Motion in action : optical flow estimation and action localization in videos

Philippe Weinzaepfel

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques, Sciences et Technologies de l'Information**

Arrêté ministériel : 7 août 2006

Présentée par

Philippe WEINZAEPFEL

Thèse dirigée par **Cordelia SCHMID**
et codirigée par **Zaid HARCHAOUI**

préparée au sein d' **Inria Grenoble**
et de **l'école doctorale MSTII : Mathématiques, Sciences et Technologies de l'Information, Informatique**

Le mouvement en action: estimation du flot optique et localisation d'actions dans les vidéos

Motion in action: optical flow estimation and action localization in videos

Thèse soutenue publiquement le **23 septembre 2016**,
devant le jury composé de :

Pr. Jean Ponce
Ecole Normale Supérieure, Paris, France, Président
Pr. Martial Hebert
Carnegie Mellon University, Pittsburgh, PA, USA, Rapporteur
Dr. Ivan Laptev
Inria Paris, France, Rapporteur
Pr. Jitendra Malik
University of California at Berkeley, Berkeley, CA, USA, Examinateur
Dr. Cordelia Schmid
Inria Grenoble, Montbonnot, France, Directeur de thèse
Pr. Zaid Harchaoui
University of Washington, Seattle, WA, USA, Co-Directeur de thèse

Abstract

With the recent overwhelming growth of digital video content, automatic video understanding has become an increasingly important issue. This thesis introduces several contributions on two automatic video understanding tasks: optical flow estimation and human action localization.

Optical flow estimation consists in computing the displacement of every pixel in a video and faces several challenges including large non-rigid displacements, occlusions and motion boundaries. We first introduce an optical flow approach based on a variational model that incorporates a new matching method. The proposed matching algorithm is built upon a hierarchical multi-layer correlational architecture and effectively handles non-rigid deformations and repetitive textures. It improves the flow estimation in the presence of significant appearance changes and large displacements. We also introduce a novel scheme for estimating optical flow based on a sparse-to-dense interpolation of matches while respecting edges. This method leverages an edge-aware geodesic distance tailored to respect motion boundaries and to handle occlusions. Furthermore, we propose a learning-based approach for detecting motion boundaries. Motion boundary patterns are predicted at the patch level using structured random forests. We experimentally show that our approach outperforms the flow gradient baseline on both synthetic data and real-world videos, including an introduced dataset with consumer videos.

Human action localization consists in recognizing the actions that occur in a video, such as 'drinking' or 'phoning', as well as their temporal and spatial extent. We first propose a novel approach based on Deep Convolutional Neural Network. The method extracts class-specific tubes leveraging recent advances in detection and tracking. Tube description is enhanced by spatio-temporal local features. Temporal detection is performed using a sliding window scheme inside each tube. Our approach outperforms the state of the art on challenging action localization benchmarks. Second, we introduce a weakly-supervised action localization method, *i.e.*, which does not require bounding box annotation. Action proposals are computed by extracting tubes around the humans. This is performed using a human detector robust to unusual poses and occlusions, which is learned on a human pose benchmark. A high recall is reached with only several human tubes, allowing to effectively apply Multiple Instance Learning. Furthermore, we introduce a new dataset for human action localization. It overcomes the limitations of existing benchmarks, such as the diversity and the duration of the videos. Our weakly-supervised approach obtains results close to fully-supervised ones while significantly reducing the required amount of annotations.

Keywords: optical flow, action localization, convolutional neural network, video analysis, computer vision, machine learning

Résumé

Avec la récente et importante croissance des contenus vidéos, la compréhension automatique de vidéos est devenue un problème majeur. Ce mémoire présente plusieurs contributions sur deux tâches de la compréhension automatique de vidéos : l'estimation du flot optique et la localisation d'actions humaines.

L'estimation du flot optique consiste à calculer le déplacement de chaque pixel d'une vidéo et fait face à plusieurs défis tels que les grands déplacements non rigides, les occlusions et les discontinuités du mouvement. Nous proposons tout d'abord une méthode pour le calcul du flot optique, basée sur un modèle variationnel qui incorpore une nouvelle méthode d'appariement. L'algorithme d'appariement proposé repose sur une architecture corrélationnelle hiérarchique à plusieurs niveaux et gère les déformations non rigides ainsi que les textures répétitives. Il permet d'améliorer l'estimation du flot en présence de changements d'apparence significatifs et de grands déplacements. Nous présentons également une nouvelle approche pour l'estimation du flot optique basée sur une interpolation dense de correspondences clairsemées tout en respectant les contours. Cette méthode tire profit d'une distance géodésique basée sur les contours qui permet de respecter les discontinuités du mouvement et de gérer les occlusions. En outre, nous proposons une approche d'apprentissage pour détecter les discontinuités du mouvement. Les motifs de discontinuité du mouvement sont prédits au niveau d'un patch en utilisant des forêts aléatoires structurées. Nous montrons expérimentalement que notre approche surclasse la méthode basique construite sur le gradient du flot tant sur des données synthétiques que sur des vidéos réelles. Nous présentons à cet effet une base de données contenant des vidéos d'utilisateurs.

La localisation d'actions humaines consiste à reconnaître les actions présentes dans une vidéo, comme 'boire' ou 'téléphoner', ainsi que leur étendue temporelle et spatiale. Nous proposons tout d'abord une nouvelle approche basée sur les réseaux de neurones convolutionnels profonds. La méthode passe par l'extraction de tubes dépendants de la classe à détecter, tirant parti des dernières avancées en matière de détection et de suivi. La description des tubes est enrichie par des descripteurs spatio-temporels locaux. La détection temporelle est effectuée à l'aide d'une fenêtre glissante à l'intérieur de chaque tube. Notre approche surclasse l'état de l'art sur des bases de données difficiles de localisation d'actions. Deuxièmement, nous présentons une méthode de localisation d'actions faiblement supervisée, c'est-à-dire qui ne nécessite pas l'annotation de boîtes englobantes. Des candidats de localisation d'actions sont calculés en extrayant des tubes autour des humains. Cela est fait en utilisant un détecteur d'humains robuste aux poses inhabituelles et aux occlusions, appris sur une base de données de poses humaines. Un rappel élevé est atteint avec seulement quelques tubes, permettant d'appliquer un apprentissage à plusieurs instances. En outre, nous présentons une nouvelle

base de données pour la localisation d'actions humaines. Elle surmonte les limitations des bases existantes, telles la diversité et la durée des vidéos. Notre approche faiblement supervisée obtient des résultats proches de celles totalement supervisées alors qu'elle réduit significativement l'effort d'annotations requis.

Mots-clefs : flot optique, localisation d'actions, réseaux de neurones convolutionnels, analyse de vidéos, vision par ordinateur, apprentissage machine

Acknowledgements

It was an exceptional chance to work in this great environment with fantastic colleagues. First of all, I would like to thank my supervisors, Cordelia Schmid and Zaid Harchaoui, for their invaluable guidance, support and drive for excellence. In particular, Cordelia's experience, intuition and vision have been extremely precious during all these years and Zaid's openness, contagious enthusiasm and scientific culture have made me pushed past my boundaries and allowed me to technically and scientifically progress throughout all projects. Furthermore, I am grateful to Professor Jitendra Malik for offering me the chance to visit UC Berkeley. I would also like to thank my co-authors Jérome Revaud and Xavier Martin. Daily discussions with Jérome have lead to exciting work in optical flow estimation. The collaboration with Xavier has been extremely precious, especially for solving the numerous challenges when collecting the DALY dataset. Without them, this work will not exist in this form. Besides my supervisors and co-authors, I would like to thank the interns I have supervised during my PhD, Quentin Cormier and Erwan Le Roux. This experience was extremely valuable. Many thanks go to my jury members – Professor Martial Hebert, Doctor Ivan Laptev, Professor Jitendra Malik and Professor Jean Ponce – for agreeing to evaluate my work. I am also grateful to all the colleagues I met over the past few years. They are too many to be exhaustively cited here. My special thanks go to Mattis Paulin, Nicolas Chesneau, Gregory Rogez, Guillaume Fortier, and to my office mates Thomas Mensink, Zeynep Akata, Dan Oneata, Shreyas Saxena, Vicky Kalogeiton and Valentin Thomas, for the valuable and daily discussions we had and their support during these years. I would also like to thank Nathalie Gillot who helped me in all administrative tasks. I finally cannot express how grateful I am to my family and friends for their tireless and unconditional support.

Contents

Chapter 1

Introduction

Contents

Automatic video understanding is increasingly relevant as the number and the quality of capturing devices have significantly grown over the past few years. For instance, in 2014, YouTube has more than one billion visitors watching hundreds of millions of hours of videos every day[1]. In 2015, YouTube's CEO revealed that more than 400 hours of videos are uploaded every minute[2]. In 2019, videos are expected to represent 80% of the internet traffic, and it would take one person over 5 million years to watch the amount of videos that will cross global IP networks each month[3]. With the overwhelming amount of video data, designing an automatic tool to analyze and understand this content has become a critical issue.

An example of application is illustrated in Figure 1.1. A user may need to retrieve a particular video instant in his personal collection which contains a huge number of photos and videos, captured by various devices such as cameras, smartphones, tablets or action cameras (*e.g.* GoPro). Such clip retrieval will require to identify people and to recognize the performed actions as well as their extent in the videos. Significant progress has been made in computer vision over the past few years, in particular in face recognition (detection, identification, verification) and in image analysis (object

1. https://www.youtube.com/yt/press/statistics.html
2. http://www.reelseo.com/
3. http://www.cisco.com/

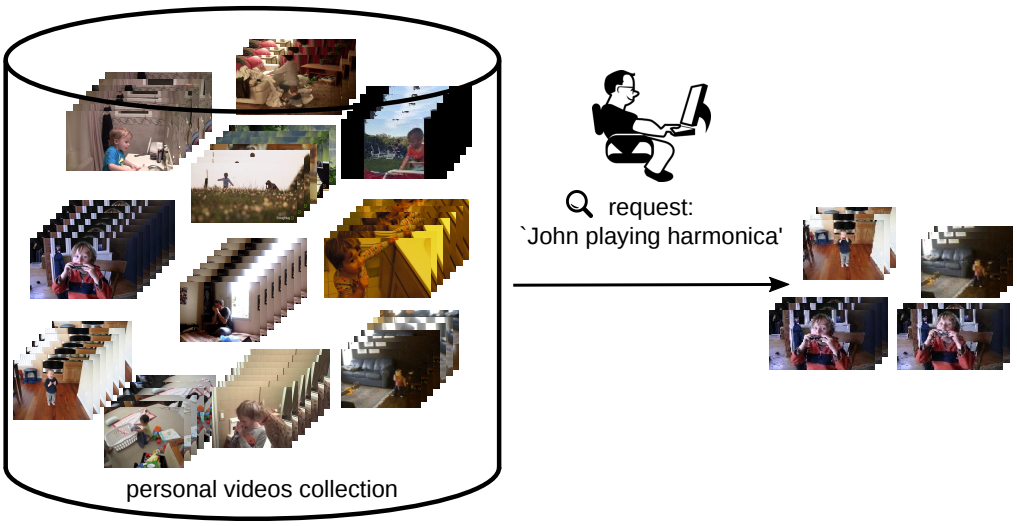


Figure 1.1 – Illustration of requesting for a particular video instant in a collection. Based on the search *John playing harmonica*, the system retrieves video clips in which this happens.

detection and segmentation, image captioning). As an example, Facebook automatically identifies people in newly uploaded photos and Google Photos allows to organize or search photos by people, places or objects. Nevertheless, the generalization to videos requires tools to efficiently analyze video content, and in particular human actions.

Videos span a wide range of sources and applications: indexing and retrieving videos from personal cameras, internet content platforms (*e.g.* YouTube, Facebook, Dailymotion), TV shows or movies; analyzing the surrounding environment for autonomous robots (*e.g.* autonomous cars, drones) or enhancing connected objects (*e.g.* augmented reality glasses); recognizing poses and motion for human-machine interfaces (*e.g.* Kinect); recognizing and predicting actions as well as behaviors for video-surveillance; tracking players and recognizing fine-grained actions in sports videos.

Making computers able to understand and interpret a huge amount of videos is thus fundamental. Compared to images, videos also convey the dynamics of a real-world scene over a given period of time that can vary from few milliseconds to multiple hours. This dynamic is a rich source of information, and analyzing it is what separates video understanding from image understanding. This explains why most video representation models [Laptev, 2005, Wang et al., 2013, Simonyan and Zisserman, 2014] integrate a component which describes the motion in the scene. For instance, Figure 1.2 illustrates a pipeline for action recognition [Wang et al., 2013] in which

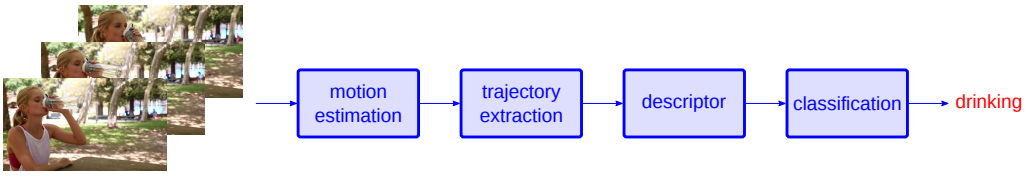


Figure 1.2 – Illustration of an action recognition pipeline [Wang et al., 2013]. First, the motion is estimated and used to extract trajectories. Then, a video description is built, based on static and motion cues. Finally, the description is fed to a classifier to predict the label.

Figure 1.3 – Example of spatio-temporal action localization of the class *drinking* from the DALY dataset [Weinzaepfel et al., 2016]. The boxes in yellow represent the ground-truth spatial localization of the action during its temporal extent.

estimated motion is used for extracting trajectories and describing them. Our work focuses on *human actions* and in particular on the problem of localizing them in uncontrolled videos. For instance, Figure 1.3 shows an example of localization in space and time of the *drinking* action from the DALY dataset [Weinzaepfel et al., 2016]. As recently shown by Jhuang et al. [2013] and Varol et al. [2016], the quality of the estimated motion in the video representation has an impact on the performance. We hence study the problem of optical flow estimation, in particular in the case of non-rigid large displacements, such as the motion of human limbs. Indeed, state-of-the-art optical flow methods tend to perform poorly on fast motion, as highlighted by the MPI-Sintel dataset [Butler et al., 2012].

1.1 Goals

This dissertation addresses two important problems in video understanding. The first one is low-level and consists in estimating optical flow in realistic videos, *i.e.*, with fast non-rigid motion such as human motion. The second task is higher-level and consists in detecting generic human actions

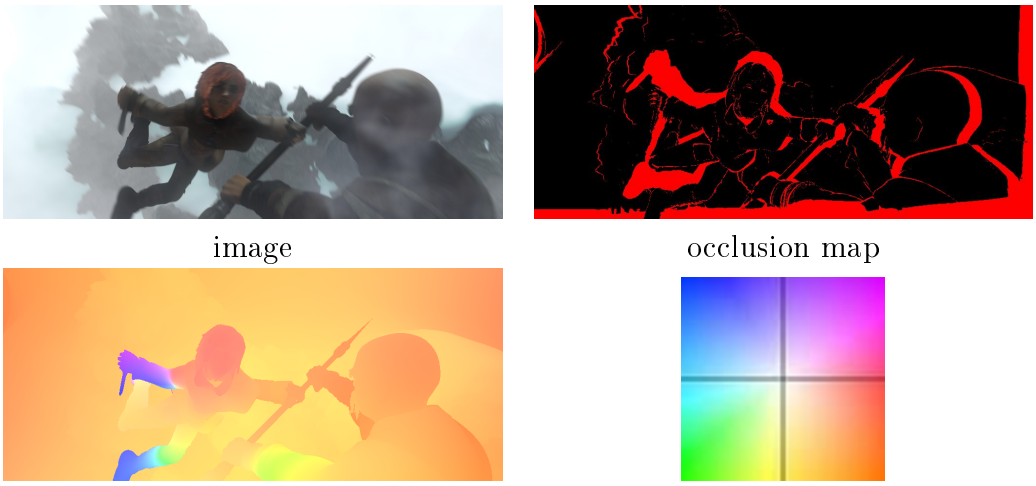


Figure 1.4 – Example optical flow from the MPI-Sintel dataset [Butler et al., 2012] with the image (top left), the ground-truth flow (bottom left) using the color code shown in bottom right, and the occluded pixels represented in red (top right).

in uncontrolled videos. We now briefly present these two tasks, as well as the challenges involved in each.

Optical flow estimation in realistic videos

Optical flow estimation consists in computing a motion vector for every pixel in a video. Figure 1.4 shows an example of ground-truth optical flow from the MPI-Sintel dataset [Butler et al., 2012]. Despite significant progress over the past decades, optical flow estimation in videos remains challenging, especially in the case of fast motion of small parts [Brox and Malik, 2011]. Such motion typically appears for deformable objects such as humans (see the limbs of the leftmost character in Figure 1.4). This dissertation is thus concerned with estimating optical flow in realistic videos, *i.e.*, with possibly fast non-rigid motion. Estimating optical flow in realistic videos also implies facing challenges, such as motion discontinuities, occlusions, large displacements and varying lighting conditions. We now present these challenges in more detail.

The first one consists in respecting **motion discontinuity**. The motion of a complex scene can be decomposed into independent moving objects, *i.e.*, layers, each one with smooth motion. Detecting the motion discontinuities is thus capital for an accurate optical flow estimation. Incorporating

discontinuities into the motion model is also difficult as most of the mathematical formulations require a continuous function. In addition, modeling the motion discontinuities and the smooth motion inside each independent layer at the same time is extremely challenging.

The second difficulty consists in handling **occlusions**. Since multiple objects move independently, foreground objects make parts of the background layers appear and disappear in each frame of the video. In a similar spirit, objects can enter or leave the field-of-view of the camera. For instance, in Figure 1.4, large occluded areas appear at the bottom and right borders due to camera motion, and at motion discontinuities. Estimating optical flow in these areas requires a high-level and long-term understanding of the motion in the scene.

The third challenge concerns **large non-rigid displacements**. These displacements are frequent in real-world videos. A typical example is the limbs of the humans that have fast motion, especially compared to their size. For instance, in Figure 1.4, the hands and the feet of the left character have motion that are not related to his torso. For small objects, few pixels can be used as evidence for their displacements. Moreover, large displacements add other difficulties, such as strong discontinuities, motion blur, or wide changes in shapes and appearances across frames due to video compression.

A fourth difficulty lies in variations of the **lighting conditions**. The appearance of a given object may vary throughout the video due to changes in the lights. In the same spirit, if an object enters in the shadow of another one, its color appearance will abruptly change. Optical flow models must incorporate such cases for accurate estimation.

Finally, optical flow is a low-level cue used for higher-level tasks such as action recognition or tracking. Consequently, **efficiency** is capital for computing flows in a huge amount of videos or for real-time processing.

Human action localization in uncontrolled videos

The second task consists in localizing actions in videos, *i.e.*, recognizing actions as well as their temporal extent and the spatial extent of the actor(s). For instance, Figure 1.3 shows one instance for the *drinking* class from the DALY dataset [Weinzaepfel et al., 2016] and Figure 1.5 shows an example video from the same dataset with a human performing multiple short actions in a long YouTube video. This dissertation addresses human action localization (called also human action detection), both in space and time, in uncontrolled videos. This problem faces multiple challenges such as designing a representation both robust to intra-class variability, and sufficiently discriminative in order to avoid inter-class confusion, as well as

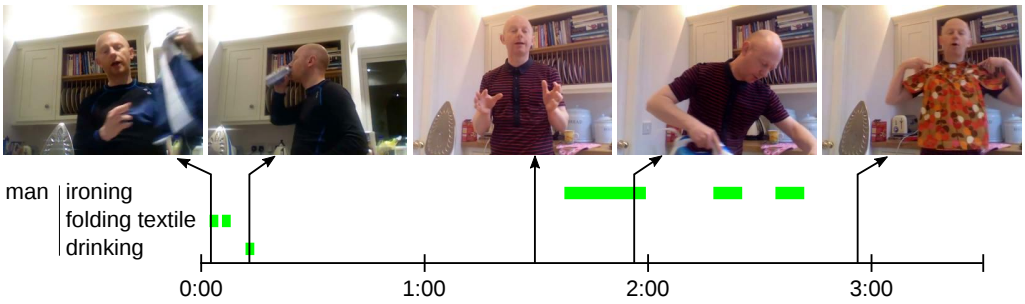


Figure 1.5 – Example of spatio-temporal action localization in a long untrimmed video from the DALY dataset [Weinzaepfel et al., 2016]. Temporal extent of the actions performed by the man are shown in green.

difficulties due to the localization in space and time. We now describe these challenges in more detail.

Some difficulties are due to **intra-class variability**. Appearance and motion may significantly differ between instances of a class. This is caused by differences in capturing conditions (*e.g.* camera motion, wide range of viewpoint, occlusion and lighting conditions) and variations in the execution style of an action (*e.g.* motions, poses, objects involved, speed). All these variations, highlighted by Figure 1.6, make modeling human action in uncontrolled videos challenging. Simple assumptions such as constant position of the human throughout the videos are not adequate for real-world data.

Another set of challenges are due to **inter-class confusion**. Different actions may share similar motions (*e.g.* drinking and smoking), similar objects (*e.g.* phoning and taking photo can both be performed with a smartphone) or similar poses (*e.g.* stand up and sitting down). For instance, hand movement happens near the mouth for the 3 actions shown in Figure 1.6. Designing an action model which is robust to intra-class variability and discriminative between classes is thus extremely difficult.

Localizing actions in addition to recognizing them adds several challenges. Spatial localization is a difficult task as the human locations may vary throughout the videos, or the actors may be partially visible. As a consequence, per-frame detectors may not have a perfect recall and trackers may fail. In addition, in the case of crowded scenes, many humans might be present at the same time with multiple action instances co-occurring. Moreover, each human can perform multiple actions, see Figure 1.5, possibly at the same time. For instance, one may be phoning while running. Temporal localization adds other challenges as it requires to accurately detect the different phases such as starting time and end time. Furthermore, action

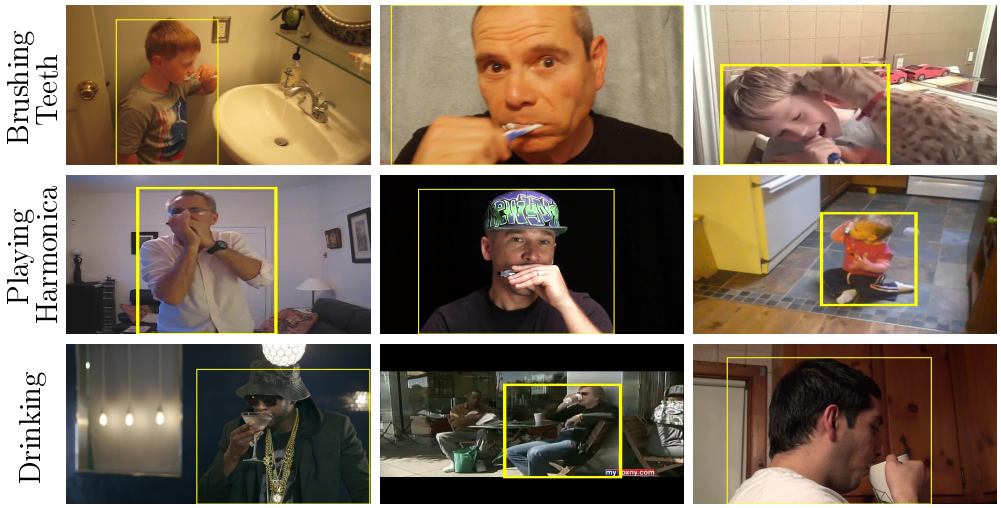


Figure 1.6 – Example frames for 3 classes from the DALY dataset [Weinzaepfel et al., 2016].

can last only several seconds in videos of multiple minutes, see Figure 1.5.

Finally, the level of **supervision** required for training an accurate human action localization approach is important. A fully-supervised model, *i.e.* which requires bounding box annotations around the human performing an action for training, can be extended to many action classes only at the cost of a huge annotation effort. Reducing the supervision, for instance to video labels or to the temporal extent of the actions in training videos, is necessary for a better and easier generalization.

1.2 Context

Automatic video understanding is one of the main challenge in computer vision. Making computers able to capture the dynamics of a scene started in the 1970s. The first attempts have mainly studied the problem of segmenting moving objects in a scene or separating the foreground from the background [Martin and Aggarwal, 1977, Jain et al., 1977]. Most of these works rely on the difference between consecutive images in a sequence [Jain and Nagel, 1979]. Some other approaches proposed models to estimate the velocity field and its discontinuity [Fennema and Thompson, 1979, Nakayama and Loomis, 1974], assuming rigid objects.

More robust models for motion estimation appeared in the 1980s based on optical flow. Optical flow describes the apparent motion of an image brightness pattern. In general, optical flow is a projection of the motion

field that links pixels between 2D frames. Nevertheless, a small difference is that optical flow may also be due to difference in brightness pattern not related to motion; for instance due to changes in lighting conditions. Optical flow and motion estimation are often confused in the literature, in particular when models integrate a constancy assumption of the gradient, as this is the case in this dissertation. In their pioneering work, Horn and Schunck [1981] minimize a global energy composed of a data-fidelity term and a smoothness term to estimate the optical flow. This formulation has been improved over the years [Black and Anandan, 1996, Brox et al., 2004, Bruhn et al., 2005a, Werlberger et al., 2009, Baker et al., 2011, Vogel et al., 2013a, Sun et al., 2014b] and obtains excellent performance on small displacements. Integration of image matching [Tola et al., 2008, Brox and Malik, 2011, Xu et al., 2012] was recently proposed in order to improve robustness to large displacements, as highlighted by the performance on MPI-Sintel benchmark [Butler et al., 2012], see Figure 1.7. Optical flow has been widely used in computer vision as a motion field for a variety of tasks such as tracking [Mae et al., 1996, Shin et al., 2005], driver assistance systems [Geiger et al., 2013, Fletcher et al., 2003, Sun et al., 2004], motion segmentation [Brox and Malik, 2010, Papazoglou and Ferrari, 2013], or action recognition [Wang et al., 2013, Simonyan and Zisserman, 2014].

Recognizing human actions is a higher-level task in video understanding. First attempts from the 1990s were based on volumetric human models [Rohr, 1994, Campbell et al., 1996] and then simplified by using 2D silhouettes [Brand, 1999]. In the same spirit, Bobick and Davis [2001] represent the temporal evolution of a silhouette using Motion History Images. Nevertheless, these models are limited to constrained capturing conditions since they rely on background subtraction or silhouette extraction. Thus, they can not generalize well to real-world videos. Driven by their success on image classification, local features [Chomat and Crowley, 1999, Schüldt et al., 2004, Dollár et al., 2005, Laptev, 2005, Wang et al., 2013] have been widely used in the 2000s. They either describe the appearance [Kläser et al., 2008, Gorelick et al., 2007] or the optical flow [Laptev et al., 2008, Wang et al., 2013]. Actions are represented by an aggregation of the local features, for instance based on bag-of-words [Sivic and Zisserman, 2003] or on Fisher Vectors [Sánchez et al., 2013]. Local feature representations have been successfully applied in real-world videos thanks to their lack of global assumptions such as geometric relations. To improve these models with a more structured description, several extensions were proposed such as spatio-temporal pyramids [Laptev et al., 2008] or stacking over supervoxels [Peng et al., 2014]. More recently, the temporal order of the features [Gaidon et al., 2013, Fernando et al., 2015] has been integrated.

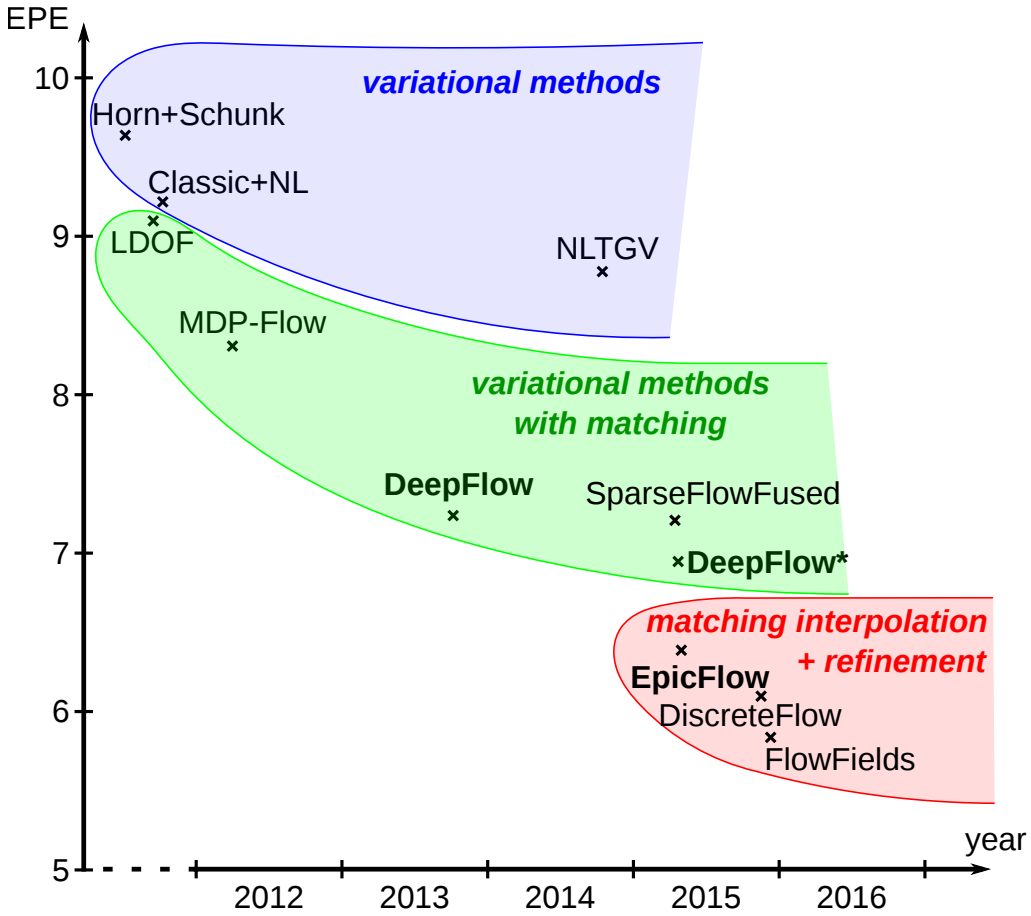


Figure 1.7 – Schematic view of the advances in optical flow methods. The reported results is the average endpoint error (EPE) on the MPI-Sintel dataset [Butler et al., 2012]. Methods are grouped into 3 categories: variational methods, variational methods integrating matching, matching interpolation followed by a variational refinement. Our methods, DeepFlow (and its improved version DeepFlow*) and EpicFlow, are shown in bold. The three areas are shown for better readability but do not rely on any theoretical limit of the categories.

For localizing human actions, the first models were either based on cuboids [Laptev and Pérez, 2007, Cao et al., 2010, Yuan et al., 2009] or on figure-centric models [Kläser et al., 2010, Lan et al., 2011]. Cuboids, *i.e.*, fixed human positions over frames, can not generalize to the case of moving actors or moving camera. Figure-centric models leverage a human detector [Kläser et al., 2010] or treat the actor position as a latent variable [Lan et al., 2011]. Moeslund et al. [2006] decompose human action recognition systems into four stages. The first one is an initialization step which consists in detecting the human at the beginning of a video. The second step consists in tracking this human throughout the video. Third, the human track is modeled, using for instance a pose representation. This modeled feature is finally used for classification. More recently, other approaches have been proposed based on extensions of successful methods for object detection in images, such as part-based models or proposals. For instance, Tian et al. [2013] extend the deformable parts model [Felzenszwalb et al., 2010] to videos. Proposals have been extended to actions in videos, for instance based on clustering supervoxels [Jain et al., 2014b, Oneata et al., 2014a] or trajectories [van Gemert et al., 2015, Marian Puscas et al., 2015].

1.3 Contributions

1.3.1 Optical flow estimation

Our work on optical flow focuses on the case of large displacements, occlusions and discontinuities. After reviewing related work in Chapter 2, we describe our three contributions. The first one is a variational approach (DeepFlow) that integrates a new matching algorithm (DeepMatching), leading to a significant boost in performance, see Figure 1.7. The second contribution is a novel scheme (EpicFlow) for interpolating matches while respecting the edges, before performing a full-scale variational approach. EpicFlow has opened a new category of methods based on matches' interpolation, see Figure 1.7. Our third contribution is a learning-based approach to detect motion boundaries. We now present these three contributions in more detail.

DeepFlow: Large displacement optical flow with DeepMatching. Inspired by the large displacement optical flow (LDOF) of Brox and Malik [2011], we introduce DeepFlow which blends a new matching algorithm, called DeepMatching, with a variational approach for optical flow. Deep-

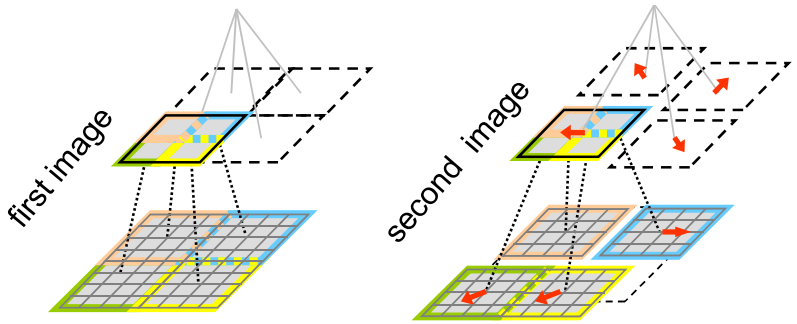


Figure 1.8 – Illustration of possible matches obtained with DeepMatching. *Left:* patch hierarchy in the first image: each squared patch is composed of four quadrants. *Right:* one possible displacement of corresponding patches in the second image: each of the four quadrants can move, according to its parent displacement, in any direction for a limited extent.

Matching relies on a hierarchical, multi-layer, correlational architecture designed for matching images and was inspired by deep convolutional approaches. The proposed matching algorithm can handle non-rigid deformations, repetitive textures, and efficiently determines dense correspondences in the presence of significant changes between images. Figure 1.8 illustrates the approach. We evaluate the performance of DeepMatching, in comparison with state-of-the-art matching algorithms, on the Mikolajczyk [Mikolajczyk et al., 2005], the MPI-Sintel [Butler et al., 2012] and the Kitti [Geiger et al., 2013] datasets. DeepMatching outperforms the state-of-the-art algorithms and shows excellent results, especially for repetitive textures. For optical flow estimation, DeepFlow is competitive with the state of the art on public benchmarks thanks to additional robustness to large displacements and complex motion obtained by integrating DeepMatching. This work was published in ICCV'13 [Weinzaepfel et al., 2013], will appear soon in IJCV [Revaud et al., 2016], and is presented in Chapter 3.

EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. We propose a novel approach for optical flow estimation, targeted at large displacements with significant occlusions. The method is based on an edge-preserving dense interpolation of a sparse set of matches. The sparse-to-dense interpolation relies on an appropriate choice of the distance, namely an edge-aware geodesic distance. This distance is tailored to handle occlusions and motion boundaries – two common and difficult issues for optical flow computation. We also propose an approximation scheme for the geodesic distance to allow fast computation without loss of performance.

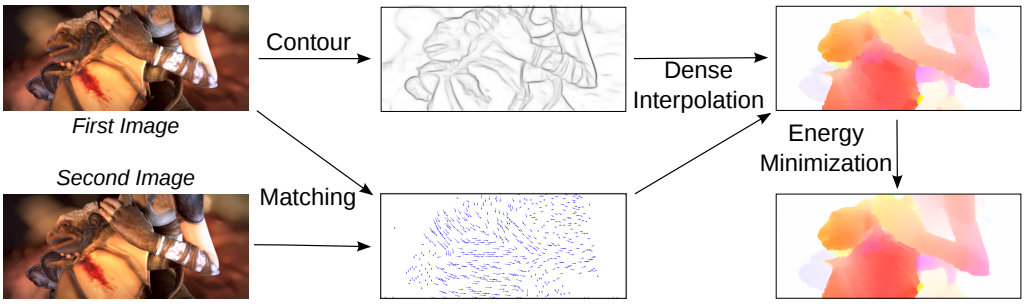


Figure 1.9 – Overview of EpicFlow. Given two images, we compute matches using DeepMatching [Revaud et al., 2016] and the edges of the first image using SED [Dollár and Zitnick, 2013]. We combine these two cues to densely interpolate matches and obtain a dense correspondence field. This is used as initialization of a one-level energy minimization framework.

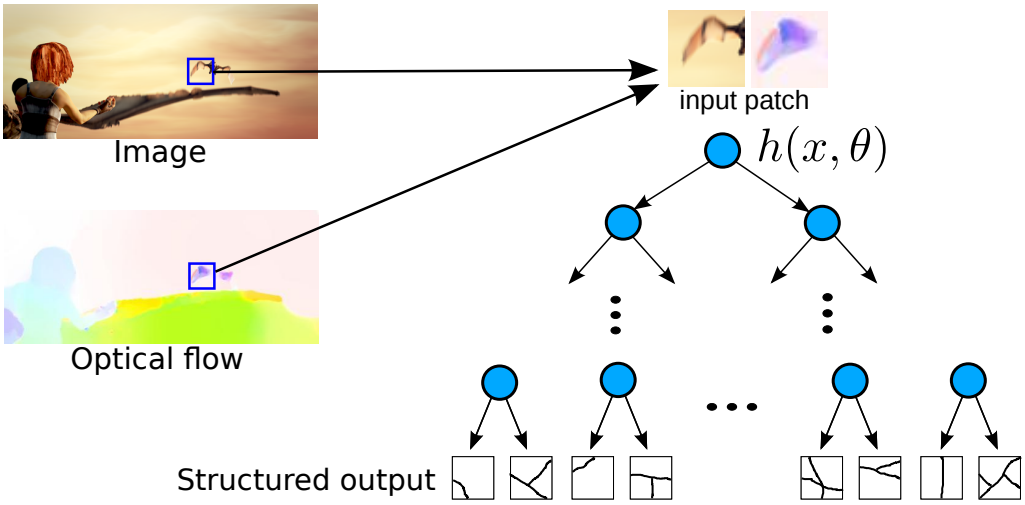


Figure 1.10 – Illustration of the motion prediction process with our structured decision tree. Given an input patch from the left image (represented here by image and flow channels), we predict a binary boundary mask, *i.e.*, a leaf of the tree. Predicted masks are averaged across all trees and all overlapping patches to yield the final soft-response boundary map.

Subsequent to the dense interpolation step, standard one-level variational energy minimization is carried out to refine the flow estimation. The proposed approach, called Edge-Preserving Interpolation of Correspondences *(EpicFlow)* is fast and robust to large displacements. Figure 1.9 summarizes the approach. This work was published in CVPR'15 [Revaud et al., 2015] and is presented in Chapter 4.

Motion boundaries detection. We introduce a learning-based approach for motion boundaries detection. The proposed method relies on a structured random forest trained on the ground-truth of the MPI-Sintel dataset. The random forest leverages several cues at the patch level, namely appearance (RGB color) and motion cues (optical flow estimated by state-of-the-art algorithms). Figure 1.10 summarizes the motion boundaries detection process. Experimental results show that the proposed approach is both robust and computationally efficient. It significantly outperforms state-of-the-art motion-difference approaches on the MPI-Sintel and Middlebury datasets. We compare the results obtained with several state-of-the-art optical flow approaches and study the impact of the different cues used in the random forest. Furthermore, we introduce a new dataset, the YouTube Motion Boundaries dataset (**YMB**), that comprises 60 sequences taken from real-world videos with manually annotated motion boundaries. On this dataset, our approach, although trained on MPI-Sintel, also outperforms by a large margin state-of-the-art algorithms based on optical flow. This work was published in CVPR'15 [Weinzaepfel et al., 2015b] and is presented in Chapter 5.

1.3.2 Human action localization

Our work on action localization is built upon successful detectors in images and tracking-by-detection approaches. By combining these two components, we obtain candidate localization of the actions, which can be represented by classical aggregation of local features. Furthermore, we also show that accurate results can be obtained without spatial supervision. After reviewing related work in Chapter 6, we introduce the two following contributions: an action-specific tracker for fully-supervised action localization, and a human-specific tracker for weakly-supervised action localization. We now present these two contributions in more detail.

Action-specific tracks for fully-supervised action localization. We propose an effective approach for spatio-temporal action localization in realistic videos. The approach first detects proposals at the frame-level and scores them with a combination of static and motion CNN features. It then tracks high-scoring proposals throughout the video using a tracking-by-detection approach. Our tracker relies simultaneously on instance-level and class-level detectors. The tracks are scored using aggregation of local features in combination with the CNN features. Finally, we perform temporal localization of the action using a sliding window at the track level. We outperform the state of the art for spatio-temporal action localization on

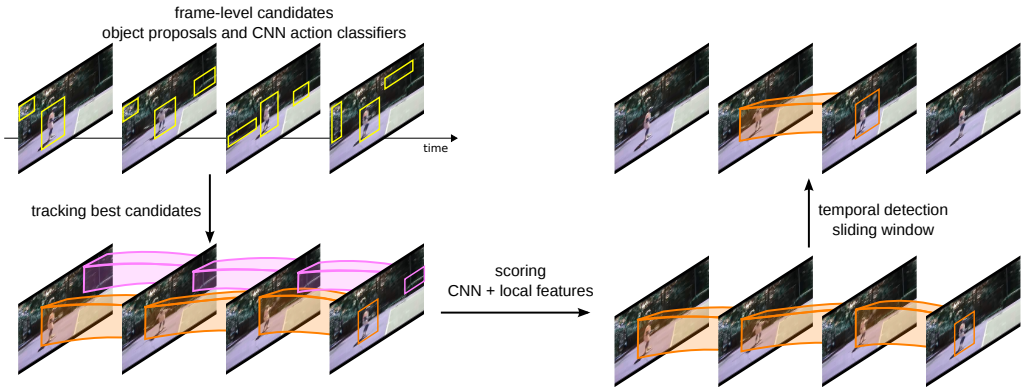


Figure 1.11 – Overview of our fully-supervised action localization approach. We detect frame-level object proposals and score them with CNN action classifiers. The best candidates, in term of scores, are tracked throughout the video. We then score the tracks with CNN and local features classifiers. Finally, we perform a temporal sliding window for detecting the temporal extent of the action.

the UCF-Sports, J-HMDB and UCF-101 datasets. This work was published in ICCV'15 [Weinzaepfel et al., 2015a] and is presented in Chapter 7.

Human-specific tracks for weakly-supervised action localization. We present a novel approach for weakly-supervised action localization, *i.e.*, that does not require per-frame spatial annotations for training. We first introduce an effective method for extracting human tubes by combining a state-of-the-art human detector with a tracking-by-detection approach. Our tube extraction leverages the large amount of annotated humans available today and outperforms the state of the art by an order of magnitude: with less than 5 tubes per video, we obtain a recall of 95% on the UCF-Sports and J-HMDB datasets. Given these human tubes, we perform weakly-supervised selection based on multi-fold Multiple Instance Learning (MIL) with dense trajectories and achieve excellent results. We obtain a mAP of 84% on UCF-Sports, 54% on J-HMDB and 45% on UCF-101, which outperforms the state of the art for weakly-supervised action localization and is close to the performance of the best fully-supervised approaches. In addition, we introduce a new realistic dataset for action localization, named **DALY** (Daily Action Localization in YouTube). It contains high quality temporal and spatial annotations for 10 actions in 31 hours of videos (3.3M frames), which is an order of magnitude larger than standard action localization datasets. Figure 1.13 compares the main features of DALY

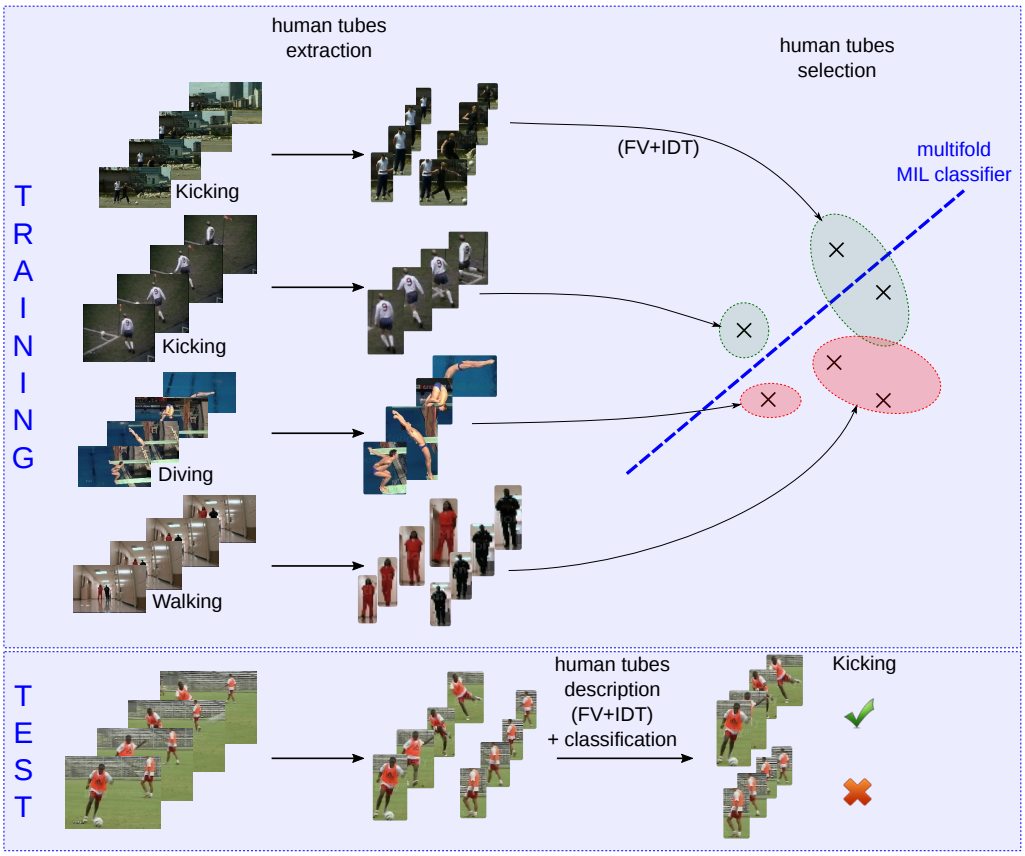


Figure 1.12 – Overview of our weakly-supervised action localization method. For training, we extract human tubes for all videos using a human detector and a human-specific tracker. These tubes are described using Improved Dense Trajectories [Wang et al., 2015]. Multiple-Instance Learning is then used to learn a classifier from video labels. At test time, human tubes are extracted with their features and the learned classifier is used to predict the label.

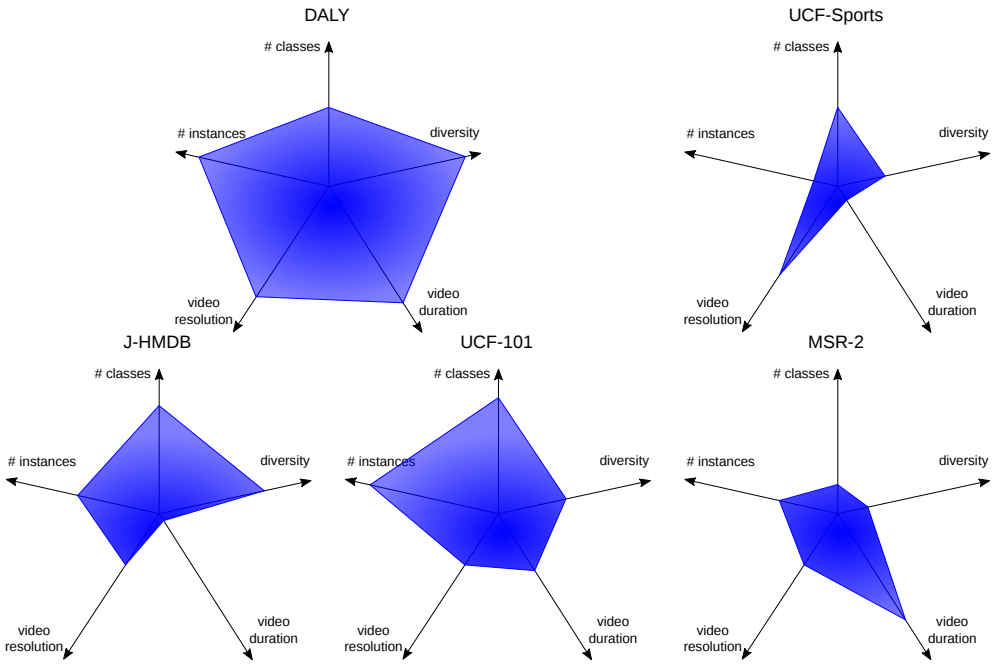


Figure 1.13 – Comparison of the DALY dataset [Weinzaepfel et al., 2016] with existing action detection datasets.

with existing spatio-temporal action localization datasets. On the DALY dataset, our tubes have a spatial recall of 82%, but the localization task is extremely challenging, we obtain 10.8% mAP. This work is presented in Chapter 8.

Part I

Optical Flow Estimation in Realistic Videos

Chapter 2

Related Work on Optical Flow

Contents

In this chapter, we review related work on optical flow estimation and present the datasets and metrics used for evaluation. We start by introducing optical flow in Section 2.1. Most existing optical flow methods are based on a variational formulation. We give more details on variational methods in Section 2.2 and briefly present other approaches in Section 2.3.

Recently, image matching was integrated into optical flow formulation. We review image matching as well as their integration into optical flow models in Section 2.4. Finally, Section 2.5 presents the datasets and metrics for evaluating optical flow methods.

2.1 Optical Flow

When viewing a sequence of images, *e.g.* when watching a movie, the human gets the illusion of motion while objects are simply represented at different locations in each still image. Motion perception is actually inferred by illumination changes of a point at the retina with connection to the neighboring points. Optical flow denotes these changes of the brightness pattern over time. It represents the 2D vectors that link points of two consecutive images together, *i.e.*, it is the 2D projection of the real-world 3D motion. We review in this section the basic concepts of optical flow.

2.1.1 Optical Flow Constraint

Let I_1 and I_2 be two consecutive images, defined in the space $\Omega \subset \mathbb{R}^2$. The task consists in estimating the optical flow $\mathbf{w} : \Omega \to \mathbb{R}^2$ between these two images. For each pixel $\mathbf{x} = (x, y)^\top \in \Omega$, the flow $\mathbf{w}(\mathbf{x})$ can be decomposed in its $x-$ and $y-$ component $\mathbf{w}(\mathbf{x}) = (u(\mathbf{x}), v(\mathbf{x}))^\top$.

The most basic assumption for optical flow is the constancy of the brightness along the displacement, for instance used by Horn and Schunck [1981] and Lucas and Kanade [1981]:

$$I_1\big(\mathbf{x}\big) = I_2\big(\mathbf{x} + \mathbf{w}(\mathbf{x})\big) \quad . \tag{2.1}$$

Assuming that the images are smooth and the displacements are small, first order Taylor expansion can be applied, resulting in: $I_2\big(\mathbf{x} + \mathbf{w}(\mathbf{x})\big) = I_2(\mathbf{x}) + I_x u(\mathbf{x}) + I_y v(\mathbf{x})$ with $I_x = \frac{\partial I_2}{\partial x}$ and $I_y = \frac{\partial I_2}{\partial y}$. If we denote by I_t the temporal derivative $I_t = I_2 - I_1$, we obtain:

$$I_t + I_x u + I_y v = 0 = I_t + (\nabla_2^\top I)\mathbf{w} = (\nabla_3^\top I)\mathbf{W} \quad , \tag{2.2}$$

where ∇_2 denotes the 2D partial derivatives, *i.e.*, $\nabla_2 = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})^\top$, ∇_3 denotes the 3D partial derivatives, *i.e.*, $\nabla_3 = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial t})^\top$, and $\mathbf{W} = (u, v, 1)^\top$. Equation 2.2 is well known as the *optical flow constraint*.

This equation with two unknowns is ill-posed. This ambiguity is called *the aperture problem*. Consider for instance a moving edge structure seen through an aperture. The motion can only be estimated along the normal

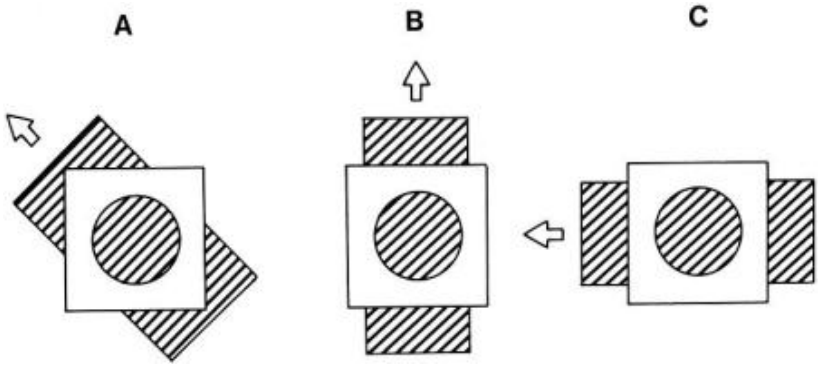


Figure 2.1 – Illustration of the aperture problem. Different motions can explain the repetitive pattern observed in the circle.

of this edge. In the same spirit, if a moving repetitive texture is seen through the aperture (with the borders out of the field of view), its motion is ambiguous, see Figure 2.1.

Additional constraints must be added to make Equation 2.2 well-posed. These constraints can be either global or local, resulting in two families of approaches that we now describe.

2.1.2 Local approach

Local approaches for optical flow estimate the displacement at one pixel based only on information around this pixel. A classical example is the work from Lucas and Kanade [1981]. The optical flow constraint (Equation 2.2) is solved for a pixel $\mathbf{x}$ based on the assumption that the motion field is constant in the neighborhood $\mathcal{N}(\mathbf{x})$ of $\mathbf{x}$. An overdetermined system of equations is obtained and the flow can be computed by minimizing the least squared errors:

$$E\big(\mathbf{w}(\mathbf{x})\big) = \sum_{\mathbf{x'}\in\mathcal{N}(\mathbf{x})} g(\mathbf{x'},\mathbf{x})\Big(\big(\nabla_3^\top I(\mathbf{x'})\big)\mathbf{W}(\mathbf{x})\Big)^2 \quad , \tag{2.3}$$

where g is a weighting function which typically decreases when the distance between $\mathbf{x}$ and $\mathbf{x'}$ increases. This energy function being convex, the global minimum of Equation 2.3 is obtained when the derivatives with respect to the components of the flow are zeros:

$$\begin{cases} \frac{\partial E\left(\mathbf{w}(\mathbf{x})\right)}{\partial u} = \sum g\big(I_t I_x + u I_x^2 + v I_y I_x\big) = 0 \quad , \\ \frac{\partial E\left(\mathbf{w}(\mathbf{x})\right)}{\partial v} = \sum g\big(I_t I_y + u I_x I_y + v I_y^2\big) = 0 \quad . \end{cases} \tag{2.4}$$

This set of equations can be written in a matrix form:

$$\begin{pmatrix} \sum gI_x^2 & \sum gI_xI_y \\ \sum gI_xI_y & \sum gI_y^2 \end{pmatrix} \mathbf{w}(\mathbf{x}) = \begin{pmatrix} \sum gI_tI_x \\ \sum gI_tI_y \end{pmatrix} \quad . \tag{2.5}$$

Let M be the 2×2 matrix on the left. When $rank(M) = 2$, the least-squares estimate is given by inverting Equation 2.5. This is when the local image structure over the neighborhood contains sufficient information for solving the aperture problem. The optical flow can thus be estimated in a sparse set of points where the problem is well-posed. If a dense field is needed, one can increase the size of the neighborhood at the cost of assuming a constant displacement over a larger area. This assumption is not valid for realistic videos. Various modifications have thus been proposed over the years, *e.g.* using a coarse-to-fine scheme, more robust penalties or layers. We refer to [Baker and Matthews, 2004] for an overview of them.

2.1.3 Global approach

A dense optical flow field is required in many applications. In addition, local approaches can not accurately estimate the flow in a number of the image regions such as homogeneous areas. To overcome these limitations, global approaches were proposed. They leverage a smoothness term to propagate the flow in poorly textured areas. The first global approach was proposed in the seminal work of Horn and Schunck [1981]. An energy E, sum of a data term E_{data}, that penalizes the violation of the optical flow constraint, and of a smoothness term E_{smooth}, that penalizes a strong optical flow gradient, is minimized:

$$E(\mathbf{w}) = \int_\Omega E_{data}(\mathbf{w}) + \alpha E_{smooth}(\mathbf{w}) d\mathbf{x} \quad . \tag{2.6}$$

Horn and Schunck [1981] use a L2 penalty of the flow gradient as smoothness term $E_{smooth}(\mathbf{w}) = \|\nabla_2 \mathbf{w}\|_2^2$ and a L2 penalty of the optical flow constraint as data term $E_{data}(\mathbf{w}) = ((\nabla_3^\top I)\mathbf{W})^2 = \mathbf{W}^\top \mathbb{J}_0 \mathbf{W}$ with $\mathbb{J}_0$ being the tensor defined by $\mathbb{J}_0 = (\nabla_3 I)(\nabla_3^\top I)$. This global energy is minimized using a variational formulation most of the time, *i.e.* the goal is to find the extrema of a functional. Variational methods have been widely used over the years and we review related work in the next section. Our proposed optical flow models, DeepFlow and EpicFlow, both involve a variational formulation.

2.2 Variational approaches

Variational models are based on the minimization of a global energy, composed of a data term and a smoothness term. We review in this section formulations for both terms, as well as the minimization techniques.

2.2.1 Data term

The data term measures the consistency of the optical flow with respect to the input images. Horn and Schunck [1981] use a quadratic penalization of the optical flow constraint, *i.e.*, the linearization of the brightness constancy assumption. Nevertheless, this assumption is often violated, for instance in case of illumination changes or moving shadows. The data term has been improved over the years by adding other constancy assumptions and by using robust penalizers.

Constancy assumption. When dealing with multi-band images, the brightness constancy assumption can be replaced by a color constancy assumption. A straightforward way of dealing with color images is to sum over channels the constancy term of each color. More sophisticated models [Markandey and Flinchbaugh, 1990, Golland and Bruckstein, 1997, Zimmer et al., 2009] with multi-banded images have been introduced. For instance, Zimmer et al. [2009] use separate norms and gradients in the HSV colorspace.

Instead of relying on brightness or color constancy, it is also possible to consider any features based on the images. Such approaches have already been proposed in the 1980s [Burt et al., 1983, Anandan, 1989]. More recently, Brox et al. [2004] have proposed to use a constancy of the gradient, combined with the classical color constancy assumption. The gradient has the advantage of being more robust to illumination changes. Nevertheless, such constraints assume that the flow is locally translational. For instance, the gradient will change in case of rotation or scaling, in contrast to the color.

More complex constancy assumption can be used, as for instance the constancy of the Hessian. Papenberg et al. [2006] and Vogel et al. [2013a] perform a comparison of most of them, see Table 2.1. More recently, more robust features have been proposed such as SIFT descriptor [Liu et al., 2011] or the census transform [Zabih and Woodfill, 1994, Stein, 2004b, Müller et al., 2011, Hafner et al., 2013] which has shown excellent performance, in particular in case of illumination changes.

constancy assumption	intensity function
brightness	$I_1(\mathbf{x}) = I_2(\mathbf{x} + \mathbf{w}(\mathbf{x}))$
gradient	$\nabla_2 I_1(\mathbf{x}) = \nabla_2 I_2(\mathbf{x} + \mathbf{w}(\mathbf{x}))$
Hessian	$\mathcal{H}(I_1(\mathbf{x})) = \mathcal{H}(I_2(\mathbf{x} + \mathbf{w}(\mathbf{x})))$
Laplacian	$\Delta I_1(\mathbf{x}) = \Delta I_2(\mathbf{x} + \mathbf{w}(\mathbf{x}))$
norm of the gradient	$\|\nabla_2 I_1(\mathbf{x})\| = \|\nabla_2 I_2(\mathbf{x} + \mathbf{w}(\mathbf{x}))\|$
norm of the Hessian	$\|\mathcal{H}(I_1(\mathbf{x}))\| = \|\mathcal{H}(I_2(\mathbf{x} + \mathbf{w}(\mathbf{x})))\|$
determinant of the Hessian	$\det\mathcal{H}(I_1(\mathbf{x})) = \det\mathcal{H}(I_2(\mathbf{x} + \mathbf{w}(\mathbf{x})))$
census transform	$\mathcal{C}(I_1(\mathbf{x})) = \mathcal{C}(I_2(\mathbf{x} + \mathbf{w}(\mathbf{x})))$

Table 2.1 – Variants of the constancy assumptions [Papenberg et al., 2006, Vogel et al., 2013a].

The brightness constancy assumption may also be enhanced to incorporate an illumination change model. This can be done by explicitly estimating the changes: $g(\mathbf{x})I_1(\mathbf{x}) = I_2(\mathbf{x} + \mathbf{w}(\mathbf{x})) + b(\mathbf{x})$ where g and b are two additional unknowns. The problem is thus even more under-constrained, with 4 unknowns per pixel. Nevertheless, by adding constraints on the illumination changes, the problem can still be solved [Seitz and Baker, 2009, Negahdaripour, 1998]. Such a formulation can be generalized to model blur [Seitz and Baker, 2009].

Robust penalizer. The optical flow constraint provides one error per pixel. Aggregating them over all pixels can be done in multiple ways. In their seminal work, Horn and Schunck [1981] use the L2 norm: $E_{data} = \int_\Omega \|\mathbf{W}_T \mathbb{J}_0 \mathbf{W}\|_2^2 d\mathbf{x}$. This implicitly assumes that the errors are Gaussian and iid. This assumption is violated most of the time, particularly for occluded pixels, *i.e.*, regions that are present only in one of the two images. Robust penalization, *i.e.* different from the L2 norm with less importance given to outliers, has been proposed, leading to the following data term: $E_{data} = \int_\Omega \Psi_{data}(\mathbf{W}_T \mathbb{J}_0 \mathbf{W}) d\mathbf{x}$, with Ψ_{data} the robust penalizer. A common choice [Brox et al., 2004, Wedel et al., 2009b] is to use the L1 norm, or its differentiable approximation, the Charbonnier penalizer: $\Psi_{data}(s^2) = \sqrt{s^2 + \epsilon^2}$ with ϵ being a small constant. Compared to the L2 norm, L1 norm gives less importance to outliers. Other penalizers have been proposed such as Lorentzian [Black and Anandan, 1996], generalized Charbonnier [Sun et al., 2014b] or Huber-L1 [Werlberger et al., 2009]. Figure 2.2 compares the different penalizers.

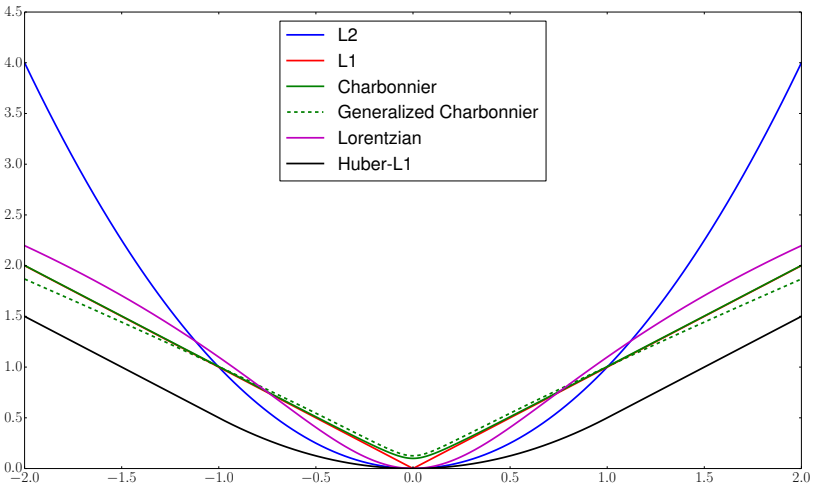


Figure 2.2 – Variants of robust penalizers.

2.2.2 Regularization term

The regularization term encodes some priors which add constraints to the under-constrained data term, thus making the problem solvable. A common choice is a smoothness term which is based on flow gradient, *i.e.*, that favors flow with small derivatives.

Horn and Schunck [1981] use a L2 penalty on the flow gradient: $E_{smooth}(\mathbf{w}) = \int_\Omega \|\nabla_2 \mathbf{w}(\mathbf{x})\|_2^2 d\mathbf{x}$. Again, such a formulation that assumes Gaussian and iid distribution is often violated. This is why more robust penalizers have been proposed, such as a Lorentzian penalty [Black and Anandan, 1996] or the L1 norm and its approximation [Brox et al., 2004, Wedel et al., 2009b]. In this case, the formulation becomes a Total Variation (TV) method. Roth and Black [2007] show that logarithmic penalties have probabilistic interpretations related to the distribution of flow derivatives.

Even with a robust penalizer, the flow still tends to be oversmooth at boundaries. Consequently, adding a spatial weighting to the penalty has been proposed. A typical example is a weight that depends on image gradient. This can be used to reduce the weight near image edges, thus encoding the fact that flow discontinuities mainly appear at image edges. Regularization with weights based on image statistics is called image-driven. Other variants that are oversegmentation-driven [Seitz and Baker, 2009], flow-driven [Wedel et al., 2009a] or data-driven [Zimmer et al., 2009] have been proposed.

The weighting function can also depends on a direction in addition to the spatial position, resulting in an anisotropic smoothness term. For instance,

Nagel and Enkelmann [1986] and Werlberger et al. [2009] give less weight to the direction along the image gradient less than the direction orthogonal to it.

Instead of relying on flow gradient, higher-order constraints can be used. For instance, Anandan and Weiss [1985] and Trobin et al. [2008] favor flow with small second-order derivatives. In the same spirit, Ranftl et al. [2014] apply the generalized total variation framework to optical flow estimation. This regularization favors piecewise affine solutions. Several other constraints can be used, such as an affine over-parameterization [Nir et al., 2008] or a rigidity prior [Adiv, 1985, Wedel et al., 2009a].

In case of image sequences with more than 2 frames, the gradient of the flow can be computed in 3D [Black, 1991, Brox et al., 2004], with a separate weight for the temporal and the spatial derivatives. Nevertheless, such derivatives are not robust to large displacements. Regularizing along trajectories is thus necessary [Salgado and Sánchez, 2007, Volz et al., 2011]. In this dissertation, we restrict to the case of optical flow computation over 2 consecutive frames.

2.2.3 Coarse-to-fine scheme

Variational formulations rely on the optical flow constraint, derived from the brightness constancy assumption using a first-order Taylor expansion. This approximation is only valid for small flows. To handle larger displacements, a classical strategy consists in using a coarse-to-fine scheme. Coarse-to-fine strategy is also beneficial from a computational point of view, it was already used by Lucas and Kanade [1981] and Horn and Schunck [1981] in their pioneering work.

Figure 2.3 illustrates a coarse-to-fine scheme. Image pyramids are built by repeatingly blurring and subsampling [Lucas and Kanade, 1981, Anandan, 1989, Bruhn et al., 2005b, Black and Anandan, 1996]. Optical flow is first estimated at the coarsest scale. It is then upsampled to be used to initialize the estimate at the next level. Median filtering can optionally be applied at each upsampling: Sun et al. [2014b] show that this is equivalent to adding a non-local smoothness term to the energy. The second image is sometimes warped according to the flow, and only a flow increment with respect to the current estimation is computed. This process is repeated until the last level at original resolution. In a similar spirit, Bruhn et al. [2006] propose a multigrid strategy where the flow pass both up and down in the pyramid hierarchy.

Compared to single-level optimization, a coarse-to-fine scheme allows to avoid many local minima and to speed-up the optimization. However, it

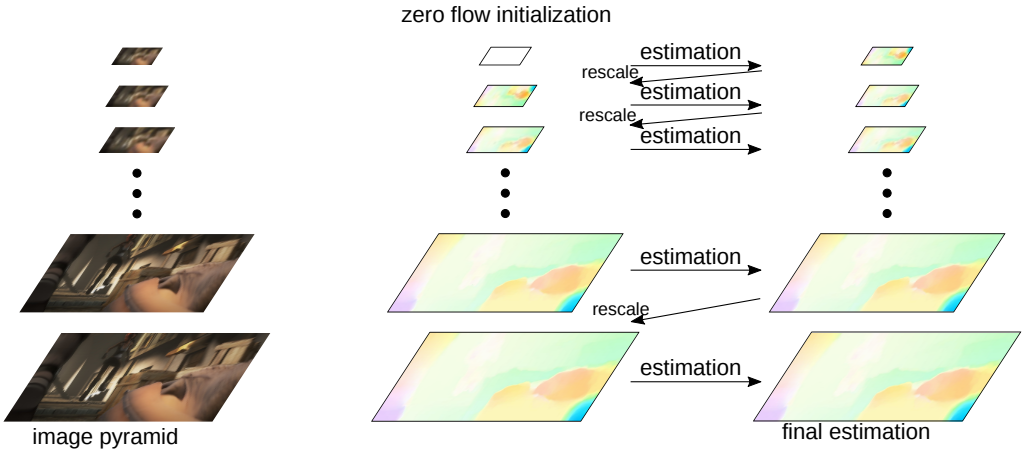


Figure 2.3 – Illustration of a coarse-to-fine scheme. The images are iteratively rescaled and blurred (left). Starting from a zero flow initialization, the optical flow is estimated at the coarsest level. This estimation is rescaled to the size of the next level where the flow is updated from this current estimation. The process is repeated until the last level at original resolution.

tends to over-smooth fine structures and fails to capture small objects with fast motion. Indeed, such objects are not visible at coarse scales and as a consequence, the initialization tends to ignore their motion.

2.2.4 Minimization

Variational methods. Consider an energy formulation where the global energy only depends on $\mathbf{x}$, $\mathbf{w}$ and its gradient $\nabla_2\mathbf{w}$:

$$E_{global} = \int_\Omega E(u, v, x, y, u_x, u_y, v_x, v_y)dxdy \quad , \tag{2.7}$$

with $u_x = \frac{\partial u}{\partial x}$ and similarly for u_y, v_x, v_y. Note that such formulation includes most energy models [Horn and Schunck, 1981, Brox et al., 2004, Nir et al., 2008, Bruhn et al., 2005b, Zimmer et al., 2009]. If the flow is treated as a 2D continuous function (and not only 2D vectors at each pixel), E_{global} can be treated as a calculus of variations, explaining the name of *variational* methods. Euler-Lagrange equations can be applied, ensuring that at the minimum:

$$\begin{cases} \frac{\partial E_{global}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial E_{global}}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial E_{global}}{\partial u_y} = 0 \quad , \\ \frac{\partial E_{global}}{\partial v} - \frac{\partial}{\partial x}\frac{\partial E_{global}}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial E_{global}}{\partial v_y} = 0 \quad . \end{cases} \tag{2.8}$$

While these equations are linear in case of Horn and Schunck [1981]'s energy, this is not the case for more general formulation. As a consequence,

a fixed point iteration is often used [Brox et al., 2004]. The flow is decomposed into $\mathbf{w} + d\mathbf{w}$ where $\mathbf{w}$ is assumed to be known (from the previous level of the coarse-to-fine scheme) and the flow update $d\mathbf{w}$ is computed. The flow update is assumed to be small, and by applying Taylor expansion on $\mathbf{w}$, a set of linear equations on $d\mathbf{w}$ is obtained. It is often solved using a Gauss-Seidel variant, namely the Successive Over Relaxation methods [Young and Rheinboldt, 1971], or a preconditioned conjugate gradient for better parallelization [Sundaram et al., 2010]. For better accuracy, the flow update is then added to the current flow estimate, and a new flow update is estimated. This process is repeated several times before moving to the next pyramid level. More details for approximating the solution are given in Appendix A.

Other approaches. Other minimization approaches rely for instance on gradient descent [Baker and Matthews, 2004, Black and Anandan, 1996] but are most of the time limited by the presence of a huge number of local minima. Another strategy [Wedel et al., 2009b, Trobin et al., 2008] consists in decoupling the flow $\mathbf{w}$ in the data term $\mathbf{w}_{data}$ and the smoothness term $\mathbf{w}_{smooth}$, with an additional penalty on the differences between these two flows:

$$E_{global} = \int_{\Omega} E_{data}(\mathbf{w}_{data}) + E_{smooth}(\mathbf{w}_{smooth}) + \gamma \|\mathbf{w}_{smooth} - \mathbf{w}_{data}\|^2 d\mathbf{x} \ . \quad (2.9)$$

This equation is then minimized iteratively for $\mathbf{w}_{data}$ and $\mathbf{w}_{smooth}$ while fixing the other one and increasing γ. As a consequence, the two flows are closed at the end. The two obtained optimization problems are simpler to solve.

2.3 Other optical flow approaches

2.3.1 Layered and segmentation-based methods

The optical flow field can be segmented into regions with coherent motions, referred to as layers [Darrell and Pentland, 1995, Wang and Adelson, 1994]. Several works have considered the joint estimation of motion layers and their motions [Brox et al., 2006, Sun et al., 2013, Unger et al., 2012], in some cases including a depth ordering of the layer. Most methods rely on locally-connected Markov Random Fields (MRF), resulting in a poor segmentation. More recently, Sun et al. [2013] propose a fully-connected MRF but limit the labeling to 2 layers. Note that the task of jointly segmenting

the video into layers and estimating their motions is extremely challenging, and yields to a complex minimization of non-convex energy functions. As a consequence, the estimation is unreliable for difficult, yet common cases, such as videos with fast motion, large displacements or compression artifacts. Moreover, the computational time is usually extremely high, thus limiting the practical use of layered methods.

2.3.2 Learning-based methods

The data term in the energy formulation relies on various choices such as the constancy assumptions and the penalizer. Learning techniques have been applied to justify designs and parameters choices [Roth and Black, 2007, Sun et al., 2008]. Training data are used to build probabilistic models of the optical flow.

Another learning approach was proposed by Wulff and Black [2015] based on principal component analysis (PCA). From a training set, a principal component basis is fitted. At test time, a set of sparse matches are computed and the method then estimates dense flow by estimating the weights of the PCA components that best fit the sparse matches.

More recently, Convolutional Neural Network (CNN) has been used to learn to estimate optical flow [Dosovitskiy et al., 2015, Teney and Hebert, 2016]. FlowNet architecture is either based either on computing convolutions on the concatenated images, or on a siamese networks that mimic feature extraction, followed by a correlational layer. Both architectures obtain a similar performance. Deconvolution architecture is also used followed by a variational refinement to obtain the final estimation. The network is trained using an artificial dataset with moving chairs added on top of images. Teney and Hebert [2016] rely on 3D convolutions and add invariance properties using various normalizations. The network is trained from scratch (even on small datasets) to learn to classify the flow vector at a pixel among a fixed set of candidates, and then fine-tuned with a Euclidean loss on the flow.

2.3.3 Discrete optimization

Discrete optimization techniques such as graph-cut or belief propagation has also been applied to optical flow estimation. Some algorithms assume that a set of flow candidates are available for each pixel. The flow is then defined as an assignment to a candidate at every pixel. For instance, FusionFlow [Lempitsky et al., 2008] use multiple binary graph-cuts problem to refine the current flow estimate. More recently, Menze et al. [2015] propose

to use CRF to obtain a pixel-accurate flow field, which is then refined using a variational method. The CRF is made tractable by restricting to flow candidates, by using block coordinate descent for optimization and by leveraging the structure of the regularizer. Chen and Koltun [2016] show that discrete optimization can be applied to optical flow estimation without pruning the flow candidates despite the large space of possible displacements.

2.4 Image matching in optical flow estimation

Image matching techniques have recently been integrated in optical flow formulation, mainly to give more robustness to the case of small fast-moving objects [Brox and Malik, 2011] for which variational methods with coarse-to-fine schemes tend to miss their motions. Optical flow can be viewed as a dense matching approach between consecutive frames of a video. In particular, our proposed flow approaches, DeepFlow and EpicFlow, rely on a proposed matching algorithm named DeepMatching. In this section, we review related work on image matching techniques as well as their integration in flow formulation.

2.4.1 Image matching

Image matching based on local features has been extensively studied in the past decade. It has been applied successfully to various domains, such as wide baseline stereo matching [Furukawa et al., 2010] or image retrieval [Philbin et al., 2010]. It consists of two steps: extracting local descriptors and matching them. Image descriptors are extracted in rigid (generally square) local areas at sparse invariant image locations [Mikolajczyk et al., 2005, Szeliski, 2010] or on a dense grid [Wills et al., 2006, Tola et al., 2008, Brox and Malik, 2011]. Matching is then performed by nearest neighbor search between descriptors, followed by an optional geometric verification. Note that a confidence value can be obtained by computing the uniqueness of a match, *i.e.*, by looking at the distance of its nearest neighbors [Lowe, 2004, Brox and Malik, 2011]. Despite their excellent performance for matching well-textured rigid objects, local descriptors are not reliable for non-rigid objects and weakly textured regions.

Recently, fast algorithms for dense patch matching have taken advantage of the redundancy between overlapping patches [Barnes et al., 2010, Korman and Avidan, 2011, Sun, 2012, Yang et al., 2014]. The main idea is to propagate good matches to their neighborhood in a loose fashion, yielding dense non-rigid matches. In practice, however, the lack of a smoothness

constraint leads to highly discontinuous matches. Several works have proposed ways to fix this. HaCohen et al. [2011] reinforce neighboring matches using an iterative multiscale expansion and contraction strategy, performed in a coarse-to-fine manner. Yet, the algorithm matches poorly discriminative patches and, as such, cannot overcome the inherent weaknesses of patch matching approaches. Yang et al. [2014] include a guided filtering stage on top of PatchMatch, which obtains smooth correspondence fields by locally approximating a MRF. Finally, Kim et al. [2013] propose a hierarchical matching to obtain dense correspondences, using a coarse-to-fine (top-down) strategy. Loopy belief propagation is used to perform inference.

2.4.2 Integration of matching in optical flow estimation

In SIFT-Flow, Liu et al. [2011] leverage a data-term based on SIFT matching in a global energy formulation. This energy is non-differentiable and the optimization is performed using belief propagation. SIFT-Flow has been mainly developed for the application of matching scenes. Hassner et al. [2012] improve over SIFT-flow by using multi-scale patches. However, this decreases performance when scale invariance is not required.

For optical flow formulation, Brox and Malik [2011] propose to integrate matching into the energy by adding a matching term. This new term penalizes the difference between precomputed matches and the flow estimation, thus guiding the flow. Brox and Malik [2011] use HOG descriptors [Dalal and Triggs, 2005] with a reciprocal nearest-neighbor verification to prune most of the false matches. Xu et al. [2012] integrate matching of SIFT [Lowe, 2004] and PatchMatch [Barnes et al., 2010] to refine the flow initialization at each level. Promising results are obtained for optical flow estimation, yet at the cost of expensive fusion steps [Lempitsky et al., 2008]. Braux-Zin et al. [2013] used segment features in addition to keypoints. DeepFlow proposes an efficient and competitive approach for large displacement optical flow by integrating the proposed DeepMatching algorithm into the approach of Brox and Malik [2011].

Several works have also extended PatchMatch [Barnes et al., 2010] for optical flow estimation. Lu et al. [2013] propose a variant of PatchMatch, which uses SLIC superpixels [Achanta et al., 2012] as basic blocks in order to better respect image boundaries. The purpose is to produce a nearest-neighbor-field (NNF) which is later translated into a flow. Similarly, Chen et al. [2013] propose to compute an approximate NNF, and then estimate the dominant motion patterns using RANSAC. Next, they use a multi-

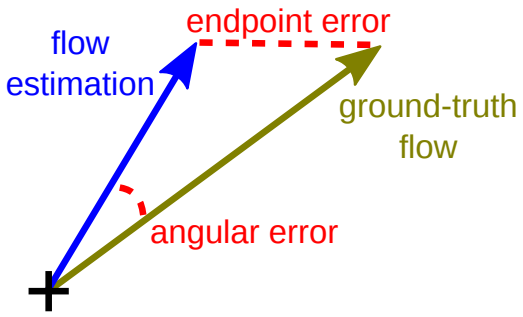


Figure 2.4 – Illustration of optical flow evaluation metrics for a given pixel.

label graph-cut to solve the assignment of each pixel to a motion pattern candidate. Their multi-label optimization can be interpreted as a motion segmentation problem or as a layered model [Sun et al., 2010]. These problems are hard and a small error in the assignment can lead to large errors in the resulting flow.

2.5 Datasets and evaluation

In this section, we review the metrics and the datasets used for evaluating optical flow estimation.

2.5.1 Metrics

To evaluate optical flow, we use the average endpoint error over all pixels, denoted as EPE. The endpoint error is the L2 norm of the difference between the estimated optical flow and the ground-truth field, see Figure 2.4. More precisely, if the flow estimation is denoted as $\mathbf{w}$ and the ground-truth flow as $\mathbf{w}_{gt}$, the EPE is defined by:

$$\frac{1}{|\Omega|}\sum_{\mathbf{x}\in\Omega}\|\mathbf{w}(\mathbf{x})-\mathbf{w}_{gt}(\mathbf{x})\|_2 \quad . \tag{2.10}$$

We also measure EPE on a subset of pixels of interest. For instance, the 's10-40' variant measures the EPE only for pixels with a ground-truth displacement between 10 and 40 pixels, and likewise for 's0-10' and 's40+'. We measure EPE-occ, which is the EPE over occluded pixels, and similarly EPE-noc for non-occluded pixels. In all cases, scores are averaged over all corresponding pixels in the dataset to obtain the final evaluation result.

On Kitti, it is standard to report the ratio of missed flow, *i.e.*, the ratio of pixels for which the endpoint error is over a threshold δ:

$$\frac{1}{|\Omega|}\sum_{\mathbf{x}\in\Omega}\mathbb{1}_{\|\mathbf{w}(\mathbf{x})-\mathbf{w}_{gt}(\mathbf{x})\|_2<\delta} \quad . \tag{2.11}$$

'Out-All 3' denotes the ratio of pixels with an error over 3 pixels over all pixels and 'Out-Occ 3' is only computed for occluded pixels.

On Middlebury, the average angular error (AAE) is also reported. This is the average over all pixels of the angular difference between the estimation and the ground-truth:

$$\frac{1}{|\Omega|}\sum_{\mathbf{x}\in\Omega}\arccos\left(\mathbf{W}(\mathbf{x}),\mathbf{W}_{gt}(\mathbf{x})\right) \quad . \tag{2.12}$$

2.5.2 Datasets

We use three datasets for evaluating optical flow: Middlebury, Kitti and MPI-Sintel. Note that for all these datasets, the publicly available data on the test sets are restricted to images and the evaluation runs on a server. Figure 2.5 shows a few frames from each one. We now present them in more detail.

- The **Middlebury** dataset [Baker et al., 2011] contains a training set of 12 short sequences with the ground-truth flow for the middle frame. The test consists of 12 other short sequences. The ground-truth flow is obtained from hidden texture or by using synthetic data. The dataset contains complex motions, but most of the displacements are small. Less than 3% of the pixels have a motion over 20 pixels, and no motion exceeds 25 pixels (training set). Consequently, optical flow performance is near saturation with an EPE around 0.2 pixels for the best performing method.
- The **Kitti** dataset [Geiger et al., 2013] contains real-world sequences taken from a driving platform. The dataset includes non-Lambertian surfaces, different lighting conditions, a large variety of materials and large displacements. More than 16% of the pixels have motion over 20 pixels. We use the version released in 2012; a newer version has been published in 2015. There are 194 training and 195 test image pairs in the dataset. The scene remains limited to the static case and most of the motion fields correspond to a zoom effect, *i.e.*, a car moving forward in the street.
- The **MPI-Sintel** dataset [Butler et al., 2012] is a challenging evaluation benchmark for optical flow estimation, constructed from realistic computer-animated films. The dataset contains 23 training and 12 test sequences with large motions and specular reflections. Each sequence contains up

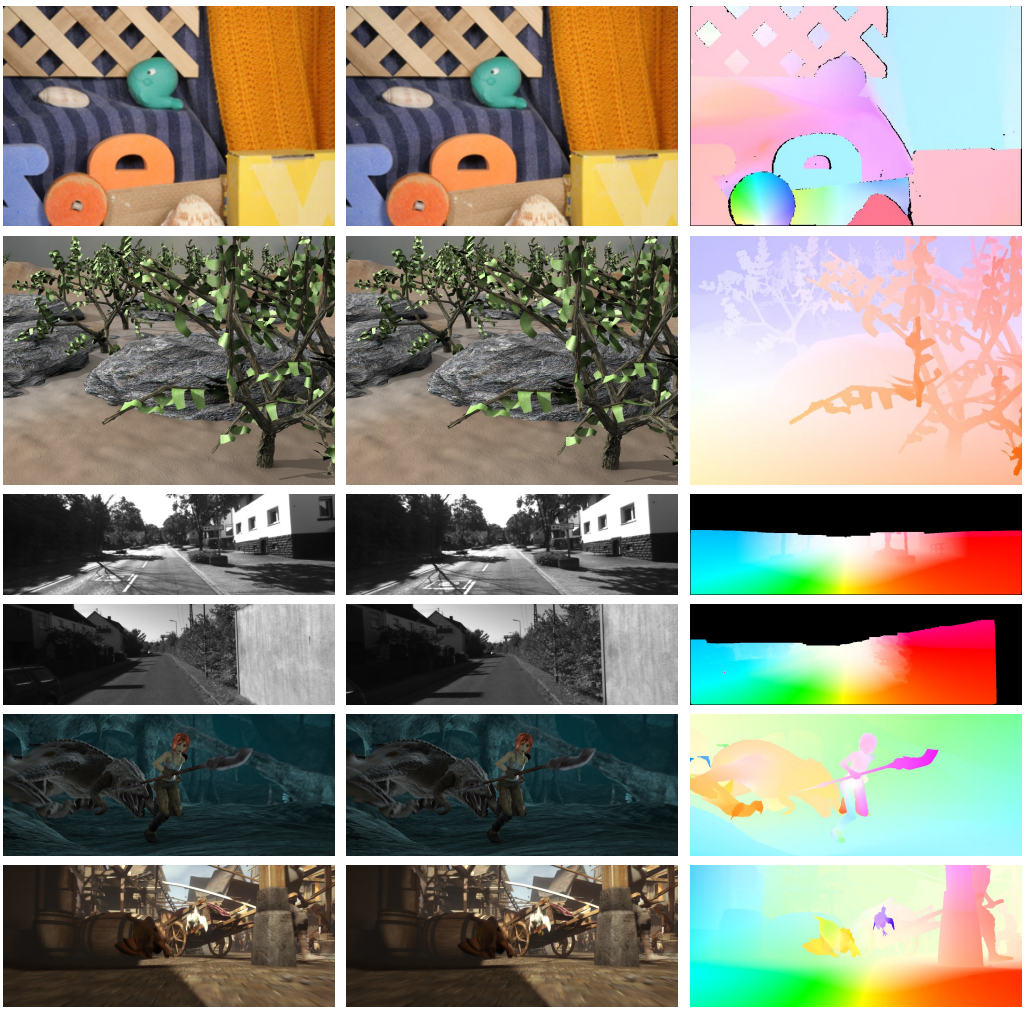


Figure 2.5 – Examples of image pairs (first two columns) and ground-truth flow (right column) with two examples from the Middlebury dataset (two first rows), the Kitti dataset (two middle rows) and the MPI-Sintel dataset (two last rows).

to 50 images. In the training set, more than 17.5% of the pixels have a motion over 20 pixels, approximately 10% over 40 pixels. The images are available in two rendering versions: 'clean' and 'final'. We use this latter one which adds realistic rendering effects such as motion blur, defocus blur and atmospheric effects. The scene also contains non-rigid objects such as characters.

Chapter 3

DeepFlow: Large Displacement Optical Flow with DeepMatching

Contents

3.1 Introduction

Although variational methods have shown promising results on small displacements [Horn and Schunck, 1981, Sun et al., 2014b], recent efforts have tried to address the more challenging case of estimating optical flow in realistic videos with fast motion [Brox and Malik, 2011, Xu et al., 2012].

Such approaches integrate precomputed matches [Dalal and Triggs, 2005, Liu et al., 2011, Barnes et al., 2010] into the variational formulation or the optimization scheme. However, these matching approaches are limited either to rigid patches [Dalal and Triggs, 2005], thus failing to estimate motion with weak or repetitive textures as well as non-rigid object, or to propagation-based approaches [Barnes et al., 2010] which are based on small patches, making repetitive textures beyond the reach of these methods.

In this chapter we introduce DeepFlow, a variational optical flow approach, that integrates a novel matching approach, called DeepMatching, using a similar formulation as Brox and Malik [2011]. DeepMatching gracefully combines the strengths of rigid matching and propagation-based algorithms using a multi-layer architecture which breaks down patches into a hierarchy of sub-patches. This architecture allows to work at several scales and handles repetitive textures. Furthermore, within each layer, local matches are computed assuming a restricted set of feasible rigid deformations. Local matches are then propagated up the hierarchy, which progressively discards spurious incorrect matches. We called the proposed matching algorithm DeepMatching, as it is inspired by deep convolutional approaches.

This chapter presents the three following contributions:

- Dense matching: we propose a matching algorithm, DeepMatching, that allows to robustly determine dense correspondences between two images. It explicitly handles non-rigid deformations, with bounds on the deformation tolerance, and incorporates a multi-scale scoring of the matches, making it robust to repetitive or weak textures. Furthermore, our approach is based on gradient histograms, and is thus robust to appearance changes caused by illumination and color variations.
- Fast, scale/rotation-invariant matching: we propose a computationally efficient version of DeepMatching, which performs almost as well as exact DeepMatching, but at a much lower memory cost. Furthermore, this fast version of DeepMatching can be extended to a scale and rotation-invariant version, making it an excellent competitor to state-of-the-art descriptor matching approaches.
- DeepFlow: we propose an optical flow approach which uses DeepMatching in the matching term of the large displacement variational energy minimization of Brox and Malik [2011]. We show that DeepMatching is a better choice compared to the HOG descriptor used by Brox and Malik [2011] and other state-of-the-art matching algorithms. The approach, named DeepFlow, obtains competitive results on public optical flow benchmarks.

Closest references. DeepFlow is based on the same formulation as Brox and Malik [2011] to integrate matches into a variational model. Instead of using rigid matches, the proposed matching algorithm, DeepMatching, is inspired by non-rigid 2D warping and deep convolutional networks [LeCun et al., 1998a, Uchida and Sakoe, 1998, Keysers et al., 2007]. This family of approaches explicitly models non-rigid deformations. We employ a novel family of feasible warpings that does not enforce monotonicity nor continuity constraints, in contrast to traditional 2D warping [Uchida and Sakoe, 1998, Keysers et al., 2007]. This makes the problem much less expensive in term of computational cost.

It is also worthwhile to mention the similarity with non-rigid matching approaches developed for a broad range of applications. Ecker and Ullman [2009] propose an approach related to ours. Their method computes a hierarchical alignment of image sub-parts in a bottom-up fashion using dynamic programming. The minimal hierarchical matching cost is then returned as a global similarity score. Our approach goes further and produces pixel-level correspondences by backtracking high-level patch matches. For the purpose of establishing dense correspondences between images, Wills et al. [2006] estimated a non-rigid matching by robustly fitting smooth parametric models (homography and splines) to local descriptor matches. In contrast, DeepMatching is non-parametric and model-free.

Unlike dense patch matching algorithms based on propagation [Barnes et al., 2010, Korman and Avidan, 2011, Sun, 2012, Yang et al., 2014], DeepMatching proceeds bottom-up first and top-down second. Due to its hierarchical nature, DeepMatching is able to consider patches at several scales, thus overcoming the lack of distinctiveness that affects small patches. The multi-layer construction also allows to efficiently perform matching with semi-rigid local deformations. In addition, DeepMatching can be computed efficiently, and can be further accelerated to satisfy low-memory requirements with negligible loss in accuracy.

Outline. This chapter is organized as follows. We start by presenting the proposed matching algorithm, DeepMatching, in Section 3.2. Section 3.3 then describes two extensions of DeepMatching: an approximate and faster version of DeepMatching (Section 3.3.1), a scale and rotation invariant version (Section 3.3.2). Next, we present DeepFlow in Section 3.4, an optical flow approach that integrates DeepMatching into a variational formulation. Finally, we present experimental results in Section 3.5. In particular, we show that DeepMatching obtains excellent matching performance and can be effectively integrated into an optical flow approach. DeepFlow outper-

formed the state of the art at publication time on challenging optical flow benchmarks.

Preliminary version. A preliminary version of this work has appeared in ICCV'13 [Weinzaepfel et al., 2013]. The version presented in this dissertation, that will appear in IJCV [Revaud et al., 2016], adds (1) an in-depth presentation of DeepMatching; (2) an enhanced version of DeepMatching, which improves the match scoring and the selection of entry points for backtracking; (3) proofs on time and memory complexity of DeepMatching as well as its deformation tolerance; (4) a discussion on the connection between Deep Convolutional Neural Networks and DeepMatching; (5) a fast approximate version of DeepMatching; (6) a scale and rotation invariant version of DeepMatching; and (7) an extensive experimental evaluation of DeepMatching on several state-of-the-art benchmarks. The code for DeepMatching as well as DeepFlow are available at `http://lear.inrialpes.fr/src/deepmatching/` and `http://lear.inrialpes.fr/src/deepflow/`. Note that we provide a GPU implementation in addition to the CPU one.

3.2 DeepMatching

This section introduces our matching algorithm DeepMatching. DeepMatching is a matching algorithm based on correlations at the patch-level, that proceeds in a multi-layer fashion. The multi-layer architecture relies on a quadtree-like patch subdivision scheme, with an extra degree of freedom to locally re-optimize the positions of each quadrant. In order to enhance the contrast of the spatial correlation maps output by the local correlations, a non-linear transformation is applied after each layer.

We first give an overview of DeepMatching in Section 3.2.1 and show that it can be decomposed in a bottom-up pass followed by a top-down pass. We, then, present the bottom-up pass in Section 3.2.2 and the top-down one in Section 3.2.3. Finally, we analyze DeepMatching in Section 3.2.4.

3.2.1 Overview of the approach

A state-of-the-art approach for matching regions between two images is based on the SIFT descriptor [Lowe, 2004]. SIFT is a histogram of gradients with 4×4 spatial and 8 orientation bins, yielding a robust descriptor $\mathbf{R} \in \mathbb{R}^{4\times4\times8}$ that effectively encodes a square image region. Note that its 4×4 cell grid can also be viewed as 4 so-called 'quadrants' of 2×2 cells, see Figure 3.1. We can, then, rewrite $\mathbf{R} = [\mathbf{R}_0, \mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3]$ with $\mathbf{R}_i \in \mathbb{R}^{2\times2\times8}$.

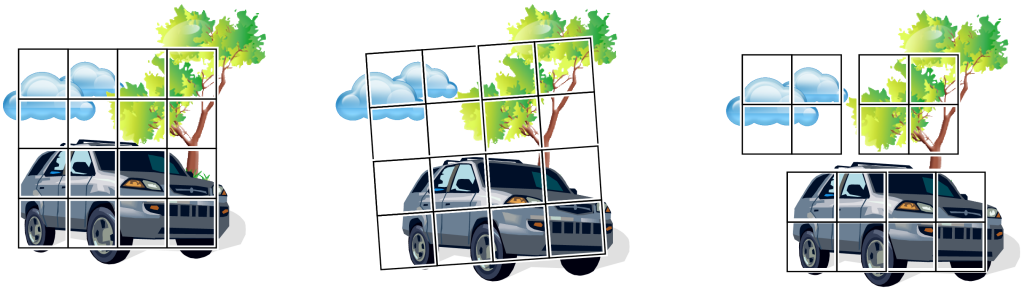


Figure 3.1 – Illustration of moving quadrant similarity: a quadrant is a quarter of a SIFT patch, *i.e.*, a group of 2×2 cells. *Left:* SIFT descriptor in the first image. *Middle:* second image with optimal standard SIFT matching (rigid). *Right:* second image with optimal *moving quadrant* SIFT matching. In this example, the patch covers various objects moving in different directions: for instance the car moves to the right while the cloud to the left. Rigid matching fails to capture this, whereas the moving quadrant approach is able to follow each object.

Let $\mathbf{R}$ and $\mathbf{R}'$ be the SIFT descriptors of the corresponding regions in the source and target image. In order to remove the effect of non-rigid motion, we propose to optimize the *positions* $p_i \in \mathbb{R}^2$ of the 4 quadrants of the target descriptor $\mathbf{R}'$ (rather than keeping them fixed), in order to maximize

$$\mathrm{sim}(\mathbf{R}, \mathbf{R}') = \max_{\{p_i\}} \frac{1}{4} \sum_{i=0}^{3} \mathrm{sim}\left(\mathbf{R}_i, \mathbf{R}'_i(p_i)\right) \quad , \tag{3.1}$$

where $\mathbf{R}'_i(p_i)$ is the descriptor of a single quadrant extracted at position p_i and sim() a similarity function. Now, $\mathrm{sim}(\mathbf{R}, \mathbf{R}')$ is able to handle situations such as the one presented in Figure 3.1, where a region contains multiple objects moving in different directions. Furthermore, if the four quadrants can move independently (of course, within some extent), it can be calculated more efficiently as:

$$\mathrm{sim}(\mathbf{R}, \mathbf{R}') = \frac{1}{4} \sum_{i=0}^{3} \max_{p_i} \mathrm{sim}\left(\mathbf{R}_i, \mathbf{R}'_i(p_i)\right) \quad , \tag{3.2}$$

When applied recursively to each quadrant by subdividing it into 4 sub-quadrants until a minimum patch size is reached (atomic patches), this strategy allows for accurate non-rigid matching. Such a recursive decomposition can be represented as a quad-tree, see Figure 3.2. Given an initial pair of two matching regions, retrieving atomic patch correspondences is

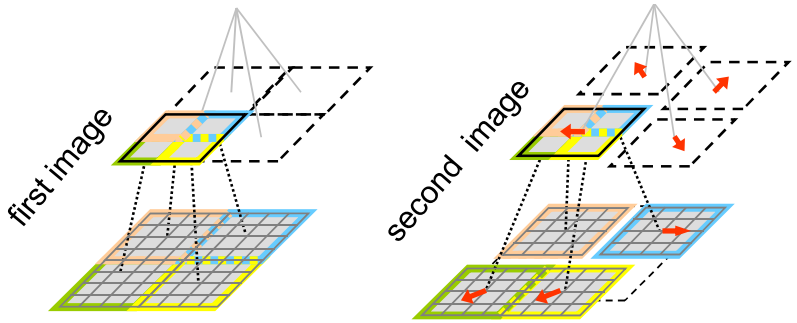


Figure 3.2 – *Left:* Quadtree-like patch hierarchy in the first image. *Right:* one possible displacement of corresponding patches in the second image.

then done in a top-down fashion (*i.e.*, by recursively applying Equation 3.2 to the quadrant's positions $\{p_i\}$).

Nevertheless, in order to first determine the set of matching regions between the two images, we need to compute beforehand the matching scores (*i.e.*, similarity) of all large-enough patches in the two images (as in Figure 3.1), and keep the pairs with maximum similarity. As indicated by Equation 3.2, the score is formed by averaging the max-pooled scores of the quadrants. Hence, the process of computing the matching scores is bottom-up. In the following, we call *correlation map* the matching scores of a single patch from the first image at every position in the second image. Selecting matching patches then corresponds to finding local maxima in the correlation maps.

To sum-up, the algorithm can be decomposed in two steps: (i) first, correlation maps are computed using a bottom-up algorithm, as shown in Figure 3.6. Correlation maps of small patches are first computed and then aggregated to form correlation maps of larger patches; (ii) next, a top-down method estimates the motion of atomic patches starting from matches of large patches.

In the remainder of this section, we detail the two steps described above (Section 3.2.2 and Section 3.2.3), before analyzing the properties of DeepMatching in Section 3.2.4.

3.2.2 Bottom-up correlation pyramid computation

Let I and I' be two images of resolution $W \times H$ and $W' \times H'$.

Bottom level. We use patches of size 4×4 pixels as atomic patches. We split I into non-overlapping atomic patches, and compute the correlation

map with image I' for each of them, see Figure 3.3. The score between two atomic patches $\mathbf{R}$ and $\mathbf{R}'$ is defined as the average pixel-wise similarity:

$$\text{sim}(\mathbf{R}, \mathbf{R}') = \frac{1}{16} \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{R}_{i,j}^{\top} \mathbf{R}'_{i,j} \quad , \tag{3.3}$$

where each pixel $\mathbf{R}_{i,j}$ is represented as a histogram of oriented gradients pooled over a local neighborhood. We detail below how the pixel descriptor is computed.

Pixel descriptor $\mathbf{R}_{i,j}$. We rely on a robust pixel representation that is similar in spirit to SIFT and DAISY [Lowe, 2004, Tola et al., 2010]. Given an input image I, we first apply a Gaussian smoothing of radius ν_1 in order to denoise I from potential artifacts caused for example by JPEG compression. We then extract the gradient $(\delta x, \delta y)$ at each pixel and compute its non-negative projection onto 8 orientations $\left\{(\cos i\frac{\pi}{4}, \sin i\frac{\pi}{4})\right\}_{i=1\ldots8}$. At this point, we obtain 8 oriented gradient maps. We smooth each map with a Gaussian filter of radius ν_2. Next we cap strong gradients using a sigmoid $x \mapsto 2/(1+\exp(-\varsigma x)) - 1$, to help canceling out effects of varying illumination. We smooth gradients one more time for each orientation with a Gaussian filter of radius ν_3. Finally, the descriptor for each pixel is obtained by the ℓ_2-normalized concatenation of 8 oriented gradients and a ninth small constant value μ. Appending μ amounts to adding a regularizer that will reduce the importance of small gradients (*i.e.*, noise) and ensures that two pixels lying in areas without gradient information will still correlate positively. Pixel descriptors $\mathbf{R}_{i,j}$ are compared using dot-product and the similarity function takes value in the interval $[0, 1]$. In Section 3.5.2, we evaluate the impact of the parameters of this pixel descriptor.

Bottom-level correlation map. We can express the correlation map computation obtained from Equation 3.3 more conveniently in a convolutional framework. Let $I_{N,\boldsymbol{p}}$ be a patch of size $N \times N$ from the first image centered at $\boldsymbol{p}$ ($N \geq 4$ is a power of 2). Let $\mathcal{G}_4 = \{2, 6, 10, ..., W-2\} \times \{2, 6, 10, ..., H-2\}$ be a grid with step 4 pixels. $\mathcal{G}_4$ is the set of the centers of the atomic patches. For each $\boldsymbol{p} \in \mathcal{G}_4$, we convolve the flipped patch $I^{\text{F}}_{4,\boldsymbol{p}}$ over I'

$$C_{4,\boldsymbol{p}} = I^{\text{F}}_{4,\boldsymbol{p}} \star I' \quad , \tag{3.4}$$

to get the correlation map $C_{4,\boldsymbol{p}}$, where $.^{\text{F}}$ denotes an horizontal and vertical flip [1]. Examples of such correlation maps are shown in Figures 3.3 and 3.4.

1. This amounts to the cross-correlation of the patch and I'.

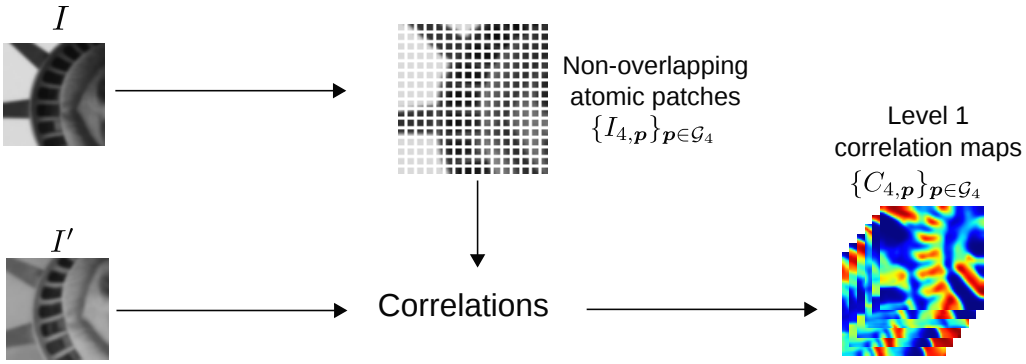


Figure 3.3 – Computing the bottom level correlation maps $\{C_{4,\boldsymbol{p}}\}_{\boldsymbol{p}\in\mathcal{G}_4}$. Given two images I and I', the first one is split into non-overlapping atomic patches of size 4×4 pixels. For each patch, we compute the correlation at every location of I' to obtain the corresponding correlation map.

Without surprise, we can observe that atomic patches are not discriminative. Recursive aggregation of patches in subsequent stages will be the key to create discriminative responses.

Iteration. We then compute the correlation maps of larger patches by aggregating those of smaller patches. As shown in Figure 3.5, a $N \times N$ patch $I_{N,\boldsymbol{p}}$ is the concatenation of 4 patches of size $N/2 \times N/2$:

$$I_{N,\boldsymbol{p}} = \left[I_{\frac{N}{2},\boldsymbol{p}+\frac{N}{4}\boldsymbol{o}_i} \right]_{i=0..3} \quad \text{with} \begin{cases} \boldsymbol{o}_0 = [-1,-1]^\top , \\ \boldsymbol{o}_1 = [-1,+1]^\top , \\ \boldsymbol{o}_2 = [+1,-1]^\top , \\ \boldsymbol{o}_3 = [+1,+1]^\top . \end{cases} \tag{3.5}$$

They correspond respectively to the bottom-left, top-left, bottom-right and top-right quadrants. The correlation map of $I_{N,\boldsymbol{p}}$ can thus be computed using its children's correlation maps. For the sake of clarity, we define the short-hand notation $\mathrm{s}_{N,i} = \frac{N}{4}\boldsymbol{o}_i$ describing the positional shift of a children patch $i \in [0,3]$ relatively to its parent patch (see Figure 3.5).

Using the above notations, we rewrite Equation 3.2 by replacing $\mathrm{sim}(\mathbf{R},\mathbf{R}') \overset{def}{=} C_{N,\boldsymbol{p}}(\boldsymbol{p}')$ (*i.e.*, assuming here that patch $\mathbf{R} = I_{N,\boldsymbol{p}}$ and that $\mathbf{R}'$ is centered at $\boldsymbol{p}' \in I'$). Similarly, we replace the similarity between children patches $\mathrm{sim}\left(\mathbf{R}_i, \mathbf{R}'_i(\boldsymbol{p}'_i)\right)$ by $C_{\frac{N}{2},\boldsymbol{p}+\mathrm{s}_{N,i}}(\boldsymbol{p}'_i)$. For each child, we retain the maximum similarity over a small neighborhood Θ_i of width and height $\frac{N}{8}$ centered at $\boldsymbol{p}' + \mathrm{s}_{N,i}$. We then obtain:

$$C_{N,\boldsymbol{p}}(\boldsymbol{p}') = \frac{1}{4}\sum_{i=0}^{3} \max_{\boldsymbol{m}'\in\Theta_i} \; C_{\frac{N}{2},\boldsymbol{p}+\mathrm{s}_{N,i}}(\boldsymbol{m}') \quad . \tag{3.6}$$

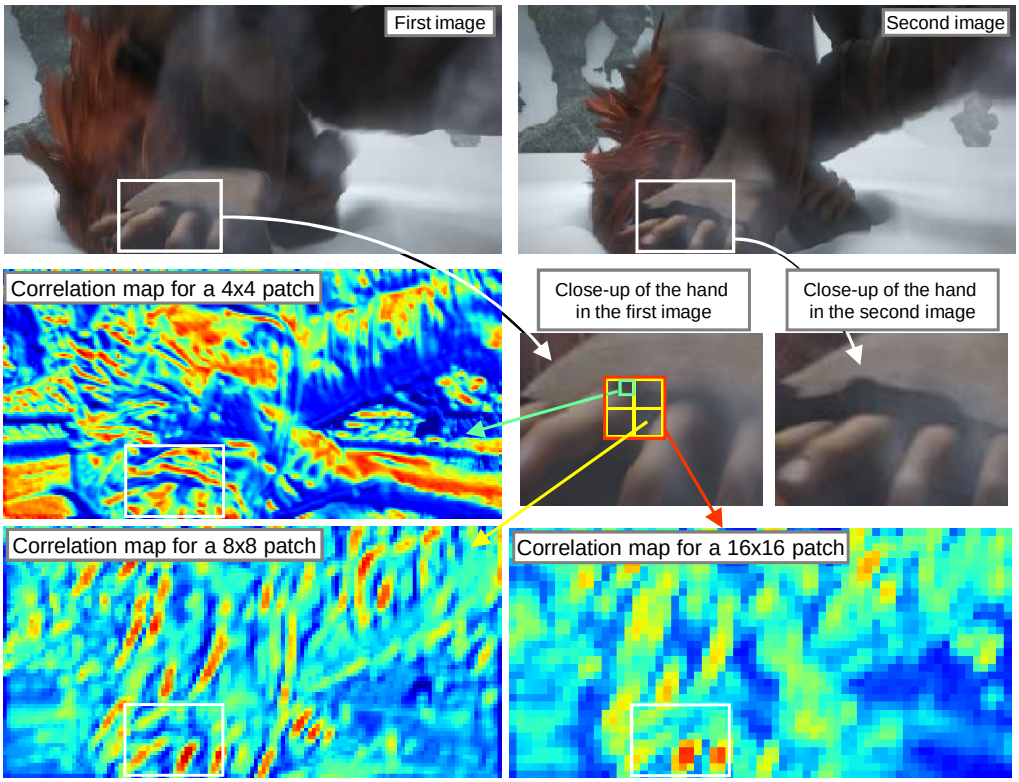


Figure 3.4 – Correlation maps for patches of different size. *Middle-left*: correlation map of a 4x4 patch. *Bottom-right*: correlation map of a 16x16 patch obtained by aggregating correlation responses of children 8x8 patches (*bottom-left*), themselves obtained from 4x4 patches. The map of the 16x16 patch is clearly more discriminative than previous ones despite the change in appearance of the region.

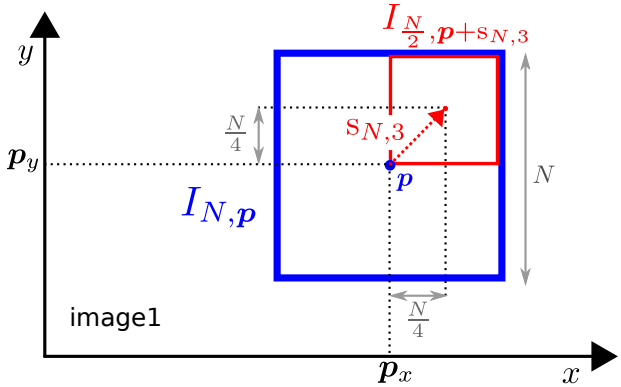


Figure 3.5 – A patch $I_{N,\boldsymbol{p}}$ from the first image (blue box) and one of its 4 quadrants $I_{\frac{N}{2},\boldsymbol{p}+\frac{N}{4}\boldsymbol{o}_3}$ (red box).

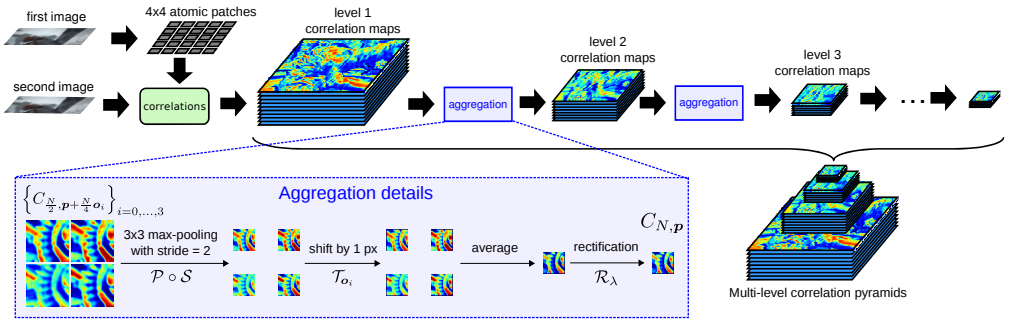


Figure 3.6 – Computing the multi-level correlation pyramid. Starting with the bottom-level correlation maps, see Figure 3.3, they are iteratively aggregated to obtain the upper levels. Aggregation consists of max-pooling, subsampling, computing a shifted average and a non-linear rectification.

We now explain how we can break down Equation 3.6 into a succession of simple operations. First, let us assume that $N = 4 \times 2^\ell$, where $\ell \geq 1$ is the current iteration. During iteration ℓ, we want to compute the correlation maps $C_{N,\boldsymbol{p}}$ of every patch $I_{N,\boldsymbol{p}}$ from the first image for which correlation maps of its children have been computed in the previous iteration. Formally, the position $\mathcal{G}_N$ of such patches is defined according to the position of children patches $\mathcal{G}_{\frac{N}{2}}$ according to Equation 3.5:

$$\mathcal{G}_N = \{\boldsymbol{p} \quad | \quad \boldsymbol{p} + \mathrm{s}_{N,i} \in [0, W-1] \times [0, H-1] \wedge \boldsymbol{p} + \mathrm{s}_{N,i} \in \mathcal{G}_{\frac{N}{2}}, i = 0, \ldots, 3\} \quad . \tag{3.7}$$

We observe that the larger a patch is (*i.e.*, after several iterations), the smaller the spatial variation of its correlation map (see Figure 3.4). This is due to the statistics of natural images, in which low frequencies significantly dominate over high frequencies. As a consequence, we choose to subsample each map $C_{N,\boldsymbol{p}}$ by a factor 2. We express this with an operator $\mathcal{S}$:

$$\mathcal{S} : C(\boldsymbol{p}') \to C(2\boldsymbol{p}') \quad . \tag{3.8}$$

The subsampling reduces by 4 the area of the correlation maps and, as a direct consequence, the computational requirements. Instead of computing the subsampling on top of Equation 3.6, it is actually more efficient to propagate it towards the children maps and perform it jointly with max-pooling. It also makes the max-pooling domain Θ_i become independent from N in the subsampled maps, as it exactly cancels out the effect of doubling $N = 4 \times 2^\ell$ at each iteration. We call $\mathcal{P}$ the max-pooling operator

with the iteration-independent domain $\Theta = \{-1, 0, 1\} \times \{-1, 0, 1\}$:

$$\mathcal{P} : C(\boldsymbol{p}') \rightarrow \max_{\boldsymbol{m} \in \{-1,0,1\}^2} C(\boldsymbol{p}' + \boldsymbol{m}) \quad . \tag{3.9}$$

For the same reason, the shift $\mathrm{s}_{N,i} = \frac{N}{4}\boldsymbol{o}_i = 2^\ell \boldsymbol{o}_i$ applied to the correlation maps in Θ_i's definition becomes simply $\boldsymbol{o}_i$ after subsampling. Let $\mathcal{T}_{\boldsymbol{t}}$ be the shift (or translation) operator on the correlation map:

$$\mathcal{T}_{\boldsymbol{t}} : C(\boldsymbol{p}') \rightarrow C(\boldsymbol{p}' - \boldsymbol{t}) \quad . \tag{3.10}$$

Finally, we incorporate an additional non-linear mapping at each iteration on top of Equation 3.6 by applying a power transform $\mathcal{R}_\lambda$ [Malik and Perona, 1990, LeCun et al., 1998a]:

$$\mathcal{R}_\lambda : C(\boldsymbol{.}) \rightarrow C(\boldsymbol{.})^\lambda \quad . \tag{3.11}$$

This step, commonly referred to as rectification, is added in order to better propagate high correlations after each level, or, in other words, to counterbalance the fact that max-pooling tends to retain only high scores. Indeed, its effect is to decrease the correlation values (which are in $[0, 1]$) as we use $\lambda > 1$. Such post-processing is commonly used in deep convolutional networks [LeCun et al., 1998b, Bengio, 2009]. In practice, good performance is obtained with $\lambda \simeq 1.4$, see Section 3.5. The final expression of Equation 3.6 is:

$$C_{N,\boldsymbol{p}} = \mathcal{R}_\lambda \left(\frac{1}{4} \sum_{i=0}^{3} (\mathcal{T}_{\boldsymbol{o}_i} \circ \mathcal{S} \circ \mathcal{P}) \left(C_{\frac{N}{2}, \boldsymbol{p} + \mathrm{s}_{N,i}} \right) \right) \quad . \tag{3.12}$$

Figure 3.6 illustrates the computation of correlation maps for different patch sizes and Algorithm 3.1 summarizes our approach. The resulting set of correlation maps across iterations is referred to as multi-level correlation pyramid.

Boundary effects. In practice, a patch $I_{N,\boldsymbol{p}}$ can overlap with the image boundary, as long as its center $\boldsymbol{p}$ remains inside the image (from Equation 3.7). For instance, a patch $I_{N,\boldsymbol{p}_0}$ with center at $\boldsymbol{p}_0 = (0, 0) \in \mathcal{G}_N$ has only a single valid child (the one for which $i = 3$ as $\boldsymbol{p}_0 + \mathrm{s}_{N,3} \in I$). In such degenerate cases, the average sum in Equation 3.12 is carried out on valid children only. For $I_{N,\boldsymbol{p}_0}$, it thus only comprises one term weighted by 1 instead of $\frac{1}{4}$.

Note that Equation 3.12 implicitly defines the set of possible displacements of the approach, see Figures 3.2 and 3.9. Given the position of a parent patch, each child patch can move only within a small extent, equal

to the quarter of its own size. Figure 3.4 shows the correlation maps for patches of size 4, 8 and 16. Clearly, correlation maps for larger patches are more and more discriminative, while still allowing non-rigid matching.

Algorithm 3.1 Computing the multi-level correlation pyramid.

Input: Images I, I'
For $\boldsymbol{p} \in \mathcal{G}_4$ **do**
—– $C_{4,\boldsymbol{p}} = I^{\mathrm{F}}_{4,\boldsymbol{p}} \star I'$ (convolution, Equation 3.4)
—– $C_{4,\boldsymbol{p}} \leftarrow \mathcal{R}_\lambda(C_{4,\boldsymbol{p}})$ (rectification, Equation 3.11)
$N \leftarrow 4$
While $N < \max(W, H)$ **do**
—– **For** $\boldsymbol{p} \in \mathcal{G}_N$ **do**
——— $C'_{N,\boldsymbol{p}} \leftarrow (\mathcal{S} \circ \mathcal{P})(C_{N,\boldsymbol{p}})$ (max-pooling and subsampling)
—– $N \leftarrow 2N$
—– **For** $\boldsymbol{p} \in \mathcal{G}_N$ **do**
——— $C_{N,\boldsymbol{p}} = \frac{1}{4}\sum_{i=0}^{3} \mathcal{T}_{\boldsymbol{o}_i}\left(C'_{\frac{N}{2},\boldsymbol{p}+\mathrm{s}_{N,i}}\right)$ (shift and average)
——— $C_{N,\boldsymbol{p}} \leftarrow \mathcal{R}_\lambda(C_{N,\boldsymbol{p}})$ (rectification, Equation 3.11)
Return the multi-level correlation pyramid $\{C_{N,\boldsymbol{p}}\}_{N,\boldsymbol{p}}$

3.2.3 Top-down correspondence extraction

A score $S = C_{N,\boldsymbol{p}}(\boldsymbol{p}')$ in the multi-level correlation pyramid represents the deformation-tolerant similarity of two patches $I_{N,\boldsymbol{p}}$ and $I'_{N,\boldsymbol{p}'}$. Since this score is built from the similarity of 4 matching sub-patches at the lower pyramid level, we can thus recursively backtrack a set of correspondences to the bottom level (corresponding to matches of atomic patches). In this section, we first describe this backtracking. We, then, present the procedure for merging atomic correspondences backtracked from different entry points in the multi-level pyramid, which constitute the final output of DeepMatching.

Compared to our initial version of DeepMatching [Weinzaepfel et al., 2013], we have updated match scoring and entry point selection to optimize the execution time and the matching accuracy. A quantitative comparison is provided in Section 3.5.2.

Backtracking atomic correspondences. Given an entry point $C_{N,\boldsymbol{p}}(\boldsymbol{p}')$ in the pyramid (*i.e.*, a match between two patches $I_{N,\boldsymbol{p}}$ and $I'_{N,\boldsymbol{p}'}$ [2]), we re-

2. Note that $I'_{N,\boldsymbol{p}'}$ only roughly corresponds to a $N \times N$ square patch centered at $2^\ell \boldsymbol{p}'$ in I', due to subsampling and possible deformations.

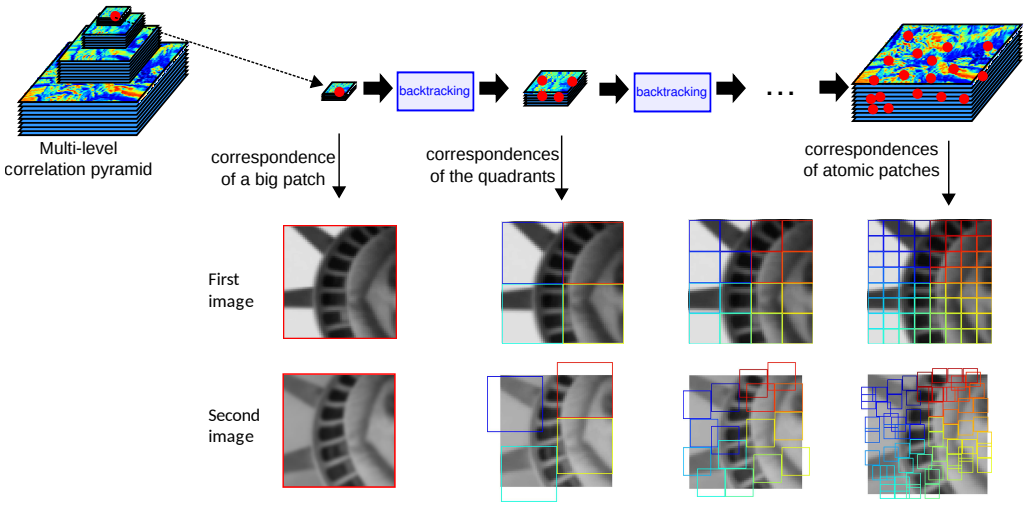


Figure 3.7 – Backtracking atomic correspondences from an entry point (red dot) in the top pyramid level *(left)*. At each level, the backtracking consists in undoing the aggregation performed previously in order to recover the position of the four children patches in the lower level. When the bottom level is reached, we obtain a set of correspondences for atomic patches *(right)*.

trieve atomic correspondences by successively undoing the steps used to aggregate correlation maps during the pyramid construction, see Figure 3.7. The entry patch $I_{N,\boldsymbol{p}}$ is itself composed of four moving quadrants $I^i_{N,\boldsymbol{p}}$, $i = 0...3$. Due to the subsampling, the quadrant $I^i_{N,\boldsymbol{p}} = I_{\frac{N}{2},\boldsymbol{p}+\mathrm{s}_{N,i}}$ matches with $I_{\frac{N}{2},2(\boldsymbol{p}'+\boldsymbol{o}_i)+\boldsymbol{m}_i}$ where

$$\boldsymbol{m}_i = \underset{\boldsymbol{m}\in\{-1,0,1\}^2}{\operatorname{argmax}} \; C_{\frac{N}{2},\boldsymbol{p}+\mathrm{s}_{N,i}}\left(2(\boldsymbol{p}'+\boldsymbol{o}_i)+\boldsymbol{m}\right) \quad . \tag{3.13}$$

For the sake of clarity, we define the short-hand notations $\boldsymbol{p}_i = \boldsymbol{p} + \mathrm{s}_{N,i}$ and $\boldsymbol{p}'_i = 2(\boldsymbol{p}' + \boldsymbol{o}_i) + \boldsymbol{m}_i$. Let B be the function that assigns to a tuple $(N, \boldsymbol{p}, \boldsymbol{p}', s)$, representing a correspondence between pixel $\boldsymbol{p}$ and $\boldsymbol{p}'$ for patch of size N with a score $s \in \mathbb{R}$, the set of the correspondences of children patches:

$$B(N,\boldsymbol{p},\boldsymbol{p}',s) = \begin{cases} \{(\boldsymbol{p},\boldsymbol{p}',s)\} & \text{if } N = 4, \\ \left\{\left(\frac{N}{2},\boldsymbol{p}_i,\boldsymbol{p}'_i,s + C_{\frac{N}{2},\boldsymbol{p}_i}(\boldsymbol{p}'_i)\right)\right\}_{i=0}^{3} & \text{else.} \end{cases} \tag{3.14}$$

Given a set $\mathcal{M}$ of such tuples, let $\mathcal{B}(\mathcal{M})$ be the union of the sets $B(c)$ for all $c \in \mathcal{M}$. Note that if all candidate correspondences $c \in \mathcal{M}$ corresponds to atomic patches, then $\mathcal{B}(\mathcal{M}) = \mathcal{M}$.

Thus, the algorithm for backtracking correspondences is the following. Consider an entry match $\mathcal{M} = \{(N, \boldsymbol{p}, \boldsymbol{p}', \ C_{N,\boldsymbol{p}}(\boldsymbol{p}'))\}$. We repeatedly apply $\mathcal{B}$ on $\mathcal{M}$. After $\mathfrak{N} = \log_2(N/4)$ calls, we get one correspondence for each of the $4^{\mathfrak{N}}$ atomic patches. Furthermore, their score is equal to the sum of all patch similarities along their backtracking path.

Merging correspondences. We have shown how to retrieve atomic correspondences from a match between two deformable (potentially large) patches. Despite this flexibility, a single match is unlikely to explain the complex set of motions that can occur, for example, between two adjacent frames in a video, *i.e.*, two objects moving independently with significantly different motions exceeds the deformation range of DeepMatching. We quantitatively specify this range in the next subsection.

We thus merge atomic correspondences gathered from different entry points (matches) in the pyramid. In the initial version of DeepMatching [Weinzaepfel et al., 2013], entry points were local maxima over all correlation maps. This is now replaced by a faster procedure, that starts with all possible matches in the top pyramid level (*i.e.*, $\mathcal{M} = \{(N, \boldsymbol{p}, \boldsymbol{p}', C_{N,\boldsymbol{p}}(\boldsymbol{p}'))|N = N_{\max}\}$). Using this level only results in significantly less entry points than starting from all maxima in the entire pyramid. We did not observe any impact on the matching performance, see Section 3.5.2. Because $\mathcal{M}$ contains a lot of overlapping patches, most of the computation during repeated calls to $\mathcal{M} \leftarrow \mathcal{B}(\mathcal{M})$ can be factorized. In other words, as soon as two tuples in $\mathcal{M}$ are equal in terms of N, $\boldsymbol{p}$ and $\boldsymbol{p}'$, the one with the lowest score is simply eliminated. We thus obtain a set of atomic correspondences $\mathcal{M}'$:

$$\mathcal{M}' = (\mathcal{B} \circ \ldots \circ \mathcal{B})(\mathcal{M}) \quad . \tag{3.15}$$

that we filter with reciprocal match verification. The final set of correspondences $\mathcal{M}''$ is obtained as:

$$\mathcal{M}'' = \{(\boldsymbol{p}, \boldsymbol{p}', s)|\text{BestAt}(\boldsymbol{p}) = \text{BestAt}'(\boldsymbol{p}')\}_{(\boldsymbol{p},\boldsymbol{p}',s)\in\mathcal{M}'} \quad . \tag{3.16}$$

where $\text{BestAt}(\boldsymbol{p})$ (resp. $\text{BestAt}'(\boldsymbol{p}')$) returns the best match in a small vicinity of 4×4 pixels around $\boldsymbol{p}$ in I (resp. around $\boldsymbol{p}'$ in I') from $\mathcal{M}'$.

3.2.4 Discussion and Analysis of DeepMatching

Multi-size patches and repetitive textures. During the bottom-up pass of the algorithm, we iteratively aggregate correlation maps of smaller

Figure 3.8 – Matching result between two images with repetitive textures. Nearly all output correspondences are correct. Wrong matches are due to occluded areas (bottom-right of the first image) or situations where the deformation tolerance of DeepMatching is exceeded (bottom-left of the first image).

patches to form the correlation maps of larger patches. Doing so, we effectively consider patches of different sizes $(4 \times 2^\ell, \ell \geq 0)$, in contrast to most existing matching methods. This is a key feature of our approach when dealing with repetitive textures. As one moves up to upper levels, the matching problem gets less ambiguous. Hence, our method can correctly match repetitive patterns, see for instance Figure 3.8.

Quasi-dense correspondences. Our method retrieves dense correspondences for every single match between large regions (*i.e.*, entry point for the backtracking in the top-level correlation maps), even in weakly textured areas; this is in contrast to correspondences obtained when matching descriptors (*e.g.* SIFT). A quantitative assessment, which compares the coverage of matches obtained with several matching schemes, is given in Section 3.5.

Non-rigid deformations. Our matching algorithm is able to cope with various sources of image deformations: object-induced or camera-induced. The set of feasible deformations, explicitly defined by Equation 3.6, theoretically allows to deal with a scaling factor in the range $[\frac{1}{2}, \frac{3}{2}]$ and rotations approximately in the range $[-26^o, 26^o]$. Note also that DeepMatching is translation-invariant by construction, thanks to the convolutional nature of the processing.

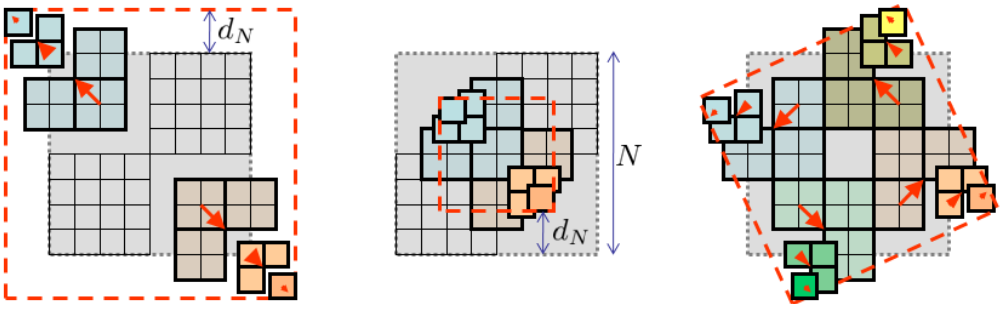


Figure 3.9 – Extent of the tolerance of DeepMatching to deformations. From left to right: up-scale of 1.5x, down-scale of 0.5x, rotation of 26°. The plain gray (resp. dashed red) square represents the patch in the reference (resp. target) image. For clarity, only the corner pixels are maximally deformed.

Proof. Given a patch of size $N = 4 \times 2^\ell$ located at level $\ell \geqslant 1$, Equation 3.6 allows each of its children patches to move by at most $N/8$ pixels from their ideal location in Θ_i. By recursively summing the displacements at each level, the maximal displacements for an atomic patch is $d_N = \sum_{i=1}^{\ell} 2^{i-1} = 2^\ell - 1$. An example is given in Figure 3.9 with $N = 32$ and $\ell = 3$. Relatively to N, we thus have $\lim_{N\to\infty} (N+2d_N)/N = \frac{3}{2}$ and $\lim_{N\to\infty} (N-2d_N)/N = \frac{1}{2}$. For a rotation, the rationale is similar, see Figure 3.9. □

Note that the displacement tolerance in Θ_i from Equation 3.6 could be extended to $x \times N/8$ pixels with $x \in \{2, 3, \ldots\}$ (instead of $x = 1$). Then the above formula for computing the lower bound on the scale factor of DeepMatching generalizes to $\mathrm{LB}(x) = \lim_{N\to\infty} (N - 2xd_N)/N$. Hence, for $x \geq 2$ we obtain $\mathrm{LB}(x) = 0$ instead of $\mathrm{LB}(1) = \frac{1}{2}$. This implies that the deformation range is extended to a point where any patch can be matched to a single pixel, *i.e.*, this results in unrealistic deformations. For this reason, we choose not to expand the deformation range of DeepMatching.

Built-in smoothing. Furthermore, correspondences generated through backtracking of a single entry point in the correlation maps are naturally smooth. Indeed, feasible deformations cannot be too 'far' from the identity deformation. To verify this assumption, we conduct the following experiment. We artificially generate two types of correspondences between two images of size 128×128. The first one is completely random, *i.e.*, for each atomic patch in the first image we assign randomly a match in the second image. The second one respects the backtracking constraints. Starting from a single entry point in the top level we simulate the backtracking procedure

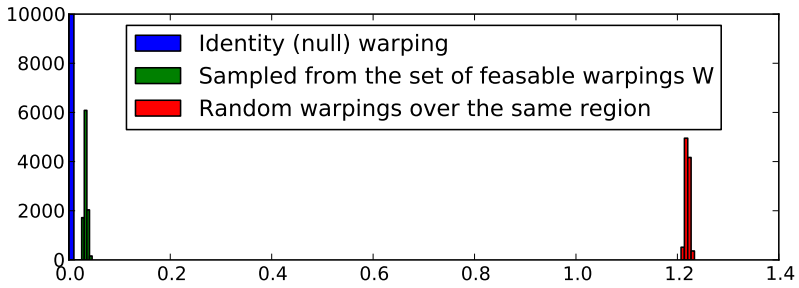


Figure 3.10 – Histogram over smoothness for identity warping, warping respecting the built-in constraints in DeepMatching and random warping. The x-axis indicates the smoothness value. The smoothness value is low when there are few discontinuities, *i.e.*, the warpings are smooth. The histogram is obtained with 10,000 different artificial warpings. See text for details.

from Section 3.2.3 by replacing in Equation 3.13 the max operation by a random sampling over $\{-1, 0, 1\}^2$. By generating 10,000 sets of possible atomic correspondences, we simulate a set which respects the deformations allowed by DeepMatching. Figure 3.10 compares the smoothness of these two types of artificial correspondences. Smoothness is measured by interpreting the correspondences as flow and measuring the gradient flow norm, see Equation 3.19. Clearly, the two types of warpings are different by orders of magnitude. Furthermore, the one which respects the built-in constraints of DeepMatching is close to the identity warping.

Relation to Deep Convolutional Neural Networks (CNNs). DeepMatching relies on a hierarchical, multi-layer, correlational architecture designed for matching images and was inspired by deep convolutional approaches [LeCun et al., 1998a]. In the following we describe the major similarities and differences.

Deep networks learn from data the weights of the convolutions. In contrast, DeepMatching does not learn any feature representations and instead directly computes correlations at the patch level. It uses patches from the first image as convolution filters for the second one. However, the bottom-up pipeline of DeepMatching is similar to CNNs. It alternates aggregating channels from the previous layer with channel-wise max-pooling and subsampling. As in CNNs, max-pooling in DeepMatching allows for invariance *w.r.t.* small deformations. Likewise, the algorithm propagates pairwise patch similarity scores through the hierarchy using non-linear rectifying stages in-between layers. Finally, DeepMatching includes a top-down

pass which is not present in CNNs. Note that this pass is fundamentally different from deconvolutional architecture as it does not require additional parameters.

Time and space complexity. DeepMatching has a complexity $O(LL')$ in memory and time, where $L = WH$ and $L' = W'H'$ are the number of pixels per image.

Proof. Computing the initial correlations is a $O(LL')$ operation. Then, at each level of the pyramid, the process is repeated while the complexity is divided by a factor 4 due to the subsampling step in the target image (since the cardinality of $|\{\mathcal{G}_N\}|$ remains approximately constant). Thus, the total complexity of the correlation maps computation is, at worst, $O(\sum_{n=0}^{\infty} LL'/4^n) = O(LL')$. During the top-down pass, most backtracking paths can be pruned as soon as they cross a concurrent path with a higher score (see Section 3.2.3). Thus, all correlations will be examined at most once, and there are $\sum_{n=0}^{\infty} LL'/4^n$ values in total. However, this analysis is worst-case. In practice, only correlations lying on maximal paths are actually examined. □

3.3 Extensions of DeepMatching

3.3.1 Approximate DeepMatching

As a consequence of its $O(LL')$ space complexity, DeepMatching requires an amount of RAM that is orders of magnitude above other state-of-the-art matching methods. This could correspond to several gigabytes for images of moderate size (800×600 pixels); see Section 3.5.2. This section introduces an approximation of DeepMatching that allows to trade matching quality for reduced time and memory usage. As shown in Section 3.5.2, near-optimal results can be obtained at a fraction of the original cost.

Our approximation proposes to compress the representation of atomic patches $\{I_{4,\boldsymbol{p}}\}$. Atomic patches carry little information, and thus are highly redundant. For instance, in uniform regions, all patches are nearly identical (*i.e.*, gradient-wise). To exploit this property, we index atomic patches with a small set of patch prototypes. We substitute each patch with its closest neighbor in a fixed dictionary of D prototypes. Hence, we need to perform and store only D convolutions at the first level, instead of $O(L)$ (with $D \ll O(L)$). This significantly reduces both memory and time complexity. Note that higher pyramid levels also benefit from this optimization. Indeed,

two parent patches at the second level have the exact same correlation map in case their children are assigned the same prototypes. The same reasoning also holds for all subsequent levels, but the gains rapidly diminish due to statistical unlikeliness of the required condition. This is not really an issue, since the memory and computational cost mostly rests on the initial levels; see Section 3.2.4.

In practice, we build the prototype dictionary using k-means, as it is designed to minimize the approximation error between input descriptors and resulting centroids (*i.e.*, prototypes). Given a pair of images to match, we perform on-line clustering of all descriptors of atomic patches $\{I_{4,\boldsymbol{p}}\} = \{\mathbf{R}\}$ in the first image. Since the original descriptors lie on an hypersphere (each pixel descriptor $\mathbf{R}_{i,j}$ has norm 1), we modify the k-means approach so as to project the estimated centroids on the hypersphere at each iteration. We find experimentally that this is important to obtain good results.

3.3.2 Scale and rotation invariant DeepMatching

For a variety of tasks, objects to be matched can appear under image rotations or at different scales [Lowe, 2004, Mikolajczyk et al., 2005, Szeliski, 2010, HaCohen et al., 2011]. As discussed above, DeepMatching (DM) is only robust to moderate scale changes and rotations. We now present a scale and rotation invariant version.

Algorithm 3.2 Scale and rotation invariant version of DeepMatching (DM). I_σ denotes the image I downsized by a factor σ, and $\mathcal{R}_\theta$ denotes rotation by an angle θ.

Input: I, I' are the images to be matched
Initialize an empty set $\mathcal{M}' = \{\}$ of correspondences
For $\sigma \in \{-2, -1.5, \dots, 1.5, 2\}$ **do**
—– $\sigma_1 \leftarrow \max(1, 2^{+\sigma})$ # either downsize image 1
—– $\sigma_2 \leftarrow \max(1, 2^{-\sigma})$ # or downsize image 2
—– **For** $\theta \in \left\{0, \frac{\pi}{4}, \dots, \frac{7\pi}{4}\right\}$ **do**
get raw atomic correspondences (Equation 3.15)
——— $\mathcal{M}'_{\sigma,\theta} \leftarrow \text{DeepMatching}\left(I_{\sigma_1}, \mathcal{R}_{-\theta} * I'_{\sigma_2}\right)$
Geometric rectification to the input image space
——— $\mathcal{M}'^{R}_{\sigma,\theta} \leftarrow \left\{(\sigma_1 \boldsymbol{p},\ \sigma_2 \mathcal{R}_\theta \boldsymbol{p}',\ s) \mid \forall (\boldsymbol{p}, \boldsymbol{p}', s) \in \mathcal{M}'_{\sigma,\theta}\right\}$
Concatenate results
——— $\mathcal{M}' \leftarrow \mathcal{M}' \bigcup \mathcal{M}'^{R}_{\sigma,\theta}$
$\mathcal{M}'' \leftarrow \text{reciprocal}(\mathcal{M}')$ # keep reciprocal matches (Equation 3.16)
Return $\mathcal{M}''$

The approach is straightforward: we apply DM to several rotated and scaled versions of the second image. According to the invariance range of DM, we use steps of $\pi/4$ for image rotation and power of $\sqrt{2}$ for scale changes. While iterating over all combinations of scale changes and rotations, we maintain a list $\mathcal{M}'$ of all atomic correspondences obtained so far, *i.e.*, corresponding positions and scores. As before, the final output correspondences consists of the reciprocal matches in $\mathcal{M}'$. Storing all matches and finally choosing the best ones based on reciprocal verification permits to capture distinct motions possibly occurring together in the same scene (*e.g.* one object could have undergone a rotation, while the rest of the scene did not move). The steps of the approach are described in Algorithm 3.2.

Since we iterate sequentially over a fixed list of rotations and scale changes, the space and time complexity of the algorithm remains unchanged (*i.e.*, $O(LL')$). In practice, the run-time compared to DM is multiplied by a constant approximately equal to 25, see Section 3.5.2. Note that the algorithm permits a straightforward parallelization.

3.4 DeepFlow

We now present our approach for optical flow estimation, DeepFlow. We adopt the formulation introduced by Brox and Malik [2011], where a matching term penalizes the differences between optical flow and input matches, and replace their matching approach by DeepMatching. In addition, we make a few minor modifications introduced recently in the state of the art: (i) we add a normalization in the data term to downweight the impact of locations with high spatial image derivatives [Zimmer et al., 2011]; (ii) we use a different weight at each level to downweight the matching term at finer scales [Stoll et al., 2012]; and (iii) the smoothness term is locally weighted [Xu et al., 2012].

Let $I_1, I_2 : \Omega \to \mathbb{R}^c$ be two consecutive images defined on Ω with c channels. The goal is to estimate the flow $\boldsymbol{w} = (u, v)^\top : \Omega \to \mathbb{R}^2$. We assume that the images are already smoothed using a Gaussian filter of standard deviation σ. The energy we optimize is a weighted sum of a data term E_D, a smoothness term E_S and a matching term E_M:

$$E(\boldsymbol{w}) = \int_\Omega E_D + \alpha E_S + \beta E_M d\boldsymbol{x} \quad . \tag{3.17}$$

For the three terms, we use a robust penalizer $\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$ with $\epsilon = 0.001$ which has shown excellent results [Sun et al., 2014b].

Data term. The data term is a separate penalization of the color and gradient constancy assumptions with a normalization factor as proposed by Zimmer et al. [2011]. We start from the optical flow constraint assuming brightness constancy: $(\nabla_3^\top I)\boldsymbol{W} = 0$ with $\nabla_3 = (\partial x, \partial y, \partial t)^\top$ the spatio-temporal gradient and $\boldsymbol{W} = (u, v, 1)^\top$. A basic way to build a data term is to penalize it, *i.e.*, $E_D = \Psi(\boldsymbol{W}^\top \mathbb{J}_0 \boldsymbol{W})$ with $\mathbb{J}_0$ the tensor defined by $\mathbb{J}_0 = (\nabla_3 I)(\nabla_3^\top I)$. As highlighted by Zimmer et al. [2011], such a data term adds a higher weight in locations corresponding to high spatial image derivatives. We normalize it by the norm of the spatial derivatives plus a small factor to avoid division by zero, and to reduce a bit the influence in tiny gradient locations [Zimmer et al., 2011]. Let $\bar{\mathbb{J}}_0$ be the normalized tensor $\bar{\mathbb{J}}_0 = \theta_0 \mathbb{J}_0$ with $\theta_0 = (\|\nabla_2 I\|^2 + \zeta^2)^{-1}$. We set $\zeta = 0.1$ in the following. To deal with color images, we consider the tensor defined for a channel i denoted by upper indices $\bar{\mathbb{J}}_0^i$ and we penalize the sum over channels: $\Psi(\sum_{i=1}^c \boldsymbol{W}^\top \bar{\mathbb{J}}_0^i \boldsymbol{W})$. We consider images in the RGB color space.

We separately penalize the gradient constancy assumption [Bruhn et al., 2005a]. Let I_x and I_y be the derivatives of the images with respect to the x and y axis respectively. Let $\bar{\mathbb{J}}_{xy}^i$ be the tensor for the channel i including the normalization:

$$\bar{\mathbb{J}}_{xy}^i = (\nabla_3 I_x^i)(\nabla_3^\top I_x^i)/(\|\nabla_2 I_x^i\|^2 + \zeta^2) + (\nabla_3 I_y^i)(\nabla_3^\top I_y^i)/(\|\nabla_2 I_y^i\|^2 + \zeta^2) \quad .$$

The data term is the sum of two terms, balanced by two weights δ and γ:

$$E_D = \delta \Psi \left(\sum_{i=1}^{c} \boldsymbol{W}^\top \bar{\mathbb{J}}_0^i \boldsymbol{W} \right) + \gamma \Psi \left(\sum_{i=1}^{c} \boldsymbol{W}^\top \bar{\mathbb{J}}_{xy}^i \boldsymbol{W} \right) \quad . \tag{3.18}$$

Smoothness term. The smoothness term is a robust penalization of the gradient flow norm:

$$E_S = \Psi \left(\|\nabla u\|^2 + \|\nabla v\|^2 \right) \quad . \tag{3.19}$$

The smoothness weight α is locally set according to image derivatives [Wedel et al., 2009a, Xu et al., 2012] with $\alpha(\boldsymbol{x}) = \exp(-\kappa \nabla_2 I(\boldsymbol{x}))$ where κ is experimentally set to $\kappa = 5$.

Matching term. The matching term encourages the flow estimation to be similar to a precomputed vector field $\boldsymbol{w}'$. To this end, we penalize the difference between $\boldsymbol{w}$ and $\boldsymbol{w}'$ using the robust penalizer Ψ. Since the matching is not totally dense, we add a binary term $c(\boldsymbol{x})$ which is equal to 1 if and only if a match is available at $\boldsymbol{x}$.

We also multiply each matching penalization by a weight $\phi(\boldsymbol{x})$, which is low in uniform regions where matching is ambiguous and when matched patches are dissimilar. To that aim, we rely on $\tilde{\lambda}(\boldsymbol{x})$, the minimum eigenvalue of the autocorrelation matrix multiplied by 10. We also compute the visual similarity between matches as $\Delta(\boldsymbol{x}) = \sum_{i=1}^{c} |I_1^i(\boldsymbol{x}) - I_2^i(\boldsymbol{x} - \boldsymbol{w}'(\boldsymbol{x}))| + |\nabla I_1^i(\boldsymbol{x}) - \nabla I_2^i(\boldsymbol{x} - \boldsymbol{w}'(\boldsymbol{x}))|$. We then compute the score ϕ as a Gaussian kernel on Δ weighted by $\tilde{\lambda}$ with a parameter σ_M, experimentally set to $\sigma_M = 50$. More precisely, we define $\phi(\boldsymbol{x})$ at each point $\boldsymbol{x}$ with a match $\boldsymbol{w}'(\boldsymbol{x})$ as:

$$\phi(\boldsymbol{x}) = \sqrt{\tilde{\lambda}(\boldsymbol{x})}/(\sigma_M\sqrt{2\pi}) \ \exp(-\Delta(\boldsymbol{x})/2\sigma_M) \quad .$$

The matching term is then $E_M = c\phi\Psi(\|\boldsymbol{w} - \boldsymbol{w}'\|_2^2)$.

Minimization. This energy objective is non-convex and non-linear. To solve it, we use a numerical optimization algorithm similar as Brox et al. [2004]. An incremental coarse-to-fine warping strategy is used with a downsampling factor $\eta = 0.95$. The remaining equations are still non-linear due to the robust penalizers. We apply 5 inner fixed point iterations where the non-linear weights and the flow increments are iteratively updated while fixing the other. To approximate the solution of the linear system, we use 25 iterations of the Successive Over Relaxation (SOR) method [Young and Rheinboldt, 1971]. More details about the minimization of the energy are given in Appendix A.

To downweight the matching term on fine scales, we use a different weight β^k at each level as proposed by Stoll et al. [2012]. We set $\beta^k = \beta(k/k_{\max})^b$ where k is the current level of computation, $k_{\max}$ the coarsest level and b a parameter which is optimized together with the other parameters, see Section 3.5.3.

3.5 Experiments

This section presents an experimental evaluation of DeepMatching and DeepFlow. The datasets and metrics used to evaluate DeepMatching and DeepFlow are introduced in Section 3.5.1. Experimental results are given in Sections 3.5.2 and 3.5.3 respectively.

3.5.1 Datasets and metrics

In this section, we briefly introduce the matching and flow datasets used in our experiments. Since consecutive frames of a video are well-suited

to evaluate a matching approach, we use several optical flow datasets for evaluating both the quality of matching and flow, but we rely on different metrics. We present experimental results for the MPI-Sintel dataset, the Kitti dataset and the Middlebury dataset. We refer to Section 2.5 for details about these optical flow benchmarks.

In addition, we use a matching benchmark, namely the **Mikolajczyk** dataset. It was originally proposed by Mikolajczyk et al. [2005] to evaluate and compare the performance of keypoint detectors and descriptors. It is one of the standard benchmarks for evaluating matching approaches. The dataset consists of 8 sequences of 6 images each viewing a scene under different conditions, such as illumination changes or viewpoint changes. The images of a sequence are related by homographies. During the evaluation, we comply to the standard procedure in which the first image of each scene is matched to the 5 remaining ones. Since our goal is to study robustness of DeepMatching to geometric distortions, we follow HaCohen et al. [2011] and restrict our evaluation to the 4 most difficult sequences with viewpoint changes: *bark*, *boat*, *graf* and *wall*.

Performance metric for matching. Choosing a performance measure for matching approaches is delicate. Matching approaches typically do not return dense correspondences, but output varying numbers of matches. Furthermore, correspondences might be concentrated in different areas of the image.

Most matching approaches, including DeepMatching, are based on establishing correspondences between patches. Given a pair of matching patches, it is possible to obtain a list of pixel correspondences for all pixels within the patches. We introduce a measure based on the number of correctly matched *pixels* compared to the overall number of pixels. We define 'accuracy@T' as the proportion of 'correct' pixels from the first image with respect to the total number of pixels. A pixel is considered correct if its pixel match in the second image is closer than T pixels to ground-truth. In practice, we use a threshold of $T = 10$ pixels, as this represents a sufficiently precise estimation (about 1% of image diagonal for all datasets), while allowing some tolerance in blurred areas that are difficult to match exactly. If a pixel belongs to several matches, we choose the one with the highest score to predict its correspondence. Pixels which do not belong to any patch have an infinite error.

Performance metric for optical flow. To evaluate optical flow, we follow the standard protocol and measure the average endpoint error over

all pixels, denoted as EPE. The 's10-40' variant measures the EPE only for pixels with a ground-truth displacement between 10 and 40 pixels, and likewise for 's0-10' and 's40+'. More details about the optical flow metrics can be found in Section 2.5.

3.5.2 Matching Experiments

In this section, we evaluate DeepMatching (DM). We present results for all datasets presented above but Middlebury, which does not feature long-range motions, the main difficulty in image matching. When evaluating on the Mikolajczyk dataset, we employ the scale and rotation invariant version of DM presented in Section 3.3.2. For all the matching experiments reported in this section, we use the Mikolajczyk dataset and the training sets of MPI-Sintel and Kitti.

Impact of the parameters

We optimize the different parameters of DM jointly on all datasets. To prevent overfitting, we use the same parameters across all datasets.

Pixel descriptor parameters. We first optimize the parameters of the pixel representation (Section 3.2.2): ν_1, ν_2, ν_3 (different smoothing stages), ς (sigmoid slope) and μ (regularization constant). After performing a grid search, we find that good results are obtained at $\nu_1 = \nu_2 = \nu_3 = 1$, $\varsigma = 0.2$ and $\mu = 0.3$ across all datasets. Figure 3.11 shows the accuracy@10 in the neighborhood of these values for all parameters. Image pre-smoothing seems to be crucial for JPEG images (Mikolajczyk dataset), as it smooths out compression artifacts, whereas it slightly degrades performance for uncompressed PNG images (MPI-Sintel and Kitti). As expected, similar findings are observed for the regularization constant μ since it acts as a regularizer that reduces the impact of small gradients (*i.e.*, noise). In the following, we thus use low values of ν_1 and μ when dealing with PNG images (we set $\nu_1 = 0$ and $\mu = 0.1$, other parameters are unchanged).

Non-linear rectification. We also evaluate the impact of the parameter λ of the non-linear rectification obtained by applying power normalization, see Equation 3.11. Figure 3.12 displays the accuracy@10 for various values of λ. We can observe that the optimal performance is achieved at $\lambda = 1.4$ for all datasets. We use this value in the remainder of our experiments.

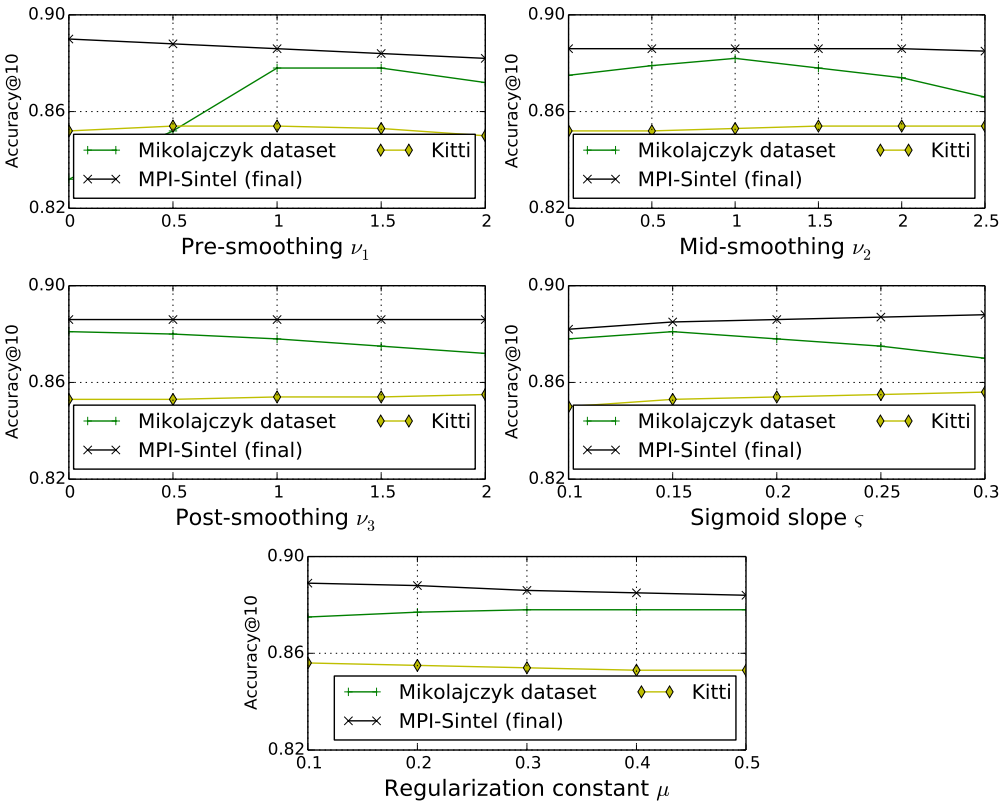


Figure 3.11 – Impact of the parameters to compute pixel descriptors on the different datasets.

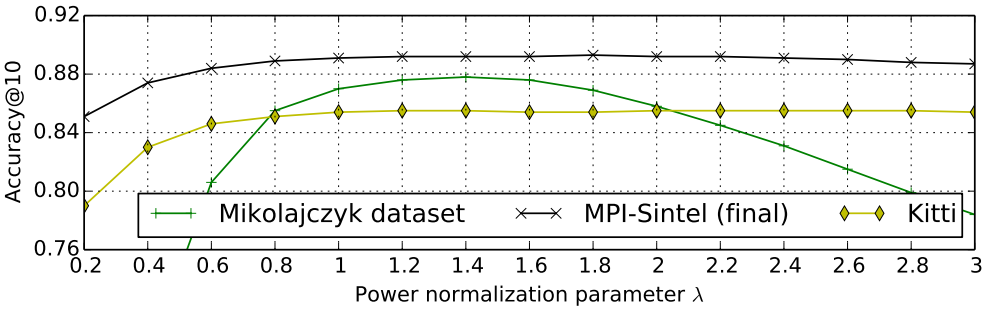


Figure 3.12 – Impact of the non-linear response rectification (Equation 3.11).

R	New BT entry points	New scoring scheme	accuracy@10	memory usage	matching time
Mikolajczyk dataset					
1/4			0.620	0.9 GB	1.0 min
1/2			0.848	5.5 GB	20 min
1/2		✓	0.864	5.5 GB	7.3 min
1/2	✓	✓	**0.878**	4.4 GB	6.3 min
MPI-Sintel dataset (final)					
1/4			0.822	0.4 GB	2.4 sec
1/2			0.880	6.3 GB	55 sec
1/2		✓	0.890	6.3 GB	16 sec
1/2	✓	✓	**0.892**	4.6 GB	16 sec
Kitti dataset					
1/4			0.772	0.4 GB	2.0 sec
1/2			0.841	6.3 GB	39 sec
1/2		✓	0.855	6.3 GB	14 sec
1/2	✓	✓	**0.856**	4.7 GB	14 sec

Table 3.1 – Detailed comparison between the preliminary and current versions of DeepMatching in terms of performance, run-time and memory usage. R denotes the input image resolution and BT backtracking. Run-times are computed on 1 core @ 3.6 GHz.

Evaluation of the backtracking and scoring schemes

We now evaluate two improvements of DM with respect to the previous version published in [Weinzaepfel et al., 2013], referred to as DM*:

- Backtracking (BT) entry points: in DM* we select as entry points local maxima in the correlation maps from all pyramid levels. The new alternative is to start from all possible points in the top pyramid level.
- Scoring scheme: In DM* we scored atomic correspondences based on the correlation values of start and end point of the backtracking path. The new scoring scheme is the sum of correlation values along the full backtracking path.

We report results for the different variants in Table 3.1 on each dataset. The first two rows for each dataset correspond to the exact settings used for DM* (*i.e.*, with an image resolution of 1/4 and 1/2). We observe a steady increase in performance on all datasets when we add the new scoring and backtracking approach. We can observe that starting from all possible entry points in the top pyramid level (*i.e.*, considering all possible

translations) yields slightly better results than starting from local maxima. This demonstrates that some ground-truth matches are not covered by any local maximum. By enumerating all possible patch translations on the top-level, we instead ensure to fully explore the space of all possible matches. We verify this experimentally for artificial images with small objects in fast motion. One could expect that the old BT scheme is more appropriate for matching small objects as it produces local maxima at intermediate levels of the pyramid. However, this is not confirmed by our experiment. As stated above a possible explanation is the exhaustive coverage of translations at the top-level combined with an efficient backtracking scheme. Note, however, that the performance of matching small objects with rapid motion is relatively low compared to matching large areas.

Furthermore, it is interesting to note that memory usage and run-time significantly decreases when using the new options. This is because (1) searching and storing local maxima (which are exponentially more numerous in lower pyramid levels) is not necessary anymore, and (2) the new scoring scheme allows for further optimization, *i.e.*, early pruning of backtracking paths (Section 3.2.3).

Approximate DeepMatching

We now evaluate the performance of approximate DeepMatching (Section 3.3.1) and report its run-time and memory usage. We evaluate and compare two different ways of reducing the computational load. The first one simply consists in downsizing the input images, and upscaling the resulting matches accordingly. The second option is the compression scheme proposed in Section 3.3.1.

We evaluate both schemes jointly by varying the input image size (expressed as a fraction R of the original resolution) and the size D of the prototype dictionary (*i.e.*, parameter of k-means in Section 3.3.1). $R = 1$ corresponds to the original dataset image size (no downsizing). We display the results in terms of matching accuracy (accuracy@10) against memory consumption in Figure 3.13 and as a function of D in Figure 3.14. Figure 3.13 shows that DeepMatching can be computed in an approximate manner for any given memory budget. Unsurprisingly, too low settings (*e.g.* $R \leq 1/8$, $D \leq 64$) result in a strong loss of performance. It should be noted that that we were unable to compute DeepMatching at full resolution ($R = 1$) for $D > 64$, as the memory consumption explodes. As a consequence, all subsequent experiments in the chapter are done at $R = 1/2$. In Figure 3.14, we observe that good trades-off are achieved for dictionary sizes comprised in $D \in [64, 1024]$. For instance, on MPI-Sintel, at $D = 1024$, 94%

of the performance of the uncompressed case ($D = \infty$) is reached for half the computation time and one third of the memory usage. Detailed timings of the different stages of DeepMatching are given in Table 3.2. As expected, only the bottom-up pass is affected by the approximation, with a run-time of the different operations involved (patch correlations, max-pooling, sub-sampling, aggregation and non-linear rectification) roughly proportional to D (or to $|\mathcal{G}_4|$, the actual number of atomic patches, if $D = \infty$). The overhead of clustering the dictionary prototypes with k-means appears negligible, with the exception of the largest dictionary size ($D = 4096$) for which it induces a slightly longer run-time than in the uncompressed case. Overall, the proposed method for approximating DeepMatching is highly effective.

GPU Implementation. We have implemented DM on GPU in the `Caffe` framework [Jia et al., 2014]. Using existing `Caffe` layers like ConvolutionLayer and PoolingLayer, the implementation is straightforward for most layers. We had to specifically code a few layers which are not available in `Caffe` (*e.g.* the backtracking pass [3]). For the aggregation layer which consists in selecting and averaging 4 children channels out of many channels, we relied on the sparse matrix multiplication in the cuSPARSE toolbox. Detailed timings are given in Table 3.2 on a GeForce Titan X. Our code runs in about 0.2s for a pair of MPI-Sintel image. As expected, the computation bottleneck essentially lies in the computation of bottom-level patch correlations and the backtracking pass. Note that computing patch descriptors takes significantly more time, in proportion, than on CPU: it takes about 0.024s = 11% of total time (not shown in table). This is because it involves a succession of many small layers (image smoothing, gradient extraction and projection, *etc.*), which causes overhead and is rather inefficient.

Comparison to the state of the art

We compare DM with several baselines and state-of-the-art matching algorithms, namely:

- SIFT keypoints extracted with DoG detector [Lowe, 2004] and matched with FLANN [Muja and Lowe, 2009], referred to as SIFT-NN, [4]
- dense HOG matching, followed by nearest-neighbor matching with reciprocal verification as done in LDOF [Brox and Malik, 2011], referred to as HOG-NN [4],

3. Although the backtracking is conceptually close to the back-propagation training algorithm, it differs in term of how the scores are accumulated for each path.

4. We implemented this method ourselves.

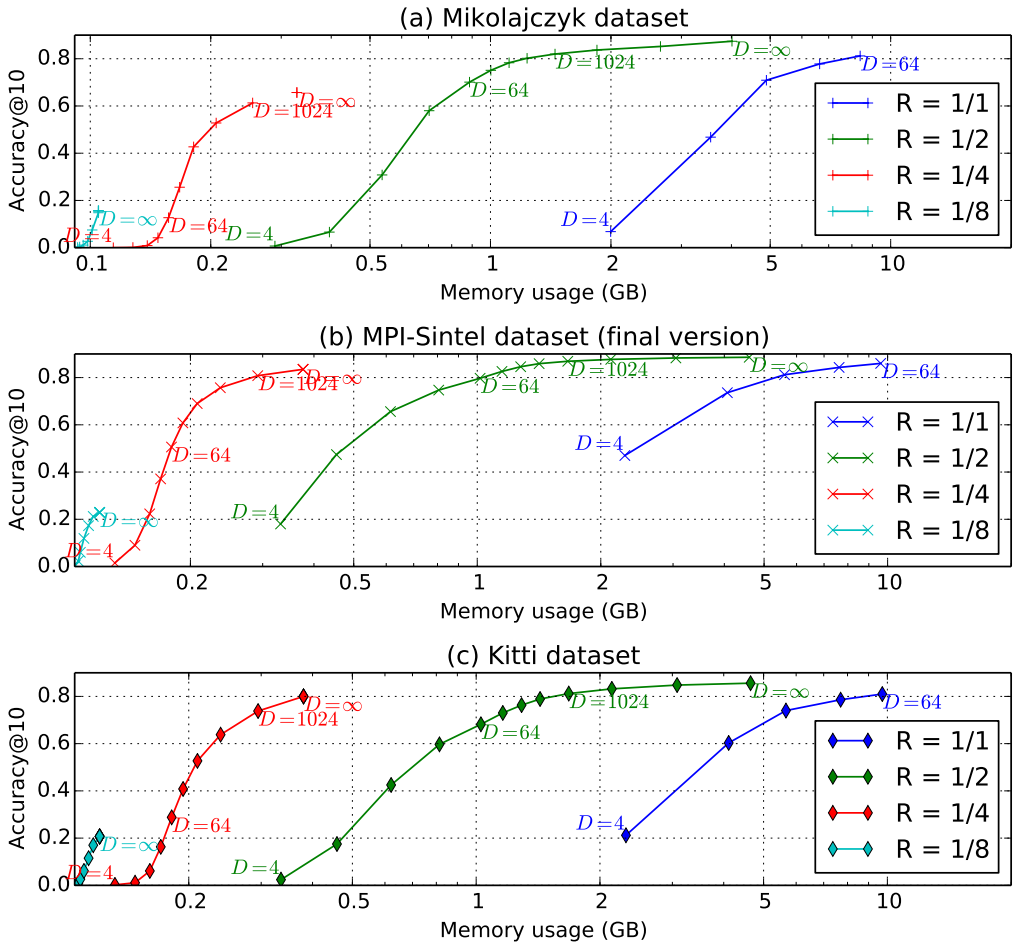


Figure 3.13 – Trade-off between memory consumption and matching performance for the different datasets. Memory usage is controlled by changing image resolution R (different curves) and dictionary size D (curve points).

- Generalized PatchMatch (GPM) [Barnes et al., 2010], with default parameters, 32x32 patches and 20 iterations (best settings in our experiments) [5],
- Kd-tree PatchMatch (KPM) [Sun, 2012], an improved version of PatchMatch based on better patch descriptors and kd-trees optimized for correspondence propagation [4],
- Non-Rigid Dense Correspondences (NRDC) [HaCohen et al., 2011], an improved version of GPM based on a multiscale iterative expansion/contraction strategy [6],

5. We used the online code.
6. We report results from the original paper.

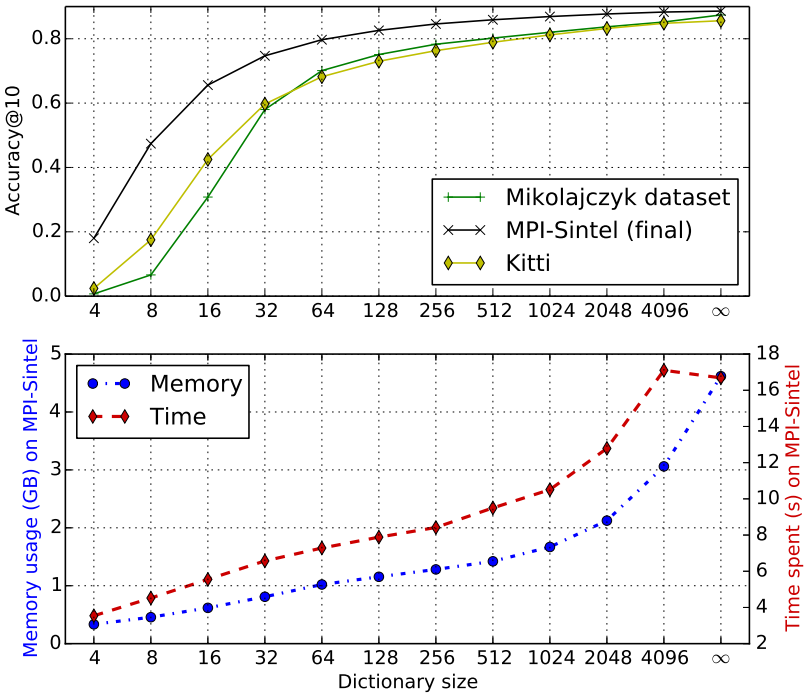


Figure 3.14 – Performance, memory usage and run-time for different levels of compression corresponding to the size D of the prototype dictionary (we set the image resolution to $R = 1/2$). A dictionary size $D = \infty$ stands for no compression. Run-times are for a single image pair on 1 core @ 3.6 GHz.

Proc. Unit	R	D	Patch clustering	Patch Correlations	Max-pooling +subsampling	Aggre-gation	Non-linear rectification	Back-tracking	Total time
CPU	1/2	64	0.3	0.2	0.4	0.9	0.8	5.1	7.7
CPU	1/2	1024	1.3	0.7	0.6	1.0	1.3	5.8	10.7
CPU	1/2	∞	-	4.3	1.6	1.0	3.2	6.2	16.4
GPU	1/2	∞	-	0.084	0.012	0.017	0.013	0.053	0.213

Table 3.2 – Detailed timings of the different stages of DeepMatching, measured for a single image pair from MPI-Sintel on CPU (1 core @ 3.6GHz) and GPU (GeForce Titan X) in seconds. Stages are: patch clustering (only for approximate DM, see Section 3.3.1), patch correlations (Equation 3.4), joint max-pooling and subsampling, correlation map aggregation, non linear rectification (resp. $\mathcal{S} \circ \mathcal{P}$, $\sum \mathcal{T}_{o_i}$, and $\mathcal{R}_\lambda$ in Equation 3.12), and correspondence backtracking (Section 3.2.3). Other operations (*e.g.* reciprocal verification of Equation 3.16) have negligible run-time. For operations applied at several levels like the non-linear rectification, a cumulative timing is given.

- SIFT-flow [Liu et al., 2011], a dense matching algorithm based on an energy minimization where pixels are represented as SIFT features and a smoothness term is incorporated to explicitly preserve spatial discontinuities [5],
- Scale-less SIFT (SLS) [Hassner et al., 2012], an improvement of SIFT-flow to handle scale changes (multiple sized SIFTs are extracted and combined to form a scale-invariant pixel representation) [5],
- DaisyFilterFlow (DaisyFF) [Yang et al., 2014], a dense matching approach that combines filter-based efficient flow inference and the Patch-Match fast search algorithm to match pixels described using the DAISY representation [Tola et al., 2010] [5],
- Deformable Pyramid Matching (DSP) [Kim et al., 2013], a dense matching approach based on a coarse-to-fine (top-down) strategy where inference is performed with (inexact) loopy belief propagation [5].

SIFT-NN, HOG-NN and DM output sparse matches, whereas the other methods output fully dense correspondence fields. SIFT keypoints, GPM, NRDC and DaisyFF are scale and rotation invariant, whereas HOG-NN, KPM, SIFT-flow, SLS and DSP are not. We, therefore, do not report results for these latter methods on the Mikolajczyk dataset which includes image rotations and scale changes.

Statistics about each method (average number of matches per image and their coverage) are reported in Table 3.3. Coverage is computed as the proportion of points on a regular grid with 10 pixel spacing for which there exists a correspondence (in the raw output of the considered method) within a 10 pixel neighborhood. Thus, it measures how well matches 'cover' the image. Table 3.3 shows that DeepMatching outputs 2 to 7 times more matches than SIFT-NN and a comparable number to HOG-NN. Yet, the coverage for DM matches is much higher than for HOG-NN and SIFT-NN. This shows that DM matches are well distributed over the entire image, which is not the case for HOG-NN and SIFT-NN, as they have difficulties estimating matches in regions with weak or repetitive textures.

Quantitative results are listed in Table 3.4, and qualitative results in Figures 3.15, 3.16 and 3.17. Overall, DM significantly outperforms all other methods, even when reduced settings are used (*e.g.* for image resolution $R = 1/2$ and $D = 1024$ prototypes). As expected, SIFT-NN performs rather well in presence of global image transformation (Mikolajczyk dataset), but yields the worst result for the case of more complex motions (flow datasets). Figures 3.16 and 3.17 illustrate the reason: SIFT's large patches are way too coarse to follow motion boundaries precisely. The same issue also holds for HOG-NN. Methods predicting dense correspondence fields return a more precise estimate, yet most of them (KPM, GPM, SIFT-flow, DSP) are not robust to repetitive textures in the Kitti dataset (Figure 3.17) as they rely

Method	Mikolajczyk		MPI-Sintel (final)		Kitti	
	#	coverage	#	coverage	#	coverage
SIFT-NN	2084	0.59	836	0.25	1299	0.38
HOG-NN	-	-	4576	0.39	4293	0.34
KPM	-	-	446K	1	462K	1
GPM	545K	1	446K	1	462K	1
NRDC	545K	1	446K	1	462K	1
SIFT-flow	-	-	446K	1	462K	1
SLS	-	-	446K	1	462K	1
DaisyFF	545K	1	446K	1	462K	1
DSP	-	-	446K	1	462K	1
DM (ours)	3120	0.81	5920	0.96	5357	0.88

Table 3.3 – Statistics of the different matching methods. The "#" column refers to the average number of matches per image, and the coverage to the proportion of points on a regular grid with 10 pixel spacing that have a match within a 10px neighborhood. We use the raw matches output by each method, *i.e.*, without any post-processing. Matches are not necessarily correct.

on weakly discriminative small patches. Despite this limitation, SIFT-flow and DSP are still able to perform well on MPI-Sintel as this dataset contains little scale changes. Other dense methods, NRDC, SLS and DaisyFF, can handle patches of different sizes and thus perform better on Kitti. But in turn this is at the cost of reduced performance on the MPI-Sintel or Mikolajczyk datasets (qualitative results are in Figure 3.15). In conclusion, DM outperforms all other methods on the 3 datasets, including DSP which also relies on a hierarchical matching.

In terms of computing resources, DeepMatching with full settings ($R = 1/2$, $D = \infty$) is one of the most costly method (only SLS and DaisyFF require the same order of memory and longer run-time). The scale and rotation invariant version of DM, used for the Mikolajczyk dataset, is slow compared to most other approaches, due to its sequential processing (*i.e.*, treating each combination of rotation and scaling sequentially), yet yields near perfect results. However, running DM with reduced settings is very competitive to the other approaches. On MPI-Sintel and Kitti, for instance, DM with a quarter resolution has a run-time comparable to the fastest method, SIFT-NN, with a reasonable memory usage, while still outperforming nearly all methods in terms of the accuracy@10 measure.

method	R	D	accuracy@10	memory usage	matching time
Mikolajczyk dataset					
SIFT-NN			0.674	0.2 GB	1.4 sec
GPM			0.303	0.1 GB	2.4 min
NRDC			0.692	0.1 GB	2.5 min
DaisyFF			0.410	6.1 GB	16 min
DM	1/4	∞	0.657	0.9 GB	38 sec
DM	1/2	1024	0.820	1.5 GB	4.5 min
DM	1/2	∞	**0.878**	4.4 GB	6.3 min
MPI-Sintel dataset (final)					
SIFT-NN			0.684	0.2 GB	2.7 sec
HOG-NN			0.712	3.4 GB	32 sec
KPM			0.738	0.3 GB	7.3 sec
GPM			0.812	0.1 GB	1.1 min
SIFT-flow			0.890	1.0 GB	29 sec
SLS			0.824	4.3 GB	16 min
DaisyFF			0.873	6.8 GB	12 min
DSP			0.853	0.8 GB	39 sec
DM	1/4	∞	0.835	0.3 GB	1.6 sec
DM	1/2	1024	0.869	1.8 GB	10 sec
DM	1/2	∞	**0.892**	4.6 GB	16 sec
Kitti dataset					
SIFT-NN			0.489	0.2 GB	1.7 sec
HOG-NN			0.537	2.9 GB	24 sec
KPM			0.536	0.3 GB	17 sec
GPM			0.661	0.1 GB	2.7 min
SIFT-flow			0.673	1.0 GB	25 sec
SLS			0.748	4.4 GB	17 min
DaisyFF			0.796	7.0 GB	11 min
DSP			0.580	0.8 GB	2.9 min
DM	1/4	∞	0.800	0.3 GB	1.6 sec
DM	1/2	1024	0.812	1.7 GB	10 sec
DM	1/2	∞	**0.856**	4.7 GB	14 sec

Table 3.4 – Matching performance, run-time and memory usage for state-of-the-art methods and DeepMatching (DM). For the proposed method, R and D denote the input image resolution and the dictionary size (∞ stands for no compression). Run-times are computed on 1 core @ 3.6 GHz.

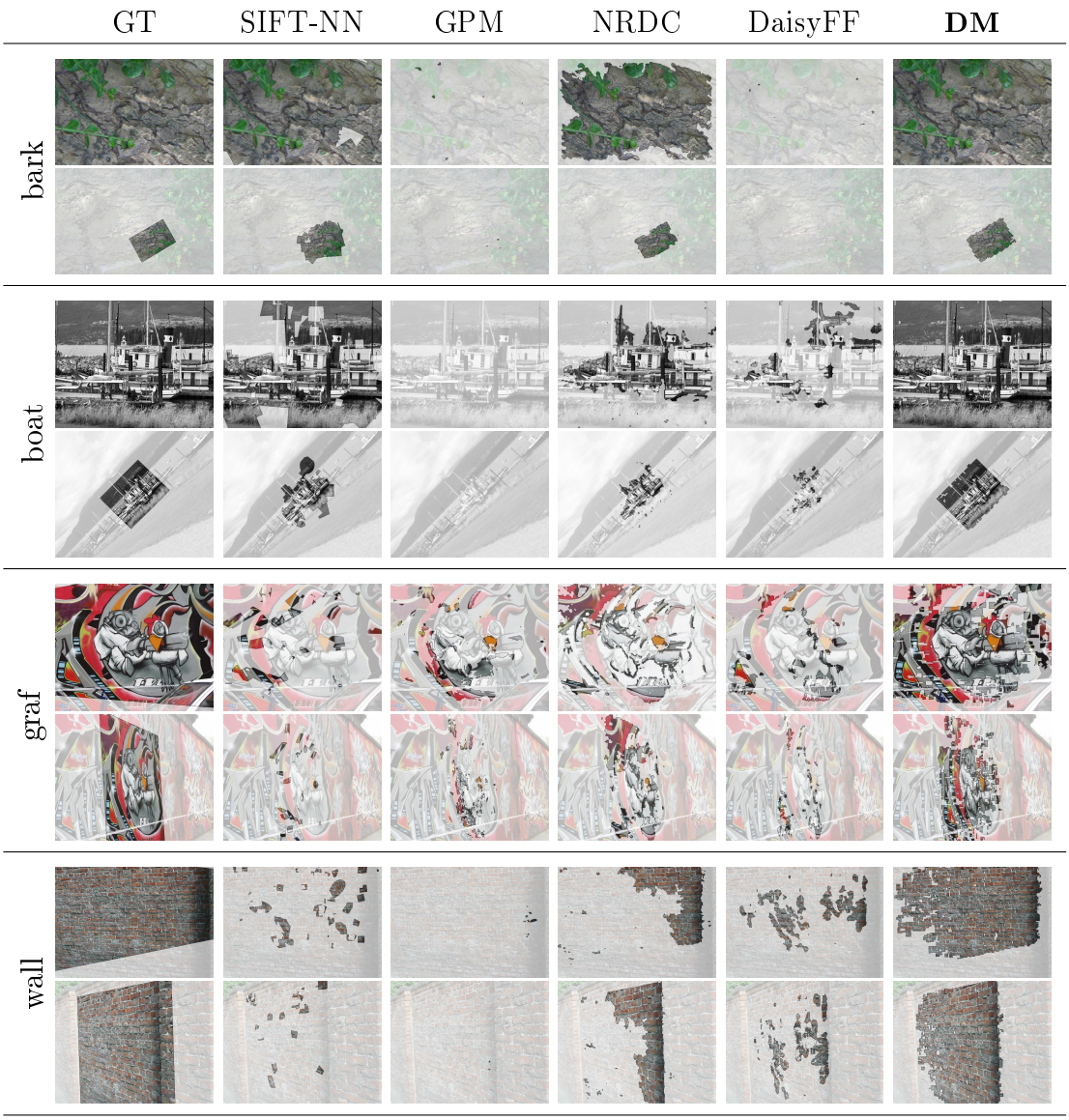


Figure 3.15 – Comparison of matching results of different methods on the Mikolajczyk dataset. Each column shows pixels with correct correspondences for different methods with from left to right: ground-truth (GT), SIFT-NN, GPM, NRDC and DeepMatching (DM). For each scene, we select two images to match and fade out regions which are unmatched, *i.e.*, those for which the matching error is above 15px or can not be matched. DeepMatching outperforms the other methods, especially on difficult cases like *graf* and *wall*.

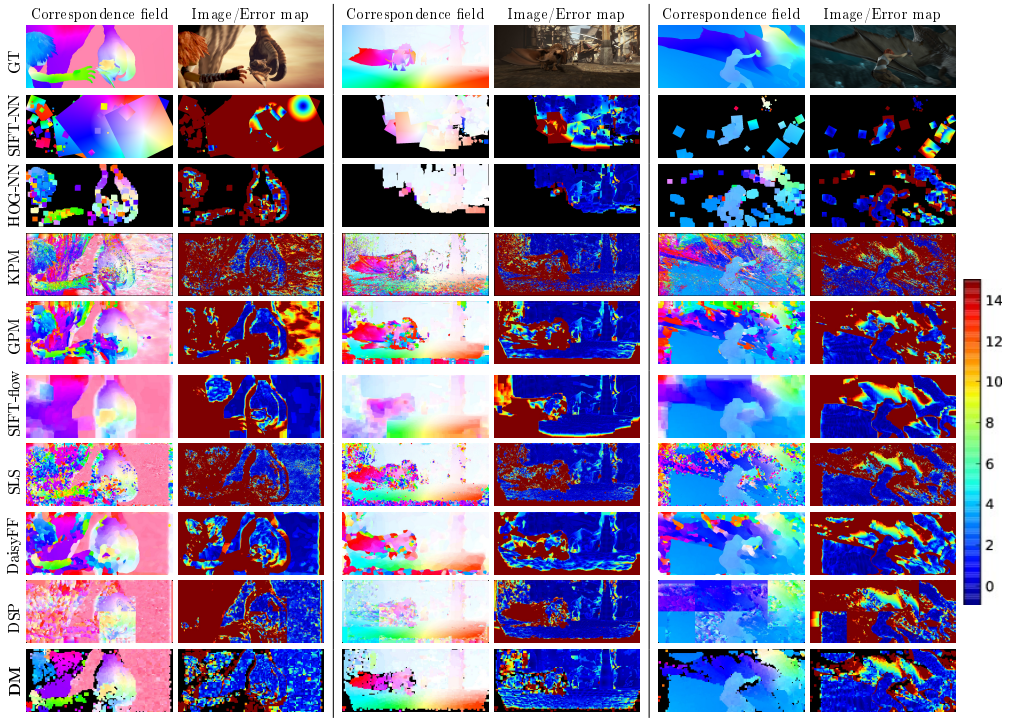


Figure 3.16 – Comparison of different matching methods on three challenging pairs with non-rigid deformations from MPI-Sintel. Each pair of columns shows motion maps (left column) and the corresponding error maps (right column). The top row presents the ground-truth (GT) as well as one image. For non-dense methods, pixel displacements have been inferred from matching patches. Areas without correspondences are in black.

3.5.3 Optical Flow Experiments

We now present experimental results for the optical flow estimation. Optical flow is predicted using the variational framework presented in Section 3.4 that takes as input a set of matches. In the following, we evaluate the impact of DeepMatching against other matching methods, and compare to the state of the art.

Optimization of the parameters

We optimize the parameters of DeepFlow on a subset of the MPI-Sintel training set (20%), called 'small' set, and report results on the remaining image pairs (80%, called 'validation set') and on the training sets of Kitti and Middlebury. Ground-truth optical flows for the three test sets are not publicly available, in order to prevent parameter tuning on the test set.

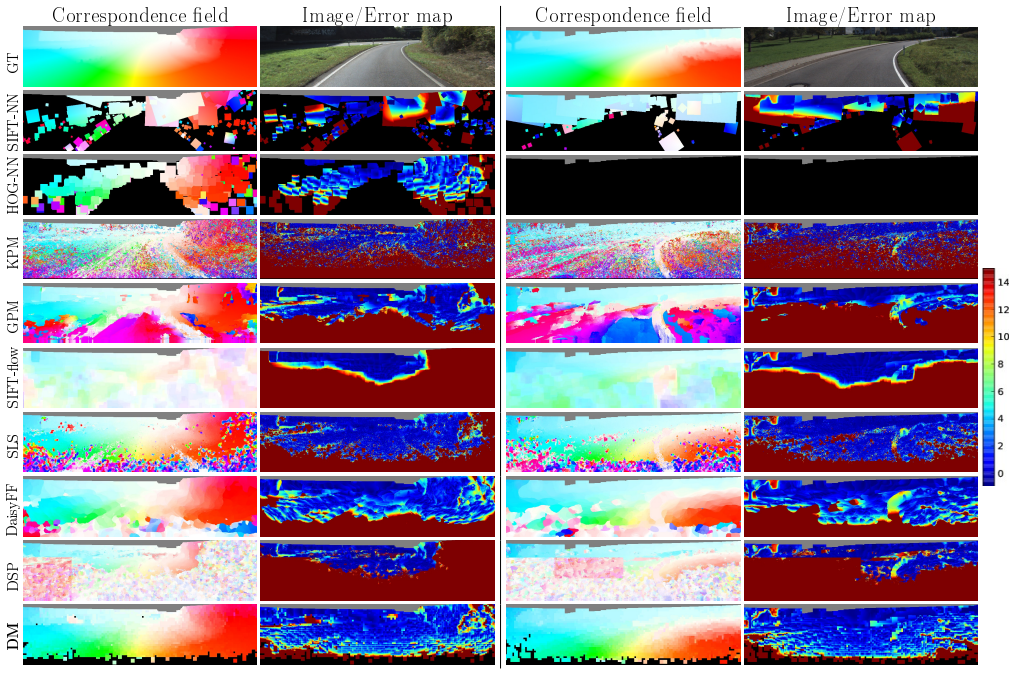


Figure 3.17 – Comparison of different matching methods on three challenging pairs from Kitti. Each pair of columns shows motion maps (left column) and the corresponding error maps (right column). The top row presents the ground-truth (GT) as well as one image. For non-dense methods, pixel displacements have been inferred from matching patches. Areas without correspondences are in black. To improve visualization, the sparse Kitti ground-truth is made dense using bilateral filtering.

We first optimize the different flow parameters (β, γ, δ, σ and b) by employing a gradient descent strategy with multiple initializations followed by a local grid search. For the data term, we find an optimum at $\delta = 0$, which is equivalent to removing the color constancy assumption. This can be explained by the fact that the 'final' version contains atmospheric effects, reflections, blurs, etc. The remaining parameters are optimal at $\beta = 300$, $\gamma = 0.8$, $\sigma = 0.5$, $b = 0.6$. These parameters are used in the remaining of the experiments for DeepFlow, *i.e.*, using matches obtained with DeepMatching, except when reporting results on Kitti and Middlebury test sets in Section 3.5.3. In this case the parameters are optimized on their respective training set.

Method	R	D	MPI-Sintel	Kitti	Middlebury
No Match			5.863	8.791	0.274
SIFT-NN			5.733	7.753	0.280
HOG-NN			5.458	8.071	**0.273**
KPM			5.560	15.289	0.275
GPM			5.561	17.491	0.286
SIFT-flow			5.243	12.778	0.283
SLS			5.307	10.366	0.288
DaisyFF			5.145	10.334	0.289
DSP			5.493	15.728	0.283
DM	1/2	1024	4.350	7.899	0.320
DM	1/2	∞	**4.098**	**4.407**	0.328

Table 3.5 – Comparison of average endpoint error on different datasets when changing the input matches in the flow computation.

Impact of the matches on the flow

We examine the impact of different matching methods on the flow, *i.e.*, different matches are used in DeepFlow, see Section 3.4. For all matching approaches evaluated in the previous section, we use their output as matching term in Equation 3.17. Because these approaches may output matches with statistics different from DM, we separately optimize the flow parameters for each matching approach on the small training set of MPI-Sintel[7].

Table 3.5 shows the endpoint error, averaged over all pixels. Clearly, a sufficiently dense and accurate matching like DM allows to considerably improve the flow estimation on datasets with large displacements (MPI-Sintel, Kitti). In contrast, none of the methods presented have a tangible effect on the Middlebury dataset, where the displacements are small.

The relatively small gains achieved by SIFT-NN and HOG-NN on MPI-Sintel and Kitti are due to the fact that a lot of regions with large displacements are not covered by any matches, such as the sky or the blurred character in the first and second column of Figure 3.18. Hence, SIFT-NN and HOG-NN have only a limited impact on the variational approach. On the other hand, the gains are also small (or even negative) for the dense methods despite the fact that they output significantly more correspondences. We observe for these methods that the weight β of the matching term tends to be small after optimizing the parameters, thus indicating that the matches are found unreliable and noisy during training. The cause

7. Note that this systematically improves the endpoint error compared to using the raw dense correspondence fields as flow.

is clearly visible in Figure 3.17, where large portions containing repetitive textures (*e.g.* road, trees) are incorrectly matched. The poor quality of these matches even leads to a significant drop in performance on the Kitti dataset.

In contrast, DeepMatching generates accurate matches well covering the image that enable to boost the optical flow accuracy in case of large displacements. Namely, we observe a relative improvement of 30% on MPI-Sintel and of 50% on Kitti. It is interesting to observe that DM is able to effectively prune false matches arising in occluded areas (black areas in Figures 3.16 and 3.17). This is due to the reciprocal verification filtering incorporated in DM (Equation 3.16). When using the approximation with 1024 prototypes, however, a significant drop is observed on the Kitti dataset, while the performance remains good on MPI-Sintel. This indicates that approximating DeepMatching can result in a significant loss of robustness when matching repetitive textures, that are more frequent in Kitti than in MPI-Sintel.

Comparison to the state of the art

In this section, we compare DeepFlow to the state of the art on the test sets of MPI-Sintel, Kitti and Middlebury datasets. For theses datasets, the results are submitted to a dedicated server which performs the evaluation. Prior to submitting our results for Kitti and Middlebury test sets, we have optimized the parameters on the respective training set.

Results on MPI-Sintel. Table 3.6 compares our method to state-of-the-art algorithms on the MPI-Sintel test set. A comparison with the preliminary version of DeepFlow [Weinzaepfel et al., 2013], referred to as DeepFlow*, is also provided. In this early version, we used a constant smoothness weight instead of a local one here (see Section 3.4) and used DM* as input matches. We can see that DeepFlow is among the best performing methods on MPI-Sintel, particularly for large displacements. This is due to the use of a reliable matching term in the variational approach, and this property is shared by all top performing approaches, *e.g.* [Revaud et al., 2015, Leordeanu et al., 2013]. Furthermore, it is interesting to note that among the top performers on MPI-Sintel, few methods actually employ DeepMatching. For instance, EpicFlow [Revaud et al., 2015] relies on the output of DeepMatching to produce a piece-wise affine flow, and SparseFlowFused [Timofte and Van Gool, 2015] combines matches obtained with DeepMatching and another algorithm.

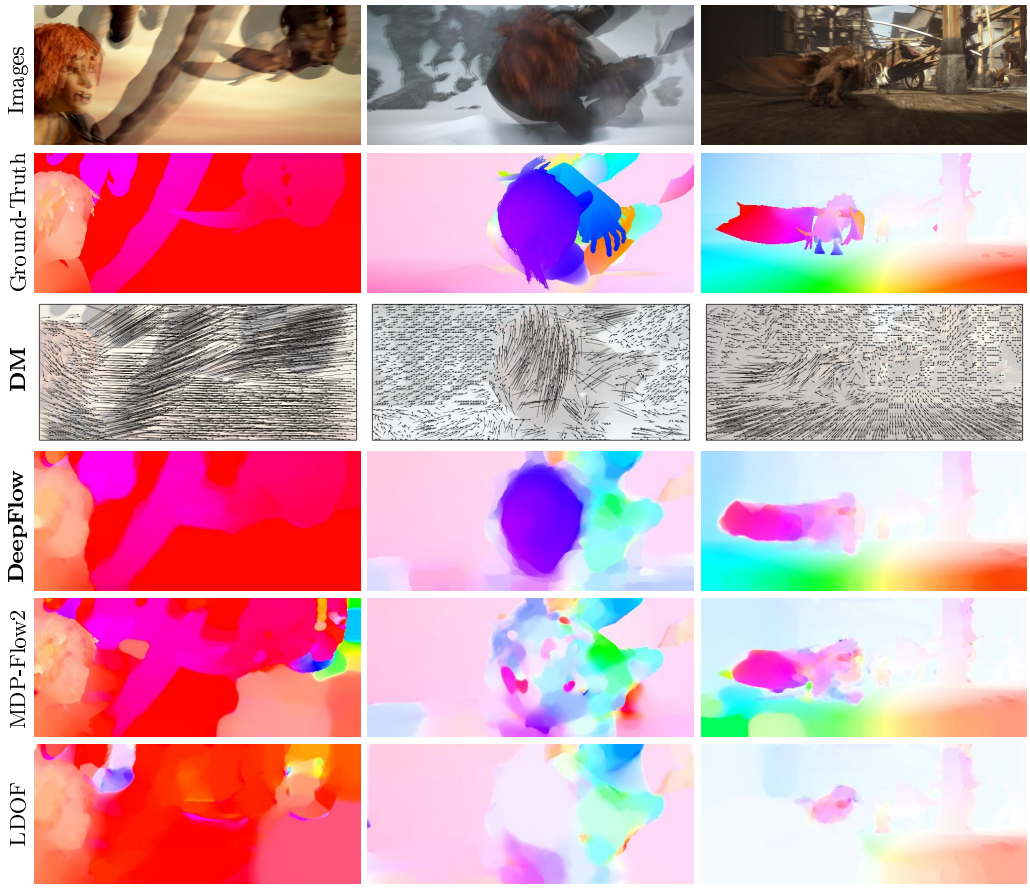


Figure 3.18 – Each column shows from top to bottom: two consecutive images, the ground-truth optical flow, the DeepMatching (DM), our flow prediction (DeepFlow), and two state-of-the-art methods, LDOF [Brox and Malik, 2011] and MDP-Flow2 [Xu et al., 2012].

We refer to the webpage of the MPI-Sintel dataset [8] for complete results including the 'clean' version.

Timings. As mentioned before, DeepMatching at half the resolution takes 15 seconds to compute on CPU and 0.2 second on GPU. The variational part requires 10 additional seconds on CPU. Note that by implementing it on GPU, we could obtain a significant speed-up as well. DeepFlow consequently takes 25 seconds in total on a single CPU core @ 3.6 GHz or 10.2s with GPU+CPU. This is in the same order of magnitude as the fastest among the best competitors, EpicFlow [Revaud et al., 2015].

8. http://sintel.is.tue.mpg.de/results

Method	EPE	EPE-occ	s0-10	s10-40	s40+	Time
FlowFields [Bailer et al., 2015]	5.810	31.799	1.157	3.739	33.890	23s
DiscreteFlow [Menze et al., 2015]	5.810	31.799	1.157	3.739	33.890	180s
EpicFlow [Revaud et al., 2015]	6.285	32.564	1.135	3.727	38.021	16.4s
TF+OFM [Kennedy and Taylor, 2015]	6.727	33.929	1.512	3.765	39.761	∼400s
DeepFlow	6.928	38.166	1.182	3.859	42.854	25s
SparseFlowFused [Timofte and Van Gool, 2015]	7.189	3.286	1.275	3.963	44.319	20s
DeepFlow* [Weinzaepfel et al., 2013]	7.212	38.781	1.284	4.107	44.118	19s
S2D-Matching [Leordeanu et al., 2013]	7.872	40.093	1.172	4.695	48.782	∼2000s
LocalLayering [Sun et al., 2014a]	8.043	40.879	1.186	4.990	49.426	
Classic+NL-P [Sun et al., 2014b]	8.291	40.925	1.208	5.090	51.162	∼800s
MDP-Flow2 [Xu et al., 2012]	8.445	43.430	1.420	5.449	50.507	709s
NLTGV-SC [Ranftl et al., 2014]	8.746	42.242	1.587	4.780	53.860	
LDOF [Brox and Malik, 2011]	9.116	42.344	1.485	4.839	57.296	30s

Table 3.6 – Results on MPI-Sintel test set (final version). EPE-occ is the EPE on occluded areas. s0-10 is the EPE for pixels with motions between 0 and 10 px and similarly for s10-40 and s40+. DeepFlow* denotes the preliminary version of DeepFlow published in [Weinzaepfel et al., 2013].

Results on Kitti. Table 3.7 summarizes the main results on the Kitti benchmark (see official website [9] for complete results), when optimizing the parameters on the Kitti training set. EPE-Noc is the EPE computed only in non-occluded areas. 'Out-all 3' corresponds to the proportion of incorrect pixel correspondences for an error threshold of 3 pixels, *i.e.*, it corresponds to 1 − accuracy@3, and likewise for 'Out-Noc 3' for non-occluded areas, see Section 2.5. In terms of EPE-noc, DeepFlow is on par with the best approaches, but performs somewhat worse in the occluded areas. This is due to a specificity of the Kitti dataset, in which motion is mostly homographic (especially on the image borders, where most surfaces like roads and walls are planar). In such cases, flow is better predicted using an affine motion prior, which locally well approximates homographies (a constant motion prior is used in DeepFlow). As a matter of facts, all top performing methods in terms of total EPE output piece-wise affine optical flow, either due to affine regularizers (BTF-ILLUM [Demetz et al., 2014], NLTGB-SC [Ranftl et al., 2014], TGV2ADCSIFT [Braux-Zin et al., 2013]) or due to local affine estimators (EpicFlow [Revaud et al., 2015]).

Note that the learned parameters on Kitti and MPI-Sintel are close. In particular, running the experiments with the same parameters as MPI-Sintel decreases EPE-Noc by only 0.1 pixel on the training set. This shows that our method does not suffer from overfitting.

9. http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=flow

Method	EPE-noc	EPE	Out-Noc 3	Out-all 3	Time
DiscreteFlow [Menze et al., 2015]	1.3	3.6	5.77%	16.63%	180s
FlowFields [Bailer et al., 2015]	1.4	3.5	6.23%	14.01%	23s
DeepFlow	1.4	5.3	6.61%	17.35%	22s
BTF-ILLUM [Demetz et al., 2014]	1.5	2.8	6.52%	11.03%	80s
EpicFlow [Revaud et al., 2015]	1.5	3.8	7.88%	17.08%	16s
TGV2ADCSIFT [Braux-Zin et al., 2013]	1.5	4.5	6.20%	15.15%	12s•
DeepFlow* [Weinzaepfel et al., 2013]	1.5	5.8	7.22%	17.79%	17s
NLTGV-SC [Ranftl et al., 2014]	1.6	3.8	5.93%	11.96%	16s•
Data-Flow [Vogel et al., 2013b]	1.9	5.5	7.11%	14.57%	180s
TF+OFM [Kennedy and Taylor, 2015]	2.0	5.0	10.22%	18.46%	350s

Table 3.7 – Results on Kitti test set. EPE-noc is the EPE over non-occluded areas. Out-Noc 3 (resp. Out 3) refers to the percentage of pixels where flow estimation has an error above 3 pixels in non-occluded areas (resp. all pixels). DeepFlow* denotes the preliminary version of DeepFlow published in Weinzaepfel et al. [2013]. • denotes the usage of a GPU.

Results on Middlebury. We optimize the parameters on the Middlebury training set by minimizing the average angular error with the same strategy as for MPI-Sintel. We find weights quasi-zero for the matching term due to the absence of large displacements. DeepFlow obtained an average endpoint error of 0.4 on the test which is competitive with the state of the art.

3.6 Conclusion

In this chapter we introduced DeepMatching, a dense matching algorithm tailored to optical flow estimation. The proposed algorithm gracefully handles complex non-rigid object deformations and regions with repetitive textures. Integrating DeepMatching into a variational formulation leads to a performance boost for large displacement optical flow.

Nevertheless, DeepFlow still suffers from inherent issues of a variational formulation optimized with a coarse-to-fine scheme, such as oversmoothing or missing small objects with fast motion. Indeed, even in the presence of correct matches for these objects, the coarse-to-fine scheme tends to miss their motion, mainly because the objects overlap at coarse scales and errors are not recovered at finer scales. Since DeepMatching already outputs quasi-dense matches, in the next chapter we propose an alternative to the coarse-to-fine scheme which is faster and outputs a more accurate optical flow.

Chapter 4

EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow

Contents

4.1 Introduction

The main challenges towards an accurate optical flow estimation in real-world videos are occlusions, motion discontinuities and large displacements. Large displacement optical flow approaches [Brox and Malik, 2011, Xu et al., 2012, Revaud et al., 2016] have recently emerged by integrating matching in

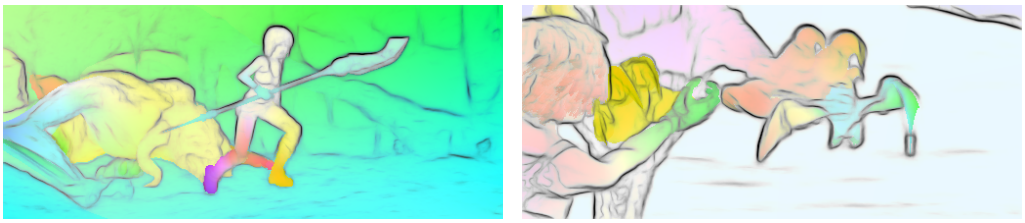


Figure 4.1 – Image edges detected with SED [Dollár and Zitnick, 2013] and ground-truth optical flow. Motion discontinuities appear most of the time at image edges.

a variational formulation. However, they still rely on coarse-to-fine scheme in order to obtain a full-scale dense flow field guided by the matches. A major drawback of coarse-to-fine schemes is error-propagation, *i.e.*, errors at coarser levels, where different motion layers can overlap, are propagated across scales. Even if coarse-to-fine techniques work well in most cases, we are not aware of a theoretical guarantee or proof of convergence.

Instead, this chapter introduces a novel approach that simply interpolates a sparse set of matches in a dense manner to initiate the optical flow estimation. We then use this estimate to initialize a one-level energy minimization problem, and obtain the final optical flow estimation. This enables us to leverage recent advances in matching algorithms, which can now output quasi-dense correspondence fields [Barnes et al., 2010, Revaud et al., 2016]. In the same spirit as Leordeanu et al. [2013], we perform a sparse-to-dense interpolation by fitting a local affine model at each pixel based on nearby matches. A major issue arises for the preservation of motion boundaries. We make the following observation: *motion boundaries often tend to appear at image edges*, see Figure 4.1. Consequently, we propose to exchange the Euclidean distance with a better, *i.e.*, edge-aware, distance and show that this offers a natural way to handle motion discontinuities. Moreover, we show how an approximation of the edge-aware distance allows to fit only one affine model per input match (instead of one per pixel). This leads to an important speed-up of the interpolation scheme without loss in performance.

The obtained interpolated field of correspondences is sufficiently accurate to be used as initialization of a one-level energy minimization. Our work suggests that there may be better initialization strategies than the well-established coarse-to-fine scheme, see Figure 4.2. In particular, our approach, *EpicFlow* (edge-preserving interpolation of correspondences) shows excellent performance on the challenging MPI-Sintel dataset [Butler et al., 2012] and is competitive on Kitti [Geiger et al., 2013] and Middlebury [Baker

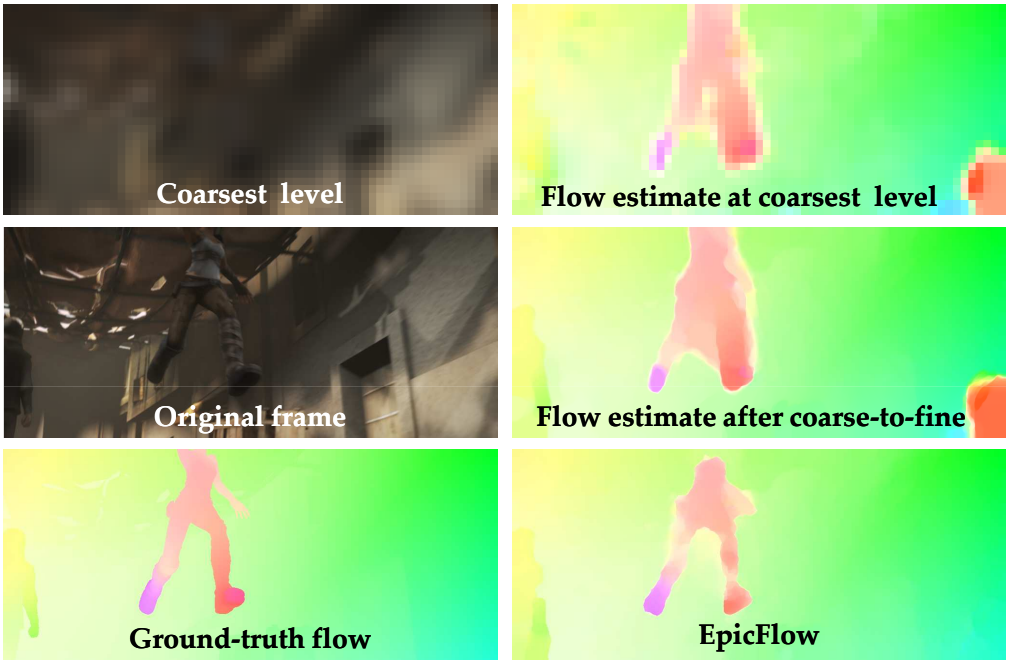


Figure 4.2 – Comparison of coarse-to-fine flow estimation and EpicFlow. Errors at the coarsest level of estimation, due to a low resolution, often get propagated to the finest level (right, top and middle). In contrast, our interpolation scheme benefits from an edge prior at the finest level (right, bottom).

et al., 2011]. An overview of EpicFlow is given in Figure 4.3. To summarize, we make three main contributions:

- We propose *EpicFlow*, a novel sparse-to-dense interpolation scheme of matches based on an edge-aware distance. We show that it is robust to motion boundaries, occlusions and large displacements.

- We propose an approximation scheme for the edge-aware distance, leading to a significant speed-up without loss of accuracy.

- We show empirically that the proposed optical flow estimation scheme is more accurate than estimations based on coarse-to-fine minimization.

Closest references. Several works have proposed to integrate matches in an optical flow formulation [Brox and Malik, 2011, Xu et al., 2012, Weinzaepfel et al., 2013]. However, these methods rely on a coarse-to-fine scheme, that suffers from intrinsic flaws. Namely, details are lost at coarse scales,

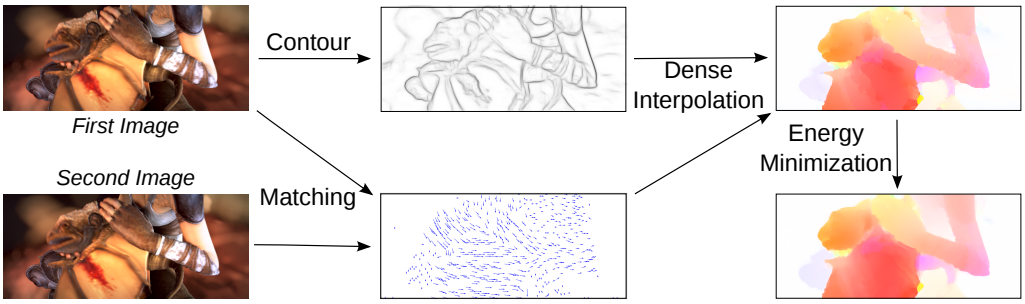


Figure 4.3 – Overview of EpicFlow. Given two images, we compute matches using DeepMatching [Weinzaepfel et al., 2013] and the edges of the first image using SED [Dollár and Zitnick, 2013]. We combine these two cues to densely interpolate matches and obtain a dense correspondence field. This is used as initialization of a one-level energy minimization framework.

and thin objects with substantially different motions cannot be detected. These errors correspond to local minima, hence they cannot be recovered and are propagated across levels, see Figure 4.2.

In contrast, EpicFlow relies on sparse-to-dense interpolation of matches while taking into account image edges. In the same spirit, Ren [2008] proposes to use edge-based affinities to group pixels and estimate a piece-wise affine flow. Nevertheless, this work relies on a discretization of the optical flow constraint, which is valid only for small displacements. Closely related to EpicFlow, Leordeanu et al. [2013] also investigate sparse-to-dense interpolation. Their initial matching is obtained through the costly minimization of a global non-convex matching energy. In contrast, we directly use state-of-the-art matches [Revaud et al., 2016, He and Sun, 2012] as input. Furthermore, during their sparse-to-dense interpolation, they compute an affine transformation independently for each pixel based on its neighborhood matches, which are found in a Euclidean ball and weighted by an estimation of occluded areas that involves learning a binary classifier. In contrast, we propose to use an edge-preserving distance that naturally handles occlusions, and can be very efficiently computed.

Outline. This chapter is organized as follows. We start by presenting the sparse-to-dense interpolation as well as its approximated scheme in Section 4.2. We then describe the energy minimization for optical flow computation in Section 4.3. Finally, Section 4.4 presents experimental results. Source code is available online at `http://lear.inrialpes.fr/software`.

4.2 Sparse-to-dense interpolation

The proposed approach, EpicFlow, consists of three steps, as illustrated in Figure 4.3. First, we compute a sparse set of matches between the two images, using a state-of-the-art matching algorithm. Second, we perform a densification of this set of matches, by computing a sparse-to-dense interpolation from the sparse set of matches, which yields an initial estimate of the optical flow. Third, we compute the final optical flow estimation by performing one step of variational energy minimization using the dense interpolation as initialization, see Section 4.3.

4.2.1 Sparse set of matches

The first step of our approach extracts a sparse set of matches, see Figure 4.3. Any state-of-the-art matching algorithm can be used to compute the initial set of sparse matches. In our experiments, we compare the results when using DeepMatching [Weinzaepfel et al., 2013] or a subset of an estimated nearest-neighbor field [He and Sun, 2012]. We defer to Section 4.4.1 for a description of these matching algorithms. In both cases, we obtain ~ 5000 matches for an image of resolution 1024×436, *i.e.*, an average of around one match per 90 pixels. We also evaluate the impact of matching quality and density on the performance of EpicFlow by generating artificial matches from the ground-truth in Section 4.4.3. In the following, we denote by $\mathcal{M} = \{(\boldsymbol{p}_m, \boldsymbol{p}'_m)\}$ the sparse set of input matches, where each match $(\boldsymbol{p}_m, \boldsymbol{p}'_m)$ defines a correspondence between a pixel $\boldsymbol{p}_m$ in the first image and and a pixel $\boldsymbol{p}'_m$ in the second image.

4.2.2 Interpolation method

We estimate a dense correspondence field $\boldsymbol{F} : I \to I'$ between a source image I and a target image I' by interpolating a sparse set of inputs matches $\mathcal{M} = \{(\boldsymbol{p}_m, \boldsymbol{p}'_m)\}$. The interpolation requires a distance $D : I \times I \to \mathbb{R}^+$ between pixels, see Section 4.2.3. We consider here two options for the interpolation.

- **Nadaraya-Watson (NW) estimation [Wasserman, 2010].** The correspondence field $\boldsymbol{F}_{NW}(\boldsymbol{p})$ is interpolated using the Nadaraya-Watson estimator at a pixel $\boldsymbol{p} \in I$ and is expressed by a sum of matches weighted

by their proximity to $\boldsymbol{p}$:

$$\boldsymbol{F}_{NW}(\boldsymbol{p}) = \frac{\sum\limits_{(\boldsymbol{p}_m, \boldsymbol{p}'_m) \in \mathcal{M}} k_D(\boldsymbol{p}_m, \boldsymbol{p}) \boldsymbol{p}'_m}{\sum\limits_{(\boldsymbol{p}_m, \boldsymbol{p}'_m) \in \mathcal{M}} k_D(\boldsymbol{p}_m, \boldsymbol{p})} \quad , \tag{4.1}$$

where $k_D(\boldsymbol{p}_m, \boldsymbol{p}) = \exp\left(-aD(\boldsymbol{p}_m, \boldsymbol{p})\right)$ is a Gaussian kernel for a distance D with a parameter a.

- **Locally-weighted affine (LA) estimation [Hartley and Zisserman, 2003].** The second estimator is based on fitting a local affine transformation. The correspondence field $\boldsymbol{F}_{LA}(\boldsymbol{p})$ is interpolated using a locally-weighted affine estimator at a pixel $\boldsymbol{p} \in I$ as $\boldsymbol{F}_{LA}(\boldsymbol{p}) = A_{\boldsymbol{p}}\boldsymbol{p} + t_{\boldsymbol{p}}^{\top}$, where $A_{\boldsymbol{p}}$ and $t_{\boldsymbol{p}}$ are the parameters of an affine transformation estimated for the pixel $\boldsymbol{p}$. These parameters are computed as the least-square solution of an overdetermined system obtained by writing two equations for each match $(\boldsymbol{p}_m, \boldsymbol{p}'_m) \in \mathcal{M}$ weighted as previously:

$$k_D(\boldsymbol{p}_m, \boldsymbol{p}) \left(A_{\boldsymbol{p}}\boldsymbol{p}_m + t_{\boldsymbol{p}}^{\top} - \boldsymbol{p}'_m\right) = 0 \quad . \tag{4.2}$$

Local interpolation. Note that the influence of remote matches is either negligible, or could harm the interpolation, for example when objects move differently. Therefore, we restrict the set of matches used in the interpolation at a pixel $\boldsymbol{p}$ to its K nearest neighbors according to the distance D, which we denote as $\mathcal{N}_K(\boldsymbol{p})$. In other words, we replace the summation over $\mathcal{M}$ in the NW operator by a summation over $\mathcal{N}_K(\boldsymbol{p})$, and likewise for building the overdetermined system to fit the affine transformation for $\boldsymbol{F}_{LA}$.

4.2.3 Edge-preserving distance

Using the Euclidean distance for the interpolation presented above is possible. However, in this case, the interpolation is simply based on the position of the input matches and does not respect motion boundaries. Suppose for a moment that the motion boundaries are known. We can, then, use a geodesic distance D_G based on these motion boundaries. Formally, the geodesic distance between two pixels $\boldsymbol{p}$ and $\boldsymbol{q}$ is defined as the shortest distance with respect to a cost map C:

$$D_G(\boldsymbol{p}, \boldsymbol{q}) = \inf_{\Gamma \in \mathcal{P}_{\boldsymbol{p},\boldsymbol{q}}} \int_{\Gamma} C(\boldsymbol{p}_s) d\boldsymbol{p}_s \quad , \tag{4.3}$$

where $\mathcal{P}_{\boldsymbol{p},\boldsymbol{q}}$ denotes the set of all possible paths between $\boldsymbol{p}$ and $\boldsymbol{q}$, and $C(\boldsymbol{p}_s)$ the cost of crossing pixel $\boldsymbol{p}_s$ (the viscosity in physics). In our settings, C

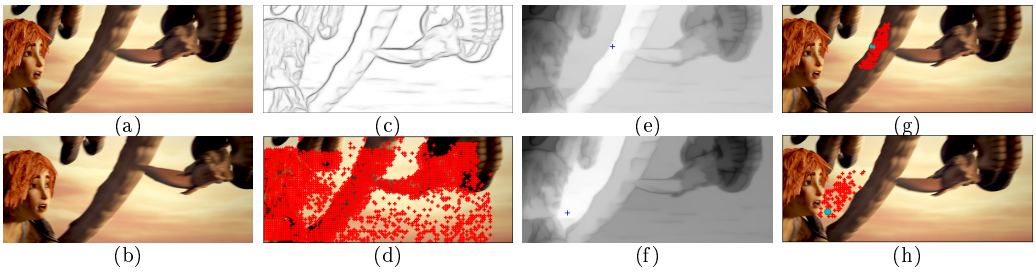


Figure 4.4 – (a-b) two consecutive frames; (c) contour response C from SED [Dollár and Zitnick, 2013] (the darker, the higher); (d) match positions $\{\boldsymbol{p}_m\}$ from DeepMatching [Weinzaepfel et al., 2013]; (e-f) geodesic distance from a pixel $\boldsymbol{p}$ (marked in blue) to all others $D_G(\boldsymbol{p}, .)$ (the brighter, the closer); (g-h) 100 nearest matches, *i.e.*, $\mathcal{N}_{100}(\boldsymbol{p})$ (red) using geodesic distance D_G from the pixel $\boldsymbol{p}$ in blue.

corresponds to the motion boundaries. Hence, a pixel belonging to a motion layer is close to all other pixels from the same layer according to D_G, but far from everything beyond the boundaries. Since each pixel is interpolated based on its neighbors, the interpolation will respect the motion boundaries.

In practice, we use an alternative to true motion boundaries, making the plausible assumption that *image edges* are a superset of *motion boundaries*. This way, the distance between pixels belonging to the same region will be low. It ensures a proper edge-respecting interpolation as long as the number of matches in each region is sufficient. Similarly, Criminisi et al. [2010] showed that geodesic distances are a natural tool for edge-preserving image editing operations (denoising, texture flattening, etc.) and it was also used recently to generate object proposals [Krähenbühl and Koltun, 2014]. In practice, we set the cost map C using a recent state-of-the-art edge detector, namely the 'structured edge detector' (SED) [Dollár and Zitnick, 2013] [1]. Figure 4.4 shows an example of a SED map, as well as examples of geodesic distances and neighbor sets $\mathcal{N}_K(\boldsymbol{p})$ for different pixels $\boldsymbol{p}$. Notice how neighbors are found on the same objects/parts of the image with D_G, in contrast to Euclidean distance (see also Figure 4.6).

4.2.4 Fast approximation

The geodesic distance can be rapidly computed from a point to all other pixels. For instance, Weber et al. [2008] propose parallel algorithms that simulate an advancing wavefront. Nevertheless, the computational cost for

1. https://github.com/pdollar/edges

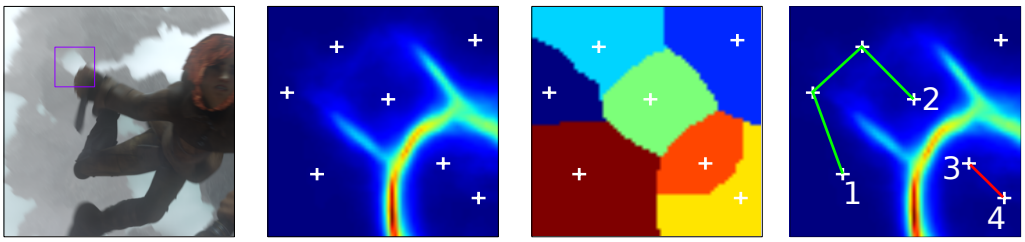


Figure 4.5 – For the region shown in (a), (b) shows the image edges C and white crosses representing the match positions $\{\boldsymbol{p}_m\}$. (c) displays the assignment L, *i.e.*, geodesic Voronoi cells. We build a graph $\mathcal{G}$ from L (see text). (d) shows the shortest path between two neighbor matches, which can go through the edge that connects them (3-4) or a shorter path found by Dijkstra's algorithm (1-2).

computing the geodesic distance between all pixels and all matches (as required by our interpolation scheme) is high. We now propose an efficient approximation $\tilde{D}_G$.

A key observation is that neighboring pixels are often interpolated similarly, suggesting a strategy that would leverage such local information. In this section, we employ the term 'match' to refer to $\boldsymbol{p}_m$ instead of $(\boldsymbol{p}_m, \boldsymbol{p}'_m)$.

Geodesic Voronoi diagram. We first define a clustering L, such that $L(\boldsymbol{p})$ assigns a pixel $\boldsymbol{p}$ to its closest match according to the geodesic distance, *i.e.*, we have $L(\boldsymbol{p}) = \text{argmin}_{\boldsymbol{p}_m} D_G(\boldsymbol{p}, \boldsymbol{p}_m)$. L defines geodesic Voronoi cells, as shown in Figure 4.5(c).

Approximated geodesic distance. We then approximate the distance between a pixel $\boldsymbol{p}$ and any match $\boldsymbol{p}_m$ as the distance to the closest match $L(\boldsymbol{p})$ plus an approximate distance between matches:

$$\tilde{D}_G(\boldsymbol{p}, \boldsymbol{p}_m) = D_G(\boldsymbol{p}, L(\boldsymbol{p})) + D^{\mathcal{G}}_G(L(\boldsymbol{p}), \boldsymbol{p}_m) \quad , \tag{4.4}$$

where $D^{\mathcal{G}}_G$ is a graph-based approximation of the geodesic distance between two matches. To define $D^{\mathcal{G}}_G$ we use a neighborhood graph $\mathcal{G}$ whose nodes are $\{\boldsymbol{p}_m\}$. Two matches $\boldsymbol{p}_m$ and $\boldsymbol{p}_n$ are connected by an edge if they are neighbors in L. The edge weight is then defined as the geodesic distance between $\boldsymbol{p}_m$ and $\boldsymbol{p}_n$, where the geodesic distance calculation is restricted to the Voronoi cells of $\boldsymbol{p}_m$ and $\boldsymbol{p}_n$. We, then, calculate the approximate geodesic distance between any two matches $\boldsymbol{p}_m, \boldsymbol{p}_n$ using Dijkstra's algorithm on $\mathcal{G}$, see Figure 4.5(d).

Piecewise field. So far, we have built an approximation of the distance between pixels and match points. We now show that our interpolation model results in a piece-wise correspondence field (either constant for the Nadaraya-Watson estimator, or piece-wise affine for LA). This property is crucial to obtain a fast interpolation scheme, and experiments shows that it does not impact the accuracy. Let us consider a pixel $\boldsymbol{p}$ such that $L(\boldsymbol{p}) = \boldsymbol{p}_m$. The distance between $\boldsymbol{p}$ and any match $\boldsymbol{p}_n$ is the same as the one between $\boldsymbol{p}_m$ and $\boldsymbol{p}_n$ up to a constant independent from $\boldsymbol{p}_n$ (Equation 4.4). As a consequence, we have $\mathcal{N}_K(\boldsymbol{p}) = \mathcal{N}_K(\boldsymbol{p}_m)$ and $k_{\tilde{D}_G}(\boldsymbol{p}, \boldsymbol{p}_n) = k_{D_G}(\boldsymbol{p}, \boldsymbol{p}_m) \times k_{D_G^{\mathcal{G}}}(\boldsymbol{p}_m, \boldsymbol{p}_n)$. For the Nadaraya-Watson estimator, we thus obtain:

$$\begin{aligned} \boldsymbol{F}_{NW}(\boldsymbol{p}) &= \frac{\sum_{(\boldsymbol{p}_n, \boldsymbol{p}'_n)} k_{\tilde{D}_G}(\boldsymbol{p}, \boldsymbol{p}_n) \boldsymbol{p}'_n}{\sum_{(\boldsymbol{p}_n, \boldsymbol{p}'_n)} k_{\tilde{D}_G}(\boldsymbol{p}, \boldsymbol{p}_n)} \\ &= \frac{k_{D_G}(\boldsymbol{p}, \boldsymbol{p}_m) \sum_{(\boldsymbol{p}_n, \boldsymbol{p}'_n)} k_{D_G^{\mathcal{G}}}(\boldsymbol{p}_m, \boldsymbol{p}_n) \boldsymbol{p}'_n}{k_{D_G}(\boldsymbol{p}, \boldsymbol{p}_m) \sum_{(\boldsymbol{p}_n, \boldsymbol{p}'_n)} k_{D_G^{\mathcal{G}}}(\boldsymbol{p}_m, \boldsymbol{p}_n)} = \boldsymbol{F}_{NW}(\boldsymbol{p}_m) \quad , \end{aligned} \tag{4.5}$$

where all the sums are for $(\boldsymbol{p}_n, \boldsymbol{p}'_n) \in \mathcal{N}_K(\boldsymbol{p}) = \mathcal{N}_K(\boldsymbol{p}_m)$. The same reasoning holds for the weighted affine interpolator, which is invariant to a multiplication of the weights by a constant factor. As a consequence, it suffices to compute $|\mathcal{M}|$ estimations (one per match) and to propagate it to the pixel assigned to this match. This is orders of magnitude faster than an independent estimation for each pixel, *e.g.* as done by Leordeanu et al. [2013]. We summarize the approach in Algorithm 4.1 for Nadaraya-Watson estimator. The algorithm is similar for LA interpolator (*e.g.* line 6 becomes 'Estimate affine parameters $A_{\boldsymbol{p}_m}, t_{\boldsymbol{p}_m}$' and line 8 'Set $\boldsymbol{W}_{LA}(\boldsymbol{p}) = A_{L(\boldsymbol{p})}\boldsymbol{p} + t^{\top}_{L(\boldsymbol{p})}$').

Algorithm 4.1 Interpolation with Nadaraya-Watson.

Input: a pair of images I, I', a set $\mathcal{M}$ of matches
Output: dense correspondence field $\boldsymbol{F}_{NW}$

1 Compute the cost C for I using SED [Dollár and Zitnick, 2013]
2 Compute the assignment map L
3 Build the graph $\mathcal{G}$ from L
4 **For** $(\boldsymbol{p}_m, \boldsymbol{p}'_m) \in \mathcal{M}$
5 —– Compute $\mathcal{N}_K(\boldsymbol{p}_m)$ from $\mathcal{G}$ using Dijkstra's algorithm
6 —– Compute $\boldsymbol{F}_{NW}(\boldsymbol{p}_m)$ from $\mathcal{N}_K(\boldsymbol{p}_m)$ using Equation 4.1
7 **For** each pixel $\boldsymbol{p}$
8 —– Set $\boldsymbol{F}_{NW}(\boldsymbol{p}) = \boldsymbol{F}_{NW}(L(\boldsymbol{p}))$

4.3 Optical Flow Estimation

Coarse-to-fine versus EpicFlow. The output of the sparse-to-dense interpolation is a dense correspondence field. This field is used as initialization of a variational energy minimization method. In contrast to our approach, state-of-the-art methods usually rely on a coarse-to-fine scheme to compute the full-scale correspondence field. To the best of our knowledge, there exists no theoretical proof or guarantee that a coarse-to-fine minimization leads to a consistent estimation that accurately minimizes the full-scale energy. Thus, the coarse-to-fine scheme should be considered as a heuristic to provide an initialization for the full-scale flow.

Our approach can be thought of as an alternative to the above strategy, by offering a smart heuristic to accurately initialize the optical flow before performing energy minimization at the full-scale. This offers several advantages over the coarse-to-fine scheme. First, the cost map C in our method acts as a prior on boundary location. Such a prior could also be incorporated by a local smoothness weight in the coarse-to-fine minimization, but would then be difficult to interpret at coarse scales where boundaries might strongly overlap. In addition, since our method directly works at the full image resolution, it avoids possible issues related to the presence of thin objects that could be oversmoothed at coarse scales. Such errors at coarse scales are propagated to finer scales as the coarse-to-fine approach proceeds, see Figure 4.2.

Variational Energy Minimization. We minimize an energy defined as a sum of a data term and a smoothness term. We use the same data term as Zimmer et al. [2011], based on a classical color-constancy and gradient-constancy assumption with a normalization factor. For the smoothness term, we penalize the flow gradient norm, with a local smoothness weight α as in [Wedel et al., 2009a, Xu et al., 2012]: $\alpha(\boldsymbol{x}) = \exp\big(-\kappa\|\nabla_2 I(\boldsymbol{x})\|\big)$ with $\kappa = 5$. We have also experimented using SED instead and obtained similar performance.

For minimization, we initialize the solution with the output of our sparse-to-dense interpolation and use the approach of Brox et al. [2004] without the coarse-to-fine scheme. More precisely, we perform 5 fixed point iterations, *i.e.*, compute the non-linear weights (that appear when applying Euler-Lagrange equations [Brox et al., 2004]) and the flow updates 5 times iteratively. The flow updates are computed by solving linear systems using 30 iterations of the successive over relaxation method [Young and Rheinboldt, 1971].

4.4 Experiments

In this section, we evaluate EpicFlow on three state-of-the-art datasets: the MPI-Sintel dataset, the Kitti dataset and the Middlebury dataset. We refer to Section 2.5 for details about these benchmarks.

As in [Weinzaepfel et al., 2013], we optimize the parameters on a subset (20%) of the MPI-Sintel training set. We then report average endpoint error (EPE) on the remaining MPI-Sintel training set (80%), the Kitti training set and the Middlebury training set. This allows us to evaluate the impact of parameters on different datasets and avoid overfitting. The parameters are typically $a \simeq 1$ for the coefficient in the kernel k_D, the number of neighbors is $K \simeq 25$ for NW interpolation and $K \simeq 100$ when using LA. Timing is reported for one CPU-core at 3.6GHz.

In the following, we first describe two types of input matches in Section 4.4.1. Section 4.4.2 then studies the different parameters of our approach. In Section 4.4.3, we compare our method to a variational approach with a coarse-to-fine scheme. Finally, we compare EpicFlow with the state of the art on the test sets in Section 4.4.4. In this case, the parameters are optimized on the training set of the corresponding dataset.

4.4.1 Input matches

To generate input matches, we use and compare two recent matching algorithms. They each produce about 5000 matches per image.

- The first one is DeepMatching (**DM**), used in DeepFlow [Weinzaepfel et al., 2013], which has shown excellent performance for optical flow. It builds correspondences by computing similarities of non-rigid patches, allowing for some deformations. We use the online code [2] on images downscaled by a factor 2. A reciprocal verification is included in DM. As a consequence, the majority of matches in occluded areas are pruned, see matches in Figure 4.6 (left).
- The second one is a recent variant of PatchMatch [Barnes et al., 2010] that relies on kd-trees and local propagation to compute a dense correspondence field [He and Sun, 2012] (**KPM**). We use the online code to extract the dense correspondence field [3]. It is noisy, as it is based on small patches without global regularization, as well as often incorrect in case of occlusion. Thus, we perform a two-way matching and eliminate non-reciprocal matches to remove incorrect correspondences. We also subsample these

2. http://lear.inrialpes.fr/src/deepmatching/
3. http://j0sh.github.io/thesis/kdtree/

Figure 4.6 – *Left:* Match positions returned by DeepMatching [Weinzaepfel et al., 2013] are shown in blue. Red denotes occluded areas. *Right:* Yellow (resp. blue) squares correspond to the 100 nearest matches with a Euclidean (resp. edge-aware geodesic) distance for the occluded pixel shown in red. All neighbor matches with a Euclidean distance holds to a different object while the geodesic distance allows to capture matches from the same region, even in the case of large occluded areas.

pruned correspondences to speed-up the interpolation. We have experimentally verified on several image pairs that this subsampling does not result in a loss of performance.

Pruning of matches. In both cases, matches are extracted locally and might be incorrect in regions with low texture. Thus, we remove matches corresponding to patches with low saliency, which are determined by the eigenvalues of autocorrelation matrix. Furthermore, we perform a consistency check to remove outliers. We run the sparse-to-dense interpolation once with the Nadaraya-Watson estimator and remove matches for which the difference to the initial estimate is over 5 pixels.

We also experiment with synthetic sparse matches of various densities and noise levels in Section 4.4.3, in order to evaluate the sensitivity of EpicFlow to the quality of the matching approach.

4.4.2 Impact of the different parameters

In this section, we evaluate the impact of the matches and the interpolator. We also compare the quality of the sparse-to-dense interpolation and EpicFlow. Furthermore, we examine the impact of the geodesic distance and its approximation as well as the impact of the quality of the contour detector.

Matches and interpolators. Table 4.1 compares the result of our sparse-to-dense interpolation, *i.e.*, before energy minimization, and EpicFlow for different matches (**DM** and **KPM**) and for the two interpolation schemes:

	Matching	Interpolator	MPI-Sintel	Kitti	Middlebury
Interpolation	KPM	NW	6.052	15.679	0.765
	KPM	LA	6.334	12.011	0.776
	DM	NW	4.143	5.460	0.898
	DM	LA	4.068	3.560	0.840
EpicFlow	KPM	NW	5.741	15.240	0.388
	KPM	LA	5.764	11.307	**0.315**
	DM	NW	3.804	4.900	0.485
	DM	LA	**3.686**	**3.334**	0.380

Table 4.1 – Comparison of average endpoint error (EPE) for different sparse matches (DM, KPM) and interpolators (NW, LA) as well as for sparse-to-dense interpolation (top) and EpicFlow (bottom). The approximated geodesic distance $\tilde{D}_G$ is used.

Nadaraya-Watson (**NW**) and locally-weighted affine (**LA**). The approximated geodesic distance is used in the interpolation, see Section 4.2.4.

We can observe that KPM is consistently outperformed by DeepMatching (DM) on MPI-Sintel and Kitti datasets, with a gap of 2 and 8 pixels respectively. Kitti contains many repetitive textures like trees or roads, which are often mismatched by KPM. Note that DM is significantly more robust to repetitive textures than KPM, as it uses a multi-scale scoring scheme. The results on Middlebury are comparable and below 1 pixel.

We also observe that LA performs better than NW on Kitti, while the results are comparable on MPI-Sintel and Middlebury. This is due to the specificity of the Kitti dataset, where the scene consists of planar surfaces and, thus, affine transformations are more suitable than translations to approximate the flow. Based on these results, we use DM matches and LA interpolation in the remainder of the experimental section.

The interpolation is robust to the neighborhood size K with for instance an EPE of $4.082, 4.053, 4.068$ and 4.076 for $K = 50, 100, 160$ (optimal value on the training set), 200 respectively, on MPI-Sintel with the LA estimator and before variational minimization. We also implemented a variant where we use all matches closer than a threshold and obtained similar performance.

Sparse-to-dense interpolation versus EpicFlow. We also evaluate the gain due to the variational minimization using the interpolation as initialization. We can see in Table 4.1 that this step clearly improves the performance in all cases. The improvement is around 0.5 pixel. Figure 4.7 presents results for three image pairs with the initialization only and the

Contour	Distance	MPI-Sintel	Kitti	Middlebury	Time
SED [Dollár and Zitnick, 2013]	**Geodesic (approx.)**	3.686	3.334	**0.380**	16.4s
SED [Dollár and Zitnick, 2013]	Geodesic (exact)	**3.677**	**3.216**	0.393	204s
-	Euclidean	4.617	3.663	0.442	40s
SED [Dollár and Zitnick, 2013]	mixed	3.975	3.510	0.399	300s
gPb [Arbelaez et al., 2011]	Geodesic (approx.)	4.161	3.437	0.430	26s
Canny [Canny, 1986]	Geodesic (approx.)	4.551	3.308	0.488	16.4s
$\|\nabla_2 \mathcal{I}\|_2$	Geodesic (approx.)	4.061	3.399	0.388	16.4s
GT boundaries	Geodesic (approx.)	3.588			

Table 4.2 – Comparison of the EPE of EpicFlow (with DM and LA) for different distances and different contour extractors. The time (right column) is reported for a MPI-Sintel image pair.

final result of EpicFlow (row three and four). While the flow images look similar overall, the minimization allows to further smooth and refine the flow, explaining the gain in performance. Yet, it preserves discontinuities and small details, such as the legs in the right column. In the following, results are reported for EpicFlow, *i.e.*, after the variational minimization step.

Edge-aware versus Euclidean distances. We now study the impact of different distances. First, we examine the effect of approximating the geodesic distance (Section 4.2.4). Table 4.2 shows that our approximation has a negligible impact when compared to the exact geodesic distance. Note that the exact version performs distance computation as well as local estimation per pixel and is, thus, an order of magnitude slower to compute, see last column of Table 4.2.

Next, we compare the geodesic distance and Euclidean distances. Table 4.2 shows that using a Euclidean distance leads to a significant drop in performance, in particular for the MPI-Sintel dataset, the drop is 1 pixel. This confirms the importance of our edge-preserving distance. Note that the result with the Euclidean distance is reported with an exact version, *i.e.*, the interpolation is computed pixelwise.

We also compare to a mixed approach, in which the neighbor list $\mathcal{N}_K$ is constructed using the Euclidean distance, but weights $k_{\tilde{D}}(\boldsymbol{p}_m, \boldsymbol{p})$ are set according to the approximate geodesic distance. Table 4.2 shows that this leads to a drop of performance by around 0.3 pixels for MPI-Sintel and Kitti. Figure 4.6 illustrates the reason: none of the Euclidean neighbor matches (yellow) belong to the region corresponding to the selected pixel (red), but all of geodesic neighbor matches (blue) belong to it. This demonstrates the importance of using an edge-preserving geodesic distance throughout the whole pipeline, in contrast to Leordeanu et al. [2013] who interpolate

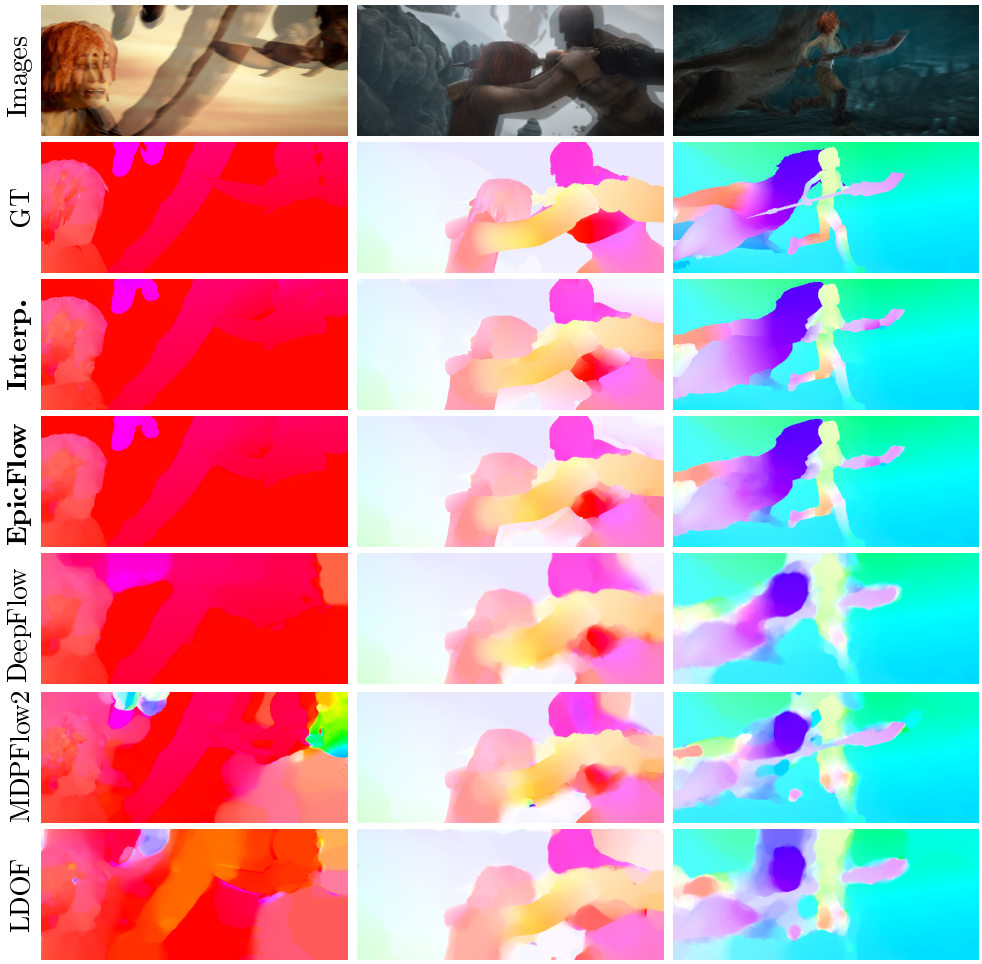


Figure 4.7 – Each column shows from top to bottom: mean of two consecutive images, ground-truth flow (GT), result of sparse-to-dense interpolation (Interp.), full method (EpicFlow), and 3 state-of-the-art methods: DeepFlow [Weinzaepfel et al., 2013], MDPFlow2 [Xu et al., 2012] and LDOF [Brox and Malik, 2011]. EpicFlow better respects motion boundaries, is able to capture small parts like the limbs of the character (right column) and successfully estimates the flow in occluded areas (right part of left column).

matches found in a Euclidean neighborhood.

Impact of contour detector. We also evaluate the impact of the contour detector in Table 4.2, *i.e.*, the SED detector [Dollár and Zitnick, 2013] is replaced by the Berkeley gPb detector [Arbelaez et al., 2011] or the Canny edge detector [Canny, 1986]. Using gPb leads to a small drop in performance (around 0.1 pixel on Kitti and 0.5 on MPI-Sintel) and significantly increases the computation time. Canny edges perform similar to the Euclidean distance. This can be explained by the insufficient quality of the Canny contours. Using the norm of image's gradient improves slightly over gPb. We found that this is due to the presence of holes when estimating contours with gPb. Finally, we perform experiments using ground-truth motion boundaries, computed from the norm of ground-truth flow gradient, and obtain an improvement of 0.1 on MPI-Sintel (0.2 before the variational part). The ground-truth flow is not dense enough on Middlebury and Kitti datasets to estimate GT boundaries.

4.4.3 EpicFlow versus coarse-to-fine scheme

To show the benefit of our approach, we have carried out a comparison with a coarse-to-fine scheme. Our implementation of the variational approach is the same as in Section 4.3, with a coarse-to-fine scheme and DeepMatching integrated in the energy through a penalization of the difference between flow and matches [Brox and Malik, 2011, Weinzaepfel et al., 2013]. Table 4.3 compares EpicFlow to the variational approach with coarse-to-fine scheme, using exactly the same matches as input. EpicFlow performs better and is also faster. The gain is around 0.4 pixel on MPI-Sintel and over 1 pixel on Kitti. The important gain on Kitti might be explained by the affine model used for interpolation, which fits well the piecewise planar structure of the scene. On Middlebury, the variational approach achieves slightly better results, as this dataset does not contain large displacements.

Figure 4.7 shows a comparison to three state-of-the-art methods, all built upon a coarse-to-fine scheme. Note how motion boundaries are preserved by EpicFlow. Even small details, like the limbs in the right column, are captured. Similarly, in the case of occluded areas, EpicFlow benefits from the geodesic distance to produce a correct estimation, see the right part of the left example.

Sensitivity to the matching quality. In order to get a better understanding of why EpicFlow performs better than a coarse-to-fine scheme,

Flow method	MPI-Sintel	Kitti	Middlebury	Time
DM+coarse-to-fine	4.095	4.422	**0.321**	25s
DM+EpicFlow	**3.686**	**3.334**	0.380	16.4s

Table 4.3 – Comparison of EPE for EpicFlow (with DM + LA) and a coarse-to-fine scheme (with DM).

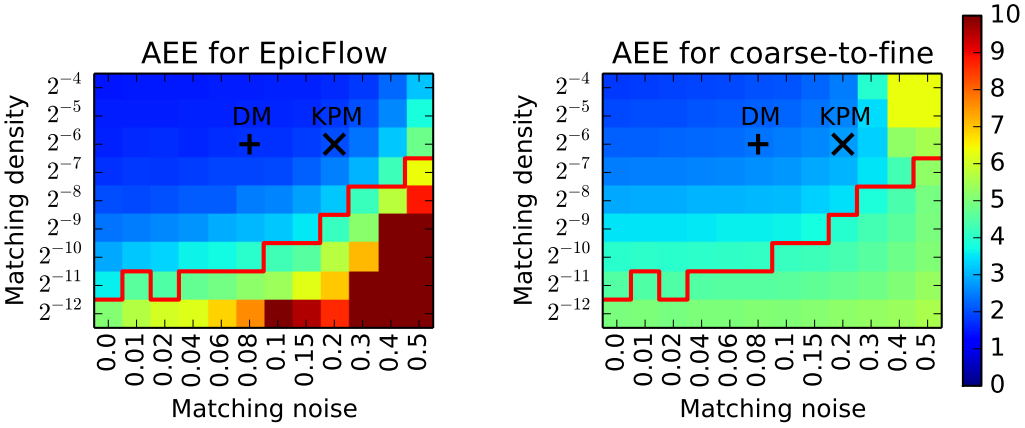


Figure 4.8 – Comparison of EPE between EpicFlow (left) and a coarse-to-fine scheme (right) for various synthetic input matches with different densities and error levels. For positions above the red line, EpicFlow performs better.

we have evaluated and compared their performances for different densities and error rates of the input matches. To that aim, we generated synthetic matches by taking the ground-truth flow, removing points in the occluded areas, subsampling to obtain the desired density and corrupting the matches to the desired percentage of incorrect matches. For each set of matches with a given density and quality, we have carefully determined the parameters of EpicFlow and the coarse-to-fine method on the MPI-Sintel training subset, and then evaluated them on the remaining training images.

Results in term of EPE are given in Figure 4.8, where density is represented vertically as the ratio of #matches / #non-occluded pixels and matching error is represented horizontally as the ratio of #false matches / #matches. We can observe that EpicFlow yields better results provided that the matching is sufficiently dense for a given error rate. For low-density or strongly corrupted matches, EpicFlow yields unsatisfactory performance (Figure 4.8 left), while the coarse-to-fine method remains relatively robust (Figure 4.8 right). This shows that our interpolation-based heuristic for initializing the flow takes better advantage of the input matches than a

Method	EPE	EPE-occ	s0-10	s10-40	s40+	Time
FlowFields [Bailer et al., 2015]	**4.851**	31.799	1.157	3.739	**33.890**	23s
DiscreteFlow [Menze et al., 2015]	6.077	**31.685**	**1.074**	3.832	36.339	180s
EpicFlow	6.285	32.564	1.135	**3.727**	38.021	**16.4s**
TF+OFM [Kennedy and Taylor, 2015]	6.727	33.929	1.512	3.765	39.761	∼400s
DeepFlow [Revaud et al., 2016]	6.928	38.166	1.182	3.859	42.854	25s
S2D-Matching [Leordeanu et al., 2013]	7.872	40.093	1.172	4.695	48.782	∼2000s
Classic+NLP [Sun et al., 2014b]	8.291	40.925	1.208	5.090	51.162	∼800s
MDP-Flow2 [Xu et al., 2012]	8.445	43.430	1.420	5.449	50.507	709s
NLTGV-SC [Ranftl et al., 2014]	8.746	42.242	1.587	4.780	53.860	
LDOF [Brox and Malik, 2011]	9.116	42.344	1.485	4.839	57.296	30s

Table 4.4 – Results on MPI-Sintel test set (final version). EPE-occ is the EPE on occluded areas. s0-10 is the EPE for pixels whose motions is between 0 and 10 px and similarly for s10-40 and s40+. Methods in italic has been published after EpicFlow.

coarse-to-fine schemes for sufficiently dense matches and is able to recover from matching failures. We have indicated the position of DeepMatching and KPM in terms of density and quality on the plots: they lie inside the area in which EpicFlow outperforms a coarse-to-fine scheme.

4.4.4 Comparison with the state of the art

Results on MPI-Sintel test set are given in Table 4.4. Parameters are optimized on the MPI-Sintel training set. EpicFlow was outperforming the state of the art with at publication time by a gap of 0.5 pixel in EPE compared to the second best performing method, TF+OFM [Kennedy and Taylor, 2015], and 0.7 pixel compared to the third one, DeepFlow [Revaud et al., 2016] presented in Chapter 3. In particular, we improve for both EPE on occluded areas and EPE over all pixels and for all displacement ranges. In addition, our approach is significantly faster than most of the methods, *e.g.* an order of magnitude faster than the second best. Interestingly, more recent approaches such as FlowFields [Bailer et al., 2015] or DiscreteFlow [Menze et al., 2015] are built upon EpicFlow but change the input matches, showing the effectiveness of our approach.

Table 4.5 reports the results on the Kitti test set for methods that do not use epipolar geometry or stereo vision. Parameters are optimized on the Kitti training set. We can see that EpicFlow perform on part with the state of the art best in terms of EPE on non-occluded areas or percentage of erroneous pixels. When comparing the methods on both Kitti and MPI-Sintel, we outperform TF+OFM [Kennedy and Taylor, 2015] and DeepFlow [Weinzaepfel et al., 2013] (second and third on MPI-Sintel at

Method	EPE-noc	EPE	Out-Noc 3	Out-All 3	Time
DiscreteFlow [Menze et al., 2015]	**1.3**	3.6	**5.77%**	16.63%	180s
FlowFields [Bailer et al., 2015]	1.4	3.5	6.23%	14.01%	23s
DeepFlow [Revaud et al., 2016]	1.4	5.3	6.61%	17.35%	22s
EpicFlow	1.5	3.8	7.88%	17.08%	16s
BTF-ILLUM [Demetz et al., 2014]	1.5	**2.8**	6.52%	**11.03%**	80s
TGV2ADCSIFT [Braux-Zin et al., 2013]	1.5	4.5	6.20%	15.15%	12s (GPU)
NLTGV-SC [Ranftl et al., 2014]	1.6	3.8	5.93%	11.96%	16s (GPU)
Data-Flow [Vogel et al., 2013b]	1.9	5.5	7.11%	14.57%	180s
TF+OFM [Kennedy and Taylor, 2015]	2.0	5.0	10.22%	18.46%	350s

Table 4.5 – Results on Kitti test set. EPE-noc is the EPE over non-occluded areas. Out-Noc 3 (resp. Out-all 3) refers to the percentage of pixels where flow estimation has an error above 3 pixels in non-occluded areas (resp. all pixels). Methods in italic has been published after EpicFlow.

publication time) on the Kitti dataset, in particular for occluded areas. We perform on par with NLTGV-SC [Ranftl et al., 2014] and DeepFlow Revaud et al. [2016] on Kitti that we outperform by 2.5 and 0.7 pixels respectively on MPI-Sintel. More recent approaches [Bailer et al., 2015, Menze et al., 2015] based on EpicFlow slightly improves our performance.

On the Middlebury test set, we obtain an EPE below 0.4 pixel. This is competitive with the state of the art. In this dataset, there are no large displacements, and consequently, the benefits of a matching-based approach are limited. Note that we have slightly increased the number of fixed point iterations to 25 in the variational method for this dataset (still using one level) in order to get an additional smoothing effect. This leads to a gain of 0.1 pixels (measured on the Middlebury training set when setting the parameters on MPI-Sintel training set).

Timings. While most methods often require several minutes to run on a single image pair, ours runs in 16.4 seconds for a MPI-Sintel image pair (1024×436 pixels) on one CPU-core at 3.6Ghz. In detail, computing DeepMatching takes 15s, extracting SED edges 0.15s, dense interpolation 0.25s, and variational minimization 1s. We can observe that 91% of the time is spent on matching. Note that by using the GPU version of DeepMatching, our runtime can significantly decrease.

Failure cases. EpicFlow can be incorrect due to errors in the sparse matches or errors in the contour extraction. Figure 4.9 (left column) shows an example where matches are missing on thin elements (spear and horns of the dragon). Thus, the optical flow takes the value of the surrounding region for these elements. An example for incorrect contour extraction is

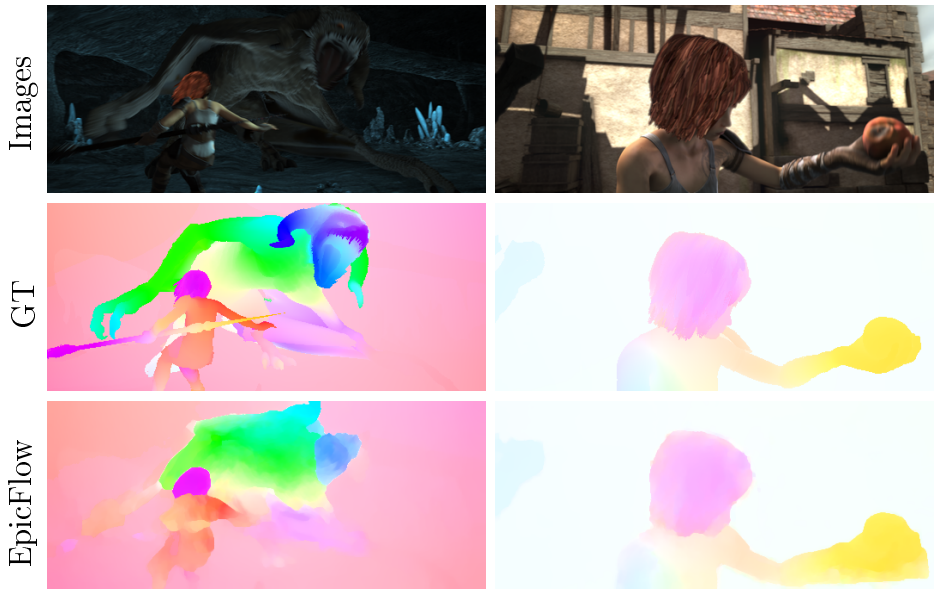


Figure 4.9 – Failure cases of EpicFlow due to missing matches on spear and horns of the dragon (left column) and missing contours on the arm (right column).

presented in Figure 4.9 (right column). The contour of the character's left arm is poorly detected. As a result, the motion of the arm spreads into the background.

4.5 Conclusion

This chapter introduced EpicFlow, a novel optical flow estimation method. EpicFlow computes a dense correspondence field by performing a sparse-to-dense interpolation from an initial sparse set of matches, leveraging contour cues using an edge-aware geodesic distance. The resulting dense correspondence field is fed as an initial optical flow estimate to a one-level variational energy minimization.

Both the sparse set of input matches and the contour estimates are key to our approach. The approach builds upon the assumption that contours often coincide with motion discontinuities. The next chapter focuses on the problem of detecting the motion boundaries.

Chapter 5

Learning to Detect Motion Boundaries

Contents

5.1 Introduction

Optical flow can be simply described as a field that consists of large regions with smooth variations, divided by boundaries with abruptly changes. Yet, energy minimization frameworks assume that the flow is continuous. Consequently, while smooth variations of optical flow are estimated well

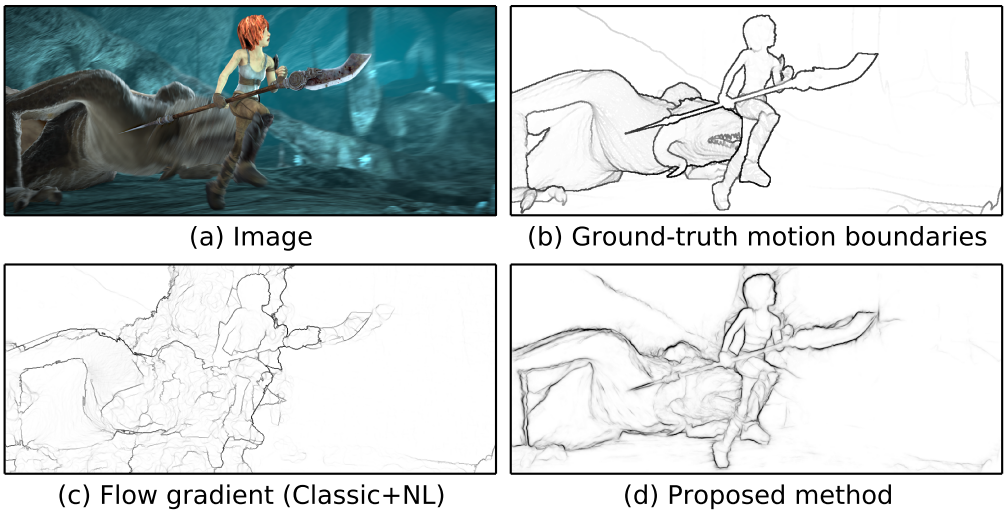


Figure 5.1 – For the image in (a), we show in (b) its ground-truth motion boundaries, in (c) motion boundaries computed as gradient of the Classic+NL [Sun et al., 2014b] flow, and in (d) our proposed motion boundary detection. Despite using the Classic+NL flow as one of our cues, our method is able to detect motion boundaries even at places where the flow estimation failed, such as on the spear or the character's arm.

most of the time, capturing the sharp discontinuities remains more challenging. In this chapter, we focus on the prediction of such motion boundaries, see Figure 5.1. To prevent any ambiguities, we define this notion precisely as follows: *motion boundaries (MB) are the discontinuities of the ground-truth optical flow between 2 frames.*

Several approaches have been proposed to predict motion boundaries [Birchfield, 1999, Black and Fleet, 2000, Middendorf and Nagel, 2001, Spoerri, 1991]. In particular, motion boundaries have recently been computed using the norm of the gradient of the optical flow [Papazoglou and Ferrari, 2013, Wang et al., 2013]. However, even using state-of-the-art optical flow estimation [Sun et al., 2014b] as input, such an approach can result in disappointing results, see Figure 5.1.

Instead, we choose a learning-based approach for the motion boundary prediction problem. This requires a high volume of training data. Thankfully, the MPI-Sintel [Butler et al., 2012] dataset, composed of animated movies generated using computer graphics, is now available and contains ground-truth optical flow. Thus, motion boundaries can be directly computed from the ground-truth flow. The dataset is large (more than 1000 high resolution training images), and paves the way to the training of so-

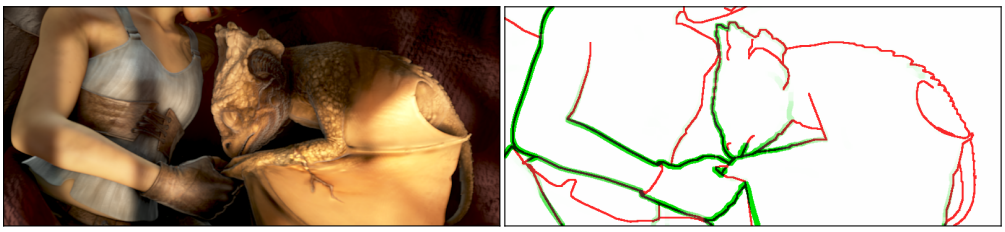


Figure 5.2 – Example of depth discontinuities (in red) and motion boundaries (in green, extracted from the ground-truth flow). Notice how *flow* discontinuities differ from *depth* discontinuities. In addition, the presence of non-rigid objects causes most motion boundaries to form non-closed contours.

phisticated learning algorithms for this task. We choose random forests as the learning algorithm, as they are are both flexible and stable.

In this chapter, we present the following contributions:

- We propose a learning-based motion boundary prediction approach, using structured random forests [Dollár and Zitnick, 2013] in combination with image and optical flow cues, which accurately estimate motion boundaries.
- We show in experiments that our approach is robust to failure cases of the input optical flow.
- We introduce a new dataset, called the YouTube Motion Boundaries (**YMB**) dataset, that comprises 60 real-world videos downloaded from YouTube, for which we provide annotations of the motion boundaries.

Closest references. Most methods for estimating motion boundaries are based on optical flow [Papazoglou and Ferrari, 2013, Wang et al., 2013]. The early work of Spoerri [1991] shows that local flow histograms have bimodal distributions at motion boundaries. Using statistical tests on histograms and structural saliency based postprocessing, this work develops a method to recover and segment motion boundaries in synthetic footage. Similar considerations are used later by Fleet et al. [2000] to propose a low-level motion boundary detector. This detector measures the squared error in fitting a local linear parameterized motion model: the fitting error is larger at motion boundaries. This model is in turn improved by Black and Fleet [2000] by casting the motion boundaries detection problem into a probabilistic framework. In their work, local pixel patches are either modeled with a translational motion or as a motion discontinuity. In the latter case, different configurations of depth ordering are incorporated in the model. This aspect (depth ordering at discontinuities) is also leveraged

by Liu et al. [2006] to estimate optical flow in textureless images. They propose a bottom-up approach to track and group hypothetical motion edge fragments. However, their approach heavily depends on the preliminary detection of edge fragments, and is not applicable to realistic videos with textures. Furthermore, none of these approaches are based on learning the relation between local features and motion boundaries.

Closely related to estimating motion boundaries is the task of segmenting a video frame into different regions with coherent motion, referred to as layers [Darrell and Pentland, 1995, Wang and Adelson, 1994]. Recently, several works have considered the joint estimation of motion layers and optical flow [Brox et al., 2006, Sun et al., 2013, Unger et al., 2012]. However, the joint task is challenging, and these methods depend on a complex minimization of non-convex energy functions. As a result, the estimation is unreliable for difficult, yet common cases, such as videos with fast motion, large displacements or compression artifacts. More generally, motion layer segmentation can be ill-defined, as there exist cases where motion boundaries form non-closed regions, see Figure 5.2.

The related task of occlusion boundary detection has recently received some attention [Hoiem et al., 2011, Sundberg et al., 2011, Humayun et al., 2011]. Occlusion boundaries refer to depth discontinuities. They can correspond to motion boundaries, as they can create differences in flow when the camera or the objects are moving. However, in many cases two regions of different depth can have the same flow, see Figure 5.2 where many of the depth discontinuities actually do not correspond to motion boundaries. Most approaches for occlusion boundary detection [Sundberg et al., 2011, Hoiem et al., 2011, Stein and Hebert, 2009] rely on an oversegmentation of the image, using for instance Arbelaez et al. [2011], followed by a merging procedure. Like our approach, they all use temporal information as a cue, *e.g.* as the difference between consecutive images [Sundberg et al., 2011]. Nevertheless, the final result highly depends on the optical flow accuracy, while our method is robust to failures in the optical flow.

Our method is also related to recent works on edge and occlusion detection cast into a learning framework [Dollár and Zitnick, 2013, Humayun et al., 2011]. These approaches rely on a random forest classifier applied to features extracted in a local neighborhood. The approach of Humayun et al. [2011] for occlusion detection takes as input optical flow estimated with four different algorithms and learns pixel-wise random forests. In contrast, our method leverages information at the patch level and is robust to failures in the optical flow by using an estimated flow error. Dollár and Zitnick [2013] use structured random forests for edge detection, which is shown to outperform the state of the art. We build on their approach and

show how to extend it to motion boundary detection.

Outline. This chapter is organized as follows. We first present our approach for learning motion boundaries in Section 5.2. We then introduce the datasets used in our experiments, including our YMB dataset, and the evaluation protocol in Section 5.3. Finally, Section 5.4 presents the experimental results. The dataset and code are available online at http://lear.inrialpes.fr/research/motionboundaries.

5.2 Learning motion boundary detection

In this section, we first present the structured random forests approach. We then detail the set of cues used for motion boundary detection.

5.2.1 Structured Random Forests

We propose to cast the prediction of local motion boundary masks as a learning task using structured random forests. Motion boundaries in a local patch often have similar patterns, *e.g.* straight lines, parallel lines or T-junctions. The structured random forest framework leverages this property by predicting boundaries at the patch level. In practice, several trees are learned independently with randomization on the feature set, leading to a forest of decision trees. Each tree takes as input a patch and predicts a structured output, here a boundary patch. Given an input image, the predictions of each tree (Figure 5.3) for each (overlapping) local patch are averaged in order to yield a final soft boundary response map. Structured random forests have a good performance and are extremely fast to evaluate.

We now describe the learning model in more detail. Here, a decision tree $f_t(x)$ is a structured classifier that takes an $N \times N$ input patch with K channels, vectorized as $x \in \mathbb{R}^{KN^2}$, and returns a corresponding binary edge map $y \in \mathbb{B}^{N^2}$. Internally, each tree f_t has a binary structure, *i.e.*, each node is either a leaf or has two children nodes. During inference, a binary split function $h(x, \theta_j) \in \{0, 1\}$ associated to each node j is evaluated to decide whether the sample x descends the left or right branch of the tree until a leaf is reached. The output y associated to the leaf at training time is then returned. The whole process is illustrated in Figure 5.3.

The split functions $h(x, \theta)$ considered in this work are efficient decision stumps of two forms: (i) a thresholding operation on a single component of x. In this case, $\theta = (k, \tau)$ and $h_1(x, \theta) = [x(k) < \tau]$, where $[.]$ denotes the indicator function; (ii) a comparison between two components of x.

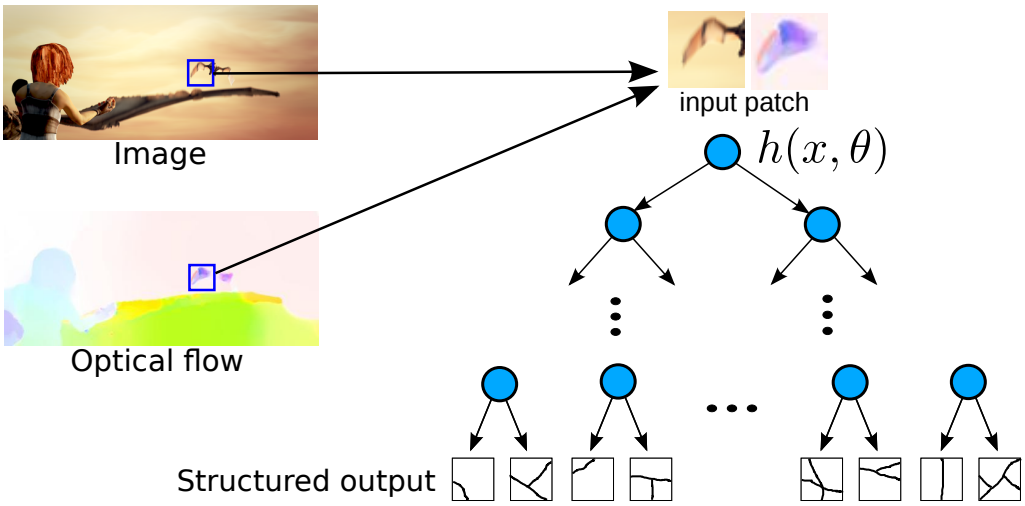


Figure 5.3 – Illustration of the prediction process with our structured decision tree. Given an input patch from the left image (represented here by image and flow channels), we predict a binary boundary mask, *i.e.*, a leaf of the tree. Predicted masks are averaged across all trees and all overlapping patches to yield the final soft-response boundary map.

Thus, $\theta = (k_1, k_2, \tau)$ and $h_2(x, \theta) = [x(k_1) - x(k_2) < \tau]$. We choose the same training algorithm for learning the random forest model as Dollár and Zitnick [2013], using the publicly available code [1]. The success of our approach lies in the choice and design of the features, which we now detail.

5.2.2 Spatial and Temporal Cues

We consider here static appearance features and temporal features to predict motion boundaries. We use the index t to denote the frame for which we predict the motion boundaries ($t+1$ being the next frame).

Color *(13 channels).* We use the three RGB channels in addition to 10 gradient maps, computed in the luminance channel from the Lab color space. We compute the norm of the gradient and oriented gradient maps in 4 directions, both at coarse and fine scales, resulting in $(1+4) \times 2$ channels.

Optical flow *(7 channels).* We also use the optical flow $\boldsymbol{w}_{t,t+1}$ between frame t and $t+1$. Let u and v be the components of $\boldsymbol{w}_{t,t+1}$. In addition to u and v channels, we use an unoriented gradient map computed

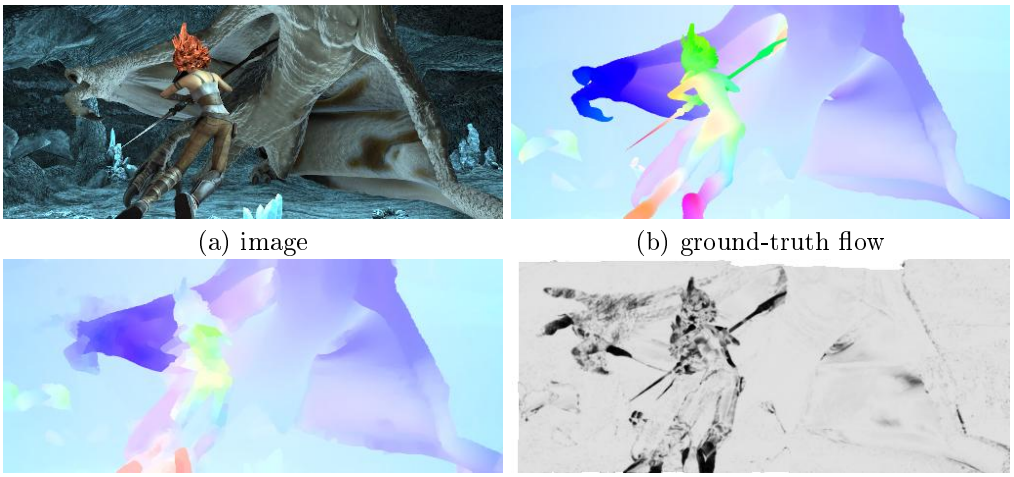
1. https://github.com/pdollar/edges

as $\sqrt{\|\nabla u\|^2 + \|\nabla v\|^2}$, and oriented gradient maps (again at 4 orientations) where the gradient orientation is obtained by averaging the orientations of ∇u and ∇v, weighted by their magnitudes. Contrary to the RGB case, we compute these 5 gradient maps at a coarse scale only. We found that adding the fine scale does not improve the results, probably due to the blur in optical flow estimation. To compute the optical flow, we experiment with different state-of-the-art algorithms and compare their performance in Section 5.4.

Image warping errors *(2 channels).* Optical flow estimation can often be partially incorrect. For instance, in Figure 5.4(c), some object motions are missing (spear) and some others are incorrect (feet). To handle these errors, we propose to add channels indicating where the flow estimation is likely to be wrong, see Figure 5.4(d). To this end, we measure how much the color and gradient constancy assumptions [Brox et al., 2004, Vogel et al., 2013b] are violated. We compute the image warping error, which is defined at a pixel $\boldsymbol{p}$ as $E_D(\boldsymbol{p}) = \|D_t(\boldsymbol{p}) - D_{t+1}(\boldsymbol{p} + \boldsymbol{w}_{t,t+1}(\boldsymbol{p}))\|_2$, where D is an image representation dependent on which constraint (color or gradient) is considered. For the color case, D corresponds to the *Lab* color-space, in which Euclidean distance is closer to perceived color distances. For the gradient case, D is a pixel-wise histogram of oriented gradients (8 orientations), individually normalized to unit norm. We set the warping error to 0 for pixels falling outside the image boundaries.

Backward flow and error *(9 channels).* There is no reason to consider that the forward flow may provide better information for detecting motion boundaries than the backward flow. Consequently, we add the same 7 channels of optical flow and the 2 channels of image warping errors with the backward flow $\boldsymbol{w}_{t,t-1}$.

Summary. By concatenating all these channels, we obtain a feature representation at the patch level that combines several cues: appearance, motion and confidence in motion. The feature representation includes 31 channels in total. Since the 32×32 patches are subsampled by a factor 2 when fed to the classifiers, the final dimension for an input vector x is $(32/2)^2 \times 31 = 7936$.



(a) image (b) ground-truth flow

(c) estimated flow (Classic+NL) (d) image warping error

Figure 5.4 – Illustration of the image warping error (d) computed for the image in (a) and the estimated flow in (c). The ground-truth flow is shown in (b). Errors in flow estimation clearly appear in (d), *e.g.* for the spear, the dragon's claws or the character's feet.

5.3 Datasets and evaluation protocol

In this section, we first present existing optical flow datasets used to train and evaluate our approach. We then introduce our YouTube Motion Boundaries (YMB) dataset and explain the evaluation protocol.

5.3.1 Optical flow datasets

For training and evaluating our approach, we rely on two state-of-the-art optical flow datasets: Middlebury and MPI-Sintel. Both come with dense ground-truth optical flow, which allows to extract ground-truth motion boundaries. We refer to Section 2.5 for details on these datasets. For MPI-Sintel, we compare results for both 'clean' and 'final' versions in the experiments, see Section 5.4. We train our model using all sequences, except when testing on MPI-Sintel. In this case, we alternatively train on half of the sequences and test on the other half.

Ground-truth motion boundaries from flow. For evaluation, we need to compute binary motion boundaries from ground-truth optical flow. However, the resulting boundaries depend on a threshold applied to the norm of the flow gradient. We, thus, propose to generate, for each image, several

Figure 5.5 – From top to bottom: example images, corresponding ground-truth flow, ground-truth motion boundaries and motion layers used for training.

versions of ground-truth boundaries corresponding to different thresholds. Thresholds are spread regularly on a logarithmic scale. For our experimental evaluation, we have set the lowest threshold to a norm of 0.5 for Middlebury and to 1 for MPI-Sintel. Note that the threshold for Middlebury is lower than for MPI-Sintel, as motions in this dataset are smaller. Examples for ground-truth motion boundaries (extracted at norm 2) are shown in Figure 5.5. We refer to Section 5.3.3 for the evaluation protocol.

5.3.2 The YMB dataset

Existing optical flow benchmarks have several limitations. They are often restricted to synthetic and high quality videos and have limited vari-

Figure 5.6 – Illustration of our YMB dataset. The two top examples comes from YouTube Objects, the three other ones from Sports1M. *Left:* images. *Middle:* human annotations of motion boundaries. *Right:* our predictions.

Videos from	#videos	resolution	#annotated px	max $\|\boldsymbol{w}\|_2$	mean $\|\boldsymbol{w}\|_2$
YouTube Objects [Prest et al., 2012b]	30	225×400	700 (0.7%)	16	5
Sports1M [Karpathy et al., 2014]	30	1280×720	3200 (0.3%)	50	8

Table 5.1 – Some statistics of our YMB dataset (averaged across the videos for each part). $\boldsymbol{w}$ denotes the flow, here estimated with LDOF [Brox and Malik, 2011].

ability. For instance, MPI-Sintel contains 23 synthetic sequences sharing characters, objects and backgrounds.

Therefore, we propose a new dataset, the YouTube Motion Boundaries dataset (YMB), composed of 60 real-world videos sequences, ranging from low to moderate quality, with a great variability of persons, objects and poses. For each sequence, motion boundaries are manually annotated in one frame by three independent annotators. The dataset includes two types of videos: 30 sequences from the YouTube Objects dataset [Prest et al., 2012b], and 30 others from the Sports1M [Karpathy et al., 2014] dataset.

YouTube Objects dataset [Prest et al., 2012b] is a collection of video shots representing 10 object categories, such as train, car or dog. For the sake of diversity, we select 3 shots per category. The annotated frame is the same as the one annotated by Prest et al. [2012b] for object detection.

Another 30 sequences are sampled from the Sports1M [Karpathy et al., 2014] dataset. This dataset comprises 487 classes and is dedicated to action recognition in YouTube videos. We select each video from a different class. The annotated frame is chosen to be challenging for optical flow estimation, see Figure 5.6.

Table 5.1 shows some statistics about image sizes and motions. Videos from YouTube Objects have a lower resolution (225×400) than the ones from Sports1M (1280×720). Both datasets contain large motions, some of them of thin parts, *e.g.* the limbs of the humans. Note that a supplementary challenge of the YMB dataset is due to the high compression level of the videos, which causes many block-like artifacts to appear.

We evaluate the consistency between the annotations. To this end, we compute precision and recall using the protocol described below (Section 5.3.3), using one annotator as ground-truth, another one as estimate, and averaging across all pairs of annotators. We obtain a precision and recall over 91%, showing that the annotations are consistent.

5.3.3 Evaluation protocol

We quantitatively evaluate our motion boundary predictions in term of precision-recall. We use the evaluation code of the BSDS [Martin et al., 2001] [2] edge detection benchmark. Given a binary ground-truth and a soft-response motion boundary prediction, we compute the pixel-wise recall and precision curve (Figure 5.10), where each point of the curve corresponds to a different threshold on the predicted motion boundary strength. For instance, a high threshold will lead to few predicted pixels, *i.e.*, a low recall

2. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

and high precision. For a lower threshold, the recall will be higher but the precision will drop. To avoid issues related to the over/under-assignment of ground-truth and predicted pixels, a non-maxima suppression step is performed on the predicted motion boundary map, and a bipartite graph problem is solved to perform a 1-to-1 assignment between each detected and ground-truth boundary pixel.

Precision-recall curves are finally averaged for all images and all binary versions of the ground-truth (*i.e.*, annotations for the YMB dataset, thresholded maps for the flow benchmarks) to compute mean Average-Precision (mAP). In other words, for optical flow benchmarks, the stronger is a motion boundary, the higher is its impact on the evaluation score.

5.4 Experimental results

In this section, we first give details on how we train the structured random forest. We then evaluate different aspects of the method, in particular the impact of optical flow algorithms and different cues. Furthermore, we compare our approach to various baselines.

5.4.1 Training the forest

Training random forests typically requires a large number of examples in order to learn a model that generalizes well. MPI-Sintel constitutes an excellent choice for training our model, as the dataset is large and comes with reliable ground-truth flow.

Generating motion layers. When training each node, Dollár and Zitnick [2013] map the output structured labels (*i.e.*, edge map of a patch) into a set of discrete labels that group similar structured labels. However, computing similarities between edge maps of patches is not well defined. Consequently, they propose to approximate it by computing a distance based on the ground-truth segmentation of the image. In the same spirit, training our motion boundary detector will require a segmentation, *i.e.*, motion layers, in addition to the ground-truth motion boundary patches. We now describe the method we use to compute motion layers from the ground-truth optical flow.

We employ a hierarchical clustering approach on flow pixels, where each pixel is connected to its 4 neighbors with a connection weight set to the magnitude of the flow difference between them. From this initial graph, we then grow regions using average linkage by iteratively merging regions

Training / Test	Middlebury	MPI-Sintel clean	MPI-Sintel final	YMB
train on clean	91.1	76.3	68.5	72.2
train on final	90.9	74.6	67.6	70.7

Table 5.2 – Comparison of the performance (mAP) of our motion boundary estimation, when training on the clean or the final version of the MPI-Sintel dataset. The flow is estimated with Classic+NL [Sun et al., 2014b].

with the lowest connection weight. For each image, we generate 3 segmentations, with different number of target regions and with randomness in the clustering process. Generating several segmentations helps to deal with the intrinsic ambiguity of motion layers, whose boundaries only partially correspond to motion boundaries, *e.g.* in the case of deformable objects, see Figure 5.2. Examples of the resulting segmentation are shown in Figure 5.5 (bottom).

Random forest parameters. The parameters of the structured random forest are the following. The forest has 8 trees, each with a maximum depth of 64 levels. Trees are trained using a pool of 500k patches containing boundaries and 500k without any boundary. To introduce randomization in the tree structure, each tree is trained from a random subset (25%) of all training patches.

Clean versus final version. The training set of MPI-Sintel comes in two different versions (clean and final). We conduct an experiment in which we train two separate models, one for each set. Their performance on the different datasets is evaluated in Table 5.2 using Classic+NL [Sun et al., 2014b] flow estimation. It turns out that, surprisingly, results are consistently better across all datasets when the model is trained on the clean version of MPI-Sintel – even when detecting motion boundaries on the final version. This might be explained by the fact that training is affected by noise, which is *de facto* absent from the clean set. In contrast, noise is clearly present in the final version, in particular in the form of motion blur that tends to smooth evidence of motion boundaries. This result is confirmed for all the optical flow algorithms evaluated. We, thus, choose the clean version of MPI-Sintel to train our models in the remainder of the experiments.

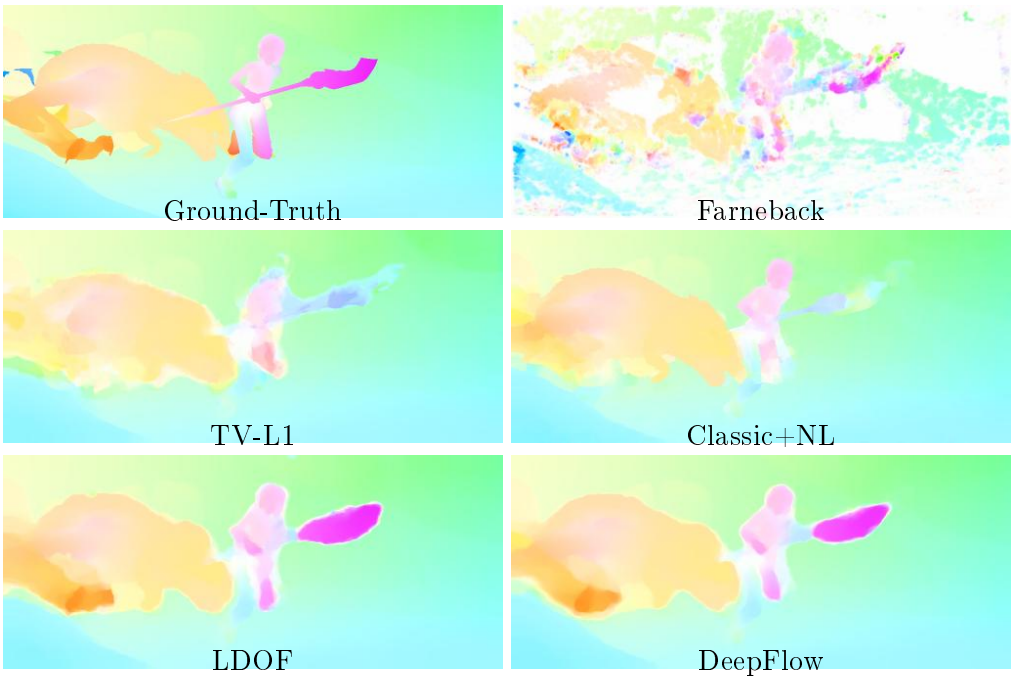


Figure 5.7 – Optical flow estimated with different methods.

	Middlebury		MPI-Sintel clean		MPI-Sintel final		YMB	
	flow MB	ours	flow MB	ours	flow MB	ours	flow MB	ours
Farneback	26.6	66.0	18.4	60.0	19.3	52.2	28.4	59.4
TV-L1	78.4	85.7	44.3	73.0	38.1	62.7	45.4	70.1
Classic+NL	90.5	91.1	68.5	76.3	58.0	68.5	59.4	72.2
LDOF	70.2	86.7	50.7	75.2	42.0	65.6	48.9	70.5
DeepFlow	80.6	89.0	56.9	75.8	46.3	67.7	44.7	68.6

Table 5.3 – Comparison of the performance (mAP) of our approach for different input flows. We also compare to a baseline of motion boundaries directly computed from the flow (flow MB).

5.4.2 Impact of the optical flow algorithm

Our approach relies on several temporal cues, see Section 5.2.2. These cues directly depend on the algorithm used to estimate the optical flow. We compare five different algorithms: Farneback [Farnebäck, 2003], TV-L1 [Zach et al., 2007], Classic+NL [Sun et al., 2014b], LDOF [Brox and Malik, 2011] and DeepFlow [Revaud et al., 2016], see Figure 5.7. We can observe that Farneback's approach results in a noisy estimation and is unreliable in untextured regions. The reason is that this approach is local and does not incorporate a global regularization. The four other approaches minimize a global energy using a coarse-to-fine scheme. TV-L1 uses the dual space for minimizing this energy. A fixed point iterations allows the three remaining approaches to obtain the linear system of equations derived from the energy. They, thus, produce more accurate flow estimations, see Figure 5.7. Classic+NL includes an additional non-local smoothness term that enhances the sharpness of motion boundaries. For instance, the contour of the character is better respected than with the other methods. LDOF and DeepFlow integrate a descriptor matching term, allowing to better handle large displacements. This is visible on the spear in Figure 5.7, whose motion is partially captured. DeepFlow improves over LDOF in the matching scheme, making it top-performer on MPI-Sintel [Butler et al., 2012] at publication time. Both DeepFlow and LDOF tend to over-smooth the flow: for instance, the motion of the spear spreads in the background.

For each of these flows, we train a separate model and report mean Average-Precision (mAP) for all datasets in Table 5.3. The performance of our approach is rather independent of the flow algorithm used, with the exception of Farneback's method which results in a significantly worse performance. Classic+NL gives the best performance on all datasets. This can be explained by the sharpness of the flow boundaries thanks to the non-local regularization.

5.4.3 Comparison to a state-of-the-art baseline

In Table 5.3, we compare our method to baseline motion boundaries, extracted as the gradient norm of each flow. Note that the performance of our approach largely outperforms this baseline, for all flow methods and on all datasets. The gap is especially large for the most challenging datasets (*e.g.* +25% in mAP for LDOF on MPI-Sintel and YMB). Note that results on Middlebury and YMB datasets are obtained with the model trained on MPI-Sintel. This demonstrates that our approach for motion boundary estimation performs well on low-resolution YouTube videos despite the fact

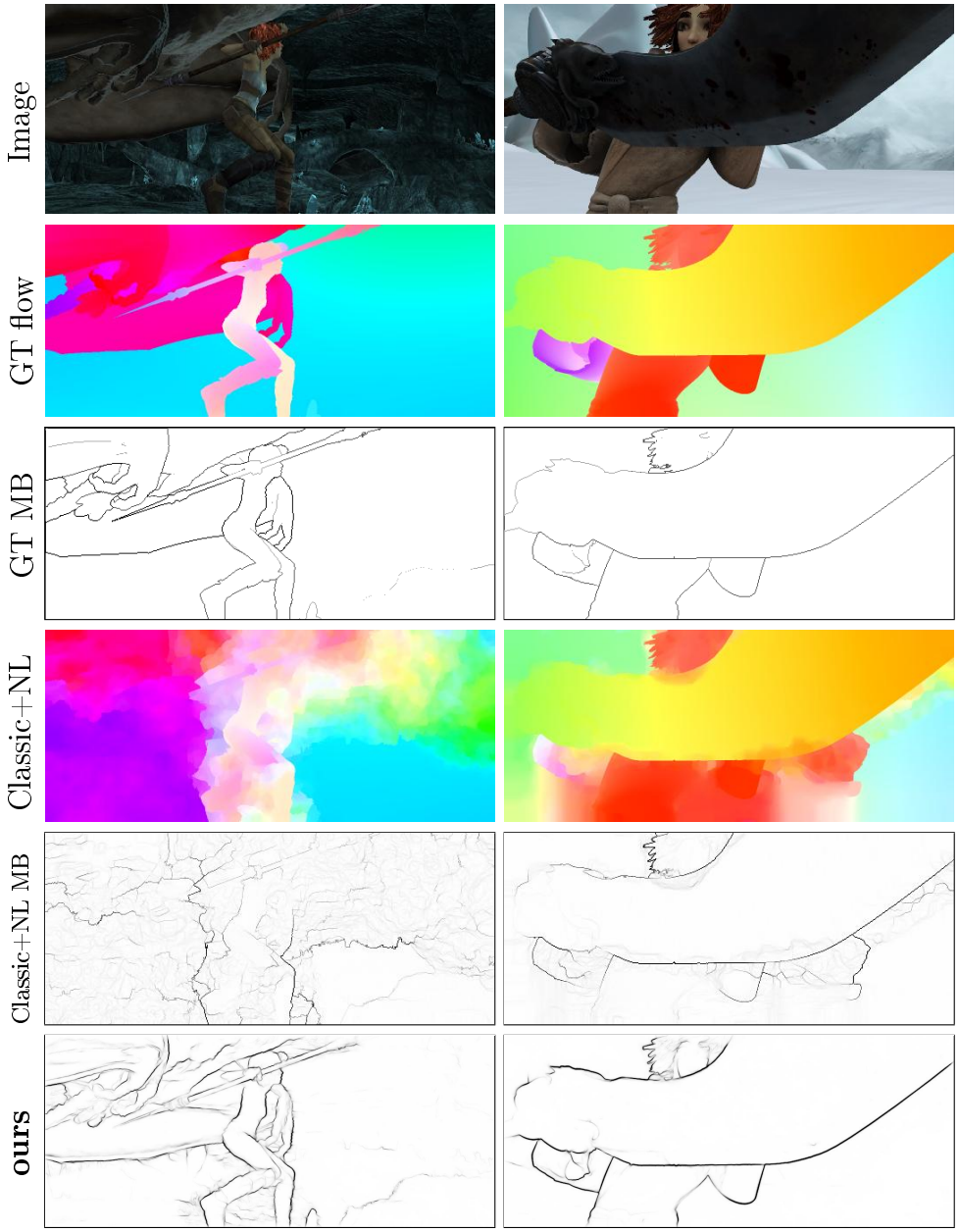


Figure 5.8 – Example results from the MPI-Sintel dataset with, from top to bottom: image, ground-truth flow, ground-truth motion boundaries, flow estimation using Classic+NL [Sun et al., 2014b], norm of the flow gradient (Classic+NL MB), and the motion boundaries estimated by our method (ours).

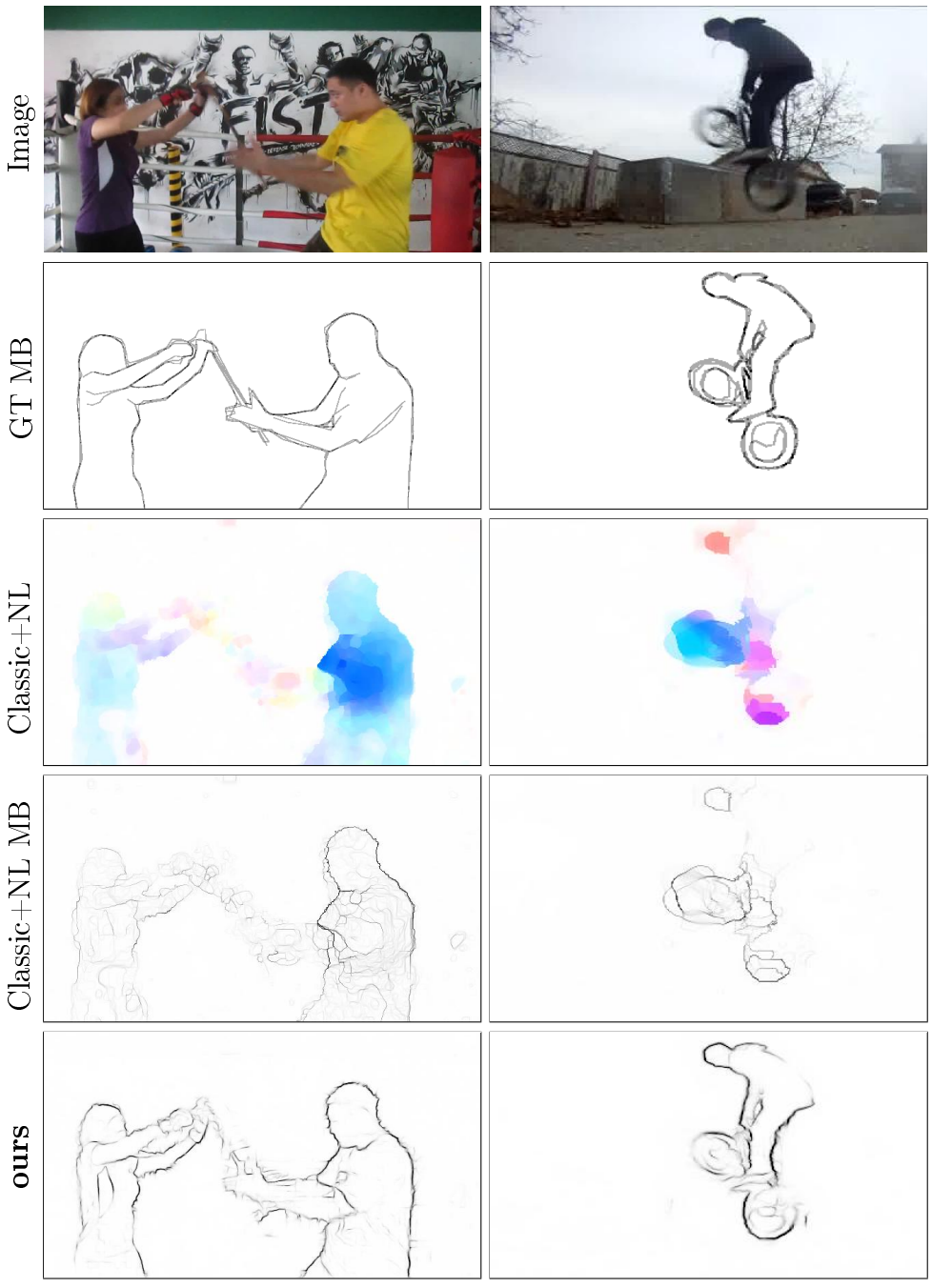


Figure 5.9 – Example results from the YMB dataset with, from top to bottom: images, annotated motion boundaries, flow estimation using Classic+NL [Sun et al., 2014b], norm of the flow gradient (Classic+NL MB), and the motion boundaries estimated by our method (ours).

channels used	Middlebury	MPI-Sintel clean	MPI-Sintel final	YMB
SED [Dollár and Zitnick, 2013]	48.8	32.4	30.1	31.3
RGB only	48.0	41.1	37.5	36.3
+Flow	**91.8**	72.2	66.3	69.6
+Image warping error	91.2	74.2	66.1	70.5
+Backward flow&error	91.1	**76.3**	**68.5**	**72.2**

Table 5.4 – Importance of temporal cues for predicting motion boundaries. We also compare to SED [Dollár and Zitnick, 2013] that uses the same framework, learned on different data.

that it was trained on synthetic high resolution data from MPI-Sintel. In addition, this shows that our method generalizes well to another dataset with different content and does not require specific tuning.

Figure 5.8 provides qualitative comparisons between Classic+NL flow boundaries and our predictions for two images from MPI-Sintel [Butler et al., 2012]. Some object motions, like the character in the left column, are missed in the flow estimation. Likewise, errors due to over-smoothing are visible at the bottom of the right column. They are well recovered by our model, which accurately predicts the motion boundaries. The robustness of our model to incorrect or over-smooth flow estimates is confirmed by the examples from YMB shown in Figure 5.9. The motion of the arm is badly estimated by the flow (left) and the motion of the wheels (right) spreads in the background. In both cases, our model is able to accurately estimate motion boundaries. This resilience can be explained by the integration of appearance and flow confidence cues (Section 5.2.2) in our model, which certainly helps the classifier to recover from errors in the flow estimation, as shown in the next section.

5.4.4 Impact of the temporal cues

We conduct an ablative study to determine the importance of the temporal cues used as feature channels by the classifier. Table 5.4 shows the improvements resulting from adding one cue at a time. In addition, Figure 5.10 shows the precision recall curves for the MPI-Sintel dataset. Performance is reported for Classic+NL, but all flow estimators result in a similar behavior.

First, we notice that using static cues alone already outperforms the SED edge detector [Dollár and Zitnick, 2013], which uses the same RGB

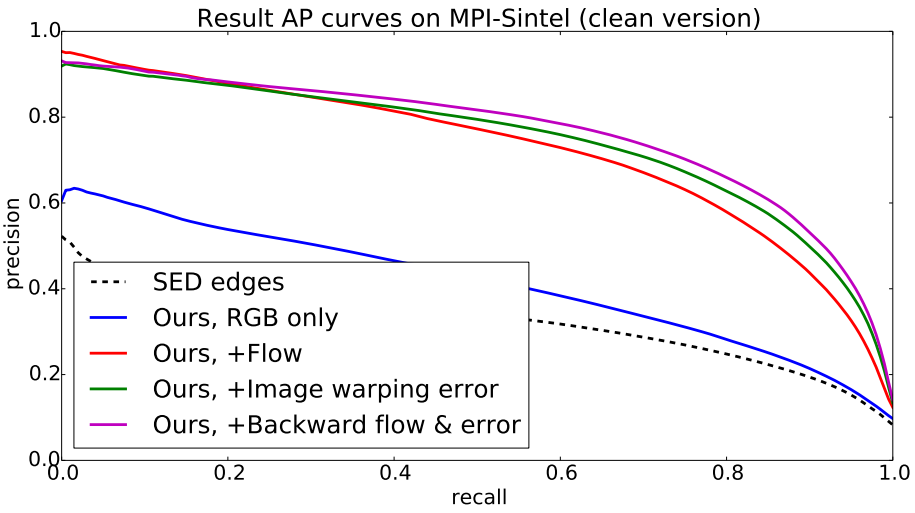


Figure 5.10 – Precision-recall curves when studying the importance of temporal cues on MPI-Sintel dataset, clean version.

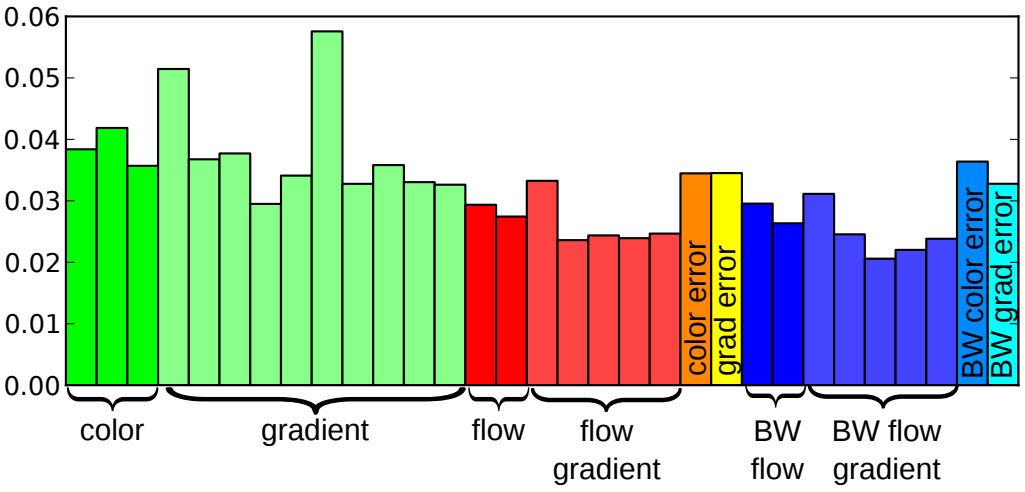


Figure 5.11 – Frequency of each feature channel in the decision stumps of the random forest learned on MPI-Sintel clean. 'BW' refers to backward, 'color error' (resp. 'grad error') denotes the color-based (resp. gradient-based) image warping error. All channels are about equally important.

cues and learning approach, but a different training set. This indicates that, based on appearances cues alone, one is able to 'learn' the location of motion boundaries. After examining the decision tree, we find that, in this case, the classifier learns that a color difference between two objects is likely to yield a motion boundary.

On Middlebury, using only the first two cues (appearance and flow) suffices to accurately predict motion boundaries. The initial flow estimate is already very close to the ground-truth for this relatively easy dataset. On the more challenging datasets (MPI-Sintel and YMB), adding the flow confidence cue (*i.e.*, image warping errors) allows to further gain up to 2% in mean Average-Precision. As shown in Figure 5.4, the error maps indeed accurately indicate errors in the flow estimation. Finally, backward flow cues lead to an additional gain of 2%. We also conduct an analysis of the frequency of usage of each channel in the decision stumps of our learned forest. Figure 5.11 plots the resulting histogram, which confirms that, overall, all channels have approximately the same importance.

5.5 Conclusion

In this chapter, we showed that a learning-based approach using structured random forests is successful for detecting motion boundaries. Thanks to the integration of diverse appearance and temporal cues, our method is resilient to errors in flow estimation. Our approach outputs accurate motion boundaries and largely outperforms flow gradient baseline, in particular on challenging video sequences with large displacements, motion blur and compression artifacts.

The upcoming question concerns its impact on optical flow estimation. While it is not straightforward to integrate it into a coarse-to-fine scheme, we perform an experiment in which the edges in EpicFlow are replaced by the predicted motion boundaries. The gain (in EPE) is limited to 0.01 pixel on the MPI-Sintel training set. First, we notice that even with ground-truth motion boundaries, the performance boost is limited to around 0.1 pixel, see Table 4.2. Second, the matches are more critical than the boundaries in EpicFlow performance. Indeed, in the presence of edges which are not motion boundaries, we find that the texture carried by the edges allows to find correct correspondences. As a consequence, the performance is not harmed at these locations. In summary, the question regarding how the proposed motion boundaries detection can help optical flow estimation remains open.

Part II

Action Localization in Uncontrolled Videos

Chapter 6

Related Work on Action Localization

Contents

In this chapter, we review related work on action localization. We start by a brief overview of recent advances in video classification in Section 6.1. We then review the state of the art in action localization in Section 6.2. Finally, Section 6.3 presents the datasets and the metrics used for evaluating action localization.

6.1 Video classification

In this section, we briefly present the main families of methods for video classification or action recognition. Reviewing all works on video classification is far beyond the scope of this thesis. We refer to [Aggarwal and Ryoo, 2011, Poppe, 2010, Weinland et al., 2011, Herath et al., 2016] for recent surveys. We first propose an overview of methods based on local fea-

tures (Section 6.1.1) and then briefly present approaches leveraging Deep Convolutional Neural Networks (Section 6.1.2).

6.1.1 Local features

Most recent methods [Laptev, 2005, Wang et al., 2013] are based on local features: a video is represented as a collection of descriptors, representing small volumes or sequences of image patches. In contrast to global representation, local features have shown robustness under uncontrolled video settings thanks to the absence of a strict assumption on the global structure of the action. In addition, descriptors are directly computed from pixel values or optical flows, thus avoiding error-prone processing steps such as silhouette extraction, segmentation or long-term tracking. These methods have shown excellent results on a wide variety of video data such as sports broadcasts [Niebles et al., 2010, Rodriguez et al., 2008], movies [Laptev et al., 2008, Marszalek et al., 2009], TV broadcasts [Patron et al., 2010] or consumer videos [Liu et al., 2009, Ikizler-Cinbis et al., 2009].

Spatio-temporal features. Local spatio-temporal features aim at representing a video by detecting and describing small video volumes. Most approaches are extensions of successful methods for images. For instance, for selecting regions that are robust to capturing conditions, Laptev [2005] extends the Harris cornerness criterion [Harris and Stephens, 1988] to videos. In a similar spirit, the Hessian detector proposed by Willems et al. [2008] is an extension of the well-known blob detector [Beaudet, 1978] in images. For descriptors, Kläser et al. [2008] extend the successful Histogram of Oriented Gradient (HOG) [Dalal and Triggs, 2005] to video volumes by quantizing the gradient angles in 3D. It is also common to extract a Histogram of Optical Flow (HOF) [Laptev et al., 2008] or a Motion Boundary Histogram (MBH) [Wang et al., 2013], which is based on the gradient of the optical flow.

Trajectory features. Another way to leverage motion consists in extracting trajectory features, *i.e.*, the temporal evolution of some point coordinates. The temporal dimension is thus treated separately from the spatial axes. Few approaches [Sand and Teller, 2008, Brox and Malik, 2010, Lezama et al., 2011] were based on long-term trajectories. However, tracking points across many frames is expensive and faces different challenges, *e.g.* large displacements or occlusions.

Consequently, most recent works use an aggregation of short-term trajectories (around 15 frames), often referred to as tracklets [Matikainen et al., 2009, Wang et al., 2013]. Such trajectories can be directly computed from the optical flow [Lucas and Kanade, 1981, Farnebäck, 2003] and are less sensitive to drifting thanks to their short lengths. In addition to their spatial evolution, a subvolume centered at the trajectory is often described with its appearance and motion. For instance, Wang et al. [2013] compute a Histogram of Oriented Gradient (HOG), a Histogram of Optical Flow (HOF) and a Motion Boundary Histogram (MBH). In contrast to most approaches, they do not rely on interest points but compute features from trajectories extracted from a dense grid.

More recently, it has been proposed to cancel out the camera motion when extracting trajectories and features. For instance, Uemura et al. [2008] segment the images and use feature matching in order to estimate the camera motion, as well as separate tracks induced by moving objects from those due to camera motion. Wang et al. [2015] improve the dense trajectories features by incorporating a camera motion compensation based on matches from SURF [Bay et al., 2006] and optical flow. In order to avoid using matches from the actors, *i.e.*, motion not induced by the camera, a human detector can be leveraged to remove these correspondences.

Aggregation. For classifying videos, a representation is built by aggregating local features. A typical example is the Bag-of-Words (BoW) approach [Sivic and Zisserman, 2003, Csurka et al., 2004], which was originally designed for representing textual documents as word frequencies. For computer vision, *visual words* are obtained by clustering a huge number of local features, *e.g.* using k-means. The set of visual words is often referred to as dictionary. An image or video is then represented by the frequency of assignment of the local features to each visual word. This approach has been successful in many applications of computer vision, in particular for videos [Wang et al., 2013], at the cost of using non-linear kernels.

Bag-of-Words only counts the number of occurrences when assigning features to the dictionary. Improved aggregation techniques have been proposed over the years. For instance, van Gemert et al. [2010] propose a soft assignment scheme instead of the hard assignment in BoW. Higher-order statistics can be used. VLAD [Jégou et al., 2012] and super vector coding [Zhou et al., 2010] model the mean of the feature points assigned to a visual word. Fisher Vectors [Sánchez et al., 2013] include the second-order statistics (*i.e.*, the variance) and have obtained state-of-the-art performance [Wang et al., 2015] in video classification, using a linear kernel

after proper normalization [Sánchez et al., 2013]. In our work, we use the representation of Wang et al. [2015] based on Fisher Vectors and improved dense trajectories.

These aggregation techniques do not model any geometric relation between feature points. Several extensions incorporating local or global relations between local features have thus been proposed. For instance, Laptev et al. [2008] extend the spatial pyramid [Lazebnik et al., 2006] to videos: the aggregation is computed for several video volumes and then concatenated. In the same spirit, Gaidon et al. [2013] concatenate BoW representations for 3 sub-actions. Another example is to pool the features over supervoxels as proposed by Taralova et al. [2014] or Peng et al. [2014].

6.1.2 Deep learning approaches

Driven by the recent success of Deep Convolutional Neural Networks (CNN) [Krizhevsky et al., 2012, Simonyan and Zisserman, 2015, Szegedy et al., 2015, He et al., 2016] in image classification and in other computer vision tasks such as object detection [Girshick et al., 2014] or segmentation [Chen et al., 2015], extensions to video classification have been proposed. Such approaches can be split in 3 categories. The first one leverages 3D convolutions [Ji et al., 2013, Karpathy et al., 2014, Tran et al., 2015]. Karpathy et al. [2014] compare different architectures in which information over consecutive frames is fused at various levels. Tran et al. [2015] show that using $3 \times 3 \times 3$ filter for all convolution layers performs best in various tasks such as action classification, action similarity labeling, scene classification or object recognition. The second category is built upon recurrent neural network [Donahue et al., 2015]. Visual features are computed at every frame and then Long Short-Term Memory (LSTM) units allow to incorporate temporal modeling in the architecture. In the latter category, images and optical flows are processed in two separate streams [Simonyan and Zisserman, 2014]. For the motion stream, stacking optical flows over several frames boosts performance. Most promising results have been obtained by the two-streams architecture. Results obtained with CNN are on the same order of magnitude as improved dense trajectories [Wang et al., 2015], and these two approaches can be combined efficiently [Simonyan and Zisserman, 2014].

6.2 Action localization

Action detection, called also action localization, refers to the problem of recognizing the actions as well as their extent. In this thesis, we focus on human action localization in space and time. We do not consider actions performed by animals [Xu et al., 2015]. In this section, we review most techniques for temporal and spatio-temporal action localization.

6.2.1 Temporal localization

Initial attempts for temporal action localization are based on a sliding-window scheme and focus on improving the search complexity [Yuan et al., 2009, Duchenne et al., 2009, Gaidon et al., 2013]. The problem is treated as a localized classification: uniformly sampled windows are extracted and the one that obtains the maximum score is considered as the action's location. The search space remains acceptable for temporal localization, which is a 1-dimensional problem. Few improvements have been proposed. For instance, Gaidon et al. [2013] use a more structured representation by dividing actions into 3 sub-actions at the cost of additional annotations for these sub-actions. Niebles et al. [2010] apply the Deformable Part Models (DPM) [Felzenszwalb et al., 2010] to the temporal domain by inferring temporal anchor points and scales for sub-events of each class. To speed-up the temporal localization, Oneata et al. [2014c] proposed an approximately normalized Fisher Vector, allowing to replace the sliding window scheme by a more efficient branch-and-bound search [Lampert et al., 2009].

Several weakly-supervised approaches have also been proposed [Bojanowski et al., 2014, Duchenne et al., 2009, Hoai et al., 2014]. Duchenne et al. [2009] use movie scripts to obtain a coarse localization of the actions. The localization is then refined by leveraging discriminative clustering and used to learn a classifier. In the same spirit, Satkin and Hebert [2010] find discriminative segments in training videos using a max-margin objective function with temporal extents acting as latent variables. Given an ordered list of the actions in each clip, Bojanowski et al. [2014] assign temporal segments to action based on discriminative clustering. The assignment then allows to learn a detector for each action. More recently, Hoai et al. [2014] extend a Multiple Instance SVM to time series while allowing discontinuities in the positive samples.

6.2.2 Spatio-temporal localization

Sliding window. The set of of spatio-temporal tubes is much too large to perform sliding window in space and time. Most of the first attempts for spatio-temporal localization [Laptev and Pérez, 2007, Cao et al., 2010] assume a fixed spatial extent of the action, *i.e.*, output a cuboid. Such assumption is not realistic for uncontrolled videos in which the camera and the actor may move. More recently, DPM has been extended to videos. Tian et al. [2013] replace the HOG features of DPM by its 3D version, namely HOG-3D proposed by Kläser et al. [2008].

Figure-centric model. Lan et al. [2011] extend the Latent Support Vector Machine framework used in DPM by considering the actor location as a latent variable and add a prior to enforce similar locations across time. Such model centered at human location is often referred to as figure-centric model. Prest et al. [2012a] propose to detect humans and objects and then model their interaction. Kläser et al. [2010] use a human detector and build human tracks using KLT features tracks. The human tracks are then classified with HOG-3D descriptors [Kläser et al., 2008]. Other models centered on humans often require more supervision. For instance, the pose can be leveraged to improve action recognition [Jhuang et al., 2013]. For localizing the action, Wang et al. [2014] first use a temporal sliding window and then model the relations between dynamic-poselets. The authors thus require pose annotations in training videos.

Action proposals. Driven by the success of proposals for object detection [Zitnick and Dollár, 2014, Uijlings et al., 2013], several recent methods for action localization are based on action proposals to reduce the search complexity. Jain et al. [2014b] and Oneata et al. [2014a] construct action tubes by hierarchically merging supervoxels based on various features such as color, texture, motion or their size. They then rely on dense trajectories features for tube classification. Similarly, van Gemert et al. [2015] propose to cluster dense trajectories with a similarity measure defined by their descriptors (HOG, HOF and MBH) and use the resulting tubes for action detection. Dense trajectories were also used by Marian Puscas et al. [2015], to link per-frame proposals, namely SelectiveSearch [Uijlings et al., 2013] throughout the video. Tubes are then refined using transductive learning. Action proposals from Yu and Yuan [2015] are based on an actionness measure [Chen et al., 2014], which requires localized training samples.

Recently, Gkioxari and Malik [2015] proposed to use object proposals and object detector based on CNN [Girshick et al., 2014] for action local-

ization. Object proposals from SelectiveSearch [Uijlings et al., 2013] are detected in each frame, scored using features from a two-streams CNN architecture, and linked across the video. Our approaches also rely on a per-frame detector, but we use a tracking-by-detection to obtain tubes, which is more robust for instance in case of multiple actors.

Weakly-supervised action localization Annotating all videos with bounding boxes in every frame of an action is not realistic for large-scale datasets. Weakly-supervised action localization is thus necessary, but has received little attention so far. Siva and Xiang [2011] define cuboids of different time lengths around detected humans, describe them with space-time interest points (STIPs) [Laptev, 2005] and then use Multiple Instance Learning (MIL). Their method can thus only be used for static humans. Mosabbeb et al. [2014] use a subspace segmentation clustering approach applied on groups of trajectories in order to segment videos into parts. Low-rank matrix completion then estimates the contribution of each cluster to the different labels. Hence, the approach detects several disjoint action parts and not one spatio-temporal consistent localization. Ma et al. [2013] first extract a per-frame hierarchical segmentation, which is tracked over the videos. Using a foreground scoring, they obtain a hierarchy of spatio-temporal segments where the upper level corresponds to human body location candidates. More recently, Chen and Corso [2015] propose to generate unsupervised proposals by clustering intentional motion based on dense trajectories. A classifier is then learned using the best proposal of a video as positive sample. It is thus a weakly-supervised method but which assumes only one action per video during training. Furthermore, this method is not robust to nearby motions and assumes significant motion.

Some action classification methods automatically discover discriminative parts to improve the performance, but do not aim at precisely localizing the action. For instance, Shapovalova et al. [2012] extend latent SVM to model pairwise similarities between latent variables, aiming at discovering common parts for a particular action. Boyraz et al. [2014] optimize jointly the classification error with the location of a fixed number of discriminative parts. Lan et al. [2015] leverage a discriminative clustering approach for parsing complex actions into mid-level action elements.

6.3 Datasets and metrics

In this section, we first present the datasets for evaluating human action localization. We then present the standard evaluation metrics. The DALY

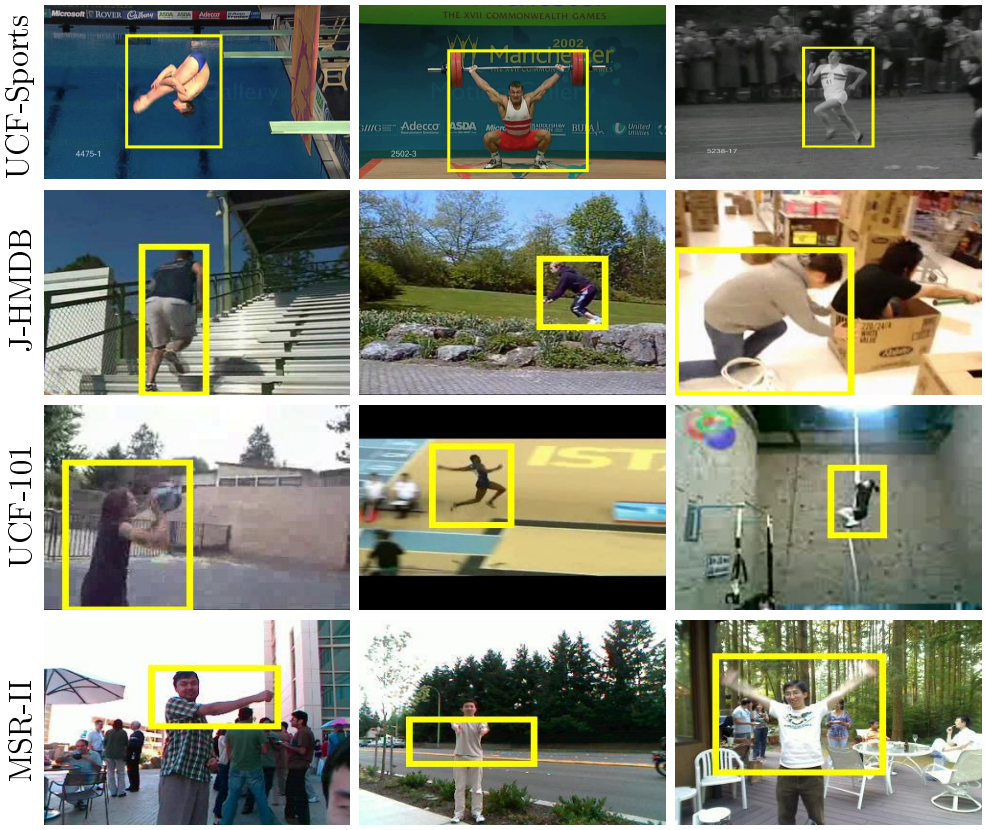


Figure 6.1 – Few frames from the existing datasets.

dataset we introduced in this thesis is presented in Chapter 8.

6.3.1 Datasets

We describe the datasets for evaluating action localization, namely UCF-Sports, J-HMDB, UCF-101 and MSR-II. Figure 6.1 illustrates few frames from these benchmarks.

- The **UCF-Sports** dataset [Rodriguez et al., 2008] consists of 150 sports videos with 10 actions, such as *diving* or *running*. The number of videos is limited, with for instance 4 training and 2 test videos for the class *lifting*. Videos are trimmed to the action and every frame is annotated with a bounding box. For each class, the sequences present similarities in background, camera viewpoint and actors, which is a limitation of this dataset. In our experiments, we use the train/test split defined by Lan et al. [2011].
- The **J-HMDB** dataset [Jhuang et al., 2013] contains 928 short videos with 21 actions, including *stand up*, *run* and *pour*. It is a subset of the HMDB

dataset [Kuehne et al., 2011] for action classification for which additional annotations such as the pose and the silhouette are provided. The videos are trimmed to the action, are very short (1.4 sec on average) and contain only one human most of the time. We use the bounding boxes around the silhouettes as ground-truth. The dataset has 3 train/test splits.

- The **UCF-101** dataset [Soomro et al., 2012] is dedicated to action classification with more than 13000 videos and 101 classes. For a subset of 24 sports labels, the spatio-temporal extent of the actions is annotated. This represents 3207 videos. In contrast to UCF-Sports and J-HMDB, the detection is also temporal but the videos remain short; for half of the classes, the action lasts for more than 80% of the video duration. Figure 6.2 shows a histogram of the action durations in the training set, averaged over all 24 classes. Some of the actions are long, such as 'soccer juggling' or 'ice dancing', whereas others last only few frames, *e.g.* 'tennis swing' or 'basketball dunk'. There are 3 train/test splits.
- The **MSR-II** dataset [Cao et al., 2010] consists of 54 videos that last 51s on average. Actors alternate between performing 3 artificial actions (namely *handwaving*, *handclapping* and *handboxing*) and walking around in different realistic places. Annotations consist in a bounding volume around the moving parts (*e.g.* the hands) in the action. The standard protocol [Cao et al., 2010] consists in using the KTH dataset [Schüldt et al., 2004] for training. The KTH dataset is one of the first benchmark for action recognition. The videos contain a fixed and almost uniform background with few actors performing a set of 6 actions (including the three labels from MSR-II) multiple times. This protocol can not be used for evaluating weakly-supervised approach as the spatial extent of the actions in the training videos corresponds to the full frames.

6.3.2 Metrics

The standard metric consists in computing the mean Average-Precision (mAP) at a given threshold δ. A detection is correct if the IoU (Intersection over Union) with the ground-truth is over a threshold δ. The IoU between tubes is defined as the IoU over the temporal domain, multiplied by the average of the spatial IoU over all overlapping frames. As this is standard in detection, duplicate detections are considered as wrong. The average precision is then computed for each class, with a threshold $\delta = 50\%$ when the localization is limited to the spatial domain (UCF-Sports, J-HMDB) and $\delta = 20\%$ for spatio-temporal detection (UCF-101). The reported results are averaged over the classes and the splits.

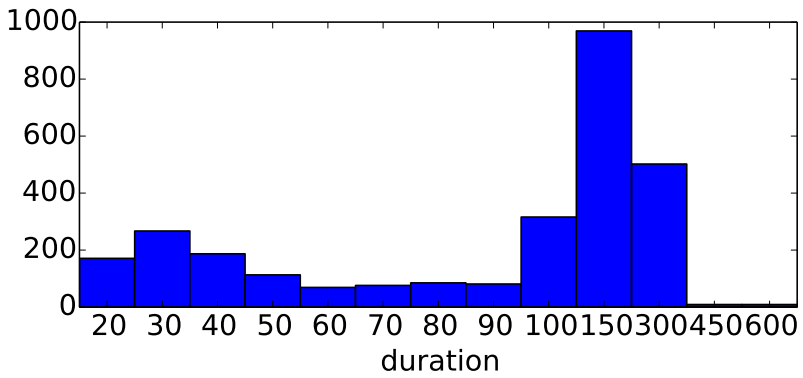


Figure 6.2 – Histogram of action durations (in number of frames) for the 24 classes with spatio-temporal annotations in the UCF-101 dataset (training set).

When comparing to the state of the art on UCF-Sports, we also plot ROC curves and report the Area Under the Curve (AUC) as done by previous work. Note that this metric is impacted by the set of negatives detections and, thus, may not be suited for a detection task [Everingham et al., 2011]. Indeed, if one adds many easy negatives, *i.e.*, negatives that are ranked after all positives, the AUC increases while the mAP remains the same. Some other works also report the mean-IOU, which is the average IoU between the best detection in a test video and the ground-truth tube.

Chapter 7

Action-specific Tracks for Action Localization

Contents

7.1 Introduction

The main challenge in spatio-temporal localization is to accommodate the uncertainty of per-frame spatial localization and the temporal consistency. If the spatial localization performed independently on each frame is too selective and at the same time uncertain, then enforcing the temporal consistency across frames of the localization may fail. In this chapter, we propose to use a set of per-frame region proposals and enforce temporal

consistency based on a tracker, that simultaneously relies on instance-level and class-level detectors.

Our approach starts from frame-level proposals extracted with a high-recall proposal algorithm [Zitnick and Dollár, 2014]. Proposals are scored using CNN descriptors based on appearance and motion information [Gkioxari and Malik, 2015]. To ensure the temporal consistency, we propose to track them with a tracking-by-detection approach combining instance-level and class-level detectors. We then score the tracks with the CNN features as well as spatio-temporal local features that capture the dynamics of an action. At this stage, the tracks are localized in space, but the temporal localization needs to be determined. Temporal localization is performed using a multi-scale sliding-window approach at the track level.

In summary, this chapter introduces an approach for fully-supervised spatio-temporal action localization with state-of-the-art experimental results on UCF-Sports, J-HMDB and UCF-101. Spatio-temporal local features allow to single out more relevant tracks and temporally localize the action at the track level. As local features, we compare a proposed spatio-temporal motion histogram (STMH) descriptor at the track level and the standard improved dense trajectories (IDT) [Wang et al., 2015]. In the preliminary version from ICCV'15 [Weinzaepfel et al., 2015a], only results with STMH were reported.

Closest references. Our proposed method is related to figure-centric models. In particular, Kläser et al. [2010] use a human detector. The detected humans are then tracked across frames using optical flow and the track is classified using HOG-3D [Kläser et al., 2008]. Our approach also relies on tracking, but is more robust to appearance and pose changes by using a tracking-by-detection approach [Hare et al., 2011, Kalal et al., 2012], in combination with a class-specific detector. In addition, we classify the tracks using per-frame CNN features and spatio-temporal features.

Recently, Gkioxari and Malik [2015] proposed to use object proposals for action localization. Object proposals from SelectiveSearch [Uijlings et al., 2013] are detected in each frame, scored using features from a two-streams CNN architecture, and linked across the video. Our approach is more robust since we do not force detections to pass through proposals at every frame. Moreover, we combine the per-frame CNN features with descriptors extracted at a spatio-temporal level to capture the dynamics of the actions.

Outline. This chapter is organized as follows. We present an overview of our approach in Section 7.2 and then give the details in Section 7.3. Finally,

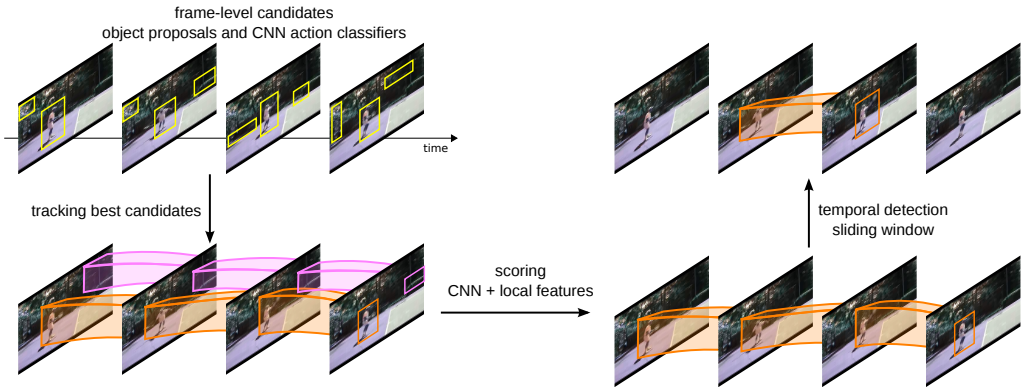


Figure 7.1 – Overview of our action localization approach. We detect frame-level object proposals and score them with CNN action classifiers. The best candidates, in term of scores, are tracked throughout the video. We then score the tracks with CNN and spatio-temporal local features classifiers. Finally, we perform a temporal sliding window for detecting the temporal extent of the action.

Section 7.4 presents experimental results.

7.2 Overview of the approach

Our approach for spatio-temporal action localization consists of four stages, see Figure 7.1. We now briefly present them and then provide a detailed description in Section 7.3.

Extracting and scoring frame-level proposals. Our method extracts a set of candidate regions at the frame level. We use EdgeBoxes [Zitnick and Dollár, 2014], as they obtain a high recall even when considering relatively few proposals [Hosang et al., 2015]. Each proposal is represented with CNN features [Gkioxari and Malik, 2015]. These CNN features leverage both static and motion information and are trained to discriminate the actions against background regions. This is capital since most of the proposals do not contain any action. For each class, a hard negative mining procedure is performed in order to train an action-specific classifier. Given a test video, frame-level candidates are scored with these action-specific classifiers.

Tracking best candidates. Given the frame-level candidates of a video, we select the highest scoring ones per class and track them throughout the video. Our tracking method is based on a standard tracking-by-detection

approach leveraging an instance-level detector as well as a class-level classifier. The detector is based on the same CNN features as the first stage. We perform the tracking multiple times for each action, starting from the proposal with the highest score that does not overlap with previous computed tracks.

Scoring tracks. The CNN features only contain information extracted at the frame level. Consequently, they are not able to capture the dynamics of an action across multiple frames. Thus, we propose to use spatio-temporal local features. We introduce a spatio-temporal motion histogram (STMH). It is inspired by the success of dense trajectory descriptors [Wang et al., 2013]. Given a fixed-length chunk from a track, we divide it into spatio-temporal cells and compute an histogram of gradient, optical flow and motion boundaries in each cell. A hard negative mining is employed to learn a classifier for each class. We also tried using the classical Fisher Vector aggregation of improved dense trajectories (IDT) [Wang et al., 2015]. The final score is obtained by combining CNN with STMH or IDT classifiers.

Temporal localization. To detect the temporal extent of an action, we use a multi-scale sliding window approach over tracks. At test time, we rely on temporal windows of different lengths that we slide with a stride of 10 frames over the tracks. We score each temporal window according to CNN features, local features (STMH or IDT) and a duration prior learned on the training set. For each track, we then select the window with the highest score.

7.3 Detailed description of the approach

In this section, we detail the four stages of our action localization approach. Given a video of T frames $\{I_t\}_{t=1..T}$ and a class $c \in \mathcal{C}$ ($\mathcal{C}$ being the set of classes), the task consists in detecting if the action c appears in the video and if yes, when and where. In other words, the approach outputs a set of regions $\{R_t\}_{t=t_b..t_e}$ with t_b (resp. t_e) the beginning (resp. end) of the predicted temporal extent of the action c and R_t the detected region in frame I_t.

7.3.1 Frame-level proposals with CNN classifiers

Frame-level proposals. State-of-the-art methods [Girshick et al., 2014] for object localization replace the sliding-window paradigm used in the past

decade by object proposals. Instead of scanning the image at every location, at several scales, object proposals allow to significantly reduce the number of candidate regions, and narrow down the set to regions that are most likely to contain an object.

For every frame, we extract EdgeBoxes [Zitnick and Dollár, 2014] using the online code [1] and keep the best 256 proposals according to the EdgeBox score. We denote by $\mathcal{P}_t$ the set of object proposals for a frame I_t. In Section 7.3.2, we introduce a tracking approach that makes our method robust to missing proposals.

CNN features. Recent work on action recognition [Simonyan and Zisserman, 2014] and localization [Gkioxari and Malik, 2015] have demonstrated the benefit of CNN feature representations, applied separately on images and optical flows. We use the same set of CNN features as Gkioxari and Malik [2015].

Given a region resized to 227×227 pixels, a spatial-CNN operates on RGB channels and captures the static appearance of the actor and the scene, while a motion-CNN takes as input optical flow and captures motion pattern. The optical flow signal is transformed into a 3-dimensional image by stacking the x-component, the y-component and the magnitude of the flow. Each image is then multiplied by 16 and converted to the closest integer between 0 and 255. In practice, optical flow is estimated using the online code [2] from Brox et al. [2004]. For a region R, the CNN features we use are the concatenation of the fc7 layer (4096 dimensions) from the spatial-CNN and motion-CNN, see Figure 7.2.

CNN training. We use the same architecture and training procedure as Gkioxari and Malik [2015]. We give a brief presentation below and refer to their work for more details. The architecture is the same for both networks with 5 convolution layers interleaved by pooling and normalization, and then 3 fully connected layers interleaved with dropout. The last fully connected layer (fc8) has $|\mathcal{C}| + 1$ outputs, one per class and an additional output for the background. Similar to Girshick et al. [2014], during training, the proposals that overlap more than 50% with the ground-truth are considered as positives, the others as background. Regions are resized to fit the network size (227×227) and randomly flipped. The spatial-CNN is initialized with a model trained on full images from ImageNet and fine-tuned for object detection on Pascal VOC 2012 [Girshick et al., 2014]. For

1. https://github.com/pdollar/edges
2. http://lmb.informatik.uni-freiburg.de/resources/software.php

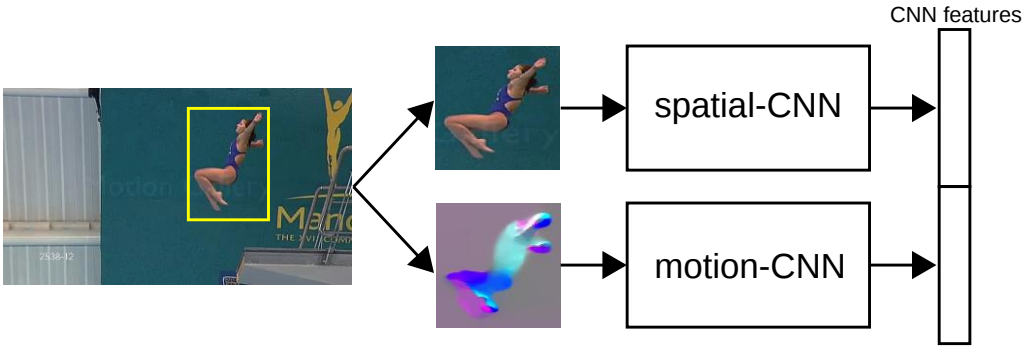


Figure 7.2 – Illustration of CNN features for a region R. The CNN features are the concatenation of the fc7 layer from the spatial-CNN and motion-CNN, *i.e.*, a 2x4096 dimensional descriptor.

the motion-CNN, initialization weights are trained for the task of action recognition on the UCF-101 dataset [Soomro et al., 2012] with full frames of the training set of split 1. We then fine-tune the networks with back-propagation using Caffe [3] [Jia et al., 2014] on the proposal regions for each dataset. Each batch contains 25% of non-background regions.

Action classifiers. For each action class $c \in \mathcal{C}$, we train a linear SVM using hard negative mining. The positives are given by the ground-truth annotations and negatives by all proposals whose overlap with a ground-truth region is below 30%. At test time, we denote by $S_{\text{CNN}}(c, R)$ the score of a region R for the action class c given by the trained classifier. This yields a confidence score for the region R and an action class c.

7.3.2 Tracking

The second stage consists in tracking the best proposals over the video. We use a tracking-by-detection approach that leverages instance-level and class-level detectors. Let R be a region in frame I_τ for the class c to be tracked. As a result, the tracking stage will output a track $\mathcal{T}_c = \{R_t\}_{t=1..T}$. The track provides a candidate localization for the action c. We first present how the tracker is initialized. Then, we detail the tracking procedure. Finally, we explain the selection of the regions to track.

Initialization. Given a region R to be tracked in frame I_τ, the first step is to refine the position and size of the region by performing a sliding-

3. http://caffe.berkeleyvision.org/

window search both in scale and space in the neighborhood of R. Let $\mathcal{N}(R)$ be the set of windows scanned with a sliding window around the region R. The best region according to the action-level classifier is selected: $R_\tau = \text{argmax}_{r \in \mathcal{N}(R)} S_{\text{CNN}}(c, r)$. The sliding-window procedure using CNN features can be performed efficiently [Giusti et al., 2013, Sermanet et al., 2014].

Given the refined region, we train an instance-level detector using a linear SVM. The set of negatives comprises the instances extracted from boxes whose overlap with the original region is less than 10%. The boxes are restricted to regions in $\mathcal{P}_\tau$, *i.e.*, the proposals in frame τ. The set of positives is restricted to the refined region R_τ. This strategy is consistent with current tracking-by-detection approaches [Hua et al., 2014]. Denote by $S_{\text{inst}}(R)$ the score of the region R with the instance-level classifier. We now present how the tracking proceeds over the video. We first do a forward pass from frame I_τ to the last frame I_T, and then a backward pass from frame I_τ to the first frame.

Algorithm 7.1 Class-specific tracking.

Input: a region R in frame I_τ to track, a class c
Output: a track $\mathcal{T}_c = \{R_t\}_{t=1..T}$
$R_\tau \leftarrow \text{argmax}_{r \in \mathcal{N}(R)} S_{\text{CNN}}(c, r)$
$\mathsf{Pos} \leftarrow \{R_\tau\}$
$\mathsf{Neg} \leftarrow \{r \in \mathcal{P}_\tau | \text{ IoU}(r, R_\tau) < 0.1\}$
For $i = \tau + 1 \ldots T$ and $\tau - 1 \ldots 1$:
—– Learn instance-level classifier from Pos and Neg
—– $R_i \leftarrow \text{argmax}_{r \in \mathcal{N}(R'_i)}(S_{\text{CNN}}(c, r) + S_{\text{inst}}(r))$
—– $\mathsf{Neg} \leftarrow \mathsf{Neg} \cup \{r \in \mathcal{P}_\tau | \text{ IoU}(r, R_i) < 0.1\}$
—– $\mathsf{Neg} \leftarrow \{r \in \mathsf{Neg} | S_{\text{inst}}(r) \geq -1\}$ (restrict to hard negatives)
—– $\mathsf{Pos} \leftarrow \mathsf{Pos} \cup \{R_i\}$

Update. Given a tracked region R_t in frame I_t, we now want to find the most likely location in frame I_{t+1}. We first map the region R_t into R'_{t+1}, by shifting the region with the median of the flow between frame I_t and I_{t+1} inside the region R_t. We then select the best region in the neighborhood of R'_{t+1} using a sliding window that leverages both class-level and instance-level classifiers:

$$R_{t+1} = \underset{r \in \mathcal{N}(R'_{t+1})}{\text{argmax}} \; S_{\text{inst}}(r) + S_{\text{CNN}}(c, r) \quad . \tag{7.1}$$

In addition, we update the instance-level classifier by adding R_{t+1} as a positive exemplar and proposals $\mathcal{P}_{t+1}$ from frame I_{t+1} that do not overlap

with this region as negatives. Note that at each classifier update, we restrict the set of negatives to the hard negatives.

The tracking algorithm is summarized in Algorithm 7.1. By combining instance-level and class-level information, our tracker is robust to significant changes in appearance and occlusion. Note that category-specific detectors were previously used in other contexts, such as face [Kalal et al., 2010] or people [Gall et al., 2010] tracking. We demonstrate the benefit of such detectors in our experiments in Section 7.4.

Proposals selection. We now present how we chose the proposals to track. We first select the subset of classes for which the tracking is performed. To this end, we assign a score to the video for each class $c \in \mathcal{C}$ and keep the top-5. The score for a class c is defined as $\max_{r \in \mathcal{P}_t, t=1..T} S_{\text{CNN}}(c, r)$, *i.e.*, we keep the maximum score for c over all proposals of the video.

When generating tracks for the class c, we first select the proposal with the highest score over the entire video. We run the tracker starting from this region and obtain a first track. We then perform the tracking iteratively, starting a new track from the best proposal that does not overlap with any previous track from the same class. In practice, we compute 2 tracks for each selected class.

7.3.3 Track descriptor

So far, we have only used features extracted on individual frames. Clearly, this does not capture the dynamics of the action over time. To overcome this issue, we compare two approaches. As a first option, we introduce spatio-temporal motion histogram (STMH) feature at the track level. The second approach consists in using Fisher Vector on improved dense trajectories (IDT) from a track.

Spatio-temporal motion histogram. Similar to Wang et al. [2013], we rely on histograms of gradient and motion extracted in spatio-temporal cells. Given a track $\mathcal{T}_c = \{R_t\}_{t=1..T}$, it is divided into temporal chunks of $L = 15$ frames, with a chunk starting every 5 frames. Each chunk is then divided into N_t temporal cells, and each region R_t into $N_s \times N_s$ spatial cells, as shown in Figure 7.3. For each spatio-temporal cell, we perform a quantization of the per-pixel image gradient into an histogram of gradients (HOG) with 8 orientations. The histogram is then normalized with the L2-norm. Similarly, we compute HOF, MBHx and MBHy by replacing the image gradient by the optical flow and the gradient of its x and y

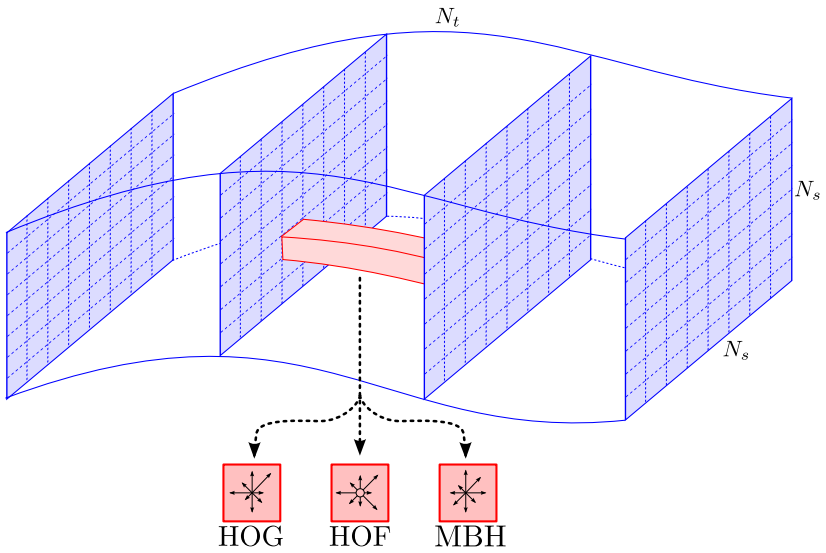


Figure 7.3 – Illustration of STMH. A chunk is split into spatio-temporal cells for which an histogram of gradient, optical flow and motion boundaries is computed.

components. For HOF, a bin for an almost zero value is added, with a threshold at 0.04. In practice, we use 3 temporal cells and 8×8 spatial cells, resulting in $3 \times 8 \times 8 \times (8+9+8+8) = 6336$ dimensions. Note that we use more spatial cells than Wang et al. [2013], as our regions are on average significantly larger than the 32×32 patch they use.

Improved dense trajectories. We compare STMH with the classical improved dense trajectories (IDT) representation [Wang et al., 2015]. Dense trajectories are extracted and described using HOG, HOF and MBHx and MBHy. For a given track, we build a Fisher Vector per descriptor type, using only the trajectories that start inside the track. Each of the 4 Fisher Vectors is then independently power-normalized and L2-normalized [Sánchez et al., 2013]. A tube is finally described by the concatenation of the 4 normalized Fisher Vectors, resulting in 102400 dimensions.

Fusion. For each action, we train a linear SVM using hard negative mining. The set of positives is given by features extracted along the ground-truth annotations, while the negatives are given by cuboids (spatially and temporally) centered at the proposals that do not overlap with the ground-truth. Let $S_{\text{desc}}(c, \mathcal{T})$ be the score according to local features classifiers. For IDT, $S_{\text{desc}}(c, \mathcal{T})$ is the output of the linear classifer for the representation

extracted from the track $\mathcal{T}$ for the action c. For STMH, it is the average of the scores for all the chunks of length L inside the track.

Given a track $\mathcal{T} = \{R_t\}_{t=1..T}$, we score it by summing the scores from the CNN averaged over all frames, and the scores from the local descriptors:

$$S(\mathcal{T}) = \sigma\Big(S_{\text{desc}}(c, \mathcal{T})\Big) + \sigma\Big(\sum_{t=1}^{T} S_{\text{CNN}}(c, R_t)\Big)\,, \tag{7.2}$$

where $\sigma(x) = 1/(1+e^{-x})$. We summarize the resulting approach for spatio-temporal detection in Algorithm 7.2.

7.3.4 Temporal localization

Similar to the winning approach in the temporal action detection track of the Thumos 2014 challenge [Oneata et al., 2014b], we use a sliding-window strategy for temporal localization. However, we apply the sliding window directly on each track $\mathcal{T}$, while Oneata et al. [2014b] used features extracted for the full frames. The window length takes values of 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 300, 450 and 600 frames. The sliding window has a stride of 10 frames. For each action c, we learn the frequency of its durations on the training set. We score each window using the score described above based on CNNs features and spatio-temporal local features (STMH or IDT), normalized with a sigmoid, and multiply it with the per-class duration prior. For each track, we keep the top-scoring window as spatio-temporal detection.

7.4 Experimental results

In this section, we present experiment results on three action localization approach: UCF-Sports, J-HMDB and UCF-101. We refer to Section 6.3 for details on these benchmarks as well as the evaluation metrics. We first study the impact of both the tracking and the class selection, and then provide a parametric study of STMH. Finally, we show that our approach outperforms the state of the art for spatio-temporal action localization.

7.4.1 Impact of the tracker

The strength of our approach lies in the combination of class-specific and instance-level detectors in the tracker. To measure the benefit of this combination, Table 7.1 compares the performance when removing one of

Algorithm 7.2 Spatio-temporal detection in a test video.

Input: a test video $\{I_t\}_{t=1...T}$
Output: a list of detections $(c, \mathcal{T}, \text{score})$
For $t = 1..T$
—— $\mathcal{P}_t = \text{EdgeBoxes}(I_t)$
—— **For** $r \in \mathcal{P}_t$
———— Compute $S_{\text{CNN}}(c, r)$
$\mathcal{C}' \leftarrow$ class selection (see Sec. 7.3.2)
Detections $\leftarrow$ [ ]
For $c \in \mathcal{C}'$
—— **For** $i = 1...$ntracks (we generate ntracks=2 tracks per label, see Sec. 7.3.2)
———— $R, \tau \leftarrow \text{argmax}_{r \in \mathcal{P}_t, t=1..T} S_{\text{CNN}}(c, r)$
(proposal to track without overlap with previous tracks)
———— $\mathcal{T} \leftarrow \text{Tracking}(R, I_\tau, c)$ (Algorithm 7.1)
———— score $\leftarrow \sigma(S_{\text{desc}}(c, \mathcal{T})) + \sigma(\sum_{R_t \in \mathcal{T}} S_{\text{CNN}}(c, R_t))$ (Equation 7.2)
———— Detections $\leftarrow$ Detections $\cup \{(c, \mathcal{T}, \text{score})\}$

Detectors in the tracker	recall-track		mAP	
	UCF-Sports	J-HMDB	UCF-Sports	J-HMDB
instance-level + class-level	**98.75%**	91.74%	**90.50%**	**59.74%**
instance-level only	85.42%	**94.59%**	74.27%	54.32%
class-level only	92.92%	81.28%	85.67%	53.25%

Table 7.1 – Impact of the detectors used in the tracker. We measure if the tracks generated for the ground-truth label cover the ground-truth tracks (recall-track). We also measure the impact of the tracker on the final detection performance (mAP). The experiments are done on UCF-Sports and J-HMDB (split 1 only) using CNN and STMH features.

them. 'Recall-tracks' measures if at least one of the 2 generated tracks for the ground-truth action covers the ground-truth annotations (IoU ≥ 0.5), *i.e.*, it measures the recall at the track level. We also measure the impact on the final detection performance (mAP) by running our full pipeline with each tracker.

On UCF-Sports, tracking obtained by combining the detectors leads to the highest recall. Using the instance-level detector significantly degrades the recall by 13%. This can be explained by the abrupt changes in pose and appearance for actions such as diving or swinging. On the other hand, the instance-level detector performs well on the J-HMDB dataset, which contains more static actions.

Combining instance-level and class-specific classifiers also gives the best performance in term of final detection results. On UCF-Sports, this is

	CNN		CNN + STMH	
	Linking	Tracking	Linking	Tracking
SelectiveSearch [Uijlings et al., 2013]	75.94%	83.77%	77.1%	84.9%
EdgeBoxes-256 [Zitnick and Dollár, 2014]	79.89%	**88.23%**	83.2%	**90.5%**

Table 7.2 – Comparison of tracking and linking, SelectiveSearch and EdgeBoxes-256 proposals with CNN features only or CNN + STMH on UCF-Sports (localization in mAP).

mainly due to the higher recall. On J-HMDB, we find that using the instance-level detector only leads to a better recall but the precision decreases because there are more tracks from an incorrect label that have a high score.

Table 7.2 compares the localization mAP on UCF-Sports when using our proposed tracker or a linking strategy as Gkioxari and Malik [2015]. We experiment with proposals from SelectiveSearch [Uijlings et al., 2013] (approximately 2700 proposals per frame) or EdgeBoxes [Zitnick and Dollár, 2014] (top-256), with CNN features only or combined with STMH. We can see that using EdgeBoxes instead of SelectiveSearch leads to a gain of 6% when using STMH in addition to CNN features. Using a tracking strategy leads to a further gain of 7%, with in addition a more refined localization, see Figure 7.5. This shows that the tracker is a key component to the success of our approach.

7.4.2 Class selection

We now study the impact of selecting the top-5 classes based on the maximum score over all proposals from a video for a given class, see Section 7.3.2. We measure the percentage of cases where the correct label is in the top-k classes and shows the results in Figure 7.4 (blue curve). Most of the time, the correct class has the highest maximum score (around 85% on UCF-Sports and 61% on J-HMDB). If we use top-5, we misclassify less than 10% of the videos on J-HMDB, and 0% on UCF-Sports.

Figure 7.4 also shows that recall (green) is lower than the top-k accuracy because the generated tracks might not have a sufficient overlap with the ground-truth due to a failure of the tracker. The difference between recall and top-k accuracy is more important for large k. This can be explained by the fact that the class-level detector performs poorly in videos for which the correct label has a low rank, therefore the class-specific tracker performs poorly as well.

In addition, we display in red the evolution of the mAP on UCF-Sports

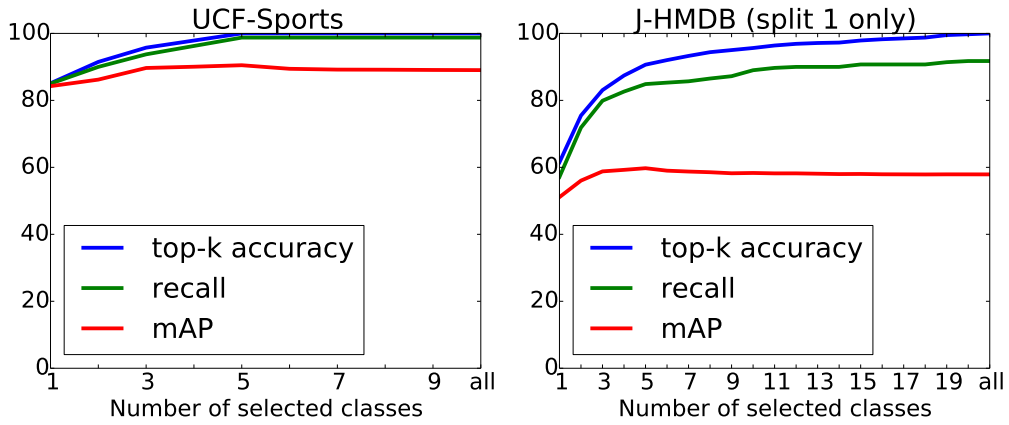


Figure 7.4 – Impact of the class selection on UCF-Sports (*left*) and J-HMDB (*right*) datasets. In blue, top-k accuracy is shown, *i.e.*, the percentage of cases where the correct label is in the top-k classes. The recall when changing the number of selected classes is shown in green and the mAP in red (using CNN and STMH features).

and J-HMDB (split 1 only) using CNN and STMH features when changing the number of selected classes. Initially, the performance significantly increases as this corrects the cases where the correct label is top-k but not first, *i.e.*, the recall increases. The performance then saturates since, even in the case where a new correct label is tracked over a video, the final score will be low and will not have an important impact on the precision. As a summary, selecting the top-k classes performs similar as keeping all classes while it significantly reduces the computational time.

7.4.3 STMH parameters

We now study the impact of the number of temporal and spatial cells in STMH. For evaluation, we consider the classification task and learn a linear SVM on the descriptors extracted from the ground-truth annotations of the training set. We then predict the label on the test set, assuming the ground-truth localization is known, and report mean Accuracy. Results are shown in Table 7.3. We can see that the best performance is obtained with $N_s = 8$ spatial cells on both datasets, independently of the number of temporal cells N_t. By increasing the number of cells to a higher value, *e.g.* 16, the descriptor becomes too specific for a class. When using a unique temporal cell, *i.e.*, $N_t = 1$, the performance is significantly worse than for $N_t = 3$. We choose $N_s = 8$ and $N_t = 3$ in the remainder of the experiments. The

N_t	N_s	dimension	UCF-Sports	J-HMDB
1	2	132	76.07%	38.78%
	4	528	80.00%	48.58%
	8	2112	82.50%	51.71%
	16	8448	81.67%	49.54%
2	2	264	77.98%	41.21%
	4	1056	80.00%	49.41%
	8	4224	87.50%	52.72%
	16	16896	82.50%	48.89%
3	2	396	82.74%	41.38%
	4	1584	83.33%	50.52%
	8	6336	87.50%	**54.26%**
	16	25344	84.17%	47.98%
5	2	660	79.64%	41.51%
	4	2640	80.00%	50.84%
	8	10560	**88.33%**	52.11%
	16	42240	84.17%	47.81%
IDT			**91.90%**	**57.99%**

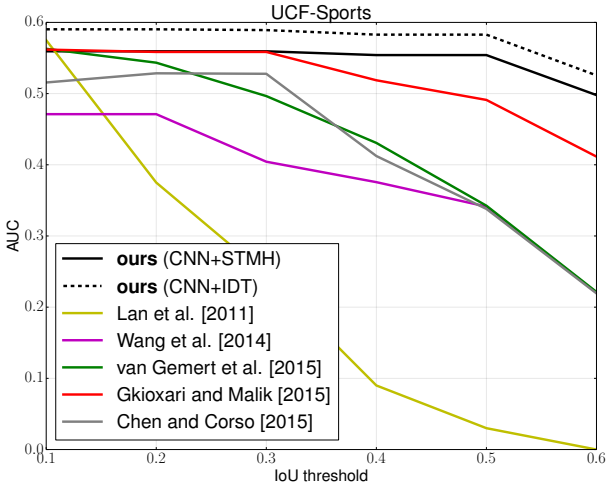
Table 7.3 – Comparison of mean-Accuracy when classifying ground-truth tracks using STMH with different numbers of temporal (N_t) and spatial (N_s) cells.

resulting STMH descriptor has $6,336$ dimensions.

Using the same protocol, we obtain a slightly better performance of 91.9% for UCF-Sports and 57.99% for J-HMDB using IDT. Note that STMH is an order of magnitude faster to extract as it does not require features aggregation and has a lower dimension.

7.4.4 Comparison to the state of the art

In this section, we compare our approach to the state of the art. On UCF-Sports, past works usually plot ROC curves and report Area Under the Curve (AUC). Figure 7.5 (left) shows a comparison with the state of the art using the same protocol for different IoU thresholds δ. We can observe that our approach outperforms the state of the art. Note that at a low threshold, all methods obtain a comparable performance, but the gap widens for larger one, *i.e.*, more precise detections. Indeed, our spatial localization enjoys a high precision thanks to the tracking: the position of the detected region is refined in each frame using a sliding window. As a consequence, the IoU between our detected tracks and the ground-truth is high, explaining why our performance remains constant between a low threshold and a high threshold δ. Figure 7.6 shows example results. Despite important changes



Method	mAP
Gkioxari and Malik [2015]	75.8
ours (CNN only)	88.2
ours (CNN+STMH)	90.5
ours (CNN+IDT)	**95.1**

Figure 7.5 – Comparison to the state of the art on UCF-Sports. *Left:* AUC for varying IoU thresholds. *Right:* mAP at $\delta = 50\%$ with different variants for the features used for scoring tracks.

Figure 7.6 – Example results from the UCF-Sports dataset.

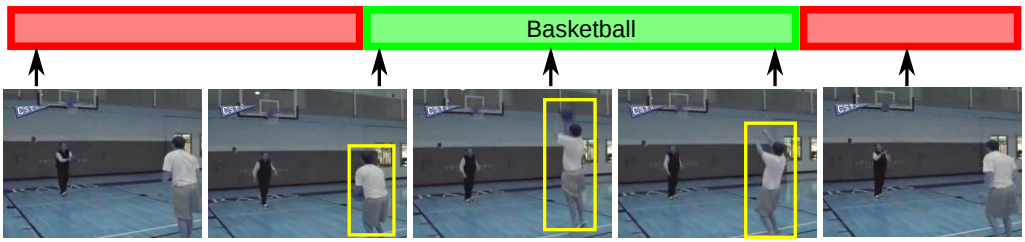


Figure 7.7 – Example of spatio-temporal detection for the Basketball action on UCF-101.

δ	0.2	0.3	0.4	**0.5**
Gkioxari and Malik [2015]				53.3
ours (CNN only)	58.1 ± 2.1	58.0 ± 1.9	57.7 ± 2.1	56.5 ± 2.6
ours (CNN + STMH)	63.1 ± 1.8	63.5 ± 1.8	62.2 ± 1.9	60.7 ± 2.7
ours (CNN + IDT)	68.9 ± 1.1	68.7 ± 1.1	68.2 ± 1.2	66.8 ± 1.5

Table 7.4 – Comparison to the state of the art on J-HMDB using mAP for varying IoU thresholds δ. We also report the standard deviation among the splits.

δ	0.05	0.1	**0.2**	0.3
Yu and Yuan [2015]	42.8			
ours (CNN + STMH)	54.3	51.7	46.8	37.8
ours (CNN + IDT)	69.7	68.1	60.6	46.3

Table 7.5 – Localization results (mAP) on UCF-101 (split 1) for different IoU thresholds δ.

in appearance, the actor is successfully tracked throughout the video. For detection, mAP is more suitable as it does not depend on negatives. Results are shown in Figure 7.5 (right). We outperform the state of the art with a margin of 19% and obtain a mAP of 95.1% using CNN and IDT features. We also compute the mAP when scoring with CNN features only (resp. with CNN and STMH features), and observe a drop of 7% (resp. 5%).

The results for the J-HMDB dataset are given in Table 7.4. We also outperform the state of the art by more than 13% on J-HMDB at a standard threshold $\delta = 0.5$. In particular, adding IDT (resp. STMH) to CNN features leads to an improvement of 10% (resp. 4%). This shows that CNN and spatio-temporal local features are complementary. We can also see that the mAP is stable *w.r.t.* the threshold δ. This highlights once again the high precision of the spatial detections, *i.e.*, they all have a high overlap with the ground-truth, thanks to the tracking.

Figure 7.8 – Example results from the UCF-101 dataset.

Finally, we report the results for spatio-temporal detection on the UCF-101 dataset in Table 7.5. We obtain a mAP of more than 61% (resp. 47%) using CNN and IDT (resp. STMH) features at a standard threshold $\delta = 20\%$ despite the challenge of detecting an action both spatially and temporally. At a threshold $\delta = 5\%$, we obtain a mAP of 70% compared to 43% reported by Yu and Yuan [2015]. Figure 7.7 and 7.8 show example results. We can observe that the result for the action 'Basketball' is precise both in space and time. While most of the 24 action classes cover almost the entire video, *i.e.*, there is no need for temporal localization, the action 'Basketball' covers on average one fourth of the video, *i.e.*, it has the shortest relative duration in UCF-101. For this class our temporal localization approach improves the performance significantly. The AP for Basketball is 28.6% ($\delta = 20\%$) with our full approach. If we remove the temporal localization step, the performance drops to 9.63%. This shows that our approach is capable of localizing actions in untrimmed videos. With respect to tracking in untrimmed videos, tracking starts from the highest scoring proposal in both directions (forward and backward) and continues even if the action is no longer present. The temporal sliding window can then localize the action and removes parts without the action.

7.5 Conclusion

In this chapter, we presented an effective approach for action localization in both space and time. Our approach builds upon object proposals

extracted at the frame level that we track throughout the video. Tracking is effective, as we combine instance-level and class-level detectors. The resulting tracks are scored by combining classifiers learned on CNN features and spatio-temporal descriptors. A sliding window finally performs the temporal localization of the action. The proposed approach improves on the state of the art by a large margin on three benchmarks: UCF-Sports, J-HMDB and UCF-101.

Nevertheless, this approach requires bounding box annotation in the training videos in order to train the class-specific detector. Moreover, the tracking is performed independently for each class, thus multiplying the complexity of the approach. The next chapter introduces an approach to overcome these limitations by leveraging a generic human detector.

Chapter 8

Human Tracks for Weakly-Supervised Action Localization

Contents

8.1 Introduction

Localizing actions in videos is a challenging task which has received increasing attention over the past few years. Recently, significant progress

was achieved, see for example [Gkioxari and Malik, 2015, Weinzaepfel et al., 2015a, Wang et al., 2014]. Nevertheless, these methods require a large amount of supervision. For instance, per-frame bounding box annotations are used for training class-specific detectors [Gkioxari and Malik, 2015, Weinzaepfel et al., 2015a]. Wang et al. [2014] additionally require pose annotations, as they represent actions as a sequence of skeleton models. Several works have suggested to generate action proposals before classifying them [Marian Puscas et al., 2015, van Gemert et al., 2015]. However, supervision is still required for learning to classify these hundreds or thousands of proposals. Consequently, all these approaches are restricted to relatively small datasets because of the supervision cost and can not be generalized easily to more classes.

In this chapter, we propose a weakly-supervised action localization method, *i.e.*, that does not require any spatial annotation for training. The first step of our approach consists in extracting human tubes from videos. Using human tubes for action recognition is not a novel idea [Kläser et al., 2010, Laptev and Pérez, 2007, Yu and Yuan, 2015]. However, we show for the first time that extracting highly robust human tubes is possible by leveraging a recent state-of-the-art object detection approach [Ren et al., 2015], a large annotated dataset of humans in a variety of poses [Andriluka et al., 2014] and state-of-the-art tracking-by-detection [Hare et al., 2011, Kalal et al., 2012]. We show that a small number of human tubes per video is sufficient to obtain more than 95% recall on challenging action localization datasets. Our approach outperforms existing video proposal methods by an order of magnitude, as shown in Section 8.3.3. As a second step, we describe human tubes with improved dense trajectories [Wang et al., 2015] and use multi-fold Multiple Instance Learning (MIL) [Cinbis et al., 2016] to select human tubes containing the action. This manages to select proposals corresponding to the annotated action category accurately. During testing, we extract human tubes and we obtain a mAP using classifiers learned from the tubes selected in the training videos. Figure 8.1 illustrates our weakly-supervised action localization method. Our approach significantly outperforms the state of the art for weakly-supervised action localization and is only slightly worse than the best fully-supervised methods, with a mAP of 84% on UCF-Sports, 54% on J-HMDB and 45% on UCF-101.

We also introduce a new action localization dataset, named **DALY** for Daily Action Localization in YouTube. It overcomes the limitations of existing datasets which mostly consist of videos trimmed to the action, have limited action types, *i.e.*, often sports only, and contain in many cases only one human per video. The DALY dataset consists of more than 31 hours of videos from YouTube with high quality spatial and temporal annotations

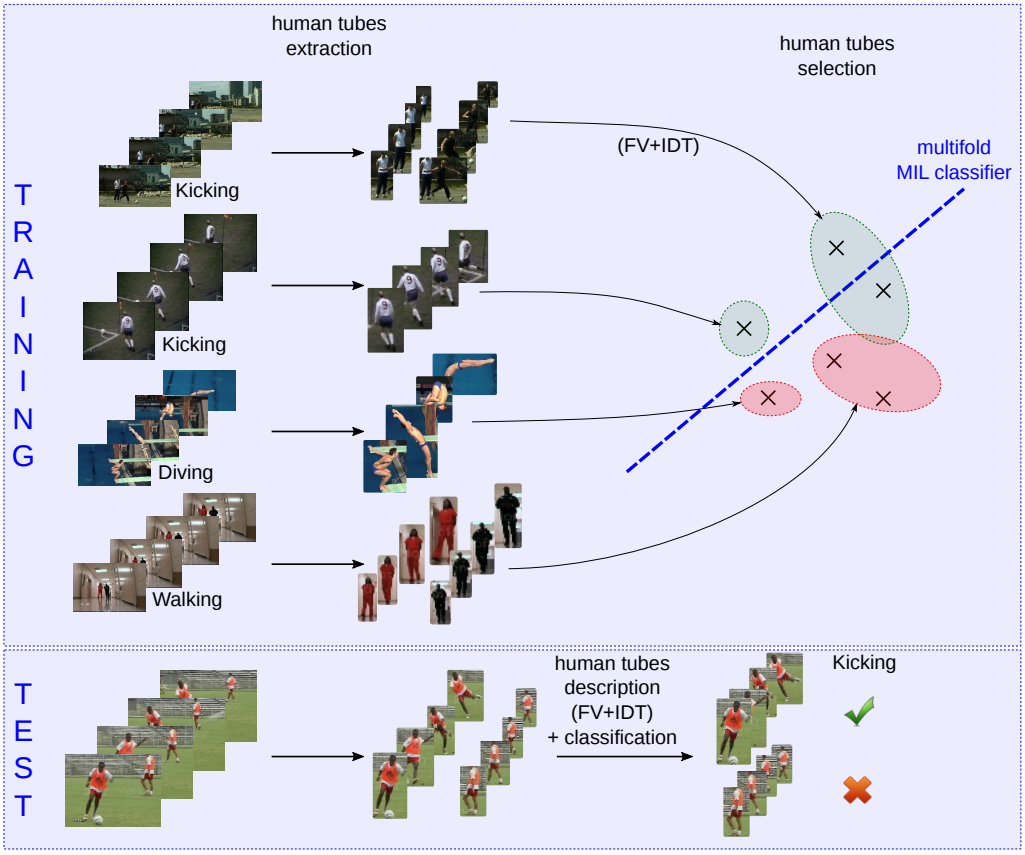


Figure 8.1 – Illustration of our weakly-supervised action localization approach. For training, we extract human tubes in each video and learn a classifier with weak supervision using Multiple Instance Learning for selecting tubes containing the action. At test time, human tubes are extracted and scored using the learned classifiers.

for 10 realistic daily actions. The task is to localize relatively short actions in long untrimmed videos. Furthermore, it includes videos with multiple humans performing actions at the same time. On the DALY dataset, our tubes have a spatial recall of 82%, but the detection task is extremely challenging, and we obtain 10.8% mAP.

Closest references. Our approach is based on human tubes. In a similar spirit, Kläser et al. [2010] use a human detector and build human tubes using KLT features tracks. The human tracks are then classified with HOG-3D descriptors [Kläser et al., 2008]. Our human tubes are significantly more robust to huge variations in pose and appearance thanks to a human-specific

tracking-by-detection approach [Hare et al., 2011, Kalal et al., 2012] as well as recent advances in detectors [Ren et al., 2015] and datasets [Andriluka et al., 2014]. In addition, our approach is weakly-supervised, *i.e.*, does not require bounding box annotation for labeling the training samples.

Recently, CNNs for human action localization have emerged [Gkioxari and Malik, 2015, Weinzaepfel et al., 2015a]. These approaches rely on appearance and motion CNNs for classifying region proposals in individual frames. Tracks are obtained by combining class-specific detections with either temporal linking based on proximity [Gkioxari and Malik, 2015] or by class-specific tracking-by-detection [Weinzaepfel et al., 2015a]. Our approach also relies on tracking-by-detection, but performs generic human detections and tracking. Thus the complexity is divided by the number of classes and the tracks can be used for weakly-supervised learning. Furthermore, our tracker is significantly faster, as it uses the region-pooling layer of faster R-CNN [Ren et al., 2015] for feature computation.

Our human tubes can also be viewed as action proposals [Jain et al., 2014b, van Gemert et al., 2015, Yu and Yuan, 2015, Marian Puscas et al., 2015, Oneata et al., 2014a]. While all these methods generate thousands of proposals and require ground-truth to annotate the proposals used for training, we obtain only several human tube proposals, thus allowing us to apply Multiple Instance Learning (MIL) effectively. More recently, Mettes et al. [2016] learn to classify proposals with less supervision, requiring only several points inside the ground-truth tube.

For weakly-supervised action localization, Siva and Xiang [2011] define cuboids of different time lengths around detected humans, describe them with STIPs [Laptev, 2005] and then use Multiple Instance Learning (MIL). Their method can thus only be used for static humans whereas ours can be generalized to more complex and realistic motions. Similar to the method from Chen and Corso [2015], our approach is also based on action proposals (human tubes in our case). However, in contrast to their method, ours uses MIL and thus does not assume only one action per video during training.

Outline. This chapter is organized as follows. Section 8.2 presents the datasets used for evaluation and introduces our DALY dataset. We then describe our approach for extracting human tubes in a video (Section 8.3) and our weakly-supervised learning method using multi-fold MIL (Section 8.4). Finally, experimental results are presented in Section 8.5.

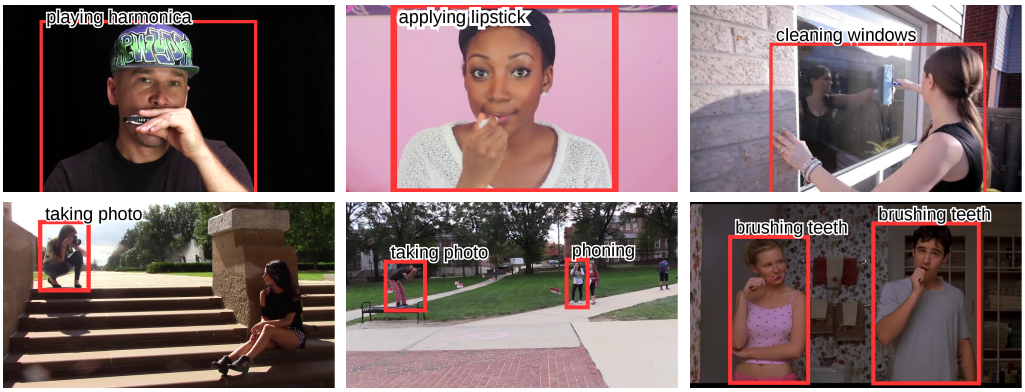


Figure 8.2 – Example frames from the DALY dataset.

8.2 Dataset and evaluation

Experiments are conducted three existing action localization datasets (namely UCF-Sports, J-HMDB and UCF-101), described in Section 6.3. However, these benchmarks are limited by the type of actions (mainly sports), the lack of diversity in the videos and the duration of the videos, see Table 8.1 for details. There is a clear need for a realistic dataset for spatio-temporal action localization. We introduce DALY, a dataset for **D**aily **A**ction **L**ocalization in **Y**ouTube[1]. The DALY dataset consists of 31 hours of YouTube videos, with spatial and temporal annotations for 10 everyday human actions: *applying make up on lips, brushing teeth, cleaning floor, cleaning windows, drinking, folding textile, ironing, phoning, playing harmonica and taking photos/videos*, see Figure 8.2. The videos are collected from YouTube using related queries. We collect 510 videos, *i.e.*, 51 videos per category, representing a total of around 3.3M frames. Each video lasts between 1 and 20 minutes with an average duration of 3min 45s. We generate a split with 31 training videos and 20 test videos for each class, ensuring that videos with the same characters or scenes are in the same set.

Temporal annotations of the 10 actions result in 3724 instances in total. Actions are short (8 seconds on average) with some classes having very brief instances (*e.g. drinking*) or somewhat longer (*e.g. brushing teeth*). The actions cover about 25% of the videos. For each instance in the test set, we annotate the spatial extent, *i.e.* a bouding box around the actor, for 5 regularly sampled frames. Note that videos can contain simultaneously multiple actions, see bottom row, middle and right column in Figure 8.2.

1. The DALY dataset is available online at http://pascal.inrialpes.fr/data/daly/.

	DALY	UCF-Sports	J-HMDB	UCF-101	MSR-II
#classes	10	10	21	24	3
action types	everyday	sports	everyday	sports	artificial
#videos	510	150	928	3207	54
avg resolution	1290x790	690x450	320x240	320x240	320x240
total #frames	3.3M	10k	32k	558k	41k
avg video dur.	3min 45s	5.8s	1.4s	5.8s	51s
avg action dur.	8s	5.8s	1.4s	4.5s	6s
#instances	3724	154	928	4030	203
spatial annotation	subset	all	all	all	cuboid

Table 8.1 – Comparison of our DALY dataset with existing action localization datasets.

More details on the DALY dataset are given in Appendix B.

8.3 Building human tubes

Given a video V with F frames, spatio-temporal action localization aims at detecting when and where actions are performed. More precisely, for each instance of an action, a tube T, *i.e.*, a set of bounding boxes $T = \{b_f\}_{f=f_s..f_e}$ around the actor is returned, with one box b_f per frame f during the predicted temporal extent of the action between frames f_s and f_e.

This section presents the human detector (Section 8.3.1) and the human-specific tracker (Section 8.3.2) used to obtain reliable human tubes. We denote by $\mathcal{T}_V$ the set of human tubes proposals extracted for a video V. Section 8.3.3 presents an evaluation of the obtained human tubes and a comparison with state-of-the-art action proposals.

8.3.1 Human detector

Faster R-CNN detector. We use the state-of-the-art detector Faster R-CNN [Ren et al., 2015] to train our human detector. Faster R-CNN integrates a Region Proposals Network (RPN) to produce 300 proposals per image, that are classified as background or a particular class (here 'human'). Detection is extremely fast (around 200ms per image on a GPU) since convolutions are only computed once and then used both for generating proposals and for scoring each proposal using a region pooling layer. Faster R-CNN also includes bounding box regression to overcome the stride of the network which limits the precision of the estimated localization. We use the

Figure 8.3 – Examples of bounding boxes (green) estimated from annotated joints and head (yellow) on the MPII Human Pose dataset.

Figure 8.4 – Example results of our human detector.

variant with end-to-end training[2] using VGG net with 16 layers [Simonyan and Zisserman, 2015]. Given a frame f, let $\mathcal{B}_f$ be the set of bounding box detections output by the network and $s_H(b)$ be the human score after softmax of a box $b \in \mathcal{B}_f$. We now present the data we leverage for training the network.

External training data. We use the MPII Human Pose dataset [Andriluka et al., 2014] to leverage training data with sufficient variability. It contains more than 40k annotated poses, including a bounding box around the head and joint positions for the full body. The images represent several frames from around 4000 videos, selected to contain around 500 different activities. We use the training set for which annotations are publicly available. It represents 28778 annotated poses in 17372 images. We obtain a bounding box for each person by taking the box containing the head and all visible joints, with a fixed additional margin of 20 pixels. The bounding boxes are thus not perfect, see Figure 8.3. For instance, they can be slightly too large (top of bounding boxes from left image) or may not cover the extremity of the limbs (hands and feet in the second image). In particular, the bounding boxes are also cropped if some joints are not visible (right image). Nevertheless, this dataset remains large enough and offers a huge variability in the poses. It is thus well suited for training an accurate human detector.

2. https://github.com/rbgirshick/py-faster-rcnn

Qualitative results. Example detections for different action localization datasets (UCF-Sports and J-HMDB) are shown in Figure 8.4. One can see that the obtained human detector is robust to unusual poses (first two examples) and to humans that are not fully-visible (third example). Nevertheless, detection are sometimes imprecise, for example not completely covering the human (second example). Note that there can be multiple detections when different people are present (fourth example).

8.3.2 Human-specific tracker

Once humans are detected, the second step consists in tracking these humans to build tube proposals. To this end, we design a human-specific tracking-by-detection approach. In the following, we present the different steps of our approach. It is summarized in Algorithm 8.1.

Boxes to track. The list of boxes to track is initialized with all detected candidates $\mathcal{B}_f$ in all frames f. We start by building a first track $T = \{b_f\}_{f=1..F}$ using the detection with the highest human score in the entire video sequence. Once we have tracked this detection over the whole sequence, we remove all detections that have an Intersection Over Union (IoU) above 0.3 with any box b_f in this track. We then run the tracker a second time starting from the remaining detection with the highest human score and repeat the process until no boxes are left.

Initial box refinement. Let $\mathfrak{b}$ be the selected box from frame f. As a first step, we refine its position by performing a search for a higher scoring location in its neighborhood. To this end, we perform a forward pass of the network on the frame f using the neighborhood $N(\mathfrak{b})$ of $\mathfrak{b}$, *i.e.*, we use the network without the region proposal part and the bounding box regression branch. The neighborhood $N(\mathfrak{b})$ of $\mathfrak{b}$ simulates a sliding window in scale and space. More precisely, we use the box $\mathfrak{b}$ plus its translation with $0, \pm\mathfrak{s}, \pm 2\mathfrak{s}, \pm 3\mathfrak{s}, \pm 4\mathfrak{s}$ where $\mathfrak{s}$ is the stride of the network, and similarly for rescaled versions of the box by a factor $0, \pm 10\%, \pm 20\%$. The refined position b_f of the initialization box is set to the region of interest $b \in N(\mathfrak{b})$ that maximizes the human score $s_H(b)$.

Instance-level detector initialization. Based on this box b_f, we learn an instance-level detector using the features from the last fully connected layer denoted by $fc7$. More precisely, we learn a linear SVM using as positive $\mathcal{P}$ the feature from the refined box $\mathcal{P} = \{b_f\}$ and as negatives

$\mathcal{N}$ the features from boxes in $\mathcal{B}_f$ that have (almost) no overlap with b_f: $\mathcal{N} = \{b \mid b \in \mathcal{B}_f \text{ s.t. } \text{IoU}(b, b_f) < 0.1\}$.

Tracking procedure. Starting from the refined box b_f, we first track it forward until the end of the sequence. The box b_i of the track at a frame i is set using a sliding window to optimize the human score and the instance-level score. More precisely, at each frame i, we perform a forward pass of the network using the boxes in the neighborhood $N(b_{i-1})$ of the tracked box from the previous frame. b_i is then set to the box b that maximizes the sum of the human score $s_H(b)$ and the instance-level detector probability $s_I(b)$, computed using a sigmoid on the SVM score, among all boxes $b \in N(b_{i-1})$. We continue until we reach the end of the sequence updating the instance-level detector on the fly, see next paragraph. At the end of the forward pass, we reset the instance-level detector and track the initial refined box backward until the beginning of the sequence.

Instance-level detector update. At each frame, we also update the instance-level detector by adding b_i to the set of positive features $\mathcal{P}$ and boxes from $\mathcal{B}_i$ with (almost) no overlap with b_i to the negatives $\mathcal{N}$. At each iteration, we furthermore restrict $\mathcal{N}$ to the hard negatives.

Efficient computation. Launching the tracker multiple times on a sequence can be performed efficiently. To extract human tubes in a video V, we first run Faster R-CNN in every frame f to detect humans. We also keep in cache the results of the last convolution layer denoted by $conv5_f$, which will be used for the sliding window, and the last fully connected layer $fc7_f$, which will be used as feature descriptors for the negatives of the instance-level detector. Now, when performing a sliding window at frame f using a forward pass of the network, we do not need to re-compute the convolutions, and we can directly start from $conv5_f$. Similarly, the $fc7$ feature descriptors of all detections are cached, so updating the set of negatives does not require any additional computation.

8.3.3 Evaluation of our human tube proposals

We compare our human tubes to the state-of-the-art action proposals on the UCF-Sports dataset. Two different metrics are used. The first one is recall@0.5, *i.e.*, the ratio of ground-truth tubes covered by at least one proposal with average spatial IoU over 0.5, while varying the number of proposals. The second metric is the recall, computed over all proposals,

Algorithm 8.1 Extracting human tubes from a video.

Input: a video sequence V with F frames.
Output: a list of human tubes $\mathcal{T}_V = \{T\}$ with $T = \{b_f\}_{f=1..F}$.

$\mathcal{T}_V \leftarrow \varnothing$
ToTrack $\leftarrow \varnothing$
For $f = 1 \ldots F$
—— $\mathcal{B}_f \leftarrow FasterRCNN(f)$ (*human detection*; cache $conv5_f$ and $fc7_f$)
—— ToTrack $\leftarrow$ ToTrack $\cup\{(b, f) \mid b \in \mathcal{B}_f\}$
While ToTrack $\neq \varnothing$
—— $\mathfrak{b}, f \leftarrow \text{argmax}_{(b,i)\in\text{ToTrack}}\, s_H(b)$ (*select highest scoring box to track*)
—— $b_f \leftarrow \text{argmax}_{b\in N(\mathfrak{b})}\, s_H(b)$ (*initial refinement*; forward from $conv5_f$)
—— $\mathcal{P} \leftarrow \{b_f\}$
—— $\mathcal{N} \leftarrow \{b \mid b \in \mathcal{B}_f \text{ s.t. } \text{IoU}(b, b_f) < 0.1\}$ (using cached $fc7_f$)
—— **For** $i = f + 1 \ldots F$ **and** $i = f - 1 \ldots 1$
———— Learn a linear SVM with positives $\mathcal{P}$ and negatives $\mathcal{N}$
———— $b_i \leftarrow \text{argmax}_{b\in N(b_{i\pm1})}\, s_H(b) + s_I(b)$ (*sliding window*; forward from $conv5_i$)
———— $\mathcal{P} \leftarrow \mathcal{P} \cup \{b_i\}$
———— $\mathcal{N} \leftarrow \mathcal{N} \cup \{b \mid b \in \mathcal{B}_i \text{ s.t. } \text{IoU}(b, b_i) < 0.1\}$ (using cached $fc7_i$)
———— $\mathcal{N} \leftarrow \{b \mid b \in \mathcal{N} \text{ s.t. } s_I(b) \geqslant -1\}$ (*restrict to hard negatives*)
—— $\mathcal{T}_V \leftarrow \mathcal{T}_V \cup \{T = \{b_i\}_{i=1..F}\}$ (*add the track to the tube list*)
—— ToTrack $\leftarrow \{(b, i) \mid (b, i) \in \text{ToTrack s.t. } \text{IoU}(b_i, b) < 0.3\}$ (*remove overlapping detections*)

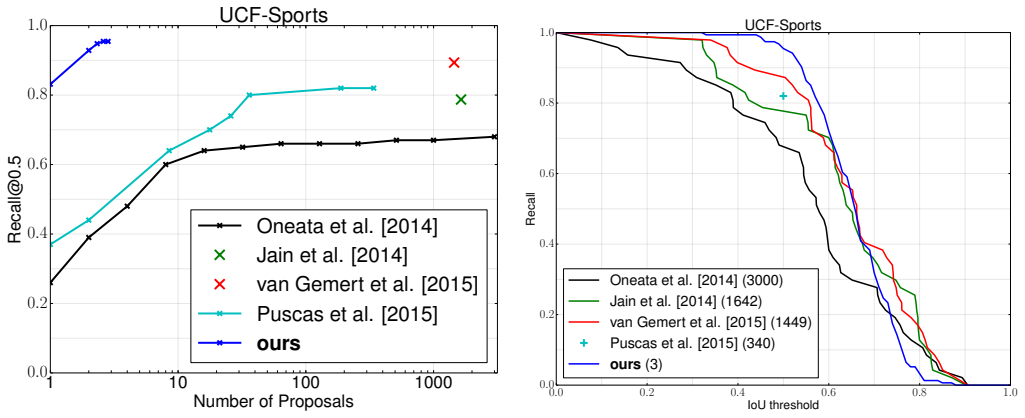


Figure 8.5 – Comparison of our human tubes to state-of-the-art proposals on the UCF-Sports dataset. Left: Recall@0.5 when varying the number of proposals. Right: Recall for varying IoU thresholds for a number of extracted proposals (indicated in parentheses).

for varying IoU thresholds. Results are presented in Figure 8.5. We can clearly see that our human tubes outperform the other approaches by a large margin. Given few proposals (3 on average), we obtain a recall of 95% at 0.5 IoU, whereas state-of-the-art approaches do not reach this recall with significantly more proposals (hundreds or thousands).

On J-HMDB, we also reach a recall of 95% for an IoU threshold at 0.5 with 3 proposals on average per video, demonstrating the effectiveness of our approach. For UCF-101, we obtain a recall of 80% at IoU 0.2 and of 48% at IoU 0.5. The significant decrease in performance is due to the low quality of the videos, sequences have low-resolution and are strongly compressed, and humans tend to be small. For our new DALY dataset, we obtain a spatial recall of 82% at an IoU threshold of 0.5. This is measured on the test set for which spatial annotations are available for a subset of frames. The excellent quality of our human tubes is, thus, confirmed on a realistic challenging dataset.

Figure 8.6 shows a few examples of the highest scoring human tube for several sequences of the DALY dataset. The first four examples show that the human tube extraction performs well despite motion of one arm (first row), turning of the person (second and third row), camera motion (third row) or presence of an animal close to the human (fourth row).

Nevertheless, there are some failure cases due to the fact that the full body disappears (fifth row). In this case, only the feet remain visible causing the failure of the human tracker, as the human detector performs poorly and the instance-level detector is trained on previous frames where the full body is visible. Other failure cases can be explained by a poor performance of the human detector, and thus of the human tube (last two rows). This typically happens when only a small part of the body is visible (sixth row) and in the case of occlusion (last row).

8.4 Weakly-supervised human tube classifier

In this section, we first introduce the feature descriptor of the human tubes (Section 8.4.1). We then present the weakly-supervised MIL learning (Section 8.4.2) and the detection stage (Section 8.4.3). We finally evaluate the MIL training procedure (Section 8.4.4).

8.4.1 Human tube features

For each human tube, we extract improved dense trajectories and aggregate them with a Fisher Vector representation [Wang et al., 2015]. We

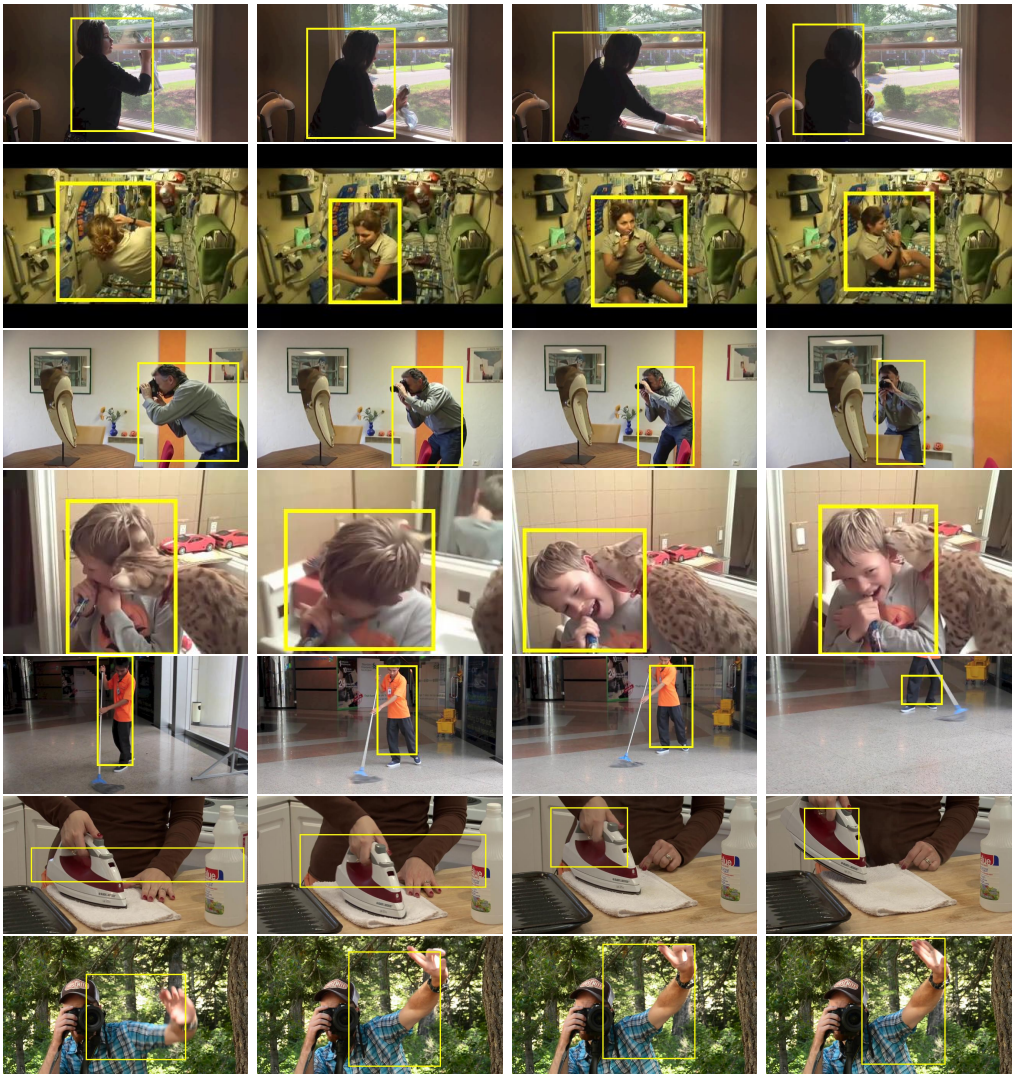


Figure 8.6 – Example of human tubes with successful human tube extraction in the first four rows, and some failure cases in the last three rows. Failures are caused by partial visibility of the human (end of fifth row and sixth row) and missed human detection caused by an occluding camera (last row).

start by extracting the trajectories features for the entire video[3]. For each descriptor type (HOG, HOF, MBHx, MBHy), we first reduce its initial dimension by a factor of 2 using PCA and learn a codebook of 256 Gaussians. For each tube, we build a Fisher Vector per descriptor type, using only the trajectories that start inside the track (increased by a margin of 10%). Each of the 4 Fisher Vectors is then independently power-normalized and L2-normalized [Sánchez et al., 2013]. A tube is finally described by the concatenation of the 4 normalized Fisher Vectors, resulting in 102400 dimensions.

8.4.2 Multi-fold multiple instance learning

We use a Multiple Instance Learning (MIL) formulation to learn a detector in a weakly-supervised setting, *i.e.*, given only information about the presence/absence of a class in the training videos. For a given label, let $\mathcal{V}_P$ (resp. $\mathcal{V}_N$) be the set of positive (resp. negative) videos. MIL alternates between inferring the localization of the action in the positive videos $\mathcal{V}_P$ and using these locations to train a detector. One issue is that it tends to lock onto the initial locations [Cinbis et al., 2016], which is particularly the case for high dimensional descriptors. We therefore resort to the multi-fold variant proposed by Cinbis et al. [2016]. In the following, we present the different steps of our approach. It is summarized in Algorithm 8.2.

Initialization. For each video V, we use the track $T \in \mathcal{T}_V$ with the highest average human score $s_H(T) = \sum_{b_f \in T} s_H(b_f)/F$ as initial tube. Videos are positive and negative according to the annotated video labels.

Iteration. At each iteration, we learn a linear SVM using the current positive and negative tubes, denoted by $\mathcal{T}_P$ and $\mathcal{T}_N$ respectively. We then perform two hard negative mining iterations with negatives being mined only in negative videos $\mathcal{V}_N$. The next step consists in re-estimating the tube used as positive in each positive video. We randomly split the positive videos $\mathcal{V}_P$ into $K = 4$ folds. For each fold, we learn a linear SVM using the positives from all other folds and all negatives. This classifier is then run on videos from this fold. The new estimated location in each positive video of this fold is set to the tube with the highest score and will be used as positive in the next iteration. We perform 10 iterations of multi-fold MIL.

3. https://lear.inrialpes.fr/people/wang/improved_trajectories

Algorithm 8.2 Learning a classifier on human tubes with weak supervision.

Input: a set of positive ($\mathcal{V}_P$) and negative ($\mathcal{V}_N$) videos, each with a list of tubes.
Output: a linear SVM.
$\mathcal{T}_P \leftarrow \{\text{argmax}_{T \in \mathcal{T}_V} S_H(T) \mid V \in \mathcal{V}_P\}$ (*initialization with the tube with highest human detection score*)
$\mathcal{T}_N \leftarrow \{\text{argmax}_{T \in \mathcal{T}_V} S_H(T) \mid V \in \mathcal{V}_N\}$
For each iteration
—– Learn a linear SVM using all positives $\mathcal{T}_P$ and all negatives $\mathcal{T}_N$
—– Perform 2 hard negative iterations in $\mathcal{V}_N$ (*update* $\mathcal{T}_N$)
—– Randomly split $\mathcal{V}_P$ in K folds
—– **For** each fold k
————– Learn a linear SVM using positives from other folds and all negatives
————– Re-estimate the tube in the videos from fold k (*update* $\mathcal{T}_P$)
Learn a linear SVM using all positives $\mathcal{T}_P$ and all negatives $\mathcal{T}_N$

8.4.3 Temporal supervision and detection

Given a test video, we first extract human tubes using the method presented in Section 8.3. We then extract a feature vector for each tube as described in Section 8.4.1. Next, we score each tube for all classes using the classifier learned in a weakly-supervised setting (Section 8.4.2).

For the UCF-101 dataset, which also requires temporal localization, we perform in addition a multi-scale temporal sliding window inside each tube. When training, since the videos are almost trimmed, we first perform multi-fold MIL iterations using features from the whole video. Next, we assume temporal supervision in the training set and train a new classifier using as positive features the descriptors from the track selected by multi-fold MIL, restricted to the ground-truth temporal extent. At test time, we perform a sliding window with the same temporal lengths as in [Weinzaepfel et al., 2015a] ($\{20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 300, 450, 600\}$ frames) and the same stride (10 frames). In order to penalize short tubes, we score each tube using the SVM score minus α/L where α is a parameter experimentally set to 20 and L is the length of the detection.

For the DALY dataset, we assume temporal supervision during training, *i.e.*, the start and end time of each action. For training, we thus extract human tubes only for the temporal extent of the actions. We then run multi-fold MIL to select the relevant tubes and train the detector. Additional temporal localization is beyond the scope of this manuscript and would require an additional step for pre-selecting relevant shots. At test time, we do not suppose temporal localization given and perform spatio-temporal localization on the entire test set. We first divide the videos into shots using

Learning	UCF-Sports	
	CorLoc@0.5 (training set)	mAP@0.5 (test set)
Initialization	80.14%	-
MIL ($K = 1$)	80.14%	83.38%
multi-fold MIL ($K = 4$)	81.99%	83.87%

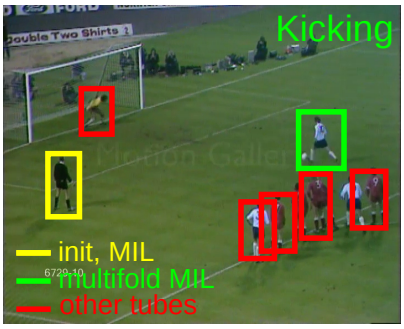


Figure 8.7 – *Left:* Comparison of multi-fold MIL to standard MIL on UCF-Sports. *Right:* an example where multi-fold MIL allows to select the correct tube. The yellow box is the tube with highest human score used as initialization. The location of the positive tube remains locked to this initialization when using MIL, while multi-fold MIL allows to move to the human performing the kicking action (green). Red boxes indicate other human tubes.

a shot detector [4]. We generate human tubes inside each shot and perform a temporal sliding window inside each tube. Since videos/actions are longer, we add $\{900, 1200, 1500, 1800, 2400, 3000, ..., 12000\}$ to the temporal scales used on UCF-101.

8.4.4 Evaluation of our MIL learning

Figure 8.7 shows an evaluation of our multi-fold MIL algorithm. We report the mAP@0.5 on the test set, as well as the CorLoc@0.5 on the training set defined as the ratio of selected tubes (*i.e.*, the tube with highest score in each positive video) which have an IoU with any ground-truth over 0.5. When using MIL, the CorLoc remains constant: the locations of the positive tubes do not change. This can be explained by the small number of positive videos and the high dimensional features. Multi-fold MIL overcomes this limitation, the CorLoc@0.5 increases and so does the mAP. The difference is moderate, as in many cases only one human is present in the test videos. An example for an improved selection of the human performing the action is shown in Figure 8.7 (right).

4. https://github.com/johmathe/Shotdetect

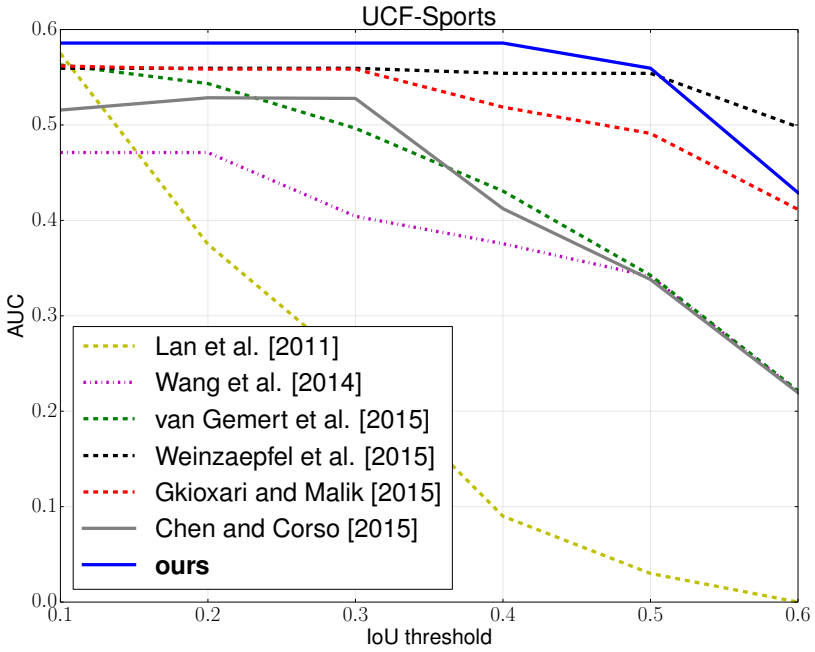


Figure 8.8 – Comparison of the AUC at various IoU threshold with the state of the art on the UCF-Sports dataset.

8.5 Experimental results

We have already assessed the quality of the human tubes in Section 8.3.3 and the impact of multi-fold MIL in Section 8.4.4. In this section, we evaluate the overall performance of our weakly-supervised action localization approach and compare to the state-of-the-art weakly- and fully-supervised approaches (Section 8.5.1). We then propose an evaluation of our approach on the DALY dataset (Section 8.5.2).

8.5.1 Comparison to the state of the art

UCF-Sports. We first evaluate our method and compare to the state of the art on UCF-Sports. Most of previous works plot ROC curves at different IoU thresholds until a false positive rate of 0.6 and report the Area Under the Curve (AUC). We show a comparison to the state of the art in Figure 8.8. Plain lines indicate weakly-supervised methods and dashed ones full supervision. Our method outperforms all approaches based on this metric, even if they leverage supervision. We beat by a large margin the only other weakly supervised approach [Chen and Corso, 2015] which reports results for this metric. Interestingly, our performance decreases only

	ours	[Mosabbeb et al., 2014]	[Ma et al., 2013]
Diving	56.2	43.7	44.3
Golf	50.6	52.3	50.5
Kicking	68.7	52.9	48.3
Lifting	62.4	63.5	41.4
Riding	63.5	32.5	30.6
Run	57.2	30.1	33.1
Skateboard	69.1	43.2	38.5
Swing1	64.3	57.5	54.3
Swing2	68.6	44.1	20.6
Walkg	68.3	47.1	39.0
avg.	**62.9**	46.7	41.0

Table 8.2 – Comparison of the mean-IOU with the state of the art on the UCF-Sports dataset.

Method	Supervision	UCF-Sports	J-HMDB	UCF-101		DALY
		mAP@0.5	mAP@0.5	mAP@0.05	mAP@0.2	mAP@0.2
Gkioxari and Malik [2015]	fully	75.8%	53.3%	-	-	-
Weinzaepfel et al. [2015a]	fully	90.5%	60.7%	54.3%	46.8%	-
Weinzaepfel et al. [2015a]•	fully	95.1%	66.8%	69.7%	60.6%	-
van Gemert et al. [2015]	fully	-	-	58.0%	37.8%	-
Yu and Yuan [2015]	fully	-	-	42.8%	-	-
ours	weakly	83.9%	54.1%	62.8%	45.4%	10.8%

Table 8.3 – Comparison to the state of the art with mAP@0.5 on the UCF-Sports and J-HMDB datasets and mAP@0.2 on the UCF-101 (split 1) and DALY datasets. For UCF-101 we also report mAP@0.05 to compare to Yu and Yuan [2015]. Weinzaepfel et al. [2015a]• refers to the version with IDT instead of STMH features.

slowly with respect to the IoU threshold. This indicates the high IoU of our detections with respect to the ground-truth, thanks to accurate human detection and tracking. We have also evaluated the mean-IoU, which is the average IoU between the best detection in a test video and the ground-truth tube. A comparison with other weakly-supervised approaches [Mosabbeb et al., 2014, Ma et al., 2013] reporting this metric is shown in Table 8.2. We outperform them by more than 15%. Finally, we compare with the more meaningful mAP metric in Table 8.3. We obtain a mAP of 84% which is close to the state-of-the-art fully-supervised method [Weinzaepfel et al., 2015a] and better than Gkioxari and Malik [2015], despite the fact that we use significantly less supervision.

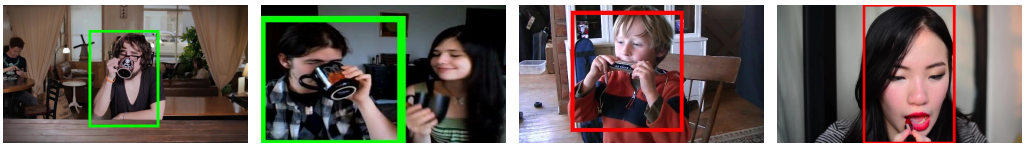


Figure 8.9 – Example of highly-ranked drinking detections on the DALY dataset. The two left examples are correct, whereas the two examples on the right correspond to confusions with playing harmonica and applying make up on lips.

J-HMDB. Mean Average-Precision (mAP) on the J-HMDB dataset is shown in Table 8.3. Similar to the UCF-Sports dataset, our method is in between the two supervised methods from Weinzaepfel et al. [2015a] and Gkioxari and Malik [Gkioxari and Malik, 2015]. Since the J-HMDB dataset contains only one human per video, the main challenge actually consists in classifying the tubes. Thus, a tube description that also leverages CNN features, in the same spirit as Gkioxari and Malik [2015] and Weinzaepfel et al. [2015a] would probably increase the performance.

UCF-101. We report results with the standard mAP@0.2 metric for this dataset in Table 8.3. Our method also obtains a performance slightly below the state-of-the-art fully-supervised method [Weinzaepfel et al., 2015a]. We perform better than the 37.8% mAP@0.2 reported by van Gemert et al. [2015], which extract thousands of proposals before using supervision for learning a classifier. Yu and Yuan [2015] report a mAP@0.05 of 42.8% compared to our 62.8% at this threshold. At this low overlap threshold, we obtain better results than all other fully-supervised approaches, except [Weinzaepfel et al., 2015a] with IDT features presented in Chapter 7. This can be explained by the fact that our approach obtains less false positive detections due to the quality of our human tubes. Nevertheless, the IoU between our detections and the ground-truth tubes are often below 0.2, due to the low quality of the videos which makes the human tube extraction challenging.

8.5.2 Evaluation on the DALY dataset

Spatio-temporal action localization. For spatio-temporal action localization, we obtain a mAP@0.2 of 10.8% and a mAP@0.1 of 15.8%, see Tables 8.3 and 8.4. These results are obtained by training in a spatially weakly-supervised setting. Note that the IoU is defined as the temporal IoU multiplied by the spatial IoU obtained by averaging over the annotated

	mAP@0.1	mAP@0.2
clip classification	72.9%	
action localization in trimmed clips	49.7%	48.6%
action localization in shots with actions	23.5%	16.1%
action localization	15.8%	10.8%

Table 8.4 – Detailed performance analysis on the DALY dataset. See text for more details.

keyframes, since spatial annotations are not available at every frame. The relatively low performance can be explained by several factors. First, temporal detection is difficult as actions are short in long untrimmed videos. Moreover, there are many actions that are close in time and our method tends to return one detection that covers multiple ground-truth short actions. Second, the different labels have similarities, leading to confusion between classes, as shown in Figure 8.9. For instance, drinking, playing harmonica and applying make up on lips all involve poses where the hands come near the mouth. The relatively low performance highlights the difficulty of the introduced DALY dataset and demonstrates that significant improvements are necessary to tackle challenging real-world videos.

In order to tease apart the challenges of the DALY dataset, we separately evaluate class confusion, temporal detection and spatio-temporal detection, see Table 8.4.

Clip classification. We first evaluate the performance of clip classification, *i.e.* of video clips trimmed to the duration of the annotated actions. Each clip is represented by a Fisher Vector of improved dense trajectories and a linear SVM is learned for clip classification. When learning a linear SVM for one class, we do not use the clips from other actions that temporally overlap with this class. For evaluation, we use the mAP metric since one clip can have multiple labels. As during training, we remove the clips from other actions that temporally overlap with this class during evaluation. We report a mAP of 73%. This is lower than the IDT+FV results for video classification on UCF-Sports (88%) and UCF-101 (88%), and in the same order of magnitude on J-HMDB (65%). Distinguishing the classes is not straightforward since many classes contain similar motion patterns (hand moving near the head).

Action localization in trimmed clips. We also evaluate action localization in these trimmed clips. This setting is similar to action localization

datasets such as J-HMDB or UCF-Sports where clips are trimmed to the action. We obtain a mAP@0.2 of 48.6% and a mAP@0.5 of 31.7%. This relatively low performance can be explained by the fact that many instances are short and can easily be confused.

Action localization in shots containing actions. We run the spatio-temporal localization only in shots that contain an action. This removes, compared to the standard setting, the incorrect detections in shots that do not contain any action, and reduces the total duration from 31 to 21 hours. The mAP@0.2 decreases to 16.1% when adding this temporal component, but it is above the result with all shots (10.8%). Once again, the challenge is to obtain a precise temporal localization without fragmenting long actions into multiple small temporal parts, which leads to false detections. Note that this is significantly different from the UCF-101 dataset in which videos are almost trimmed. Here, the shots can last for multiple minutes while the actions only last several seconds.

8.6 Conclusion

In this chapter, we introduced a novel approach for extracting human tubes that outperforms state-of-the-art proposals by a large margin. We showed that they can be used effectively for weakly-supervised action localization, with a performance close to fully-supervised approaches. We also introduced a new challenging dataset, DALY, that overcomes the limitations of the existing benchmarks and will allow to measure progress in the field over the next few years.

Chapter 9

Conclusion

Contents

In this thesis, we focused on two tasks related to video understanding: optical flow estimation in realistic videos, *i.e.*, with fast non-rigid motion, and human action localization in uncontrolled videos. This chapter is organized as follows. We summarize the contributions of the thesis on optical flow estimation in Section 9.1.1 and on action localization in Section 9.1.2. We then conclude this dissertation with directions for future research, on optical flow estimation (Section 9.2.1) and on action localization (Section 9.2.2).

9.1 Summary of contributions

9.1.1 Optical flow estimation

We have introduced several contributions to improve optical flow estimation near motion boundaries and in presence of large displacements and occlusions. We have proposed DeepFlow in Chapter 3, a variational model that integrates a novel matching algorithm, robust to non-rigid motions and repetitive textures. In addition, we have introduced EpicFlow in Chapter 4, a novel scheme for optical flow estimation based on a sparse-to-dense inter-

polation of matches while respecting edges. Finally, we have shown that a learning-based method is able to predict motion boundaries, see Chapter 5. We now present a summary for each of these contributions.

DeepFlow: large displacement optical flow with DeepMatching. DeepFlow is a variational approach for optical flow estimation that integrates a novel matching algorithm, called DeepMatching. DeepMatching is based on correlations between image patches and relies on a hierarchical multi-layer architecture. The architecture is inspired by deep convolutional neural network approaches, with interleaved pooling and aggregation. DeepMatching efficiently handles non-rigid deformations, repetitive textures and outputs quasi-dense correspondences, even in cases of significant change between images. Experiments show that the matching algorithm performs well on matching benchmarks, particularly in terms of accuracy and coverage. We have also proposed an approximation scheme to save computational time and memory without significant loss in performance. In addition, we have shown that DeepMatching is well suited for optical flow by integrating it into a variational formulation. The resulting optical flow approach obtains competitive performance on various benchmarks thanks to the accuracy and coverage of DeepMatching. In particular, it significantly improves the case of large displacements compared to previous methods.

DeepMatching and DeepFlow have been used in several recent works. For instance, Timofte and Van Gool [2015] use DeepFlow with a combination of DeepMatching and a new matching algorithm based on sparse decomposition of local patches to improve optical flow. DeepMatching has been used in other computer vision tasks such as tracking faces [Aghaei et al., 2015], in which it is used as a similarity score, and SLAM (Simultaneous Localization and Mapping) [Chhaya et al., 2016], for which it is used as a component for tracking dynamic objects throughout a sequence. DeepFlow has been used in many applications of computer vision such as pose estimation in videos [Jain et al., 2014a, Pfister et al., 2015], SLAM [Reddy et al., 2015], or object reconstruction from videos [Lebeda et al., 2015]. It has also been used for style transfer [Ruder et al., 2016] in order to ensure the consistency of the transfer throughout the video.

EpicFlow: edge-preserving interpolation of correspondences for optical flow. EpicFlow operates by densifying quasi-dense input matches, and leverages an edge-aware geodesic distance tailored to respect motion boundaries and to handle occlusions. The interpolation step outputs an

optical flow estimation, which can be further refined using a variational formulation. We have also introduced an approximation scheme for the interpolation that significantly speeds up the computation without harming the performance. Experimental results justify the choice of the proposed edge-aware geodesic distance and its approximation. They also show that sparse-to-dense interpolation overcomes the limitations of coarse-to-fine schemes, such as error propagation across pyramid levels or oversmoothing. At publication time, EpicFlow obtained best performance on the MPI-Sintel dataset.

EpicFlow with its interpolation scheme has recently been used in many optical flow approaches which improve the input matches [Bailer et al., 2015, Menze et al., 2015, Chen and Koltun, 2016]. Recent state-of-the-art methods [Menze et al., 2015, Chen and Koltun, 2016] are based on a pixel-accurate flow estimation obtained by solving discrete optimization problem. EpicFlow is then used to refine the estimation. EpicFlow is also used in various applications such as oversegmentation of videos [Khoreva et al., 2016] or unsupervised learning of edges [Li et al., 2016]. This latter approach proposes to compute motion edges from a color-coded representation of the EpicFlow estimation, in order to learn an edge detector without supervision. Interestingly, this learned edge detector can be used to slightly improve the result of EpicFlow.

Learning to detect motion boundaries. We have proposed a learning-based approach for detecting motion boundaries in videos. The method is based on the structured random forest framework. A forest of multiple trees is trained to predict the motion boundaries of small patches. A soft response map is obtained by averaging the results from all trees and overlapping patches at each pixel. In addition, we introduce the YouTube Motion Boundaries (YMB) dataset which consists of 60 challenging video sequences with motion boundaries, with a central frame annotated by 3 people. Experiments show that the proposed approach performs significantly better than the widely used flow gradient baseline, both on high quality data from optical flow benchmarks and on highly compressed videos from YouTube.

Our approach has been used for tracking [Hua et al., 2015], in which motion boundaries are combined with edges to extract proposals. It has also been used for discovering parts of articulated objects in videos [Del Pero et al., 2016], in which the motion boundaries are computed with our method, instead of the flow gradient baseline, in order to improve video segmentation into foreground and background [Papazoglou and Ferrari, 2013].

9.1.2 Human action localization

We have introduced novel approaches for action localization in videos, which extract action-specific or human-specific tubes based on detection and tracking. In Chapter 7, we have proposed a method leveraging a class-specific detector and tracker based on CNNs. In Chapter 8, we have introduced a weakly-supervised method, *i.e.*, which does not require spatial supervision, leveraging a human detector together with a human-specific tracker. We now present a summary for each of these contributions.

Action-specific tracks for action localization. We have proposed a novel action localization method based on class-specific tubes. These tubes are extracted using a detector and a tracking-by-detection approach based on CNN features. Each tube is then described using the CNN features as well as spatio-temporal local features. We have compared improved dense trajectories (IDT) using Fisher Vector aggregation, with an introduced descriptor that describes a chunk from a track. Temporal detection is performed using a sliding window scheme inside each tube. Our approach outperforms the state of the art by a significant margin on three action localization benchmarks: UCF-Sports, J-HMDB and UCF-101. Experiments highlight the benefits of tracking compared to an approach in which the per-frame detections are linked throughout the video. They also show that local trajectory features such as IDT are well suited for describing tubes and are complementary to CNN features.

Human tubes for weakly-supervised action localization. The previous approach requires bounding box annotations in every frame of the training videos in order to train the class-specific detector. Moreover, the tracking is performed independently for each class. In order to scale to a higher number of classes, we have introduced a weakly-supervised approach leveraging existing human annotations available today. More precisely, the first step of the method consists in extracting tubes around the humans. To this end, a human detector is trained on a human pose database. Thanks to recent advances in deep learning for object detection, this human detector is robust to unusual poses and occlusions. We then run a human-specific tracker in order to obtain human tubes. Experiments highlight the benefit of these human tubes: a high recall is reached with only several tubes. Multiple Instance Learning may then be applied effectively. The tubes are described using Fisher Vector on improved dense trajectories. A multi-fold variant of MIL is necessary with this high-dimensional representation. Our approach outperforms other weakly-supervised methods and is close to the

best fully-supervised approaches on existing benchmarks. In order to validate our approach on more realistic data, we have introduced the DALY (Daily Action Localization in YouTube) dataset. DALY consists of more than 31 hours of YouTube videos with spatio-temporal annotations for 10 daily action classes. The diversity and the duration of the videos are unprecedented in existing spatio-temporal action localization benchmarks. We obtain 10.8% mAP on DALY with temporal supervision only, *i.e.*, without spatial supervision. This dataset will allow to measure progress in the field over the next few years.

9.2 Perspectives for future research

In this section, we propose directions for future research based on the experiments presented in this thesis as well as recent advances in the field of computer vision and machine learning.

9.2.1 Optical flow estimation

Improved variational modeling. Our variational model used in DeepFlow and EpicFlow can benefit from recent advances in optical flow estimation. For instance, the regularization assumes a constant flow by penalizing its gradient. Nevertheless, this assumption is often violated, such as on the Kitti dataset in which motion can be mainly modeled by homographies near image borders. Recently, the Total Generalized Variation framework has been applied to optical flow [Ranftl et al., 2014], and models piece-wise affine solutions. In addition, our variational model does not integrate robust detection and modeling of the occlusions [Xu et al., 2012, Kennedy and Taylor, 2015, Fortun et al., 2015, Leordeanu et al., 2013]. Occlusions can be detected by solving a binary classification problem [Leordeanu et al., 2013] based on various features such as forward-backward consistency, errors in the data term [Sun et al., 2014b] or mapping uniqueness criterion [Xu et al., 2012], *i.e.*, counting the number of reference pixels mapped to a particular position by the flow. Indeed, if the optical flow estimation makes multiple pixels moving to the same location, then most of them (all of them except one) correspond to occluded pixels. To integrate the occlusions into the model, a standard approach [Xu et al., 2012] consists in canceling the data term at these locations. Finally, the data term of our variational model assumes color and gradient constancy. Recent works [Vogel et al., 2013a, Stein, 2004a] have shown the effectiveness of the Census Transform (and its

differentiable approximation) for optical flow estimation, in particular for outdoor scenes.

Structured hierarchical matching. DeepMatching leverages a multi-scale hierarchical correlational architecture. However, the structure of the patches is fixed to be a square, itself composed of four squared quadrants. In particular, issues appear at strong motion boundaries for which the non-rigidity handled by DeepMatching is insufficient. One possible solution consists in weighting the four quadrants differently. This can be based for example on a precomputed segmentation. In the same spirit, Sevilla-Lara et al. [2016] propose to use semantic segmentation the context of optical flow by. Furthermore, in DeepMatching, the parameters of the first correlational layer are given by the first image. More recently, deep learning techniques, in which the parameters are learned from huge training data, have been applied to optical flow [Dosovitskiy et al., 2015, Teney and Hebert, 2016]. In particular, for DeepMatching, the descriptor of the atomic patches is hand-crafted and could be learned instead.

Structured interpolation. In EpicFlow, the interpolation only relies on a soft boundary map (edges). Consequently, matches from another region might still have a (slight) impact on the estimation. Hard constraints from a precomputed segmentation can be added to EpicFlow by setting to infinity the weights of the soft boundary map at segmentation boundaries. One other limitation of EpicFlow concerns non-contiguous regions: if a large region is split (due to occlusions from foreground objects), the motion is independently estimated for each of the part. However, it would make sense to correlate the interpolation of all these parts. Such complex structured interpolation will require designing an appropriate distance to replace the geodesic one. In the same spirit, Drayer and Brox [2015] use a discrete regularization in which non-contiguous regions are connected if the distance between their color histograms is small.

Discrete optimization. DeepFlow and EpicFlow have shown that state-of-the-art performance can be obtained from a quasi-dense set of matches. Many recent works now focus on obtaining such matches, and use EpicFlow to estimate dense optical flow from them. As a consequence, discrete optimization techniques [Menze et al., 2015, Chen and Koltun, 2016] have recently shown promising results. Sub-pixel accuracy is then obtained using EpicFlow.

Iterative estimation. The estimation of matches and boundaries can be further refined in an iterative scheme. In this scenario, the flow will be computed a first time. Matches and/or motion boundaries will then be reestimated and used for refining the flow estimation. This process can be repeated several times. For boundaries, this has been for instance applied to learn edges without supervision [Li et al., 2016]: motion edges are extracted from EpicFlow estimation and used as supervision for training a new edge detector. Edge maps are reestimated and used to update EpicFlow estimation. A small gain in performance is obtained for the flow estimation after several iterations. For matches, DeepMatching tends to miss tiny objects. This is mainly due to the scoring scheme which depends on scores of patches at different scales. At the same time, illustrations in Chapter 5 clearly show that regions with incorrect flow estimation can be easily identified. These errors are mainly due to occlusions or missing matches. Thus, DeepMatching can be run a second time, only in regions in which the flow estimation is considered as incorrect. In a similar spirit, Stoll et al. [2012] first estimate flow without matches, then identify regions with wrong estimation and recompute matches at these locations.

9.2.2 Human action localization

Temporal supervision. Our weakly-supervised method for action localization does not require spatial annotation but still needs temporal supervision. The next step consists in reducing or even removing the temporal supervision. Since actions typically last several seconds for videos of multiple minutes, relying on other sources of data is a more realistic scenario. First of all, one can assume a semi-supervised setting in which only several positive frames are annotated. For each action, a detector can then be learned with positives from this set of annotated frames and with negatives coming from videos that do not contain the action. A second solution consists in using webly-supervised data to learn a temporal detector. Effectiveness of using web data to recognize actions has recently been proven [Ma et al., 2015, Nguyen et al., 2016] and these works may be extended to the action localization task. For instance, [Ma et al., 2015] show that CNNs can be trained on downloaded images obtained by querying for the actions. [Nguyen et al., 2016] show that a dataset can be automatically built from a short-form video sharing service, namely Vine. These short videos (5-10 seconds) may also be used to learn an action detector, assuming that the actions cover the whole temporal extent of the video. Other level of supervision can consist in an ordered list of the actions occurring in training videos, as used by Bojanowski et al. [2014]. To fully remove the temporal

supervision, one can rely on ideas from co-segmentation in which similar patterns from the training videos are found. This can be for instance done using discriminative clustering, as proposed by Duchenne et al. [2009].

Temporal detection. Experimental results, in particular on the DALY dataset, show that many failure cases are due to inaccurate temporal detections: in particular, long actions tend to be split into multiple small detections. This can be explained by the fact that Fisher Vector encoding is not sufficiently discriminative for an accurate temporal detection of the actions inside each tube. One way of solving this problem may be to not only answer the question 'is the action occurring' but also 'is the action not occurring just before and after the temporal window'. Currently, the Fisher Vector aggregates trajectory features that are inside a tube and inside the temporal window. One way to implement the previous idea is to subtract to this aggregation the features from the trajectories that are inside the tube before and after the temporal window of interest. Another solution consists in relying on other features that might be more robust for finding the beginning and the end of the action. For instance, in a fully-supervised setting, Shou et al. [2016] first learn a CNN to classify chunks of videos, and then fine-tune the network in order to localize the action in time. More structured representations might also be helpful. For instance, Gaidon et al. [2013] represent actions as a flexible sequence of several sub-actions. Actions can also be temporally decomposed using a deformable part model, as shown by Niebles et al. [2010].

Modeling pose, objects and interactions. A richer representation and understanding of the action can be beneficial for action localization. For instance, incorporating the pose has shown significant gain for action recognition [Jhuang et al., 2013, Wang et al., 2014, Chéron et al., 2015]. The main issue is that pose estimator tends to fail in realistic videos, in particular in case of high compression and occlusion. Nevertheless, pose is an informative cue, especially for the temporal detection of an action since many actions last until a particular motion of the hand is performed. For instance, 'brushing teeth' lasts as long as the hand holding the toothbrush is moving at the mouth. In a similar spirit, other cues can be used such as the interactions between humans, or between actor and object. Such modeling has already been proposed in limited settings, for instance by Prest et al. [2012a], and should benefit from recent advances in detections and tracking for a generalization to more realistic data.

Incorporating segmentation. Another useful cue consists in segmenting the humans. In particular, this will allow to focus on features from trajectories that belong to the human, whereas boxes also contain background. Jhuang et al. [2013] have shown that the human segmentation helps the action recognition task. Human segmentation can be obtained without additional annotation: recent works [Papandreou et al., 2015, Kolesnikov and Lampert, 2016] show that reasonable segmentation performance can be obtained in a weakly-supervised setting. CNNs are learned using an estimation of the ground-truth segmentation based on the current estimate and priors such as the image or video labels. In addition, segmentation may help the detection when humans are partially occluded.

Publications

This thesis has led to several publications summarized below.

International Conferences

- P. Weinzaepfel, J. Revaud, Z. Harchaoui, C. Schmid.
  DeepFlow: Large displacement optical flow with DeepMatching.
  Proceedings of the IEEE International Conference on Computer Vision (ICCV) 2013.

- J. Revaud, P. Weinzaepfel, Z. Harchaoui, C. Schmid.
  EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow.
  Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015.

- P. Weinzaepfel, J. Revaud, Z. Harchaoui, C. Schmid.
  Learning to Detect Motion Boundaries.
  Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015.

- P. Weinzaepfel, Z. Harchaoui, C. Schmid.
  Learning to Track for Spatio-Temporal Action Localization.
  Proceedings of the IEEE International Conference on Computer Vision (ICCV) 2015.

International Journal

- J. Revaud, P. Weinzaepfel, Z. Harchaoui, C. Schmid.
  DeepMatching: Hierarchical Deformable Dense Matching.
  International Journal of Computer Vision (IJCV), to appear.

Other publication

- P. Weinzaepfel, X. Martin, C. Schmid.
  Towards Weakly-Supervised Action Localization.
  arXiv 2016.

Softwares and datasets

Several softwares and datasets from this thesis are available online.

Softwares

- Source code for computing DeepMatching on CPU and GPU.
- Source code for computing DeepFlow.
- Source code for computing EpicFlow.
- Source code for detecting motion boundaries using structured random forest.

Datasets

- YouTube Motion Boundary (YMB) dataset.
- Daily Action Localization in YouTube (DALY) dataset.

Bibliography

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. PAMI*, 2012. 30

G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. PAMI*, 1985. 25

J. Aggarwal and M. Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 2011. 118

M. Aghaei, M. Dimiccoli, and P. Radeva. Multi-face tracking by extended bag-of-tracklets in egocentric videos. *Computer Vision and Image Understanding*, 2015. 167

P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *IJCV*, 1989. 22, 25

P. Anandan and R. Weiss. Introducing a smoothness constraint in a matching approach for the computation of displacement fields. In *Image Understanding Workshop*, 1985. 25

M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014. 147, 149, 152

P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. PAMI*, 2011. 89, 91, 99

C. Bailer, B. Taetz, and D. Stricker. Flow Fields: Dense Correspondence Fields for Highly Accurate Large Displacement Optical Flow Estimation. In *ICCV*, 2015. 74, 75, 93, 94, 168

S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *IJCV*, 2004. 21, 27

S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011. 8, 32, 77

C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized PatchMatch correspondence algorithm. In *ECCV*, 2010. 29, 30, 36, 37, 63, 77, 86

H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006. 120

P. R. Beaudet. Rotationally invariant image operators. In *International Joint Conference on Pattern Recognition*, 1978. 119

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2009. 45

S. T. Birchfield. *Depth and motion discontinuities*. PhD thesis, Stanford University, 1999. 97

M. J. Black. Robust dynamic motion estimation over time. In *CVPR*, 1991. 25

M. J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 1996. 8, 23, 24, 25, 27

M. J. Black and D. J. Fleet. Probabilistic detection and tracking of motion boundaries. *IJCV*, 2000. 97, 98

A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Trans. PAMI*, 2001. 8

P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. In *ECCV*, 2014. 122, 172

H. Boyraz, S. Z. Masood, B. Liu, M. Tappen, and H. Foroosh. Action recognition by weakly-supervised discriminative region localization. In *BMVC*, 2014. 124

M. Brand. Shadow puppetry. In *ICCV*, 1999. 8

J. Braux-Zin, R. Dupont, and A. Bartoli. A general dense image matching framework combining direct and feature-based costs. In *ICCV*, 2013. 30, 74, 75, 94

T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. 8, 119

T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Trans. PAMI*, 2011. 4, 8, 10, 29, 30, 35, 36, 37, 54, 62, 73, 74, 76, 78, 90, 91, 93, 105, 110, 204

T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004. 8, 22, 23, 24, 25, 26, 27, 56, 85, 102, 132, 198

T. Brox, A. Bruhn, and J. Weickert. Variational motion segmentation with level sets. In *ECCV*, 2006. 27, 99

A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Variational optical flow computation in real time. *IEEE Trans. on Image Processing*, 2005a. 8, 55

A. Bruhn, J. Weickert, and C. Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *IJCV*, 2005b. 25, 26

A. Bruhn, J. Weickert, T. Kohlberger, and C. Schnörr. A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. *IJCV*, 2006. 25

P. J. Burt, C. Yen, and X. Xu. Multiresolution flow-through motion analysis. In *CVPR*, 1983. 22

D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 3, 4, 8, 9, 11, 32, 77, 97, 110, 113

L. W. Campbell, D. A. Becker, A. Azarbayejani, A. F. Bobick, and A. Pentland. Invariant features for 3-d gesture recognition. In *International Conference and Workshops on Automatic Face and Gesture Recognition*, 1996. 8

J. Canny. A computational approach to edge detection. *IEEE Trans. PAMI*, 1986. 89, 91

L. Cao, Z. Liu, and T. S. Huang. Cross-dataset action detection. In *CVPR*, 2010. 10, 123, 126

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 121

Q. Chen and V. Koltun. Full flow: Optical flow estimation by global optimization over regular grids. In *CVPR*, 2016. 29, 168, 171

W. Chen and J. J. Corso. Action detection by implicit intentional motion clustering. In *ICCV*, 2015. 124, 149, 161

W. Chen, C. Xiong, R. Xu, and J. Corso. Actionness ranking with lattice conditional ordinal random fields. In *CVPR*, 2014. 123

Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. Large displacement optical flow from nearest neighbor fields. In *CVPR*, 2013. 30

G. Chéron, I. Laptev, and C. Schmid. P-cnn: pose-based cnn features for action recognition. In *ICCV*, 2015. 173

F. Chhaya, N. D. Reddy, S. Upadhyay, V. Chari, M. Z. Zia, and K. M. Krishna. Monocular reconstruction of vehicles: Combining slam with shape priors. In *International Conference on Robotics and Automation (ICRA)*, 2016. 167

O. Chomat and J. L. Crowley. Probabilistic recognition of activity using local appearance. In *CVPR*, 1999. 8

R. G. Cinbis, J. Verbeek, and C. Schmid. Weakly Supervised Object Localization with Multi-fold Multiple Instance Learning. *IEEE Trans. PAMI*, 2016. 147, 158

A. Criminisi, T. Sharp, C. Rother, and P. Pérez. Geodesic image and video editing. *ACM Trans. Graph.*, 2010. 82

G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, 2004. 120

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, June 2005. 30, 36, 119

T. Darrell and A. Pentland. Cooperative robust estimation using layers of support. *IEEE Trans. PAMI*, 1995. 27, 99

L. Del Pero, S. Ricco, R. Sukthankar, and V. Ferrari. Discovering the physical parts of an articulated object class from multiple videos. In *CVPR*, 2016. 168

O. Demetz, M. Stoll, S. Volz, J. Weickert, and A. Bruhn. Learning brightness transfer functions for the joint recovery of illumination changes and optical flow. In *ECCV*, 2014. 74, 75, 94

P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 12, 77, 79, 82, 84, 89, 91, 98, 99, 101, 107, 113

P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005. 8

J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 121

A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 28, 171

B. Drayer and T. Brox. Combinatorial regularization of descriptor matching for optical flow estimation. In *BMVC*, 2015. 171

O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009. 122, 173

A. Ecker and S. Ullman. A hierarchical non-parametric method for capturing non-rigid deformations. *Image and Vision Computing*, 2009. 37

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL VOC Challenge 2011, 2011. 127

G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian conference on Image analysis*, 2003. 110, 120

P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. PAMI*, 2010. 10, 122

C. L. Fennema and W. B. Thompson. Velocity determination in scenes containing several moving objects. *Computer Graphics and Image Processing*, 1979. 7

B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015. 8

D. J. Fleet, M. J. Black, Y. Yacoob, and A. D. Jepson. Design and use of linear models for image motion analysis. *IJCV*, 2000. 98

L. Fletcher, L. Petersson, and A. Zelinsky. Driver assistance systems based on vision in and out of vehicles. In *Intelligent Vehicles Symposium*, 2003. 8

D. Fortun, P. Bouthemy, and C. Kervrann. Aggregation of local parametric candidates with exemplar-based occlusion handling for optical flow. *Computer Vision and Image Understanding*, 2015. 170

Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *CVPR*, 2010. 29

A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal Localization of Actions with Actoms. *IEEE Trans. PAMI*, 2013. 8, 121, 122, 173

J. Gall, N. Razavi, and L. Van Gool. On-line adaption of class-specific codebooks for instance tracking. In *BMVC*, 2010. 135

A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *IJRR*, 2013. 8, 11, 32, 77

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 121, 123, 131, 132

A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. In *ICIP*, 2013. 134

G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 123, 129, 130, 132, 139, 142, 143, 147, 149, 162, 163

P. Golland and A. M. Bruckstein. Motion from color. *Computer Vision and Image Understanding*, 1997. 22

L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE Trans. PAMI*, 2007. 8

Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski. Non-rigid dense correspondence with applications for image enhancement. *SIGGRAPH*, 2011. 30, 53, 57, 63

D. Hafner, O. Demetz, and J. Weickert. *Scale Space and Variational Methods in Computer Vision*, chapter Why Is the Census Transform Good for Robust Optic Flow Computation? Springer, 2013. 22

S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. 129, 147, 149

C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, 1988. 119

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2003. 81

T. Hassner, V. Mayzels, and L. Zelnik-Manor. On sifts and their scales. In *CVPR*, 2012. 30, 65

K. He and J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *CVPR*, 2012. 79, 80, 86

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 121

S. Herath, M. Harandi, and F. Porikli. Going deeper into action recognition: A survey. *arXiv preprint arXiv:1605.04988*, 2016. 118

M. Hoai, L. Torresani, F. De la Torre, and C. Rother. Learning discriminative localization from weakly labeled data. *Pattern Recognition*, 2014. 122

D. Hoiem, A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. *IJCV*, 2011. 99

B. K. P. Horn and B. G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 1981. 8, 19, 21, 22, 23, 24, 25, 26, 35

J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *IEEE Trans. PAMI*, 2015. 130

Y. Hua, K. Alahari, and C. Schmid. Occlusion and Motion Reasoning for Long-term Tracking. In *ECCV*, 2014. 134

Y. Hua, K. Alahari, and C. Schmid. Online object tracking with proposal selection. In *ICCV*, 2015. 168

A. Humayun, O. Mac Aodha, and G. J. Brostow. Learning to Find Occlusion Regions. In *CVPR*, 2011. 99

N. Ikizler-Cinbis, R. G. Cinbis, and S. Sclaroff. Learning actions from the web. In *ICCV*, 2009. 119

A. Jain, J. Tompson, Y. LeCun, and C. Bregler. Modeep: A deep learning framework using motion features for human pose estimation. In *ACCV*, 2014a. 167

M. Jain, J. Gemert, H. Jégou, P. Bouthemy, and C. Snoek. Action localization with tubelets from motion. In *CVPR*, 2014b. 10, 123, 149

R. Jain and H.-H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Trans. PAMI*, 1979. 7

R. Jain, D. Militzer, and H.-H. Nagel. *Separating non-stationary from stationary scene components in a sequence of real world TV-images*. Institut für Informatik, Universität Hamburg, 1977. 7

H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Trans. PAMI*, 2012. 120

H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013. 3, 123, 125, 173, 174

S. Ji, W. Xu, M. Yang, and K. Yu. 3D convolutional neural networks for human action recognition. *IEEE Trans. PAMI*, 2013. 121

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 62, 133

Z. Kalal, K. Mikolajczyk, and J. Matas. Face-TLD: Tracking-Learning-Detection Applied to Faces. In *ICIP*, 2010. 135

Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Trans. PAMI*, 2012. 129, 147, 149

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 105, 106, 121

R. Kennedy and C. J. Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In *EMMCVPR*, 2015. 74, 75, 93, 94, 170

D. Keysers, T. Deselaers, C. Gollan, and H. Ney. Deformation models for image recognition. *IEEE Trans. PAMI*, 2007. 37

A. Khoreva, R. Benenson, F. Galasso, M. Hein, and B. Schiele. Improved image boundaries for better video segmentation. *arXiv preprint arXiv:1605.03718*, 2016. 168

J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *CVPR*, 2013. 30, 65

A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008. 8, 119, 123, 129, 148

A. Kläser, M. Marszalek, C. Schmid, and A. Zisserman. Human Focused Action Localization in Video. In *International Workshop on Sign, Gesture, and Activity (SGA)*, 2010. 10, 123, 129, 147, 148

A. Kolesnikov and C. H. Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *ECCV*, 2016. 174

S. Korman and S. Avidan. Coherency sensitive hashing. In *ICCV*, 2011. 29, 37

P. Krähenbühl and V. Koltun. Geodesic object proposals. In *ECCV*, 2014. 82

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 2012. 121

H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 126

C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Trans. PAMI*, 2009. 122

T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *ICCV*, 2011. 10, 123, 125

T. Lan, Y. Zhu, A. Roshan Zamir, and S. Savarese. Action recognition by hierarchical mid-level action elements. In *ICCV*, 2015. 124

I. Laptev. On space-time interest points. *IJCV*, 2005. 2, 8, 119, 124, 149

I. Laptev and P. Pérez. Retrieving actions in movies. In *ICCV*, 2007. 10, 123, 147

I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 8, 119, 121

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 121

K. Lebeda, S. Hadfield, and R. Bowden. Dense rigid reconstruction from unstructured discontinuous video. In *ICCV*, 2015. 167

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998a. 37, 45, 51

Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In *Neural Networks: Tricks of the trade*, 1998b. 45

V. S. Lempitsky, S. Roth, and C. Rother. Fusionflow: Discrete-continuous optimization for optical flow estimation. In *CVPR*, 2008. 28, 30

M. Leordeanu, A. Zanfir, and C. Sminchisescu. Locally affine sparse-to-dense matching for motion and occlusion estimation. In *ICCV*, 2013. 72, 74, 77, 79, 84, 89, 93, 170

J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*, 2011. 119

Y. Li, M. Paluri, J. M. Rehg, and P. Dollár. Unsupervised learning of edges. In *CVPR*, 2016. 168, 172

C. Liu, W. T. Freeman, and E. H. Adelson. Analysis of contour motions. In *Advances in Neural Information Processing Systems*, 2006. 99

C. Liu, J. Yuen, and A. Torralba. SIFT flow: Dense correspondence across scenes and its applications. *IEEE Trans. PAMI*, 2011. 22, 30, 36, 65

J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild". In *CVPR*, 2009. 119

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 29, 30, 38, 41, 53, 62

J. Lu, H. Yang, D. Min, and M. Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *CVPR*, 2013. 30

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981. 19, 20, 25, 120

S. Ma, J. Zhang, N. Ikizler-Cinbis, and S. Sclaroff. Action recognition and localization by hierarchical space-time segments. In *ICCV*, 2013. 124, 162

S. Ma, S. A. Bargal, J. Zhang, L. Sigal, and S. Sclaroff. Do less and achieve more: Training cnns for action recognition utilizing action images from the web. *arXiv preprint arXiv:1512.07155*, 2015. 172

Y. Mae, Y. Shirai, J. Miura, and Y. Kuno. Object tracking in cluttered background based on optical flow and edges. In *ICVPR*, 1996. 8

J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America A: Optics, Image Science, and Vision*, 1990. 45

M. Marian Puscas, E. Sangineto, D. Culibrk, and N. Sebe. Unsupervised tube extraction using transductive learning and dense trajectories. In *ICCV*, 2015. 10, 123, 147, 149

V. Markandey and B. E. Flinchbaugh. Multispectral constraints for optical flow computation. In *ICCV*, 1990. 22

M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, 2009. 119

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 106

W. N. Martin and J. Aggarwal. Dynamic scene analysis: The study of moving images. Technical report, DTIC Document, 1977. 7

P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *ICCV Workshops*, 2009. 120

M. Menze, C. Heipke, and A. Geiger. Discrete Optimization for Optical Flow. In *GCPR*, 2015. 28, 74, 75, 93, 94, 168, 171

P. Mettes, J. C. van Gemert, and C. G. Snoek. Spot on: Action localization from pointly-supervised proposals. In *ECCV*, 2016. 149

M. Middendorf and H. Nagel. Estimation and interpretation of discontinuities in optical flow fields. In *ICCV*, 2001. 97

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *IJCV*, 2005. 11, 29, 53, 57

T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and Image Understanding*, 2006. 10

E. A. Mosabbeb, R. Cabral, F. De la Torre, and M. Fathy. Multi-label discriminative weakly-supervised human activity recognition and localization. In *ACCV*, 2014. 124, 162

M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009. 62

T. Müller, C. Rabe, J. Rannacher, U. Franke, and R. Mester. Illumination-robust dense optical flow using census signatures. *Pattern Recognition*, 2011. 22

H. H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Trans. PAMI*, 1986. 25

K. Nakayama and J. Loomis. Optical velocity patterns, velocity-sensitive neurons, and space perception: a hypothesis. *Perception*, 1974. 7

S. Negahdaripour. Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. *IEEE Trans. PAMI*, 1998. 23

P. X. Nguyen, G. Rogez, C. Fowlkes, and D. Ramamnan. The open world of micro-videos. *arXiv preprint arXiv:1603.09439*, 2016. 172

J. C. Niebles, C.-W. Chen, , and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010. 119, 122, 173

T. Nir, A. M. Bruckstein, and R. Kimmel. Over-parameterized variational optical flow. *IJCV*, 2008. 25, 26

D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-Temporal Object Detection Proposals. In *ECCV*, 2014a. 10, 123, 149

D. Oneata, J. Verbeek, and C. Schmid. The LEAR submission at Thumos 2014, 2014b. URL https://hal.inria.fr/hal-01074442. 137

D. Oneata, J. Verbeek, and C. Schmid. Efficient Action Localization with Approximately Normalized Fisher Vectors. In *CVPR*, 2014c. 122

G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a dcnn for semantic image segmentation. In *ICCV*, 2015. 174

A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013. 8, 97, 98, 168

N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *IJCV*, 2006. 22, 23

A. Patron, M. Marszalek, A. Zisserman, and I. Reid. High five: Recognising human interactions in tv shows. In *BMVC*, 2010. 119

X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014. 8, 121

T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *ICCV*, 2015. 167

J. Philbin, M. Isard, J. Sivic, and A. Zisserman. Descriptor learning for efficient retrieval. In *ECCV*, 2010. 29

R. Poppe. A survey on vision-based human action recognition. *Image Vision Computing*, 2010. 118

A. Prest, V. Ferrari, and C. Schmid. Explicit modeling of human-object interactions in realistic videos. *IEEE Trans. PAMI*, 2012a. 123, 173

A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, 2012b. 105, 106

R. Ranftl, K. Bredies, and T. Pock. Non-local total generalized variation for optical flow estimation. In *ECCV*, 2014. 25, 74, 75, 93, 94, 170

N. D. Reddy, P. Singhal, V. Chari, and K. M. Krishna. Dynamic body vslam with semantic constraints. In *International Conference on Intelligent Robots and Systems (IROS)*, 2015. 167

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 147, 149, 151

X. Ren. Local grouping for optical flow. In *CVPR*, 2008. 79

J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *CVPR*, 2015. 12, 72, 73, 74, 75

J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. DeepMatching: Hierarchical Deformable Dense Matching. *IJCV*, 2016. 11, 12, 38, 76, 77, 79, 93, 94, 110

M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008. 119, 125

K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image understanding*, 1994. 8

S. Roth and M. J. Black. On the spatial statistics of optical flow. *IJCV*, 2007. 24, 28

M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. *arXiv preprint arXiv:1604.08610*, 2016. 167

A. Salgado and J. Sánchez. Temporal constraints in large optical flow estimation. In *Computer Aided Systems Theory–EUROCAST 2007*, pages 709–716. Springer, 2007. 25

J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 2013. 8, 120, 121, 136, 158

P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *IJCV*, 2008. 119

S. Satkin and M. Hebert. Modeling the temporal extent of actions. In *ECCV*, 2010. 122

C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *ICPR*, 2004. 8, 126

S. M. Seitz and S. Baker. Filter flow. In *ICCV*, 2009. 23, 24

P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using CNN. In *ICLR*, 2014. 134

L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *CVPR*, 2016. 171

N. Shapovalova, A. Vahdat, K. Cannons, T. Lan, and G. Mori. Similarity constrained latent support vector machine: An application to weakly supervised action classification. In *ECCV*, 2012. 124

J. Shin, S. Kim, S. Kang, S.-W. Lee, J. Paik, B. Abidi, and M. Abidi. Optical flow-based real-time object tracking using non-prior training active feature model. *Real-Time Imaging*, 2005. 8

Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, 2016. 173

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 8, 121, 132

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 121, 152

P. Siva and T. Xiang. Weakly supervised action detection. In *BMVC*, 2011. 124, 149

J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 8, 120

K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. In *CRCV-TR-12-01*, 2012. 126, 133

A. Spoerri. *The Early Detection of Motion Boundaries*. PhD thesis. Massachusetts Institute of Technology, Department of Brain and Cognitive Sciences, 1991. 97, 98

A. Stein and M. Hebert. Occlusion boundaries from motion: Low-level detection and mid-level reasoning. *IJCV*, 2009. 99

F. Stein. Efficient computation of optical flow using the census transform. *Pattern recognition*, 2004a. 170

F. Stein. Efficient Computation of Optical Flow Using the Census Transform. In *Proceedings of the 26th DAGM Symposium*, Lecture Notes in Computer Science, 2004b. 22

M. Stoll, S. Volz, and A. Bruhn. Adaptive integration of feature matches into variational optical flow methods. In *ACCV*, 2012. 54, 56, 172

D. Sun, S. Roth, J. P. Lewis, and M. J. Black. Learning optical flow. In *ECCV*, 2008. 28

D. Sun, E. B. Sudderth, and M. J. Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *NIPS*, 2010. 31

D. Sun, J. Wulff, E. Sudderth, H. Pfister, and M. Black. A fully-connected layered model of foreground and background flow. In *CVPR*, 2013. 27, 99

D. Sun, C. Liu, and H. Pfister. Local layering for joint motion estimation and occlusion detection. In *CVPR*, 2014a. 74

D. Sun, S. Roth, and M. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *IJCV*, 2014b. 8, 23, 25, 35, 54, 74, 93, 97, 108, 110, 111, 112, 170

J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *CVPR*, 2012. 29, 37, 63

Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection using optical sensors: A review. In *Intelligent Transportation Systems*, 2004. 8

N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, 2010. 27, 203

P. Sundberg, T. Brox, M. Maire, P. Arbelaez, and J. Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *CVPR*, 2011. 99

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 121

R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010. 29, 53

E. H. Taralova, F. De la Torre, and M. Hebert. Motion words for videos. In *ECCV*, 2014. 121

D. Teney and M. Hebert. Learning to extract motion from videos in convolutional neural networks. *arXiv preprint arXiv:1601.07532*, 2016. 28, 171

Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *CVPR*, 2013. 10, 123

R. Timofte and L. Van Gool. Sparse flow: Sparse matching for small to large displacement optical flow. In *Applications of Computer Vision (WACV)*, 2015. 72, 74, 167

E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *CVPR*, 2008. 8, 29

E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *IEEE Trans. PAMI*, 2010. 41, 65

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 121

W. Trobin, T. Pock, D. Cremers, and H. Bischof. An unbiased second-order prior for high-accuracy motion estimation. *Pattern Recognition*, 2008. 25, 27

S. Uchida and H. Sakoe. A monotonic and continuous two-dimensional warping based on dynamic programming. In *ICPR*, 1998. 37

H. Uemura, S. Ishikawa, and K. Mikolajczyk. Feature tracking and motion compensation for action recognition. In *BMVC*, 2008. 120

J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013. 123, 124, 129, 139

M. Unger, M. Werlberger, T. Pock, and H. Bischof. Joint motion estimation and segmentation of complex scenes with label costs and occlusion modeling. In *CVPR*, 2012. 27, 99

J. C. van Gemert, C. J. Veenman, A. W. Smeulders, and J.-M. Geusebroek. Visual word ambiguity. *IEEE Trans. PAMI*, 2010. 120

J. C. van Gemert, M. Jain, E. Gati, and C. G. Snoek. Apt: Action localization proposals from dense trajectories. In *BMVC*, 2015. 10, 123, 147, 149, 162, 163

G. Varol, Y. Laptev, and C. Schmid. Long-term Temporal Convolutions for Action Recognition. *arXiv preprint arXiv:1604.04494*, 2016. 3

C. Vogel, S. Roth, and K. Schindler. An evaluation of data costs for optical flow. In *GCPR*, 2013a. 8, 22, 23, 170

C. Vogel, K. Schindler, and S. Roth. Piecewise rigid scene flow. In *ICCV*, 2013b. 75, 94, 102

S. Volz, A. Bruhn, L. Valgaerts, and H. Zimmer. Modeling temporal coherence for optical flow. In *ICCV*, 2011. 25

H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 2013. 2, 3, 8, 97, 98, 119, 120, 131, 135, 136

H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. *IJCV*, 2015. 15, 120, 121, 129, 131, 136, 147, 156

J. Wang and E. Adelson. Representing moving images with layers. *IEEE Trans. Image Processing*, 1994. 27, 99

L. Wang, Y. Qiao, and X. Tang. Video action detection with relational dynamic-poselets. In *ECCV*, 2014. 123, 147, 173

L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2010. 80

O. Weber, Y. S. Devir, A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Trans. Graph.*, 2008. 82

A. Wedel, D. Cremers, T. Pock, and H. Bischof. Structure-and motion-adaptive regularization for high accuracy optic flow. In *ICCV*, 2009a. 24, 25, 55, 85

A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers. An improved algorithm for tv-l 1 optical flow. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, 2009b. 23, 24, 27

D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 2011. 118

P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 11, 38, 46, 48, 60, 72, 74, 75, 78, 79, 80, 82, 86, 87, 90, 91, 93

P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, 2015a. 14, 129, 147, 149, 159, 162, 163

P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Learning to Detect Motion Boundaries. In *CVPR*, 2015b. 13

P. Weinzaepfel, X. Martin, and C. Schmid. Towards Weakly-Supervised Action Localization. *arXiv preprint arXiv:1605.05197*, 2016. 3, 5, 6, 7, 16

M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *BMVC*, 2009. 8, 23, 25

G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, 2008. 119

J. Wills, S. Agarwal, and S. Belongie. A feature-based approach for dense segmentation and estimation of large disparity motion. *IJCV*, 2006. 29, 37

J. Wulff and M. J. Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *CVPR*, 2015. 28

C. Xu, S.-H. Hsieh, C. Xiong, and J. J. Corso. Can humans fly? action understanding with multiple classes of actors. In *CVPR*, 2015. 122

L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Trans. PAMI*, 2012. 8, 30, 35, 54, 55, 73, 74, 76, 78, 85, 90, 93, 170

H. Yang, W. Lin, and J. Lu. DAISY filter flow: A generalized discrete approach to dense correspondences. In *CVPR*, 2014. 29, 30, 37, 65

D. M. Young and W. Rheinboldt. *Iterative solution of large linear systems*. Academic Press, New York, NY, 1971. 27, 56, 85, 203

G. Yu and J. Yuan. Fast action proposals for human action detection and search. In *CVPR*, 2015. 123, 143, 144, 147, 149, 162, 163

J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, 2009. 10, 122

R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV*, 1994. 22

C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. *Pattern Recognition*, 2007. 110

X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010. 120

H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn, and H.-P. Seidel. Complementary optic flow. In *EMM-CVPR*, 2009. 22, 24, 26

H. Zimmer, A. Bruhn, and J. Weickert. Optic flow in harmony. *IJCV*, 2011. 54, 55, 85

C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 123, 129, 130, 132, 139

Appendix A

Computation of optical flow using variational models

Contents

In this appendix, we explain the minimization technique used for instance by Brox et al. [2004], to estimate the optical flow based on Euler-Lagrange equation. Assume that the energy to minimize has a data term with brightness and gradient constancy assumptions, a smoothness term that penalizes the norm of flow gradient, and a matching term to penalize the difference between flow and input matches:

$$\begin{aligned} E(\boldsymbol{w}) &= \underbrace{\int_\Omega \delta\Psi_D(\boldsymbol{W}^T\mathbb{J}_0\boldsymbol{W}) + \gamma\Psi_D(\boldsymbol{W}^T\mathbb{J}_{xy}\boldsymbol{W})d\boldsymbol{x}}_{\text{Data term}} \\ &+ \underbrace{\int_\Omega \Psi_S(\|\nabla_2\boldsymbol{w}\|_2^2)d\boldsymbol{x}}_{\text{Smoothness term}} \\ &+ \beta\underbrace{\int_\Omega \rho\Psi_M\Big(\|\boldsymbol{w}-\boldsymbol{w}_{match}\|_2^2\Big)d\boldsymbol{x}}_{\text{Matching term}} , \end{aligned} \tag{A.1}$$

with β, γ and δ some positive weights, $\mathbb{J}_0 = (\nabla_3 I)(\nabla_3^\top I)$ and $\mathbb{J}_{xy} = \mathbb{J}_x + \mathbb{J}_y = (\nabla_3 I_x)(\nabla_3^\top I_x) + (\nabla_3 I_y)(\nabla_3^\top I_y)$ the tensors for brightness and gradient constancy assumptions, Ψ_* some robust penalizers, $\boldsymbol{w}_{match}$ the displacement

estimated by a matching approach and ρ a per-matching weight with zero value at pixel without precomputed match.

By treating this energy as a calculus of variations, the Euler-Lagrange equation ensures that at the minimum:

$$\begin{cases} \frac{\partial E}{\partial u} - \frac{\partial}{\partial x}\frac{\partial E}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial E}{\partial u_y} = 0, \\ \frac{\partial E}{\partial v} - \frac{\partial}{\partial x}\frac{\partial E}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial E}{\partial v_y} = 0. \end{cases} \tag{A.2}$$

This leads to the following equations for each pixel:

$$\begin{cases} \delta\Psi'_D(I_t^2)I_tI_x \ + \ \gamma\Psi'_D(I_{xt}^2+I_{yt}^2)(I_{xx}I_{xt}+I_{xy}I_{yt}) \\ \qquad + \beta\rho\Psi'_M(\|\boldsymbol{w}-\boldsymbol{w}_{match}\|_2^2)(u-u_{match}) \\ \qquad - div(\Psi'_S(\|\nabla_2\boldsymbol{w}\|_2^2)\nabla u) = 0 \\ \delta\Psi'_D(I_t^2)I_tI_y \ + \ \gamma\Psi'_D(I_{xt}^2+I_{yt}^2)(I_{xy}I_{xt}+I_{yy}I_{yt}) \\ \qquad + \beta\rho\Psi'_M(\|\boldsymbol{w}-\boldsymbol{w}_{match}\|_2^2)(v-v_{match}) \\ \qquad - div(\Psi'_S(\|\nabla_2\boldsymbol{w}\|_2^2)\nabla v) = 0 \end{cases}, \tag{A.3}$$

where Ψ'_* is the derivative of Ψ_* with respect to its argument.

A.1 Optimization

Equation A.3 leads to a non-linear and highly non-convex problem. The first idea to approximate the solution consists in using a fixed point iteration on $\boldsymbol{w}$. Since the function is highly non-convex, the risk to fall into local minima is high. To this end, the fixed point iteration is not sufficient, we consequently use the classical coarse-to-fine scheme. A pyramid of images is built using smoothing and downsampling with a factor $\eta \in (0,1)$. We first initialize the flow with zero values at each pixel. We compute the flow at the coarsest level. It is then used to initialize the flow computation at the next level and so on. This strategy has shown powerful properties. It has the advantage to first capture the motion of big parts and then to refine the flow of smallest structures contained into them. Moreover, since the images at the first levels have only few pixels, the computation is fast.

Let $\boldsymbol{w}^k$ be the computed flow at level k. We define also the warped image derivatives at this level:

$$\begin{gathered} I_x^k(\boldsymbol{x}) = \frac{\partial I_2\big(\boldsymbol{x}+\boldsymbol{w}^k(\boldsymbol{x})\big)}{\partial x} \qquad I_y^k(\boldsymbol{x}) = \frac{\partial I_2\big(\boldsymbol{x}+\boldsymbol{w}^k(\boldsymbol{x})\big)}{\partial y} \\ I_{xx}^k(\boldsymbol{x}) = \frac{\partial^2 I_2\big(\boldsymbol{x}+\boldsymbol{w}^k(\boldsymbol{x})\big)}{\partial x^2} \qquad I_{xy}^k(\boldsymbol{x}) = \frac{\partial^2 I_2\big(\boldsymbol{x}+\boldsymbol{w}^k(\boldsymbol{x})\big)}{\partial x\partial y} \qquad I_{yy}^k(\boldsymbol{x}) = \frac{\partial^2 I_2\big(\boldsymbol{x}+\boldsymbol{w}^k(\boldsymbol{x})\big)}{\partial y^2} \\ I_t^k(\boldsymbol{x}) = I_2\big(\boldsymbol{x}+\boldsymbol{w}^k(\boldsymbol{x})\big) - I_1\big(\boldsymbol{x}\big) \qquad I_{xt}^k(\boldsymbol{x}) = \frac{\partial I_t(\boldsymbol{x})}{\partial x} \qquad I_{yt}^k(\boldsymbol{x}) = \frac{\partial I_t(\boldsymbol{x})}{\partial y}\,. \end{gathered} \tag{A.4}$$

While warping, some pixels can fall outside the image boundaries. For these pixels, we remove the data term. Spatial image derivatives are computed using the mean of the two images and a five-point stencil resulting in a fourth-order approximation, *i.e.*, with a $(1, -8, 0, 8, -1)^\top/12$ filter. $\boldsymbol{w}^{k+1}$ is now given by solving:

$$\begin{cases} \delta\Psi'_D\big((I_t^{k+1})^2\big)I_t^{k+1}I_x^k \;+\; \gamma\Psi'_D\big((I_{xt}^{k+1})^2+(I_{yt}^{k+1})^2\big)\big(I_{xx}^kI_{xt}^{k+1}+I_{xy}^kI_{yt}^{k+1}\big) \\ \qquad +\beta\rho\Psi'_M(\|\boldsymbol{w}^{k+1}-\boldsymbol{w}_{match}^{k+1}\|_2^2)(u^{k+1}-u_{match}^{k+1}) \\ \qquad -div(\Psi'_S(\|\nabla_2\boldsymbol{w}^{k+1}\|_2^2)\nabla u^{k+1})=0 \\ \delta\Psi'_D\big((I_t^{k+1})^2\big)I_t^{k+1}I_y^k \;+\; \gamma\Psi'_D\big((I_{xt}^{k+1})^2+(I_{yt}^{k+1})^2\big)\big(I_{xy}^kI_{xt}^{k+1}+I_{yy}^kI_{yt}^{k+1}\big) \\ \qquad +\beta\rho\Psi'_M(\|\boldsymbol{w}^{k+1}-\boldsymbol{w}_{match}^{k+1}\|_2^2)(v^{k+1}-v_{match}^{k+1}) \\ \qquad -div(\Psi'_S(\|\nabla_2\boldsymbol{w}^{k+1}\|_2^2)\nabla v^{k+1})=0 \end{cases}. \tag{A.5}$$

This new system is still non-linear because of the Ψ'_* terms and of I_*^{k+1} containing temporal derivative. To this last issue, we can use a first-order Taylor expansion. However this is valid only under the assumption that the displacement is small. Consequently, we use an incremental method, by splitting $\boldsymbol{w}^{k+1}$ into two terms, $\boldsymbol{w}^{k+1}=\boldsymbol{w}^k+\boldsymbol{dw}^k$. If the η parameter is sufficiently close to 1, $\boldsymbol{dw}^k$ will be small. The first-order Taylor expansion gives:

$$\begin{aligned} I_t^{k+1} &\simeq I_t^k + I_x^k du^k + I_y^k dv^k, \\ I_{xt}^{k+1} &\simeq I_{xt}^k + I_{xx}^k du^k + I_{xy}^k dv^k, \\ I_{yt}^{k+1} &\simeq I_{yt}^k + I_{xy}^k du^k + I_{yy}^k dv^k. \end{aligned} \tag{A.6}$$

This leads to a new system of equations where the unknown are now du^k and dv^k:

$$\begin{cases} \delta\Psi'^k_1\big(I_x^k(I_t^k+I_x^kdu^k+I_y^kdv^k)\big) \\ \quad +\gamma\Psi'^k_2\big(I_{xx}^k(I_{xt}^k+I_{xx}^kdu^k+I_{xy}^kdv^k)+I_{xy}^k(I_{yt}^k+I_{xy}^kdu^k+I_{yy}^kdv^k)\big) \\ \quad +\beta\rho\Psi'^k_3\big(u^k+du^k-u_{match}^k\big) \\ \quad -div\big(\Psi'^k_4\nabla(u^k+du^k)\big)=0 \\ \delta\Psi'^k_1\big(I_y^k(I_t^k+I_x^kdu^k+I_y^kdv^k)\big) \\ \quad +\gamma\Psi'^k_2\big(I_{xy}^k(I_{xt}^k+I_{xx}^kdu^k+I_{xy}^kdv^k)+I_{yy}^k(I_{yt}^k+I_{xy}^kdu^k+I_{yy}^kdv^k)\big) \\ \quad +\beta\rho\Psi'^k_3\big(v^k+dv^k-v_{match}^k\big) \\ \quad -div\big(\Psi'^k_4\nabla(v^k+dv^k)\big)=0 \end{cases}, \tag{A.7}$$

with:

$$\begin{aligned}
\Psi'^k_1 &= \Psi'_D\big((I^k_t + I^k_x du^k + I^k_y dv^k)^2\big),\\
\Psi'^k_2 &= \Psi'_D\big((I^k_{xt} + I^k_{xx} du^k + I^k_{xy} dv^k)^2 + (I^k_{yt} + I^k_{xy} du^{k,l} + I^k_{yy} dv^k)^2\big),\\
\Psi'^k_3 &= \Psi'_M\big((u^k + du^k - u^k_{match})^2 + (v^k + dv^k - v^k_{match})^2\big),\\
\Psi'^k_4 &= \Psi'_S\big(\|\nabla_2(u^k + du^k)\|^2_2 + \|\nabla_2(v^k + dv^k)\|^2_2\big).
\end{aligned} \tag{A.8}$$

This system is still non-linear due to the Ψ'_*. We so use another inner fixed point iteration: at each level, we alternatively compute the Ψ'_* coefficients with the current flow and update the incremental flow while Ψ'_* values are fixed. In practice, we perform $n_{inner} = 5$ iterations of this strategy.

We denote by l the index of this inner iteration and by $\boldsymbol{dw}^{k,l}$ the flow increment after l inner iterations at level k. We start with $\boldsymbol{w}^0$ and $\boldsymbol{dw}^{0,0}$ with full zeroes, compute the $\Psi'^{0,0}_*$ coefficients with the flow $\boldsymbol{w}^{0,0} = \boldsymbol{w}^0 + \boldsymbol{dw}^{0,0}$, we solve the linear system to compute a flow increment $\boldsymbol{dw}^{0,1}$. It is used to update the $\Psi'^{0,1}_*$ values, and so on until $\boldsymbol{dw}^{0,n_{inner}}$. Then $\boldsymbol{w}^1$ is set to the sum of $\boldsymbol{w}^0$ and $\boldsymbol{dw}^{0,n_{inner}}$ and we start the next level with $\boldsymbol{dw}^{1,0}$ set to 0.

Concerning the smoothing term, we compute the flow gradient using the filter $(-1, 1)^\top$. For a pixel $\boldsymbol{x}$, let $\mathcal{N}(\boldsymbol{x})$ be the 4-neighborhood of $\boldsymbol{x}$. We discretize the divergence operator $div(f_{u,v}\nabla u)$ by:

$$\sum_{j\in\mathcal{N}(i)} \frac{f_{u,v}(i) + f_{u,v}(j)}{2}\big(u(j) - u(i)\big). \tag{A.9}$$

This results in a linear system with $2 \times n_k$ equations and unknowns, where n_k denotes the number of pixels of the level k. For an image of resolution 640×480, this gives more than 600000 constraints. The size of the system is too huge to find an exact solution. Consequently, we use an iterative method to approximate the solution. For each pixel $i \in \Omega$, we obtain the following two equations, where the indices i is only written in the smoothness for a better readability:

$$\begin{cases}
\delta\Psi'^{k,l}_1\big(I^k_x(I^k_t + I^k_x du^{k,l+1} + I^k_y dv^{k,l})\big)\\
\quad + \gamma\Psi'^{k,l}_2\big(I^k_{xx}(I^k_{xt} + I^k_{xx} du^{k,l+1} + I^k_{xy} dv^{k,l}) + I^k_{xy}(I^k_{yt} + I^k_{xy} du^{k,l+1} + I^k_{yy} dv^{k,l+1})\big)\\
\quad + \beta\rho\Psi'^{k,l}_3\big(u^k + du^{k,l+1} - u^k_{match}\big)\\
\quad - \sum_{j\in\mathcal{N}(i)} \frac{\Psi'^{k,l}_4(i) + \Psi'^{k,l}_4(j)}{2}(u^k(j) + du^{k,l+1}(j) - u^k(i) - du^{k,l+1}(i)) = 0\\
\delta\Psi'^{k,l}_1\big(I^k_y(I^k_t + I^k_x du^{k,l+1} + I^k_y dv^{k,l+1})\big)\\
\quad + \gamma\Psi'^{k,l}_2\big(I^k_{xy}(I^k_{xt} + I^k_{xx} du^{k,l+1} + I^k_{xy} dv^{k,l}) + I^k_{yy}(I^k_{yt} + I^k_{xy} du^{k,l+1} + I^k_{yy} dv^{k,l+1})\big)\\
\quad + \beta\rho\Psi'^{k,l}_3\big(v^k + dv^{k,l+1} - v^k_{match}\big)\\
\quad - \sum_{j\in\mathcal{N}(i)} \frac{\Psi'^{k,l}_4(i) + \Psi'^{k,l}_4(j)}{2}(v^k(j) + dv^{k,l+1}(j) - v^k(i) - dv^{k,l+1}(i)) = 0
\end{cases}, \tag{A.10}$$

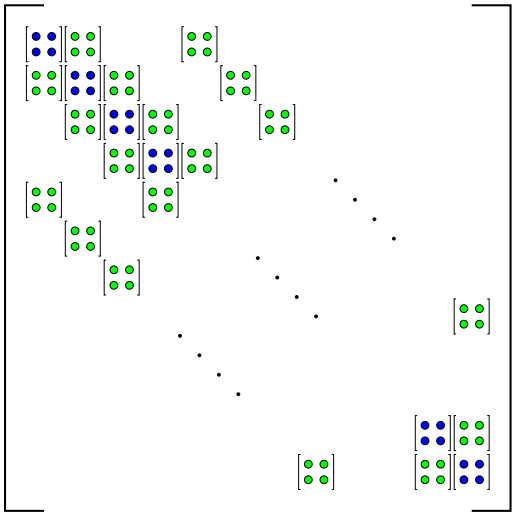


Figure A.1 – Structure of the matrix for the linear system. The blue blocks represent the diagonal submatrices while the green blocks correspond to submatrices defining the smoothness between neighborhood pixels. For an isotropic smoothness term, the green matrices are the identity times a coefficient. The other block matrices are all-zeroes.

with:

$$\begin{aligned}
\Psi_1'^{k,l} &= \Psi'_D\big((I_t^k + I_x^k du^{k,l} + I_y^k dv^{k,l})^2\big),\\
\Psi_2'^{k,l} &= \Psi'_D\big((I_{xt}^k + I_{xx}^k du^{k,l} + I_{xy}^k dv^{k,l})^2 + (I_{yt}^k + I_{xy}^k du^{k,l} + I_{yy}^k dv^{k,l})^2\big),\\
\Psi_3'^{k,l} &= \Psi'_M\big((u^k + du^{k,l} - u_{match}^k)^2 + (v^k + dv^{k,l} - v_{match}^k)^2\big),\\
\Psi_4'^{k,l} &= \Psi'_S\big(\|\nabla(u^k + du^{k,l})\|_2^2 + \|\nabla(v^k + dv^{k,l})\|_2^2\big). \qquad \text{(A.11)}
\end{aligned}$$

A.2 Linear solver

The set of equations can also be written as a linear system $\mathcal{A}^{k,l}x = \mathcal{B}^{k,l}$. If we use a row-wise order on the pixels, and if we define x by the concatenation of vectors $(u^{k,l+1}, v^{k,l+1})^T$ for each pixel, $\mathcal{A}^{k,l}$ has the structure shown in Figure A.1. We decompose the matrix into 2×2 blocks. Let $\mathfrak{A}_{i,j}$ be the block at position (i,j). We have the following structure for the diagonal block $\mathfrak{A}_{i,i}$ at position $(i,i), 1 \leqslant i \leqslant n_k$, with $\Psi_{4,i}'^{k,l} = \sum_{j\in\mathcal{N}(i)} \frac{\Psi_4'^{k,l}(j)+\Psi_4'^{k,l}(i)}{2}$:

$$\begin{pmatrix} \delta\Psi_1'^{k,l} {I_x^k}^2 + \gamma\Psi_2'^{k,l}({I_{xx}^k}^2 + {I_{xy}^k}^2) + \beta\rho\Psi_3'^{k,l} + \Psi_{4,i}'^{k,l} & \delta\Psi_1'^{k,l} I_x^k I_y^k + \gamma\Psi_2'^{k,l}(I_{xx}^k I_{xy}^k + I_{xy}^k I_{yy}^j) \\ \delta\Psi_1'^{k,l} I_x^k I_y^k + \gamma\Psi_2'^{k,l}(I_{xx}^k I_{xy}^k + I_{xy}^k I_{yy}^j) & \delta\Psi_1'^{k,l} {I_y^k}^2 + \gamma\Psi_2'^{k,l}({I_{xy}^k}^2 + {I_{yy}^k}^2) + \beta\rho\Psi_3'^{k,l} + \Psi_{4,i}'^{k,l} \end{pmatrix} \qquad \text{(A.12)}$$

For the non-diagonal block $\mathfrak{A}_{i,j}$, *i.e.* with $j \in \mathcal{N}(i)$, we obtain:

$$\begin{pmatrix} -\frac{\Psi_4'^{k,l}(j)+\Psi_4'^{k,l}(i)}{2} & 0 \\ 0 & -\frac{\Psi_4'^{k,l}(j)+\Psi_4'^{k,l}(i)}{2} \end{pmatrix}. \tag{A.13}$$

And finally for the right hand side $\mathfrak{b}_i$:

$$\begin{pmatrix} -\delta\Psi_1'^{k,l} I_x^k I_t^k - \gamma\Psi_2'^{k,l}(I_{xx}^k I_{xt}^k + I_{xy}^k I_{yt}^k) - \beta\rho\Psi_3'^{k,l}(u^k - u_{match}^k) \\ + \sum_{j\in\mathcal{N}(i)} \frac{\Psi_4'^{k,l}(j)+\Psi_4'^{k,l}(i)}{2}\big(u^k(j) - u^k(i)\big) \\ -\delta\Psi_1'^{k,l} I_y^k I_t^k - \gamma\Psi_2'^{k,l}(I_{yy}^k I_{yt}^k + I_{xy}^k I_{xt}^k) - \beta\rho\Psi_3'^{k,l}(v^k - v_{match}^k) \\ + \sum_{j\in\mathcal{N}(i)} \frac{\Psi_4'^{k,l}(j)+\Psi_4'^{k,l}(i)}{2}\big(v^k(j) - v^k(i)\big) \end{pmatrix}. \tag{A.14}$$

Since $\mathfrak{A}$ is highly dimensional (2 equations per pixel), only iterative solvers can be applied. Note that $\mathfrak{A}$ is sparse: there are 6 non-zero values per row/column (or 10 in case of an anisotropic smoothness term). Moreover, $\mathfrak{A}$ is structured as a 7-diagonal matrix and as a 5-diagonal 2×2 blocks, see FigureA.1. $\mathfrak{A}$ is positive semi-definite and also block diagonally dominant [Sundaram et al., 2010] if the robust penalizers are increasing. The diagonal blocks are positive definite. These properties are sufficient to prove the convergence of most solvers.

The most used ones are the Successive Over Relaxation (SOR) [Young and Rheinboldt, 1971] which is a relaxation of Gauss-Seidel method, and a conjugate gradient descent with an appropriate preconditioner. Let n be the dimension of the problem and x^k the estimation after k iterations of the iterative solver. The SOR algorithm is sequential, making it inefficient for a GPU implementation, see Algorithm A.1. With SOR, the flow component u and v are not updated simultaneously. We consequently use a coupled version of the algorithm in which the sequential update operates on 2×2 sub-matrices, see Algorithm A.2.

Algorithm A.1 Successive Over Relaxation (SOR) method. ω is a parameter.

For k from 1 to n_{solver}
—— **For** i from 1 to $\frac{n}{2}$
——— $x^k(i) = (1-\omega)x^{k-1}(i)$
$+\frac{\omega}{\mathfrak{A}(i,i)}\Big(\mathfrak{b}(i) - \sum_{j>i}\mathfrak{A}(i,j)x^{k-1}(j) - \sum_{j<i}\mathfrak{A}(i,j)x^k(j)\Big)$

For a GPU implementation, conjugate gradient descent (see Algorithm A.3) is better suited [Sundaram et al., 2010] as it only involves matrix operations. A preconditioning with block-Jacobi preconditioner is necessary for fast convergence.

Algorithm A.2 Coupled variant of Successive Over Relaxation (SOR) method. ω is a parameter.

For k from 1 to n_{solver}
—– **For** i from 1 to n
———– $x_i^k = (1-\omega)x_i^{k-1}$
$+\omega\mathfrak{A}_{i,i}^{-1}\left(\mathfrak{b}_i - \sum_{j>i}\mathfrak{A}_{i,j}x_j^{k-1} - \sum_{j<i}\mathfrak{A}_{i,j}x_j^k\right)$

Algorithm A.3 Preconditioned Conjugate Gradient (PCG) with a preconditioner M.

$r_0 = \mathfrak{b} - \mathfrak{A}x^0$
$z_0 = M^{-1}r_0$
$p_0 = z_0$
For k from 0 to $n_{solver} - 1$
—– $\alpha_k = \frac{r_k^T z_k}{p_k^T \mathfrak{A} p_k}$
—– $x^{k+1} = x^k + \alpha_k p_k$
—– $r_{k+1} = r_k + \alpha_k \mathfrak{A} p_k$
—– $z_{k+1} = M^{-1}r_{k+1}$
—– $\beta_k = \frac{z_{k+1}^T r_{k+1}}{z_k^T r_k}$
—– $p_{k+1} = z_{k+1} + \beta_k p_k$

A.3 Summary of the algorithm

Algorithm A.4 summarizes the whole optimization process. Note that we perform a last iteration with $\beta = 0$ as Brox and Malik [2011]. Indeed, descriptors matching is important at the beginning to encourage the flow estimate to follow the matching. Once the initialization is well done, this is less important to integrate it into the equation's system.

Algorithm A.4 Variational estimation of optical flow.

Input: I_1, I_2
Output: $\boldsymbol{w}$
Presmooth images with Gaussian filter
Compute descriptor matches $\boldsymbol{w}_{match}$ and their weights ρ
Build image pyramids with factor η
Initialize $\boldsymbol{w}^0$ to 0 (size of the coarsest level)
For k from 0 to n_{level} *(from coarsest to finest level)*
—– Warp second images according to $\boldsymbol{w}^k$
—– Resize matches and weights to the current level size
—– Compute image derivative I_*^k
—– Initialize $\boldsymbol{dw}^{k,0}$ to 0 (size of current level)
—– **For** l from 0 to $n_{inner} - 1$
———– Compute $\Psi_*'^{k,l}$
———– Build system matrix $\mathcal{A}^{k,l}$ and $\mathcal{B}^{k,l}$
———– Apply linear solver to obtain $\boldsymbol{dw}^{k,l+1}$
—– $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + \boldsymbol{dw}^{k,l+1}$ *(add the increment to the current estimation)*
—– Resize $\boldsymbol{w}^{k+1}$ to the size of the next level
Perform a last iteration with $\beta = 0$

Appendix B

The DALY dataset

Contents

In this appendix, we describe how DALY was collected. Section B.1 details action class selection, video filtering and spatio-temporal annotation of action instances. Section B.2 presents per-class statistics for the dataset.

B.1 Dataset collection

Picking action classes. In order to allow precise annotation, we choose action classes with clearly defined temporal boundaries. For instance, the *brushing teeth* action is defined as *'toothbrush inside the mouth'*. Another example is *cleaning windows* for which the moment where *'the tool is in contact with the window'* is annotated.

Some of the classes are chosen to contain similar motion patterns, in order to make the class distinction difficult. Several of our action classes imply motion of the hands near the head (*taking photos, phoning*) or the mouth (*playing harmonica, drinking, brushing teeth, applying make up on lips*).

In summary, we kept the following 10 actions: *applying make up on lips, brushing teeth, cleaning floor, cleaning windows, drinking, folding textile, ironing, phoning, playing harmonica and taking photos/videos*, see Figure B.1.

Video collection. The videos are gathered from YouTube using manually designed queries related to the selected action classes. For the class

Figure B.1 – One example frame for each of the 10 classes of the DALY dataset.

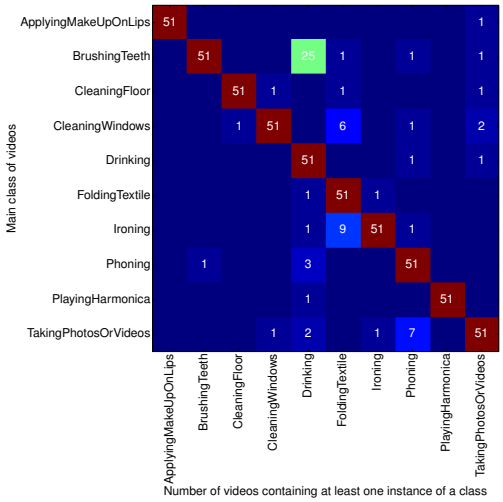
cleaning floor, the queries include *'sweeping floor'*, *'mopping floor'*, *'cleaning floor'*, etc. We only keep videos with a duration between 1 and 20 minutes. A minimum duration of 1 min ensures that temporal localization will be meaningful (shorter videos contain only one action from the beginning to the end in most cases), and a maximum duration of 20 minutes avoids issues related to computational time or memory consumption.

Videos are filtered to remove cartoons, slideshows, actions performed by animals and first-person viewpoints. We also remove videos in which the human is not visible when the action occurs, for instance when the camera focuses on the mop while performing the *cleaning floor* action.

51 videos are selected for each action class such that they contain at least one instance of the action class. In total this corresponds to 31 hours of video, or 3.3 million frames. Videos from a given action class often contain multiple instances of the main action, and may contain instances of other action classes, which we annotate exhaustively.

Temporal annotation. Selected videos are carefully watched by members of our research team in order to catch all actions, including those happening in the background. The *begin* and *end* time is annotated for all instances found. Precise guidelines are established prior to annotation. For example, the *phoning* action lasts as long as the phone remains close to the actor's ears. In case of a shot change during an action, we annotate it as two separate instances and set a *'shotcut'* flag. DALY contains 3724 action instances in total, with an average duration of 8 seconds.

Spatial annotation. There are more than 700k frames containing at least one action. Annotating all of them is a tall order, as it would take a year



class	avg video dur.	#inst.	avg inst. dur.
ApplyingMakeUpOnLips	376.8s ± 265.1	421	3.7s ± 3.3
BrushingTeeth	176.0s ± 120.3	277	9.3s ± 15.9
CleaningFloor	194.2s ± 128.7	200	14.1s ± 15.2
CleaningWindows	196.2s ± 131.9	468	7.1s ± 10.0
Drinking	202.4s ± 130.9	304	2.6s ± 3.0
FoldingTextile	184.1s ± 150.1	257	14.6s ± 22.4
Ironing	233.2s ± 183.8	424	7.2s ± 8.2
Phoning	217.9s ± 140.7	514	9.9s ± 30.2
PlayingHarmonica	190.2s ± 139.8	306	13.8s ± 21.4
TakingPhotosOrVideos	283.0s ± 207.3	553	3.1s ± 3.5
all	225.4s ± 175.7	3724	7.8s ± 16.4

Figure B.2 – *Left:* Statistics of multiple classes per video. Each row considers the 51 videos downloaded for a given class, each column counts the videos containing at least one instance of the column class. *Right:* Statistics for each class on the video duration (average and standard deviation), the number of instances, and the instance duration (average and standard deviation).

and a half for one annotator to complete this task, assuming 15 seconds per annotation and a 40-hour week. Of course this does not include verifying the results which, albeit quicker, is also time consuming when you need to outsource the job. Thus, we subsample frames to be annotated, such that enough information is present for a reliable evaluation of spatio-temporal detections.

For each temporal instance in the test set, we pick 5 uniformly sampled frames, with a maximum of 1 frame per second. For each frame, annotators are asked to draw a bounding box around the actor. Some of the spatial annotations are carried out by external workers. In all cases, the bounding boxes are reviewed and adjusted by members of our research team.

B.2 Dataset statistics

The selected action classes are sufficiently common such that multiple action classes can be found in a single video, see Figure B.2, left. The matrix should be read row by row, each row considers only the 51 videos selected for a given class. For example, out of the 51 videos selected for the class *brushing teeth*, 51 videos contain *brushing teeth* instances, 25 contain

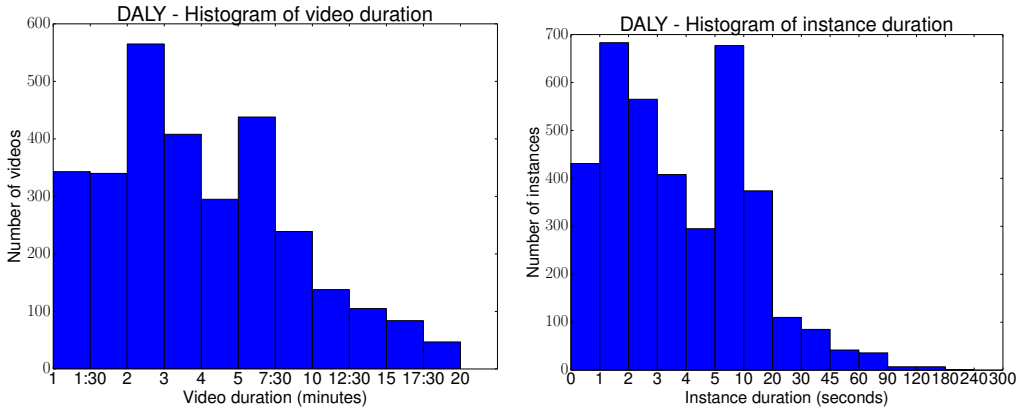


Figure B.3 – Histogram of duration of the videos (left) and instances (right).

drinking instances, etc.

Some classes have expected overlap such as *brushing teeth* and *drinking*, *ironing* and *folding textile*. There is also overlap between *taking photos/videos* and *phoning*, which can be explained by the fact that taking photos is mostly performed outdoors, where other people are *phoning*.

Figure B.3 shows the histogram of video duration and instance duration. Per-class statistics are presented in Figure B.2, right. One can see that most of the videos last several minutes (less than 10). Videos are longest in average for *ApplyingMakeUpOnLips*, mainly because this action is present in a multitude of full-face make-up tutorials. Concerning the instances, most of them are shorter than 10 seconds. Nevertheless, DALY also contains instances of several minutes, especially for actions such as *brushing teeth*, *playing harmonica* or *folding textile*. In some cases, short instance duration for actions can be explained by video editing. The uploader may cut the action to a few seconds and include it in a long video. Instances are shortest on average for *drinking* and *taking photos*, simply because drinking and taking a photo tend to take a short time (put the cup to the mouth and back for drinking, press the button to take a photo).