gamma \|\mathbf{w}\|^2 + \eta \sum_{n=1}^{N-1} \|w_n - w_{n+1}\|, \end{aligned} \tag{4.12}$$

with κ, γ, and η positive parameters. Optimization of the proposed model is explained in Section 4.6.

4.5 Using ROAM

Sequential alternating inference. Using ROAM to outline the object of interest in a new image amounts to solving the discrete optimization problem:

$$\min_{\mathscr{X}, \mathscr{Y}} E(\mathscr{X}, \mathscr{Y}; \mathscr{I}), \tag{4.13}$$

where E is defined by (4.1) and depends on previous curve/landmarks configuration $(\hat{\mathscr{X}},\hat{\mathscr{Y}})$ through several of its components. Despite this problem could be formulated as an integer linear program, we opt for simpler alternating optimization with exact minimization at each step which converges within a few iterations.

In the first step, we fix the roto-curve $\mathscr{X}$ and find the best configuration of landmarks $\mathscr{Y}$ using dynamic programming (DP). Exact solution for such a problem can be obtained in two passes, solving exactly

$$\min_{\mathbf{y}_0}\min_{\mathbf{y}_{1:M}}\sum_{m=1}^{M}\big(\phi_m(\mathbf{y}_m)+\varphi_m(\mathbf{y}_0,\mathbf{y}_m)+\sum_{n\sim m}\xi_{mn}(\mathbf{x}_n,\mathbf{y}_m)\big). \tag{4.14}$$

Default implementation leads to complexity $\mathscr{O}(MS^2)$, with S the size of individual landmark state-spaces, *i.e.*, the number of possible pixel positions allowed for each. However, the quadratic form of the pairwise terms allows making it linear in the number of pixels, *i.e.*, $\mathscr{O}(MS)$, by resorting to generalized distance transform [FGMR10].

In the second step, we fix the landmarks $\mathscr{Y}$ and find the best configuration of contour $\mathscr{X}$. This is a classic first-order active contour problem. Allowing continuous values for nodes coordinates, a gradient descent can be conducted with all nodes being moved simultaneously at each iteration. We prefer the discrete approach, whereby only integral positions are allowed and dynamic programming can be used [AWJ90]. In that formulation, exact global inference is theoretically possible, but with a prohibitive complexity of $\mathscr{O}(NP^3)$, where $P=\mathrm{card}(\Omega)$ is the number of pixels in images. We follow the classic iterative approach that considers only D possible moves $\Delta\mathbf{x}$ for each node around its current position. For each of the D positions of first node $\mathbf{x}_1$, Viterbi algorithm provides the best moves of all others in two passes and with complexity $\mathscr{O}(ND^2)$. Final complexity is thus $\mathscr{O}(ND^3)$ for each iteration of optimal update of previous contour, solving:

$$\min_{\Delta\mathbf{x}_1}\min_{\Delta\mathbf{x}_{2:N}}\sum_{n=1}^{N}\big(\psi_n^{\mathrm{loc}}(\mathbf{e}_n+\Delta\mathbf{e}_n)+\psi_n^{\mathrm{glob}}(\mathbf{e}_n+\Delta\mathbf{e}_n)+\sum_{m\sim n}\xi_{mn}(\mathbf{x}_n+\Delta\mathbf{e}_n,\mathbf{y}_m)\big). \tag{4.15}$$

Note that sacrifying optimality of each update, the complexity could even been reduced as much as $\mathscr{O}(ND)$ [WS92].

Given some initialization for $(\mathscr{X},\mathscr{Y})$[6], we thus alternate between two *exact* blockwise inference procedures. This guaranties convergence toward a local minima of joint energy $E(\mathscr{X},\mathscr{Y};\mathscr{I})$. Also, the complexity of each iteration is linear in the number of vertices and landmarks, linear in the number of pixels, and cubic in the small number of allowed moves for a curve's vertex.

[6] Recall that, for a first frame, initialization of closed contour $\mathscr{X}$ can be provided by the user through manipulation of Bézier curves or any other interactive procedure. Furthermore, $\mathscr{Y}$ is automatically initialized once a closed contour is defined through MSER feature detection inside the roto-curve. For next frames, initialization is provided by previous solution of the pair $(\mathscr{X},\mathscr{Y})$, or further human interaction.

Online learning of appearance models. Local fg/bg colour models p_ns and global colour model p_0 are GMMs. Given the roto-curve in the initial frame, these GMMs are first learned over region pairs $(R_n^{\text{in}}, R_n^{\text{out}})$s and $(R, \Omega \setminus R)$ respectively and subsequently adapted through time using Stauffer and Grimson's classic technique [SG99].

Selection and adaption of landmarks. A pool of distinctive landmarks is maintained at each instant. They can be any type of classic interest points. In order to handle texture-less objects, we use maximally stable extremal regions (MSERs) [MCUP04], although we do not enter in the discussion of which feature/region detector would be best for our model. Each landmark is associated with a correlation filter whose response over a given image area can be computed very efficiently [HCMB15]. At any time, landmarks whose filter response is too ambiguous are deemed insufficiently discriminative and removed from the current pool in the same way tracker loss is monitored in [HCMB15]. The collection is re-populated through new detections. Note that correlation filters can be computed over arbitrary features and kernelized [HCMB15]; for simplicity, we use just grayscale features without kernel function.

Allowing topology changes: contour reparametrization. Using a closed curve is crucial to comply with rotoscoping workflows and allows the definition of a rich appearance model. Also, it prevents abrupt changes of topology. While this behavior is overall beneficial (See §4.7), segmenting a complete articulated 3D object as in Fig. 1 might turn difficult. Roto-artists naturally handle this by using multiple independent roto-curves, one per meaningful part of the object. As an alternative for less professional, more automatic scenarios, we propose to make ROAM benefit from the best of both worlds: standard graph-cut based segmentation [BJ01], with its superior agility, is used to *propose* drastic changes to current curve, if relevant. Space-dependent unaries are derived in ad-hoc way from both global and local colour models and combined with classic contrast-dependent spatial regularization.[7] The exact minimizer of this instrumental cost function is obtained through graph-cut (or its dynamic variant for efficiency [KT07]) and compared to the binary segmentation associated to the current shape $\mathscr{X}$. At places where the two differ significantly, a modification of current configuration (displacement, removal or addition of vertices) is proposed and automatically accepted if it reduces the energy $E(\mathscr{X}, \mathscr{Y}; \mathscr{I})$.

[7] Note that this instrumental energy is too poor to compete on its own with our proposed model, but is a good enough proxy for the purpose of proposing possibly interesting new shapes at certain instants. It is also very different from the one in the final graph-cut of VIDEO SNAPCUT where unaries are based on the already computed soft segmentation to turn it into a hard segmentation. Also, graph-cut segmentation is the final output in VIDEO SNAPCUT, unless further interaction is used, while we only use it to explore alternative topologies under the control of our joint energy model.

Relative weights. The relative weight of the local appearance was 50, global appearance 0.002, $\lambda = 1.0$, $\mu = 0.75$, $P = \{9 \times 9\}$, $M = 20$ and other weights were set to 1. The aforementioned numbers correspond to our best tested configuration, and were found by cross-validation as described in the experimental part.

4.6 Using ROAM+

The model described in Eq. 4.12 is somewhat independent of ROAM (Eq. 4.1). However, in order to solve for the details of Eq. 4.10, ROAM needs to be executed before any attempts to solve for ROAM+. Other than that, Eq. 4.12 can be solved exactly in a single pass using DP, by allowing the width vector to only take integer values. Moreover, our trimap tracking can be updated after interactive modification of the roto-curve. This ability makes our method highly desirable in real scenarios.

It must be observed that ROAM+ enforces a slack region symmetry around the main roto-curve. This constraint facilitates control over undesired trimap solutions (like inversion of the regions) and eases optimization (with respect to, *e.g.*, a model with independent inner and outer width chains). However, the same model might also affect results in cases where the uncertainty region is strongly asymmetrical. In practice, we do not observe strong deviation from acceptable solutions.

We set the weights by hand, and they are specified as follows: $\kappa = 3.3\text{e}^{-4}$, $\gamma = 5\text{e}^{-3}$, and $\eta = 2.5\text{e}^{-4}$.

4.7 Results

We report experimental comparisons that focus on the minimum input scenario: an initial object selection (curve or mask, depending on the tool) is provided to the system and automatic object segmentation is produced in the rest of the shot. We do not consider additional user interactions.

Datasets. We evaluate our approach on the recent CPC rotoscoping dataset [LBSW16]. It contains 9 videos consisting of 60 to 128 frames which represent typical length of shots for rotoscoping. These sequences were annotated by professional artists using standard post-production tools. We also provide qualitative results on shots from the ROTO++ [LVS+16] dataset for which the authors have not released the ground-truth yet, as well as from the VIDEO SNAPCUT dataset [BWSS09].

In addition to that, we use the DAVIS video segmentation dataset [PPTM+16] which comprises 50 challenging sequences with a wide range of difficulties: large occlusions, long-range displacements, non-rigid deformations, camouflaging effects

Table 4.1: Quantitative evaluation on CPC dataset (*: partial evaluation only, see text)

Method	Avg. Accuracy	Avg. IoU	Time (s) / frame min	max	avg
GCUT [RKB04] + KCF [HCMB15]	.891	.572	0.394	0.875	0.455
AE ROTOBRUSH [BWSS09]	.992	.895	—	—	—
ROTO++(1 keyframe) [LVS+16]	.969	.642	—	—	0.108
ROTO++(2 keyframes) [LVS+16]	.974	.691	—	—	0.156
CPC [LBSW16]	.998*	.975*	—	—	—
NLCV [FI14]	.896*	.194*	—	—	—
BSVS [MPWSH16b]	.991	.872	—	—	—
OBJECTFLOW [TYB16]	.968	.502	—	—	—
ROAM: Baseline Conf.	.993	.932	0.011	0.155	0.040
ROAM: Lean Conf.	**.995**	.938	0.092	0.377	0.102
ROAM: Medium Conf.	**.995**	.939	0.279	0.875	0.652
ROAM: Full Conf.	**.995**	**.951**	0.874	8.78	3.052
ROAM+	**.996**	**.954**	—	—	—

and complex multi-part objects. Let us note that this dataset is intended to benchmark pixel-based video segmentation methods, not rotoscoping tools based on roto-curves.

Evaluation Metrics. We use standard video segmentation evaluation metrics and report the average *accuracy*, *i.e.*, the proportion of ground-truth pixels that are correctly identified, and the more demanding average *intersection-over-union* (IoU), *i.e.*, the area of the intersection of ground-truth and extracted objects over the area of their union. We also report runtimes and evolution of IoU as sequences proceed.

Baselines. We compare with several state-of-the-art methods. Our main comparison is with recent approaches that rely on a closed-curve, *i.e.*, CPC [LBSW16] and ROTO++ [LVS+16]. We initialize all methods with the same object and measure their performance over the rest of each sequence. Since ROTO++ requires at least two key-frames to benefit from its online shape model, we report scores with letting the method access the ground-truth of the last frame as well. We also run it with the initial keyframe only, a configuration in which ROTO++ boils down to the Blender planar tracker.

In addition to that, we also compare with two approaches based on pixel-wise labelling: JUMPCUT [FZL+15], and VIDEO SNAPCUT [BWSS09] as implemented in After Effects ROTOBRUSH and three recent video-segmentation approaches [FI14, MPWSH16b, TYB16]. As a naive baseline, we use a combination of a bounding-box tracker [HCMB15] and GRABCUT [RKB04].

Table 4.2: Quantitative comparisons on DAVIS dataset

Method	Average Accuracy			Average IoU			Time / frame (s)		
	Val. set	Tra. set	Full set	Val. set	Tra. set	Full set	min	max	avg
GRABCUT [RKB04] + Tracker [HCMB15]	0.896	0.914	0.907	0.277	0.296	0.289	0.405	0.675	0.461
JUMPCUT [FZL+15]	0.952	0.957	0.956	0.570	0.632	0.616	—	—	—
AE ROTOBRUSH [BWSS09]	0.951	0.942	0.946	0.533	0.479	0.500	—	—	—
ROTO++ (1 keyframe)[LVS+16]	0.910	0.922	0.917	0.248	0.310	0.284	—	—	0.118
ROTO++ (2 keyframes) [LVS+16]	0.925	0.933	0.930	0.335	0.394	0.358	—	—	0.312
NLCV [FI14]	0.948	0.963	0.957	0.551	0.701	0.641	—	—	—
BSVS [MPWSH16b]	**0.966**	**0.974**	**0.971**	**0.683**	0.709	0.665	—	—	—
OBJECTFLOW [TYB16]	—	—	—	0.600	**0.732**	**0.711**	—	—	—
ROAM: Baseline Conf.	0.930	0.937	0.932	0.358	0.385	0.377	0.017	0.113	0.049
ROAM: Lean Conf.	0.935	0.937	0.936	0.409	0.417	0.412	0.187	0.641	0.342
ROAM: Medium Conf.	0.942	0.952	0.948	0.532	0.591	0.564	0.302	1.785	0.746
ROAM: Full Conf.	0.952	0.956	0.953	0.583	0.624	0.615	0.546	7.952	3.058

Ablation study. To evaluate the importance of each part of our model, we consider 4 different configurations:

- Baseline: negative gradient with ℓ_2-regularizer;
- Lean: baseline + local appearance model;
- Medium: lean + landmarks;
- Full: medium + automatic re-parametrization and global appearance model.

For all configurations, we used cross-validation (maximizing the mean IoU) on the training fold of the DAVIS dataset to set the parameters and kept them fixed for all experiments.

Quantitative results. The quantitative results for the CPC dataset are summarized in Tab. 4.1. While average accuracy is quite similar and saturated for all methods, all configurations of ROAM outperform all baselines. In terms of IoU, all versions of ROAM outperform significantly all others with the full configuration being the best. The reason why landmarks ("medium conf.") do not add much to ROAM is that the CPC dataset does not exhibit many large displacements. The CPC method [LBSW16] was evaluated only on the first ten frames of each sequence since their authors have released results only on these frames and not yet their code. Hence, the scores reported in Tab. 4.1 for CPC are based on partial videos, as opposed to the scores for all the other methods (including ours). When similarly restricted to the first 10 frames, ROAM performs on par with CPC for all the sequences except "drop" sequence. This sequence shows a water drop falling down – a transparent object,

Table 4.3: Different types of contour warping for handling long displacements on a subset of sequences of the DAVIS dataset.

Warping method	Average Accuracy	Average IoU
Optical flow	0.878	0.312
Node projection from landmark tracking	0.906	0.480
Robust rigid transf. from landmarks	**0.934**	**0.581**

making color models (both local and global) useless if not harmful, and exhibits a very smooth round shape. For this sequence, the CPC method [LBSW16] performs better since it uses Bézier curves and relies solely on the strength of the image gradients.

Results for the DAVIS dataset are reported in Tab. 4.2. While our method is on par with JUMPCUT (pixel-wise labelling), we again significantly outperform ROTO++ by almost 25 IoU points (note that using ROTO++ with only two keyframes is not a typical scenario, however, this shows how complementary our approaches are). Despite [MPWSH16b] is better by 100 and [TYB16] by 17 IoU points on DAVIS, our model outperforms [MPWSH16b] by 80 and [TYB16] by 450 points on the CPC. In other words, our approach should in the worst case be considered on par. However, we would like to stress that [FI14, MPWSH16b, TYB16] are not our (main) competitors since all are based on pixel-wise labelling and as such **cannot** provide the same flexibility for rotoscoping as the closed contour counterpart [LVS+16]. Note, that we could not provide more quantitative comparisons since results/implementations of other methods were not available from the authors. In particular, comparison with the CPC method [LBSW16] would be interesting since the DAVIS dataset [PPTM+16] exhibits many large displacements and major topology changes.

Comparing the different configurations of ROAM – local appearance models add 3 points, landmarks 15 and global model with re-parametrization another 5 points – demonstrates the importance of all components of our framework. To examine behaviour of each method in detail, we report IoU score for each frame in Fig. 4.8, with in addition the effect of varying the size of the displacement space in ROAM (from windows of 3×3 to 13×13 pixels) represented with a blue shadow. It can be seen that ROAM is more robust in time, not experiencing sudden performance drops as others.

Importance of landmarks and warping. Using alternating optimization has one more benefit. We can use the predicted position of landmarks in the next frame to estimate the transformation between the two and "warp" the contour to the next frame. This allows us to reduce the number D of possible moves of nodes which i) significantly speeds-up the algorithm, ii) allows us to handle large displacement and iii) provides better control of non-rigid deformations.

Figure 4.5: **Qualitative results on the DAVIS dataset**: Comparisons on *blackswan* and *car-roundabout* sequences, between (from top to bottom for each sequence): JUMPCUT, ROTOBRUSH, ROTO++ and ROAM.

We have experimented with three settings for warping of contour: using a smoothed optical flow masked as in Chapter 3, moving each node by averaging the motion of all landmarks connected to given node and robustly estimating similarity transformation with RANSAC from position of landmarks. Table 4.3 and Fig. 4.9 show the effect of using robustly estimated similarity transformation from position of landmarks.

Global colour models and reparametrization. We investigated the effects of adding reparametrization and global colour models to our framework. The numeric benefits of these elements can be seen in Tab. 4.2 and qualitative results on the *surfer* sequence from VIDEO SNAPCUT dataset are provided in Figs. 4.6 and 4.7. Observe that the local colour models are a powerful way to capture local appearance complexities of an object through a video sequence. However, self-occlusions and articulated motion can cause these models to fail (right arm crossing the right leg of the surfer). Our contour reparametrization allows the efficient handling of this situation (explained at the end of Section 4.5). Furthermore, the beneficial effect of the global colour models can be observed in Fig. 4.7, where the right foot of the surfer is successfully tracked along the whole video.

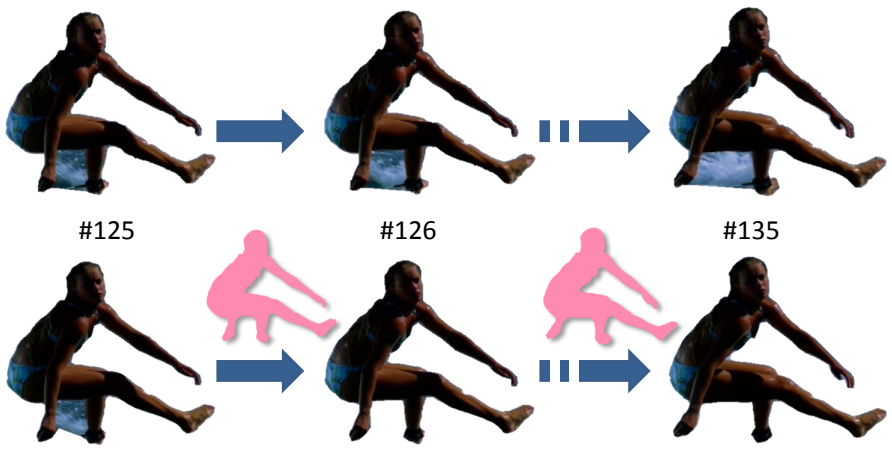


Figure 4.6: **Using proposals based on graph-cut**: Proposals (in pink) obtained through graph-cut minimization of an instrumental labeling energy using current colour models allows ROAM to monitor and accommodate drastic changes of object's outline (Bottom). Without this mechanism, parts of surrounding water get absorbed in surfer's region, between the leg and the moving arm (Top).

Roto-curve refinement with alpha-matting. In order to asses quantitatively the merit of ROAM+ as a possible way to improve roto-curve tracking, we propose to exploit the alpha-mattes derived from our trimaps with [HRR+11]. The refinement process consists first in extracting the matte level set at level 0.5. This curve is discretized and matched with ROAM's roto-curve (using Hungarian bi-partite graph matching). Each ROAM control point is finally moved to the position of its correspondent on the level set. Our hypothesis is that this refinement step should improve the quality of the rotoscoping, and thus benefit not only to compositing but also to other rotoscoping-based edits. This refinement is illustrated for a real example in

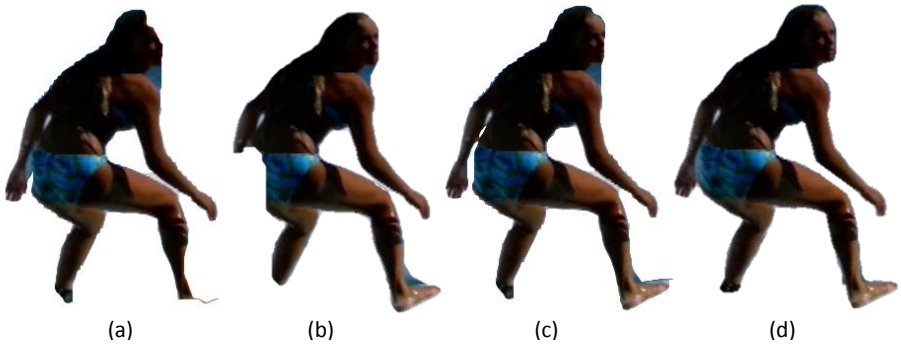


Figure 4.7: **Assessing first part of the model**. (a) Edge strength only; (b) Global colour model; (c) Edge strength combined with global colour model; (d) With full cost function E^C, including local colour modeling, on frame 13 from *surfer* sequence.

Fig. 4.11. Furthermore, we add a final row to Table 4.1, showing the gain obtained by using our roto-curve refinement method. A significant gain under IoU regime can be appreciated. In fact, overall, our method delivers the best score (excluding CPC, due to reasons previously exposed). The trimap reasoning provides ROAM with the capabilities of dealing with strong transparencies like the ones present in the "drop" sequence in Fig. 4.12. In fact, ROAM, and all the previous methods fail to correctly delineate such a difficult curve.

Finally, in Figs. 4.13 and Fig. 4.14, the reader can observe trimaps obtained by the proposed method. As can be seen, clear border fragments correspond to thin intermediate areas (see, for example, Sunset sequence) while more ambiguous ones yield larger slack regions (see locations where the roto-curve goes through the junction between the arms and the body of the toy in Minion sequence for instance). Fuzzy or blurred borders also lead to large intermediate areas, as seen for example around the fur of the stuffed horse in Fig. 4.14. Our automatic trimap tracking is thus behaving in the desired way.

Qualitative results. Result samples on several sequences from DAVIS dataset in Fig. 4.5 demonstrate the superior robustness of ROAM compared to other approaches when rotoscoping of the first image only is provided as input (and last image as well for ROTO++). Additional results obtained by ROAM on a variety of sequences are illustrated in Fig. 4.15.

Timing breakdown. Table 4.4 provides detailed timing breakdown for our algorithm. These timings were obtained on an Intel Xeon 32@3.1GHz CPU machine with 8GB RAM and Nvidia GeForce Titan X GPU. Note that only part of the approach (evaluation of various potentials and dynamic programming) was run on the GPU. In particular,

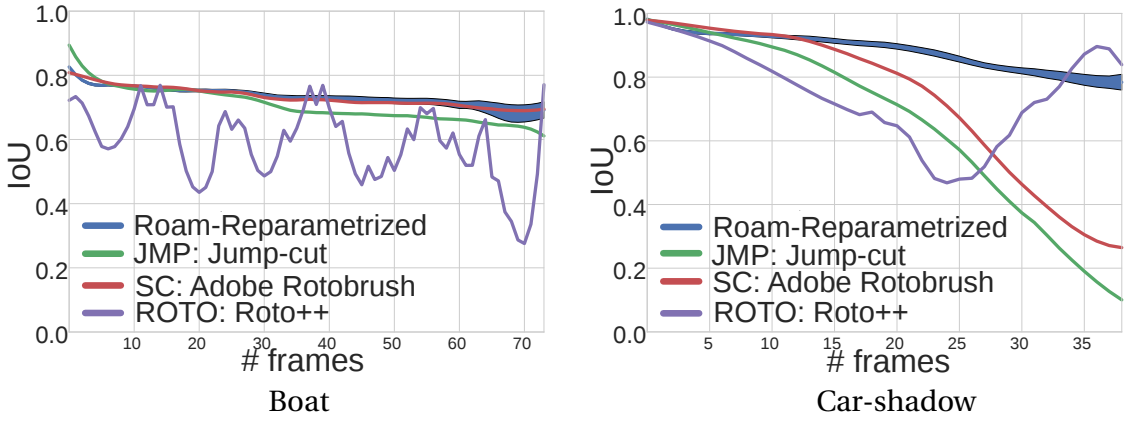


Figure 4.8: **Evolution of IoU for different sequences of the DAVIS dataset**. For our method, the blue shadow indicates influence of varying the label space size for each node (set of possible moves in dynamic programming inference).

the re-parametrization steps could also be easily run on the graphics card, yielding real-time performance.

Table 4.4: Timing details for full configuration of ROAM in seconds-per-frame.

Step	Min.	Max.	Avg.
DP Contour	0.018	0.113	0.084
DP Landmarks	0.003	0.072	0.052
Local models edge terms	0.342	0.671	0.581
Other terms	0.012	0.015	0.013
Reparametrization	0.032	7.403	2.226

Convergence. Fig. 4.10 demonstrates that the alternating optimization described in §4.5 converges quickly within a few iterations.

4.8 Conclusion

We have introduced ROAM, a model to capture the appearance of the object defined by a closed curve. This model is well suited to conduct rotoscoping in video shots, a difficult task of considerable importance in modern production pipelines. We have demonstrated its merit on various competitive benchmarks. Beside its use within a full rotoscoping pipeline, ROAM could also be useful for various forms of object editing that require both accurate enough segmentation of arbitrary objects in videos and tracking through time of part correspondences, *e.g.* [KRFB06, RAKRF08]. Due to

Figure 4.9: **Benefit of landmarks-based modeling**. Automatically detected landmarks (orange bounding boxes) are accurately tracked on the *plane* sequence. This further improves the control of the boundary (bottom), as compared to ROAM without landmarks (top).

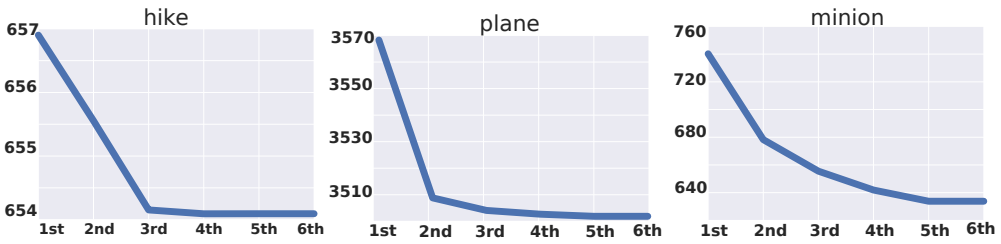


Figure 4.10: **Energy vs. number of iterations** on three sequences from the experimental datasets.

its flexibility, ROAM can be easily extended; in particular, we have presented a trimap tracking algorithm within the ROAM framework, coined ROAM+. This extension of ROAM builds a missing link between two important steps in the video compositing pipelines: rotoscoping and video matting. Future directions include exploring the recent ROTO++ and its powerful low-dimensional shape model in combination with our rich appearance modeling.

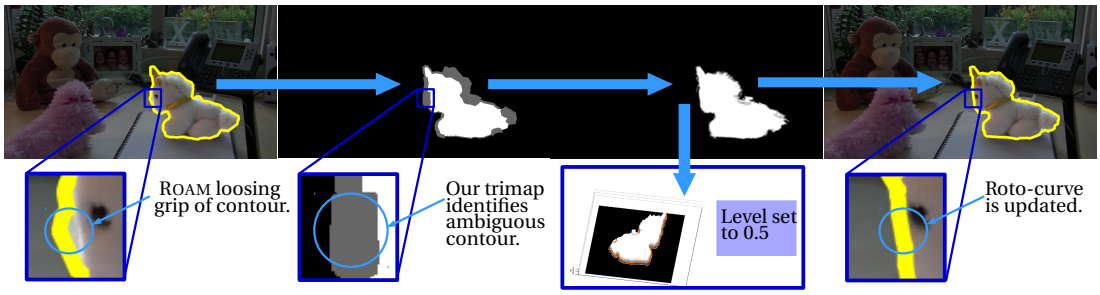


Figure 4.11: **Roto-curve refinement with ROAM+.** Originally tracked roto-curve contains a few errors around the horse face. Initial trimap estimation and matting allow roto-curve refinement.

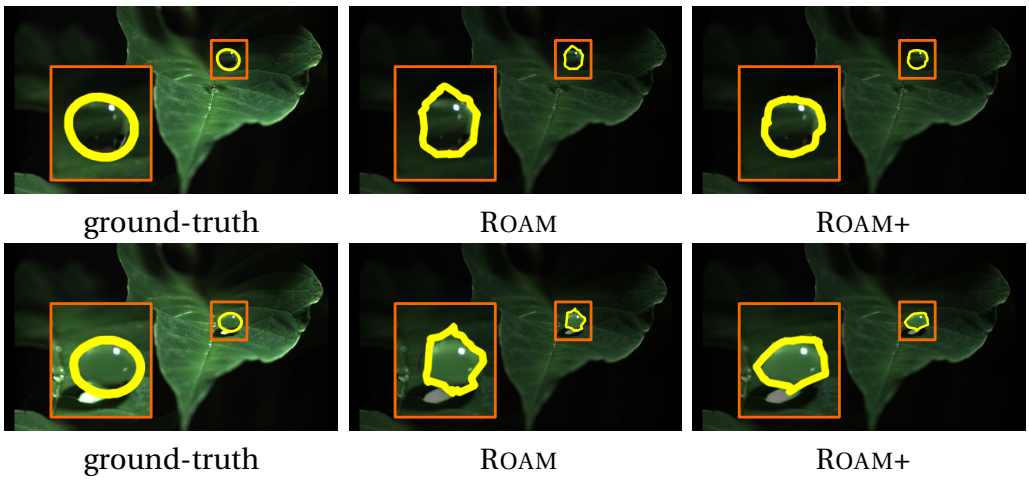
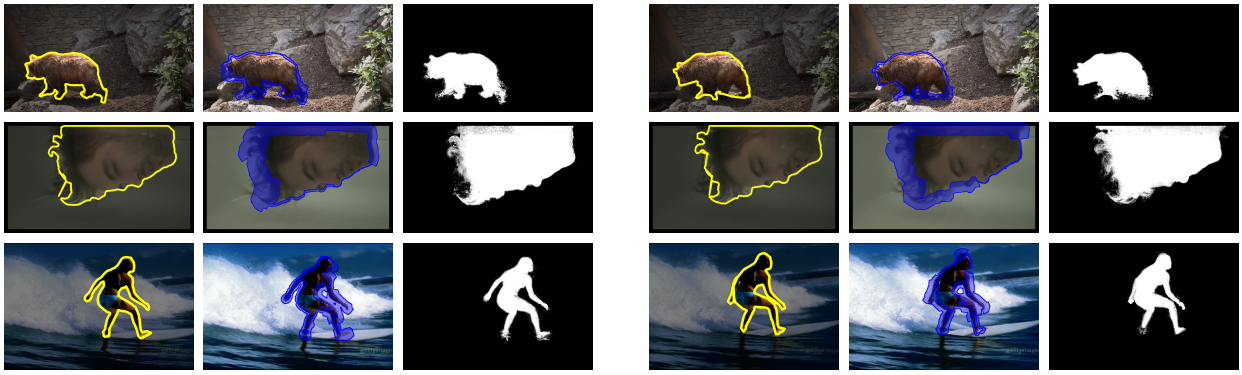


Figure 4.12: **ROAM+ roto-curve refinement in comparison with ROAM.** Detailed visual results (orange window zoomed-in) for two frames of a challenging scene with strong transparency. For each frame (#10 top, and #19 bottom) we display: annotation, roto-curve with ROAM, and ROAM+. Observe that ROAM contour deforms very early in the sequence, presenting irregular border, and departing from the ground-truth. ROAM+ results are smoother and more precise.



(a) (b) (c) (a) (b) (c)

Figure 4.13: **ROAM+ results on the DAVIS, and ROTO++ datasets**. Yellow: Roto-curve (a); Blue: intermediate region of the trimap (b). Alpha-mattes computed by [HRR+11] from our trimaps (c). Results on non-successive frames.

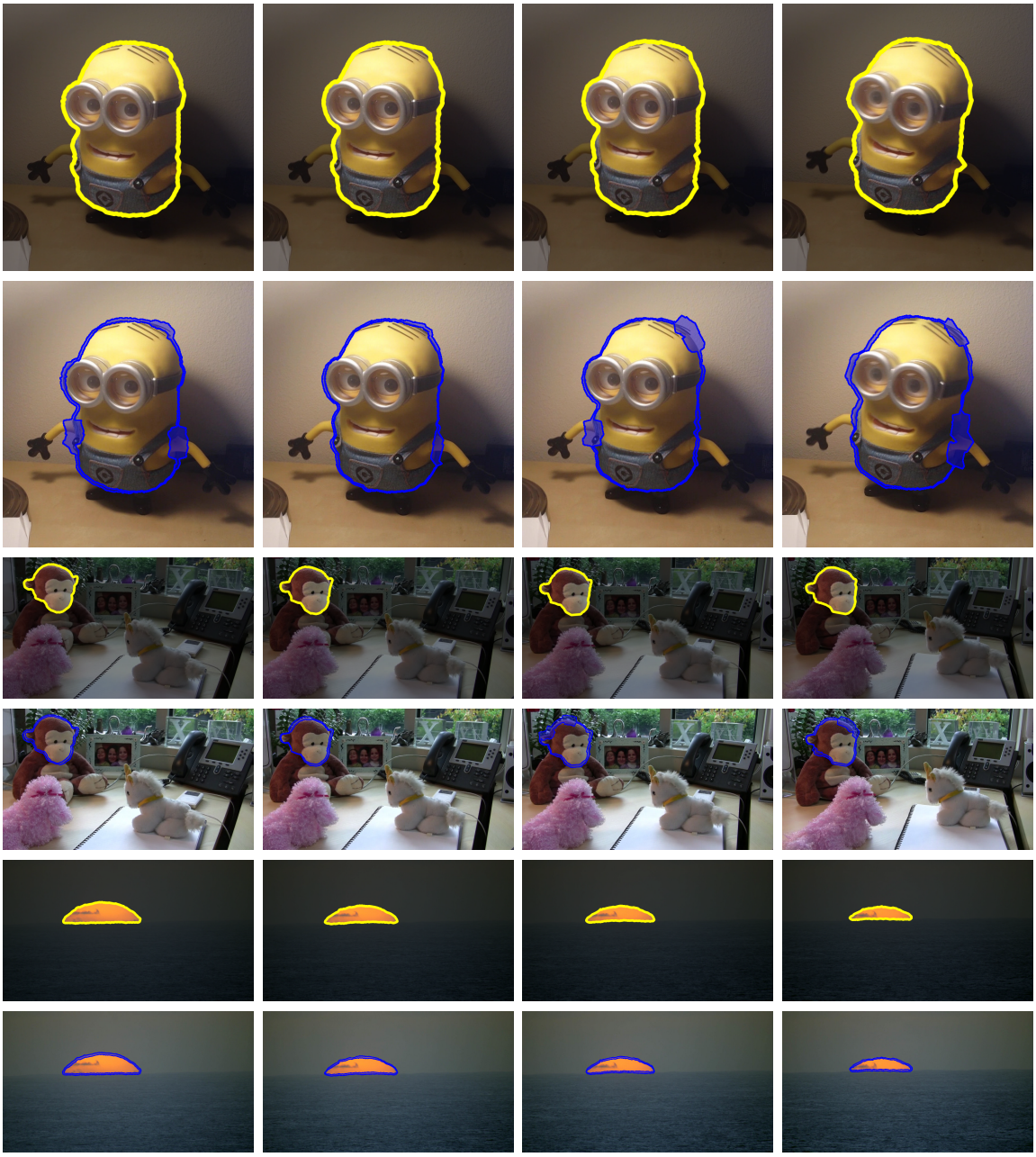
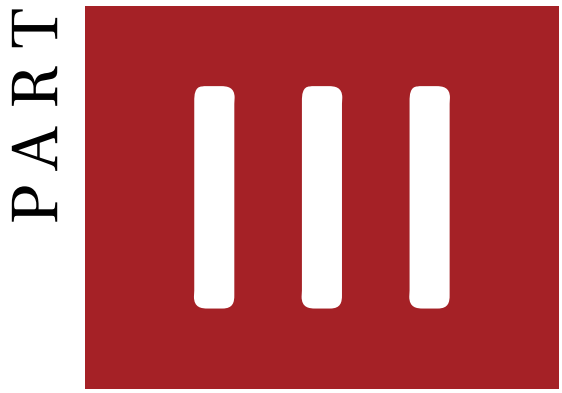


Figure 4.14: **Output trimaps obtained by ROAM+ on the CPC dataset**. Non-succesive frames of the Minion and Monkey-head sequences. Yellow: Roto-curve; Blue: Trimap.

Figure 4.15: **More qualitative results:** Output of ROAM on very different sequences from DAVIS, CPC and VIDEO SNAPCUT datasets among others.



FULLY AUTOMATIC VIDEO ANALYSIS

CHAPTER 5

OCCLUSION DETECTION

The problem of localizing occlusions between consecutive frames of a video is important but rarely tackled on its own. In most works, it is tightly interleaved with the computation of accurate optical flows, which leads to a delicate chicken-and-egg problem. With this in mind, we propose a novel approach to occlusion detection where visibility or not of a point in next frame is formulated in terms of visual reconstruction. The key issue is now to determine how well a pixel in the first image can be “reconstructed" from co-located colors in the next image. We first exploit this reasoning at the pixel level with a new detection criterion. Contrary to the ubiquitous displaced-frame-difference and forward-backward flow vector matching, the proposed alternative does not critically depend on a pre-computed, dense displacement field, while being shown to be more effective. We then leverage this local modeling within an energy-minimization framework that delivers occlusion maps. An easy-to-obtain collection of parametric motion models is exploited within the energy to provide the required level of motion information. Our approach outperforms state-of-the-art occlusion detection methods on challenging datasets.

5.1 Introduction

Detecting occluded areas at each instant of a video sequence is of utmost interest for many computer vision applications. In fact, even though occlusion detection is mostly associated with the problem of computing inter-image correspondences (optical flow for monocular vision, or disparity map in stereo vision), it is very informative on its own. Among other applications, occlusion-based reasoning has been applied to

contour and object tracking [PH07, YLS04], segmentation of multiple objects [WNL08], action recognition [WÖF10], pose estimation [WM08], depth ordering [SDC04], and scene interpretation [TNL14]. Moreover, as mentioned in the introductory chapter of this manuscript, detecting occlusions is key for quality propagation of edits in the post-production pipeline.

In spite of the usual association between motion field estimation and occlusion detection, it is worth noting that physical motion within the scene by itself does not determine if an element is hidden at a given instant. An additional factor is needed in the equation, that is, the observer point of view, or in other words, the observed 2D visual representation of the real 3D world, *i.e.*, the image. This is a well-known fact and limitation of the optical flow as a representation of physical motion [VP89]. When working with a succession of discrete-in-time and discrete-in-space 2D images, one can define that a point of a given image is occluded in the next (or other) image of the sequence, if it is not visible by the observer in the latter.

Many state-of-the-art approaches tackle occlusion detection based on the following question: *Does an image point have a correspondent in the other image that can be confidently identified as physically identical?* In practice, this is indeed evidenced, either implicitly or explicitly, by a wide range of formulations that consider the problem of occlusion detection as inseparable of displacement estimation [ADPS07, FBK16a, KZ01, SLKS05, XCJ08]. This simple question leads however to obstacles in formulating the problem. First, true dense correspondences between images are not easily obtained, especially on occluded pixels where the optical flow (or disparity, likewise) is not well defined. Second, even in non-occluded areas, rapid changes in appearance and scale make the definition and the estimation of unique correspondences difficult.

We strive for an alternative, more direct, approach to occlusion detection. We adopt an image reconstruction viewpoint that frees us, to a large extent, from the need of jointly estimating an accurate, dense motion field. Instead, we only make use of "plausible" motions simply extracted from the image pair. It consists in formulating the task as an image reconstruction problem, instead of a motion and occlusion joint estimation problem. The main idea is to assess whether or not pixel appearance can be equally well explained by its spatial neighborhood in the same frame and by suitable co-located pixels in the other frame. If not, this point is likely to be occluded in the next image. In this way, occlusion detection can be sought as independent of the knowledge of an accurate, univocal motion field between the two images. It suffices to exploit loosely the spatiotemporal coherency in order to select a suitable spatiotemporal neighborhood.

5.2 On true motion and occlusion models

Seeing occlusion detection as only part of a joint motion-occlusion problem requires (1) to model accurately visual motion and (2) to model occlusions in the light of this motion. As we will show, even if the true motion field is known, approaches of this type might fail to produce accurate occlusion maps. As a consequence, we argue that motion should remain an auxiliary variable and not the main object of interest. This idea deeply contrasts with the reasonings one can find in the literature.

A popular idea is that an occlusion is a violation of the optical flow constraint: *"Occlusions generally become apparent when integrated over time because violations of the brightness-constancy constraint of optical flow accumulate in occluded areas"* [ES15].

Other authors make similar claims, without referring to optical flow integration but stating instead that, under Lambertian reflections and the constant illumination assumption, a brightness change between corresponding points indicates an occlusion of the point in the second image [ARS12, XCS+06].

Another assertion is that flow errors occur in occlusion areas or that an occlusion is an explanation of motion mismatching: *"... the most probable reasons for such a situation* [flow mismatching] *is an occlusion problem or an error in the estimated matching"* [ADPS07]. This notion has also been exploited by other works [FBK16a, PVGPO94, YL15] where forward-backward flow inconsistency is used to detect occlusions.

Some authors have proposed that a point is occluded if it switches from one motion layer or segment to another between consecutive frames: *"To consistently reason about occlusions, we examine the layer assignments* [of two points in consecutive frames] *at locations corresponded by the underlying flow fields"* [SSB10]. In our opinion, this argument falls short for non-planar motions and self-occlusions even when optical flow is allowed to deviate from the assumed parametric motion as in [SSB10]. Ambiguities on occlusion estimation from layer assignment are alleviated by enforcing temporal layer consistency [SSB12].

Relying on a joint motion-occlusion estimation leads to a chicken-and-egg situation, which is handled in alternation: "[The algorithm] *iterates between estimating values for* [occlusion map]*, and optimizing the current optical flow estimates by differential techniques"* [SFVG04].

Other approaches include the uniqueness criterion [BBH03], which is known for producing a large amount of false positives [XJM12], and combinations of the criteria explained above. For instance, [IK08] considers flow symmetry in combination with the violations of the optical flow brightness constancy constraint. Another example is [SLKS05] which enforces disparity consistency by labeling points that cannot be matched with another point in the second image as occluded, while proposing a sequential approach that computes occlusion and disparity iteratively. Different views

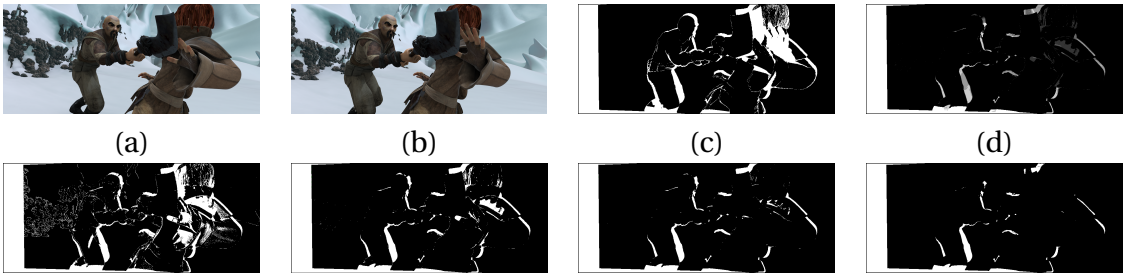


Figure 5.1: **Occlusions from color inconsistencies along *true* flow**. (a-b) Two successive frames of the clean *ambush_2* sequence from the *MPI Sintel* dataset [BWSB12a]. (c) Occlusion ground truth with occlusions shown in white; (d) Norm of the color difference between points matched by the true flow (2D projection of the known 3D motions); (e)-(h) Occlusion maps obtained by thresholding at 0.015, 0.050, 0.100 and 0.250 respectively, the *min-max* normalized (between 0 and 1) color differences along the flow. They are all unsatisfactory (low precision and/or low recall).

on the related problem of finding occlusion borders, but without determining exactly the occlusion regions are found in [AF05, JFN12, RHE+15, SBMR09, SH09, SBM+11].

Doubtless, the underlying motion is indeed helpful to find occlusions. This is further confirmed by the work of Humayun *et al.* [HMAB11] where a large amount of features were used to train a random forest, giving a high importance to motion-based features. In the same way, knowing the occlusion labeling of image pixels clearly helps motion estimation. Specifically, it better guides regularization and smoothing. Many optical flow methods try to deal with occlusions by embedding a discrete state into a continuous numerical scheme [BGLC12], or an aggregation framework [FBK16a], ending with complex, usually joint formulations that improve motion estimation. Similarly, stereo vision approaches that are formulated as discrete label-selection problems can be naturally extended to handle occlusions by adding a label for the occlusion state [KSC01, KZ01], relying on efficient discrete energy optimization techniques.

In order to pin down our claims, let us assume that we know the true motion field for an image pair[1] and let us analyze the most common underlying reasonings for occlusion detection. Figure 5.1 shows results for the occlusion detection by finding violations of the color constancy assumption along this motion, that is by thresholding the so-called displaced frame difference (DFD). If $\mathbf{x}$ and $\mathbf{x}'$ are locations on image grid Ω of two truly corresponding pixels in color images I_1 and I_2 respectively, $\mathbf{x}$ is declared occluded if

$$\left\| I_1(\mathbf{x}) - I_2(\mathbf{x}') \right\| > \varepsilon_c, \tag{5.1}$$

[1] In the sense of the projection of the true physical motion onto the image plane.

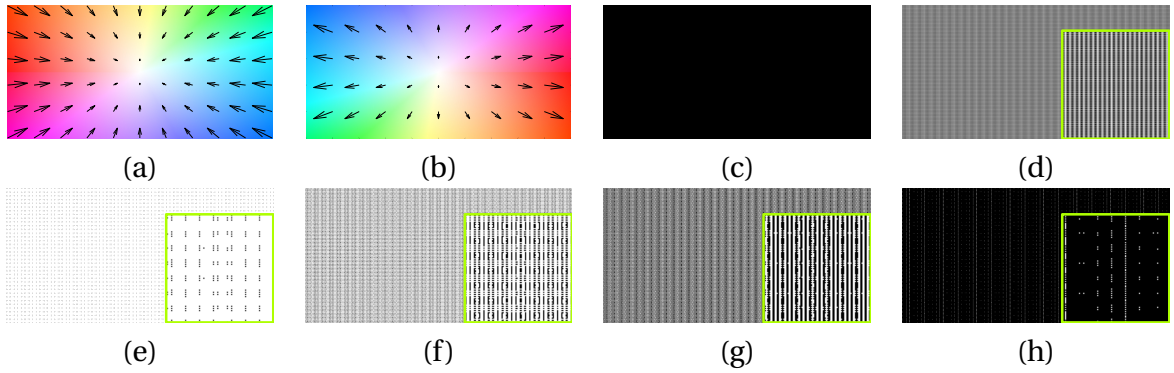


Figure 5.2: **Fictitious occlusions from forward-backward inconsistencies of divergent flows**. (a)-(b) Forward-backward divergent flows associated to a zoom-in (classic hue-saturation color coding at the pixel level, and subset of motion vectors superimposed for better visualization). (c) True occlusion map, devoid of occlusions; (d) Norm of the forward-backward flow difference between points matched by the true flow. Because of interpolations required to evaluate backward flows at non-pixel positions, differences are space dependent (e)-(h) Occlusion maps obtained by thresholding at 0.100, 0.333, 0.500 and 0.900, respectively, the normalized forward-backward differences along the flow. For better visualization of the error pattern, we zoom-in the resulting images (bottom-left corner). False positives occur all over the image grid.

where ε_c is a threshold, which is proposed with variants in [ARS12, XCS+06]. Even though the image sequence in this example does not contain significant illumination changes nor extra post-processing effects like mist or motion blur, the *color constancy criterion is not robust enough to detect occlusions even if the true motion is known.* This is easily verified visually by looking at regions where the norm of the color difference does not have large enough values even across occluded areas (Fig.5.1).

A similar experiment can demonstrate that surprisingly, even when the true optical flow is available, the forward-backward flow consistency criteria might fail to accurately capture the real occlusion map (Fig.5.2). This criterion assumes that given corresponding points $\mathbf{x}$ and $\mathbf{x}'$ and their associated forward and backward flows, $\mathbf{w}_f(\mathbf{x})$ and $\mathbf{w}_b(\mathbf{x}')$, $\mathbf{x}$ is declared occluded if $\|\mathbf{w}_f(\mathbf{x}) + \mathbf{w}_b(\mathbf{x}')\| > \varepsilon_f$, where ε_f is a threshold [ADPS07, BAS07, FBK16a]. Implementation-wise, the backward flow at $\mathbf{x}'$ is obtained by bilinear interpolation, since $\mathbf{x}'$ is generally not an integer grid position. Such a detail by itself generates erroneous flow mismatching which may be enlarged in different situations such as motion discontinuities or zooming. Even if ground-truth motion is available at image grid points, $\mathbf{w}_b(\mathbf{x}')$ may introduce a position drift while going backwards to the first image. Figure 5.2 shows occlusion maps obtained by this criterion for a synthetic zoom. The errors in the occlusion map can be explained by the grid discretization of the flows, making the forward-backward flow difference grow

in several zones of the non-occluded area (whiter pixels in Fig.5.2), and leading to a large number of false positives.

As the aforementioned two criteria are the most commonly used [ADPS07, ARS12, BAS07, FBK16a, XCS+06], the amount of errors they can lead to should not be neglected. Reducing the dependency of occlusion detection on flow quality, as we propose, is one answer to this problem.

5.3 From image reconstructions to occlusion

We consider that the property of being occluded is intrinsic to each one of the points of an image. This means that detecting occlusions may be posed as an independent problem, and not necessarily strongly attached to a per-pixel estimation of motion. We start from the standard concept of an occluded point, represented in the image space through the simplified concept of a pixel, as one that is visible in a first image and not visible in a second image.

A visible to non-visible transition implies a loss of information between the two images. This means that there is a pixel in the first image that cannot be explained using the second image. At a larger scale, to pin things down, suppose a well-defined object present at one instant. The question we ask is: *Can the visual information carried by the object be* ***plausibly*** *explained or "reconstructed" by visual data from the second image?* Failing to perform this reconstruction implies an absence of information and thus, a disappearing object. Occlusion detection can then be defined as a spatiotemporal reconstruction problem.

For this problem to be well-posed there are two main issues. First, quality of the visual reconstruction has to be assessed with respect to a reference information. Incidentally, *the true reconstruction is available here*, and is precisely the same first image! Note that we reason directly on the quality of the reconstruction (with known reference) rather than indirectly on the quality of motion (unknown field). Second, the reconstruction should be *plausible*, meaning that the way we pull information from the second image to reconstruct the first, must be consistent with apparent scene changes. Without the latter condition, one can get away with physically improbable information flows.

We start by focusing on the first aspect of the problem, that is, how to construct an occlusion criterion able to reason on the basis of image reconstruction. The criterion itself assumes that the reconstruction is plausible and consistent. A plausible reconstruction from a pair of images can be generated given a plausible correspondence motion field between them. This defines a natural way of pulling information from one image towards the other. A non-plausible reconstruction would be one that propagates color information between points that do not physically relate. Such a reconstruction is not necessarily useless, as many problems in image processing are not

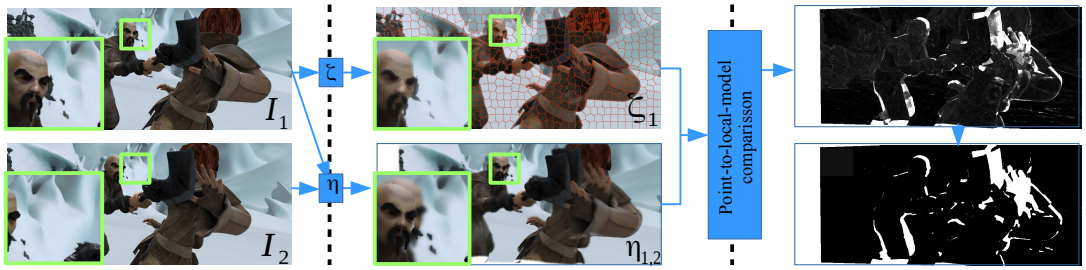


Figure 5.3: **Pipeline for proposed occlusion detection criterion**. Given successive images I_1 and I_2, functions ζ and η generate two "reconstructions" ζ_1 and $\eta_{1,2}$ of I_1. The second reconstruction is obtained from I_2, under the guidance of a given motion field and of I_1 (to preserve the structures of it). An arbitrary window of an occluded zone is zoomed-in for inspection. The likelihood of the color at each pixel of $\eta_{1,2}$ is evaluated under the corresponding local model extracted from ζ_1 at the super-pixel level (*point-to-local-color-model comparison*). This provides a soft-occlusion map that can be either thresholded pixel-wise to obtain a binary occlusion map, or embedded in the unaries of a joint labeling cost function (not shown here).

interested in the interpretation of the correspondence itself, like motion-compensated image compression, nearest neighbor search (*e.g.* [BSFG09]) or video denoising.

This puts forward the fact that motion indeed intervenes, leading us to the second aspect of the problem. We propose a complete framework for occlusion map estimation which exploits dynamic idiosyncrasies of the scene of interest. Indeed, the plausibility of the reconstruction (loosely, how probable it is) *does not demand accuracy in motion estimation nor a hard decision on which is the optimal correspondence vector.*

5.4 Proposed occlusion detection

We start by defining a reconstruction-based criterion for independently detecting occluded pixels (Section 5.4.1), assuming the knowledge of a correspondence map. In Section 5.4.2, we then leverage this new local model within an image-wise formulation of occlusion detection, where instrumental correspondences are obtained from a collection of suitable parametric motion models. No accurate optical flow is thus required while searching the best binary occlusion labeling over the image.

5.4.1 A reconstruction-based criterion

Let us define two functions $\zeta(I)$ and $\eta(I';I,\mathbf{w})$ that provide two different *reconstructions* of the same image I, either from itself (intra-image reconstruction) or from

another related image I', and a correspondence field between the two, $\mathbf{w}$, being given (inter-image reconstruction under correspondence guidance). Given a pair (I_1, I_2) of successive video frames and some correspondence field $\mathbf{w}$ from the first image to the second one, we shall denote in short $\zeta_1 = \zeta(I_1)$ and $\eta_{1,2} = \eta(I_2; I_1, \mathbf{w})$.

In essence, $\eta_{1,2} = \{\eta_{1,2}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$ conveys the appearance of image I_1 that is retained by the second image. As such, under the true motion field, $\eta_{1,2}$ is expected to deviate towards the appearance of I_2 for all the occluded points. This particular behavior is clearly visible in Fig. 5.3, where segments with largest motions appear doubled in a stroboscopic-like effect. On the other hand, ζ_1 captures the intrinsic appearance of I_1 revisited from its own perspective for every $\mathbf{x}$ on the image grid Ω.

This means that, if the two functions are defined in a suitable way, one could deduce whether a pixel at location $\mathbf{x}$ in the first image of the pair is visible or not in the second one by comparing ζ_1 and $\eta_{1,2}$ around this location. Experiments showed that this comparison is better conducted in an asymmetric way whereby a local color model g (defined later in Eq. 5.6) is fitted to ζ_1 in the neighborhood of $\mathbf{x}$ and used to assess the likelihood of $\eta_{1,2}(\mathbf{x})$ under visibility hypothesis.

Reasoning independently at the pixel level for now, point $\mathbf{x}$ will be considered as not visible in the second image if:

$$-\ln g\big(\eta_{1,2}(\mathbf{x})\big) > \varepsilon_v, \tag{5.2}$$

where ε_v is a conveniently chosen threshold, and g an exponential density function.

The function ζ aims mostly at "simplifying" the first input image I_1 such that robust comparisons can be conducted later on. Yet, it is important to preserve the structure of the input image. A natural choice for this function is the classic bilateral filter [TM98]:[2]

$$\zeta_1(\mathbf{x}) = \frac{1}{Z(\mathbf{x}; I_1)} \sum_{\mathbf{y}\in N_\mathbf{x}} \alpha(\mathbf{x}, \mathbf{y}; I_1) I_1(\mathbf{y}), \tag{5.3}$$

where $N_\mathbf{x}$ is a square window centered at $\mathbf{x}$, $Z(\mathbf{x}; I_1) = \sum_{\mathbf{y}\in N_\mathbf{x}} \alpha(\mathbf{x}, \mathbf{y}; I_1)$ is a normalization factor and the weighting function α depends on both appearance and spatial proximity within pixel pair:

$$\alpha(\mathbf{x}, \mathbf{y}; I_1) = f_a(\big\| I_1(\mathbf{y}) - I_1(\mathbf{x}) \big\|) f_s(\big\| \mathbf{y} - \mathbf{x} \big\|), \tag{5.4}$$

with f_a and f_s being Gaussian kernels.

In a similar fashion, $\eta_{1,2}$ will form a structure-preserving reconstruction of the first image I_1 but, this time, using colors from the second image I_2 under the guidance of correspondence map $\mathbf{w}$:

$$\eta_{1,2}(\mathbf{x}) = \frac{1}{Z(\mathbf{x}; I_1)} \sum_{\mathbf{y}\in N_\mathbf{x}} \alpha(\mathbf{x}, \mathbf{y}; I_1) I_2\big(\mathbf{y} + \mathbf{w}(\mathbf{y})\big). \tag{5.5}$$

[2] Any other discontinuity-preserving image filter could be used, provided it possesses a guided version.

This can be seen as a “displaced cross-bilateral filter", that is, the cross-bilateral filtering [KCLU07] of a warped image. Note that, as previously stated, we make use of a correspondence map only as a tool to find a valid reconstruction of I_1.

It is important that the two reconstruction functions share the same filter weights.[3] This way, $\eta_{1,2}$ captures as much as possible the local structure of I_1 and both reconstructions are comparable pixel-wise. Intentionally, this structure-mimicking behavior is not favorable for reconstructing points that are visible only in the first image, *i.e.*, occluded points.

The next step in the procedure consists in assessing whether a point $\mathbf{x}$ is occluded or not based on the criterion defined in (5.2). In order to conduct this step in practice, we propose here to over-segment the first reconstructed image into homogeneous segments where meaningful local color models can be estimated. In our experiments, these homogeneous segments are SLIC superpixels [ASS+12], and a Gaussian Mixture Model (GMM) of color is extracted for each of them, defining the density g in (5.2).

Image ζ_1 is segmented into J super-pixels and the j-th one, $S_j \subset \Omega$, is equipped with the mixture:

$$g_j = \sum_{k=1}^{K_j} \pi_k^j \mathcal{G}(\mu_k^j, \Sigma_k^j), \tag{5.6}$$

where π_k^j, μ_k^j and Σ_k^j are respectively the weight, the mean and the covariance matrix of the k-th component of the mixture. These GMMs provide good local models of ζ_1 in the sense that one can assume that

$$\forall \mathbf{x} \in \Omega,\ \zeta_1(\mathbf{x}) \sim g_{s(\mathbf{x})}, \tag{5.7}$$

where $s(\mathbf{x}) \in [\![1, J]\!]$ is the index of super-pixel containing $\mathbf{x}$. This should also hold for $\eta_{1,2}(\mathbf{x})$ provided the point is not occluded in second image. The novel reconstruction-based test for occlusion detection at the pixel level (5.2) finally reads:

$$\mathbf{x} \text{ occluded if } -\ln g_{s(\mathbf{x})}\big(\eta_{1,2}(\mathbf{x})\big) > \varepsilon_v. \tag{5.8}$$

Contrary to DFD-based test (5.1), this one is not based on a point-to-point comparison but on a point-to-local-model one. We shall demonstrate experimentally that it is a more powerful alternative. Yet, as DFD-based test, it still assumes that a correspondence map is available to produce the temporal image reconstruction $\eta_{1,2} = \eta(I_2; I_1, \mathbf{w})$. Next, we explain how to use this novel modeling over the whole image without depending on a single, accurate motion field.

[3] In particular, we have the desirable property $\zeta(I) = \eta(I; I, \mathbf{0})$.

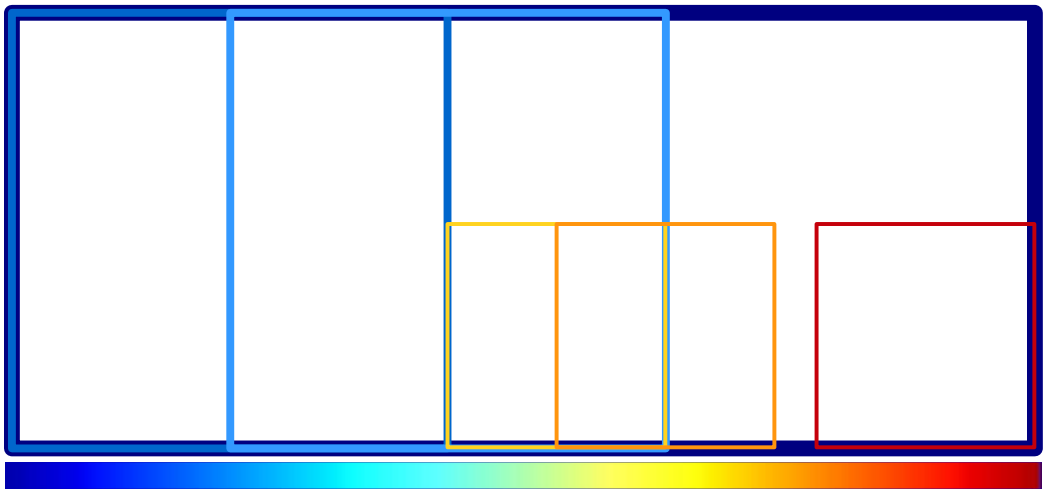


Figure 5.4: **Motion models supports**: Sample windows from the collection used to obtain supports for parametric motion models estimation. The different windows sizes and positions are encoded with the *jet* colormap, from bigger to smaller sizes, and from top-left to bottom-right.

5.4.2 From motion models to occlusions without stopping by optical flow

We propose a method for detecting occlusions which uses the image-reconstruction reasoning explained above. In order to be agnostic to optical flow computation, we propose to rely on a collection of *motion models* that spans the various dynamics of the scene, and thus enables plausible image reconstructions. As classically exploited in video segmentation and analysis, the apparent motion at work in natural dynamic scenes can often be decomposed into a set of low-complexity models, typically region-wise affine models. Such a paradigm recently proved useful also to estimate dense optical flows [YL15]. In our case, such models will provide candidate correspondences at each pixel, leading to a discrete labeling problem intertwined with the main one of occlusion detection.

Building a set of plausible motion candidates We start by computing a set $\mathcal{W} = \{\mathbf{w}_k,\ k = 1 \cdots K\}$ of K parametric motion models that are relevant to different sub-regions of the scene. We extract a large number K of overlapping windows of different sizes, starting with a window encompassing the full image support, and subsequently reducing the size by a half and changing position of the windows with a fixed overlap factor of 50%, covering the whole image for every window size, as illustrated in Figure 5.4. For the image size of the Sintel dataset, with four levels of window sizes we obtain, for instance, $K = 115$ windows.

For each window, we robustly estimate a parametric warp that captures at best the motion of corresponding scene fragment. Several classic techniques can be used to this end. In our experiments, we combine semi-dense matching, to handle large displacements, with robust affine motion estimation. We first extract point matches between images I_1 and I_2 with *DeepMatching*[4] [RWHS15a] and fit an initial affine motion to the matches originated from the window of interest. These models are then refined with *Motion2D*[5], an M-estimator relying on all support intensities [OB95].

This multi-window motion estimation approach arguably provides a partially redundant collection of motion models, but it is simple, fast, and it circumvents in particular the intricate problem of motion segmentation. From the set $\mathscr{W}$ of K parametric motion models thus obtained, we want to exploit the most plausible at each pixel to achieve occlusion detection. Observe that with this procedure, pixel-wise occlusion modeling is not tied to a single, accurate, dense optical flow, but rather to a region-wise characterization of the scene dynamics.

Alternating between occlusions and motion models The task to solve is now the one of jointly selecting a motion model and deciding on visibility for each pixel. We pose it as an energy minimization problem with respect to a motion model labeling $M = \{m(\mathbf{x})\}_{\mathbf{x}\in\Omega} \in [\![1,K]\!]^{\Omega}$ and an occlusion map $O = \{o(\mathbf{x})\}_{\mathbf{x}\in\Omega} \in \{0,1\}^{\Omega}$, where assigning 1 to a point means it is occluded, and 0 means it is visible. For pixel location $\mathbf{x}$, label $m(\mathbf{x})$ indicates which one of the available parametric motion models is relevant, while $o(\mathbf{x})$ establishes if there is an occlusion or not. For $m(\mathbf{x}) = k$, the associated inter-image reconstruction (5.5) is denoted $\eta^k_{1,2}(\mathbf{x})$. The joint energy to minimize is defined as:

$$E(O,M) = \sum_{\mathbf{x}\in\Omega} \phi_{\mathbf{x}}(o(\mathbf{x}), m(\mathbf{x})) + \mathrm{DL}(M) + \sum_{\mathbf{x}\sim\mathbf{y}} \left(\psi^m_{\mathbf{x},\mathbf{y}}(m(\mathbf{x}), m(\mathbf{y})) + \psi^o_{\mathbf{x},\mathbf{y}}(o(\mathbf{x}), o(\mathbf{y}))\right) \tag{5.9}$$

where the second sum is taken over all pairs of neighboring pixels. The unary potential reads

$$\phi_{\mathbf{x}}(0,k) = -\ln g_{s(\mathbf{x})}(\eta^k_{1,2}(\mathbf{x})),\ \phi_{\mathbf{x}}(1,k) = \alpha_v, \tag{5.10}$$

where $\alpha_v > 0$ is the cost of labeling a single pixel as occluded. It is related to ε_v in pixel-wise test (5.8). This data term is thus not based on point-wise displaced frame differences as classically done, but on reconstruction-based local modeling. Although this modeling effectively penalizes motions that do not preserve color information up to the precision of the local model, this data-term, as seen from the unknown motion point-of-view, would not be suitable to yield an accurate pixel-wise motion

[4] `http://lear.inrialpes.fr/src/deepmatching/`
[5] `http://www.irisa.fr/vista/Motion2D/`

estimation. Again, this is not the intention anyway, the sole aim being to reason locally on as plausible as possible inter-image reconstructions. This is further analyzed in Section 5.5.

Since each motion label k corresponds to a specific image window, one could restrict the labeling of pixel $\mathbf{x}$ according to the windows it belongs to. We found, however, that this restriction can cause block-like artifacts in the label assignment, with subsequent damage to final occlusion labeling. To capture motion model locality in a less drastic way, we propose instead to double $\phi_{\mathbf{x}}(0,k)$ (unary potential for visible points) for motion models k stemming from windows $\mathbf{x}$ does not belong to.

The binary potentials share a similar form of contrast-sensitive smoothing:

$$\psi^{a}_{\mathbf{x},\mathbf{y}}(k,k') = \lambda_a \exp\left(-\beta_a \left\| I_1(\mathbf{x}) - I_1(\mathbf{y}) \right\|\right) [k \neq k'] \tag{5.11}$$

with $a \in \{\text{“}o\text{”}, \text{“}m\text{”}\}$, where $[\cdot]$ is Iverson bracket and λ_o, λ_m, β_m, β_o are positive parameters.

Finally, the global motion label cost $\mathrm{DL}(M)$ penalizes the complexity of the labeling through its “description length”, i.e., the number of motion labels effectively used:

$$\mathrm{DL}(M) = \lambda_c \sharp\{k : \exists \mathbf{x} \in \Omega, m(\mathbf{x}) = k\}. \tag{5.12}$$

Optimization of the objective function The occlusion map O and, as a by-product, the motion-model label map M, are obtained by minimizing $E(O,M)$ (Eq. 5.9) with a block coordinate descent in an alternate way. Given occlusion assignment O, minimizing $E(O,M)$ w.r.t. M only amounts to minimizing

$$\sum_{\mathbf{x}\in\Omega} \phi_{\mathbf{x}}(o(\mathbf{x}), m(\mathbf{x})) + \sum_{\mathbf{x}\sim\mathbf{y}} \psi^{m}_{\mathbf{x},\mathbf{y}}(m(\mathbf{x}), m(\mathbf{y})) + \mathrm{DL}(M). \tag{5.13}$$

This can be done approximately by using α-expansions with the method of [DOIB12] in order to handle the global label cost term. Subsequently, for a given motion model label map, the occlusion map can be recovered by minimizing w.r.t. O the following function:

$$\sum_{\mathbf{x}\in\Omega} \phi_{\mathbf{x}}(o(\mathbf{x}), m(\mathbf{x})) + \sum_{\mathbf{x}\sim\mathbf{y}} \psi^{o}_{\mathbf{x},\mathbf{y}}(o(\mathbf{x}), o(\mathbf{y})), \tag{5.14}$$

with graph-cuts [BVZ01]. The occlusion and label maps are alternatively updated for a small number of iterations. Recall that this process is not oriented at optical flow recovery, but at selecting plausible motion models.

5.5 Experimental results

For the quantitative evaluations reported in this section, we rely on the MPI Sintel dataset[6] [BWSB12a]. This dataset comprises 69 sequences from the open-source CGI

[6] `http://sintel.is.tue.mpg.de/downloads`

movie Sintel[7] for which ground-truth optical flows and occlusion maps have been computed from the known 3D dynamic structure of the scenes.

5.5.1 Evaluation of the occlusion criterion

Let us first demonstrate the value of the reconstruction-based criterion introduced in Section 5.4.1 by performing pixel-wise occlusion detection under the guidance of three different correspondence fields: (1) The true optical flow, as accessible in MPI Sintel sequences; (2) The true flow contaminated by independent additive Gaussian noise of standard deviation 2.5; (3) The flow estimated with *DeepFlow*[8] [WRHS13], a state-of-art optic flow estimator which does not handle occlusions but solves for long displacements. For all the experiments, we fixed the number of super-pixels J to 700. We found experimentally that having only 2 components for each *GMM* provides good results in general. Furthermore, the standard deviation of the Gaussian kernels was set to 1.0, with a window size of 5 pixels. These parameters are related to the local color model variability within the superpixel supports.

Comparisons are conducted against DFD-based detection, which is at the heart of approaches based on analyzing violations of the brightness constancy assumption [ARS12, XCS+06]. For eight sequences of the dataset, we report in Fig. 5.5 occlusion detection ROC curves (true-positive-rate (TPR) vs. false-positive-rate (FPR)) and area under the ROC curves (AUC), both varying the threshold ε_c in the rule (5.1) and varying ε_v in our own framework (5.8). It can be seen that our reconstruction-based criterion has more discriminative power than classic DFD criterion in the prospect of occlusion detection. In all regimes and on all the sequences the former outperforms the latter, often by a substantial margin. More interestingly, for most of the sequences, *i.e.*, "alley_1", "alley_2", "ambush_4", "ambush_5", "ambush_7" and "bamboo_1", the decrease in occlusion detection performance due to flow inaccuracies (with noisy ground-truth or *DeepFlow* estimate) is less significant for our method. Remarkably, when used together with *DeepFlow*, our criterion outperforms on several sequences the DFD criterion based on true motion field.

5.5.2 Full system evaluation

We now turn to the complete minimization-based method introduced in Section 5.4.2. We compare it on MPI Sintel sequences to several recent approaches: (1) a learning-based occlusion detection algorithm, which uses a non trivial ensemble of hand-designed features [HMAB11], including forward-backward flow inconsistencies for several optical flow methods; (2) a method based on layer assignment and depth

[7] `https://durian.blender.org/download/`
[8] `http://lear.inrialpes.fr/src/deepflow/`

ordering [SSB10]; (3) a method that leverages reasoning on local layers relationships [SLP14]; (4) a sparse occlusion detection method that relies on departures from the optical flow color constancy assumption [ARS12]. The MPI Sintel dataset, as previously mentioned, is a synthetic dataset generated for evaluation of optical flow. It contains 69 sequences with available ground-truth optical flow and occlusions. The 69 sequences correspond to three different rendering passes on 23 computer-generated videos, namely, "albedo", "clean", and "final". The first rendering pass, as its names indicates, is pure unshaded texture information. The second pass, "clean" adds some realistic effects like specular reflections and smooth shading. The "final" pass is even more complex, adding motion blur, camera-depth of field, and atmospheric effects.

In the experiments, we limit the number of iterations of our alternate minimization method to 2, as it converges quickly. Furthermore, we set $\lambda_o = 20.0$, $\lambda_m = 33.3$, $\beta_o = 0.1$, $\beta_m = 0.2$, and $\lambda_c = 10^3$ by hand-tuning. Setting critical parameter α_v is discussed below.

Table 5.1 summarizes results by the average F-score computed over all 69 ground-truth MPI Sintel sequences, using two ways of setting the main occlusion parameter for each method, *e.g.*, α_v for our method. In a first set of experiments ("Global 69" column), it is manually set at once for all sequences. For [SLP14] and [HMAB11], the corresponding parameters are set to 0.5, as reported by [SLP14]; for our method α_v is set to 10; finally, for [ARS12][9] and [SSB10][10] we set the parameters to the values proposed by the respective authors. In a second round ("Oracle 69" column), we tuned the key parameter of each method so as to maximize the F-scores on the whole dataset. As it can be appreciated, our reconstruction-based method outperforms all the other reported methods in both settings. We also present some results (as available) by tuning the parameters to maximize the F-scores only in the 23 sequences of the final rendering pass of the Sintel dataset ("Oracle Final" column). Finally, we show in Fig. 5.8 additional comparative results on real-world image sequences. Qualitative assessment of these results further confirm the merit of our approach: more accurate occlusion maps are produced compared to [ARS12] and [SSB10].

We provide in Figs. 5.6 and 5.7 several samples of our results on the Sintel dataset for visual inspection. Despite the complexity of some of these scenes, occlusion maps of good quality are obtained. In particular, extended occlusions produced by large displacements are very well captured, while retaining some details of smaller occluded regions. It is also interesting to examine associated motion model labelings $M = \{m(\mathbf{x})\}_{\mathbf{x}\in\Omega}$ and associated motion flows $\{\mathbf{w}^{m(\mathbf{x})}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$ (Fig. 5.6 e-f). While motion labels define segments that relate well to actual moving regions (*e.g.* the arm of the woman in the fourth sequence or the whole person in the first one), the flows are not very accurate for all the sequences. This is not surprising since source motion

[9] `http://vision.ucla.edu/~ayvaci/spaocc.html`

[10] We thank authors for providing us with their source code.

Table 5.1: **Quantitative comparisons on MPI Sintel dataset**. For each method, the average F-score (the higher, the better) is computed with two different ways of setting the main detection parameter over all the available training sequences and rendering passes. We also present results by tuning the main detection parameter on the Final rendering pass only (last column).

Detection method	Oracle 69	Global 69	Oracle Final
Learning [HMAB11]	0.535	0.448	-
Depth order [SSB10]	0.465	0.449	0.398
Local layers [SLP14]	0.474	0.376	-
Sparse method[ARS12]	0.310	0.259	0.258
Ours	**0.550**	**0.540**	**0.491**

models have been estimated beforehand over arbitrary image windows. Motion model selection with no further processing cannot produce accurate optical flows. This reveals, as already highlighted in Section 5.4.2, that the distinctive nature of our framework is to focus on good reconstruction-based modeling of occlusion, with inference requiring only plausible correspondences, not accurate per-pixel optical flow.

Regarding execution times, for a sample image pair of size 436×1024, our full method takes about 1212 s (29 s for the multi-window motion estimation and 1183 s for the energy minimization) on an Intel i7-3540M CPU @ 3.00GHz. In comparison, [ARS12] takes 1601 s, while [SSB10] takes 2201 s on the same machine.

5.5.3 Failure modes

The occlusion detection method that is proposed in this manuscript is relatively reliable for estimating occlusions under a few basic assumptions. Firstly, the plausible motion models do not need to provide an accurate per-pixel optical flow vector, but the available motions should at least be relevant to the scene. Secondly, we assume that strong illumination changes in between frames do not occur. Indeed, illuminations changes affect both the parametric model estimation and our occlusion detection model. Violating one or both of these assumptions would lead to failures of our method. The local color modeling, in particular, carry a few problems that somewhat limit the performance of our method. For example, GMMs easily overfit (very narrow modes on the estimated color distributions) in textureless regions, causing false positives even under slight temporal illumination variations. This all sums up to the final scores we presented in Table 5.1 for the challenging Sintel dataset. Even though our method provides encouraging state-of-the-art results, slightly above 50% for F-scores, there is still room for improvement and further research.

5.6 Concluding remarks

We have introduced a new approach to detect occlusions that occur from one frame to another in a video sequence. Departing from classic approaches that tightly link this task to the accurate computation of optical flow, we propose a local spatio-temporal reconstruction model that only requires the knowledge of a plausible motion. Given a collection of parametric motion models simply extracted from the scene at hand, we show how this local reconstruction model can be harnessed within a global energy function to deliver high quality occlusion maps.

Quantitative experiments yielded two important findings: (1) Proposed reconstruction based modeling provides a detection criterion at the pixel level that is a powerful alternative to classic criterion based on intensity inconsistency along the flow. Even when exploiting the true flow, latter criteria perform less well that ours with the output of an off-the-shelf flow estimator. (2) Our complete energy-based framework consistently outperforms state-of-the-art approaches on MPI Sintel dataset. More generally, we qualitatively observed the ability of our framework to produce good quality occlusion detection maps, even in scenes comprising large occluded regions and complex motions.

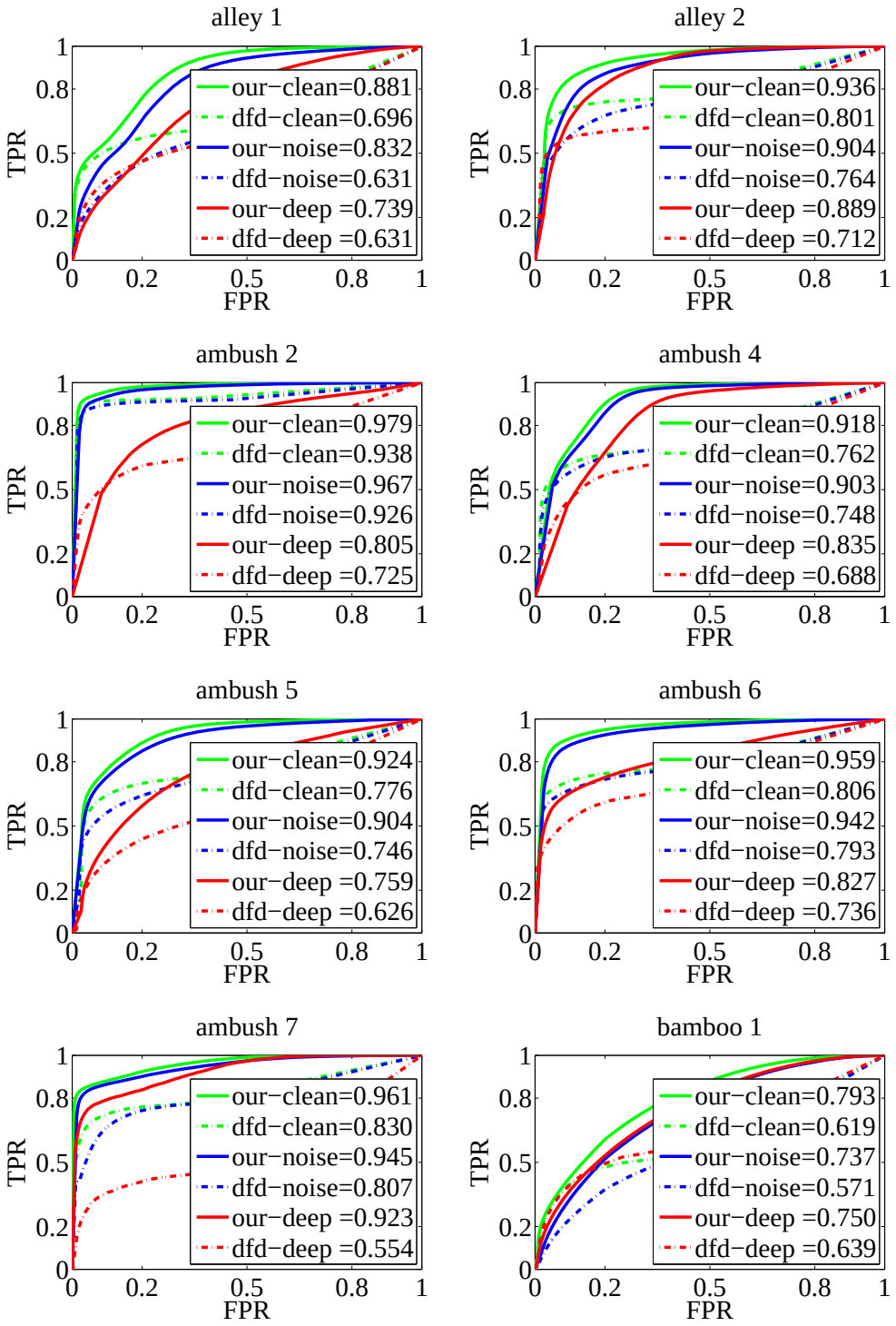


Figure 5.5: **Quantitative comparison of proposed reconstruction-based criterion and classic DFD-based criterion**. Occlusion detection ROC curves and associated AUC on eight sequences of the *MPI Sintel* dataset, obtained by varying the threshold of proposed criterion (solid lines) and of DFD criterion (dashed lines). Colors indicate the origin of the motion field: true optical flow (green), true optical flow contaminated by Gaussian noise (blue), *DeepFlow* estimate (red).

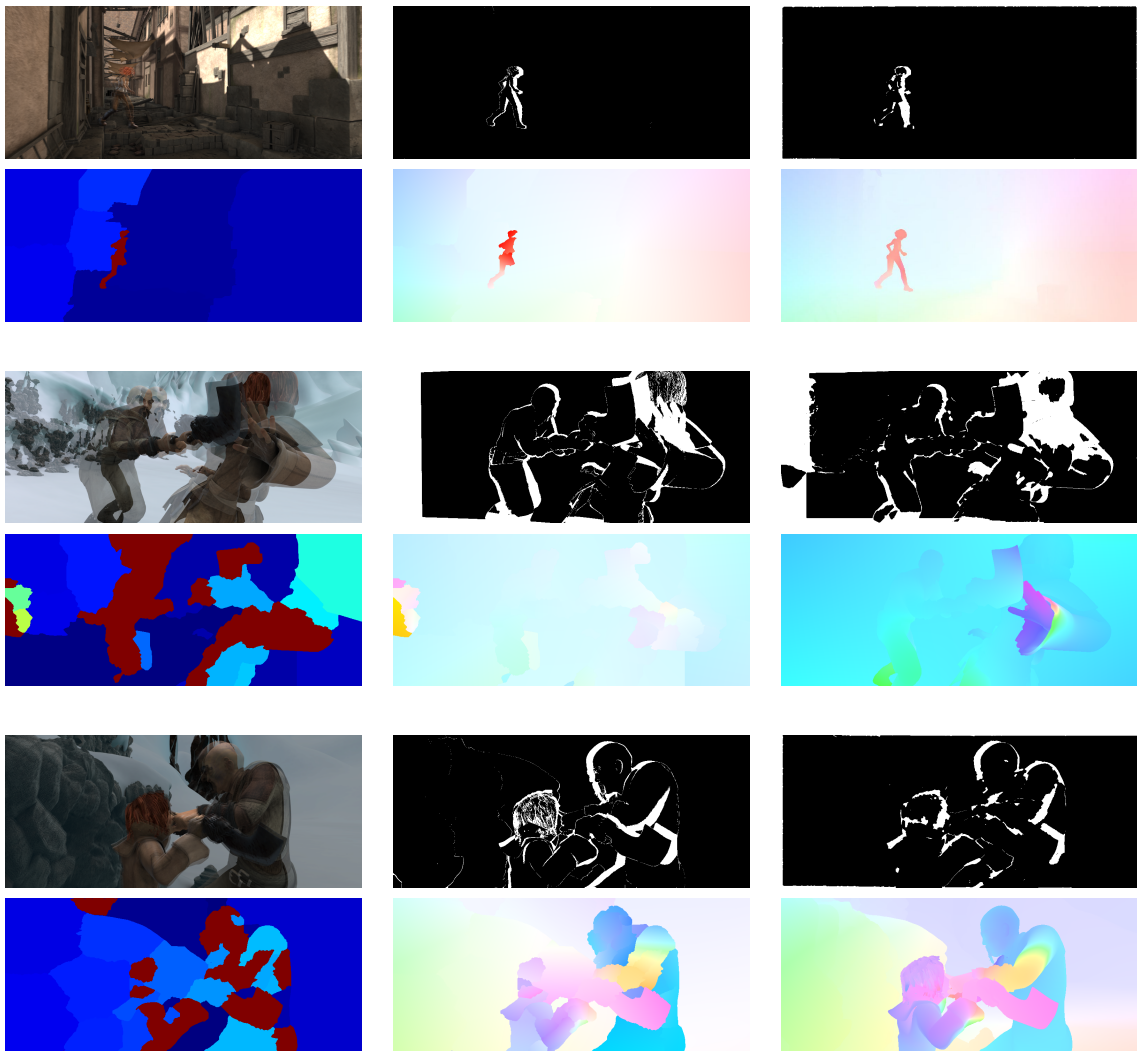


Figure 5.6: **Qualitative results of our occlusion detection method on several MPI Sintel scenes**. From left to right: First row for each sequence: Average of the two input frames; Occlusion ground-truth; Occlusion map with proposed method; Second row for each sequence: Final motion model labeling, where each color loosely represents the size and position of the window linked to the selected motion model through a jet-map colorization (Big to small and from upper-left to bottom-right); Optical flow obtained by evaluating the selected motion model at each pixel (with classic hue-saturation color coding); And true optical flow with same color coding.

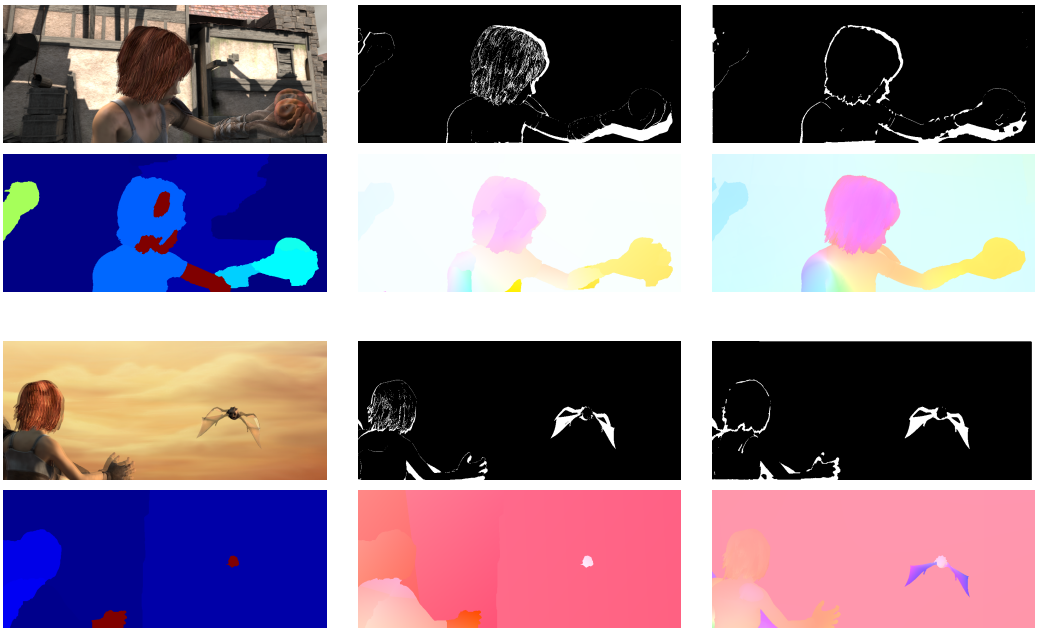


Figure 5.7: **Qualitative results of our occlusion detection method on other MPI Sintel scenes**. The images are shown in the same order as in Fig.5.6.

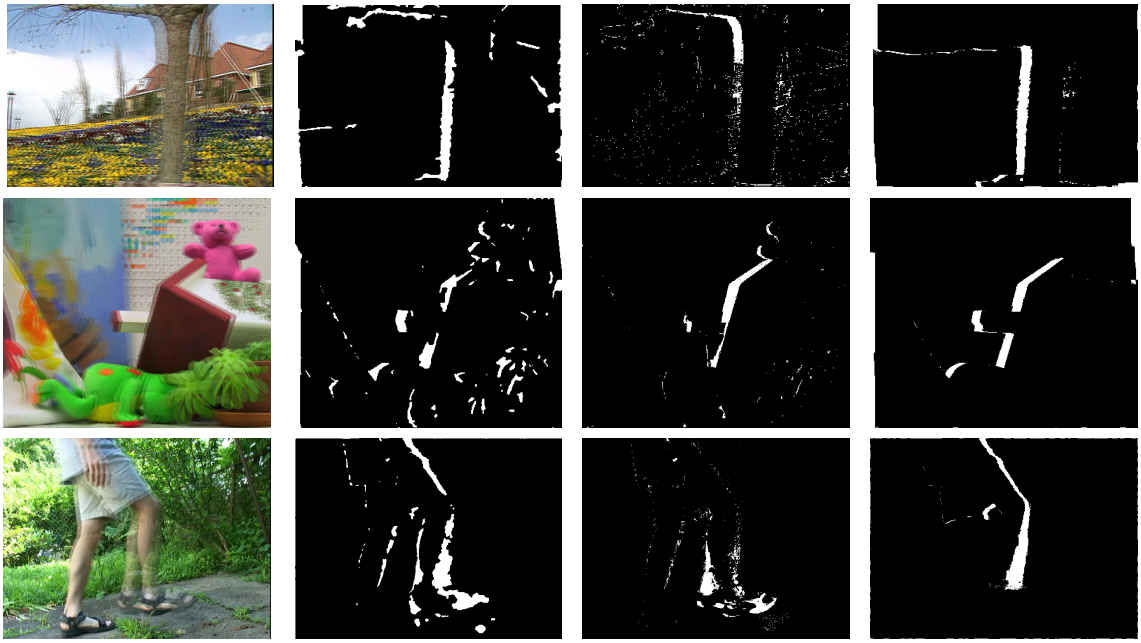


Figure 5.8: **Qualitative comparison with state-of-the-art on real images**. From left to right: Average of the two input frames, final occlusion map with proposed method, results of [ARS12], and of [SSB10].

CHAPTER 6

HIERARCHICAL MOTION ANALYSIS

The dynamic content of physical scenes is largely compositional, that is, the movements of the objects and of their parts are hierarchically organized and relate through composition along this hierarchy. This structure also prevails in the apparent 2D motion that a video captures. Since the visual motion of all scene components is, in addition, influenced by the motion of the camera, there is another level to this compositional hierarchy that turns it into a rooted tree. This idea is illustrated in Fig. 6.1, where a video sequence of a walking bear and its corresponding compositional motion tree are shown. Each one of the nodes of the motion tree corresponds with an actual object or object part in the scene.

Accessing this visual motion hierarchy is important to get a better understanding of dynamic scenes and is useful for video manipulation and video segmentation. In the present chapter we present methods to tackle this problem.

6.1 Introduction

Hierarchical motion decomposition of a scene is a process that can be useful for a number of applications in video processing. In particular, object segmentation in videos is a generic problem with far reaching applications. As such, it has received a lot of attention in the computer vision literature where both fully automatic and user-assisted pipelines are proposed. Automatic segmentation tools provide key building blocks for solving higher level computer vision problems like action localization, *e.g.*, [JVGJ+14]. On the other hand, interactive tools for video object segmentation are at the heart of complex video editing tasks such as cutout and rotoscoping, *e.g.*,

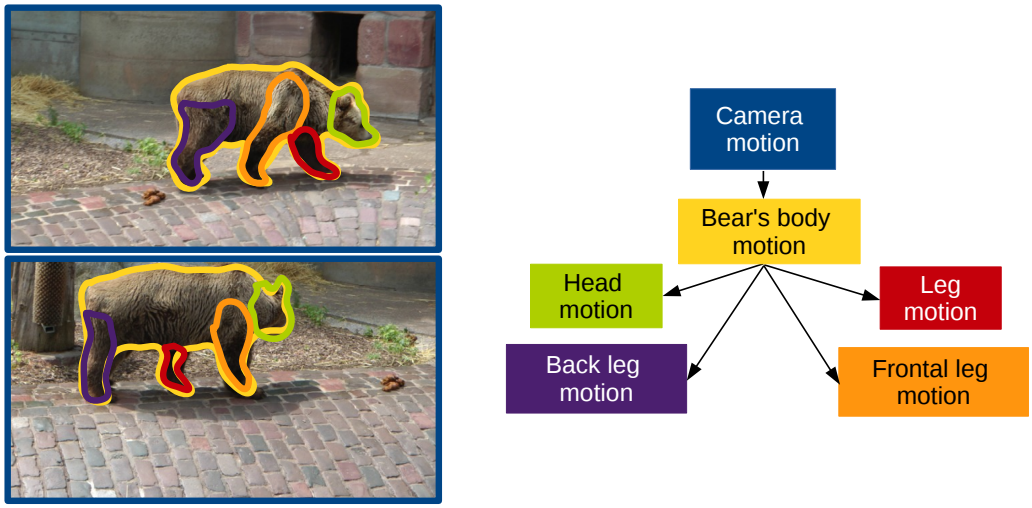


Figure 6.1: Hierarchical organization of visual motions in a natural scene.

[AHSS04, BWSS09], and can be used to ease the arduous tasks of video annotation [SNSB15].

Particularly for rotoscoping in video post-processing for the film industry, it is often required to delineate by hand several independent moving objects and their moving parts across video shots. However, current computer-aided solutions only focus on single object instances [BWSS09, LKG11], and usually do not take into account mid-level motion dynamics of scenes, like composition, for identifying the moving parts. Object instances for rotoscoping are annotated by an artist for a set of keyframes. The problem of finding which are the relevant moving objects that are to be subject to the rotoscoping process depends on the user expertise and the specific task at hand. For example, for 2D to 3D conversion, a different depth layer is assigned to each one of the moving object parts. As previously mentioned in the introductory chapters of this manuscript, artists usually prefer to use keyframes where most or all of the objects of interest are visible. Evidently, understanding the hierarchical structure of a moving scene, as imagined in Fig. 6.1, is a step towards simplifying the post-processing pipeline for certain applications. Indeed, such knowledge could be used to replace human annotation of moving objects for rotoscoping.

In another context, one could think of new ways to propose spatio-temporal regions as candidates for action localization in video. For instance, Jain *et al.* [JVGJ+14, JvGJ+17] proposed to use an off-the-shelf segmentation method [XC12] to produce supervoxels that are later merged by criteria encompassing color, texture and motion information. The intermediate output of [JVGJ+14] is a tree-like set of spatio-temporal windows ("tubelets"), which is later used to feed bag-of-words classifiers to localize actions in video. However, hierarchical decomposition of video in the current state-

of-the-art [JVGJ+14, XC12, WTXC04, XXC12, GKHE10] does not usually manage to preserve the intermediate semantic characteristics of the scene that are useful to outline *actual* moving objects and moving parts. Hence, over-segmentation is a common outcome due to the complexities of motion and color information in real imagery. In the action localization context, this could mean noisy results and processing overhead.

The importance of acknowledging and interpreting the structured motion information of videos is also motivated by early works in biological vision research. In fact, it has been found that the visual human system decomposes objects into parts through the analysis of motion nesting [Joh73]. Johansson showed in particular that removing the motion of the main body from the image reveals the distinctive motion of its parts. In the same line, Gershman *et al.* [GTJ15] have recently proposed a computational model that can decompose dynamic sensory data into a hierarchy of components. It was shown experimentally in psychological vision that such system is important in several tasks like segmentation, detection, and recognition of people and actions. The hierarchical decomposition of visual motion information is thus clearly identified as a key step in complex biological vision systems.

6.2 Related work

In this section, we discuss the relevant literature on three key aspects of our work: the compositional hierarchical modeling of visual motion, the problem of representing and clustering point tracks from a video sequence and, finally, the learning of dictionaries under hierarchical structure constraints.

Hierarchical motion estimation The idea of analyzing visual motion through compositional hierarchies has a long history. It goes as far as early works on estimation of optical flow, where it appeared in the form of incremental coarse-to-fine estimation, either to speed-up computations in the spirit of multigrid methods [Enk88] or to facilitate non-convex minimization in presence of large motions [BA96, OB95]. Combining additively dense optic flow and piece-wise parametric motions as in [JBJ96] also exploits a shallow hierarchy. In these works, the final goal remains however the estimation of a single motion field, whether dense or parametric, at the pixel level. Closer to our goal, several motion-based image segmentation techniques exploit nested parametric models [OB98, CB99]. Here again, the structure is kept shallow, typically providing a flat motion-based segmentation that captures independent moving regions in front of a background with dominant motion. In any case, above mentioned works concern motion between two successive video frames, as opposed to the shot-level analysis that we conduct using point trajectories.

Representing and clustering point trajectories A number of recent approaches to trajectory grouping make use of a spectral embedding [BM10, OMB14, OB12, FAFM15]. Based on a suitable similarity measure between trajectories, a pairwise affinity matrix is built and used to produce a low-dimensional embedding for each trajectory (based on the bottom eigen-vectors of the associated graph Laplacian). Clustering is then conducted in this embedded space, *e.g.*, through k-means clustering in the case of spectral clustering. For this segmentation step, Brox and Malik [BM10] minimize instead a clustering cost that also enforces cluster separability and penalizes model complexity, Ochs and Brox [OB12] consider higher order relationships between trajectories, and Ochs et al. [OMB14] extend [BM10] within an MRF-based spatial prior. In one of the approaches that we will present shortly, we also proceed through encoding of trajectories followed by clustering. However, the encoding is through a special form of dictionary learning and sparse coding and the clustering through hierarchical K-means. As opposed to methods discussed above, we obtain a hierarchical partition of the tracks set. Also, we outperform [OMB14] on their FMBS-59 benchmark.

As an alternative to spectral embedding that requires pairwise similarities, the preliminary low-dimensional encoding of the point tracks can be obtained through low-rank factorization of the data matrix [VM04, VH04, RTVM08, EV09, CR09]. Non-negative matrix factorization (NMF) and semi-non-negative matrix factorization (SNMF) are for instance used in [CR09] and [MD12] respectively. Dictionary learning is also a form of data matrix factorization, but under sparsity rather than rank constraints. To our knowledge, it has not been used so far to encoding point trajectories. Exploiting the expression power offered by a large dictionary, we propose a simple way to enforce the desired tree-based structure into the learning and the encoding steps. It is not clear that other low-rank factorization methods are amenable to this structuring. While the SNMF-based approach of [MD12] for instance does capture low-level motion segments, it does not have an explicit mechanism to extract higher-level motion segments.

Once encoded through data factorization or any other method, trajectories can be clustered through off-the-shelf or specially designed means. As for spectral clustering, K-means clustering is a simple option but more sophisticated alternatives, such as multiple subspace learning [RTVM08, EV09] and J-linkage [FRP09], have also been investigated. As already said, we adopt top-down hierarchical K-means for it fits ideally our aim of unveiling the hierarchical nature of visual motion.

Structured dictionary learning and sparse coding Representing data as sparse codes over learned dictionaries is a very powerful paradigm to process or analyse collections of signals, including images and image patches within an image. Among the numerous tools that have been developed in this domain, several forms of struc-

tured sparsity have been explored in conjunction with dictionary learning [ZRY09, JMOB10, HZM11]. The tree-based structured sparsity introduced by Jenatton *et al.* [JMOB10] is particularly interesting for our approach. The atoms of the dictionary being attached to the nodes of a rooted tree, this approach imposes that an input signal must be encoded only with atoms that form a (small) rooted subtree. As we shall see in Section 6.5.1, our requirement is more drastic: only atoms forming a branch to the root can be jointly used to encode a given trajectory.

6.3 Hierarchical motion decomposition

Discovering the underlying motion composition tree for a given video shot is a complicated task. We have devised two distinct approaches as first attempts to solve this problem. The first one of them is based on an original Conditional Random Field (CRF) model that is solved in a per-frame basis. It leverages motion information through the analysis of optical flow fields inside a structured energy minimization framework that accounts for motion composition. The output of such a method is a per-frame hierarchical segmentation of the scene. The second approach that we present, directly tackles the whole video shot at once by working on pre-computed point trajectories. The method is based on a novel structured dictionary learning algorithm to discover moving object parts and their hierarchy. We start by introducing the first approach in the next section.

6.4 A hierarchical analysis of motion from pairs of frames

Our goal is to obtain a hierarchical motion partition of the image represented by a tree delivering a view-based understanding of the dynamics of the scene. We call it the *motion decomposition tree* $\mathscr{T}$. This tree is composed of nodes which are associated with parametric motion models. In order to facilitate its estimation, we start with a virtual compositional tree of candidate motion models attached to its nodes. We coin the latter the *proposal tree* $\mathscr{M}$, which is a superset of $\mathscr{T}$.

Given the color image pair (I_1, I_2), formed by two proximate frames of a video sequence, and a correspondence field $\mathbf{f}$ between the two, all defined on the image grid Ω, we are interested in recovering the tree $\mathscr{T}$ of motion models as a specific subtree of $\mathscr{M}$. The proposal tree $\mathscr{M}$ forms the search space of our problem. It contains a set of plausible compositional motion models likely to explain local evidence in different parts of the scene. We want to find the optimal decomposition tree $\mathscr{T}$ but also to associate a node to each pixel such that the visual motion at this pixel is approximated

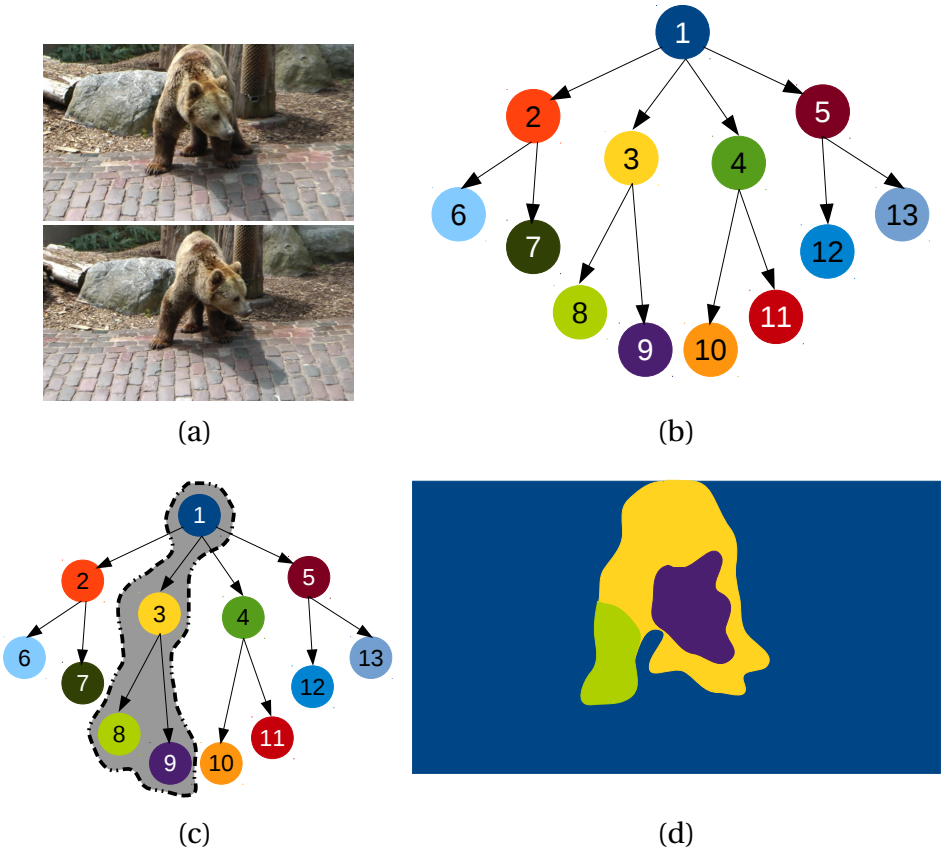


Figure 6.2: Illustration of hierarchical motion decomposition on an image pair from the FMBS dataset [BM10]. (a) Input pair. (b) Initial proposal tree $\mathscr{M}$, with $K = 13$. (c) Decomposition tree $\mathscr{T}$ is emphasized with a gray mask. It corresponds to the selected node-labels from the proposal tree that better describe the dynamic scene (d). The per-pixel node-label assignment can be interpreted as a mid-level hierarchical motion segmentation of the scene. The colors of the segmentation result correspond to the colors of the nodes in the initial tree.

by the composition of all the motions on the path from the root to that particular node.

Each node $k \in \{1,\dots,K\}$ of the proposal tree, numerated in level-order, is associated with a vector $\boldsymbol{\theta}_k$ of motion parameters defining a particular motion model. We may deal with different types of motion models depending on the nodes. Specifically, we will use a 8-parameter quadratic model for the root (0-th layer) and 6-parameter affine motion models for subsequent layers. For every pixel $\mathbf{x} \in \Omega$, we define an index vector:

$$\boldsymbol{\alpha}(\mathbf{x}) = (\alpha_1(\mathbf{x}), \cdots, \alpha_k(\mathbf{x}), \cdots, \alpha_K(\mathbf{x})), \tag{6.1}$$

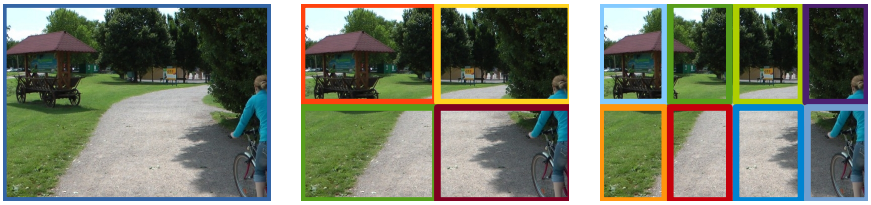


Figure 6.3: From left to right, windows are constructed from the root window by splitting it iteratively. For illustrative purposes, the proposal tree follows the structure and color code of Fig.6.2.

of K elements such that $\alpha_k(\mathbf{x}) = 1$ if the pixel motion conforms to the parametric motion model given by $\boldsymbol{\theta}_k$, and $\alpha_k(\mathbf{x}) = 0$ otherwise. Note that, $\alpha_k(\mathbf{x}) = 1$ implies that $\alpha_{k'}(\mathbf{x}) = 1$, for all nodes k' that are ancestors of k. Equivalently, if $\alpha_k(\mathbf{x}) = 0$, then $\alpha_{k'}(\mathbf{x}) = 0$ for all descendants of k. That is, the motion decomposition tree $\mathcal{T}$ is formed by the nodes of the larger proposal tree $\mathcal{M}$ for which there is at least one pixel $\mathbf{x}$ such that $\alpha_k(\mathbf{x}) = 1$.

We group the K parameter vectors $\boldsymbol{\theta}_k$ in a collection $M = \{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_K\}$. We also define the displacement function $\mathcal{F}_{\mathbf{x}}(M, \boldsymbol{\alpha}(\mathbf{x}))$ that results from composing the motion models indicated by $\boldsymbol{\alpha}(\mathbf{x})$ at $\mathbf{x}$:

$$\mathcal{F}_{\mathbf{x}}(M, \boldsymbol{\alpha}(\mathbf{x})) = \sum_{k=1}^{K} \mathbf{w}_{\boldsymbol{\theta}_k}(\mathbf{x})\alpha_k(\mathbf{x}), \tag{6.2}$$

where $\mathbf{w}_{\boldsymbol{\theta}_k}(\mathbf{x})$ is the motion vector determined by motion model $\boldsymbol{\theta}_k$ at pixel $\mathbf{x}$. We want compositional motion (6.2) at pixel $\mathbf{x}$ to agree with the reference correspondence field $\mathbf{f}(\mathbf{x})$.

An illustrative example can be found in Fig. 6.2. From the proposal tree $\mathcal{M}$ composed of nodes that describe distinctive motion models through the image pair, a subtree is selected to explain the motion of the different image parts. The background, for instance is clearly associated with the root node. The motion of the background is determined by the camera motion, however, the camera motion does not only affect this region, but the rest of the image as well. This is why, the body of the bear is associated to a child node of the root node. Thus, the motion model in this region is computed incrementally from its parent motion (Eq.6.2). Finally, one leg and the head of the bear exhibit movements of their own, which add up to the bear global motion. How these models are actually extracted, and how the label assignment is performed is explained hereafter.

6.4.1 Constructing proposal motion tree $\mathscr{M}$

This step consists in building the initial proposal tree, while estimating the parametric motion models associated to each of its nodes. We first perform a window sampling by hierarchically partitioning the image frame, as described in Fig. 6.3. It yields the desired tree structure, and each window will be the estimation support of the motion models attached to the corresponding nodes of the proposal tree.

The parametric motion models are computed robustly with the publicly available *Motion2D* software [OB95]. It is important to note here that the motion models describe per-region motion characteristics, in contrast to the instrumental motion field $\mathbf{f}$, which is an unstructured dense reference [1]. Reasoning on piecewise parametric motion models has been successfully used recently to solve intricate problems like optical flow [YL15] and occlusion detection [PRCBP16].

In order to compute the motion model corresponding to each window, the reference color image is compositely warped as we descend the tree. In this way, every motion model captures compositional aspects of the scene motion. The window sampling initialization procedure might seem somewhat arbitrary. Nevertheless, it provides an efficient means for the initial estimation of the motion models which will be further updated along with the determination of moving regions, as explained in the next section.

6.4.2 Estimating decomposition tree $\mathscr{T}$ and pixel labels

To recover the decomposition tree $\mathscr{T}$ for the image pair of interest, we formulate the problem as a per-pixel label selection interleaved with motion models estimation. The labels represent the set of nodes from the proposal tree $\mathscr{M}$ which have to be selected to explain globally the input correspondence field $\mathbf{f}$. We introduce the collection of assignment vectors $A = \{\boldsymbol{\alpha}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$, where $\boldsymbol{\alpha}(\mathbf{x})$ is given by Eq. 6.1. We want to estimate the elements of the collection of motion models M and the assignment vectors A by minimizing the following objective function:

$$E(M,A) = \sum_{\mathbf{x}\in\Omega} ||\mathbf{f}(\mathbf{x}) - \mathscr{F}_{\mathbf{x}}(M, \boldsymbol{\alpha}(\mathbf{x}))||_2^2 + \lambda \sum_{\{\mathbf{x},\mathbf{y}\}\subset\Omega} \mu(\boldsymbol{\alpha}(\mathbf{x}), \boldsymbol{\alpha}(\mathbf{y})) \exp\left(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\beta_a^2} - \frac{||I_1(\mathbf{x}) - I_1(\mathbf{y})||^2}{2\beta_b^2}\right), \quad (6.3)$$

where $\mathbf{f}$ is the reference flow field which is precomputed and used as input of our algorithm. We use *DeepFlow* [WRHS13] to compute it. The smoothness term in (6.3), where $\mu(\cdot,\cdot) = 1$ if the two input index vectors are different and 0 otherwise, is weighted by a positive parameter λ and by a joint spatial and color Gaussian kernel. The

[1] In this work, we rely on *DeepFlow* [WRHS13].

Algorithm 2 Motion decomposition tree estimation

procedure OPTIMIZATION(M, A)
  $M = \{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_K\}$ ▷ Compute proposal tree
  while `Not converged` **do** ▷ Or max. iterations
    $A \leftarrow$ Minimize (6.3) for a given M with message passing [KK11]
    for $k = 1 \cdots K$ **do**
      if $\alpha_k(\mathbf{x}) = 1$ `for some` $\mathbf{x} \in \Omega$ **then**
        $\boldsymbol{\theta}_k \leftarrow$ Update model with [OB95] for image warped with composed motion models of all ancestors of k.
      end if
    end for
  end while
  return (M, A)
end procedure

Figure 6.4: Color code used to show our hierarchical segmentation results for a 3-deep tree where nodes in 0-th, first and second layer have respectively 8, 4 and 2 children each. Each node corresponds to a different color of the HSV colormap.

space and color parameters in this kernel, β_a and β_b, are hand-tuned. This pairwise energy term is computed over all possible pixel pairs. That is, our CRF model is fully connected. We have found that full connectivity allows our model to outline more accurately moving objects and moving objects parts.

This energy is minimized by performing coordinate descent on the collections M and A, as presented in Alg. 2. On each iteration[2], A is updated by minimizing (6.3) with fixed M using an efficient message passing implementation based on the mean fields approximation and high dimensional filtering [KK11, KK13]. In turn, the parametric motion models M are re-estimated with *Motion2D* at each step with A fixed. Recall that the motion models are compositional. Specifically, Eq. 6.2 is applied at every iteration step on the updated elements of the collection M. The algorithms iterates until labels stop changing. Finally, the motion decomposition tree $\mathscr{T}$ is obtained by

[2]Recall that our algorithm is initialized by constructing a proposal tree $\mathscr{M}$ with each node linked to a rectangular region of the image (Fig. 6.2). Parametric motion models inside said regions are computed with [OB95] while warping the input image as the proposal tree is traversed from the top.

pruning from $\mathscr{M}$ all the nodes that are not selected by any of the elements of collection A.

6.4.3 Some visual results

We report experimental comparative results with two popular hierarchical segmentation methods, *i.e.*, a graph-based segmentation method (GBH) [GKHE10], and a streamed hierarchical segmentation method (SHS) [XXC12] (Figs. 6.5 and 6.6). We used the FMBS dataset [BM10], which provides sequences with a wide range of content and motion types. Before running our method, we apply iterative edge preserving filtering [TM98]. We tuned the parameters of the energy function (6.3) by hand and fixed them for all our experiments as follows. The proposal tree structure consists of one root node, eight nodes in the second layer, each of which subdivides in four nodes, each of which splits in two nodes ($K = 1+8+32+64 = 105$). The choice of these numbers was governed by our expectation of a maximum number of eight independent moving objects in the scene, and a limited subsequent decomposition into visible moving subparts, while bounding the size of the proposal tree. We set $\lambda = 10$, $\beta_a = 5$, and $\beta_b = 3$. For [XXC12] and [GKHE10] we use the parameters proposed by the respective authors.

For an easy comparison, we only display segmentation results at first level (root's children) and at the last level (leaves) of the hierarchy for all the presented methods. From Fig. 6.5, it can be observed that our method extracts insightful information from scenes which can contain a wide range of motion types. Furthermore, it is the only one to consistently provide segments that preserve some level of semantic meaning in terms of moving objects and moving parts, at both ends of the tree-based hierarchy. For instance, in Fig. 6.5(g) we can see that our method captures the main moving objects while also correctly labeling background pixels only with the root node. Moreover, our method is the one that suffers the least, by far, from over-segmentation as it is not driven only by local dissimilarity measures. One limitation of our method is its dependency to the reference optical flow method, which may be affected by adverse conditions as specular reflections (*i.e.*, second and last rows in Fig. 6.5).

6.4.4 About frame-based hierarchical scene decomposition

So far in this chapter, we have proposed a method that takes a pair of frames as input and delivers a tree-based model representing the hierarchical structure of the dynamic scene. A segment of the image corresponds to one of its nodes. We have coined such output as *decomposition tree* $\mathscr{T}$. For real pairs of images, our method is able to deliver a relevant partition of the image grid, with segments that somewhat correspond to objects or object parts. However, the proposed algorithm has limited applicability as presented, since its generalization to multiple frames seems not trivial. Obtaining

consistent trees for a longer video sequence with this first method might be difficult[3]. However, we have demonstrated qualitatively the value of using a motion composition based reasoning for segmentation related tasks. In the next section of this chapter we propose a method that is able to directly process a whole video shot, enabling our idea to apply in a wider variety of scenarios. Both of our methods rely on black-box motion estimation, pair-wise optical flow in one case, shot-wise point tracks in the other.

[3] A simple extension to longer video sequences by leveraging graph-matching of the output trees in a frame-by-frame basis would not deal with temporal consistency of the pixel labeling.

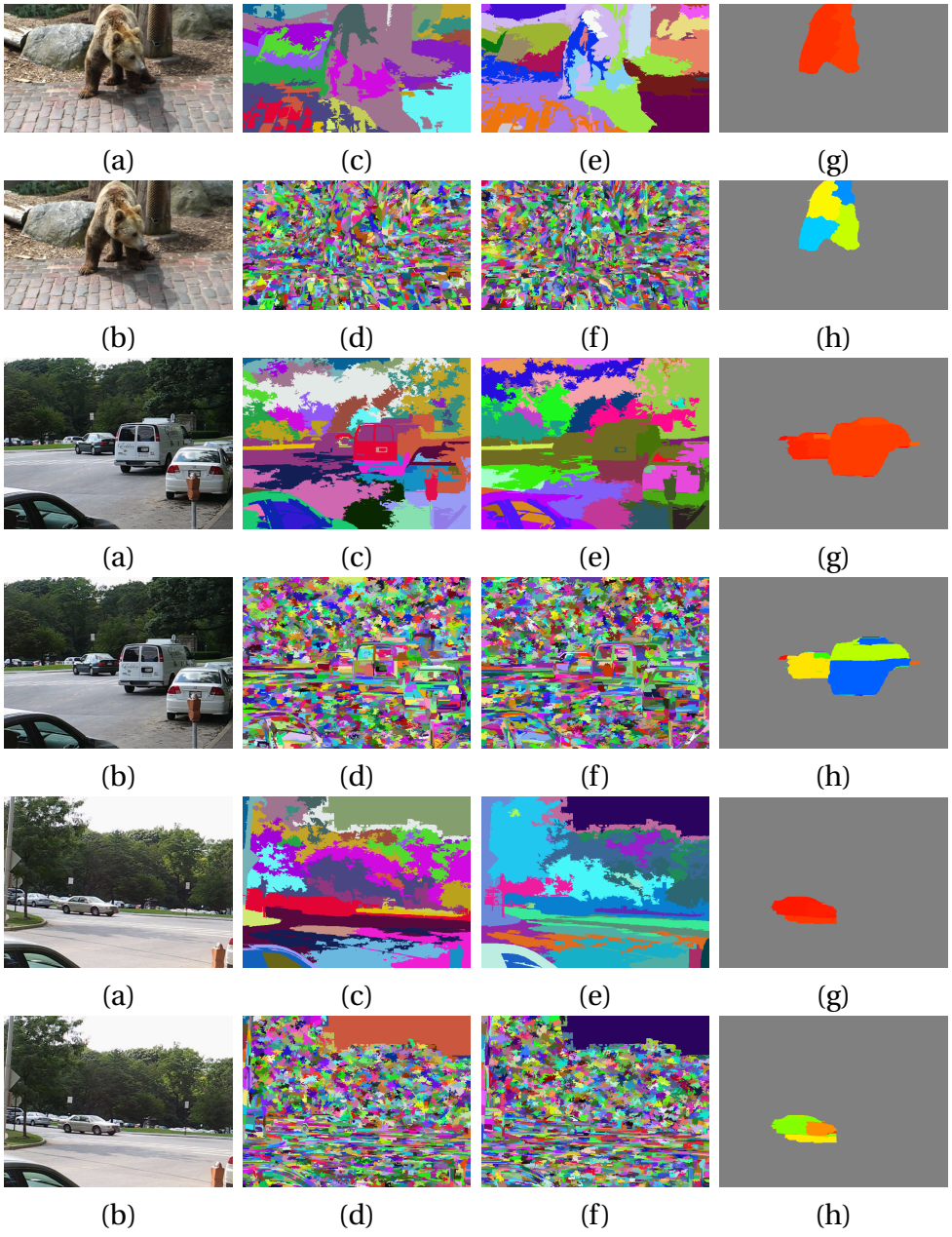


Figure 6.5: Comparative visual results of scene motion segmentation. (a-b) Input image pair. (c-d) Segmentation results at the second and 15th layer of GBH [GKHE10]. (e-f) Segmentation results at the second and 15th layer for SHS [XXC12]. (g-h) Segmentation results extracted by our motion decomposition algorithm for the second and 3rd layer. The color code used is indicated in Fig. 6.4.

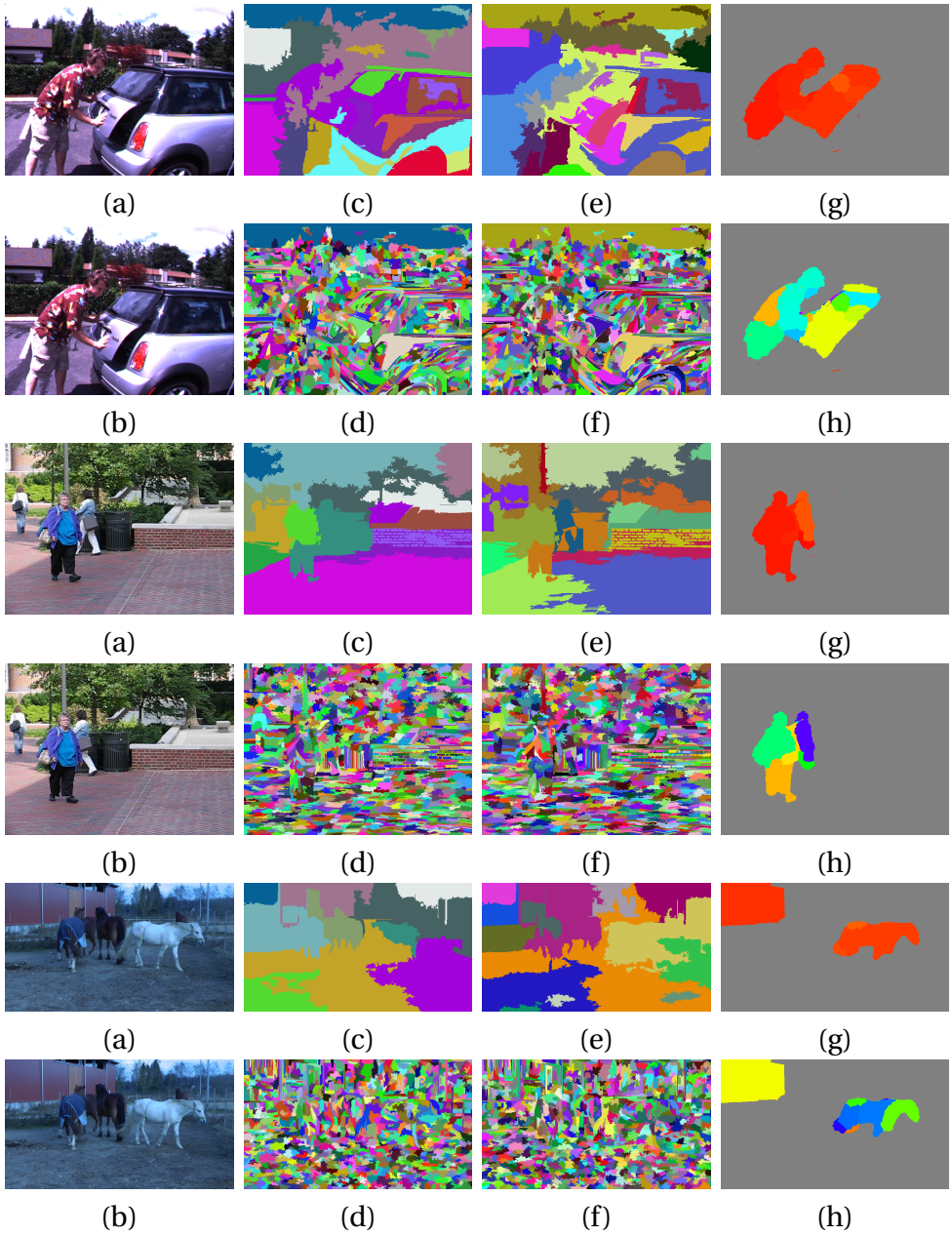


Figure 6.6: Comparative visual results of scene motion segmentation. (a-b) Input image pair. (c-d) Segmentation results at the second and 15th layer of GBH [GKHE10]. (e-f) Segmentation results at the second and 15th layer for SHS [XXC12]. (g-h) Segmentation results extracted by our motion decomposition algorithm for the second and 3rd layer. The color code used is indicated in Fig. 6.4.

6.5 A hierarchical partition by learning structured dictionaries

We start by introducing sparse coding of point trajectories. We then explain in Section 6.5.1 how such a learned representation can be hierarchically structured in order to capture the compositional nature of apparent motion. We extend this hierarchical sparse coding framework in Section 6.5.2, so that it facilitates unsupervised hierarchical clustering of the input data. Finally, we explain in Section 6.5.3 how trajectory clusters thus obtained can be turned into a final motion-based segmentation.

In this work, we exploit point trajectories extracted with the method of Sundaram *et al.* [SBK10b], which relies on forward/backward optical flows from [BBM09]. Although the results might be improved by using more recent optical flow methods, like the ones in [RWHS15b] or [FBK15a], we stick to [BBM09] for optical flow computation in order to perform a fair comparison with other methods, that is, only on the basis of the proposed representation and associated algorithms.

Given an input video sequence of $M+1$ frames and N input point trajectories extracted from it [4] $(\mathbf{x}^n_{0:M}) \in \mathbb{R}^{2\times(M+1)}$, $n = 1 \cdots N$, we define the data matrix $X \in \mathbb{R}^{2M\times N}$ as:

$$X = \begin{bmatrix} \Delta\mathbf{x}^1_1 & \Delta\mathbf{x}^2_1 & \cdots & \Delta\mathbf{x}^N_1 \\ \Delta\mathbf{x}^1_2 & \Delta\mathbf{x}^2_2 & \cdots & \Delta\mathbf{x}^N_2 \\ \vdots & \vdots & \cdots & \vdots \\ \Delta\mathbf{x}^1_M & \Delta\mathbf{x}^2_M & \cdots & \Delta\mathbf{x}^N_M \end{bmatrix} \tag{6.4}$$

where $\Delta\mathbf{x}^n_m = \mathbf{x}^n_m - \mathbf{x}^n_{m-1}$. In this matrix, each column stems from the sequence of displacements along one trajectory. A powerful way to discover multiple structures in such data is through sparse coding with a learned dictionary. Formally, one can seek an approximate decomposition $X \approx DA$ into a *dictionary* matrix $D = [\mathbf{d}_1 \cdots \mathbf{d}_K] \in \mathbb{R}^{2M\times K}$, possibly with K larger than $2M$ (overcomplete dictionary), and a *sparse representation* $A = [\boldsymbol{\alpha}_1 \cdots \boldsymbol{\alpha}_N] \in \mathbb{R}^{K\times N}$ of the input data. The K columns of the matrix D are unit-norm basis elements termed *atoms*, and those of A are the sparse codes associated to the N input trajectories. Such a sparse decomposition can be achieved by solving the optimization problem:

$$\underset{D,A}{\arg\min} \|X - DA\|_2^2 \quad \text{sb.t. } \|\boldsymbol{\alpha}_n\|_0 \leq s,\ \forall n \quad \text{and} \quad \|\mathbf{d}_k\|_2 = 1,\ \forall k \tag{6.5}$$

using, for example, the K-SVD algorithm [AEB06]. Positive parameter s controls the sparsity constraint and $\|X - DA\|_2$ is the reconstruction error of the trajectory

[4] We assume for simplicity that all trajectories are defined over the full temporal extent of the video sequence. Taking into account trajectories with different life-spans is readily done by introducing an appropriate masking matrix $P \in \{0,1\}^{2M\times N}$ in following derivations.

displacements. At each iteration of the K-SVD algorithm, a coding step is performed (*i.e.*, solving for A in Eq. 6.5, dictionary D being fixed) with the orthogonal matching pursuit (OMP)[TG07], followed by an SVD-based update of the dictionary's atoms. The previous formulation, however, does not enforce any structure on the elements (atoms) of the dictionary. Next, we re-formulate the problem so that the dictionary and the associated encoding follow a tree structure.

6.5.1 Hierarchical dictionary learning

As we have established already, the natural organization of the moving objects and their parts in a video is that of a tree, as illustrated in Fig. 6.2. In this simple example, the motion of one leg of the bear constitutes one node of the tree with the motion of the animal body being at its parent node, the latter being related to the root note that captures the visual motion induced in the whole scene by the movement of the camera. In fact, this hierarchical organization of visual motion is not restricted to articulated objects, but to other natural scenes as well. We aim at leveraging such a structure.

A second key feature of our approach is to represent point tracks as linear combinations of few learned atoms. We thus resort to dictionary learning and sparse coding techniques, with the goal of organizing the dictionary in a hierarchical structure that can capture, to some extent, the compositional organization of the dynamic scene. Each trajectory pattern in the dictionary is associated to a node of a tree and should ideally capture the motion of one scene element, *relative to its ancestors in the tree*. In other words, we want the movement of a given scene element to be represented *only with dictionary atoms stemming from a same branch of the tree*. This form of hierarchical tree-structured sparsity is related to the one of Jenatton *et al.* [JMOB10], but is more drastic. Jenatton *et al.* indeed stipulate that only a subtree from the root can be used to represent an input signal.

For a given rooted tree $\mathcal{T}$ of K nodes numbered in level-order,[5] we want to learn a dictionary $D = [\mathbf{d}_{1:K}] \in \mathbb{R}^{2M\times K}$ of K trajectory atoms organized according to this tree structure, together with the corresponding matrix $A = [\boldsymbol{\alpha}_{1:N}] \in \mathbb{R}^{K\times N}$ of sparse codes. To this end, we consider the following constrained minimization problem:

$$\arg\min_{D,A} \|X - DA\|_2^2, \quad \text{sb.t. } \boldsymbol{\alpha}_n \in \mathscr{A}(\mathcal{T}),\ \forall n \quad \text{and} \quad \|\mathbf{d}_k\|_2 = 1,\ \forall k, \tag{6.6}$$

where $\mathscr{A}(\mathcal{T}) \subset \mathbb{R}^K$ is the set of tree-structured codes defined as:

$$\mathscr{A}(\mathcal{T}) = \{\boldsymbol{\alpha} \in \mathbb{R}^K : \mathrm{supp}(\boldsymbol{\alpha}) \subset \mathrm{anc}(\mathrm{k}(\boldsymbol{\alpha}))\}, \tag{6.7}$$

[5] In practice, a simple regular structure is chosen, defined by its depth L, and by the common number n_ℓ of children for all nodes at a given depth level $\ell - 1$. In that case, $K = 1 + \sum_{\ell=1}^{L} \prod_{\ell'=1}^{\ell} n_{\ell'}$.

where anc(k) denotes the ancestor set of node k in $\mathcal{T}$ (the nodes, including itself, that form the unique path from k to root node 1), supp($\boldsymbol{\alpha}$) is the support of $\boldsymbol{\alpha}$, that is the index set of its non-zero entries, and $\mathrm{k}(\boldsymbol{\alpha}) = \max\big(\mathrm{supp}(\boldsymbol{\alpha})\big)$ stands for the last atom in the code. In other words, only a single branch of the tree, up to a certain node which is not necessarily a leaf, can be used to encode a given point track. This constraint also enforces the sparsity of the codes since $\|\boldsymbol{\alpha}\|_0$ cannot exceed the depth of the tree.

Algorithm 3 Tree-structured orthogonal matching pursuit

1: **procedure** TREE-OMP(`dictionary` D, `signal` $\mathbf{x}$, `target error` ε, `tree` $\mathcal{T}$)
2: $\quad\mathcal{I} \leftarrow \{1\}$
3: $\quad\boldsymbol{\alpha}_{\mathcal{I}} \leftarrow \mathbf{d}_1^\top \mathbf{x}$
4: $\quad\mathbf{r} \leftarrow \mathbf{x} - \mathbf{d}_1 \boldsymbol{\alpha}_{\mathcal{I}}$
5: $\quad$**while** $\|\mathbf{r}\|_2 > \epsilon$ and $\mathrm{size}(\mathcal{I}) < \mathrm{depth}(\mathcal{T})$ **do**
6: $\quad\quad k \leftarrow \arg\max_{k \in \mathrm{child}(\max \mathcal{I})} |\mathbf{d}_k^\top \mathbf{r}|$
7: $\quad\quad\mathcal{I} \leftarrow \mathcal{I} \cup \{k\}$
8: $\quad\quad\boldsymbol{\alpha}_{\mathcal{I}} \leftarrow D_{\mathcal{I}}^{+} \mathbf{x}$
9: $\quad\quad\mathbf{r} \leftarrow \mathbf{x} - D_{\mathcal{I}} \boldsymbol{\alpha}_{\mathcal{I}}$
10: $\quad$**end while**
11: $\quad$**return** ($\boldsymbol{\alpha}$)
12: **end procedure**

We use the K-SVD algorithm to solve the dictionary learning problem (6.6), but we modify the orthogonal matching pursuit (OMP) encoding part in order to respect the constraint in (6.7), as it can be observed in Algorithm 3. We force the use of the root node for every input datum, so in line 2, code support $\mathcal{I}$ is initialized with $\{1\}$. In subsequent steps, the greedy search for a new atom to include, in line 6, is restricted to the children of the last selected atom in the tree. In line 8, $D_{\mathcal{I}} = [\mathbf{d}_k]_{k \in \mathcal{I}}$ is the sub-dictionary indexed by $\mathcal{I}$, and $\boldsymbol{\alpha}_{\mathcal{I}}$ is the corresponding code for $\mathbf{x}$, obtained through least-squares minimization, with $(.)^+$ standing for matrix pseudo-inverse. The algorithm stops when the reconstruction is accurate enough or the maximum tree depth is reached.

6.5.2 Hierarchical coding and clustering

Having all trajectories encoded in A according to tree-structured dictionary D already provides a flat partitioning of trajectories through indices $\mathrm{k}(\boldsymbol{\alpha}_n)$ and a hierarchical one by gathering, for each node k in the tree, all trajectories such that $k \in \mathrm{k}(\boldsymbol{\alpha})$. Unfortunately, such partitions are noisy since the above dictionary learning and sparse coding are not explicitly geared toward a clustering task: nothing prevents unrelated trajectories to share atoms and, conversely, related trajectories to exhibit disjoint supports.

In the same spirit as spectral clustering conducting final K-means clustering on spectrally encoded data vectors instead of simply binarizing them, we can instead cluster the trajectories based on their codes $\boldsymbol{\alpha}_n$, with hierarchical K-means in our case. This already provides cleaner partitions. Drawing inspiration from [JLD11] who combine dictionary learning with supervised learning of linear classifiers over codes, we can go one step further: given a current hierarchical clustering of trajectories' codes, we can update our tree-structured dictionary and iterate. As will appear in the experiments, this procedure further improves the quality of track clusters. At each iteration, the dictionary learning problem to solve becomes

$$\underset{D,Y,A}{\arg\min} \|X - DA\|_2^2 + \lambda \|Q - YA\|_2^2, \quad \text{sb.t. } \boldsymbol{\alpha}_n \in \mathscr{A}(\mathscr{T}),\ \forall n, \text{ and } \quad \|\mathbf{d}_k\|_2 = 1,\ \forall k \quad (6.8)$$

where $Q \in \{0,1\}^{K \times N}$ is the binary matrix associated to the current hierarchical clustering of tracks (each of its columns belongs to $\mathscr{A}(\mathscr{T})$) and λ is a positive parameter that controls the balance between reconstruction and clustering terms. Thus, matrix Q contains the "discriminative" sparse codes of trajectories. It forces trajectories of the same clusters to have similar sparse codes. This behavior is explained and motivated by [JLD11]. On the other hand, by-product matrix Y linearly transforms the sparse code A into the most discriminative one Q.

This new objective function can be rewritten as

$$\|X - DA\|_2^2 + \lambda \|Q - YA\|_2^2 = \left\| \underbrace{\begin{bmatrix} X \\ \sqrt{\lambda} Q \end{bmatrix}}_{\bar{X}} - \underbrace{\begin{bmatrix} D \\ \sqrt{\lambda} Y \end{bmatrix}}_{\bar{D}} A \right\|_2^2, \quad (6.9)$$

and optimized w.r.t. $\bar{D}$ and A with K-SVD, after trading normalization constraints on D and Y for normalization constraints on $\bar{D}$.

6.5.3 Extension to longer videos

With the proposed approach, we are able to extract meaningful part-based clusters of tracks from short video sequences, *i.e.*, 20 frames at most. Even over such short time intervals, some of the trajectories are incomplete. We handle them by computing the reconstruction error for a trajectory only in frames it is defined, *i.e.*, replacing $\|X - DA\|_2$ by $\|M \odot X - M \odot DA\|_2$, where M is the binary masking matrix that sets to zero unknown values and $\odot$ is the Hadamard product. Examples of hierarchical track clustering obtained over short sequences are shown in Figures 6.7 and 6.8.

To process longer video shots of, say, hundreds of frames, we apply our method independently to half-overlapping chunks of 10 frames each, and we follow the spectral clustering method in [MD12] to group short-term clusters at the shot level. Given all finest-level segments (groups of tracks associated to tree leaves) extracted from all

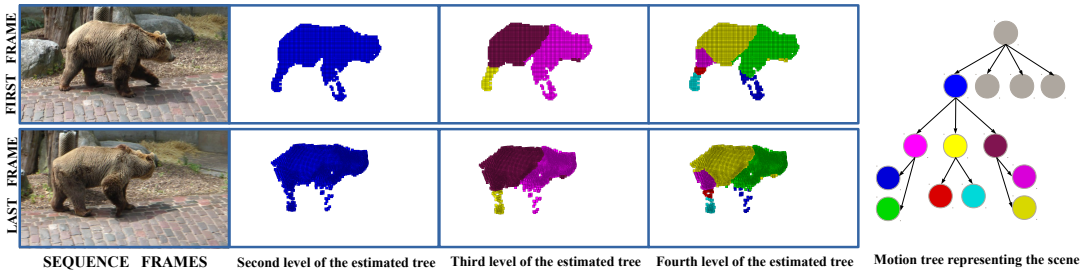


Figure 6.7: Hierarchical motion segmentation of the *bear* sequence from FMBS dataset [OMB14]. In this sequence, the camera is following a walking bear. The moving parts discovered at each level of the tree are showed in first and last frames, along with associated color-coded tree. The segments associated to the greyed nodes are not visualized for sake of readability (those from the second level of the tree are assigned to background points).

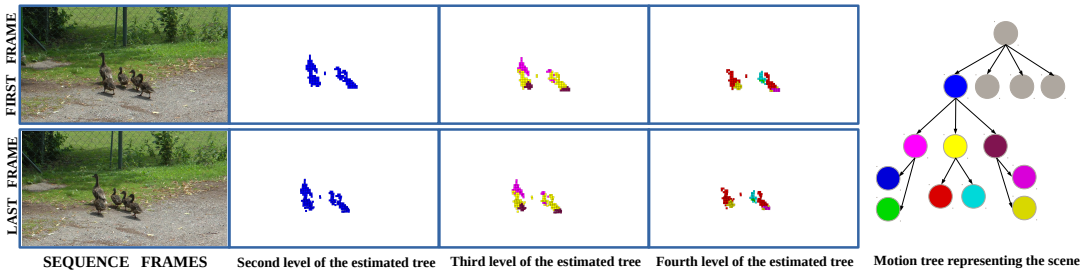


Figure 6.8: Hierarchical motion segmentation of the *ducks* sequence from FMBS dataset [OMB14]. In this sequence, a moving camera captures a small flock of ducklings. The group dynamics of the flock is correctly captured at the top of the scene motion tree. Our algorithm separates the motion of the flock at subsequent levels. However, since the motion details of the individual ducklings is not fully visible, our algorithm struggles to fully separate them. The segments associated to the greyed nodes are not visualized for sake of readability (those from the second level of the tree are assigned to background points).

temporal windows through tree-structured spare coding, we build a pairwise affinity matrix based on the number of tracks that are shared by each pair of segments. This affinity matrix is used to conduct spectral clustering. As a result, two groups of trajectories that have been formed with our approach in two distinct time windows are likely to be merged in the final segmentation if they share a lot of tracks. This might happen even if they are quite distant in time, provided long-term trajectories exist in the initial data and some of them belong to both groups.

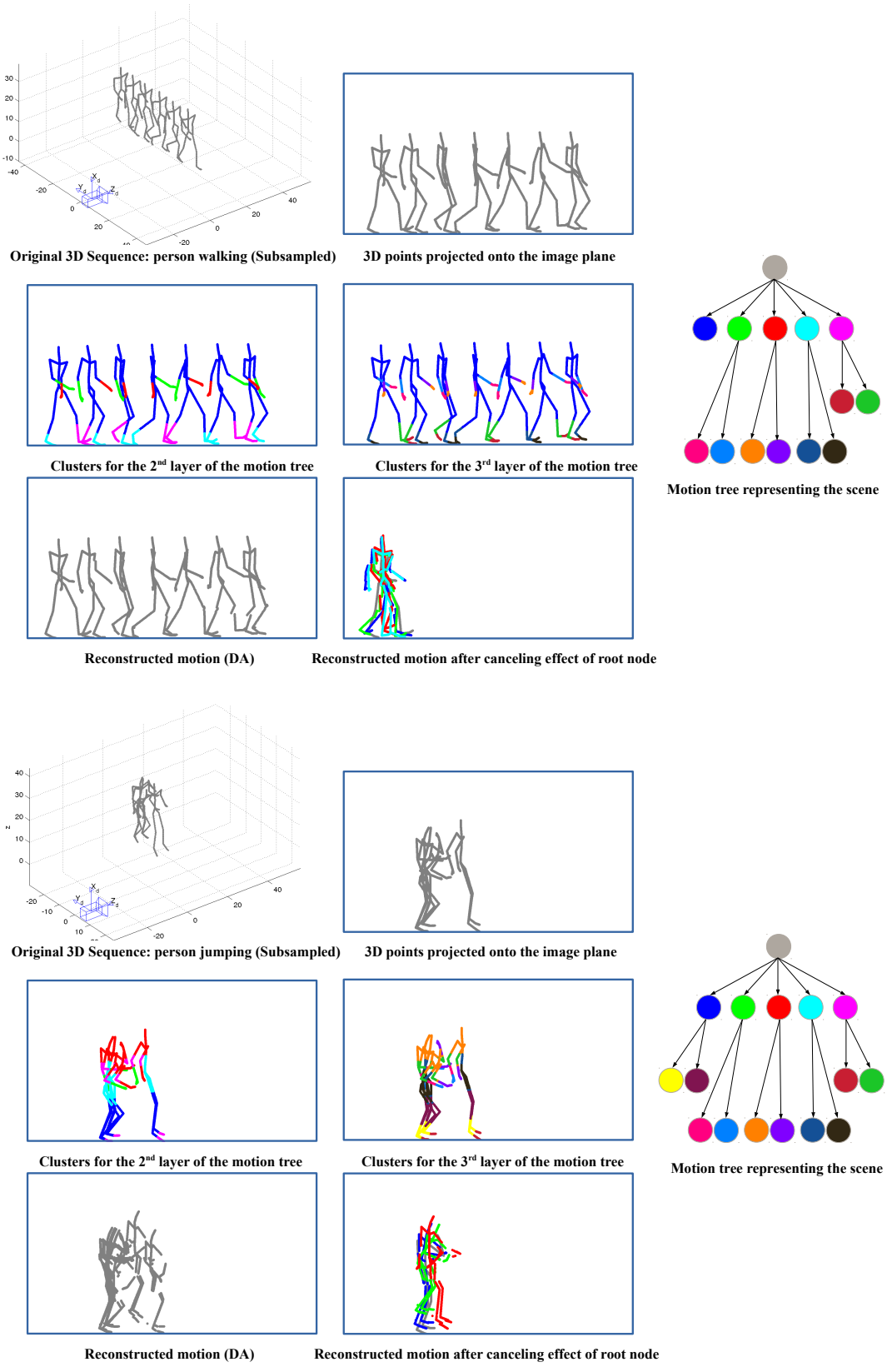


Figure 6.9: Hierarchical motion analysis of *walking* and *jumping* sequences from CMU MoCap dataset. For each sequence: (Top) Original 3D sequence and its 2D projection in the virtual camera 2D; (Middle) Clustering results on the 2nd and 3rd levels of the tree and corresponding color-coded tree. (Bottom) Motion reconstructed from complete codes and after removing the contribution of root track atom $k = 1$.

Table 6.1: Object segmentation results on the FMBS-59 test set.

	Average precision	Average recall	Average F-measure
Spectral Clustering [OMB14]	76.15%	61.11%	67.81%
Multicuts [KAB15]	81.04%	68.67%	74.34%
DL (baseline)	67.84%	58.81%	60.25%
TreeDL (ours)	76.84%	64.20%	69.46%
TreeDL$^+$ (ours)	78.41%	65.52%	72.33%

6.5.4 Experimental evaluation

Using the CMU MoCap dataset,[6] we analyse first the ability of our approach to discover motion hierarchies. From these real 3D human motion data, we can derive a structured set of 2D point trajectories: around 1500 points are sampled from the moving limbs and projected into an arbitrary virtual camera, where they produce 2D tracks. We aim at discovering a plausible hierarchical decomposition of this data, *i.e.*, one that complies with the kinematic chain of the articulated human body. For these experiments, we use a simple tree structure $\mathcal{T}$ composed of 1 root node with five children, each one with two children ($L = 2,\ n_1 = 5,\ n_2 = 2,\ K = 16$). Note that some nodes might be unused in the end, no trajectories being assigned to them. It is the case in the *walking* sequence in Fig. 6.9, for two siblings of the second level of the tree. The quality of the obtained sparse approximation $X \approx AD$ can be assessed through visualization of the corresponding track reconstructions (Fig. 6.9, third row left). Also, in order to get insight into what a specific track atom k captures, we can simply set to zero the corresponding k-th row in the code matrix A and recompute the reconstructed trajectories accordingly. We show in particular the effect of removing the influence of the root node in *walking* and *jumping* sequences (Fig. 6.9, third row right). In both cases, the root atom has captured the global trajectory of the actor. The reconstructed motion depicts the actor performing approximately the same actions, but in place. Note that each trajectory is reconstructed from at most three atoms (number of levels in the tree), which can lead to some reconstruction errors. However, despite the simplicity of the underlying tree, our approach is able to discover automatically meaningful structures among the input trajectories.

In a second series of experiments, we evaluate the clustering performance of our motion analysis framework on the FMBS-59 dataset [OMB14]. We present in Table 6.1 the results of our tree-structured dictionary learning approach with iterative refinement ("TreeDL$^+$") as described in Section 6.5.2, and compare them against state-of-the-art methods for trajectory clustering, namely the spectral clustering based method in [OMB14] and the multicuts-based approach in [KAB15]. For our approach, the tree structure is defined by $L = 3,\ n_1 = 4,\ n_2 = 3,\ n_3 = 2$ and $K = 41$. We

[6] `http://mocap.cs.cmu.edu/`

also include results for two baselines related to our approach: using unstructured dictionary learning ("DL") as introduced in Section 6.5 and a hierarchical learned representation with no further refinement ("TreeDL"), as explained in Section 6.5.1. For these two baselines and our complete system, we use the temporal sliding-window procedure described in Section 6.5.3, and use top-down hierarchical K-means on final codes to produce track clusters in each video chunk. We first note in Table 6.1 the gain brought by the tree-based structuring and by the clustering-driven extension of dictionary learning. We also note that both "TreeDL" and "TreeDL$^+$" outperform [OMB14]. While the performance of proposed system is slightly below the one of [KAB15], it gives access to precious information about the structure of the motion and the way the moving regions relate to each other (See Fig. 6.7).

6.6 Discussion

In this chapter we introduce and develop a novel idea in computer vision. That is, leveraging the compositional nature of motion for performing a hierarchical partition of the scene that is able to capture not only objects but also object parts. We initially tackled the problem on a per-frame basis by analyzing optical flow. While results proved interesting, there is no easy way to extend the approach to longer video sequences. Due to this, we proposed a method for representing and clustering point tracks that captures the natural organization of moving regions in a dynamic scene.

Our approach relies on an original dictionary learning technique that enforces a tree-based structure of dictionary and codes, and takes explicitly into account the subsequent task of clustering final codes. We showed experimentally that this method not only performs well on the difficult task of trajectory-based video segmentation, but that it also discovers automatically the hierarchical structure of dynamic scenes, both in motion capture data and in monocular video data.

CHAPTER 7

PARAMETRIC MOTION MODELING VIA DEEP NEURAL NETWORKS

In this chapter we propose an end-to-end architecture for estimating a parametric motion model for a moving scene. We demonstrate that a single deep neural network can be used for this purpose. We handle motion outliers by using the supervised training trick that is used by stacked denoising autoencoders [VLL+10]. Here we define motion outliers as those regions of the image whose motion does not correspond with the estimated parametric motion model. In other words, we seek to find a parametrized dominant motion of the dynamic scene. In order to do all of this with a single network, we leverage stacked hourglass networks [NYD16] with a final hard-coded block corresponding to the global parametric motion model estimator[1]. This block replaces the decoder part of a convolutional auto-encoder network, and it is end-to-end trainable since it is established with linear operations only. Moreover, the hard-wired decoder allows the network to output the values of the parametric motion model given an input moving scene, even when the supervision acts on optical flow maps and not the motion model values. This means that our network is able to provide, as a by-product, a concise code that could be universally used as motion descriptor without the need to provide, in the case of transmission, the offline decoders with any of the network parameters. The applications for this motion

[1] Conventionally, in the deep learning literature we call "parameters" the set of values that are learned during training (connection weights essentially). In the motion estimation literature, "parameters" also refers to those that define classic 2D motion models, such as translational or affine wraps. To avoid confusion we use the full phrase "parametric motion model" to designate the latter meaning

coding range from video compression to mobile video processing applications, where the compactness of the motion representation can reduce mobile data fares and volatile memory footprint. Furthermore, our model has also applications in the post-processing pipeline, augmented reality. The work presented in this chapter is still ongoing. However, we present encouraging preliminary results with a synthetically generated dataset.

7.1 Introduction and related work

Estimation of parametric motion models is of high interest in the computer vision literature. In particular, parametric motion models have been involved in motion segmentation [OB97, OB98, CS05], optical flow [BJ96, Far03, FBK15a, YL15], detection of anomalies [PRBB17], and tracking [MB94, BY95]. The applications span from image super-resolution [DBPP00], visual servoing [CCB98, CC01], to shot change detection [BGG99]. Even earlier in this manuscript, we demonstrated the relevance of parametric motion model estimation for detection of occlusions and hierarchical motion segmentation. Furthermore, parametric motion models offer a great balance between motion estimation accuracy, and storage efficiency in the context of mobile computing and embedded devices. In fact, in that context, parametric motion models are very valuable even when an accurate optical flow estimator is readily available. The transmission of dense optical flow fields between a remote server in the cloud and a mobile device can affect the data consumption rate, leading to high operational charges. In a more general sense, computing a parametric motion model from available optical flow maps is also meaningful when describing dominant scene motion. Furthermore, as a by-product, one can ease the task of segmenting out motion outliers when a per-pixel optical flow map and a dominant motion model are both available.

In a similar way to optical flow, classical methods of the state-of-the-art pose parametric motion model estimation as an inverse problem that is solved through minimization of an energy functional [OB95, BA96, YL15]. These methods leverage the optical flow constraint to form a data driven term encouraging motion parameters that minimize the displaced frame difference (DFD) between the input images. As opposite to per-pixel optical flow estimation, the estimation of parametric motion models is not an underdetermined problem. Indeed, the proposed models explain the image-based motion cues for all the image pixels at once (or a subset of them). Usually the number of observations, *i.e.*, pixel positions, is much greater than the number of parameters of the motion model, leading to stable solutions when no motion outliers are present in the scene.

However, under the presence of outliers, models that simply penalize the displaced frame intensity difference with the $L2$ norm encounter estimation accuracy problems. In order to overcome these issues, several methods [OB95, BA96, YL15]

propose to use different robust penalties. Furthermore, the method proposed by Black and Anandan [BA96] handles outliers directly by computing multiple parametric motion models for dominant and non-dominant motions. In the presence of long displacements, and strong camera motions, Odobez and Bouthemy [OB95] proposed a multi-resolution scheme where simpler parametric motion models are estimated at coarser scales, and incrementally updated at finer ones. In order to cope with aperture problem, an strategy to adapt the sizes of regions where motion is estimated is presented by [SES12].

Estimating motion models has been tackled simultaneously with some form of motion segmentation in the past [OB97, OB98]. This is done with the hope of increasing the scene motion estimation accuracy. In that line, Farnebäck [Far01] extends its previous method for parametric motion model estimation [Far01], sacrificing efficiency for accuracy, by proposing candidate regions that are let to grow and compete with each other according to the validity of the motion model. The procedure is akin to the one proposed in this thesis for motion segmentation in Chapter 6. Yang and Li [YL15] take this notion further and propose to segment the image and assign parametric motion models for each one of the segments, but also enforce motion continuity in between them. Larger image segments are extracted in [JBJ96] corresponding to scene motion layers. However, in this Chapter, we are more interested in those methods that fit a single parametric motion model for the whole scene, like [OB95]. This is usually known in the literature as "dominant motion" estimation. In that sense, we expect to find large motion outliers that might occupy close to 50% of the image size, since motion of complex scenes might not be explainable with a single parametric motion model. Large moving foreground, strong illumination changes, and motion blur, are all failure cases for the mentioned methods.

In the extreme cases previously mentioned, neither robust penalties nor the scale-space analysis of classical methods are enough to accurately describe motion. In scenes with large motion ambiguity, only semantic cues are able to recover the correct apparent motion. We believe, this is partly the reason why optical flow methods based on convolutional neural networks are starting to dominate modern benchmarks [DFI+15, TZTV16, IMS+17, BVS17, SYLK17]. Indeed, the hierarchical features that are learned by these approaches capture concepts that are more complex than simple color. Motivated by their success for optical flow, we propose to use Convolutional Neural Networks to predict flow maps that are subject to a parametric model. We assume for now that an optical flow estimator is already available, and can be connected to feed our own network.

We take a closer look to one of the parametric motion model estimation methods that is closer to our own work here-under. Then, we make a short introduction to the network architectures that we consider are interesting for our target problem.

7.1.1 Motion2D

Through this thesis, we have made extensive use of the parametric motion model estimator proposed by J. M. Odobez and P. Bouthemy in 1995, *Motion2D*[2] [OB95]. In particular, the method is used for estimation of parametric motion models inside the proposal regions of our per-frame hierarchical motion segmentation algorithm in Chapter 6. It is used as well for a similar task within our occlusion detection framework in Chapter 5. Indeed, more than 20 years after its creation, *Motion2D* is still a powerful tool for motion estimation[3]. However, as mentioned earlier, it struggles under contradictory visual information like large motion outliers, large occlusions, and strong motion blur.

Polynomial models of motion maps. *Motion2D* assumes that the motion map $\mathcal{V} = \{\mathbf{f}_\mathbf{x}\}_{\mathbf{x}\in\Omega}$ of a scene can be approximated with a polynomial function for every position $\mathbf{x} = (x_1, x_2)$ of the image grid Ω, so that:

$$\mathbf{f}_\mathbf{x}^\theta = \begin{bmatrix} u_\mathbf{x} \\ v_\mathbf{x} \end{bmatrix} = \mathbf{B}_\mathbf{x}\theta, \tag{7.1}$$

where, θ is a column vector containing the parameters of a polynomial motion model. Let us consider only quadratic motion models in this chapter, then, the matrix $\mathbf{B}_\mathbf{x}$ in Eq. 7.1 takes the form:

$$\mathbf{B}_\mathbf{x} = \begin{bmatrix} 1 & 0 & x_1 & x_2 & 0 & 0 & x_1^2 & x_1x_2 & x_2^2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & x_1 & x_2 & 0 & 0 & 0 & x_1^2 & x_1x_2 & x_2^2 \end{bmatrix}, \tag{7.2}$$

and $\theta^T = \begin{bmatrix} c_1 & c_2 & a_1 & a_2 & a_3 & a_4 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{bmatrix}$.

Given an input pair of images $\mathcal{I} = (I_1, I_2)$, *Motion2D* seeks to find the motion parameters θ that minimize a robust cost on the average displaced frame difference:

$$\hat{\theta} = \arg\min_\theta \sum_\mathbf{x} \rho\left(I_1(\mathbf{x}) - I_2(\mathbf{x} + \mathbf{f}_\mathbf{x}^\theta)\right), \tag{7.3}$$

where the function ρ is one of the common robust estimators [Hub11]. Since the role of the robust function in *Motion2D* is to eliminate outliers, a hard non-re-descending estimator is usually preferred. An example of such an estimator is the Tukey's biweight function.

In order to account for large motions, the minimization scheme for Eq. 7.3 includes a scale-space Gaussian pyramid, where estimation at a certain scale feeds incrementally the estimation at the following scale. The elements that form *Motion2D* are now very commonly used for any kind of motion estimation task. More precisely, the

[2] Publicly available at `http://www.irisa.fr/vista/Motion2D/`

[3] *Motion2D* is old enough to buy alcohol in the United States of America.

algorithm to minimize Eq. 7.3 is a multiscale version of the *Iteratively re-weighted least squares* (IRLS) [HW77], which tackles the problem of robust estimation by converting it into an equivalent sequence of weighted least-squares problems.

7.1.2 Deep architectures

Keeping up with the rapid advances of deep learning in a variety of problems is very tough. When given a problem and enough labeled data to tackle it with supervised learning, one is presented with the arduous task of selecting an architecture, optimize its meta-parameters, and move on with other architectures, novel or not, until results are good enough. That is why, one must first study models that have been used for tasks that are reasonably similar to the objective. Still, the literature is indeed very extensive, which is why we present a short summary of the considered architectures of the state-of-the-art.

Estimation of parametric motion models is a regression problem that share many similarities with optical flow. That is why we make a first stop to look at recent advances in this area.

Deep learning for optical flow. Convolutional neural networks were used successfully to learn a mapping function from pairs of images to optical flow by Dosovitskiy *et al.*, [DFI+15] as recently as 2015. Their model was coined FlowNet. However, the use of convolutions for optical flow has a longer history. For instance, Farnebäck implemented his motion estimation method by means of separable convolutions in [Far00]. Weinzaepfel *et al.*, [WRHS13] rely on a large stack of patch-based convolutional responses. To the best of our knowledge, however, the method of Dosovitskiy is actually the first one to use learned convolutional filters to perform the mapping [DFI+15].

A few elements from the work of [DFI+15] has to be considered when tackling similar tasks. Performing a complex map-to-map transformation requires to capture high level features from data. In order that features can pick up global information at the total spatial extension of the input maps, they are implemented in a contractive fashion. Indeed, this is a very common practice in applied deep learning. A second part of the network must then take those features and expand them so that they are able to restore the spatial resolution of the output. An encoder-decoder architecture comes easily to mind. However, special attention must be taken for the motion estimation problem. Indeed, optical flow networks must have good localization properties. **Forward skip connections** from contractive layers are connected through convolutions to the expanding part of FlowNet, and they alleviate the bad localization issue of deep networks and simple encoder-decoder architectures.

The expanding part of the network of FlowNet is implemented with upconvolutions. Due to the known artifacts of these operations [ODO16], to account for spatial

smoothness of optical flow maps that might not be captured by the network, and to help bringing the output flow to the desired resolution, FlowNet relies on a variational refinement stage.

Another interesting element of FlowNet is that it was trained on **synthetic dataset** called *FlyingChairs*. The dataset contained around 25.000 images of chairs on background images extracted randomly from *Flickr*. The backgrounds were assigned with a random rigid motion, and the foreground, composed of computer-generated chairs, with another one. A simple strategy for data augmentation allows the network to generalize from that dataset to real images.

The results of FlowNet are impressive considering that the pipeline is learned in an end-to-end fashion with synthetic data, and, powered by modern GPUs, they are computed in almost real-time. However, the results were not state-of-the-art at the moment of publication. The evolution of FlowNet, FlowNet 2.0 [IMS+17], brought together new insights that allowed the model to outperform previous methods, while also being one of the fastest. Perhaps the most important elements introduced by FlowNet 2.0 are:

- **Curriculum learning.** One of the issues of the original FlowNet is the poor behaviour for small displacements. To tackle this, FlowNet 2.0 leverages a second synthetic dataset depicting more complex motions (and of smaller magnitude in average), coined *Flying3DThings*. The optimal schedule for training was to first use *FlyingChairs*, and then the *Flying3DThings*. The offered explanation is that the network learns more easily to perform color matching with a simpler dataset. More complex priors like 3D motion and realistic lighting are better left for a second training stage. All in all, the order of presenting training data with different characteristics to the network seems to be important.

- **Correlation layer.** FlowNet 2.0 uses a correlation layers after an initial feature-extraction siamese block. The network is then released of the task of learning how to match features. The correlation layer was initially introduced by FlowNet, but only provided better results for FlowNet 2.0.

- **Image warping and task separation.** FlowNet 2.0 is a set of several networks that specialize on different things. One network solves for long displacements, another one for small displacements, and another one combines and refines them. In order to account only for small displacements, the input flow maps of the intermediate network are warped by a bi-linear interpolation layer.

A more recent work [SYLK17] offers very competitive results on recent benchmarks, while being more compact than FlowNet 2.0. The model, named PWC-Net (Pyramids-Warping-Cost Volume), combines important architectural schemes that are supported in well known priors for optical flow estimation. The first one, **pyramidal feature**

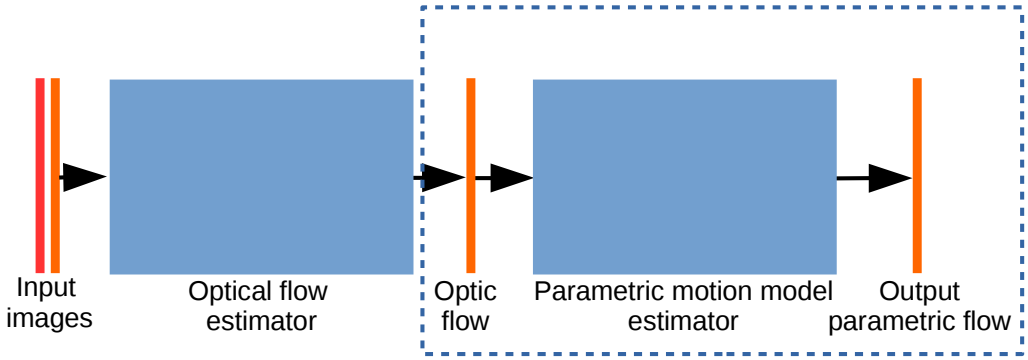


Figure 7.1: **Schematic of the pipeline for parametric motion model estimation.** The dotted rectangle indicates where we focus on this work. If the input optical flow is computed by a deep neural network, the whole pipeline from images to the output parametric motion model becomes trainable from an end-to-end and further refinement is possible.

extraction is based on the common hierarchical scheme for solving optical flow by classical methods. PWC-Net leverages a **warping layer that acts on the learned feature maps**, not the input images, allowing the network to be more compact as there is no need to learn separate feature extractors for each pyramidal scale. PWC-Net seems to be both faster and more accurate than FlowNet 2.0.

Geometric matching. A problem related to ours, geometric matching consists of finding a parametric transformation of the image grid, allowing the registration of input frames. Recently, Rocco *et al.*, [RAS17] proposed a neural network model that is capable of registering pairs of images that do not necessarily belong to the same image sequence. The target parametric transformations were affine and thin plate splines [Boo89]. In their model, the problem is divided into three tasks: symmetric feature extraction with a Siamese network initialized with VGG features [SZ14], a dense correlation layer similar to the used by FlowNet, and a regression layer, which infers the image grid transformation.

Deep learning for pose estimation. On another regression problem that has been recently tackled by CNNs, human pose estimation has achieved extraordinary results with the so-called stacked hourglass networks (SHN) [NYD16]. Newell *et al.*, stated the pose estimation problem as a dense map-to-map inference problem, making it similar to ours.

The important elements that allow such networks to perform so well can be summarized as follows:

- **Skip layers** with symmetric connection from the convolutional operations in the contractive part of the network, to the up-convolutions in the expansive part of the network. This particular design essentially allows the network to be aware of global and local information at every stage of the decoding part. A single module with this design properties is called an hourglass module.
- **Stacks.** Stacking hourglass modules seems to allow the SHN to perform repeated top-down, bottom-up operations that might be essential on capturing different aspects of the pose estimation problem at every module.
- **Residual connections.** The residual connections, as introduced by [HZRS16a] allow very deep models to be trained stably. Each residual module is by-passed by an identity transformation that allows gradients flow freely through the network. A deeper understanding of residual learning can be obtained by looking at [HZRS16b].
- **Intermediate supervision.** SHN allows intermediate outputs to be used in the training loss. This procedure guarantees that each hourglass modules learns something about the pose estimation problem, and further stabilizes the training.

7.2 Our model

We take the considerations presented in previous section, and use them to propose a new model which is tailored to the parametric motion estimation problem. First, with our model we focus only on determining the parametric motion model assuming a relatively accurate optical flow map is already available (Fig. 7.1).

7.2.1 Optical flow outlier elimination network

A first CNN for "cleaning" off motion outliers from optical flow maps is proposed. The problem of outlier removal in this context is similar to the problem of image denoising carried out by the stacked denoising autoencoders (SDA) [VLL+10][4]. The features learned by the proposed network are meant to fed a posterior processing stage that infers the underlying parametric motion model that explains the input optical flow minus the outliers.

[4]Denoising in SDA is more of a proxy task to facilitate the unsupervised learning of meaningful features from data. However, similar ideas led to a very successful method for image denoising in [JS09].

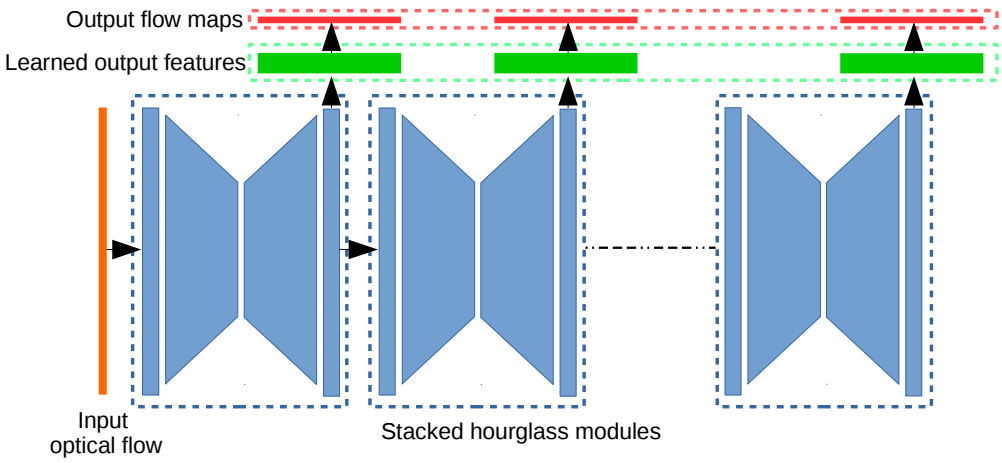


Figure 7.2: **Stacked hourglass modules with intermediate outputs** Each hourglass module (dashed blue rectangles) outputs an intermediate "cleaned" flow map that can be used for supervision. The learned output features (green) will feed posterior processing stages.

Considering the encoding-decoding architecture of [VLL+10], but looking forward to methods that are more optimal for regression problems, we focus on the stacked hourglass networks [NYD16]. As in the original paper, we use single hourglass modules as residual units. This is, the input and output feature maps of a single module are summed up so its output only needs to express a residual transformation. This trick has allowed all kinds of networks go very deep.

7.2.2 Motion2DNet

The second block of our network, coined *Motion2DNet*, takes the output features learned by the outlier-elimination network as input, and pass them through another encoder-decoder kind of network. The encoder part is pretty much identical to the modules explained in previous section, except that there are not skip connections in this one[5]. However, for the decoder part, we consider the way how parametric motion models act on the image grid. In particular, we force our network to learn quadratic motion models by employing Eq. 7.1 directly on the output of the learned encoder.

In order to achieve this, we let the encoder part of Motion2DNet learn a mapping function from the input features to the parameters of a quadratic motion, namely θ.

[5]To clarify, encoder-to-decoder skip connections are not implementable in this part of the network. The "decoder" part of Motion2DNet is in fact a hard-wired block taking the computed motion parameters as input, and delivering a motion map driven by such parametric motion model.

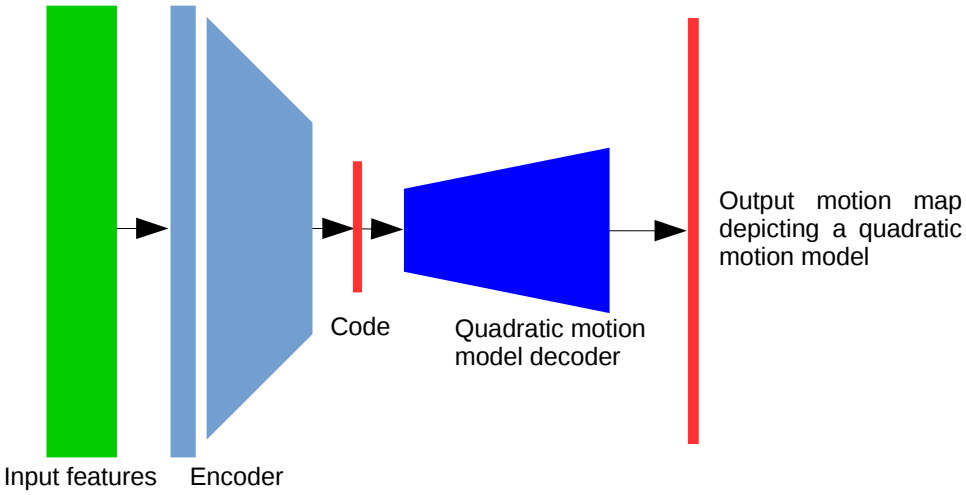


Figure 7.3: **Motion2DNet.** The input features in green, stemming from our outlier-elimination network are mapped to a code which effectively becomes the parameters of a quadratic motion model when passed through the decoder part (navy blue).

The decoder part of the network, then, takes the image grid positions $\mathbf{x}$ and performs the necessary operations to build the matrix $\mathbf{B}_\mathbf{x}$. Finally, $\mathbf{B}_\mathbf{x}\theta$ is computed, providing a motion map that effectively obeys a quadratic motion model. Higher order motion models, or even simpler ones (*i.e.*, affine), might be considered, but those are left for future work.

We make emphasis again on the fact that, since the output decoder of Motion2DNet is hard-wired, the output of the neural encoder is forced to be real quadratic motion model parameters. Thus, our complete network, after training, provides three important outputs:

1. A simplified optical flow map (outputs of hourglass modules). Based on the intermediate supervision, the computed optical flow maps at this stage of the network are cleaned-up reconstructions of the input optical flow map.

2. A clean quadratic-motion optical flow map (final output map).

3. The parameters of the quadratic motion model (intermediate code of Motion2DNet).

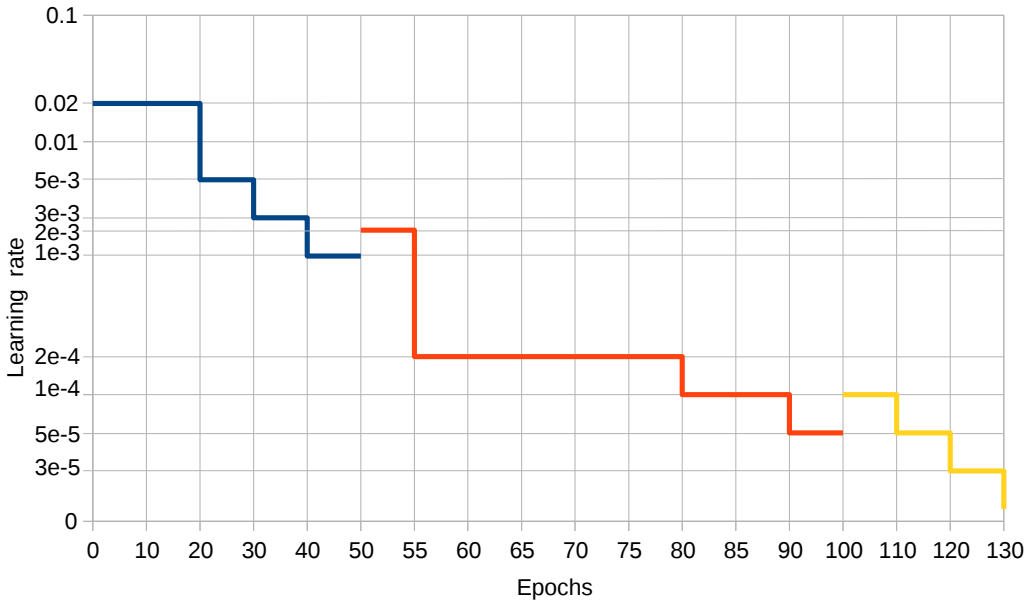


Figure 7.4: **Learning rate schedule for the first dataset.** Blue: Learning rate for the first part of the network. Red: For Motion2DNet. Yellow: the two parts of our entire model are trained together with the indicated learning rates.

7.2.3 Training schedule and datasets

Datasets. Synthetic datasets have a long history in motion estimation evaluation and training [BSL+11, ACC12, BWSB12a, DFI+15]. Inspired by recent success of such datasets for end-to-end learning of optical flow, we propose to use synthetically generated pairs of optical flow and corresponding quadratic motion maps for training our entire network.

In the spirit of the training schedule designed for FlowNet 2.0 [IMS+17], where the network first learns the difficult task of general point matching, and then continues learning other priors by splitting the learning in two corresponding datasets, we generate two slightly different synthetic datasets.

For the first dataset, we uniformly sample quadratic motion models from a mono-modal distribution of mid-range motions. For the second dataset, we sample from a multi-modal distribution encompassing small, mid, and long range motions, and more realistic motion models, *i.e.*, the affine and constant parts of the model are distinctively of larger magnitude than the quadratic part. The second dataset contains also more and larger outlier regions. For both datasets we generate 55.000 pairs of flow maps. We split them in 54.000 pairs for training, and 1.000 for testing.

Furthermore, akin to the training trick employed by stacked denoising autoen-

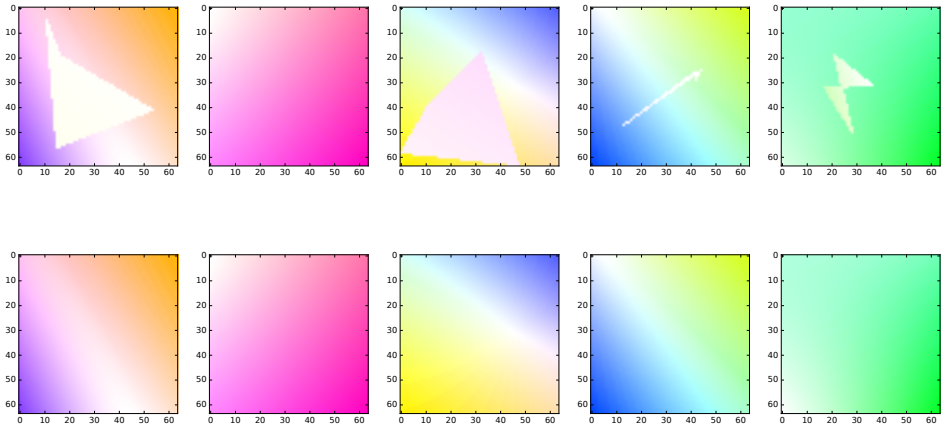


Figure 7.5: **Samples of the generated datasets.** Top: Motion maps with random outliers. Bottom: Corresponding clean flow map stemming from a quadratic motion model. The color code corresponds to standard HSV colormap for optical flow, as in [BWSB12a]. See Fig. 7.6 for details of the color code.

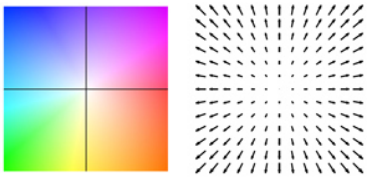


Figure 7.6: **Color code for visualization of optical flow.** HSV colormap (left) used to describe the magnitude and angle of corresponding vectors (right).

coders [VLL+10], we generate training pairs of clean optical flow maps and maps contaminated by motion outliers. The motion outliers are simulated by random regions undergoing a different affine motion model. This procedure is replicated for both of the generated datasets.

Training details. For the first dataset, the algorithm used for optimization is Adam, as described in [KB14]. We train our network by parts in order to obtain better results. For the first dataset, this leads to a mildly complicated learning rate schedule, which is summarized in Fig. 7.4. We use standard L2-norm between the probed outputs and the ground-truth flow maps as loss function[6].

[6]Recall that we employ the intermediate supervision described in [NYD16]. Effectively, the loss function is the sum of partial losses between the intermediate outputs and *the same* optical flow ground-truth map.

For the second dataset, the training aims at learning more complicated priors for parametric motion model estimation. The combination of long, mid and short range motions, plus the larger outliers are designed to achieve this goal. However, with the complicated combination of motion ranges, we noticed that simple *L2* norm as loss function did not help further learning. In fact, the learning was focusing on the larger motion vectors only, an effect that is observed as well in FlowNet [DFI+15], providing poor results for small motions. FlowNet 2.0 [IMS+17] deals with this effect by including a warping strategy and separating the task of solving for long range and short range motions in two networks.

In our case, to alleviate this problem, we change the simple *L2* norm initially used for the first part of the training, for a custom robust function defined by:

$$L_{\mathbf{x}}(\hat{\mathbf{f}}_{\mathbf{x}}, \mathbf{f}_{\mathbf{x}}^{GT}) = \begin{cases} \lambda ||\hat{\mathbf{f}}_{\mathbf{x}} - \mathbf{f}_{\mathbf{x}}^{GT}||_2^2 \text{ if } ||\mathbf{f}_{\mathbf{x}}^{GT}|| \leq \lambda \\ ||\hat{\mathbf{f}}_{\mathbf{x}} - \mathbf{f}_{\mathbf{x}}^{GT}||_1 \text{ if } ||\mathbf{f}_{\mathbf{x}}^{GT}|| > \lambda \end{cases} \tag{7.4}$$

where $\hat{\mathbf{f}}_{\mathbf{x}}$ and $\mathbf{f}_{\mathbf{x}}^{GT}$ are the estimated and ground-truth optical flow vectors, respectively. λ is a positive parameter that we set to 10.0. The total loss is then:

$$L(\hat{\mathbf{f}}, \mathbf{f}^{GT}) = \sum_{\mathbf{x}} L_{\mathbf{x}}(\hat{\mathbf{f}}_{\mathbf{x}}, \mathbf{f}_{\mathbf{x}}^{GT}) \tag{7.5}$$

Eq. 7.5 aims at balancing the ground-truth motion vector magnitudes, penalizing robustly large motions, while keeping a quadratic loss for small motions.

For the second dataset, we set the learning rate to $2e-3$ and divide it by 2 every 10 epochs for 100 epochs.

Data augmentation. In order to improve the generalization properties of our network, we perform data augmentation by adding Gaussian noise independently for the two components of the motion map with $\sigma = 0.001$ to the input data. We also perform a random 90° rotation for corresponding pairs of flow maps. Other transformations of the training data are currently being explored.

7.3 Experiments

We present qualitative and quantitative experiments on two datasets. The quantitative experiments are performed on 400 samples from the generated testing dataset. Further experiments must be carried in more realistic datasets. However, such datasets are yet to be made available. The qualitative results are obtained by running our method on the *FlyingChairs* dataset, which was not used during training.

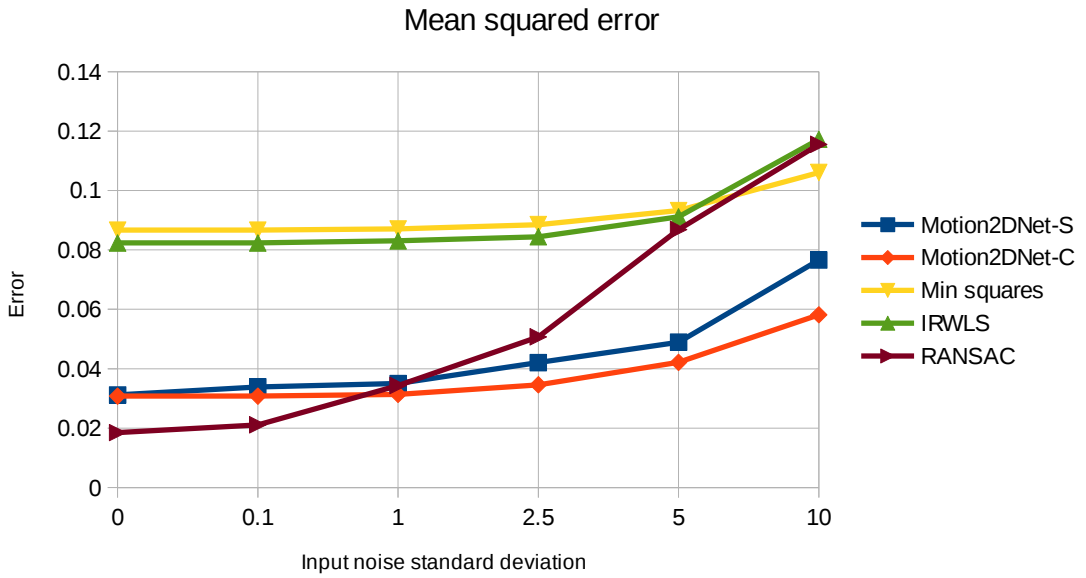


Figure 7.7: **Results on our synthetic dataset.** We take 200 samples from our testing dataset, and run all the baselines and our own full network on them. We repeat the experiments four more times with increasingly amounts of additive Gaussian noise.

Baselines. We compare our full model against four baselines. The first one is a simple minimum squares estimator for the motion parameter vector θ given matrices $\mathbf{B}$ and motion vectors $\mathbf{f_x}$. The second one, is a robust method based on RANSAC, which has demonstrated to be a very powerful estimator for visual geometry problems [HZ03]. The third one is the IRWLS robust solver by [HW77], which in turn can be considered as the classical Motion2D version of our method. In our setting, IRWLS solves for a robust cost with the Tukey function on the difference of the input flow and the flow stemming from the estimated parametric motion model. We use the same configuration parameters proposed by [OB95]. Finally, the last baseline consists of our DeepMotion2D network without the outlier removal modules (noted Motion2DNet-S, for its simpler). Regarding our complete network, we only ran experiments with a single input hourglass module. We call our complete model *Motion2DNet-C*. In future work the combination of several hourglass modules will be explored, so to exploit more deeply the top-down-bottom-up power of this architecture.

Quantitative results. We start by providing results of our method on a fold of our testing dataset consisting on 200 optical flow maps. From those 200 optical flow maps, around 130 are contaminated by small outliers occupying a maximum of 30% of the image grid. We proceed to add Gaussian noise of increasing standard deviation in sev-

Table 7.1: **Experiment on a test set with large outliers.** 200 optical flow maps generated by randomly-chosen parametric motion models were contaminated by motion outliers of a maximum area of 50% of the image grid. We report the mean squared error (MSE) for each method. The best results are shown in bold.

	Min squares	IRWLS	RANSAC	Motion2DNet-S	Motion2DNet-C
MSE	0.114931287	0.109673638	0.0730486181	0.0601921080	**0.0462929937**

eral runs on the dataset, as seen in Fig. 7.7. With this experiment we demonstrate the stability of our models with respect to input noise. The minimum squares estimator remains stable as well, since it is the optimal estimator for data contaminated with Gaussian noise. The RANSAC-based robust estimator initially provides the best results on the dataset, but quickly becomes unreliable with large amounts of noise. It is worth mentioning that, during training, our model never encountered samples with such large standard deviations. More precisely, our data augmentation procedure was set to only 0.001. With these experiments we demonstrate not only our robustness, but also that our network is not over-fitting the dataset, but rather generalizing the concept of robust parametric motion modeling. To demonstrate this further, we perform another experiment on a different set of 200 optical flow maps contaminated by large outliers occupying a maximum of 50% of the image grid. The results of such experiment are shown in Table 7.1.

The quantitative experiments show clearly the value fo adding a first hourglass module to our Motion2DNet. This motivates future work on optimizing or architecture, and determining how many hourglass modules are needed to decrease the average error by at least one order of magnitude, if even possible.

Qualitative results. We leverage the FlyingChairs dataset for visually evaluating the quality of the motion maps when the inputs are more realistic optical flow fields. We show in Fig. 7.8 an interesting behavior of our model. Perhaps excepting the third row, for all the presented examples the difference between the output and input flow maps are the regions corresponding to the moving chairs. Our network succeeds to model the dominant motion of the input images without getting too distracted by foreground motion. The last row of Fig. 7.8 depicts a very difficult case: four different moving chairs contaminate the background motion. However, the output of our network correctly captures the background motion. We believe these results are encouraging and motivate further future work.

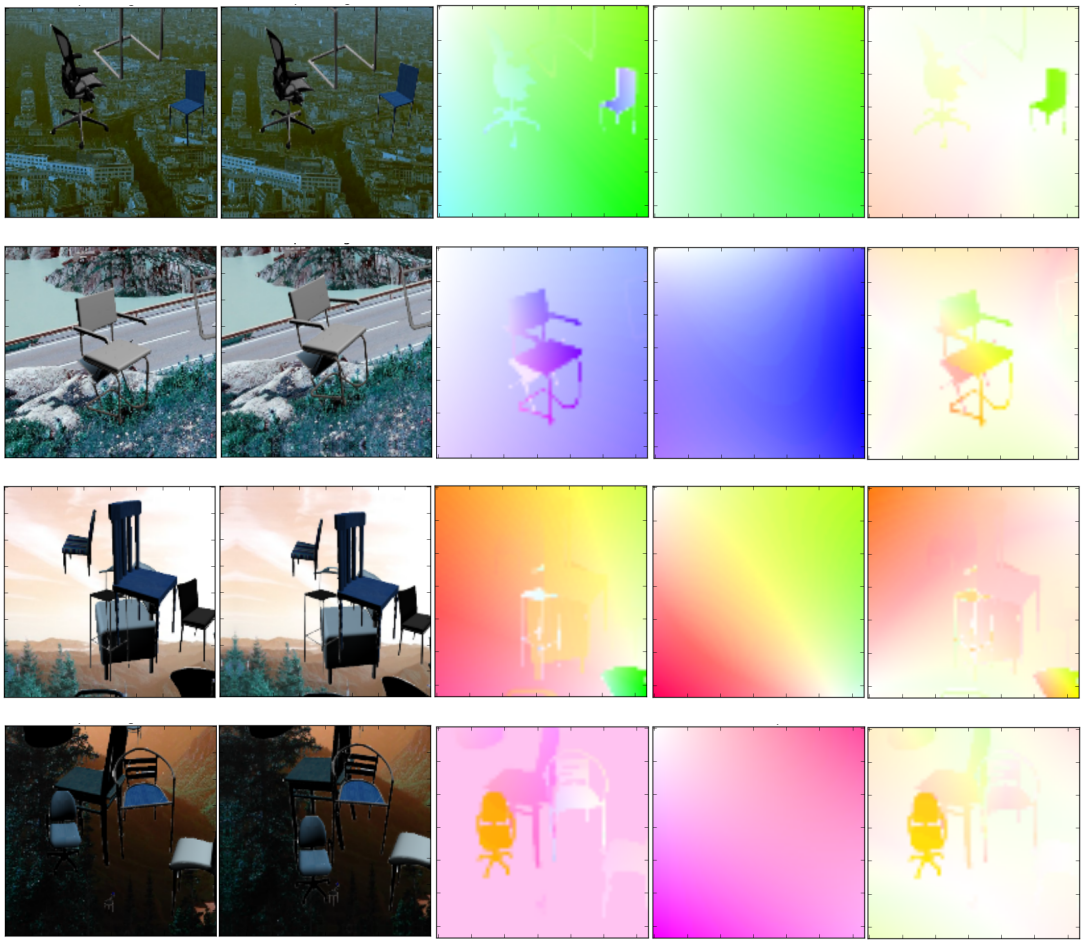


Figure 7.8: **Visual results on the FlyingChairs dataset.** A different scene is shown in each row. The input pair of images are displayed first. The third column shows the input optical flow map. Next, the output of our Motion2DNet. Observe the simplification of the input optical flow in a parametric motion model that got rid of outliers (motion of the chairs). Last column: normalized per-pixel difference between the output and input flows. Observe that the larger differences (more saturated colors) are, generally, in the pixels depicting a moving chair.

7.4 Discussion

We have presented our ongoing work on estimation of parametric motion models with convnets. We have demonstrated the value of such approach by comparing our results with classical minimum squares and a two robust methods: RANSAC and IRWLS. We score favorably through a non-trivial range of experimental settings. In particular, our model seems to be very robust to different types of outliers. We believe

further research in this direction is still necessary. In particular, more experiments must be carried on to optimize the architecture of the presented model. For example, within each hourglass module we preserve much of the original design described in [NYD16]. However, the original skip connections might be replaced in the close future to favor recent advances in deep learning. In particular, in the current model, inner skip connections in between encoder and decoder parts of a single hourglass module are carried through convolutions. Converting these skip convolutions to identity mappings might be helpful [HZRS16b]. Densely connected nets might also be important to improve performance [HLWvdM16].

Furthermore, we are currently working on a more realistic dataset for better evaluation and training of our model. Although we have shown that our model is able to generalize to a number of different cases and types of motion, we believe that there is still a gap between the presented synthetic datasets and real life scenarios. Future work includes applications to motion saliency detection, and motion outlier elimination for visual servoing and others. Of particular interest, our approach might fit the 3D data scenario, where a quadratic motion model explains perfectly the sensed motion information. In this scenario, foreground segmentation might be eased by our model.

PART

GENERAL CONCLUSIONS

CHAPTER 8

GENERAL CONCLUSIONS

In the computer vision literature, motion analysis has played an important role through the evolution of the field. Methods and algorithms have evolved gradually but steadily since the first papers appeared almost 50 years ago. However, a set of important concepts related to motion analysis have been left out or only sparsely investigated during the progress of the field. We believe that some of these concepts lie at the core of *vision* itself, and are valuable missing pieces towards the incredibly complex puzzle that artificial vision is. This manuscript revolves around some of these missing pieces, which are all related to what computer scientist usually call *dynamic scene analysis*. As opposed to static scene analysis, which has the goal of extracting as much information as possible from still images, dynamic scene analysis focuses on video sequences. The former relies solely in appearance based cues, while the latter is fed by both appearance and motion related cues.

The human vision system is dynamic by nature. Indeed, when presented with an interesting scene which is being occluded, it is only natural for humans to move and resolve for the missing data. As discussed in the introductory chapters of this thesis, this capacity allowed our vision system to evolve in such a way that complex motion-based percepts are activated even when very little appearance information is available. Two important aspects are key for the complex dynamic scene analysis carried out by the human vision system. The first of them are low level features, the ability to detect and understand optical flow and occlusions allow the human brain to encode complex information at subsequent stages of analysis. However, in the vision literature optical flow has received most of the attention, with only sporadic works focusing on occlusions. The second aspect, understanding the hierarchical structure

that motion composition builds up in natural scenes has demonstrated to be, in the human vision system, an extremely important mid-level phase of analysis upon which very complex vision tasks rely. In this thesis, we take these two key aspects of human vision and bring it to the computer vision realm.

In an attempt to convey our results in an as organized as possible way, we divided our work in two main parts. The parts group the methods according to the the necessity or availability of user interactions. Thus, we first introduce two user-guided methods for object segmentation and motion estimation, and rotoscoping that are close in spirit to the notions of compositional motion hierarchy, by proposing solutions based on shallow hierarchical structures. The second part dealt with fully automatic methods. It included first an occlusion detection method and then our algorithms for discovering motion hierarchies from video shots.

8.1 Our contributions and results

8.1.1 On object flow

We presented a first approach to the compositional nature of scene motion. By studying dense motion estimation for given scene objects, we demonstrated experimentally that a global estimation of the object motion, and its accompanying support, can leverage the dense estimation of motion vectors inside said support. In particular, we demonstrated that by explicitly analyzing coarse and fine motion vectors, motion estimation becomes more robust to occlusion and drift. The availability of such object-based dense motion estimators might help certain video editing tasks in the post-production pipeline. For example, simple re-colorization of scene objects, or non-rigid edit propagation are usually tackled by integration of optical flow over time. A precise per-object estimation of motion can provide with more efficient ways to tackle these tasks.

8.1.2 On ROAM and ROAM+

In the post-processing pipeline, rotoscoping and trimap tracking are fundamental tasks that are currently approached mostly by hand. The rapid growing of professionally produced visual content is creating a huge demand for algorithms that support these processes. With this in mind, we proposed an original graphical model, ROAM, that integrates a rich set of key ingredients for rotoscoping. These ingredients are local and global appearance models, integrated through a shallow hierarchy encompassing object parts or landmarks, and fine grained contour points. Our model provides very encouraging visual results on the video segmentation and for rotoscoping tasks, while also scoring state of the art results in modern segmentation benchmarks. Moreover,

we presented an extension to such model, ROAM+, allowing the determination of trimaps, intermediate maps that are used for high quality video compositing. We further improve rotoscoping tracking results thanks to an original roto-curve refinement method derived from the tracked trimaps. Finally, ROAM and ROAM+ are additional examples of one of our main hypotheses. That is, a hierarchical analysis of motion composition is key in solving difficult computer vision tasks, in particular those related to video object segmentation and tracking. Certainly, as demonstrated by detailed experiments, considering landmark motion as a connected tree-shaped model is one of the main reasons for the success of our method.

8.1.3 On occlusions

We think that one missing component in the current computer vision toolbox is a reliable occlusion detection method. In particular for the video post-processing pipeline, where accurate edit propagation are sought after, occlusion detection has an important role. We proposed an algorithm that aims at separating optical flow accuracy from occlusion detection accuracy as much a possible, while still acknowledging the interleaved nature of the two tasks. Our method relies on a set of parametric motion candidates, which are randomly sampled following a hierarchical partitioning of the image space. We seek after *plausible* motion maps that allow us to determine occlusions under a novel occlusion criterion. This criterion compares punctual image color values against region-based color models. This alone improves occlusion detection accuracy by a large margin in comparison to standard point-to-point criteria like displaced frame difference and forward-backward flow consistency. We support our claims with extensive experiments with real images and standard benchmarks.

8.1.4 On hierarchical motion analysis

The results found by studying object flow and ROAM are a hint that points towards exploiting the compositional nature of motion with more care. In order to do this, we propose two methods that aim at discovering the underlying motion hierarchical tree from arbitrary video scenes. We start by assuming the existence of a structure-aware set of plausible parametric motion models for a given pair of frames, and pose the problem of finding such hierarchical structure as an inverse frame-to-frame problem. We present a block-based minimization method which alternates between possible combinations of motion compositions, and the parameters of the models, obtaining a hierarchical segmentation of the input scene. This first intermediate result is interesting in that it allows to discover parts that are closely related to the real scene objects and its parts in an instantaneous basis. However, the method is based on a sampled amount of window-based motion models, similar to the ones used for the

occlusion detection part, which are only valid in a per-frame basis. This makes our first method difficult to extend to longer video sequences.

In order to overcome these issues, we depart from motion trajectories, which capture motion dynamics of whole video shots. By proposing a novel structured dictionary learning algorithm, we transform the trajectory encoding into an embedding that can be used for hierarchical segmentation. When clustering point trajectories with such embedding, we obtain on one hand a tree representing the scene motion composition, and a partition of the video into objects and object parts on the other hand. This is unlike unsupervised video segmentation methods of the state-of-the-art, which only provide with flat partitions of the scene which do not necessarily correspond to visual objects.

8.2 Current and future work

Motion2D One primary ingredient of the work presented in this manuscript is the computation of region-wise parametric motion models. As the reader might have noticed, we rely heavily on Motion2D [OB95], a method that is more than two decades old. Motion2D is based on robust statistics and an iterative optimization method. It provides surprisingly good results, even when compared to more recent methods to estimate image motion. However, due to the inherent limitations of the optical flow constraint, Motion2D will fail under difficult cases like strong color changes, textureless regions, and long displacements. Several of these limitations remain unreachable for methods that leverage hand crafted models. For this reasons, ongoing work explores learnable architectures for solving the parametric model estimation, including robustness to motion outliers. We are interested in deep convolutional neural networks, since they can provide the problem with a higher level of visual semantics, a plus when solving for the difficult cases mentioned earlier. Motivations for providing with such a solution go way beyond the spectrum of this thesis, as Motion2D lies at the core of many methods in very diverse areas of computer vision.

ROAM Two branches of future work are expected to be tackled in the close future for ROAM. The first one is to take advantage of the flexibility of the proposed hierarchical model, to extend ROAM so it can be used for multiple object instances along a video sequence. The second one is to address the roto-curve tracking problem as a fully trainable model. Our recent experiments have shown that it is possible to learn such a complex structured problem, but our current results are not better than the ones reported in this manuscript. However, we strongly believe that this is the case only because of the scarcity of the training data. In fact, as per 2017, there is not a single dataset with the required characteristics.

Hierarchical motion analysis We have demonstrated that our dictionary learning algorithm yields sensitive results. This is a small step towards interactive motion editing of video scenes. However, we have worked on motion trajectories only. Extending our work to process videos in a denser way is still an open problem. Moreover, better algorithms might provide with less noisy results. In particular, since the proposed algorithm is extremely greedy in nature, more elegant solutions that provide more guarantees can be proposed in the future. Finally, we believe that our algorithm and future related implementations can be applied to other fields. Exploration in the contexts of audio processing and general signal source separation might be worthwhile.

The video post-processing pipeline A large amount of applications derived from the film production industry are prone to be improved with novel and current computer vision techniques. We have presented a compendium of solutions in this thesis, but further research is needed to tackle other interesting problems. Most of the future work in this line will probably require low-level and intermediate motion analysis tools, so we hope that our occlusion detection method, object flow and trajectory clustering algorithms will provide the initial spark for other related problems.

APPENDIX A

A METHOD FOR VIDEO OBJECT SEGMENTATION

In this annex, we present a simple method to segment objects of interest in video sequences by combining robust background and foreground point tracking with dense joint color and motion-based segmentation. Our approach is sequential in time, avoiding a global processing of the video, while being simple and generic. This makes the method attractive for online applications, including video editing or augmented reality, as it can be adapted for both automated and interactive work-flows. We present visual and quantitative experiments to compare with existing algorithms. The video segmentation method explained in following sections lies at the core of the **simple object flow** method explained in Chapter 3.

A.1 Introduction and background

The problem of interactively segmenting objects of interest in still images has been widely and successfully approached in the last decades. Different user interactions are used, from bounding boxes to strokes indicating foreground and background elements of the image. Less successful approaches have been proposed in the past for cutting out objects in video scenes. This appendix is focused on the problem of segmenting out an object from a video sequence given a bounding box around the object in the first frame.

The most common approaches that can be found in the literature include (but are not limited to) using optical flow to propagate cut-outs from a few keyframes

through the whole sequence [CAC+02]; extending energy-based image segmentation to video volumes [APB07, TFM10]; tracking local classifiers that lie on the object borders and use color, shape and motion features [BWSS09]; or tracking spatially-aware color models [YZC+07]. While offering certain benefits, most of the previous approaches present a number of limitations that hinder the segmentation process. For instance, cut-out methods based on global partitioning of the video volumes tend to oversimplify the complexities of video scenes by simply using color information without taking into account scene interactions or motion features, leading to errors in the segmentation. Furthermore, when the user tries to correct these errors by adding more cues in the form of strokes, the new solution may affect segmentation performance on subsequent and/or previous frames, falling into an unstable pipeline. Otherwise, methods that use simple motion estimation to propagate local classifiers [BWSS09], or segmentation borders themselves [CAC+02], can be affected by the well-known optical flow drawbacks like the aperture problem or singularities across the motion boundaries. Latter problem is critical since object contours tend to agree with motion boundaries, affecting directly the segmentation outcome. In fact, most of the optical flow methods struggle to provide accurate, not over-smoothed, estimate along such borders. There is also a computational overhead in some of the state-of-art methods, which require either several (or all) of the frames be loaded in memory, or an expensive global optimization.

Looking forward to a method that can be used in a broad range of applications by being precise while maintaining computation efficiency, we propose a frame-by-frame approach that can achieve robust segmentation results even in difficult cases. Except for the first frame, where the segmentation is computed based only on user provided cues, our method relies on the previous frames cut-out by pulling high-confidence labels to the current frame. For instance, a point that is labeled as background because it is spatially far from the object of interest at a given instant in time, can keep its label even if it enters the object bounding box later on. Previously estimated labels are pushed to the current frame in a robust way by first identifying points whose appearance lends itself to good tracking with a Lucas-Kanade feature tracker. In order to find these points, we use the STAR measure [NXC14], that allows the removal of problematic features (occlusion and aperture problem) from flow estimation. The measure is complemented with a distance transform map on the previous segmentation mask to avoid using pixels that were too close to the object borders. The motion is then computed robustly [BM04a], obtaining a reliable label propagation for those points. The obtained labels are then used as *seeds* of a per-frame iterative graph-cut segmentation method. Finally, the bounding box of the next frame is computed by robustly using the motion of labeled foreground pixels.

(a) (b) (c) (d) (e)

Figure A.1: Proposed pipeline for video object segmentation, illustrated on the *Girl* sequence. (a) Algorithm initialization: the initial bounding box is provided and accurate object segmentation inside is obtained interactively (and a second, larger box, is derived to avoid unnecessary processing). (b) Points (in red) for which STAR measure is below prescribed threshold are discarded from Lucas-Kanade flow estimation. (c) Distance transform for the foreground region, complementary distance transform of the background region, and extracted trimap. (d) Points that are initially far from the object (green), eventually enter the bounding box in subsequent frames (red). The label propagation is trusted only if the necessary conditions of the STAR measure and trimap estimation are met. (e) Segmentation of the next frame by taking propagated labels as seeds for minimization of an energy function.

A.2 Video object segmentation

Given a color video sequence $I_{0:N-1} = \{I_0, \cdots, I_{N-1}\}$, and an initial bounding box around the interest object, which is loosely drawn in the first frame of this sequence and can possibly be accompanied by user provided strokes alike [RKB04], we want to obtain a sequence of binary masks $\mathscr{M}_t$, such that $\mathscr{M}_t(\mathbf{x}) = 1$, if the pixel position x belongs to the object support at time t, and $\mathscr{M}_t(\mathbf{x}) = 0$ in the opposite case. Moreover, we want this to be solved in a frame-by-frame manner avoiding global schemes over the whole video volume. The three main principles that are to be exploited here are:

- The object of interest (the “foreground") is mostly different in appearance from the rest of the scene (the “background").
- This object is also likely to follow a motion that differs form the one of the background. This is all the more true in the long term.
- In a given frame it is possible to find, inside the object current bounding box, image elements that were initially away from the object, and thus, confidently labeled as background.

These ideas are combined with color and motion-based models within a MRF on the segmentation labels. Details are explained hereafter.

Figure A.2: Result samples on SegTrack [TFM10] and FMBS [BM10] datasets. Segmentation results obtained by the proposed method in several sequences are indicated with a green overlay (accurate segmentation mask in first image is provided to the system).

A.2.1 Motion estimation and foreground-background tracking

Given a rough delimitation between background and foreground in each frame (e.g. a bounding box), it is reasonable to assume that in most cases the pixels outside this delimitation do not belong to the object (though it is not true that insiders are only foreground). In addition to this, natural video dynamics usually display a rich interaction between different background elements and the moving object. For instance, as it can be seen in Figure A.1.d, points that evidently belong to the background in a given frame as they are far from the object, can be used as a separation hint in subsequent frames. Furthermore, once the first frame segmentation, which is usually user-supervised, is completed, the extracted foreground-background labels can be propagated through motion estimation to the following images. Also, in order to limit the influence of non-interesting zones of the image and to reduce computational cost, background reasoning is only performed in a second region of interest surrounding the actual object bounding box.

The aforementioned propagation of foreground and background labels is done through a multi-scale Lucas-Kanade optical flow method [BM04a, Bou01]. However, the final flows are computed from time t to $t-1$ adopting the integration approach of [CFC+15]. Compared to classic Euler integration, this improves the robustness to discretization bias and noise, and allows the construction of accurate point trajectories of the foreground-background pixels. This means that the pixels in the current frame *search* for their label in the previous frame. However, there are several factors (i.e. aperture problem, occlusion, border singularities) that decrease the reliability of the motion estimation, and taking into account pixels that fall inside one of these categories may affect the segmentation outcome of the current frame by pulling

incorrect labels. While the popular point tracker from [S+94] deals well with the aperture problem by selecting points that show a highly textured appearance over a window, it does not prevent from using points along the object borders that exhibit motion discontinuity. In effect, windows centered at points that belong to the object contours tend to be highly textured (as long as the appearances of the object and background are different), but do not comply with the Lucas-Kanade assumption of patch-based motion uniformity. This is a very critical issue in the segmentation problem, since it is precisely the object borders what it is sought. This is why we use a robust and better suited point-wise motion-validity measure, namely the *Spatio-Temporal Variation Ratio (STAR)* defined in [NXC14]:

$$S_t(\mathbf{x}) = \frac{\sum_{\mathcal{N}_\mathbf{x}} E_X^2 \sum_{\mathcal{N}_\mathbf{x}} E_Y^2 - (\sum_{\mathcal{N}_\mathbf{x}} E_X E_Y)^2}{||A_\mathbf{x}^T A_\mathbf{x}||_1 \sum_{\mathcal{N}_\mathbf{x}} E_t^2}, \tag{A.1}$$

where $A_\mathbf{x}$ is the intensity structure tensor over patch $\mathcal{N}_\mathbf{x}$ centered at $\mathbf{x}$, and E_X, E_Y, and E_t are the spatial and temporal intensity gradients at time t. A binary mask ψ_t is built for every frame in the video, such that $\psi_t(\mathbf{x}) = 0$ if $S_t(\mathbf{x}) < \lambda$, and $\psi_t(\mathbf{x}) = 1$ otherwise, where λ is a fixed threshold (0.5 in all experiments). See an example on the *Girl* sequence in Figure A.1.b. This simple filtering technique alleviates the need for more costly motion verification such as the forward-backward flow difference as used in [CAC+02] for a similar problem. Thus, a motion vector $\mathbf{f}_\mathbf{x} = (u_\mathbf{x}, v_\mathbf{x})$ is estimated for every point that satisfies $\psi_t(\mathbf{x}) = 1$.

However, the STAR measure does not eliminate all the points that belong to the object border (See Figure A.1.b), since not all of them show evidence of motion discontinuity. We then propose to use the segmentation mask at time $t-1$ to avoid using the points at time t that, through flow integration, are matched to pixels too close to the border in the previous frame. In order to identify those pixels, a pair of normalized distance transforms computed from $\mathcal{M}_{t-1}$ are combined as seen in Figure A.1.c. By thresholding these distance transforms, a trimap $T_{t-1}(\mathbf{x}) \in \{\text{Foreground}, \text{Background}, \text{Border}\}$ is obtained. If a point $\mathbf{x}$ at time t is matched with a point $\mathbf{y}$ at time $t-1$, the associated label propagation is considered not valid if $T_{t-1}(\mathbf{y}) = \text{Border}$.

A.2.2 MRF definition and minimization

We define an energy function that combines color and/or motion cues along with a smoothing regularizer. Its minimization provides the binary segmentation $\mathcal{M}_t$ of the current frame. This function reads:

$$E(\mathcal{M}_t) = \sum_{\mathbf{x}\in\Omega} U_\mathbf{x}(\mathcal{M}_t(\mathbf{x})) + \beta \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{N}} B_{\mathbf{x},\mathbf{y}} \delta(\mathcal{M}_t(\mathbf{x}), \mathcal{M}_t(\mathbf{y})), \tag{A.2}$$

where, $\delta(\mathscr{M}_t(\mathbf{x}),\mathscr{M}_t(\mathbf{y})) = 1$ if $\mathscr{M}_t(\mathbf{x}) \neq \mathscr{M}_t(\mathbf{y})$ and 0 otherwise. Also, Ω is the image grid and the set $\mathscr{N}$ contains all pairs of adjacent pixels (8-connectivity).

The unary and binary terms in (A.2) can be defined depending on the characteristics of the object of interest. For instance, when segmenting an object that undergoes a rigid (or close to rigid) motion, using combined optical flow and color information is a good idea. As this is usually the case, we define the binary term $B_{\mathbf{x},\mathbf{y}}$ as:

$$B_{\mathbf{x},by} = \begin{cases} \exp(-\frac{||I_\mathbf{x}-I_\mathbf{y}||^2}{2\sigma^2}), \text{if } \psi_t(\mathbf{x}) = \psi_t(\mathbf{y}) = 0 \\ \exp(\frac{-\zeta||I_\mathbf{x}-I_\mathbf{y}||^2-(1-\zeta)((u_\mathbf{x}-u_\mathbf{y})^2+(v_\mathbf{x}-v_\mathbf{y})^2)}{2\sigma^2}), \text{otherwise} \end{cases}, \tag{A.3}$$

which enforces homogeneity between neighboring pixels with similar motions and colors if the *STAR* measures are high enough, and with similar colors only in the case they are not. Conversely, this term penalizes high color and/or motion differences between neighboring pixels with the same label. Parameter ζ controls the balance between color and motion terms, and it is set empirically to 0.7. The parameter β, that weights the overall contribution of binary terms in (A.2), is set to 50. Classically, unary terms are based on color models learned for foreground and background respectively. Two *Gaussian Mixture Models* with $K = 5$ components are dedicated to this. A standard iterative Expectation-Maximization-based algorithm is used to train the color models. However, better approaches could be considered in future work, for instance, the mean-shift driven initialization of Corrigan *et al.*, [CRK08]. The unary term $U_\mathbf{x}(\mathscr{M}_t(\mathbf{x}))$ is then set to the negative log-likelihood of the best fitting Gaussian for the mixture model corresponding to the label. However, for a more generic scene, where no characteristics of the object of interest are know *a priori*, classic flow-independent regularizer as in [RKB04] can be used. In any case, as the video progresses, the tracked background-foreground pixels are used as seeds in the same sense as the user strokes are used as seeds in [RKB04, BJ01]. The minimization is done using a min-cut max-flow algorithm [BJ01].

A.2.3 Bounding-box update

The bounding box provided by the user in the first frame is tracked along the rest of the video sequence by using an off-the-shelf object tracker. For the experiments presented in this chapter, we rely on a Ransac-based affine tracker derived from the point matches inside computed object masks. However, any one of the available object trackers in the state of the art is prone to be used with our method. In particular, the structured output tracker *Struck* [HST11] is used in Chapter 3.

Precision	ours	[BWSS09]	[PF13]	[BM10]	[LKG11]	[BVD11]
Birdfall2	**160**	444	217	468	288	606
Cheetah	2708	1581	**890**	1968	905	11210
Girl	3589	5450	3895	7595	**1785**	26409
Monkey	482	1178	**284**	1434	521	12662
Parachute	223	2441	855	1113	**201**	40251

Table A.1: Comparative results on *SegTrack*. The entries present the average number of mislabeled pixels per frame. Bold: Best result. Underlined: Second best. The results for [BWSS09] are presented as reported by [TFNR12].

A.3 Experimental results

Visual results on several sequences are presented in Figure A.2. In the *Rabbit* sequence (first row, second set), for instance, it can be appreciated that our method deals satisfactorily with occlusions and propagates well the foreground labels even in difficult parts of the body of the rabbit, like the ears. On the other hand, it can be observed in the *Giraffe* sequence (third row, first set) that there are difficulties at delineating precisely very long thin structures like the neck or the legs of the animals. This can be attributed in part to the shrinking bias which is inherent to graph-cuts, together with scene-specific conditions such as occlusions between the necks of the animals.

A visual comparison with the *GrabCut* [RKB04] and *Video Snapcut* [BWSS09] methods is reported on a sequence that is affected by occlusions and motion blur (Figure A.3). It can be appreciated that the proposed algorithm offers good results with fewer user inputs than the other methods.

Finally, in Table A.1 we also show that our method is competitive by reporting comparative results on the very demanding *SegTrack* [TFM10] dataset, which contains six videos accompanied by pixel-level ground truth. The videos of this dataset contain between 21 and 71 frames. We report quantitative results on the five sequences that are most commonly used in the previous literature, *i.e.*, *Birdfall2*, *Cheetah Girl*, *Monkey*, and *Parachute* [LKG11, PF13]. In this experiment, we obtained the best result for the *Birdfall2* sequence, which is one of the most demanding sequences, since the object of interest (a falling bird) is subject to a fast motion and strong blurring. We obtained the second best results for *Girl*, *Monkey*, and *Parachute* sequences, which are involved in different difficulties like strong deformations

A.4 Discussion

We have presented a new method to perform video object segmentation that leverages single-frame graph-cut methods, by using *background-foreground* tracking. Our

Figure A.3: Comparison with alternative methods on a sequence with occlusions and motion-blur (sequence courtesy of [BWSS09]) First row: Video SnapCut [BWSS09]. Second row: Frame-by-frame GrabCut [RKB04]. Third row: Proposed method. Segmentation results are marked in red. The user interactions, if any, are marked for every frame (Blue: Background seeds, Yellow: Foreground seeds, Green: Bounding box).

algorithm uses motion estimation and reasoning based on the previously obtained output to propagate trustworthy segmentation labels, and use them as seeds in final segmentation of current frame. Experiments conducted on the challenging sequences of *SegTrack* dataset demonstrated that our method is competitive and reliable, while being simple and less demanding on user that similar methods like [BWSS09].

LIST OF PUBLICATIONS

[PRCP15] Pérez-Rúa, J.M., Crivelli, T. and Pérez, P., 2015, September. Background-foreground tracking for video object segmentation. In *Image Processing (ICIP), 2015 IEEE International Conference on* (pp. 1613-1617).

[PRCP16] Pérez-Rúa, J.M., Crivelli, T. and Pérez, P., 2016. Object-guided motion estimation. *Computer Vision and Image Understanding*, 153, pp. 88-99. Elsevier.

[PRCPB16b] Pérez-Rúa, J.M., Crivelli, T., Pérez, P. and Bouthemy, P., 2016, September. Hierarchical motion decomposition for dynamic scene parsing. In *Image Processing (ICIP), 2016 IEEE International Conference on* (pp. 3952-3956).

[PRCPB16a] Pérez-Rúa, J.M., Crivelli, T., Pérez, P. and Bouthemy, P., 2016, September. Discovering motion hierarchies via tree-structured coding of trajectories. In *27th British Machine Vision Conference (BMVC 2016)* (pp 106.1-106.12).

[PRCBP16] Pérez-Rúa, J.M., Crivelli, T., Bouthemy, P. and Pérez, P., 2016. Determining occlusions from space and time image reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1382-1391).

[PRMTP17] Miksik, O., Pérez-Rúa, J.M., Torr, P.H. and Pérez, P., 2017. ROAM: a Rich Object Appearance Model with Application to Rotoscoping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1382-1391).

[PRCBP17] Pérez-Rúa, J.M., Crivelli, T., Bouthemy, P. and Pérez, P., 2017. Trimap tracking: closing the gap between rotoscoping and alpha matting. In submission.

[PRCBP18] Pérez-Rúa, J.M., Crivelli, T., Bouthemy, P. and Pérez, P., 2018. Learning how to be robust: Deep polynomial regression. In submission.

BIBLIOGRAPHY

[AAPS16] Yağiz Aksoy, Tunç Ozan Aydin, Marc Pollefeys, and Aljoša Smolić. Interactive high-quality green-screen keying via color unmixing. *Transactions on Graphics*, 35(5):152, 2016.

[AB06] Jung-Ho Ahn and Hyeran Byun. Accurate foreground extraction using graph cut with trimap estimation. *Advances in Image and Video Technology: First Pacific Rim Symposium, PSIVT*, 2006.

[ACC09] Pierre Allain, Nicolas Courty, and Thomas Corpetti. Crowd flow characterization with optimal control theory. In *Asian Conference on Computer Vision*, pages 279–290. Springer, 2009.

[ACC12] Pierre Allain, Nicolas Courty, and Thomas Corpetti. Agoraset: a dataset for crowd video analysis. In *International Conference on Pattern Recognition*, pages 1–6, 2012.

[ADPS07] Luis Alvarez, Rachid Deriche, Théo Papadopoulo, and Javier Sánchez. Symmetrical dense optical flow estimation with occlusions detection. *International Journal of Computer Vision*, 75(3):371–385, 2007.

[AEB06] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[AF05] Nicholas Apostoloff and Andrew Fitzgibbon. Learning spatiotemporal t-junctions for occlusion detection. In *Computer Vision Pattern Recognition*, 2005.

[AHSS04] Aseem Agarwala, Aaron Hertzmann, David H Salesin, and Steven M Seitz. Keyframe-based tracking for rotoscoping and animation. *Transactions on Graphics*, 23(3):584–591, 2004.

[Ana89] Padmanabhan Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989.

[APB07] Christopher J Armstrong, Brian L Price, and William A Barrett. Interactive segmentation of image volumes with live surface. *Computers & Graphics*, 31(2):212–229, 2007.

[ARS12] Alper Ayvaci, Michalis Raptis, and Stefano Soatto. Sparse occlusion detection with optical flow. *International Journal of Computer Vision*, 97(3):322–338, 2012.

[AS10] Saad Ali and Mubarak Shah. Human action recognition in videos using kinematic features and multiple instance learning. *Transactions on Pattern Analysis and Machine Intelligence*, 32(2):288–303, 2010.

[ASS+12] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[AWJ90] Amir A Amini, Terry E Weymouth, and Ramesh C Jain. Using dynamic programming for solving variational problems in vision. *Transactions on Pattern Analysis and Machine Intelligence*, 1990.

[BA93] Michael J Black and P Anandan. A framework for the robust estimation of optical flow. In *International Conference on Computer Vision*, pages 231–236. IEEE, 1993.

[BA96] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[BAHH92] James Bergen, Patrick Anandan, Keith Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, pages 237–252. Springer, 1992.

[BAS07] Rami Ben-Ari and Nir Sochen. Variational stereo vision with sharp discontinuities and occlusion handling. In *International Conference on Computer Vision*, 2007.

[BBH03] Myron Z Brown, Darius Burschka, and Gregory D Hager. Advances in computational stereo. *Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993–1008, 2003.

[BBM09] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. In *Computer Vision Pattern Recognition*, pages 41–48. IEEE, 2009.

[BBP67] Horace B Barlow, Colin Blakemore, and John D Pettigrew. The neural mechanism of binocular depth discrimination. *The Journal of Physiology*, 193(2):327, 1967.

[BBPW04] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision*, pages 25–36. Springer, 2004.

[BGG99] Patrick Bouthemy, Marc Gelgon, and Fabrice Ganansia. A unified approach to shot change detection and camera motion characterization. *Transactions on Circuits and Systems for Video Technology*, 9(7):1030–1044, 1999.

[BGLC12] Coloma Ballester, Lluis Garrido, Vanel Lazcano, and Vicent Caselles. A TV-L1 optical flow method with occlusion detection. *Pattern Recognition*, 7476:31–40, 2012.

[BI00] Andrew Blake and Michael Isard. *Active contours*. Springer, 2000.

[BJ96] Michael J Black and Allan D Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, 1996.

[BJ98] Michael J Black and Allan D Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[BJ01] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *International Conference on Computer Vision*, volume 1, pages 105–112. IEEE, 2001.

[BK10] Gary Baugh and Anil Kokaram. Semi-automatic motion based segmentation using long term motion trajectories. In *International Conference on Image Processing*. IEEE, 2010.

[BM04a] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[BM04b] Selim Benhimane and Ezio Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *International Conference on Intelligent Robots and Systems*, volume 1, pages 943–948. IEEE, 2004.

[BM10] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European Conference on Computer Vision*. 2010.

[BM11] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011.

[Boo89] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, 1989.

[Bou01] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5:1–10, 2001.

[BP08] Stanley T Birchfield and Shrinivas J Pundlik. Joint tracking of features and edges. In *Computer Vision Pattern Recognition*, pages 1–6. IEEE, 2008.

[Bra11] Benjamin Bratt. *Rotoscoping: Techniques and tools for the Aspiring Artist*. Taylor & Francis, 2011.

[BS07] Xue Bai and Guillermo Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[BSFG09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *Transactions on Graphics*, 28–37(3):24, 2009.

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

[Bur52] Luke Burke. On the tunnel effect. *Quarterly Journal of Experimental Psychology*, 4(3):121–138, 1952.

[BVD11] Olivier Barnich and Marc Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *Transactions on Image Processing*, 20:1709, 2011.

[BVS17] Christian Bailer, Kiran Varanasi, and Didier Stricker. Cnn-based patch matching for optical flow with thresholded hinge embedding loss. In *Computer Vision Pattern Recognition*, 2017.

[BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[BWS11] Xue Bai, Jue Wang, and David Simons. Towards temporally-coherent video matting. In *MIRAGE*, volume 11, pages 63–74, 2011.

[BWSB12a] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, 2012.

[BWSB12b] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*. 2012.

[BWSS09] Xue Bai, Jue Wang, David Simons, and Guillermo Sapiro. Video snapcut: robust video object cutout using localized classifiers. In *Transactions on Graphics*, volume 28, page 70. ACM, 2009.

[BY95] Michael J Black and Yaser Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *International Conference on Computer Vision*, pages 374–381. IEEE, 1995.

[BYB09] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *Computer Vision Pattern Recognition*, pages 983–990. IEEE, 2009.

[CAC+02] Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David H Salesin, and Richard Szeliski. Video matting of complex scenes. *Transactions on Graphics*, 2002.

[Cal50] ES Calvert. Visual aids for landing in bad visibility with particular reference to the transition from instrument to visual flight. *Lighting Research and Technology*, 15(6 IEStrans):183–219, 1950.

[Cas13] Dominic Case. *Film technology in post production*. Taylor & Francis, 2013.

[CB99] Gabriella Csurka and Patrick Bouthemy. Direct identification of moving objects and background from 2d motion models. In *International Conference on Computer Vision*, 1999.

[CC01] Armel Crétual and François Chaumette. Visual servoing based on image motion. *The International Journal of Robotics Research*, 20(11):857–877, 2001.

[CCB98] Armel Crétual, François Chaumette, and Patrick Bouthemy. Complex object tracking by visual servoing based on 2d image motion. In *International Conference on Pattern Recognition*, volume 2, pages 1251–1254. IEEE, 1998.

[CCR+12] Tomás Crivelli, Pierre-Henri Conze, Philippe Robert, Matthieu Fradet, and Patrick Pérez. Multi-step flow fusion: Towards accurate and dense correspondences in long video shots. In *British Machine Vision Conference*. British Machine Vision Association, 2012.

[CCRP12] Tomas Crivelli, Pierre-Henri Conze, Philippe Robert, and Patrick Pérez. From optical flow to dense long term correspondences. In *International Conference on Image Processing*, pages 61–64. IEEE, 2012.

[CCSS01] Yung-Yu Chuang, Brian Curless, David H Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *Computer Vision Pattern Recognition*, 2001.

[CFC+15] Tomas Crivelli, Matthieu Fradet, Pierre-Henri Conze, Philippe Robert, and Patrick Pérez. Robust optical flow integration. *Transactions on Image Processing*, 24(1):484–498, 2015.

[CR09] Anil M Cheriyadat and Richard J Radke. Non-negative matrix factorization of partial track data for motion segmentation. In *Computer Vision Pattern Recognition*, 2009.

[CRK08] David Corrigan, S Robinson, and A Kokaram. Video matting using motion extended grabcut. 2008.

[CS05] Daniel Cremers and Stefano Soatto. Motion competition: A variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3):249–265, 2005.

[DBPP00] Fabien Dekeyser, Patrick Bouthemy, Patrick Pérez, and Étienne Payot. Super-resolution from noisy image sequences exploiting a 2d parametric motion model. In *International Conference on Pattern Recognition*, volume 3, pages 350–353. IEEE, 2000.

[DFI+15] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *International Conference on Computer Vision*, pages 2758–2766, 2015.

[DM00] Douglas Decarlo and Dimitris Metaxas. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 38(2):99–127, 2000.

[DOIB12] Andrew Delong, Anton Osokin, Hossam N Isack, and Yuri Boykov. Fast approximate energy minimization with label costs. *International Journal of Computer Vision*, 96(1):1–27, 2012.

[DZ13] Piotr Dollar and Larry Zitnick. Sketch tokens: A learned mid-level representation for contour and object detection. In *Computer Vision Pattern Recognition*, 2013.

[EGV+15] Mikhail Erofeev, Yury Gitman, Dmitriy Vatolin, Alexey Fedorov, and Jue Wang. Perceptually motivated benchmark for video matting. In *British Machine Vision Conference*. BMVA Press, September 2015.

[Enk88] Wilfried Enkelmann. Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences. *Computer Vision, Graphics, and Image Processing*, 43(2):150–177, 1988.

[ES15] V Estellers and S Soatto. Detecting occlusions as an inverse problem. *Journal of Mathematical Imaging and Vision*, pages 1–18, 2015.

[EV09] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *Computer Vision Pattern Recognition*, 2009.

[FAFM15] Katerina Fragkiadaki, Pablo Arbelaez, Panna Felsen, and Jitendra Malik. Learning to segment moving objects in videos. In *Computer Vision Pattern Recognition*, 2015.

[Far00] Gunnar Farnebäck. Fast and accurate motion estimation using orientation tensors and parametric motion models. In *International Conference on Pattern Recognition*, volume 1, pages 135–139. IEEE, 2000.

[Far01] Gunnar Farnebäck. Very high accuracy velocity estimation using orientation tensors, parametric motion, and simultaneous segmentation of the motion field. In *International Conference on Computer Vision*, volume 1, pages 171–177. IEEE, 2001.

[Far03] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer, 2003.

[FBK15a] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Aggregation of local parametric candidates with exemplar-based occlusion handling for optical flow. *Computer Vision and Image Understanding*, 145:1–182, 2015.

[FBK15b] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Optical flow modeling and computation: a survey. *Computer Vision and Image Understanding*, 134:1–21, 2015.

[FBK16a] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Aggregation of local parametric candidates with exemplar-based occlusion handling for optical flow. *Computer Vision and Image Understanding*, 145:81–94, 2016.

[FBK16b] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. A variational aggregation framework for patch-based optical flow estimation. *Journal of Mathematical Imaging and Vision*, 56(2):280–299, 2016.

[FCNL13] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013.

[FEF+17] Damien Fourure, Rémi Emonet, Elisa Fromont, Damien Muselet, Natalia Neverova, Alain Trémeau, and Christian Wolf. Multi-task, multi-domain learning: application to semantic segmentation and pose regression. *Neurocomputing*, 251:68–80, 2017.

[FGMR10] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[FH06] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

[FI14] Alon Faktor and Michal Irani. Video segmentation by non-local consensus voting. In *British Machine Vision Conference*, 2014.

[Fle17] Max Fleischer. Method of producing moving-picture cartoons., October 9 1917. US Patent 1,242,674.

[FRP09] Matthieu Fradet, Philippe Robert, and Patrick Pérez. Clustering point trajectories with various life-spans. In *CVMP*, 2009.

[FSW12] Jialue Fan, Xiaohui Shen, and Ying Wu. Scribble tracker: a matting-based approach for robust tracking. *Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[FZ08] Adel Fakih and John Zelek. Structure from motion: Combining features correspondences and optical flow. In *International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[FZL+15] Qingnan Fan, Fan Zhong, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. JumpCut: Non-successive mask transfer and interpolation for video cutout. *Transactions on Graphics*, 2015.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision Pattern Recognition*, 2014.

[GGF+80] Alton L Gilbert, Michael K Giles, Gerald M Flachs, Robert B Rogers, and U Yee Hsun. A real-time video tracking system. *Transactions on Pattern Analysis and Machine Intelligence*, (1):47–56, 1980.

[Gib50] James J Gibson. *The perception of the visual world.* Houghton Mifflin, 1950.

[Gib58] James J Gibson. Visually controlled locomotion and visual orientation in animals. *British journal of psychology*, 49(3):182–194, 1958.

[Gib70] James J Gibson. On theories for visual space perception. *Scandinavian Journal of Psychology*, 11(1):75–79, 1970.

[Gib77] James J Gibson. On the analysis of change in the optic array. *Scandinavian Journal of Psychology*, 18(1):161–163, 1977.

[GKHE10] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *Computer Vision Pattern Recognition*, 2010.

[GKRW69] James J Gibson, George A Kaplan, Horace N Reynolds, and Kirk Wheeler. The change from visible to invisible. *Attention, Perception, & Psychophysics*, 5(2):113–116, 1969.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[GMO+10] Frederic García, Bruno Mirbach, Bjorn Ottersten, Frédéric Grandidier, and Angel Cuesta. Pixel weighted average strategy for depth sensor data fusion. In *International Conference on Image Processing*, pages 2805–2808. IEEE, 2010.

[GO10] Eduardo SL Gastal and Manuel M Oliveira. Shared sampling for real-time alpha matting. In *Computer Graphics Forum*, 2010.

[GSAW05] Leo Grady, Thomas Schiwietz, Shmuel Aharon, and Rüdiger Westermann. Random walks for interactive alpha-matting. In *VIIP*, 2005.

[GTJ15] Samuel J Gershman, Joshua B Tenenbaum, and Frank Jäkel. Discovering hierarchical motion structure. *Vision Research*, 2015.

[HAP94] Steve Hsu, P Anandan, and Shmuel Peleg. Accurate computation of optical flow by using layered motion representations. In *International Conference on Pattern Recognition*, volume 1, pages 743–746. IEEE, 1994.

[HB93] Fabrice Heitz and Patrick Bouthemy. Multimodal estimation of discontinuous optical flow using markov random fields. *Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1217–1232, 1993.

[HCMB15] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Kernelized correlation filters. *Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[HLWvdM16] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[HMAB11] Ahmad Humayun, Oisin Mac Aodha, and Gabriel J Brostow. Learning to find occlusion regions. In *Computer Vision Pattern Recognition*, 2011.

[HN99] Michael Haag and Hans-Hellmut Nagel. Combination of edge element and optical flow estimates for 3d-model-based vehicle tracking in traffic image sequences. *International Journal of Computer Vision*, 35(3):295–319, 1999.

[HPB94] Fabrice Heitz, Patrick Pérez, and Patrick Bouthemy. Multiscale minimization of global energy functions in some visual recovery problems. *CVGIP: Image Understanding*, 59(1):125–134, 1994.

[HRR+11] Kaiming He, Christoph Rhemann, Carsten Rother, Xiaoou Tang, and Jian Sun. A global sampling method for alpha matting. In *Computer Vision Pattern Recognition*, 2011.

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

[HST11] Sam Hare, Amir Saffari, and Philip HS Torr. Struck: Structured output tracking with kernels. In *International Conference on Computer Vision*, pages 263–270. IEEE, 2011.

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

[HW77] Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977.

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

[HZM11] Junzhou Huang, Tong Zhang, and Dimitris Metaxas. Learning with structured sparsity. *The Journal of Machine Learning Research*, 12:3371–3412, 2011.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision Pattern Recognition*, pages 770–778, 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[IK08] Serdar Ince and Janusz Konrad. Occlusion-aware optical flow estimation. *Transactions on Image Processing*, 17(8):1443–1451, 2008.

[IMS+17] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *Computer Vision Pattern Recognition*, 2017.

[Jar08] Kylie Jarrett. Beyond broadcast yourself: the future of youtube. *Media International Australia*, 126(1):132–144, 2008.

[JBJ96] Shanon X Ju, Michael J Black, and Allan D Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *Computer Vision Pattern Recognition*, 1996.

[JFN12] Natan Jacobson, Yoav Freund, and Truong Q Nguyen. An online learning approach to occlusion boundary detection. *Transactions on Image Processing*, 21(1):252–261, 2012.

[JHD16] J Yu Jason, Adam W Harley, and Konstantinos G Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *ECCV Workshop*, pages 3–10. Springer, 2016.

[JLD11] Zhuolin Jiang, Zhe Lin, and Larry S Davis. Learning a discriminative dictionary for sparse coding via label consistent k-svd. In *CVPR, Colorado Springs*, 2011.

[JMOB10] Rodolphe Jenatton, Julien Mairal, Guillaume Obozinki, and Francis Bach. Proximal methods for sparse hierarchical dictionary learning. In *International Conference on Machine Learning*, 2010.

[Joh50] Gunnar Johansson. Configurations in the perception of velocity. *Acta Psychologica*, 7:25–79, 1950.

[Joh70] Gunnar Johansson. On theories for visual space perception. *Scandinavian Journal of Psychology*, 11(1):67–74, 1970.

[Joh73] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Attention, Perception, & Psychophysics*, 14(2):201–211, 1973.

[JS09] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In *Conference on Neural Information Processing Systems*, pages 769–776, 2009.

[JVGJ+14] Mihir Jain, Jan Van Gemert, Hervé Jégou, Patrick Bouthemy, and Cees GM Snoek. Action localization with tubelets from motion. In *Computer Vision Pattern Recognition*, 2014.

[JvGJ+17] Mihir Jain, Jan van Gemert, Hervé Jégou, Patrick Bouthemy, and Cees GM Snoek. Tubelets: Unsupervised action proposals from spatiotemporal super-voxels. *International Journal of Computer Vision*, pages 1–25, 2017.

[KAB15] Margret Keuper, Bjoern Andres, and Thomas Brox. Motion trajectory segmentation via minimum cost multicuts. In *International Conference on Computer Vision*, 2015.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KCLU07] Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. 26(3):96, 2007.

[KK11] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *Conference on Neural Information Processing Systems*, 2011.

[KK13] Philipp Krähenbühl and Vladlen Koltun. Parameter learning and convergent inference for dense random fields. In *International Conference on Machine Learning*, 2013.

[KPL+14] Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiri Matas, Luka Čehovin, Georg Nebehay, Tomáš Vojíř, Gustavo Fernandez, Alan Lukežič, Aleksandar Dimitriev, et al. The visual object tracking VOT2014 challenge results. In *ECCV Workshop*, pages 191–217. Springer, 2014.

[KRFB06] Erum Arif Khan, Erik Reinhard, Roland W Fleming, and Heinrich H Bülthoff. Image-based material editing. *Transactions on Graphics*, 2006.

[KS83] Philip J Kellman and Elizabeth S Spelke. Perception of partly occluded objects in infancy. *Cognitive Psychology*, 15(4):483–524, 1983.

[KSA+01] Itaru Kitahara, Hideo Saito, Shinji Akimichi, Tooru Ono, Yuichi Ohta, and Takeo Kanade. Large-scale virtualized reality. *CVPR Workshop*, 2001.

[KSC01] Sing Bing Kang, Richard Szeliski, and Jinxiang Chai. Handling occlusions in dense multi-view stereo. In *Computer Vision Pattern Recognition*, 2001.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems*, pages 1097–1105, 2012.

[KT07] Pushmeet Kohli and Philip HS Torr. Dynamic graph cuts for efficient inference in Markov random fields. *Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[KVK16] Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature space optimization for semantic video segmentation. In *Computer Vision Pattern Recognition*, pages 3168–3175, 2016.

[KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1988.

[KZ01] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *International Conference on Computer Vision*, 2001.

[LBSW16] Yao Lu, Xue Bai, Linda Shapiro, and Jue Wang. Coherent parametric contours for interactive video object segmentation. In *Computer Vision Pattern Recognition*, 2016.

[LCT13] Dingzeyu Li, Qifeng Chen, and Chi-Keung Tang. Motion-aware knn laplacian for video matting. In *International Conference on Computer Vision*, 2013.

[Lev06] Marc Levoy. Light fields and computational imaging. *Computer*, 39(8):46–55, 2006.

[LHP80] H Christopher Longuet-Higgins and Kvetoslav Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London B: Biological Sciences*, 208(1173):385–397, 1980.

[LHS+13] Xi Li, Weiming Hu, Chunhua Shen, Zhongfei Zhang, Anthony Dick, and Anton Van Den Hengel. A survey of appearance models in visual object tracking. *Transactions on Intelligent Systems and Technology*, 4(4):58, 2013.

[LK81] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artifical Intelligence*, pages 674–679, 1981.

[LKG11] Yong Jae Lee, Jaechul Kim, and Kristen Grauman. Key-segments for video object segmentation. In *International Conference on Computer Vision*, 2011.

[LLW08] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[LRRB10] Victor Lempitsky, Carsten Rother, Stefan Roth, and Andrew Blake. Fusion moves for markov random field optimization. *Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1392–1405, 2010.

[LVS+16] Wenbin Li, Fabio Viola, Jonathan Starck, Gabriel J Brostow, and Neill DF Campbell. Roto++: Accelerating professional rotoscoping using shape manifolds. *Transactions on Graphics*, 35(4):62, 2016.

[LY05] Jongwoo Lim and Ming-Hsuan Yang. A direct method for modeling non-rigid motion with thin plate spline. In *Computer Vision Pattern Recognition*, volume 1, pages 1196–1202. IEEE, 2005.

[LYT+08] Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T Freeman. Sift flow: Dense correspondence across different scenes. In *European Conference on Computer Vision*, pages 28–42. Springer, 2008.

[MB94] François G Meyer and Patrick Bouthemy. Region-based tracking using affine motion models in long image sequences. *CVGIP: Image understanding*, 60(2):119–140, 1994.

[MCUP04] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 2004.

[MD12] Quanyi Mo and Bruce A Draper. Semi-nonnegative matrix factorization for motion segmentation with missing data. In *European Conference on Computer Vision*, 2012.

[MG15] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Computer Vision Pattern Recognition*, 2015.

[MHYY15] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *International Conference on Computer Vision*, pages 3074–3082, 2015.

[MKT98] Ivana Mikic, Slawomir Krucinski, and James D Thomas. Segmentation and tracking in echocardiographic sequences: Active contours guided by optical flow estimates. *IEEE Transactions on Medical Imaging*, 17(2):274–284, 1998.

[MP76] David Marr and Tomaso Poggio. *Cooperative computation of stereo disparity*. Springer, 1976.

[MP98] Etienne Mémin and Patrick Pérez. Dense estimation and object-based segmentation of the optical flow with robust techniques. *Transactions on Image Processing*, 7(5):703–719, 1998.

[MPWSH16a] Nicolas Märki, Federico Perazzi, Oliver Wang, and Alexander Sorkine-Hornung. Bilateral space video segmentation. In *Computer Vision Pattern Recognition*, pages 743–751, 2016.

[MPWSH16b] Nicolas Märki, Federico Perazzi, Oliver Wang, and Alexander Sorkine-Hornung. Bilateral space video segmentation. In *Computer Vision Pattern Recognition*, 2016.

[MRFS12] Fabrice Mériaudeau, Rindra Rantoson, David Fofi, and Christophe Stolz. Review and comparison of non-conventional imaging systems for three-dimensional digitization of transparent objects. *Journal of Electronic Imaging*, 21(2):021105–1, 2012.

[MT09] Brendan Morris and Mohan Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In *Computer Vision Pattern Recognition*, pages 312–319. IEEE, 2009.

[NL74] K Nakayama and JM Loomis. Optical velocity patterns, velocity-sensitive neurons, and space perception: a hypothesis. *Perception*, 3(1):63–80, 1974.

[NXC14] Yan Niu, Zhiwen Xu, and Xiangjiu Che. Dynamically removing false features in pyramidal lucas-kanade registration. *Transactions on Image Processing*, 23(8):3535–3544, 2014.

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[OB95] Jean-Marc Odobez and Patrick Bouthemy. Robust multiresolution estimation of parametric motion models. *Journal of Visual Communication and Image Representation*, 6(4):348–365, 1995.

[OB97] JM Odobez and P Bouthemy. Separation of moving regions from background in an image sequence acquired with a mobile camera. In *Video Data Compression for Multimedia Computing*, pages 283–311. Springer, 1997.

[OB98] Jean-Marc Odobez and Patrick Bouthemy. Direct incremental model-based image motion segmentation for video analysis. *Signal Processing*, 66(2):143–155, 1998.

[OB12] Peter Ochs and Thomas Brox. Higher order motion models and spectral clustering. In *Computer Vision Pattern Recognition*, 2012.

[ODO16] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[OGP06] Jean-Marc Odobez and Daniel Gatica-Perez. Motion likelihood and proposal modeling in model-based stochastic tracking. *Transactions on Image Processing*, 2006.

[OMB14] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1187–1200, 2014.

[PBG01] Patrick Pérez, Andrew Blake, and Michel Gangnet. Jetstream: Probabilistic contour extraction with particles. In *International Conference on Computer Vision*, 2001.

[PF13] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *International Conference on Computer Vision*, pages 1777–1784. IEEE, 2013.

[PH07] Jiyan Pan and Bo Hu. Robust occlusion handling in object tracking. In *Computer Vision Pattern Recognition*, 2007.

[PLW11] Youngmin Park, Vincent Lepetit, and Woontack Woo. Extended keyframe detection with stable tracking for multiple 3d object tracking. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1728–1735, 2011.

[PPTM+16] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision Pattern Recognition*, 2016.

[PRBB17] Juan-Manuel Pérez-Rúa, Antoine Basset, and Patrick Bouthemy. Detection and localization of anomalous motion in video sequences from local histograms of labeled affine flows. *Frontiers in ICT*, 4:10, 2017.

[PRCBP16] Juan-Manuel Pérez-Rúa, Tomas Crivelli, Patrick Bouthemy, and Patrick Pérez. Determining occlusions from space and time image reconstructions. In *Computer Vision Pattern Recognition*, 2016.

[PRCBP17] Juan-Manuel Pérez-Rúa, Tomas Crivelli, Patrick Bouthemy, and Patrick Pérez. Trimap tracking: closing the gap between rotoscoping and alpha matting. *In submission*, 2017.

[PRCBP18] Juan-Manuel Perez-Rua, Tomas Crivelli, Patrick Bouthemy, and Patrick Perez. Learning how to be robust: Deep polynomial regression. *arXiv preprint arXiv:1804.06504*, 2018.

[PRCP15] Juan-Manuel Pérez-Rúa, Tomás Crivelli, and Patrick Pérez. Background-foreground tracking for video object segmentation. In *International Conference on Image Processing*, pages 1613–1617. IEEE, 2015.

[PRCP16] Juan-Manuel Pérez-Rúa, Tomas Crivelli, and Patrick Pérez. Object-guided motion estimation. *Computer Vision and Image Understanding*, 153:88–99, 2016.

[PRCPB16a] Juan-Manuel Pérez-Rúa, Tomas Crivelli, Patrick Pérez, and Patrick Bouthemy. Discovering motion hierarchies via tree-structured coding of trajectories. In *British Machine Vision Conference*, 2016.

[PRCPB16b] Juan-Manuel Perez-Rua, Tomas Crivelli, Patrick Perez, and Patrick Bouthemy. Hierarchical motion decomposition for dynamic scene parsing. In *International Conference on Image Processing*. IEEE, 2016.

[PRMTP17] Juan-Manuel Pérez-Rúa, Ondrej Miksik, Philip HS Torr, and Patrick Pérez. Roam: a rich object appearance model with application to rotoscoping. In *Computer Vision Pattern Recognition*. IEEE, 2017.

[PSK01] HyunWook Park, Todd Schoepflin, and Yongmin Kim. Active contour model with gradient directional information: Directional snake. *Transactions on Circuits and Systems for Video Technology*, 11(2):252–256, 2001.

[PTK85] Tomaso Poggio, Vincent Torre, and Christof Koch. Computational vision and regularization theory. *Nature*, 317(6035):314–319, 1985.

[PVGPO94] Marc Proesmans, Luc Van Gool, Eric Pauwels, and André Oosterlinck. Determination of optical flow and its discontinuities using non-linear diffusion. In *European Conference on Computer Vision*. 1994.

[RAKRF08] Alex Rav-Acha, Pushmeet Kohli, Carsten Rother, and Andrew Fitzgibbon. Unwrap mosaics: A new representation for video editing. *Transactions on Graphics*, 2008.

[RAS17] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. Convolutional neural network architecture for geometric matching. In *Computer Vision Pattern Recognition*. IEEE, 2017.

[RHE+15] S Hussain Raza, Ahmad Humayun, Irfan Essa, Matthias Grundmann, and David Anderson. Finding temporally consistent occlusion boundaries in videos using geometric context. In *Winter Conference on Applications of Computer Vision*, 2015.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *Transactions on Graphics*, volume 23, pages 309–314. ACM, 2004.

[RLF12] Michael Rubinstein, Ce Liu, and William T Freeman. Towards longer long-range motion trajectories. In *British Machine Vision Conference*, 2012.

[RRG08] Christoph Rhemann, Carsten Rother, and Margrit Gelautz. Improving color modeling for alpha matting. In *British Machine Vision Conference*, 2008.

[RTVM08] Shankar R Rao, Roberto Tron, René Vidal, and Yi Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *Computer Vision Pattern Recognition*, 2008.

[RWHS15a] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Deepmatching: Hierarchical deformable dense matching. *International Journal of Computer Vision*, 2015.

[RWHS15b] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Computer Vision Pattern Recognition*, 2015.

[S+94] Jianbo Shi et al. Good features to track. In *Computer Vision Pattern Recognition*, pages 593–600. IEEE, 1994.

[SBK10a] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European Conference on Computer Vision*, pages 438–451. Springer, 2010.

[SBK10b] Narayann Sundaram, Tomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *eccv*, 2010.

[SBM+11] Patrik Sundberg, Thomas Brox, Michael Maire, Pablo Arbeláez, and Jitendra Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *Computer Vision Pattern Recognition*, 2011.

[SBMR09] Mehmet Emre Sargin, Luca Bertelli, Bangalore S Manjunath, and Kenneth Rose. Probabilistic occlusion boundary detection on spatio-temporal lattices. In *International Conference on Computer Vision*, 2009.

[SCC+14] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014.

[SCD+06] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer Vision Pattern Recognition*, volume 1, pages 519–528. IEEE, 2006.

[SDC04] Paul Smith, Tom Drummond, and Roberto Cipolla. Layered motion segmentation and depth ordering by tracking edges. *Transactions on Pattern Analysis and Machine Intelligence*, 26(4):479–494, 2004.

[SES12] Tobias Senst, Volker Eiselein, and Thomas Sikora. Robust local optical flow for feature tracking. *Transactions on Circuits and Systems for Video Technology*, 22(9):1377–1387, 2012.

[SFVG04] Christoph Strecha, Rik Fransens, and Luc Van Gool. A probabilistic approach to large displacement optical flow and occlusion detection. In *Statistical methods in video processing*, pages 71–82. 2004.

[SG99] Chris Stauffer and Eric Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision Pattern Recognition*, 1999.

[SH09] Andrew N Stein and Martial Hebert. Occlusion boundaries from motion: Low-level detection and mid-level reasoning. *International Journal of Computer Vision*, 82(3):325–357, 2009.

[SLKS05] Jian Sun, Yin Li, Sing Bing Kang, and Heung-Yeung Shum. Symmetric stereo matching for occlusion handling. In *Computer Vision Pattern Recognition*, 2005.

[SLP14] Deqing Sun, Ce Liu, and Hanspeter Pfister. Local layering for joint motion estimation and occlusion detection. In *Computer Vision Pattern Recognition*, pages 1098–1105, 2014.

[SM82] Robert J Schalkoff and Eugene S Mcvey. A model and tracking algorithm for a class of video targets. *Transactions on Pattern Analysis and Machine Intelligence*, (1):2–10, 1982.

[SNSB15] Naveen Shankar Nagaraja, Frank R Schmidt, and Thomas Brox. Video segmentation with just a few strokes. In *International Conference on Computer Vision*, 2015.

[SRB10] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Computer Vision Pattern Recognition*, pages 2432–2439. IEEE, 2010.

[SRT93] Mubarak Shah, Krishnan Rangarajan, and P-S Tsai. Motion trajectories. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(4):1138–1150, 1993.

[SS03] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision Pattern Recognition*, volume 1, pages I–I. IEEE, 2003.

[SSB10] Deqing Sun, Erik B Sudderth, and Michael J Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *Conference on Neural Information Processing Systems*, 2010.

[SSB12] Deqing Sun, Erik B Sudderth, and Michael J Black. Layered segmentation and optical flow estimation over time. In *Computer Vision Pattern Recognition*, 2012.

[SSP15] Deqing Sun, Erik B Sudderth, and Hanspeter Pfister. Layered rgbd scene flow estimation. In *Computer Vision Pattern Recognition*, pages 548–556, 2015.

[SYLK17] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. *arXiv preprint arXiv:1709.02371*, 2017.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Tan82] Gregory Y Tang. A discrete version of Green's theorem. *Transactions on Pattern Analysis and Machine Intelligence*, 1982.

[TBKP12] Michael Tao, Jiamin Bai, Pushmeet Kohli, and Sylvain Paris. Simpleflow: A non-iterative, sublinear optical flow algorithm. In *Computer Graphics Forum*, volume 31, pages 345–353, 2012.

[TFM10] David Tsai, Matthew Flagg, and James M.Rehg. Motion coherent tracking with multi-label mrf optimization. *British Machine Vision Conference*, 2010.

[TFNR12] David Tsai, Matthew Flagg, Atsushi Nakazawa, and James M Rehg. Motion coherent tracking using multi-label mrf optimization. *International Journal of Computer Vision*, 100(2):190–202, 2012.

[TFTR98] Tiziano Tommasini, Andrea Fusiello, Emanuele Trucco, and Vito Roberto. Making good features track better. In *Computer Vision Pattern Recognition*, pages 178–183. IEEE, 1998.

[TG07] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Transactions on Information Theory*, 53(12):4655–4666, 2007.

[TK91] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. *Technical Report*, 1991.

[TM98] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision*, 1998.

[TNL14] Joseph Tighe, Marc Niethammer, and Svetlana Lazebnik. Scene parsing with object instances and occlusion ordering. In *Computer Vision Pattern Recognition*, 2014.

[TV07] Roberto Tron and René Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Computer Vision Pattern Recognition*, 2007.

[TYB16] Yi-Hsuan Tsai, Ming-Hsuan Yang, and Michael J. Black. Video segmentation via object flow. In *Computer Vision Pattern Recognition*, 2016.

[TZTV16] James Thewlis, Shuai Zheng, Philip HS Torr, and Andrea Vedaldi. Fully-trainable deep matching. *British Machine Vision Conference*, 2016.

[UIM08] Hirofumi Uemura, Seiji Ishikawa, and Krystian Mikolajczyk. Feature tracking and motion compensation for action recognition. In *British Machine Vision Conference*, pages 1–10, 2008.

[VH04] René Vidal and Richard Hartley. Motion segmentation with missing data using powerfactorization and GPCA. In *Computer Vision Pattern Recognition*, 2004.

[VLL+10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[VM04] René Vidal and Yi Ma. A unified algebraic approach to 2-D and 3-D motion segmentation. In *European Conference on Computer Vision*, 2004.

[VP89] Alessandro Verri and Tomaso Poggio. Motion field and optical flow: Qualitative properties. *Transactions on Pattern Analysis and Machine Intelligence*, 11(5):490–498, 1989.

[WA93] John YA Wang and Edward H Adelson. Layered representation for motion analysis. In *Computer Vision Pattern Recognition*, pages 361–366. IEEE, 1993.

[WC+08] Jue Wang, Michael F Cohen, et al. Image and video matting: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(2):97–175, 2008.

[WKSL11] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Computer Vision Pattern Recognition*, pages 3169–3176. IEEE, 2011.

[WLY13] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Computer Vision Pattern Recognition*, pages 2411–2418, 2013.

[WM95] Joseph Weber and Jitendra Malik. Robust computation of optical flow in a multi-scale differential framework. *International Journal of Computer Vision*, 14(1):67–81, 1995.

[WM08] Yang Wang and Greg Mori. Multiple tree models for occlusion and spatial constraints in human pose estimation. In *European Conference on Computer Vision*. 2008.

[WMS10] Shandong Wu, Brian E Moore, and Mubarak Shah. Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. In *Computer Vision Pattern Recognition*, pages 2054–2060. IEEE, 2010.

[WNL08] Bo Wu, Ram Nevatia, and Yuan Li. Segmentation of multiple, partially occluded objects by grouping, merging, assigning part detection responses. In *Computer Vision Pattern Recognition*, 2008.

[WÖF10] Daniel Weinland, Mustafa Özuysal, and Pascal Fua. Making action recognition robust to occlusions and viewpoint changes. In *European Conference on Computer Vision*. 2010.

[WRHS13] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *International Conference on Computer Vision*, pages 1385–1392, 2013.

[Wri06] Steve Wright. *Digital compositing for film and video*. Taylor & Francis, 2006.

[WS92] Donna J Williams and Mubarak Shah. A fast algorithm for active contours and curvature estimation. *Computer Vision and Image Understanding*, 1992.

[WTXC04] Jue Wang, Bo Thiesson, Yingqing Xu, and Michael Cohen. Image and video segmentation by anisotropic kernel mean shift. In *European Conference on Computer Vision*, 2004.

[XC12] Chenliang Xu and Jason J Corso. Evaluation of super-voxel methods for early video processing. In *Computer Vision Pattern Recognition*, 2012.

[XCJ08] Li Xu, Jianing Chen, and Jiaya Jia. A segmentation based variational model for accurate optical flow estimation. In *European Conference on Computer Vision*. 2008.

[XCS+06] Jiangjian Xiao, Hui Cheng, Harpreet Sawhney, Cen Rao, and Michael Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *European Conference on Computer Vision*. 2006.

[XJM12] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. Motion detail preserving optical flow estimation. *Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1744–1757, 2012.

[XPCH17] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *Computer Vision Pattern Recognition*, 2017.

[XXC12] Chenliang Xu, Caiming Xiong, and Jason J Corso. Streaming hierarchical video segmentation. In *European Conference on Computer Vision*. 2012.

[YJS06] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, 2006.

[YL15] Jiaolong Yang and Hongdong Li. Dense, accurate optical flow estimation with piecewise parametric model. In *Computer Vision Pattern Recognition*, 2015.

[YLS04] Alper Yilmaz, Xin Li, and Mubarak Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, 2004.

[YMSM95] Shinya Yamamoto, Yasushi Mae, Yoshiaki Shirai, and Jun Miura. Realtime multiple object tracking based on optical flows. In *International Conference on Robotics and Automation*, volume 3, pages 2328–2333. IEEE, 1995.

[YZC+07] Ting Yu, Cha Zhang, Michael Cohen, Yong Rui, and Ying Wu. Monocular video foreground/background segmentation by tracking spatial-color gaussian mixture models. In *Motion and Video Computing. IEEE Workshop on*, page 5. IEEE, 2007.

[Ziv04] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *International Conference on Pattern Recognition*, volume 2, pages 28–31. IEEE, 2004.

[ZLY10] Hong Zhang, Bo Li, and Dan Yang. Keyframe detection for appearance-based visual slam. In *International Conference on Intelligent Robots and Systems*, pages 2071–2076. IEEE, 2010.

[ZNCT07] Yan-Tao Zheng, Shi-Yong Neo, Tat-Seng Chua, and Qi Tian. The use of temporal, semantic and visual partitioning model for efficient near-duplicate keyframe detection in large scale news corpus. In *International Conference on Image and Video Retrieval*, pages 409–416. ACM, 2007.

[ZPB07] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime TV-L1 optical flow. In *Pattern Recognition*, pages 214–223. Springer, 2007.

[ZRY09] Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pages 3468–3497, 2009.