Robust and efficient models for action recognition and localization

Dan Oneata

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

Dan ONEAŢĂ

Thèse dirigée par **Cordelia SCHMID**
et codirigée par **Jakob VERBEEK**

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

Modèles robustes et efficaces pour la reconnaissance d'actions et leur localisation

Robust and efficient models for action recognition and localization

Thèse soutenue publiquement le **20 juillet 2015**,
devant le jury composé de :

Pr. Thomas Brox
University of Freiburg, Freiburg, Germany, Rapporteur

Pr. Jason Corso
University of Michigan, Ann Arbor, MI, USA, Rapporteur

Pr. Cees Snoek
University of Amsterdam, Amsterdam, Netherlands, Examinateur

Dr. Cordelia Schmid
Inria Grenoble, Montbonnot, France, Directeur de thèse

Dr. Jakob Verbeek
Inria Grenoble, Montbonnot, France, Co-Directeur de thèse

Abstract

This thesis addresses the problem of action recognition, *i.e.*, how to determine the type of action that is happening in a video and its temporal localization.

First, we consider the problem of video representation—how to encode videos in a robust way, such that the representation is suitable for a wide variety of action classes, tasks and video types. We present an extensive evaluation study that explores the Fisher vector (FV) encoding as an alternative to the bag of words (BOW) encoding; we further investigate different ways of including spatial layout information. Our experiments show that the FV encoding is superior to BOW both in terms of performance and efficiency. For the localization task, we introduce two novel variants of non-maxima supression that correct the bias towards windows that are too short.

Second, we improve the efficiency of the Fisher vector representation in the context of action localization. Power and ℓ_2 normalizations improve Fisher vector performance, but hamper the use of efficient localization techniques, such as, integral images or branch and bound. Our approximations lead to a speedup of at least one order of magnitude, while maintaining state-of-the-art action recognition and localization performance.

Third, we investigate the task of spatio-temporal action localization. A major challenge is the size of the search space defined by spatio-temporal tubes formed by sequences of bounding boxes along the frames. Recently, methods that generate unsupervised detection proposals have proven to be very effective for object detection in still images. Here, we introduce an approach for extracting spatio-temporal region proposals. First, we extend a recent 2D object proposal method, to produce spatio-temporal proposals by a randomized supervoxel merging process. Second, we propose a new efficient supervoxel method. Experimental results on the UCF Sports dataset show that we obtain a recall of over 70% when extracting about 100 spatio-temporal tubes per video.

Finally, in the appendix of the thesis, we present our winning submission for the THUMOS 2014 challenge. For the classification task, we build upon our FV-based video representation and complement the motion features with appearance and audio features. For the localization task, we improve the performance by including the classification score into the final localization score.

Keywords. Action recognition • Action localization • Event recognition • Video representation • Efficiency • Classification.

Résumé

Cette thèse traite du problème de la reconnaissance d'action, c'est-à-dire de la détermination du type de l'action en cours, ainsi que de sa localisation temporelle.

En premier lieu, nous traitons du problème de a représentation vidéo (comment encoder des vidéos de manière robuste, de telle sorte que cette représentation soit appropriée à une grande variété de classes d'action, de taches et de types de vidéos). Nous proposons une évaluation approfondie qui explore l'encodage par vecteur de Fisher (VF), une alternative au sac-de-mots (SDM). Nous explorons de plus différentes manières de tenir compte de l'information de disposition spatiale. Notre étude prouve que VF est supérieur au SDM en termes de performances et d'efficacité. Pour la tache de localisation, nous introduisons deux nouvelles variantes de suppression non-maximale qui corrige le biais envers des fenłtres trop courtes.

En second lieu, nous améliorons l'efficacité de la représentation par VF pour la localisation d'action. Les normalisations puissance et ℓ_2 améliorent les performances mais nuisent à l'utilisation de techniques de localisation efficaces comme les images intégrales et les algorithmes par séparation et évaluation. Nos approximations entrainent des accélérations d'au moins un ordre de grandeur tout en maintenant les performances à l'état de l'art en reconnaissance et en localisation d'action.

En troisième lieu, nous étudions la tache de localisation spatio-temporelle d'action. Une des difficultés majeures en est la taille de l'espace de recherche défini par les tubes spatio-temporels formés par des suites de boîtes englobantes au cours de images. Des méthodes pour engendrer de manière non-supervisée des propositions de détection ont récemment fait montre d'une grande efficacité pour la détection d'objets dans les images figées. Ici, nous introduisons une approche pour extraire des propositions de régions spatio-temporelles. Nous étendons tout d'abord une méthode récente de propositions d'objets 2D pour produire des propsoitions spatio-temporelles par un processus de fusion aléatoire de supervoxels. Dans un second temps, nous proposons une nouvelle méthode de supervoxels efficaces. Les résultats expérimentaux sur la base UCF-Sports montrent que l'on obtient un rappel de 70% en extrayant cent tubes spatio-temporels par vidéo.

Pour finir, nous présentons dans l'appendice de cette thèse notre soumission gagnante au concours THUMOS 2014. Pour la tache de classification, nous partons de notre représentation à base de VF à laquelle nous ajoutons des descripteurs de mouvement, ainsi qu'audio. Pour la tache de localisation, nous améliorons la performance en incluant le score de classification dans celui, final, de localisation.

Mots-clés. Reconnaissance d'action • Localisation d'action • Reconnaissance d'évènements • Représentation vidéo • Efficacité • Classification.

Acknowledgements

Working beside an industrious and gifted scholar, he who is lacking in will power can receive the baptism of fire in research. In such a laboratory he will observe with commendable envy the fervent ambition to wrest secrets from the unknown; he will absorb the unrelenting scorn toward vain theories and rhetorical discourse; and, finally, on foreign soil, he will experience the rebirth of a growing patriotism.

— Santiago Ramón y Cajal*, Advice to a Young Investigator*

I could not have said it better than Ramón y Cajal: working alongside such industrious and gifted scholars as Jakob Verbeek and Cordelia Schmid was a proper introduction to research—a true baptism of fire. I am greatly indebted to both my supervisors for accepting me as their student and for gracefully accompanying me throughout the PhD. Much of this research would never have been possible without their guidance.

I would like to thank my jury members, Thomas Brox, Jason Corso, Cees Snoek, for kindly accepting to review my work and for their effort to travel long distances to attend my thesis defense.

This work was carried out at INRIA in the LEAR team. The LEAR team proved to be an excellent environment to conduct research and for social interactions. I have learnt a lot from other researchers: Karteek Alahari, Matthijs Douze, Zaid Harchaoui, Julien Mairal. I would like to especially thank Matthijs Douze for coordinating our team for the TRECVID competitions. I have enjoyed working with him and the rest of team members: Mattis Paulin, Danila Potapov, Jerome Revaud, Heng Wang. Many of the people in INRIA were not mere co-workers, but true friends. They shaped me and made me a better person. Among others, I would like to thank Zeynep Akata, Mohamed Ayari, Adnane Boukhayma, Anoop Cherian, Nicolas Chesneau, Gokberk Cinbis, Guillaume Fortier, Adrien Gaidon, Ahmed Gamal-Eldin, Nathalie Gillot, Albert Gordo, Michael Guerzhoy, Yang Hua, Hongzhou Lin, Vasiliki Kalogeiton, Federico Pierucci, Shreyas Saxena, Gaurav Sharma, Franck Thollard, Pavel Tokmakov, Vagia Tsiminaki, Philippe Weinzaepfel.

The Romanian community in Grenoble was incredibly helpful and kind to me. Not only they helped me prepare *le pot de thèse*, but they also kept me sane at times. Thanks to Paul Balint, Father Ioan Nicolae Căpuţan, Flavia Dioşan, Marius Gligor, Maria Lupşea, Tatiana Nemţeanu, Irina Nicolae, Claudia Rusu, George Toader.

Finally I would like to thank to my family (my parents and my sister) for all their support and patience. This thesis is dedicated to all my former teachers, professors and supervisors without which I would not have been here today and which cultivated in me curiosity and passion for research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

In a red-tinted view we are shown a parking lot. Numbers in a white monospaced font are flowing on both sides of the scene as each vehicle is in turn segmented and identified: a Harley Davidson motorcycle, a Yamaha one, a Plymouth sedan car. We enter into the bar at the end of the car park and we detect and classify each person as male or female, as mesomorphic, ectomorphic or endomorphic. We stop at a mesomorphic male, the typical bearded, long-haired guy wearing a leather jacket. We scan him slowly, from bottom to top, fitting 3D models to his calves, thighs, torso, hands and head. The displayed message—*match*—indicate that the recognition algorithm identified him as the target.

That is an idealized computer vision system as portrayed in the Terminator 2 movie, see Figure 1.1. The Terminator view is able to analyze data in real time and accurately detect objects and humans, perform face recognition and scene understanding. Now, after almost 25 years since the movie release, we might wonder if we have closed the gap between science fiction and real-world applications.

The efforts of the computer vision community have certainly led to impressive results on some tasks. A lot of effort has been put into face-related tasks (detection, recognition, verification) and it has been shown that they can match human performance [Taigman et al., 2014, Phillips and O'Toole, 2014]. But the strongest case for their success comes from the wide-spread adoption. We are using applications based on face detection

Figure 1.1 – The Terminator view, an idealized computer vision system.

or recognition on a daily basis: almost any camera or mobile device is able to perform face detection when taking a picture; social networks can automatically tag our friends in the photos we upload. Another application that has had a lot of commercial success is the Kinect console from Microsoft. The device uses a 3D human pose estimation algorithm to allow its user the control using the body movement.

This thesis has at its core the task of action recognition—a fundamental problem of computer vision that is yet to be solved. The goal is to automatically output the action that is happening in a video. Action recognition is one of the building blocks of an intelligent vision system. But if machine vision *à la* Terminator seems far fetched, there are other exciting applications in sight. One such example is Google Glass, a smart device, which is worn similarly to a pair of glasses, but whose usual lenses are replaced by a small screen and a camera. The most interesting and controversial feature of the Google Glass is its video recording capability. Being able to record all the daily activities will result into a tremendous amount of information. In such a scenario only an automatic system can aid us in organizing and understanding the data. A similar motivation for automated systems comes from other applications as well: indexing large collections of videos on video-sharing web-sites (YouTube, Vimeo); monitoring the behaviour and activities in surveillance applications.

1.1 Context

Action recognition is still a recent field of research, but it grew swiftly in the last couple of decades. We can trace back the first attempts at action recognition in the beginning of the 1990's: classification of tennis strokes [Yamato et al., 1992], gait recognition [Niyogi and Adelson, 1994], recognition of ballet steps [Campbell and Bobick, 1995]. Since then many vision

researchers directed their efforts on the problem of action recognition and by 2000 already seven review papers were published on action recognition and related topics, such as motion estimation [Gavrila, 1999] or motion extraction [Aggarwal and Cai, 1999].

Moeslund et al. [2006] identify four main stages of action recognition systems: (i) initialization (detecting the human in the first frame of the sequence); (ii) tracking (updating the human detections as the video progresses); (iii) modeling (representing the human, typically by estimating its pose); (iv) classification (recognizing the action performed based on the modeled features).

Early action recognition systems relied on previous work on model-driven approaches [Roberts, 1963, Marr and Nishihara, 1978]. The initial trend was to model the human as precisely as possible using prior information about the human body. In one of the first papers that analyzes human motion in videos, Hogg [1983] builds a 3D human model based on cylinders. The human model had a hierarchical structure (the body has a head, a torso, arms, legs; the leg has a calf and a lower-leg; the arm has a lower-arm and upper-arm) and the model was pre-specified (a full page of the paper was dedicated to enumerating the relative coordinates and orientation of each body part). The human is detected by first subtracting the background, then extracting the edges and the method is evaluated by projecting the human model onto the images. Initially 3D volumetric human models were used in action recognition [Rohr, 1994, Campbell et al., 1996], but progressively simpler models, like silhouettes [Yamato et al., 1992, Brand, 1999], became a popular way of representing humans. These avoid finding the location of each body part in 3D, and instead estimate the 2D position of locations in each frame.

The problem of classification is often reduced to the problem of matching: find the sequence in the training data set that aligns best with the query sequence and predict the corresponding class. Given the temporal nature of the data, tools from speech processing have been adapted and become commonly used. Dynamic time warping (DTW; Sakoe and Chiba, 1978) is a well known technique to match a test pattern with a reference pattern if their time scales are not perfectly aligned but when temporal ordering constraints do hold. For example, [Darrell and Pentland, 1993] use DTW for recognizing hand gestures. Another very popular method is to use hidden Markov models (HMM; Rabiner, 1989), which can be regarding as non-deterministic state machines that move from state to state based on learnt probabilities. Matching involves the computation of the probability that a particular HMM could have generated the test sequence which corresponds to the observed image features. Numerous

papers have leveraged the flexibility of HMMs: Yamato et al. [1992] use a discrete HMM to represent sequences over a set of vector quantized silhouette features of tennis footage, Starner and Pentland [1995] use a continuous HMM for recognition of American sign language, Wilson and Bobick [1995] are recognizing hand gestures using a HMM.

Starting from the 2000's, action recognition became one of the most popular topics in the computer vision community. The vast number of papers published in the literature each year stands as evidence for this. In Chapter 2 we review recent work and look into more detail at related methods.

1.2 Goals

This dissertation is concerned with three tasks of video understanding: action recognition, event recognition and action localization. In the following we briefly describe these tasks.

Action recognition The goal of recognition is to assign a query video to one of the given classes. We assume that we have access to a train dataset of videos with their corresponding class labels. According to the type of action that is performed in the video, we divide recognition into two tasks: action recognition and event recognition. See Figure 1.2 for examples of actions and events.

Actions typically consist of a series of atomic movements, for example, general body movement (*walk*, *sit down*, *dive*), human interactions (*kiss*, *shake hands*, *fight*), interactions between a human and an object (*weight lifting*, *ride horse*, *kick ball*). Actions are usually short, lasting from few seconds to tens of seconds.

Event recognition Events are more complex than actions and they have a greater semantic span. An event is not captured solely by the movement of the people, but scene and sound can prove important cues. For example, two different types of event categories are instructional tutorials (*make a sandwich*, *change a vehicle tire*, *repair an appliance*) and social events (*parade*, *birthday party*, *flash mob gathering*). Events usually last longer than actions, from tens of seconds to minutes.

Action localization The goal of action localization is to identify where in a video a given action is taking place. We distinguish two types of action localization: temporal localization and spatio-temporal localization. Temporal localization finds the temporal span of the action,

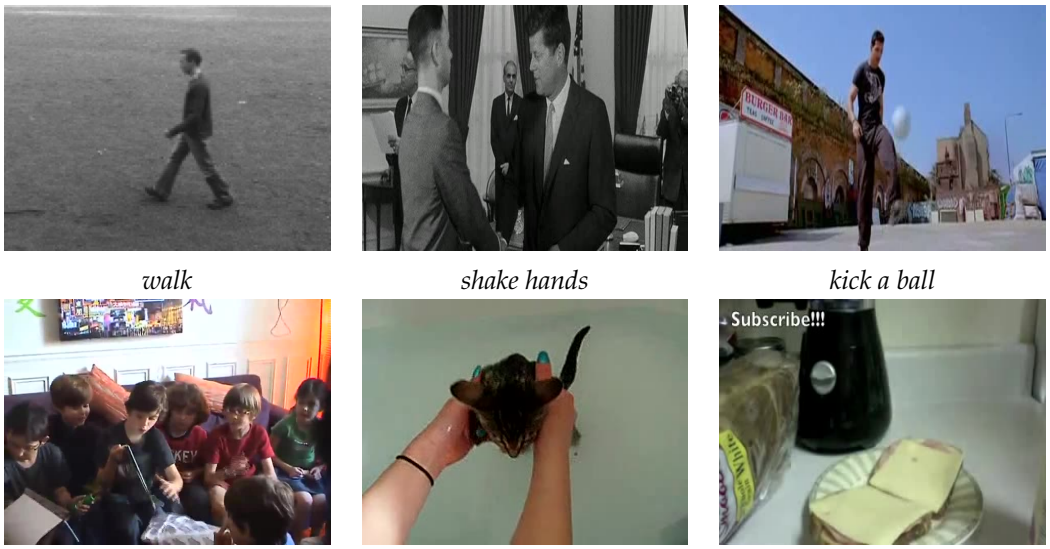


Figure 1.2 – Examples of actions (top row) and events (bottom row).

i.e. the beginning and ending frames. Spatio-temporal localization finds where the action lies both in time and in space: temporally it finds the beginning and ending frames, spatially it finds the bounding box that encloses the region of interest. Figure 1.3 illustrates the difference between temporal and spatio-temporal action localization.

In this thesis we address two specific goals to improve the techniques for video understanding.

1. **Task-independent video representation** We consider three tasks of video understanding: action recognition, action localization, event recognition. Albeit distinct, these tasks are intrinsically connected: the success is determined by how well we encode the action into a video representation. So instead of using a model specific for each task, we aim to provide a single video representation, suitable for all of them. A task-independent video representation is appealing from an engineering point of view because it enables reuse. It also allows us to transfer information from one task to another. One such example is training classifiers for the action recognition task and then using these classifiers to score chunks of video—the scores will act as mid-level features for the event recognition task.
2. **Efficiency** We live in the age of big data and video makes no exception. For example, in 2014 for the TRECVID multimedia event

Figure 1.3 – Two action localization sub-tasks: temporal action localization (top row) and spatio-temporal localization (bottom row). The red box indicates correct localization for each of the two sub-tasks. Temporal action localization is concerned with detecting the temporal boundaries. Spatio-temporal action localization is concerned with detecting the temporal boundaries and spatial boundaries, inside each selected frame.

detection competition there was provided a test set of about 200,000 videos totalling around 8,000 hours of video. Efficient video representations are crucial for tackling such challenges. Another argument for efficiency is the task of action localization. The usual way of finding the location of an action is by searching through all possible locations and scales. This approach becomes quickly infeasible if either the video grows in size or we increase the number of locations and scales. We consider methods of pruning the state space of all possible windows.

1.3 Contributions

This thesis makes the following three main contributions:

- We perform an in-depth evaluation of low-level local features and encodings for three tasks: action recognition, temporal action localization, event recognition. We consider two state-of-the-art low-level features (dense trajectories and improved trajectories) and two of the most popular encoding techniques (bag of words and Fisher vectors). We also evaluate two different approaches of including weak spatio-temporal information: spatial pyramid matching and spatial Fisher vectors. For the task of action localization we observe that the standard way of selecting the windows using non maximum suppression (NMS) results in a large number of short windows. We

propose a variant of NMS that corrects this issue without introducing any additional cost. This work was published in ICCV 2013 [Oneata et al., 2013]; we present it in Chapter 3.

- We introduce an approach for efficient temporal action localization. A state-of-the-art representation is based on the Fisher vector encoding whose performance is attributed to its high dimensionality and non-linear normalizations. These two properties impact negatively the efficiency, because of the large number of windows that need to be evaluated. We propose a way to approximate the normalizations by relying on pre-computed sums of local visual word assignments, scores, and ℓ_2 norms. This method allows to score an arbitrarily-sized window in constant time and to reduce the computational cost by an order of magnitude without any significant impact on the final performance. This work was published in CVPR 2014 [Oneata et al., 2014b]; we present it in Chapter 4.
- We propose an approach for making spatio-temporal action localization more efficient. The idea is to reduce the number of windows to be examined by generating spatio-temporal proposals. We build on the recent approach of Manen et al. [2013] that uses a randomized superpixel merging procedure to obtain object proposals. We extend their approach to the 3D video domain, using supervoxels as the units that will be merged into larger, video proposals. We also propose a supervoxel method that is based on hierarchical clustering of per-frame extracted superpixels. This work was published in ECCV 2014 [Oneata et al., 2014a]; we present it in Chapter 5.

Appendix A presents our submission at the THUMOS 2014 challenge. The task of the challenge is to evaluate action recognition and localization approaches in realistic conditions. The data consists of untrimmed videos, where the action may be short compared to the video length, and multiple instances can be present in each video. For the classification task, we use the Fisher vector representation and complement motion features with appearance (SIFT, Color and CNN) and audio features (MFCC and ASR). For the localization task, we improve the performance by including the classification score into the final score of video slices. We have ranked second for the classification task and first for the localization task.

Before moving to our main contributions, described in chapters 3, 4 and 5, we first review, in Chapter 2, related work for each of the three tasks outlined above in Section 1.2. Finally we present our conclusions and discussions in Chapter 6.

Chapter 2

Related work

Contents

In this chapter we present recent work on the problems of video understanding. We divide the related work by the task they address: action recognition (Section 2.1), event recognition (Section 2.2), action localization (Section 2.3). We draw the attention however that this separation is rather rough. On one hand, techniques that are useful for one task can prove useful for a different task. Take the example of local low-level features, which we present in Subsection 2.1.1; even if they were developed in the context of action recognition, they were later successfully applied for action localization, as building blocks of more sophisticated models. On the other hand, seemingly different methods might share similar underlying functionally. For example, Girshick et al. [2015] show that deformable part models (see Subsection 2.3.1) can be casted as a convolutional neural network (see Subsection 2.1.3).

There is a vast amount of papers published each year in the literature on action recognition and related topics. In this chapter, we mostly address recent work. The interested reader can a get broader historical overview by consulting the previously published reviews on the topics [Gavrila, 1999, Moeslund et al., 2006, Poppe, 2010, Aggarwal and Ryoo, 2011, Weinland et al., 2011, Cheng et al., 2015]. Most of these papers survey the task of action recognition, but Poppe [2010] presents also work on the action localization task, while Jiang et al. [2013a] reviews the task of event recognition.

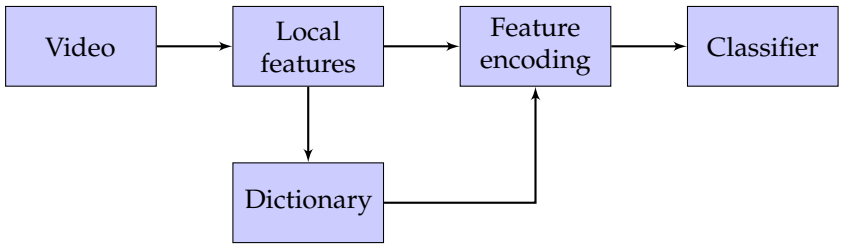


Figure 2.1 – The action recognition pipeline.

2.1 Action recognition

An important moment in the development of action recognition is the introduction of standardized datasets. At the beginning of the 2000's, Schüldt et al. [2004] introduce the KTH dataset and Blank et al. [2005] introduce the Weizmann dataset. Both these datasets contain sequences of simple actions, like *walking*, *running*, *waving*, and are filmed in constrained environments (*e.g.*, fixed camera, a single dominant actor performing the action). As performance on these datasets saturated, more challenging and realistic datasets appeared, containing clips from movies, TV broadcasts or consumer videos. On these datasets most of the previous methods proved unreliable as their assumptions were too strict: none or constant camera motion, lack of occlusions, the presence of a single person in the scene [Moeslund and Granum, 2001].

The methods that were shown to cope best with the new challenges are based on low-level local feature extraction, followed by bag of words encoding of the features. Figure 2.1 illustrates the video processing pipeline for action recognition. We dedicate the first two subsections to describe advances in these main stages: local feature extraction in Subsection 2.1.1 and the encoding methods and dictionary learning in Subsection 2.1.2. Recently, it has been proposed the replacement of this pipeline with a deep learning representation. We review these approaches in Subsection 2.1.3.

2.1.1 Local features

Local features encode appearance and motion information of small video volumes (spatio-temporal features) or sequences of image patches (trajectory features). They aim to provide an independent representation of actions with respect to their spatio-temporal shifts and scales. Their appeal comes from two reasons: (i) they make no assumptions on the global structure of actions; (ii) they are extracted directly from the video. In this way they are able to avoid error-prone pre-processing steps, such

as, motion segmentation and tracking, and, as a consequence, they have been shown to be robust under uncontrolled conditions (background clutter, multiple motions). The performance of local features has been demonstrated on a variety of data: movies [Laptev et al., 2008, Marszałek et al., 2009], TV broadcasts [Niebles et al., 2010, Patron-Perez et al., 2010], consumer videos [Liu et al., 2009].

Spatio-temporal features. Local spatio-temporal features aim to describe small three dimensional spatio-temporal volumes of the video using local descriptors. The descriptors usually encode statistics of the pixel distributions in different cells of the volume. Most of the approaches can be viewed as extensions of image descriptors. Laptev et al. [2008] encode local appearance with histograms of oriented gradients (HOG; Dalal and Triggs, 2005 and local motion with histograms of optical flow (HOF). Kläser et al. [2008] extends HOG to the space-time domain by using histograms of 3D gradient orientations.

Spatio-temporal features are usually extracted at precise locations and scales. These characteristics are selected by an *interest point detector*, whose goal is to maximize the saliency and the stability under various perturbations (*e.g.*, scale, illumination). As for the features, the methods for spatio-temporal interest point detectors draw inspiration from the image counterpart: the Harris3D detector [Laptev, 2005] extends to the spatio-temporal domain the Harris cornerness criterion [Harris and Stephens, 1988], the cuboids of Dollár et al. [2005] uses Gabor filters for the temporal domain, the Hessian detector of Willems et al. [2008] is a space-time extension of the Hessian blob detector [Beaudet, 1978].

Trajectory features. In an early psychophysics study, [Johansson, 1973] shows that humans are able to recognize actions by looking only at the motion of a few moving light displays attached to the body of the person. Inspired by this observation, research efforts have been directed into using trajectories—sequence of points tracked in time—to capture the intrinsic dynamics of video data. Compared to the spatio-temporal features, which treat the spatio-temporal volumes identically across the three axes, trajectories model the temporal dimension separately from the spatial ones. Some ways in which the feature points are tracked include using the KLT tracker Messing et al. [2009], Matikainen et al. [2009], or matching SIFT descriptors from consecutive frames Sun et al. [2009]. The resulting trajectories are often described in terms of their velocity [Song et al., 2003, Fanti et al., 2005, Messing et al., 2009], position [Song et al., 2003, Fanti et al., 2005], curvature [Rao et al., 2002], but also local-appearance [Fanti et al., 2005], saliency or shape [Gorelick et al., 2007].

Tracking feature points with long-term trajectories [Sand and Teller,

2008, Brox and Malik, 2010] is often expensive and unreliable due to occlusion, light changes and motion blur. Some approaches have shown promising results by using shorter trajectories (tens of frames) and describing them with local features [Matikainen et al., 2009, Wang et al., 2009]. The short-term trajectories are robust to drifting problems and they are based on existing optical flow algorithms [Lucas and Kanade, 1981, Farnebäck, 2003]. The method of Wang et al. [2013a] describes local information across a fixed small number of frames using histograms of optical flow, histograms of oriented gradients and motion boundary histograms. As opposed to most spatio-temporal features, they do not rely on a interest point detector, but sample the feature points densely. This approach results in more data to process, but it also leads to better results. The patches that are missed by the interest point detector can be discriminant for the task at hand, so it is recommended to rely on the power of the statistical methods (encoding and classifier steps) to automatically learn the right patterns. The dense trajectory method Wang et al. [2013a] is further detailed in Section 3.1.

Despite camera motion will influence the mentioned features, very few approaches consider it when extracting feature trajectories for action recognition. Uemura et al. [2008] combine feature matching with image segmentation to estimate the dominant camera motion, and then separate feature tracks from the background. Wu et al. [2011] apply a low-rank assumption to decompose feature trajectories into camera-induced and object-induced components. Gaidon et al. [2013] use efficient image-stitching techniques to compute the approximate motion of the background plane and generate stabilized videos before extracting dense trajectories [Wang and Schmid, 2013] for activity recognition.

2.1.2 Encoding methods

The goal of the encoding step is to aggregate local video features into a global vector representation. Being able to represent a video by a fix-sized vector allows us to leverage standard classification algorithms. Methods like logistic regression or support vector machines have been studied for a long time, they are well understood and many quality implementations are available. Another advantage of the encoding step is its efficiency: the memory footprint of the aggregated vector is usually smaller than of the local features extracted from a video. Encoding methods are also appealing because they are agnostic to the type of features. They encode in the same manner different types of features and methods developed for the image domain can be reused for video data with few changes.

We distinguish three stages for a quantization-based encoding technique:

1. Finding the most representative cluster centers in the feature space;
2. Assigning local features to the selected cluster centers;
3. Modeling the statistics of the assigned features.

To illustrate these three steps, let us consider some examples. The bag of words encoding (BOW; Csurka et al., 2004) chooses the clusters by optimizing the k-means objective function, assigns the feature points to the closest cluster (hard quantization), and models the space by counting the number of points that were assigned to each cluster. The final representation is the vector of counts, whose size is equal to the number of centroids K. In the literature the cluster centers are also referred to as *words* or *visual words*. The soft-assignment encoding (SA; Van Gemert et al., 2010) is the soft version of BOW: it models the space using the Gaussian mixture model (GMM), assigns to each cluster the feature points according to their posterior probability (soft quantization), and models the space by adding the posterior probabilities for each cluster. The final representation has again a size equal to the number of centroids K. The Fisher vector encoding (FV; Sánchez et al., 2013) is in turn a generalization of the soft-assignment encoding, as it models the space by taking into account richer statistics: the mean and the variance of the assigned features in addition to the sum of posterior probabilities. The FV representation has a size of $K + 2DK$, where D is the size of the features and hence the size of the mean and variance vectors. We further detail and evaluate these three encodings in Chapter 3. In Table 2.1 we overview the main encodings presented in this section.

Other popular encoding schemes are the vector of locally aggregated descriptors (VLAD; Jégou et al., 2012) and the super vector coding (SVC; Zhou et al., 2010). Both of them can be viewed as particular cases of the FV encoding—they model the space using only the first-order moment, *i.e.*, the mean of the feature points. In addition, VLAD is based on the k-means clustering and uses hard quantization.

Inspired by the success of sparse modeling of image patches for image denoising and inpainting, a series of encoding methods based on sparse representations have been proposed. The super vector coding (SVC) of Yang et al. [2009] is similar to BOW, but learns the visual words using an objective function regularized by a sparsity-inducing term. The locality-constrained linear encoding (LLC) of Wang et al. [2010] assigns a feature to the closest M words and it expresses the feature as a linear combination of

Method	Learning	Assign.	Modeling	Pooling	Dimen.
BOW	k-means	hard	counts	average	K
SA	GMM	soft	counts	average	K
FV	GMM	soft	counts, mean, variance	average	$K + 2DK$
VLAD	k-means	hard	mean	average	DK
SVC	sparse coding[1]	sparse	linear combination	maximum	K
LLC	sparse coding[2]	sparse	linear combination	maximum	K

[1] The SVC objective is similar to the k-means objective, but uses an ℓ_1 regularization.
[2] As opposed to SVC, the LLC objective enforces the locality of the selected words.

Table 2.1 – Summary of the main characteristics of the presented encodings.

those M words. The final representation aggregates the linear combination coefficients of the features using the max-pooling operator.

The pooling operation refers to how encodings of individual local descriptors are aggregated together to a single vector representation. The typical pooling operators are the sum and maximum functions. Sum pooling is particularly appropriate for quantization-based encodings (*e.g.*, BOW, SA, FV): encoding the features from the whole image is equivalent to sum pooling the encodings belonging to the splits of the image. Max-pooling is typical applied to sparsity-based encodings [Boureau et al., 2010b] and it is motivated by biological evidence [Serre et al., 2005]. That said, there is nothing that restricts us from doing max-pooling on quantization-based methods or alternatively average pooling on sparsity-based encodings, see [Boureau et al., 2010a].

As shown in [Perronnin et al., 2010], normalizing the encoding impacts significantly the final performance. While the theoretical motivation of different normalizations is not always evident, from a practical point of view the normalizations can introduce non-linearities in the data, which increase the discriminative capabilities of the pipeline. The power normalization [Perronnin et al., 2010] is an example of such a non-linear transformation: each entry of the encoding is sign-squared rooted, that is, transformed by the function $f(x) = \mathrm{sgn}(x)\sqrt{|x|}$. The power normalization is motivated by the phenomenon of *burstiness* of visual words—a visual word that appears in a video, it is more likely to occur again. So the idea is to weight less the more frequent visual words than the less frequent ones. Another common way of normalizing is by ensuring constant norm for the encoding of each sample. We can ℓ_1-normalize non-negative representations (such as BOW) and use them with distances between distributions (such as χ^2 distance). The ℓ_2 normalization projects the data

on the unit sphere, trying to avoid the problem of meaningless distances in high-dimensional spaces. We discuss these normalizations in more detail in Chapter 4, where we propose ways of approximating them for efficiency reasons.

All these encoding techniques have been benchmarked for image classification [Chatfield et al., 2011] as well video classification [Wang et al., 2012b, Peng et al., 2014]. Wang et al. [2012b] survey the encoding methods together with different pooling approaches (maximum and average) and normalizations (ℓ_1 and ℓ_2). Peng et al. [2014] investigates also the impact of various fusion approaches of low-level features. All three evaluation papers show FV as the best performing encoding.

The Fisher vector encoding is based on a probabilistic model—the encoding consists of the derivatives of the log-likelihood w.r.t. its parameters [Jaakkola and Haussler, 1999]. The FV encoding that we previously mentioned, of Sánchez et al. [2013], is based on the GMM model. Recently, multiple variants of FVs have been proposed by considering other probabilistic models. Sun and Nevatia [2013] models classifier scores on temporal slices by using a FV computed on the transition probabilities of an HMM model. Cai et al. [2014] incorporates features from various sources in a principled manner by deriving FV on the probabilistic canonical correlation analysis. Motivated by the fact that local features are not accurately modeled by the Gaussian distribution, but by distributions with heavier tails, Klein et al. [2014] derive FV using the Laplacian distribution and propose two models: a Laplacian mixture model and a hybrid Gaussian-Laplacian mixture model. More complex models employ Bayesian ideas, that is, treat parameters as random variables, and compute FV as the derivative of the log-likelihood w.r.t. to the hyper-parameters [Cinbis et al., 2012, Liu et al., 2014]. The log-likelihood is usually intractable in such cases and, hence, it is approximated. Cinbis et al. [2012] consider a Bayesian GMM and approximate the log-likelihood using variational inference techniques. They show that their model is complex enough to capture non-iid patterns of the data. Liu et al. [2014] propose a Gaussian distribution with a randomly generated mean vector. They show that this model is equivalent to using a GMM with an infinite number of components and obtain the Fisher vector by solving a sparse coding problem.

The VLAD encoding has been shown to perform on par with FV on image retrieval tasks, while being more compact [Jégou et al., 2012]. However, more effort has been put into further improving the representation. Wu et al. [2014] give a new interpretation to the VLAD encoding as the Maximum Entropy principle for feature learning and they also motivate

the normalizations (square rooting of descriptors increase their normality) and in the light of these observations propose a new normalization and compression technique. Yang and Tian [2014] propose an encoding that merges ideas from VLAD and sparse coding. As for the VLAD, they use the first-order moment information, but they use sparse coding for the assignment step. Sparse coding is also used to estimate the coefficients for each feature and then they are aggregated using average pooling.

The step of finding the cluster centers or, more generally, basis vectors is also known in the literature as *dictionary learning* or *codebook learning*. The methods that we discussed use unsupervised methods, *e.g.*, *k*-means, GMM. Intuitively, optimizing over the partitioning of the feature space for the task at hand should improve the performance. For example, Van der Maaten [2011] and Sydorov et al. [2014] use FV and optimize jointly over the Gaussian parameters and the classifier. The methods differ in the type of classifier used—metric learning NCA for Van der Maaten [2011], SVM for Sydorov et al. [2014]—, but the optimization is similar in both cases, as it consists of two alternating steps: optimizing the classifier and learning the dictionary. Peng et al. [2014] employ supervised dictionary learning for VLAD. They optimize a cross-entropy objective function in the context of classification w.r.t. centroid locations.

Influenced by the recent success of convolutional neural networks (which we discuss in Subsection 2.1.3), deeper approaches have been proposed in the context of encodings. The idea is to use the multiple layers of encoding steps, by feeding the output of an encoding method to the input of another encoding method. Simonyan et al. [2013b] use two layers of FV. They encode FVs, which are extracted from rectangular image cells, with another FV. A metric learning step is interposed between the layers to reduce the dimensionality of the first FV layer. This was necessary to keep the dimensionality of the final representation to a manageable size. Peng et al. [2014] reuse the ideas, but with two main distinctions. They consider action recognition instead of image classification and they pool the first FV layer on 3D cuboids (instead of 2D image patches). The work of Taralova et al. [2014] addresses also the task of action recognition, but they use the BOW encoding for both layers and supervoxels as a pooling support for the first BOW layer.

2.1.3 Deep learning

The local feature methods presented in Subsection 2.1.1 rely on incorporating into their design prior knowledge about desired invariances, *e.g.*, w.r.t. rotation, illumination and scale. An alternative direction is to learn

these invariances from data. This method was advocated by the neural network community, but its utility was hampered by the lack of training data and limited computing resources. Now that these problems start to be overcome, neural networks have regained attention.

The first neural networks methods applied for the task of action recognition were unsupervised and they were generalizations to 3D of previous techniques for 2D images: Taylor et al. [2010] extends convolutional restricted Boltzmann machines to the 3D spatio-temporal domain; Le et al. [2011] proposes a two-stack convolutional independent subspace analysis. They claim that it is possible to learn from data optical flow and motion sensitive features that are similar to the brain's V1 receptive fields.

Krizhevsky et al. [2012] won the ImageNet competition for image classification using a deep convolutional neural network (CNN)—a sequence of three types of layers (convolutional, non-linear and pooling) that ends up with a fully-connected layer used for classification. Their method obtained about 40% relative improvement over the second ranked method, which was relying on a combination of low-level features encoded with Fisher vectors (FV). Following this impressive result, deeper and supervised neural networks have been applied to action recognition also. Karpathy et al. [2014] uses a similar CNN to [Krizhevsky et al., 2012] for action recognition and evaluates several fusion techniques over the temporal dimension. Both Ji et al. [2013] and Tran et al. [2014] use CNN with spatio-temporal 3D filters, but Tran et al. [2014] uses a deeper architecture (eight layers compared to three) and trains on the full frame, as opposed of using human detections as Ji et al. [2013].

However, the best results were obtained using CNN on optical flow instead of appearance [Simonyan and Zisserman, 2014, Yue-Hei Ng et al., 2015]. Simonyan and Zisserman [2014] proposes to use two CNNs: the input to the first network is the appearance frame and the input to the second network is a stack of densely computed optical flow features. The CNNs are trained separately and the final prediction is the weighted average of the individual predictions. Interestingly, they have shown that deep learning provides complementary information to the usual local features, and that the best performance is obtained when combining both.

2.2 Event recognition

The event recognition task has been popularized by the TRECVID multimedia event detection (MED) competitions organized by National Institute of Standards and Technology (NIST) since 2010. The end goal of

the competition is to provide searching capabilities for users in multimedia video material. The videos released in the TRECVID dataset depict events, such as, *birthday party*, *changing a vehicle tire*, *making a sandwich*). Compared to actions, which are defined as a sequence of movements, events are characterized more by semantics than the dynamics of the action. Hence contextual cues (*e.g.*, scene, audio) are important when analyzing such videos. Some examples: in a *birthday party* video we expect to hear people singing "Happy birthday!"; in a *changing a vehicle tire* video the visual instructions are accompanied by audio instructions and textual descriptions; in a *making a sandwich* video the action usually takes place in a kitchen.

2.2.1 Complementary features

Given the importance of contextual features, going beyond motion features proves crucial for good performance for event recognition tasks. The top competitors on the TRECVID challenge [Jiang et al., 2010, Natarajan et al., 2012, Aly et al., 2013] complement motion with other low-level features (appearance, audio) and high-level optical character recognition and speech recognition features.

The appearance features capture static information about the texture and the shape of the scene. The most commonly used appearance features are local features that were successfully used in object recognition tasks, like scale invariant feature transform (SIFT; Lowe, 2004) or speeded-up robust features (SURF; Bay et al., 2008). Global features that capture the general shape of the scene are also often used; these include GIST descriptors [Oliva and Torralba, 2001] or self-similarity descriptors [Shechtman and Irani, 2007]. Color descriptors encode similar information to SIFT or SURF, but they have been claimed to give a better invariance to illumination [Van de Sande et al., 2010].

The single most popular audio features are the mel-frequency cepstral coefficients (MFCC; Rabiner and Juang, 1993). They are versatile features, being used for a number of tasks, such as speech recognition, speaker recognition, music retrieval. MFCC are based on the *cepstrum*—a nonlinear spectrum of a spectrum—, and use frequency bands that emulate the human auditory system.

Working on multiple features raises the question of how to combine multiple channels in order to obtain a single score. This is an important question, because not all features are equally relevant and using noisy features can adversely affect performance. We consider the typical pipeline of local features $\rightarrow$ encoding $\rightarrow$ classifier, and we organize the fusion

techniques into three types, according to the place where we fuse the features:

Early fusion Combine the features before the encoding step;

Mid-level fusion Encode separately each feature and combine the encoding before the classification step;

Late fusion Encode separately each feature and obtain a separate score for each feature and combine the classification scores.

Probably the most commonly used fusion technique is the mid-level fusion. Because many classification techniques rely on kernels to encode similarities between pairs of samples, mid-level fusion corresponds to averaging the kernel values across features; hence, in the literature is it often referred to as *kernel averaging* [Gehler and Nowozin, 2009]. The multiple kernel learning (MKL; Gönen and Alpaydın, 2011) algorithms are popular extensions of kernel averaging and they aim at learning a kernel combination weighting. More advanced MKL methods combine the kernels in an hierarchical fashion [Bach, 2009, Hwang et al., 2012, Tang et al., 2013]. Among those, Tang et al. [2013] consider the task of event recognition. They learn a AND-OR tree for each concept. The leaves of the AND-OR tree are the kernel matrices corresponding to each feature, where the AND nodes average the kernels, the OR nodes select one of multiple kernels. Peng et al. [2014] evaluate the three types of fusions for action recognition and conclude that on average the mid-level fusion performs best. Myers et al. [2014] compare nine late fusion techniques for the task of event recognition and obtain the best results with the simple arithmetic mean.

2.2.2 High-level features

Humans often describe objects or actions in terms of their constituent parts. A forehand shot is "a shot made by swinging the racket across one's body with the hand moving palm-first." If we are able to detect that there is a racket in the scene and its movement follows a trajectory across the body of the tennis player, then there are high chances that we are witnessing a forehand shot. One line of research employs exactly this idea—instead of directly predicting the class label of a video, first predict labels for more atomic classes (also known as *attributes*), and based on them make the final decision.

The idea of decomposing an action into primitives is not new. Early neurobiological experiments suggest that the human visual system perceives the visual input of an action as a sequence of motor primitives.

Inspired by these results, researchers seek to define hierarchies of action. Bobick [1997] identifies three components of the hierarchy: movements, activities and actions. Movements are the most atomic primitives, they require no contextual knowledge to be recognized; activities are sequences of movement; actions are described on top of the activities and they exhibit causal relationships or interactions with the environment. The work of Intille and Bobick [1998] on American football recognition is an early example of system that decomposes actions. Their primitives are movements such as *run in front*, *catch pass*, *turn toward*. These are connected in a rule-based Bayes network whose constraints are manually specified.

The power of high-level features lies in their interpretability. This allows us not only to name the class, but also to describe more precisely what is happening in a video. Many approaches use high-level features to generate natural language from video [Kojima et al., 2002, Barbu et al., 2012, Das et al., 2013, Rohrbach et al., 2013]. These methods use action primitives, but also object or person detectors. The sentence generation process often reduces to filling in template sentences with nouns and verbs corresponding to the detected objects and actions. But recently more data-driven approaches have been used. For example, Rohrbach et al. [2013] adapt machine translation algorithms to link the semantic representation of attributes to textual captions, which were gathered from a separate corpus.

Another use of attributes is reducing the amount of data needed for training. Farhadi et al. [2009] show that attributes significantly boost performance when learning with few examples. Lampert et al. [2009b] extend classifiers to classes that were not represented in the train set. This is possible because the attributes are shared across classes and each class can be manually described in terms of attributes.

The *action bank* method [Sadanand and Corso, 2012] represents a video with high-level features by embedding it into a space spanned by the action classifier scores. The action classifiers are applied on video subvolumes of various scales and then aggregated into a vector representation. Merler et al. [2012] introduces the equivalent of action bank for event recognition, the *concept bank*. Similarly to the action bank it provides an intermediate level semantic representation, but it uses a bank of semantically broader and more diverse concepts (*e.g.*, *outdoors*, *construction*, *clouds*).

An active research topic is how to choose the basis of attributes. For actions it is common to use fine grained motions [Liu et al., 2011, Sadanand and Corso, 2012], while for events the detectors refer to coarser concepts [Merler et al., 2012]. In an overview study, Habibian et al. [2013] explore

the type, the specificity and the quality of the detectors in the context of event recognition. Their best practice advice is to aim for increasing the number of concepts rather than improving the quality of the concept detectors. They also recommend to build a diverse concept bank—objects, actions, scenes, people, animals—, and include both general and specific concepts.

Several recent papers propose learning the most relevant concepts for each event: Ma et al. [2013b] optimize jointly over the concept classifiers and event classifiers; Mazloom et al. [2014] use cross-entropy optimization to select the most informative concepts for each event; Zheng et al. [2014] propose a greedy feature selection approach that optimizes a sub-modular function; they show that their algorithm gives a nearly-optimal solution.

2.2.3 Structured models

The previously described methods—based on encodings like bag of words or Fisher vector—are oblivious to the feature ordering. Videos as well as images are, however, highly structured signals. Thus, it is not surprising that many approaches incorporate this prior information into the models.

An effective way of going beyond the orderless representations is the spatial pyramid matching (SPM; Lazebnik et al., 2006). The idea is to encode separately features belonging to different regions of the image and then concatenate them into the final representation. SPM is particularly suitable when categories share similar layout structure for example, scene images. Laptev et al. [2008] show that this is also the case for videos. They extend SPM for action recognition and find out that partitioning the video in three horizontal stripes performs best.

The standard SPM pools the features over cells of a grid, but other more flexible ways of partitioning the image have been investigated. For example, Bilen et al. [2014] learn the cell sizes of a two-by-two grid by inferring the intersection point of the vertical and horizontal splits. Other ways of generalizing SPM is by learning weights for each cell [Bosch et al., 2007] or by encoding the local features' locations [Krapac et al., 2011, Yang and Tian, 2014].

For event recognition, latent variable models are often used to explicitly model sub-events (their presence, duration, relative order). Li et al. [2013a] and Vahdat et al. [2013] observe that the videos are not precisely cropped around the action of interest. In order to remedy this problem, they use latent models to infer which temporal segments are relevant for each video and event. Tang et al. [2012] use a variable-length dis-

criminative HMM model which infers latent sub-actions together with a non-parametric duration distribution. Izadinia and Shah [2012] use a tree-structured CRF to model co-occurrence relations among sub-events and complex event categories, but require additional labeling of the sub-events unlike Tang et al. [2012].

More sophisticated models have been proposed for the action recognition task. These usually aim to explicitly capture the spatial and temporal structure of actions [Matikainen et al., 2010, Gaidon et al., 2011, Brendel and Todorovic, 2011, Wang et al., 2014a]. Other authors have focused on modeling interactions between people and objects [Gupta et al., 2009, Prest et al., 2013], or used multiple instance learning to suppress irrelevant background features [Sapienza et al., 2012].

2.3 Localization

We have described in the first chapter of the thesis two types of action localization tasks. Most of the recent papers on action localization fall into one of these categories: temporal localization [Duchenne et al., 2009, Niebles et al., 2010, Gaidon et al., 2011], spatio-temporal localization [Kläser et al., 2010, Tran and Yuan, 2011, Lan et al., 2011, Tran and Yuan, 2012, Tian et al., 2013, Wang et al., 2014b, Jain et al., 2014].

Other variations of action localization do exist, but are not standard. Yuan et al. [2009] and Cao et al. [2012] aim to find the cuboid that best covers the action. This task is called *sub-volume search* and, in terms of localization granularity, it is situated somewhere between temporal and spatio-temporal localization: more precise than temporal localization, but less precise than spatio-temporal localization. Some of the methods for spatio-temporal localization provide finer localizations than bounding boxes, *e.g.*, cell level Lan et al. [2011], pixel-level Jain et al. [2014]. However, the standard datasets are annotated at bounding box level and we will not differentiate these categories.

Another direction of localization task is to use training data that is not precisely annotated. The clips are a rough estimate of when an action occurs and the goal is to refine temporal extents. Some examples include the work of Duchenne et al. [2009], who employ discriminative clustering, and [Satkin and Hebert, 2010], who define a max-margin objective function with temporal extents acting as latent variables.

In the following subsections, we detail the references mentioned above. In Subsection 2.3.1 we present models that were shown to be suitable for

localization and in Subsection 2.3.2 we review ways of dealing with the pressing issue of efficiency.

2.3.1 Models for localization

The most straightforward way of performing localization is by treating it as localized classification. This technique, known as *sliding window*, it applies a classifier function to uniformly extracted windows (either temporal or spatio-temporal) [Duchenne et al., 2009]. The window that yields maximum score should correspond to the action's location.

As we argued in Subsection 2.2.3, going beyond orderless representation is intuitive and often improves performance. Gaidon et al. [2011] use a more structured model for temporal action localization. They model an action as three consecutive sub-actions whose durations are learnt in a non-parametric way. Each of the sub-actions is represented by the BOW encoding.

Other structured models were inspired by object detection systems. Felzenszwalb et al. [2010] revive the pictorial structure models [Fischler and Elschlager, 1973]—a collection of parts with connections between certain pairs of parts—by taking advantage of modern features (histogram of oriented gradients) and statistical methods (latent support vector machines). They propose the deformable part model (DPM), a latent-variable model for object detection. DPM consists of a series of detectors: one for the entire object, *root filter*, and others for its parts, *part filters*. The detectors are combined into a scoring function by considering the maximum individual scores and penalizing the displacement of the parts from an initial configuration.

DPM was extended to both temporal [Niebles et al., 2010] and spatio-temporal localization [Lan et al., 2011, Tian et al., 2013]. The model of Niebles et al. [2010] infers temporal anchor points and scales for the sub-events of each class. Based on these temporal locations, they pool features and then compare them using the χ^2 kernel. Tian et al. [2013] propose a direct extension to the spatio-temporal domain by replacing the HOG filters with the HOG-3D filters of [Kläser et al., 2008]. On the other hand, the work of Lan et al. [2011] draws inspiration from the latent SVM of Felzenszwalb et al. [2010] to treat the spatio-temporal task as a set of image-level detection problems. Their latent variables model the location of the person in each frame and the deformation cost enforces similar locations across time. Lan et al. [2011] replace the DPM's root filter with a global appearance filter, which takes into account the recognition score.

2.3.2 Efficiency

Efficiency is of particular importance in localization. State-of-the-art detection systems evaluate around 100,000 windows for each image [Felzenszwalb et al., 2010]. For object detection (sub-image search) the number of candidate windows grows quadratically with the image size, while for videos (sub-volume search) the growth is even more abrupt—cubic in the video size. In this Subsection, we look at some approaches for improving the efficiency of localization methods.

Cascade filters. A natural idea of improving efficiency is by filtering out the initial pool of candidate windows based on a fast sequence of pre-processing steps. These methods are reminiscent of the face detector of Viola and Jones [2004], which relied on increasing complex classifiers (*cascade*) to quickly discard uninteresting regions. For action localization typical example of filters are human detector that retains only those frames for which a human was detected [Lan et al., 2011, Kläser et al., 2010] or a fast, albeit suboptimal, classifier that drops low-scoring windows [Kläser et al., 2010]. Reducing the initial set of candidate windows, enable the use of more expensive methods: non-linear classifiers in the case of Kläser et al. [2010] or latent models in the case of Lan et al. [2011].

Optimization methods. A different class of efficient localization methods leverages the advances in optimization. Lampert et al. [2009a] use the *branch and bound* algorithm to efficiently detect objects in images. Branch and bound (BB) is a general optimization algorithm that can find the optimal solution of a non-convex problem. In the worst case BB is as expensive as exhaustive search, but we can improve its efficiency if we provide it bounding function—a function that bounds the optimal solution from a subset of the search domain. Based on this bounding function, BB prunes out parts of the search state space that give worse estimates than the current solution. The BB variant of Lampert et al. [2009a] searches over the set of all rectangular sub-images inside the query image, and uses a bounding function derived from the classification scoring function. Cao et al. [2010] and Yuan et al. [2009] extend the method of Lampert et al. [2009a] to 3D and find the maximum scoring sub-volume inside a video. In Section 4.3, we also employ the BB algorithm for the task of action localization.

Dynamic programming is another optimization technique that was employed for spatio-temporal localization. Tran and Yuan [2011] score the cells of the video and connect the neighbouring cells in a lattice graph. Using dynamic programming they find the path that yields the maximum score (they assume the scores to be additive, so the score of a path is the

sum of window scores).

Structured learning is yet another alternative to exhaustive search for localization. Blaschko and Lampert [2008] were first to propose structured learning for object localization and Tran and Yuan [2012] followed later with an extension to spatio-temporal localization. The main idea of these approaches is to regress the coordinates of the window directly from the entire image. As opposed to using binary classification for localization, which scores each window independently, structured learning takes into account the correlations from the output space, *i.e.*, overlapping windows result in similar classification scores. Blaschko and Lampert [2008] use these dependencies to improve performance and efficiency of both the training and testing procedures. For the task of spatio-temporal action localization, Tran and Yuan [2012] consider even more complex output than [Blaschko and Lampert, 2008]: temporal sequences of the rectangle bounds.

Proposals. A class of methods that avoids the windows at every location and scale are the proposal methods. The idea is that from the set of all possible windows only a few are relevant in the sense that they contain an object. Using various image features, we can accurately decide on the quality of the window for localization. Based on this observations, Alexe et al. [2012] define a score that a certain window contains an object. The *objectness* score is based on multiple cues, like saliency, color contrast, edge density, superpixel straddling, which are combined with a Naïve Bayes classifier. Binarized normed gradients (BING; Cheng et al., 2014) are a fast way of approximating the objectness score based on the normalized gradients. They rescale each window to a small patch (8×8 pixels), then they compute the norm of the gradients, which results in a 64D vector and it is used for learning an generic objectness score using a cascaded SVM framework. Zitnick and Dollár [2014] propose a another fast proposal method that uses only edges information and a scoring function that resembles the straddling measure of Alexe et al. [2012]: the number of contours fully contained in the candidate box.

The methods of Van de Sande et al. [2011] and Manen et al. [2013] build proposals based on image segmentations. The selective search method [Van de Sande et al., 2011] constructs a tree of segmentations in a bottom-up and greedy way. They use color histograms and geometrical cues to define a distance between adjacent segments. The proposals are generated as bounding boxes of segments at each node of the tree. Manen et al. [2013], instead, define a randomized procedure to generate propoasals: select a random starting segment, which is then grown with adjacent segments (again, these are randomly selected, using a probability

distribution inversely proportional to the distance function).

Other methods are based on figure-ground segmentation [Rother et al., 2004], a binary segmentation technique that separates foreground objects from their background. These type of proposal methods start with multiple seed regions and generate a separate figure-ground segmentation for each seed. For example, Carreira and Sminchisescu [2012] solve a large number of independent binary min-cut problems on an image grid, at multiple scales. This step results in a set of segments which is then filtered to remove small segments and then scored using using a classifier trained on geometric features. Another related technique is the geodesic object proposals of Krähenbühl and Koltun [2014]. This approach uses the geodesic distance to compute the distance between each pixel and foreground-background masks. The resulting distance map exhibits stable levels and a proposal is generated for each stable level by considering the region that has a distance smaller than the corresponding stable level.

There are only a few extensions of proposal methods for action localization. Among those, Jain et al. [2014] provide the extension of the selective search method [Van de Sande et al., 2011] to video. They start from a video segmentation, the graph-based segmentation of Xu and Corso [2012], and greedily build an hierarchical tree of proposals. Based on these proposals, they are able to do spatio-temporal localization in an efficient manner. In Chapter 5, we present another way of building spatio-temporal object proposals based on the randomized procedure of Manen et al. [2013].

Chapter 3

A robust and efficient video representation for action recognition

Contents

In this chapter we present our pipeline for action recognition. We combine the state-of-the-art local dense trajectory features with the Fisher vectors encoding method, to obtain a holistic video representation. Fisher vectors have been shown to give superior performance over bag of words and other encodings in image classification [Chatfield et al., 2011, Sánchez et al., 2013]. Our experimental results prove that the same conclusion holds for a variety of recognition tasks and datasets in the video domain.

We consider three challenging problems to demonstrate the effectiveness of our framework. First, we consider the classification of basic action categories using six of the most challenging datasets. Second, we consider the localization of actions in feature length movies, including four action classes: *drinking, smoking, sit down*, and *open door* from [Duchenne et al., 2009, Laptev and Pérez, 2007]. Third, we consider classification of

more high-level complex event categories using the TRECVID MED 2011 dataset [Over et al., 2012].

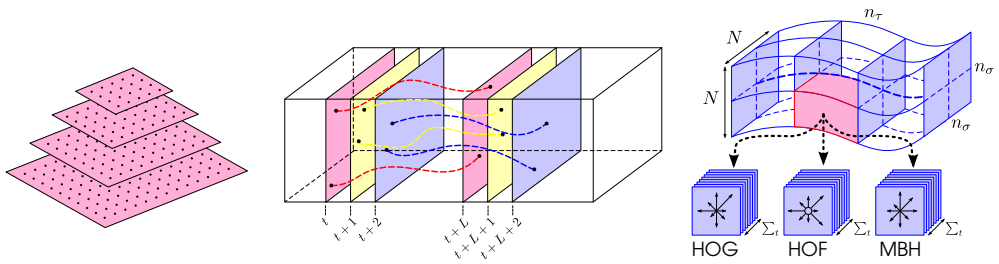
On all three tasks we obtain state-of-the-art performance, improving over earlier work that relies on combining more feature channels, or using more complex models. For action localization in full length movies, we also propose a modified non-maximum-suppression technique that avoids a bias towards selecting short segments. This technique further improves the detection performance.

The chapter is organized as follows. We start by reviewing the local features, the dense trajectories and the improved dense trajectories (Section 3.1), and the feature encoding techniques, the bag of words and Fisher vectors (Section 3.2). Then, in Section 3.3, we discuss classification and explain how to efficiently train and test when dealing with very high-dimensional vectors, such as Fisher vectors. For the action localization task we perform non-maximum-suppression (NMS); in Section 3.4, we propose and motivate new variants of NMS. The datasets and evaluation protocols are described in Section 3.5. Finally, the experimental results are given in Section 3.6.

3.1 Local features

Realistic video data poses a series of challenges: intra-class variation due to style and duration, background and motion clutter, occlusions and camera motion, and the sheer amount of data that needs to be processed. Local features are a robust way of handling these situations. The idea of local features was first popularized for images, where local features—scale invariant feature transform (SIFT; Lowe, 2004), histogram of oriented gradients (HOG; Dalal and Triggs, 2005) or speeded up robust features (SURF; Bay et al., 2008)—have been successfully applied for tasks like classification, retrieval or detection. In videos, the extension of local features are the space-time features [Dollár et al., 2005, Laptev, 2005]. Spatio-temporal features produce robust representations under uncontrolled conditions, as they make no assumptions about the global video structure and they avoid non-trivial pre-processing steps (such as object tracking or motion segmentation).

The success of local space-time features leads to a trend of generalizing classical descriptors from image to video, *e.g.*, 3D-SIFT [Scovanner et al., 2007], extended SURF [Willems et al., 2009], HOG3D [Kläser et al., 2008], and local trinary patterns [Yeffet and Wolf, 2009]. Among the local space-time features, the dense trajectories [Wang et al., 2013a] have been



(a) Dense sampling (b) Tracking the feature points (c) Extracting the descriptors

Figure 3.1 – The steps of computing dense trajectories [Wang et al., 2013a].

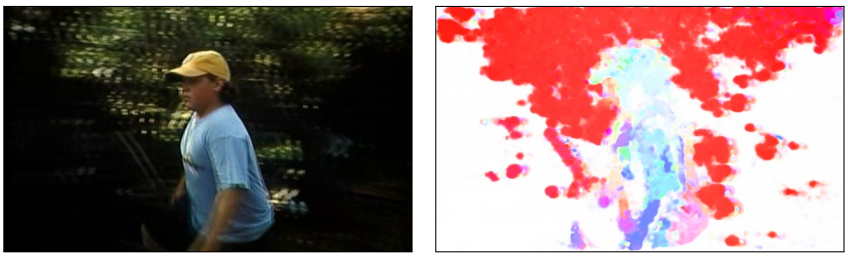
shown to perform the best on a variety of datasets. In the following two subsections, we review the dense trajectory features [Wang et al., 2013a] and their improved variant, which includes camera stabilization [Wang and Schmid, 2013].

3.1.1 Dense trajectory features

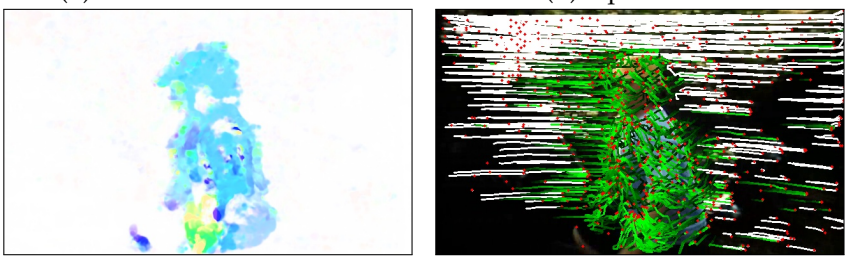
The dense trajectory features approach [Wang et al., 2013a] computes local descriptors along short trajectories. We start by sampling feature points on a dense spatial grid and across several spatial scales (Figure 3.1a). Points in homogeneous areas are suppressed, as it is impossible to track them reliably. The feature points are tracked in a dense optical flow field [Farnebäck, 2003] for only 15 frames in order to avoid drifting; then new points are sampled to replace them (Figure 3.1b). We remove static feature trajectories as they do not contain motion information, and also prune trajectories with sudden large displacements.

For each trajectory, we compute HOG, HOF and MBH descriptors with the same parameters as in [Wang et al., 2013a]; we do not use the trajectory descriptor as it does not improve the overall performance significantly. All three descriptors are computed in the space-time volume aligned with the trajectory (Figure 3.1c). HOG [Dalal and Triggs, 2005] is based on the orientation of image gradients and captures the static appearance information. Both HOF [Laptev et al., 2008] and MBH [Dalal and Triggs, 2005] measure motion information, and are based on optical flow. HOF directly quantizes the orientation of flow vectors. MBH splits the optical flow into horizontal and vertical components, and quantizes the derivatives of each component. The final dimensions of the descriptors are 96 for HOG, 108 for HOF and 2×96 for the two MBH channels.

To normalize the histogram-based descriptors, *i.e.*, HOG, HOF and



(a) Two succesive frames (b) Optical flow



(c) Warped optical flow (d) Removed trajectories

Figure 3.2 – The improved trajectory features stabilize the camera motion in order to enhance the quality of the optical flow and of the trajectories.

MBH, we apply the recent RootSIFT [Arandjelovic and Zisserman, 2012] approach, *i.e.*, square root each dimension after ℓ_1 normalization. We do not perform ℓ_2 normalization as in [Wang et al., 2013a]. This slightly improves the results without introducing additional computational cost.

3.1.2 Improved trajectory features

The improved trajectory features approach [Wang and Schmid, 2013] builds upon the dense trajectory features by adding an additional camera stabilization step. MBH is based on derivatives of optical flow, which is a simple and efficient way to achieve robustness to camera motion. However, MBH suppresses only certain camera motions (for example, translation at constant depth; but not more complex movements like zooming, panning or tilting) and, thus, we can benefit from explicit camera motion estimation. Camera motion generates many irrelevant trajectories in the background in realistic videos. We can prune them and only keep trajectories from humans and objects of interest, if we know the camera motion. Furthermore, given the camera motion, we can correct the optical flow, so that the motion vectors are independent of camera motion. This improves the performance of motion descriptors based on optical flow, *i.e.*, HOF (histograms of optical flow) and MBH. In Figure 3.2 we illustrate

the difference between the original (Figure 3.2b) and the corrected optical flow (Figure 3.2c).

The first stages—the sampling and the tracking of the feature points—are done in the same way as for the dense trajectory features. The video is stabilized by estimating a homography between each pair of consecutive frames. We extract SURF features and motion vectors of optical flow and match them using the RANSAC algorithm. The SURF and motion vectors are complementary features: SURF focuses on blob-type structures, while the motion vectors concentrate on corners and edges. Based on the homography we warp the second frame and recompute the optical flow [Farnebäck, 2003]. The motion descriptors (HOF and MBH) are computed on the re-estimated optical flow, while the appearance descriptor (HOG) remains unchanged. We further utilize these stabilized motion vectors to remove background trajectories (Figure 3.2d): for each trajectory, we compute the maximal magnitude of the motion vectors during its length of 15 frames, if the maximal magnitude is lower than a threshold then the trajectory is considered to be consistent with camera motion, and thus removed.

Videos often focus on the humans performing the action. So it is very common that humans dominate the frame, which can be a problem for camera motion estimation as human motion is in general not consistent with it. Wang and Schmid [2013] propose to use a human detector to remove matches from human regions. We used state-of-the-art human detector [Prest et al., 2012b], which adapts the general part-based human detector [Felzenszwalb et al., 2010] to action datasets. The detector combines several part detectors dedicated to different regions of the human body (including full person, upper-body and face). It is trained using the PASCAL VOC07 training data for humans as well as near-frontal upper-bodies from [Ferrari et al., 2008].

3.2 Feature encoding

The feature extraction step represents a video as a set of features. We aggregate these features into a fixed-dimensional representation using two encoding techniques: the bag of words and the Fisher vector. In order to incorporate spatio-temporal location information of the features we use the spatial pyramid matching and the spatial Fisher vector. All these techniques are detailed below.

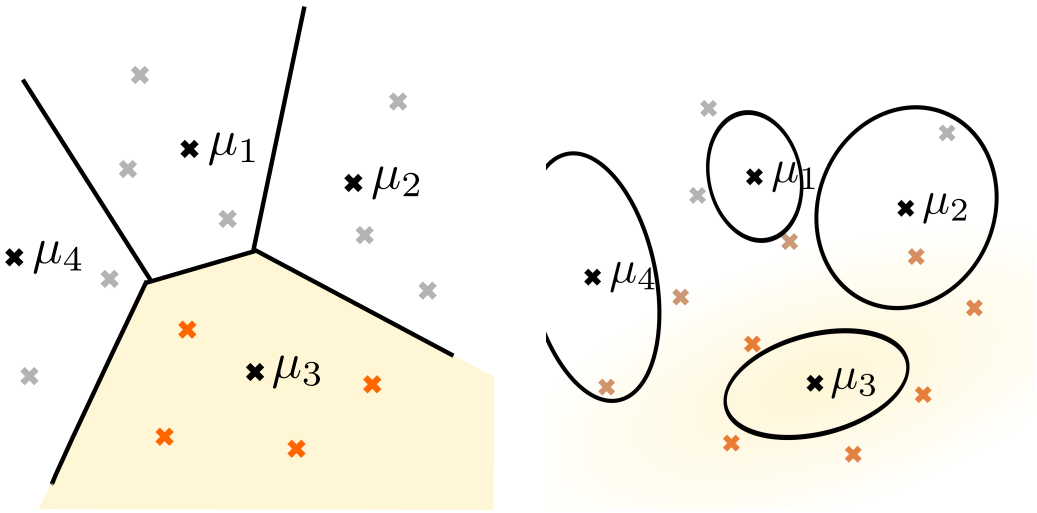
3.2.1 Bag of words

The *bag of words* (BOW) encoding is one of the most popular and successful approaches to encoding local features. It has been extensively employed by the top competitors in the PASCAL VOC [Everingham et al., 2009, 2010, 2011] and TRECVID challenges [Jiang et al., 2010, Natarajan et al., 2011, Wang et al., 2012a] along the years. The term *bag of words*—other analogous formulations are *bag of features*, *bag of visual words* or *bag of keypoints*—has its origins in the natural language processing community and it describes the technique of representing a document as an orderless collection of words by counting how many times each word appears. However, in the computer vision field the first similar methods have their roots in the work on texture classification. In 1981, Julesz published his theory of texture perception: texture, a global structure, can be decomposed into a few, local elementary units, which he called *textons*. The discrimination of different textures is the result of the textons' identity and not of their spatial arrangement. Julesz's ideas re-emerged in the early 2000s, when Malik et al. [1999] defined the texton for gray-level images as frequently co-occurring combinations of linear filter banks. The following research on texture classification [Leung and Malik, 2001, Cula and Dana, 2001] used these ideas: they built a vocabulary of textons and then represented a texture as histogram of textons. While Leung and Malik [2001] used the χ^2 distance to compare histograms, Cula and Dana [2001] used PCA to first project the histograms to a lower dimensional space and then do nearest neighbour retrieval. The next work along these lines had other applications such as category classification of zebras, cheetahs or humans [Schmid, 2001] and material classification [Varma and Zisserman, 2002]. The first two papers that present the bag of words pipeline as it is currently known are [Sivic and Zisserman, 2003] and [Csurka et al., 2004].

The bag of words encoding has two main steps:

Vocabulary training The first step is an unsupervised, training step in which we find the visual words. This step is usually done using the k-means clustering algorithm. Figure 3.3a illustrates the partitioning of the feature space; the K visual words are indicated as μ_k.

Quantization The second step represents the encoding: for each visual word we count how many feature points are most similar to it; in Figure 3.3a, we show the feature points assigned to a particular visual word (μ_3). The bag of words representation is the count histogram. A common normalization technique is to divide the histogram counts by the number of features; this insures that the representation is invariant to the size of the video.



(a) The bag of words (BOW) encoding. The feature space is partitioned into non-overlapping regions and for each cluster we count how many points are assigned to it.

(b) The Fisher vector (FV) encoding. The features space is modelled by a Gaussian mixture model and for each cluster we represent the soft-assigned points using their mean and variance.

Figure 3.3 – Illustration of the feature encoding methods (bag of words and the Fisher vector). Each point represents a local feature and the points indicated by μ_k denote the K visual words.

In recent years, many variants of bag of words have been proposed. For our experiments, we replace the k-means with Gaussian mixture model training and use soft quantization [Van Gemert et al., 2010]. Instead of counting the number of features assigned to a particular visual word, we sum the posterior probabilities. Soft-assignments have been reported to yield better performance, and we can share the GMM vocabulary with the Fisher vector encoding, thus making the results easier to compare.

3.2.2 Fisher vector

The Fisher vector (FV) encoding is the successor of bag of words. In a recent evaluation study of feature pooling techniques for object recognition [Chatfield et al., 2011] FV was found to be the most effective encoding technique. This evaluation also included bag of words (BOW), sparse coding techniques, and several variants. The Fisher vector encoding method was first proposed by Jaakkola and Haussler [1999] to combine the flexibility of generative models and the performance of discriminative classifiers.

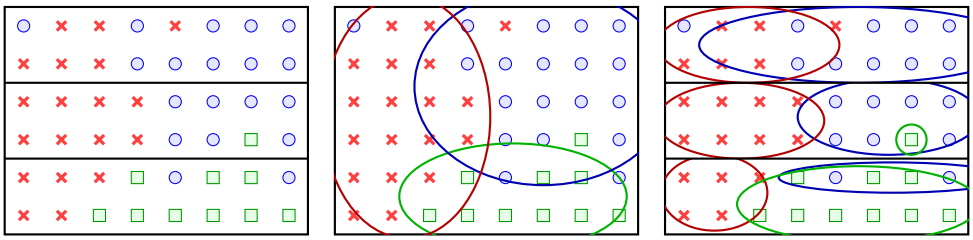
One can derive the Fisher vector of a generative model by computing the gradients of the log-likelihood with respect to the model's parameters. Jaakkola and Haussler [1999] exemplified the Fisher vector for the hidden Markov model in the context of DNA sequence alignment; they showed to obtain better results by using FV with discriminative models rather than using the HMM model directly.

In computer vision, FV was popularized through the work of Perronnin and Dance [2007], where they considered FV of the Gaussian mixture model (GMM) model. Recently the FV has appeared as the start-of-the-art representation for a wide variety of problems, including image retrieval [Jégou et al., 2012], face verification [Simonyan et al., 2013a], word spotting [Almazan et al., 2013], fine-grained image classification [Gavves et al., 2013], object detection [Cinbis et al., 2013], semantic segmentation [Li et al., 2013b], and texture classification [Cimpoi et al., 2014].

FV extends the BOW representation as it encodes both first and second-order statistics between the video descriptors and a diagonal covariance Gaussian mixture model (GMM). Higher-order statistics is the key ingredient of FV and it is what offers its main advantage over BOW: the ability to encode the rich space of descriptors in a compact, hence, efficient way. As we will see later in the experimental part (Section 3.6.1) the BOW representation can achieve comparable performance to FV, but at the expense of working with very large codebooks (several orders of magnitude larger than FV).

Another characteristic of FV is the use of GMM instead of the k-means clustering. BOW encodes each descriptor to one of the existing words in the k-means vocabulary (see Figure 3.3a). This hard assignment is prone to be impacted by small changes in the descriptor space. In the FV case, the video descriptors are soft-assigned to each Gaussian (see Figure 3.3b) making the encoding step more robust.

In the following we provide the mathematical formulation for the FV representation. Given a video, let $x_n \in \mathbb{R}^D$ denote the n-th D-dimensional video descriptor, q_{nk} the soft-assignment of x_n to the k-th Gaussian, and π_k, μ_k and σ_k are the weight, mean, and diagonal of the covariance matrix of the k-th Gaussian respectively. After normalization with the inverse Fisher information matrix (which renders the FV invariant to the parametrization) the D-dimensional gradients w.r.t. the mean and variance of the k-th



(a) Spatial pyramid matching (SPM). (b) Spatial Fisher vector (SFV). (c) Combination of SPM and SFV.

Figure 3.4 – Schematic illustration of how the spatio-temporal information is encoded. The densely sampled features are assigned to one of the visual words (○, ×, □) and the location of each visual word is modelled either using a fixed grid (left), Gaussian (middle) or both (right).

Gaussian are given by:

$$G_{\mu_k} = \sum_{n=1}^{N} q_{nk} \left[x_n - \mu_k \right] / \sqrt{\sigma_k \pi_k}, \tag{3.1}$$

$$G_{\sigma_k} = \sum_{n=1}^{N} q_{nk} \left[(x_n - \mu_k)^2 - \sigma_k^2 \right] / \sqrt{2\sigma_k^2 \pi_k}. \tag{3.2}$$

For each descriptor type x_n, we can represent the video as a $2DK$ dimensional Fisher vector. To compute FV, we first reduce the descriptor dimensionality by a factor of two using principal component analysis (PCA), as in [Sánchez et al., 2013]. We then randomly sample a subset of $1000 \times K$ descriptors from the training set to estimate a GMM with K Gaussians.

3.2.3 Weak spatio-temporal location information

To go beyond a completely orderless representation of the video content in a BOW histogram or FV, we consider including a weak notion of spatio-temporal location information of the local features. For this purpose, we use the spatio-temporal pyramid (STP) representation [Laptev et al., 2008], and compute separate BOW or FV over cells in spatio-temporal grids (Figure 3.4a). We also consider the spatial Fisher vector (SFV) of [Krapac et al., 2011], which computes per visual word the mean and variance of the 3D spatio-temporal location of the assigned features (Figure 3.4b). This is similar to extending the (HOG, HOF or MBH) feature vectors with the 3D locations, as done in [McCann and Lowe, 2012,

Sánchez et al., 2012]; the main difference being that the latter do clustering on the extended feature vectors while this is not the case for the SFV. SFV is also computed in every cell of STP (Figure 3.4c). To combine SFV with BOW or FV, we simply concatenate them together.

3.3 Classification

The tasks that we are considering—action and event recognition, localization of action in movies—can all be cast as supervised classification problems. We are given a training set with video samples and their associated class label and, at test time, for each unlabeled video sample we need to automatically predict the classes it belongs to. For the action localization task, a test sample is an entire movie and we ought to not only name the action class, but also find all the temporal intervals where it is occurring. We use the sliding window technique (generate smaller video samples at uniform locations and of different lengths), the task is to evaluate independently all these windows; this follows exactly the same procedure as the standard classification scenario.

As the classification method, we choose a discriminative classifier, the support vector machine (SVM, Cortes and Vapnik 1995). The SVM objective function maximizes the margin of the hyperplane that separates two sets of data. The key advantages of SVM are (i) the possibility of using kernel functions; (ii) the objective function is convex, hence the optimization achieves the global minimum; (iii) the strong generalization performance. There are many SVM implementations available; we use the LIBSVM package [Chang and Lin, 2011].

For BOW we consider both the linear and RBF-χ^2 kernel, while for FV we consider only the linear kernel. When using the linear kernel we apply power and ℓ_2 normalization on the feature vectors (BOW or FV) as in [Perronnin et al., 2010, Sánchez et al., 2013]; in the case of the RBF-χ^2 kernel we use ℓ_1 normalization. To combine different descriptor types, we encode each descriptor type separately and concatenate their normalized BOW or FV representations together.

In the case of multi-class classification, we use a one-against-rest approach and select the class with the highest score. For the SVM hyper-parameters, we set the class weight to be inversely proportional to the number of samples in each class so that both positive and negative classes contribute equally in the loss function. We set the regularization parameter C by cross validation on the training set, by testing values in the range

$C \in \{3^{-2}, 3^{-1}, \cdots, 3^{7}\}$. In all experiments, we use the same settings as stated above.

Implementation note. We train the classifier in the dual space by pre-computing the kernel matrix. The FV dimensionality, D, is usually in the order of hundreds of thousands, and it is much higher than the number of training samples, N, which, in our case, does not exceed a few thousands. So, it is more memory-efficient to use the kernel matrix, of size $N \times N$, than the design train matrix, of size $N \times D$. For a linear classifier, the kernel is just the inner product of the design matrix and the concatenation of features corresponds to kernel addition. So one can compute the kernels for each feature independently and then just add them together.

At test time it is, however, more efficient to operate in the primal than in the dual. To obtain the predictions in the primal we first need to compute the primal weights, a D-dimensional vector. These are obtained as a dot product between the N-dimensional vector of dual coefficients and the training matrix; the cost of this operation is $O(ND)$. The scores on the test data are obtained by another dot product between the primal weights and the test data, of size $M \times D$; this step has $O(MD)$ complexity. The total cost of predicting in the primal is $O(ND + MD)$. On the other hand, to score the data in the dual we first compute the kernel matrix by multiplying the test and train data, which scales in $O(NMD)$. We then compute the dot product between the dual coefficients and the kernel matrix to obtain the predictions; this step results in a cost of $O(NM)$. The total cost of working in the dual is $O(NMD + NM)$.

3.4 Non-maximum-suppression for localization

For the action localization task we employ a temporal sliding window approach. We score a large pool of candidate detections that are obtained by sliding windows of various lengths across the video. Non-maximum suppression (NMS) is performed to delete windows that have an overlap greater than 20% with higher scoring windows. In practice, we use candidate windows of length 30, 60, 90, and 120 frames, and slide the windows in steps of 30 frames.

Preliminary experiments on the Coffee and Cigarettes dataset showed that there is a strong tendency for the NMS to retain short windows, see Figure 3.5. This is due to the fact that if a relatively long action appears,

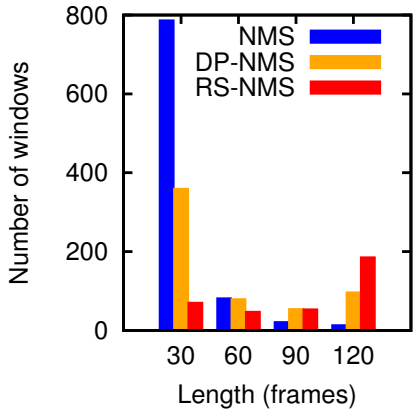
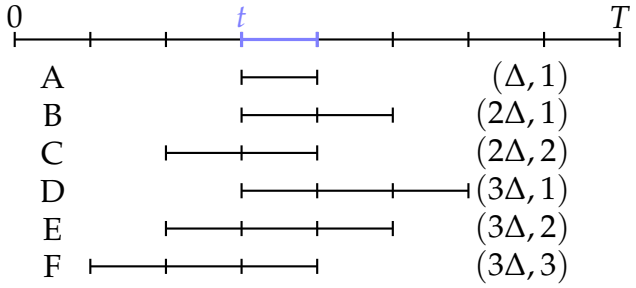


Figure 3.5 – Histograms of the window sizes on the *Coffee and Cigarettes* dataset after three variants of non-maximum suppression: classic non-maximum suppression (NMS), dynamic programming non-maximum suppression (DP-NMS), and re-scored non-maximum suppression (RS-NMS). Two of the methods, NMS and DP-NMS, select mostly short windows, 30-frames long, while the RS-NMS variant sets a bias towards longer windows, 120-frames long. In practice we prefer longer windows as they tend to better cover the action.

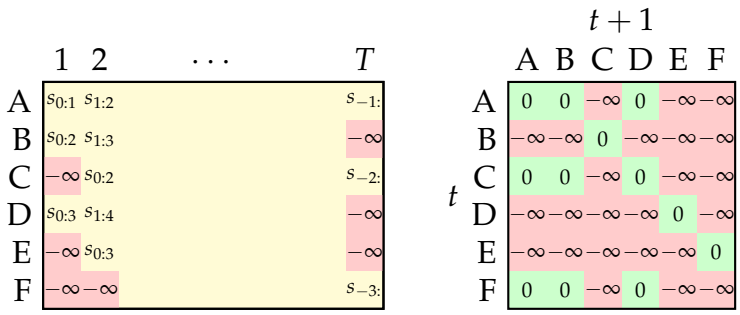
it is likely that there are short sub-sequences that just contain the most characteristic features for the action. Longer windows might cover the action better, but are likely to include less characteristic features as well (even if they lead to positive classification by themselves), and might include background features due to imperfect temporal alignment.

NMS greedily selects the highest scoring windows, being oblivious to their duration. For example, if a short window scores just a bit more than a longer window, the longer window is suppressed and the shorter one kept. We propose a version of NMS that in such cases it prefers the selection of longer windows. To achieve this, we define a global objective function that corresponds to the area under the scoring curve of the selected windows.

We formalize this NMS variant as follows. We first divide the temporal domain into T discrete temporal slices. With each temporal slice t we associate a latent state z_t, which is a window characterized by its duration and the position where the given slice t occurs. Examples of latent states are shown in Figure 3.6a. A pairwise potential $\psi(z_t, z_{t+1})$ is used to enforce that two consecutive states, z_t and z_{t+1}, are consistent: if the current selected state z_t does not correspond to a *ending window*—a



(a) Latent states for a given temporal slice t (highlighted in blue). The latent state (denoted by capital letters from A to F) are characterized by two values: their duration and the position where the temporal slice t occurs. We denote by Δ the duration of the base time slice.



(b) Potentials for the latent states described above (A to F). The left matrix shows the unary potentials. The entries correspond to the window scores for each temporal slice, from 1 to T; $s_{b:e}$ denotes the score of a window starting at slice b and lasting until slice $e-1$. The right matrix indicates the pairwise state transitions, from state t to state $t+1$. We indicate by 0 (green color) the possible transitions and by $-\infty$ (red color) the disallowed transitions. All possible transitions are equally likely.

Figure 3.6 – The optimization problem for dynamic programming NMS (DP-NMS).

window for which the slice t appears in the last position—, then the next state z_{t+1} must be a continuation of that previous window, that is, have the same duration, but an incremented position; otherwise, if the current state z_t corresponds to a ending window, the next state z_{t+1} has to be a starting window. Figure 3.6b, right, shows the transition costs between consecutive latent states, as defined in Figure 3.6a. We maximize the score based on an unary potential $\phi(z_t)$that is defined as the score of the window associated with the given state z_t (see Figure 3.6b, left). Formally, we optimize the following objective function:

$$\underset{z_1,\cdots,z_T}{\text{maximize}} \sum_{t=1}^{T} \phi(z_t) + \psi(z_t, z_{t+1}), \tag{3.3}$$

where z_t are the latent state variables; ϕ are the unary potential representing the window scores; and ψ are the binary potentials, which do not impact the final score, but enforce the consistency constraint. We use the dynamic programming Viterbi algorithm to compute the optimal solution for the problem defined by Eq. (3.3), using a forwards and backwards pass over the temporal slices. The runtime is linear in the number of temporal slices, T. We refer to this method as *dynamic programming NMS* or DP-NMS.

We have defined the optimization problem for DP-NMS such that if a window is selected then its score is counted in the objective function proportionally to the window's duration. Based on this property we define another NMS variant, similar to the original NMS in the sense that it maximizes the score locally. The new NMS variant, which we refer to as *rescored NMS* or RS-NSM, multiplies the score of each slice by their duration and then applies standard NMS.

Figure 3.5 shows the histogram of durations of the windows that pass the non-maximum suppression stage using the different techniques, for the action *smoking* used in our experiments in Section 3.6.2. The durations for the two proposed methods, DP-NMS and RS-NMS, have a more uniform distribution than that for the standard NMS method, with RS-NMS favouring the longest windows. This behaviour is also observed in Figure 3.7, which gives an example of the different windows retained for a specific video segment of the *Coffee & Cigarettes* movie. DP-NMS selects longer windows than NMS; these windows maintain a high score over a long period, as desired by the objective function, but they do not align well with the action and the scores of the negative windows are high. For this example, RS-NMS gives the best selection among the three methods, as it one long segment that covers the action accurately.

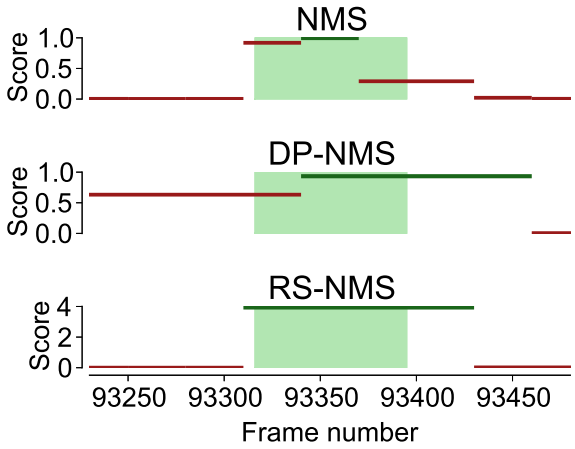


Figure 3.7 – Examples of window selection for the three variant of non-maximum suppression: classic non-maximum suppression (NMS), dynamic programming non-maximum suppression (DP-NMS), and re-scored non-maximum suppression (RS-NMS). The green region denotes the ground-truth action. For the NMS, the segments selected are too short. The DP-NMS selects longer segments, but it does not align well with the true action as it maximizes the total score over the whole video. The RS-NMS strikes a good balance of the segment's length and their score, and it gives the best solution in this example.

3.5 Datasets used for experimental evaluation

In this section, we briefly describe the datasets and their evaluation protocols for the three tasks. We use six challenging datasets for action recognition (*i.e.*, Hollywood2, HMDB51, Olympic Sports, High Five, UCF50 and UCF101), *Coffee and Cigarettes* and DLSBP for action localization, and TRECVID MED 2011 for large scale event detection. In Figure 3.8, we show some sample frames from the datasets.

3.5.1 Action recognition

The **Hollywood2** dataset [Marszałek et al., 2009] has been collected from 69 different Hollywood movies and includes 12 action classes. It contains 1,707 videos split into a training set (823 videos) and a test set (884 videos). Training and test videos come from different movies. The performance is measured by mean average precision (mAP) over all

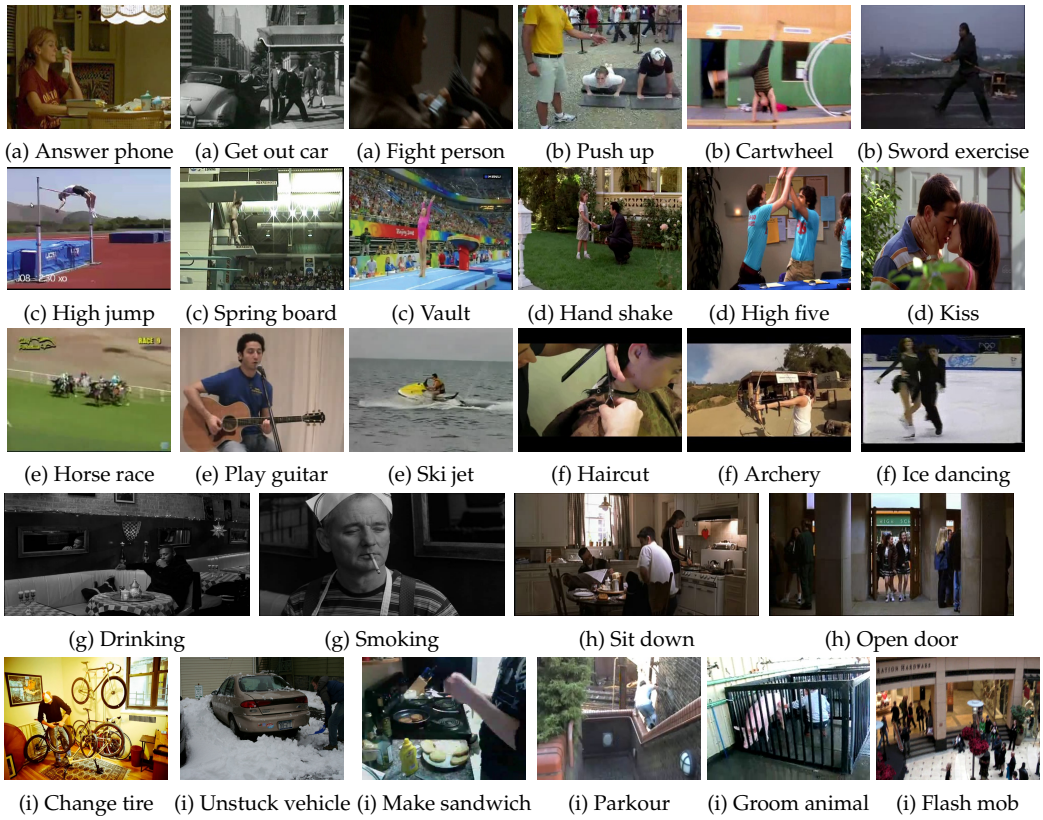


Figure 3.8 – From top to bottom, example frames from (a) Hollywood2, (b) HMDB51, (c) Olympic Sports, (d) High Five, (e) UCF50, (f) UCF101, (g) *Coffee and Cigarettes*, (h) DLSBP and (i) TRECVID MED 2011.

classes, as in [Marszałek et al., 2009].

The **HMDB51** dataset [Kuehne et al., 2011] is collected from a variety of sources ranging from digitized movies to YouTube videos. In total, there are 51 action categories and 6,766 video sequences. We follow the original protocol using three train-test splits [Kuehne et al., 2011]. For every class and split, there are 70 videos for training and 30 videos for testing. We report average accuracy over the three splits as performance measure. Note that in all the experiments we use the original videos, not the stabilized ones.

The **Olympic Sports** dataset [Niebles et al., 2010] consists of athletes practicing different sports, which are collected from YouTube and annotated using Amazon Mechanical Turk. There are 16 sports actions (such as high-jump, pole-vault, basketball lay-up, discus), represented by a total of 783 video sequences. We use 649 sequences for training and 134

sequences for testing as recommended by the authors. We report mAP over all classes, as in [Niebles et al., 2010].

The **High Five** dataset [Patron-Perez et al., 2010] consists of 300 video clips extracted from 23 different TV shows. Each of the clips contains one of four interactions: hand shake, high five, hug and kiss (50 videos for each class). Negative examples (clips that don't contain any of the interactions) make up the remaining 100 videos. Though the dataset is relatively small, it is challenging due to large intra-class variation, and all the action classes are very similar to each other (*i.e.*, interactions between two persons). We follow the original setting in [Patron-Perez et al., 2010], and compute average precision (AP) using a pre-defined two-fold cross-validation.

The **UCF50** dataset [Reddy and Shah, 2013] has 50 action categories, consisting of real-world videos taken from YouTube. The actions range from general sports to daily life activities. For all 50 categories, the videos are split into 25 groups. For each group, there are at least four action clips. In total, there are 6,618 video clips. The video clips in the same group may share some common features, such as the same person, similar background or viewpoint. We apply the leave-one-group-out cross-validation as recommended in [Reddy and Shah, 2013] and report average accuracy over all classes.

The **UCF101** dataset [Soomro et al., 2012] is extended from UCF50 with additional 51 action categories. In total, there are 13,320 video clips. We follow the evaluation guidline from the THUMOS'13 workshop [Jiang et al., 2013b] using three train-test splits. In each split, clips from seven of the 25 groups are used as test samples, and the rest for training. We report average accuracy over the three splits as performance measure.

3.5.2 Action localization

The first dataset for action localization is extracted from the movie **Coffee and Cigarettes**, and contains annotations for the actions *drinking* and *smoking* [Laptev and Pérez, 2007]. The training set contains 41 and 70 examples for each class respectively. Additional training examples (32 and eight respectively) come from the movie *Sea of Love*, and another 33 lab-recorded *drinking* examples are included. The test sets consist of about 20 minutes from *Coffee and Cigarettes* for *drinking*, with 38 positive examples; for *smoking* a sequence of about 18 minutes is used that contains 42 positive examples.

The **DLSBP** dataset [Duchenne et al., 2009] contains annotations for the actions *sit down*, and *open door*. The training data comes from 15 movies, and contains 51 *sit down* examples, as well as 38 for *open door*.

The test data contains three full movies (*Living in Oblivion*, *The Crying Game*, and *The Graduate*), which in total last for about 250 minutes, and contain 86 *sit down*, and 91 *open door* samples.

To measure performance we compute the average precision (AP) score as in [Duchenne et al., 2009, Gaidon et al., 2011, Kläser et al., 2010, Laptev and Pérez, 2007]; considering a detection as correct when it overlaps (as measured by intersection over union) by at least 20% with a ground truth annotation.

3.5.3 Event recognition

The **TRECVID MED 2011** dataset [Over et al., 2012] is the largest dataset we consider. It consists of consumer videos from 15 categories that are more complex than the basic actions considered in the other datasets, *e.g.*, *changing a vehicle tire*, or *birthday party*. For each category between 100 and 300 training videos are available. In addition, 9,600 videos are available that do not contain any of the 15 categories; this data is referred to as the *null* class. The test set consists of 32,000 videos, with a total length of over 1,000 hours, and includes 30,500 videos of the *null* class.

We follow two experimental setups in order to compare our system to previous work. The first setup is the one described above, which was also used in the TRECVID 2011 MED challenge. The performance is evaluated using average precision (AP) measure. The second setup is the one of Tang et al. [2012]. They split the data into three subsets: EVENTS, which contains 2,048 videos from the 15 categories, but doesn't include the *null* class; DEV-T, which contains 602 videos from the first five categories and the 9,600 *null* videos; and DEV-O, which is the standard test set of 32,000 videos.[1] As in [Tang et al., 2012], we train on the EVENTS set and report the performance in AP on the DEV-T set for the first five categories and on the DEV-O set for the remaining ten actions.

The videos in the TRECVID dataset vary strongly in size: durations range from a few seconds to one hour, while the resolution ranges from low quality 128×88 to full HD 1920×1080. We rescale the videos to a width of at most 480 pixels, preserving the aspect ratio, and temporally sub-sample them by discarding every second frame in order to make the dataset computationally more tractable. These rescaling parameters were selected on a subset of the MED dataset; we present an exhaustive evaluation of the impact of the video resolution in Section 3.6.3. Finally, we

1. The number of videos in each subset varies slightly from the figures reported in [Tang et al., 2012]. The reason is that there are multiple releases of the data. For our experiments, we used the labels from the `LDC2011E42` release.

also randomly sample the generated features to reduce the computational cost for feature encoding. This is done only for videos longer than 2000 frames, *i.e.*, the sampling ratio is set to 2000 divided by the total number of frames.

3.6 Experimental results

Below, we present our experimental evaluation results for action recognition in Section 3.6.1, for action localization in Section 3.6.2, and for event recognition in Section 3.6.3.

3.6.1 Action recognition

We first compare bag of words (BOW) and Fisher vectors (FV) for feature encoding, and evaluate the performance gain due to different motion stabilization steps. Then, we assess the impact of removing inconsistent matches based on human detection, and finally compare to the state of the art.

Feature encoding with BOW and FV

We begin our experiments with the original non-stabilized MBH descriptor [Wang et al., 2013a] and compare its performance using BOW and FV under different parameter settings. For this initial set of experiments, we chose the Hollywood2 and HMDB51 datasets as they are widely used and are representative in difficulty and size for the task of action recognition. We evaluate the effect of including weak geometric information using the spatial Fisher vector (SFV) and spatio-temporal pyramids (STP). We consider STP grids that divide the video in two temporal parts (T2) and three spatial horizontal parts (H3). When using STP, we always concatenate the representations (*i.e.*, BOW or FV) over the whole video. For the case of T2+H3, we concatenate all six BOW or FV representations (one for the whole video, two for T2, and three for H3). Unlike STP, the SFV has only a limited effect for FV on the representation size, as it just adds six dimensions (for the spatio-temporal means and variances) for each visual word. For the BOW representation, the situation is different, since in that case there is only a single count per visual word, and the additional six dimensions of the SFV multiply the signature size by a factor seven; similar to the factor six for STP.

Tables 3.1 and 3.2 list all the results using different settings on Hollywood2 and HMDB51. It is obvious that increasing the number of Gaussians K leads to significant performance gain for both BOW and FV. However, the performance of FV tends to saturate after $K = 256$, whereas BOW keeps improving up to $K = 1024$. This is probably due to the high dimensionality of FV which results in an earlier saturation. Both BOW and FV benefit from including STP and SFV, which are complementary since the best performance is always obtained when they are combined.

As expected, the RBF-χ^2 kernel works better than the linear kernel for BOW. Typically, the difference is around 4-5% on both Hollywood2 and HMDB51. When comparing different feature encoding strategies, the FV usually outperforms BOW by 6-7% when using the same number of visual words. Note that FV of 64 visual words is even better than BOW of 1024 visual words; confirming that for FV fewer visual words are needed than for BOW.

We further explore the limits of BOW performance by using very large vocabularies, *i.e.*, with K up to $32,768$; the results are shown in Figure 3.9. For BOW, we use χ^2 kernel and T2+H3 which give the best results on both Hollywood2 (Table 3.1) and HMDB51 (Table 3.2). For a fair comparison, we only use T2+H3 for FV without SFV. On both Hollywood2 and HMDB51, the performance of BOW becomes saturated when K is larger than $8,192$. If we compare BOW and FV representations with similar dimensions (*i.e.*, $K = 32,768$ for BOW and K between 64 and 128 for FV), FV still outperforms BOW by 2% on HMDB51 and both have comparable performance for Hollywood2. Moreover, feature encoding with large vocabularies is very time-consuming as shown in Figure 3.10, where $K = 32,768$ for BOW is eight times slower than $K = 128$ for FV. This can impose huge computational cost for large datasets such as TRECVID MED. FV is also advantageous as it achieves excellent results with a linear SVM which is more efficient than kernel SVMs. Note however, that the classifier training time is negligible compared to the feature extraction time, *e.g.* it takes around 200 seconds for FV with $K = 256$ to compute the Gram matrix and to train the classifiers on the Hollywood2 dataset. This is roughly equivalent to extracting features for 200 frames (about 10 seconds of video).

To sum up, we choose FV with both STP and SFV, and set $K = 256$ for a good compromise between accuracy and computational complexity. We use this setting in the rest of experiments unless stated otherwise.

		Bag-of-words			Fisher vector	
		χ^2 kernel	linear kernel		linear kernel	
K	STP	BOW	BOW	BOW+SFV	FV	FV+SFV
64	—	44.4	39.8	40.3	55.0	56.5
64	H3	48.0	44.9	45.0	57.9	59.2
64	T2	48.3	43.4	46.8	57.1	58.5
64	T2+H3	50.2	46.8	46.4	59.4	59.5
128	—	45.8	42.1	43.5	57.1	58.5
128	H3	51.3	46.2	48.1	58.8	60.0
128	T2	50.5	45.5	49.4	58.8	59.9
128	T2+H3	52.4	48.4	48.2	61.0	60.7
256	—	49.4	44.9	45.9	57.9	59.6
256	H3	52.9	46.0	50.6	59.0	61.0
256	T2	52.0	47.0	51.3	59.3	60.3
256	T2+H3	53.6	50.2	50.2	61.0	61.3
512	—	50.2	46.8	49.0	58.9	60.5
512	H3	53.1	49.5	51.2	59.5	61.5
512	T2	53.9	49.4	52.8	60.2	61.0
512	T2+H3	55.5	51.6	51.3	**61.7**	**61.9**
1024	—	52.3	48.5	50.4	58.9	60.9
1024	H3	55.6	50.6	52.6	59.4	61.2
1024	T2	54.6	52.0	**54.5**	59.7	60.7
1024	T2+H3	**56.6**	**52.9**	53.5	61.2	61.8

Table 3.1 – Comparison on the Hollywood2 dataset of bag of words and Fisher vectors using the non-stabilized MBH descriptor under different parameter settings. We use ℓ_1 normalization for the χ^2 kernel, and power and ℓ_2 normalization for the linear kernel.

		Bag-of-words			Fisher vector	
		χ^2 kernel	linear kernel		linear kernel	
K	STP	BOW	BOW	BOW+SFV	FV	FV+SFV
64	—	30.5	28.3	28.0	45.8	47.9
64	H3	35.8	30.1	33.1	48.0	49.4
64	T2	34.9	30.9	32.5	48.3	49.5
64	T2+H3	37.1	32.5	34.2	50.3	51.1
128	—	33.8	31.9	32.2	48.2	50.3
128	H3	38.0	32.3	37.5	49.9	51.1
128	T2	38.2	32.9	36.2	50.2	51.1
128	T2+H3	40.5	35.8	37.9	51.9	52.6
256	—	36.6	33.1	35.0	50.0	51.9
256	H3	40.6	36.2	40.4	51.4	52.3
256	T2	41.3	35.7	39.7	51.5	52.0
256	T2+H3	43.5	39.2	41.2	52.6	53.2
512	—	40.3	35.6	37.9	51.3	53.2
512	H3	43.4	38.4	41.5	51.4	52.3
512	T2	42.6	39.1	42.2	52.2	53.3
512	T2+H3	45.2	42.1	43.5	52.7	53.7
1024	—	42.3	39.2	39.9	51.4	53.9
1024	H3	45.4	40.8	44.2	51.7	52.8
1024	T2	46.0	41.8	**46.3**	52.5	53.0
1024	T2+H3	**47.5**	**43.9**	45.7	**53.3**	**53.8**

Table 3.2 – Comparison on the HMDB51 dataset of bag of words and Fisher vectors using the non-stabilized MBH descriptor under different parameter settings. We use ℓ_1 normalization for the χ^2 kernel, and power and ℓ_2 normalization for the linear kernel.

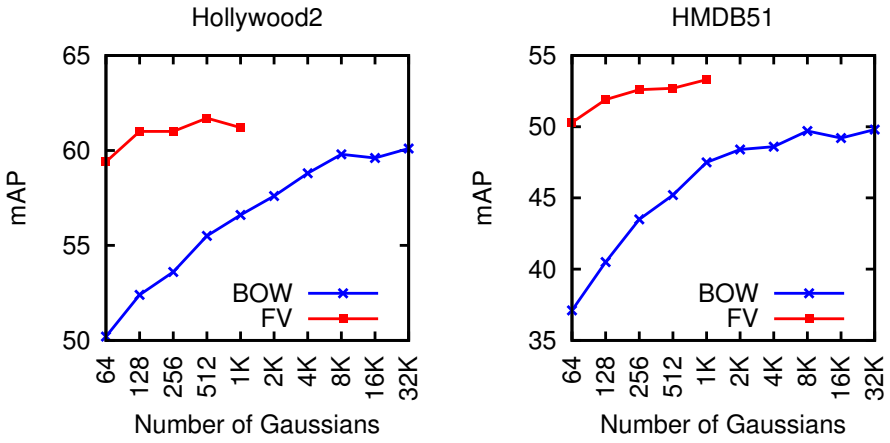


Figure 3.9 – Comparing the mAP performance of the bag of words (RBF-χ^2 kernel) and the Fisher vectors encoding (linear kernel) as a function of the number of Gaussians (K). For both cases we use non-stabilized MBH descriptors and include spatio-temporal information with STP (T2+H3), but no SFV.

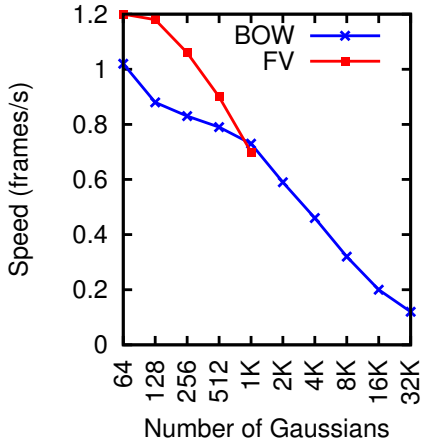


Figure 3.10 – Comparing the speed (number of frames processed in a second on a single core) of the bag of words (RBF-χ^2 kernel) and the Fisher vectors encoding (linear kernel) as a function of the number of Gaussians (K). For both cases we use non-stabilized MBH descriptors and include spatio-temporal information with STP (T2+H3), but no SFV. We use a video from the Hollywood2 dataset; it had a resolution of 720×480 pixels.

Evaluation of local features

We compare the two types of low-level features that we described in Section 3.1: the dense trajectories and the improved trajectories. For the improved trajectories we use two variants by either using or not the human detector. This gives us three features that we compare: (i) the dense trajectories (DT), (ii) the improved trajectories that do not use of the human detector (IT), (iii) the improved trajectories that use automatic human detection (ITH). For all three features we compute HOG, HOF and MBH descriptors as described in Section 3.1.1, and report results on all the combinations of them. For the versions of the improved trajectories we both warp the optical flow with the homography corresponding to the camera motion and remove background trajectories consistent with the homography. The results are presented in Table 3.3 for Hollywood2 and HMDB51.

In the following, we discuss the results per descriptor. The results of HOG are similar for different variants on both datasets. Since HOG is designed to capture static appearance information, we do not expect that compensating camera motion significantly improves its performance.

HOF benefits the most from stabilizing optical flow. On Hollywood2, the improvements are around 5%. On HMDB51, the improvements are even higher: around 10%. After motion compensation, the performance of HOF is comparable to that of MBH.

MBH is known for its robustness to camera motion [Wang et al., 2013a]. However, its performance still improves as optical flow improves and the motion boundaries become clearer. We have over 2% improvement on both datasets.

Combining HOF and MBH further improves the results as they are complementary to each other. HOF represents zero-order motion information, whereas MBH focuses on first-order derivatives. Combining all three descriptors achieves the best performance, as shown in the last row of Table 3.3.

Comparison to the state of the art

Table 3.4 compares our method with the most recent results reported in the literature. On Hollywood2, all presented results [Jain et al., 2013, Jiang et al., 2012, Mathe and Sminchisescu, 2012, Zhu et al., 2013] improve dense trajectories in different ways. Mathe and Sminchisescu [2012] prune background features based on visual saliency. Zhu et al. [2013] apply multiple instance learning on top of dense trajectory features in order to

HOG	HOF	MBH	Hollywood2			HMDB51		
			DT	IT	ITH	DT	IT	ITH
•			51.3	51.8	53.0	42.0	43.1	44.4
	•		56.4	61.8	62.4	43.3	52.3	52.3
		•	61.3	63.3	63.6	53.2	56.2	56.9
•	•		61.9	63.9	65.3	51.9	57.0	57.5
•		•	63.0	63.9	64.7	**56.3**	57.8	58.7
	•	•	62.0	65.7	65.2	53.2	58.3	58.3
•	•	•	**63.6**	**66.1**	**66.8**	55.9	**59.3**	**60.1**

Table 3.3 – Comparison of the local descriptor approaches—dense trajectories (DT), improved trajectories (IT), and improved trajectories using human detector (ITD)—and various descriptor combinations (HOG, HOF, MBH). These experiments use SFV+STP, $K = 256$.

learn mid-level "acton" to better represent human actions. Recently, Jain et al. [2013] report 62.5% by decomposing visual motion to stabilize dense trajectories. We further improve their results by over 4%.

HMDB51 [Kuehne et al., 2011] is a relatively new dataset. Jiang et al. [2012] achieve 40.7% by modeling the relationship between dense trajectory clusters. Ballas et al. [2013] report 51.8% by pooling dense trajectory features from regions of interest using video structural cues estimated by different saliency functions. The best previous result is from [Zhu et al., 2013]. We improve it further by over 5%, and obtain 60.1% accuracy.

Olympic Sports [Niebles et al., 2010] contains significant camera motion, which results in a large number of trajectories in the background. Li et al. [2013a] report 84.5% by dynamically pooling feature from the most informative segments of the video. Wang et al. [2013b] propose motion atom and phrase as a mid-level temporal part for representing and classifying complex action, and achieve 84.9%. Gaidon et al. [2013] model the motion hierarchies of dense trajectories [Wang et al., 2013a] with tree structures and report 85.0%. Using our pipeline, we outperform them by over 5%.

High Five [Patron-Perez et al., 2010] focuses on human interactions and serves as a good testbed for various structure models applied to action recognition. Ma et al. [2013a] propose hierarchical space-time

Hollywood2		HMDB51	
Jiang et al., 2012	59.5	Jiang et al., 2012	40.7
Mathe and Sminchisescu, 2012	61.0	Ballas et al., 2013	51.8
Zhu et al., 2013	61.4	Jain et al., 2013	52.1
Jain et al., 2013	62.5	Zhu et al., 2013	54.0
Dense traj.	63.6	Dense traj.	55.9
Improved traj.	66.1	Improved traj.	59.3
Improved traj. (human det.)	**66.8**	Improved traj. (human det.)	**60.1**
Olympic Sports		**High Five**	
Li et al., 2013a	84.5	Ma et al., 2013a	53.3
Wang et al., 2013b	84.9	Yu et al., 2012	56.0
Gaidon et al., 2013	85.0	Gaidon et al., 2013	62.4
Dense traj.	85.8	Dense traj.	62.5
Improved traj.	89.6	Improved traj.	68.1
Improved traj. (human det.)	**90.4**	Improved traj. (human det.)	**69.4**
UCF50		**UCF101**	
Shi et al., 2013	83.3	Peng et al., 2013	84.2
Wang et al., 2013b	85.7	Murthy and Goecke, 2013b	85.4
Ballas et al., 2013	**92.8**	Karaman et al., 2013	85.7
Dense traj.	89.1	Dense traj.	83.5
Improved traj.	91.3	Improved traj.	85.7
Improved traj. (human det.)	91.7	Improved traj. (human det.)	**86.0**

Table 3.4 – Comparison of our results (HOG+HOF+MBH) to the state of art. We present our results for FV encoding ($K = 256$) using SFV+STP. Best result for each dataset is marked in bold.

segments as a new representation for simultaneous action recognition and localization. They only extract the MBH descriptor from each segment and report 53.3% as the final performance. Yu et al. [2012] propagate Hough voting of STIP [Laptev et al., 2008] features in order to overcome their sparseness, and achieve 56.0%. With our framework we achieve 69.4% on this challenging dataset.

UCF50 [Reddy and Shah, 2013] can be considered as an extension of the widely used YouTube dataset [Liu et al., 2009]. Recently, Shi et al.

[2013] report 83.3% using randomly sampled HOG, HOF, HOG3D and MBH descriptors. Wang et al. [2013b] achieve 85.7%. The best result so far is 92.8% from Ballas et al. [2013]. We obtain a similar accuracy of 91.7%.

UCF101 [Soomro et al., 2012] is used in the recent THUMOS'13 Action Recognition Challenge [Jiang et al., 2013b]. All the top results are built on different variants of dense trajectory features [Wang et al., 2013a]. Karaman et al. [2013] extract many features (such as HOG, HOF, MBH, STIP, SIFT, *etc*.) and do late fusion with logistic regression to combine the output of each feature channel. Murthy and Goecke [2013b] combine ordered trajectories [Murthy and Goecke, 2013a] and improved trajectories [Wang and Schmid, 2013], and apply Fisher vector to encode them. With our framework we obtained 86.0%, and ranked first among all 16 participants.

3.6.2 Action localization

In our second set of experiments we consider the localization of four actions (*i.e.*, *drinking*, *smoking*, *open door* and *sit down*) in feature length movies. We use the same encoding paramters as for action recognition: $K = 256$ for Fisher vector with SFV+STP. We first consider the effect of different NMS variants using the improved trajectory features without human detection. We then compare with the local features and discuss the impact of human detection. Finally we present a comparison to the state-of-the-art methods.

Evalution of NMS variants

We report all the results by combining HOG, HOF and MBH together, and present them in Table 3.5. The dynamic programming version (DP-NMS) is better than standard NMS on a single class and its overall performance is slightly inferior to standard NMS. On the other hand, simple rescoring (RS-NMS) significantly improves over standard NMS on two out of four classes. By construction DP-NMS does not allow any overlap, so we also test the other two methods, NMS and RS-NMS, with zero overlap. The results show that for standard NMS zero or 20% overlap does not significantly change the results on all four action classes, while for RS-NMS zero overlap is beneficial on all classes. RS-NMS zero overlap performs the best among all five different variants, thus we use it in the remainder of the experiments.

Method	Overlap	Drinking	Smoking	Open door	Sit Down	Average
NMS	20	73.2	32.3	23.3	28.6	39.3
RS-NMS	20	76.5	38.0	23.2	26.6	41.1
DP-NMS	0	71.4	36.7	21.0	23.6	38.2
NMS	0	74.1	32.4	24.2	**28.9**	39.9
RS-NMS	0	**80.2**	**40.9**	**26.0**	27.1	**43.5**

Table 3.5 – Evaluation of the non-maximum suppression variants: classic non-maximum suppression (NMS), dynamic programming non-maximum suppression (DP-NMS), and re-scored non-maximum suppression (RS-NMS). The overlap parameter (second column) indicates the maximum overlap (intersection over union) allowed between any two windows after non-maximum suppression. We use HOG+HOF+MBH from improved trajectory features (without human detector) with FV ($K = 256$) augmented by SFV+STP.

Evaluation of local features

We present detailed experimental results in Table 3.6. We analyze all the combinations of the three descriptors and compare the dense trajectory features (DT) to the improved trajectory features, without (IT) and with human detection (ITH),

We observe that combining all descriptors usually gives better performance than individual descriptors. The improved trajectory features are outperformed by the dense trajectory features on three out of four classes for the case of HOG+HOF+MBH. Note that the results of different descriptors and settings are less consistent than they are on action recognition datasets, *e.g.*, Table 3.3, as here we report the results for each class separately. Furthermore, since the action localization datasets are much smaller than action recognition ones, the number of positive examples per category is limited, which renders the experimental results less stable. In randomised experiments, where we leave one random positive test sample out from the test set, we observe standard deviations of the same order as the differences between the various settings (not shown for sake of brevity).

HOG	HOF	MBH	Drinking			Smoking		
			DT	IT	ITH	DT	IT	ITH
•			44.3	52.7	51.5	31.0	32.9	33.9
	•		82.5	79.2	79.1	28.9	34.7	33.9
		•	78.7	73.0	70.4	**47.7**	**48.7**	43.2
•	•		80.8	**81.1**	**79.9**	35.5	33.5	33.0
•		•	78.2	74.3	75.0	40.5	42.7	42.3
	•	•	**85.0**	79.0	78.3	46.8	45.7	**45.0**
•	•	•	81.6	80.2	79.0	38.5	40.9	39.4
HOG	HOF	MBH	Open door			Sit down		
			DT	IT	ITH	DT	IT	ITH
•			21.6	23.8	21.4	14.9	14.3	14.3
	•		21.4	19.8	23.9	25.5	25.5	23.8
		•	29.5	23.4	22.9	26.1	25.8	25.6
•	•		20.9	27.5	26.9	24.1	21.9	22.6
•		•	**29.6**	**30.2**	**29.2**	28.3	25.0	25.2
	•	•	28.8	23.4	23.8	**30.6**	**27.2**	27.1
•	•	•	28.8	26.0	26.4	29.6	27.1	**27.6**

Table 3.6 – Comparison of dense trajectory features (DT) to the improved trajectory features, without (IT) and with human detection (ITH). We use Fisher vector ($K = 256$) with SFV+STP to encode local descriptors, and apply RS-NMS-0 for non-maximum suppression. We show results on two datasets: the *Coffee & Cigarettes* dataset [Laptev and Pérez, 2007] (*drinking* and *smoking*) and the DLSBP dataset [Duchenne et al., 2009] (*Open Door* and *Sit Down*).

As for the impact of human detection, surprisingly leaving it out performs better for *drinking* and *smoking*. Since *Coffee & Cigarettes* essentially consists of scenes with static camera, this result might be due to inaccuracies in the homography estimation.

	Drinking	Smoking	Open door	Sit Down
Laptev and Pérez, 2007	49.0	—	—	—
Duchenne et al., 2009	40.0	—	14.4	13.9
Kläser et al., 2010	54.1	24.5	—	—
Gaidon et al., 2011	57.0	31.0	16.4	19.8
RS-NMS zero overlap	**80.2**	**40.9**	**26.0**	**27.1**

Table 3.7 – Improved trajectory features without human detection compared to the state of the art for localization. We use HOG+HOF+MBH descriptors encoded with FV ($K = 256$) and SFV+STP, and apply RS-NMS zero overlap for non-maximum suppression.

Comparison to the state of the art

In Table 3.7, we compare our RS-NMS zero overlap method with previously reported state-of-the-art results. As features we use HOG+HOF+MBH of the improved trajectory features, but without human detection. We obtain substantial improvements on all four action classes, despite the fact that previous work used more elaborate techniques. For example, Kläser et al. [2010] relied on human detection and tracking, while Gaidon et al. [2011] requires finer annotations that indicate the position of characteristic moments of the actions (actoms). The biggest difference comes from the *drinking* class, where our result is over 23% better than that of [Gaidon et al., 2011].

3.6.3 Event recognition

In our last set of experiments we consider the large-scale TRECVID MED 2011 event recognition dataset. For this set of experiments, we do not use the human detector during homography estimation. We took this decision for practical reasons: running the human detector on 1,000 hours of video would have taken more than two weeks on 500 cores; the speed is about 10 to 15 seconds per frame on a single core. We also leave out the T2 split of STP, because of both performance and computational reasons. We have found on a subset of TRECVID 2011 train data that the T2 of STP

does not improve the results. This happens because the events do not have a temporal structure that can be easily captured by the rigid STP, as opposed to the actions that are temporally well cropped.

Evaluation of local features

Table 3.8 shows results on the TRECVID MED 2011 dataset. We contrast the different descriptors and their combinations for all ten event categories. We observe that the MBH descriptors are best performing among the individual channels. The fact that HOG outperforms HOF demonstrates that there is rich contextual appearance information in the scene as TRECVID MED contains complex event videos.

Between the two channel combinations, the best one is HOG+MBH, followed by HOG+HOF and HOF+MBH. This order is given by the complementarity of the features: both HOF and MBH encode motion information, while HOG captures texture information. Combining all three channels performs similarly to the best two-channel variant.

If we remove all spatio-temporal information (H3 and SFV), performance drops from 45.9 to 43.8. This underlines the importance of weak geometric information, even for the highly unstructured videos found in TRECVID MED.

We consider the effect of re-scaling the videos to different resolutions in Table 3.9 for both DT and IT. From the results we see that ITF always improves over DTF: even on low resolutions there are enough feature matches in order to estimate the homography reliably. The performance of both DTF and ITF does not improve much when using higher resolutions than 320.

The results in Table 3.9 also show that the gain from ITF on TRECVID MED is less pronounced than the gain observed for action recognition. This is possibly due to the generally poorer quality of the videos in this dataset, *e.g.* due to motion blur in videos recorded by hand-held cameras. In addition, a major challenge in this data set is that for many videos the information characteristic for the category is limited to a relatively short sub-sequence of the video. As a result the video representations are effected by background clutter from irrelevant portions of the video. This difficulty might limit the beneficial effects of the improved features.

Table 3.10 provides the speed (number of frames processed in one second on a single core) of computing our video representations when using the settings from Table 3.9. Computing ITF instead of DTF features increases the runtime by a factor between 1.5 and 2. For our final setting (videos resized to 480 px width, improved dense trajectories, HOG, HOF,

HOG	HOF	MBH	Birthday party	Changing a vehicle tire	Flash mob gathering	Getting vehicle unstuck	Grooming an animal	Making a sandwich	Parade	Parkour	Repairing an appliance	Sewing project	Mean
•			28.7	45.9	57.2	38.6	18.5	21.1	41.4	51.5	41.1	25.8	37.0
	•		18.8	28.5	54.6	37.2	24.5	17.2	44.9	66.7	35.6	28.5	35.7
		•	26.2	39.1	59.8	37.7	30.4	19.7	46.4	72.6	33.6	32.8	39.8
•	•		27.6	49.9	59.8	45.1	30.6	22.4	48.4	69.4	40.8	35.0	42.9
•		•	30.8	**53.9**	61.5	40.0	38.2	**28.8**	**53.4**	72.0	38.1	**43.3**	**46.0**
	•	•	26.8	40.7	59.8	41.2	31.2	20.3	47.6	71.8	33.5	34.7	40.8
•	•	•	**31.3**	53.0	**61.9**	**47.4**	**38.2**	23.4	51.4	**73.2**	**41.6**	37.5	45.9

Table 3.8 – Performance in terms of AP on the full TRECVID MED 2011 dataset. We use ITF and encode them with FV ($K = 256$). We also use SFV and STP, but only with a horizontal stride (H3), and no temporal split (T2). We rescale the video to a maximal width of 480 pixels.

MBH, stabilized without the human detector and encoded with FV and H3 SPM and SFV), the number of frames processed in a second is 2.31. Given that the videos have a frame rate of around 24 frames per second, the slowdown factor with respect to the real video time is around 10× on a single core. This translates in less than a day of computation for the 1000 hours of TRECVID test data on a 500-core cluster.

Comparison to the state of the art

We compare to the state-of-the-art in Table 3.11. We consider the EVENTS/DEV-O split of the TRECVID MED 2011 dataset, since most results are reported using this setup.

The top three results were reported by the following authors. Li et al. [2013a] attained 12.3% by automatically segmenting videos into coherent sub-sequences over which the features are pooled. Vahdat et al. [2013] achieved 15.7% by using multiple kernel learning to combine different

	160 px	320 px	480 px	640 px
DT	40.6	44.9	43.0	44.3
IT	41.0	45.6	**45.9**	45.4

Table 3.9 – Performance in terms of AP on the full TRECVID MED 2011 dataset using either dense trajectory (DT) or improved trajectory (IT) features. We study the impact of reducing the video resolutions for each of the two types of features. For both IT and DT, we combine HOG, HOF and MBH, and use FV ($K = 256$) augmented with SFV and STP, but only use H3 and not T2 for STP.

	160 px	320 px	480 px	640 px
DT	30.67	7.12	3.24	2.23
IT	18.72	4.97	2.31	1.38

Table 3.10 – The speed of computing our proposed video representation using different resolutions on the TRECVID MED dataset. We measure the speed as the number of frames that are processed in a second on a single core. The experiments include both computing raw features (*i.e.*, DT or IT) and encoding them into a high dimensional Fisher vector ($K = 256$).

Method	Features	mAP
Tang et al., 2012	HOG3D	4.8
Vahdat and Mori, 2013	HOG3D, textual information	8.4
Kim et al., 2013	HOG3D, MFCC	9.7
Li et al., 2013a	STIP	12.3
Vahdat et al., 2013	HOG3D, SSIM, SIFT, color	15.7
Tang et al., 2013	HOG3D, ISA, GIST, HOG, SIFT, LBP texture, color and geometry	21.8
Improved trajectories (IT)	HOG+HOF+MBH and SFV+H3	**31.6**

Table 3.11 – Performance in terms of AP on the TRECVID MED 2011 dataset using the EVENTS/DEV-O split. The feature settings are the same as Table 3.8: improved trajectory features (HOG+HOF+MBH), encoded with FV ($K = 256$) and SFV+H3.

features, and latent variables to infer the relevant portions of the videos. Tang et al. [2013] obtained the best reported result so far of 21.8%, using a method based on AND-OR graphs to combine a large set of features in different subsets.

We observe a dramatic improvement when comparing our result of 31.6% to the state of the art. In contrast to these other approaches, our work focuses on good local features and their encoding, and then learns a linear SVM classifier over concatenated Fisher vectors computed from the HOG, HOF and MBH descriptors.

3.7 Conclusion

In this chapter we set up a robust and efficient pipeline for video representation. We have demonstrated its effectiveness and flexibility through an extensive evaluation on three challenging tasks: action recognition, action localization in movies, and complex event recognition. As part of our benchmark we have varied the local features (dense trajectories and improved trajectories) and their encoding (bag of words and Fisher vectors). We have shown that Fisher vector are a viable alternative to the bag of words histograms, and that encoding weak geometric layout (spatio-temporal pyramids or spatial Fisher vectors) is a straightforward way to improve performance. We also found that action localization results can be substantially improved by using a simple re-scoring technique before applying NMS, to suppress a bias for too short windows. Our proposed pipeline significantly outperform the state of the art on all three tasks. Our approach can serve as a general pipeline for various video recognition problems and we will reuse it in the subsequent chapters.

Chapter 4

Efficient localization with approximately normalized Fisher vectors

Contents

The success of the Fisher vector (FV) representation can be ascribed to a large extent to its high dimensionality, which makes it very effective in combination with efficient linear classifiers. The flip side of its high dimensionality—representations of tens to hundreds of thousands of dimensions are more the rule than an exception—is, however, that it leads to large storage and computational requirements.

Localization of actions in video, or objects in images, can be considered as a large-scale classification problem, where we want to find the highest scoring windows in a video or image w.r.t. a classification model of the category of interest. Unlike generic large-scale image classification, however, the problem is highly structured in this case, in the sense that all windows are crops of the same video or image under consideration. This structure has been extensively exploited in the past. In particular, when the features for a detection window are obtained as sums of local features, integral images can be used to pre-compute cumulative feature sums. Once the integral images are computed, these can be used to com-

pute the sums of local features in constant time w.r.t. the window size. Viola and Jones [2004] used this idea to efficiently compute Haar filters for face detection. Recently, Chen et al. [2013] used the same idea to aggregate scores of local features in an object detection system based on a non-normalized FV representation. Another way to exploit the structure of the localization problem is to use branch-and-bound search, as *e.g.* used by Lampert et al. [2009a]. Instead of evaluating the score of one window at a time, they hierarchically decompose the set of detection windows and consider upper-bounds on the score of sets of windows to explore the most promising ones first.

While the power and ℓ_2 normalizations of Perronnin et al. [2010] have proven effective to improve the performance of the FV, the resulting normalized FV is no longer additive over local features. As a consequence, these FV normalizations prevent the use of integral image techniques to efficiently aggregate local features or scores.

Our first contribution, which we present in Section 4.2, is to show that the FV normalizations can be approximated in a way that the score for an arbitrarily large window can be computed in constant time, by relying on pre-computed cumulative sums of local visual word assignments, scores, and ℓ_2 norms. Second, in Section 4.3, we show that with our approximations the normalized FV becomes amenable to efficient localization using branch-and-bound search.

In our experimental evaluation, presented in Section 4.4, we validate on two action classification benchmarks that our approximations have only a limited impact on the effectiveness of the normalizations. Experiments on two temporal action localization datasets demonstrate that our approximations accelerate temporal localization by more than one order of magnitude when using a temporal sliding window approach. Branch-and-bound search brings further speedup when only the top scoring windows need to be reported. Before presenting our contributions and experiments in detail, we first discuss the most relevant related work in Section 4.1.

4.1 Related work

The search space for multi-dimensional localization problems grow exponentially: if we consider a D-dimensional grid with B bins per dimension, there are $C = B^D$ grid cells, and $O(C^2)$ windows defined over the grid, see Figure 4.1. Exhaustive sliding window search is therefore costly,

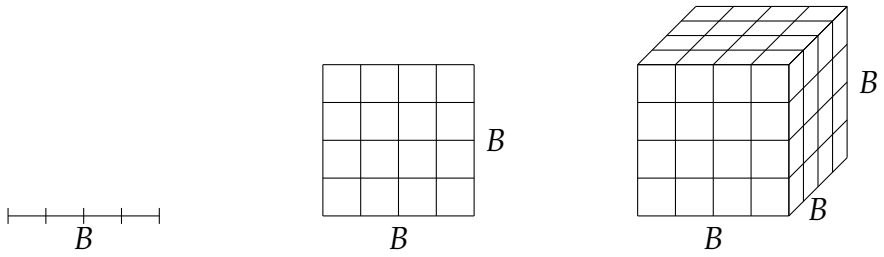


Figure 4.1 – The search space for multi-dimensional localization. Different number of dimensions correspond to various tasks: $D = 1$ for temporal localization (left), $D = 2$ for spatial localization (middle), $D = 3$ for spatio-temporal localization (right). The number of cells grows exponentially with the number of dimensions, $C = B^D$, and the number of total possible windows scales quadratically with the number of cells, $O(C^2)$, as a window is defined by two cells: a starting cell and an ending one.

unless low-dimensional features in combination with linear classifiers are used.

Viola and Jones [2004] introduced a face detector that combined efficient computation of Haar-filters over pixel intensities using integral images, with a detection-cascade that progressively rejects detection windows using an increasingly larger set of features. In this manner most computation is spent on finely analyzing the most promising image regions. Similar ideas were used by others for generic object category detection using richer image representations based on the bag of words (BOW) representation, by using progressively more expensive classifiers, see *e.g.* [Harzallah et al., 2009, Vedaldi et al., 2009].

If an additive window representation is used—such as a non-normalized BOW histogram—in combination with a linear classifier, several efficient algorithms are available for localization. These algorithms exploit the commutative property of the linear score function and the additivity of the representation. For the one-dimensional case the problem then reduces to the maximum subarray problem, which can be solved with a linear-time dynamic programming algorithm [Bellman, 1957]. In the two-dimensional case, An et al. [2009] presented an $O(C^{3/2})$ algorithm, which was used by Chen et al. [2013] for detection with non-normalized FVs. Another approach, used by Lampert et al. [2009a], is branch-and-bound search, for which the bounds are efficiently evaluated for additive features and linear classifiers. Yuan et al. [2009] generalized this approach to spatio-temporal localization in videos.

Most of the recent work that uses FV representations for object and action localization, and semantic segmentation, either uses non-normalized FVs [Chen et al., 2013, Csurka and Perronnin, 2011], or explicitly computes normalized FVs for all considered windows as in [Cinbis et al., 2013] or as in Chapter 3. The recent work of Li et al. [2013b] is an exception to this trend; they left out the power-normalization of the FV, but presented an efficient approach to incorporate exact ℓ_2 normalization. In this chapter we present approximations to both the power and ℓ_2 normalization, which allows us to compute the score of a window by aggregating locally precomputed and cached quantities. In particular, we store for each cell and visual word the local sum of assignments, scores, and ℓ_2 norms. This representation has a size that is only three times larger than a local BOW histogram, while leveraging the representational power of the normalized FV.

A different line of work focuses on using category-independent proposal methods, mainly driven by low-level contour and segmentation cues, to produce a small set of candidate detection windows, see *e.g.* [Alexe et al., 2012, Uijlings et al., 2013, Manen et al., 2013]. In this manner a set of only 1,000 to 2,000 windows suffices to capture 95% of the objects in the PASCAL VOC datasets. Since these techniques are decoupled from the actual detector, they do not impose any constraints on the detector or its features. In Chapter 5 we generalize such techniques to the video domain. The spatio-temporal object proposals can be used in combination with the approximate normalizations to further improve the efficiency of localization.

4.2 Approximate Fisher vector normalization

Below, we first briefly restate the Fisher vector (FV) image representation (for a more detailed exposition, see Section 3.2), after which we present our approximations to the power and ℓ_2 normalization. Finally, we analyze the complexity to compute the approximately normalized FV.

4.2.1 The Fisher vector and its normalizations

We apply the Fisher kernel principle [Perronnin and Dance, 2007] to obtain representations of sets of N local features. The local features are modeled as independent samples from a K-component Gaussian mixture model (GMM). We use Gaussians with diagonal co-variance matrices; let the vector σ_k to denote these diagonals. Let $x_n \in \mathbb{R}^d$ denote the n-

th d-dimensional local feature, q_{nk} the soft-assignment of x_n to the k-th Gaussian, and π_k and μ_k the mixing weight and mean of the k-th Gaussian respectively. The d-dimensional gradients w.r.t. the mean and variance of the k-th Gaussian are given by:

$$G_{\mu_k} = \sum_{n=1}^{N} q_{nk} \left[x_n - \mu_k\right] / \sqrt{\sigma_k \pi_k}, \tag{4.1}$$

$$G_{\sigma_k} = \sum_{n=1}^{N} q_{nk} \left[(x_n - \mu_k)^2 - \sigma_k\right] / \sqrt{2\sigma_k^2 \pi_k}, \tag{4.2}$$

where operations using σ_k should be understood as element-wise operations, and the normalization by the Fisher information matrix has already been taken into account. The concatenation of these d dimensional gradients, as $G = [G_1, \ldots, G_K]$ with $G_k = \left[G_{\mu_k}, G_{\sigma_k}\right]$, is then referred to as the Fisher vector (FV), which is of dimension $2Kd$. The gradient w.r.t. the mixing weights of the GMM are generally ignored since they contribute little discriminative power to the FV. See [Sánchez et al., 2013] for a recent comprehensive review of the FV image representation.

Two normalizations of the FV representation significantly improve its performance [Perronnin et al., 2010]. The first of these is the power normalization which consists in applying per-dimension a "signed" power, by transforming each element of the FV as $z \leftarrow \text{sgn}(z)\text{abs}(z)^\rho$, with $0 < \rho < 1$. The second normalization is the ℓ_2 normalization, which consists in rescaling the FV to have unit ℓ_2 norm.

While these normalizations are common in practice [Perronnin et al., 2010, Jégou et al., 2012, Arandjelović and Zisserman, 2013], the explanations on why they work still vary. One recurrent motivation is the phenomenon of *burstiness*—if a word appears once in a document, it is more likely to occur again. Burstiness was first observed in natural language [Church and Gale, 1995, Katz, 1996], but Jégou et al. [2009] have shown that visual words also exhibit a similar behaviour. One problem due to burstiness is that a few large components of the feature vector can dominate and adversely impact the classification score. Another problem was mentioned in [Cinbis et al., 2012]: if we assume a linear classifier for a given category c, its scoring function f keeps increasing with the number of words Δ_c specific to the category c: $f(h + \Delta_c) = f(h) + f(\Delta_c)$. This effect is undesirable as we would like to obtain the same classification score regardless of the size or quantity of the objects of interest. These issues are alleviated by discounting the values of the feature vector by element-wise power normalization and then re-normalizing the values by ℓ_2 normalization.

Other interpretations of the power normalization are offered in [Winn et al., 2005, Perronnin et al., 2010, Jégou et al., 2012, Cinbis et al., 2012]. Perronnin et al. [2010] observe that FVs become sparser as the number of Gaussians increases. They argue that ℓ_2 distance is not an appropriate measure for sparse data, so they propose to reduce the sparsity using the power normalization. Winn et al. [2005] and Jégou et al. [2012] motivate the power normalization as a variance stabilization transformation, assuming that the data is well modeled by a Poisson distribution. Cinbis et al. [2012] show that transformations similar to power normalization occur when using more complex models that do not assume the independence of visual words.

The ℓ_2 normalization ensures that the video size does not impact the distance between the corresponding feature vectors. Consider two videos with very similar content, but one being just much longer than the other: for example, let the second video be just the first video repeated several times. In this case the Euclidean distance between the two video representations might be very large, but the relative distributions of the features are identical. The ℓ_2 normalization cancels the effect of the magnitude, as it can be shown that the Euclidean distance between ℓ_2 normalized vectors corresponds to using a distance based on the angle α between the original vectors:

$$d(x,y) = \|x\|_2^2 + \|y\|_2^2 - 2x^\top y \tag{4.3}$$
$$= 2(1 - x^\top y) = 2(1 - \cos\alpha). \tag{4.4}$$

Another reason for the ℓ_2 normalization is proposed in [Perronnin et al., 2010], where they show that this normalization reduces the impact of the background component into the final representation.

4.2.2 Approximate power normalization

Cinbis et al. [2012] have argued that the power normalization corrects for the independence assumption that is made in the GMM model that underpins the FV representation. They presented latent variable models which do not make this independence assumption, and experimentally found that such models lead to similar performance improvements as the power-normalization. In particular, they showed that the gradients w.r.t. the mixing weights in their non-i.i.d. model take the form of discounted version of these gradients in the original i.i.d. model. The transformation they found was the di-gamma function, which, like the power-normalization, is a concave monotonic function.

Based on this analysis, we propose an approximate version of the power normalization. First, note that the gradients in G_k are *weighted sums* of contributions of local features. Let us write these in a more compact manner as:

$$G_k = \sum_n q_{nk} g_{nk} = \left(\sum_n q_{nk} \right) \sum_n \frac{q_{nk} g_{nk}}{\sum_m q_{mk}}, \tag{4.5}$$

where q_{nk} and g_{nk} denote the weight and gradient contribution of the n-th local descriptor for G_k. The last part of Eq. (4.5) re-interprets the FV as a *weighted average* of local contributions, multiplied by the sum of the weights. The power-normalization is computed as an element-wise signed-power of G_k. In our approximation we, instead, apply the power only to the (positive) sum of weights:

$$\mathcal{G}_k = \left(\sum_n q_{nk} \right)^{\rho} \sum_n \frac{q_{nk} g_{nk}}{\sum_m q_{mk}}. \tag{4.6}$$

In our approximation, the power-normalization modifies the magnitude of the gradient vector, but not its orientation (see Figure 4.2). We concatenate the $\mathcal{G}_k$ to form the normalized FV $\mathcal{G}$.

Using our approximate power-normalization, a linear function can now be computed by aggregating local scores. For a classifier weight vector $w = [w_1, \ldots, w_k]$ we have:

$$\langle w, \mathcal{G} \rangle = \sum_k \langle w_k, \mathcal{G}_k \rangle = \sum_k \left(\sum_n q_{nk} \right)^{\rho-1} \sum_n s_{nk}, \tag{4.7}$$

where $s_{nk} = \langle w_k, q_{nk} g_{nk} \rangle$ denotes the score of the local non-normalized FV, which is still additive over local terms.

4.2.3 Approximate ℓ_2 normalization

We now proceed with an approximation of the ℓ_2 norm of $\mathcal{G}$. The squared ℓ_2 norm is a sum of squared ℓ_2 norms per Gaussian component: $\|\mathcal{G}\|_2^2 = \sum_k \mathcal{G}_k^\top \mathcal{G}_k$, where

$$\mathcal{G}_k^\top \mathcal{G}_k = \left(\sum_n q_{nk} \right)^{2(\rho-1)} \sum_{n,m} q_{nk} q_{mk} \langle g_{nk}, g_{mk} \rangle . \tag{4.8}$$

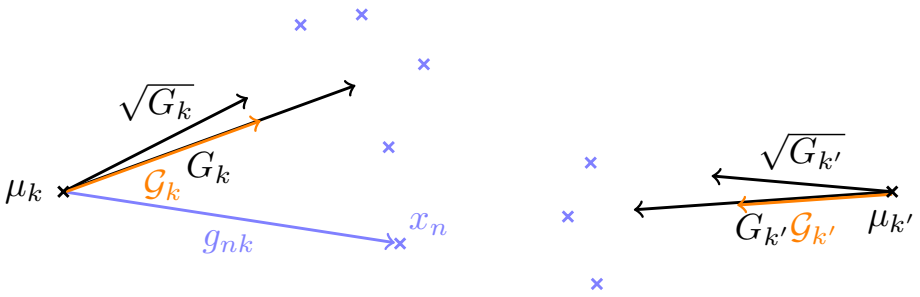


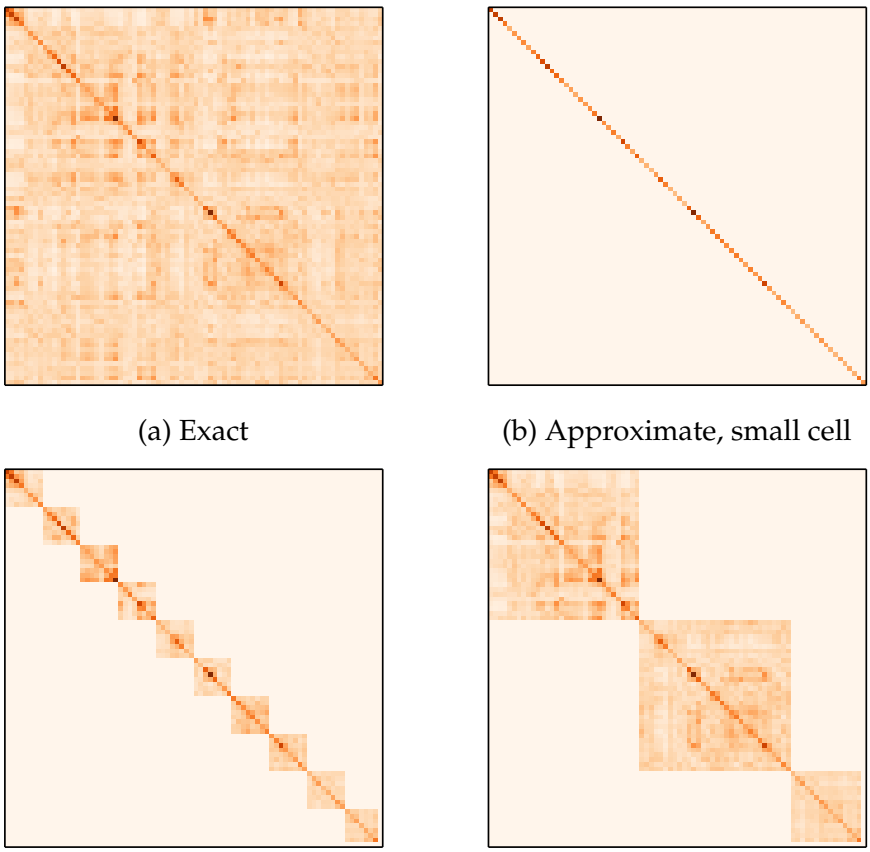
Figure 4.2 – Illustration of the Fisher vector G_k and the power normalizations: the Fisher vector after the exact ($\sqrt{G_k}$) and approximated ($\mathcal{G}_k$) power normalization (we use the $\sqrt{\cdot}$ symbol to denote a power normalization with $\rho = 0.5$). The figure depicts the descriptor space with each descriptor shown as a blue cross. The contribution of the n-th descriptor to the Fisher vector is indicated by the vector g_{nk}. Our approximated normalization $\mathcal{G}_k$ scales the magnitude of the Fisher vector G_k, as opposed to the exact power normalization $\sqrt{G_k}$ which changes also the orientation.

We approximate the double sum over dot products of local gradient contributions by assuming that most of the local gradients will be near orthogonal for high-dimensional FVs. This leads to an approximation $L(\mathcal{G}_k)$ of the squared ℓ_2 norm of $\mathcal{G}_k$ in the form of a sum of local contributions:

$$L(\mathcal{G}_k) = \left(\sum_n q_{nk}\right)^{2(\rho-1)} \sum_n q_{nk}^2 l_{nk}, \tag{4.9}$$

where $l_{nk} = \langle g_{nk}, g_{nk} \rangle$ is the local squared ℓ_2 norm. Summing these over the visual words, we approximate $\|\mathcal{G}\|_2^2$ with $L(\mathcal{G}) = \sum_k L(\mathcal{G}_k)$. The approximated ℓ_2 norm corresponds to summing only the diagonal terms of the cross-product matrix, as shown in Figure 4.3.

Theoretically, we can show the following: (i) if the FV of local descriptors are positively correlated (which we expect them to be in practice), our approximated L2 norm is an underestimate of the true norm; (ii) in expectation the approximation of ℓ_2 norm is the true ℓ_2 norm if the contributing descriptors are independently distributed and with zero mean. To prove these claims we start from the fact that the exact ℓ_2 norm, $L(G_k)$, equals to the sum of diagonal terms of the cross-product matrix, which represents our approximate ℓ_2 norm, $L(\mathcal{G}_k)$, plus the sum of the off-diagonal terms of the cross-product matrix (for brevity, we do not consider the power



(a) Exact (b) Approximate, small cell

(c) Approximate, medium cell (d) Approximate, large cell

Figure 4.3 – Cross-product matrices between the local gradients g_{nk} that contribute to the Fisher vector $\mathcal{G}_k$. The exact ℓ_2 norm of the Fisher vector can be obtained by summing the quantities in the cross-product matrix (top left). For the approximated case (top right), we consider only the diagonal elements of the matrix (the off-diagonal elements are assumed to be zero). The bottom row highlights the values retained for the approximated case when we increase the cell size. We analyze how the quality of the approximated ℓ_2 norm is impacted by the cell size in Section 4.4.

normalization in the subsequent formulas):

$$L(G_k) = L(\mathcal{G}_k) + \sum_{\substack{m \\ m \neq n}} \sum_{n} \left(q_{mk} g_{mk}\right)^\top \left(q_{nk} g_{nk}\right). \tag{4.10}$$

The first claim holds if the FV of local descriptors are positively correlated: $\left(q_{mk} g_{mk}\right)^\top \left(q_{nk} g_{nk}\right) \geq 0$ implies that $L(G_k) \geq L(\mathcal{G}_k)$. We empirically show that indeed this is the case for FVs in Section 4.4.

To show the second claim we take the expectation of (4.10) w.r.t. descriptor random variable x. Applying the linearity property of the expectation, we obtain:

$$\mathbb{E}_x L(G_k) = \mathbb{E}_x L(\mathcal{G}_k) + \sum_{\substack{m \\ m \neq n}} \sum_{n} \mathbb{E}_x \left(q_{mk} g_{mk}\right)^\top \left(q_{nk} g_{nk}\right). \tag{4.11}$$

Now we use the following two assumptions to cancel out the second term from the right hand side, and conclude the proof: (a) the data is i.i.d., so the expected value is multiplicative, that is $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$ for X, Y i.i.d. random variables; (b) the FVs have zero mean, that is $\mathbb{E}[G_k] = 0$.

4.2.4 Complexity of approximately normalized FVs

We combine the above approximations to compute a linear function of our approximately normalized FV as

$$f(\mathcal{G}; w) = \left\langle w, \mathcal{G} / \sqrt{L(\mathcal{G})} \right\rangle = \langle w, \mathcal{G} \rangle \Big/ \sqrt{L(\mathcal{G})}. \tag{4.12}$$

The $2Kd$ dimensional dot product $\langle w, \mathcal{G} \rangle$ can be rewritten as a K dimensional product between two vectors. The first is $s = \left[\sum_n s_{n1}, \ldots, \sum_n s_{nK}\right]$, which collects the accumulated scores for each visual word, with s_{nk} defined as above. The second is $q = \left[\sum_n q_{n1}, \ldots, \sum_n q_{nK}\right]$, which similarly collects the weights by summing over the q_{nk} instead. Considering Eq. (4.7), we then obtain:

$$\langle w, \mathcal{G} \rangle = \left\langle q^{(\rho-1)}, s \right\rangle, \tag{4.13}$$

where $q^{(\rho-1)}$ contains the element-wise powers of q. Using Eq. (4.9), we can similarly express $L(\mathcal{G})$ as a K-dimensional dot product if we let $l = \left[\sum_n q_{n1}^2 l_{n1}, \ldots, \sum_n q_{nK}^2 l_{nK}\right]$ denote the vector of accumulated ℓ_2 norms:

$$L(\mathcal{G}) = \left\langle q^{2(\rho-1)}, l \right\rangle. \tag{4.14}$$

Combining these formulations we obtain the classification score as a ratio of two dot products:

$$f(\mathcal{G}; w) = \left\langle q^{(\rho-1)}, s \right\rangle \Big/ \sqrt{\left\langle q^{2(\rho-1)}, l \right\rangle}. \tag{4.15}$$

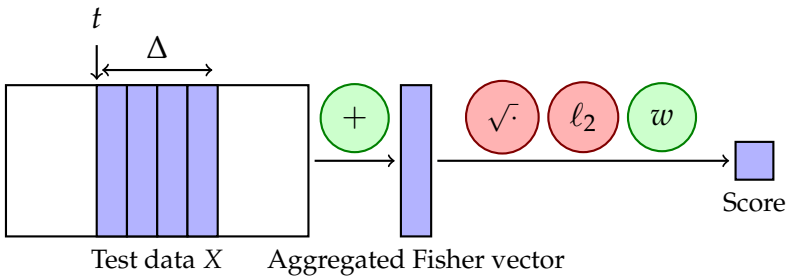
The accumulated quantities—scores s, assignments q, norms l—are all computed by aggregating local quantities over continuous regions of the search space. The cost of obtaining these quantities scales with the size of the region: a bigger window involves summing more local quantities. With the help of integral images we can compute in constant time these per-window quantities and the corresponding score function $f(\mathcal{G}; w)$. Since the scores, assignments, and norms vary per visual word, we need to compute three integral images for each visual word. As we will see shortly, we are going to make use of integral images whenever we deal with aggregation.

For the complexity analysis we assume we use a multidimensional grid, with C cells in total. However, in Figure 4.4, to facilitate visualization we illustrate the process of window scoring for a one dimensional grid ($D = 1$), corresponding to the task of temporal localization. We consider three normalization cases:

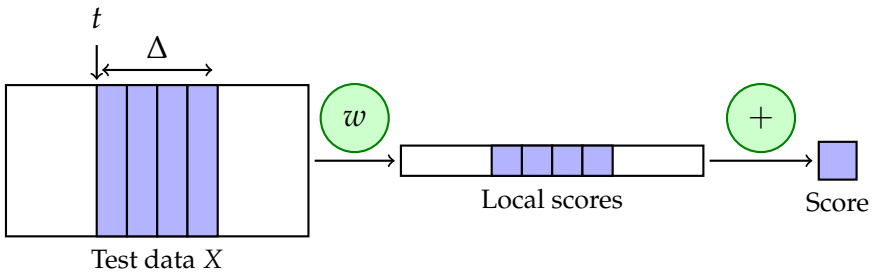
Exact normalization We apply the power and ℓ_2 normalizations before scoring a FV, as we have already described in Subsection 4.2.1. In order to score a given window we first aggregate the FVs corresponding to that window into a single vector, then normalize and score this vector. We speed up the aggregation operation by computing integral images over FVs.

No normalization We skip the normalizations and directly score a FV. As for the exact case, we could first aggregate the FVs corresponding to a given window into a single vector and then score it. But, because we assume linear classifiers, we have a more efficient way of scoring a given window, by interchanging the two steps: first score each temporal slice and then aggregate the local scores. Again we speed up the aggregation (in this case of local scores) by working on integral images.

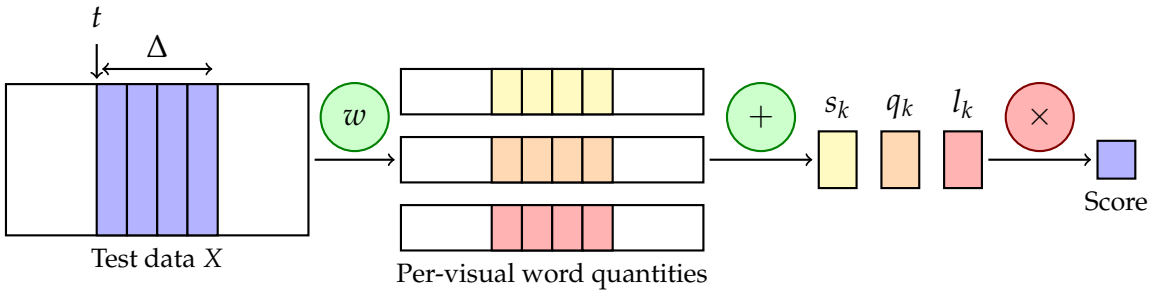
Approximate normalization We use the proposed approximations as we have detailed above, see Eq. (4.15). The score of a given window in this case depends only on the per visual word quantities (scores, assignments, ℓ_2 norms), so we just store those instead of the full FVs. We make use again of integral images to speed up the aggregation of the per visual word quantities.



(a) Exact normalization. For each location t and duration Δ, we aggregate the corresponding FVs into a single vector. We then apply the non-linear normalizations, power and ℓ_2 normalization, and score with the linear classifier w.



(b) No normalization. First we pre-compute the scores for each individual temporal slice by computing the dot product $X^\top w$ of the data matrix X with the classifier weights w. Then we can efficiently score any window (t, Δ) by just summing the local scores of the corresponding slices.



(c) Approximate normalization. First we pre-compute per-visual word quantities: scores s_{nk}, assignments q_{nk}, norms l_{nk}. Then, for a given window (t, Δ), we aggregate the quantities and combine them into the final score, see Eq. (4.15).

Figure 4.4 – Prediction step for temporal localization for three different normalization cases: exact normalization, no normalization, approximate normalization. In each case we are given the data matrix $X \in \mathbb{R}^{2Kd \times C}$, which contains a Fisher vector for each temporal slice, and the goal is to score a given temporal interval, starting at slice t and lasting for Δ slices. See the text for a discussion on the complexity of each of these three cases.

Normalization	Storage	Pre-computing	Window evaluation
Exact	$2CKd$	$O(CKd)$	$O(Kd)$
None	C	$O(CKd)$	$O(1)$
Approximate	$3CK$	$O(CKd)$	$O(K)$

Table 4.1 – Summarizing the complexity analysis for the three normalization cases. We remind that K denotes the number Gaussian clusters, d the dimensionality of the local features, C the number of cells.

The first step in all the three cases is to aggregate the local FVs per cell, and to compute the (multi-dimensional) integral image over these cells. For exact normalization (Figure 4.4a), we need to compute $2Kd$ integral images, since we first need to compute the full window-level FVs before the normalizations can be applied. If do not use any normalization (Figure 4.4b), we pre-score each temporal slice by computing the dot product between the data matrix and the classifier weights; then we compute the integral image over the array of scores. When using our approximations (Figure 4.4c), we will compute $3K$ integral images that accumulate the local weights, scores, and norms per visual word. The cost of this step for all the three cases is $O(CKd)$. Compared to storing the full FV, our representation is $2d/3$ times more compact, while compared to the "no normalization" case, our representation is $3K$ larger. When using, as in our experiments, $d = 192$ for the local features, the storage requirements for the approximate version are reduced by a factor $2 \times 192/3 = 128$, compared to the exact version.

Once the integral images are available, the cost to score a window of arbitrary size is $O(K)$ with our approximations, as opposed to $O(Kd)$ when using exact normalization. We thus obtain an $O(d)$ speedup for the window scoring. Using no normalization is fastest, as we are able to score any arbitrary window in constant time, $O(1)$. We summarize these results in Table 4.1.

The actual cost to obtain the integral images is slightly higher when using our approximate normalizations, since at this stage we already compute local scores and norms, and take powers of the weight sums. When using exact normalization, we only sum local FVs at this stage. The cost of the approximiate normalizations is, however, amortized as the per-window scoring is a factor d faster; and exact normalization requires taking powers of the full window-level FVs. In our experiments we assess the speedup as observed in different practical settings, as well as the

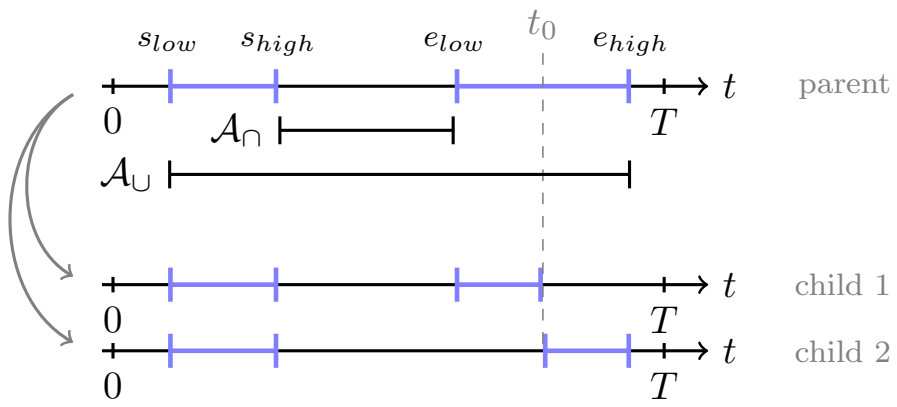


Figure 4.5 – The *branching* step of the branch-and-bound search algorithm. The initial set of possible windows contains all the windows that start between s_{low} and s_{high} and end between e_{low} and e_{high} (parent). This set is split into two subsets by separating the windows by their ending point: windows that finish before t_0 go into the first subset (child 1) and windows that finish after t_0 go into the second subset (child 2). The *bounding* step, which we illustrate for the parent, depends on quantities computed on the intersection ($\mathcal{A}_\cap$) and union ($\mathcal{A}_\cup$) of all windows.

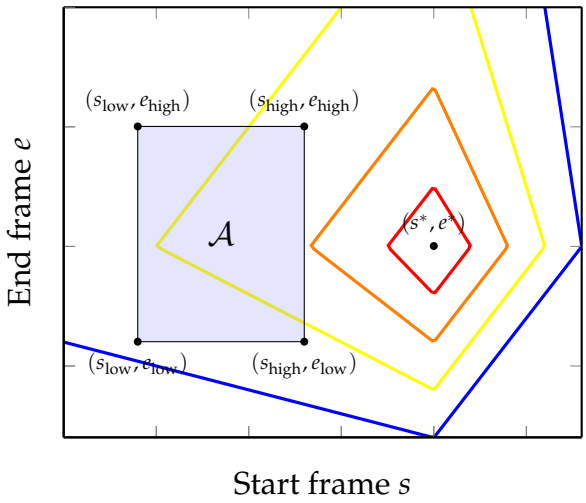
impact of the normalizations on the recognition performance.

4.3 Integration with branch-and-bound search

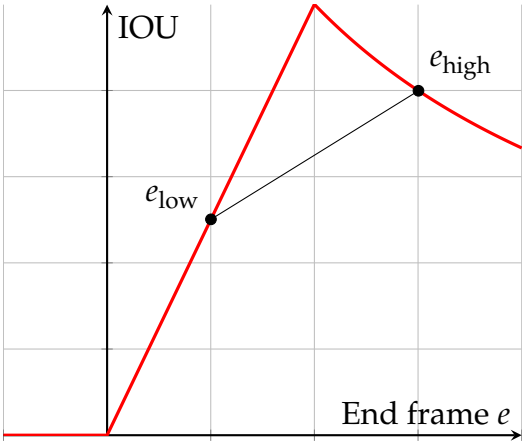
The approximations we presented above accelerate the scoring of windows by aggregating locally pre-computed scores, weights, and norms. A second method to speedup detection is to use a branch-and-bound search instead of exhaustive search. The idea of branch-and-bound is to evaluate upper bounds on the scores of windows in sets of detection windows. Starting from the set of all possible windows, the search is organized by hierarchically splitting sets of windows and computing upper bounds on the scores. A set of detection windows $\mathcal{A}$ is defined as a collection of tuples (s, e), denoting the starting and ending point of a window:

$$\mathcal{A} = \left\{(s, e) | s \in [s_{\text{low}}, s_{\text{high}}], e \in [e_{\text{low}}, e_{\text{high}}]\right\},$$

where $s_{\text{low}} \in \mathbb{R}^D$ defines the earliest starting point for the windows in $\mathcal{A}$ on the D-dimensional grid, and, similarly, s_{high} defines the latest starting point for all windows in $\mathcal{A}$, and e_{low} and e_{high} define the earliest and



(a) Contour plot of the IOU function with a given window (s^*, e^*). The rectangle denoted by $\mathcal{A}$ represents the set of windows in the branch-and-bound iteration.



(b) IOU as a function of the end frame e only (we fix the starting frame). For any two points e_{low} and e_{high}, their convex combination is greater than at least one of them. By symmetry, the same holds true for the IOU as a function of the start frame s. This means that the minimum is attained at the corners and not on the edges of the rectangle.

Figure 4.6 – Analysis of the intersection-over-union (IOU) with a reference window (s^*, e^*). The IOU is a function of the start and end frames. The function is unimodal and its maximum is attained when it matches exactly the reference window. If we constrain the search over a rectangle $\mathcal{A}$, the minimum value is obtained at one of the corners of the rectangle.

latest end points. Sets of windows are split by either splitting the start range or the end range on a single dimension, depending where the range is maximum (see Figure 4.5). The sets are explored in a best-first manner, which focuses on the most promising parts of the search space first. See [Lampert et al., 2009a] for a comprehensive introduction to branch-and-bound search.

The branch-and-bound search finds the best scoring window, but sometimes we want to report the top k scoring windows. For example, for temporal action localization it is often the case that multiple actions appear in the same movie, and we would like to be able to detect all the instances. We obtain the top k windows by running the branch-and-bound algorithm k times, after each iteration removing the selected window from the search space. To avoid redoing work at each run of the algorithm, we share information—the set of windows and their corresponding bounds—between the runs and expand it accordingly after each run.

As in Chapter 3, we apply non-maxima suppression (NMS) on the scored windows, but instead of performing it after the branch-and-bound search, we integrate it within the search. Such a strategy can improve the efficiency of our algorithm, because it allows to prune parts of the search state space. Sometimes all the windows in the current set $\mathcal{A}$ overlap more than the NMS threshold with the one of the already selected windows; in this case we can stop exploring the set $\mathcal{A}$, as any window selected from $\mathcal{A}$ will be subsequently discarded by NMS. To determine if a given set $\mathcal{A}$ still contains windows of interest, we observe that the intersection-over-union (IOU) function is lower-bounded by the values obtained on one of the following four windows: $(s_{\text{low}}, e_{\text{low}})$, $(s_{\text{low}}, e_{\text{high}})$, $(s_{\text{high}}, e_{\text{low}})$, $(s_{\text{high}}, e_{\text{high}})$; if the IOU of any pre-selected window with all these four windows is greater than the NMS threshold, then we stop exploring the set $\mathcal{A}$. See Figure 4.6 for a graphical explanation of this property; Figure 4.6a illustrates the IOU function as a contour plot and the set of windows $\mathcal{A}$ as a rectangle; the minimum value inside the rectangle is attained at one of its four corners.

In the forthcoming subsections, we derive bounds for the score defined in Eq. (4.12). For clarity of exposition, we start with a bound on a simple additive score function, and then present bounds when adding our approximate power and ℓ_2 normalizations.

4.3.1 Upper-bound for additive linear classifiers

If the function is linear and additive in the local features it is easy to obtain an upper bound by separating the positive and negative terms. In

particular, consider such a function over the non-normalized FV G:

$$f(G;w) = \langle w, G\rangle = \sum_k \langle w_k, G_k\rangle = \sum_k \sum_n s_{nk},$$
$$= \sum_k \left[\sum_{n:s_{nk}>0} s_{nk} + \sum_{n:s_{nk}<0} s_{nk} \right], \tag{4.16}$$

where as before $s_{nk} = \langle w_k, q_{nk} g_{nk}\rangle$. We can upper bound the score of the FV of any window in a set of windows $\mathcal{A}$ by accumulating positive and negative scores of the union $\mathcal{A}_\cup$ and intersection $\mathcal{A}_\cap$ of all windows in $\mathcal{A}$ respectively:

$$f(\mathcal{A};w) = \sum_k \left[\sum_{n\in\mathcal{A}_\cup:s_{nk}>0} s_{nk} + \sum_{n\in\mathcal{A}_\cap:s_{nk}<0} s_{nk} \right]. \tag{4.17}$$

4.3.2 Bounding approximate power-normalization

When using our approximate power-normalization, a linear classification score takes the form of Eq. (4.7):

$$f(\mathcal{G};w) = \langle w, G\rangle = \sum_k \left(\sum_n q_{nk} \right)^{\rho-1} \sum_n s_{nk}. \tag{4.18}$$

To bound this function for a set of windows $\mathcal{A}$, we can use the previous bound for the linear score terms $\sum_n s_{nk}$. Provided the intersection $\mathcal{A}_\cap$ is non-empty, the scalar multiplication with $(\sum_n q_{nk})^{\rho-1}$ can be bounded by accumulating the weights over the intersection $\mathcal{A}_\cap$ instead. For $0 < \rho < 1$ this leads to the upper bound

$$f_1(\mathcal{A}) = \sum_k \left(\sum_{n\in\mathcal{A}_\cap} q_{nk} \right)^{\rho-1} \left[\sum_{\substack{n\in\mathcal{A}_\cup \\ s_{nk}>0}} s_{nk} + \sum_{\substack{n\in\mathcal{A}_\cap \\ s_{nk}<0}} s_{nk} \right].$$

If the intersection $\mathcal{A}_\cap$ is empty, however, we obtain a trivial upper bound $f(\mathcal{A}) = \infty$, since $0^{(\rho-1)} = \infty$ for $0 < \rho < 1$. To bound this case, we use the interpretation of the normalized FV as a weighted average, *c.f.* Eq. (4.6), and write:

$$f(\mathcal{G};w) = \sum_k \left(\sum_n q_{nk} \right)^{\rho} \sum_n \frac{q_{nk}\langle w_k, g_{nk}\rangle}{\sum_m q_{mk}}. \tag{4.19}$$

It is then easy to see that we can upper bound the score as

$$f_2(\mathcal{A}) = \sum_k \left(\sum_{n \in \mathcal{A}_\cup} q_{nk} \right)^\rho \max_{n \in \mathcal{A}_\cup} \langle w_k, g_{nk} \rangle. \tag{4.20}$$

Since the latter bound relies on a non-linear max operation, we cannot efficiently compute it using intergral images. Therefore, the bound $f_1(\mathcal{A})$ is preferred if $\mathcal{A} \neq \emptyset$.

4.3.3 Bounding with approximate ℓ_2 norm included

To upper bound a linear function of our approximately power and ℓ_2 normalized FVs, *c.f.* Eq. (4.12), for a set of windows $\mathcal{A}$, we need to lower bound the approximate ℓ_2 norm:

$$L(\mathcal{G}) = \sum_k \left(\sum_n q_{nk} \right)^{2(\rho-1)} \sum_n q_{nk}^2 l_{nk}. \tag{4.21}$$

Since $0 < \rho < 1$, then $2(\rho - 1) < 0$, so the first term can be bounded summing over the union $\mathcal{A}_\cup$ instead. If the intersection $\mathcal{A}_\cap$ is non-empty we can bound the second term by summing over the intersection, and obtain the lower-bound on $L(\mathcal{G})$ as:

$$L_1(\mathcal{A}) = \sum_k \left(\sum_{n \in \mathcal{A}_\cup} q_{nk} \right)^{2(\rho-1)} \sum_{n \in \mathcal{A}_\cap} q_{nk}^2 l_{nk}. \tag{4.22}$$

If the intersection is empty, we can instead of the sum over $\mathcal{A}_\cap$, use the minimum over $\mathcal{A}_\cup$ to obtain the bound:

$$L_1(\mathcal{A}) = \sum_k \left(\sum_{n \in \mathcal{A}_\cup} q_{nk} \right)^{2(\rho-1)} \min_{n \in \mathcal{A}_\cup} q_{nk}^2 l_{nk}. \tag{4.23}$$

It is easy to verify that if $\mathcal{A}_\cap \neq \emptyset$ then $L_2(\mathcal{A}) \leq L_1(\mathcal{A})$, thus in this case $L_1(\mathcal{A})$ is the tightest of the two bounds.

4.4 Experiments

We use four datasets in our evaluation: Hollywood2, HMDB, Coffee & Cigarettes and Duchenne; more information about the data and their

evaluation protocol can be found in the previous chapter, see Section 3.5. We reuse the video representation pipeline that we proposed in Chapter 3. In the current subsection, we start by detailing the parameters used for features and action localization. Then we show results for our approximations in the context of action classification and, finally, evaluate the speedup our approximations brings to temporal action localization.

Features. We use the dense trajectory features of Wang et al. [2013a], with their standard parameters. We extract only MBH features and project them to 64 dimensions with PCA. As in [Oneata et al., 2013], we use 1,000 GMM components for classification, and include position information with spatial pyramids and spatial Fisher vectors. For temporal localization we use a GMM with 128 components, and no position information. This setting yields a FV of 804,000 dimensions for classifications and 16,384 for localization.

Action localization parameters. For localization we consider temporal windows with lengths from 20 to 180 frames, with increments of 5 frames. We use a stride of five frames to locate the windows on the video. We use zero-overlap non-maximum suppression, and re-scale the window scores by the duration, as in Chapter 3. When using branch-and-bound, window sets that are guaranteed to intersect already selected windows are removed from the queue.

4.4.1 Effect of approximation on action classification

In our first experiment we consider the effect of the power and ℓ_2 normalizations of the FV for action classification, and assess to which degree our approximations maintain the performance benefits of the exact normalizations. In our experiments we use the common setting of $\rho = 0.5$, see *e.g.* [Chatfield et al., 2011, Jégou et al., 2012, Oneata et al., 2013], which corresponds to a signed square-root.

The first four results in Table 4.2 assess the effectiveness of the exact power and ℓ_2 normalization for action classification. For both datasets the power normalization is the most effective one, improving performance by 6.8 and 8.6 mAP points respectively. Adding ℓ_2 normalization improves results further by 0.4 and 0.5 mAP points respectively.

When using approximate power normalization (fifth line), but exact ℓ_2 normalization, performance drops only slightly for both datasets. For Hollywood2 and HMDB the loss is only 0.3 and 0.1 points respectively; which is respectively 2.0 and 5.3 points above not using power normalization. Experimentally, we found that it is beneficial to apply an additional element-wise standardization of the FV after approximate

Power norm.	ℓ_2 normalization	Hollywood2	HMDB
No	No	55.2	43.1
Exact	No	62.0	51.7
No	Exact	60.1	46.8
Exact	Exact	62.4	52.2
Approximate	Exact	62.1	52.1
Approximate	Approximate, $n = 5$	60.1	52.6
Approximate	Approximate, $n = 10$	60.2	52.4
Approximate	Approximate, $n = 20$	60.2	52.6
Approximate	Approximate, $n = 40$	60.6	52.5
Approximate	Approximate, $n = 80$	60.7	52.2
Approximate	Approximate, $n = 160$	61.1	52.2
Wang et al., 2013a		59.9	48.3
Oneata et al., 2013		61.9	51.9
Jain et al., 2013		62.5	52.1
Wang and Schmid, 2013		64.3	57.2
ITH (Chapter 3)		66.8	60.1

Table 4.2 – Action classification performance. For the ℓ_2 approximation we evaluate using cells of n frames, for $n = 5$ to $n = 160$.

power normalization, and before ℓ_2 normalization. If this is not done, performance drops by about 1 point to 61.3% and 51.1% mAP respectively. This standardization can absorbed in the classifier weight vector w, before computing the local scores s_{nk}, and therefore does not impact the computational efficiency of our approach.

The following six results show the effect of approximate ℓ_2 normalization for various temporal cell sizes over which the features are aggregated: from 5 up to 160 frames. The smaller the cell size, the coarser our approximation, since more cross-terms will be ignored in our approximation. The results show, however, that the classification performance is only slightly impacted by using smaller cells. For Hollywood2 the best result of 61.1% is obtained for cells of 160 frames, while using 5-frame cells yields 60.1% mAP. For HMDB the variation in performance across different cell sizes is

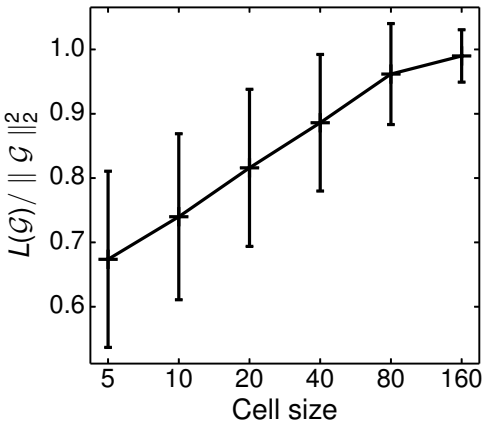


Figure 4.7 – Errors in the ℓ_2 norm approximation, see text for details.

at most 0.4 points, with better performance for smaller cells.

We further analyze the ℓ_2 norm approximation by considering the ratio between the approximate and the true norm. In Figure 4.7 we show the average of the ratio and its standard deviation for various cell sizes measured on HMDB. Note that the ratio is generally smaller than one, as expected. Moreover, even for small cell sizes the under estimation is limited to about a factor two, with limited variation in the under estimation factor. This explains the small impact on the recognition performance observed above.

To show that our video representation is competitive with the state-of-the-art, we include the results from Chapter 3 and four other recent state-of-the-art results of [Jain et al., 2013, Oneata et al., 2013, Wang et al., 2013a, Wang and Schmid, 2013]. Wang et al. [2013a] use a sum of RBF chi-squared kernels over BOW histograms for HOG, HOF, and MBH features, computed over six different SPM grids [Lazebnik et al., 2006], and using 4,000 visual words. The setup of Oneata et al. [2013] is comparable to the one we used here. Jain et al. [2013] use camera motion stabilization before computing MBH, HOG, and HOF features, in addition to their DCS flow features. They aggregate these features using VLAD descriptors [Jégou et al., 2012], a variant of FV. Our results from Chapter 3, as well as those of Wang and Schmid [2013], use the improved dense trajectories with human detector (ITH) encoded with FV, but we included also weak spatio-temporal information (spatial pyramids, spatial Fisher vector).

Search	Power	ℓ_2	Drinking	Smoking	Open Door	Sit Down
Exh.	No	No	34.0	15.6	10.3	5.9
Exh.	No	Yes	61.1	55.5	24.1	18.3
Exh.	Yes	No	64.8	23.8	20.6	17.1
Exh.	Yes	Yes	64.8	55.4	28.4	19.0
Exh.	Appr.	Appr.	67.1	52.0	18.1	13.6
Oneata et al., 2013			63.9	50.5	26.5	18.2
Gaidon et al., 2011			57	31	16.4	19.8
Laptev and Pérez, 2007			49	—	—	—
Duchenne et al., 2009			40	—	14.4	13.9
IT (Chapter 3)			80.2	40.9	26.0	27.1

Table 4.3 – Action localization performance using either no, exact, or approximate normalizations, and recent state-of-the-art results.

4.4.2 Temporal action localization

In our temporal action localization experiments, we compare using exact normalizations and our approximations in terms of localization performance and speed.

The localization results are presented in Table 4.3. As before, the first four results consider the impact of the exact normalizations on performance. For *drinking* and *sit down* the power and ℓ_2 normalization have a similar impact, and improve the results by about 30 and 12 mAP points respectively. Using both normalizations does not bring further improvements for *drinking*, but does improve results by 0.7 points for *sit down*. For *smoking* and *open door* the ℓ_2 normalization brings the largest improvement of almost 40 and 14 mAP points respectively. Additional power normalization is not effective for *smoking*, but does bring an improvement of 4.3 points for *open door*.

Next, we consider the impact of our approximate power and ℓ_2 normalizations. For *drinking* we obtain an AP of 67.1%, which is even 2.3

Search	Power	ℓ_2	Drinking	Smoking	Open Door	Sit Down
Exh.	No	No	2.8	2.2	31.1	29.9
Exh.	No	Yes	95.4	92.2	1276.7	1168.3
Exh.	Yes	No	146.3	143.4	1794.3	1781.9
Exh.	Yes	Yes	160.3	151.0	1966.8	2036.4
Exh.	Appr.	Appr.	11.3	10.3	140.6	138.0

Table 4.4 – Timings (secs.) for action localization using exhaustive search with either no, exact, or approximate normalizations.

points above the results for exact normalization. For *smoking* our approximations also lead to an AP of 52.0%, which is 3.4 points below the result for exact normalization. For *open door* and *sit down* the results are 10.3 and 5.4 points below those obtained using exact normalization. They are, however, still 7.8 and 7.7 points better than not using normalization. For *drinking* and *smoking* the gain of our approximate normalization w.r.t. no normalization is 33.1 and 36.4 mAP points.

We compare to the results from Chapter 3, where we have used the improved trajectories, and we also include other recent state-of-the-art results of [Duchenne et al., 2009, Gaidon et al., 2011, Laptev and Pérez, 2007, Oneata et al., 2013] to show that our results are competitive. The results for [Laptev and Pérez, 2007] are taken from [Duchenne et al., 2009], which have interpolated their spatio-temporal localization results to the temporal domain. Our results with exact normalization are above the best earlier reported results. Using our approximations this is still the case for *drinking* and *smoking*, but not for *open door* and *sit down*.

The timing results in Table 4.4 show that our approximations lead to a speedup of about one order of magnitude w.r.t. using exact normalization. As compared to using no normalization, our approximations are about four to five times slower, but leads to significantly better results.

Finally, we evaluate the speed of branch-and-bound search with our approximations to find the top scoring temporal windows. In Table 4.5 we give an overview of the search times when searching for the t highest

	Drinking	Smoking	Open Door	Sit Down
Pre-processing	7.65	6.99	93.01	92.98
Exhaustive search	3.65	3.31	47.67	43.63
Branch and bound, top 1	1.02	1.40	16.59	20.57
Branch and bound, top 10	1.71	2.44	35.41	30.32

Table 4.5 – Search times for the action detection datasets in seconds for exhaustive and branch-and-bound search, where both use our approximate normalization.

scoring windows, for $t = 1$ and $t = 10$. Both exhaustive and branch-and-bound search first pre-process all the cells in the temporal grid to compute the local sums of scores, assignments, and norms. We separate the time needed for pre-processing, and the actual search time. When searching for the top window, branch-and-bound is between 2.1 and 3.6 times faster. For the top 10 windows, the speedup factors are between 1.3 and 2.4. Although faster than exhaustive search, overall the speedup obtained using branch-and-bound is limited. This is because in our uni-dimensional temporal search setup, the number of windows is only 32 times larger than the number of cells in the search grid. We expect larger speedup for branch-and-bound when applied to 2D spatial, or 3D spatio-temporal localization problems.

4.5 Conclusion

We have presented approximate versions of the power and ℓ_2 normalization of the Fisher vector representation. These approximations allow efficient evaluation of linear score functions, by caching local per visual word sums of scores, assignments, and norms. We also presented corresponding upper-bounds that permit the use of our approximations in branch-and-bound search. Experimental results for action classification and localization show that these approximations only have a limited impact on performance, while yielding speedups of at least one order

of magnitude. When only the top-scoring window is required, branch-and-bound search can further speedup localization by a factor between 2 to 4, excluding pre-processing. The efficient localization techniques presented here are directly applicable to other localization tasks, such as object localization in still images, and spatio-temporal action localization. Since these tasks consider higher dimensional search spaces, we expect the speedup of our approximations, as well as branch-and-bound search, to be even larger than for temporal localization task that we considered in this chapter. We plan to explore application of our approximations for these tasks in future work.

Chapter 5

Spatio-temporal proposals

Contents

In the previous chapters we have investigated the problems of action recognition and temporal action localization. In this chapter we turn our attention towards the more challenging task of spatio-temporal localization. The goal is to temporally locate the action of interest and for each selected frame estimate a bounding box. The concatenation of bounding boxes forms a spatio-temporal tube that locates the action both in space and time.

Whereas the sliding window search is still viable for temporal action localization, as we have seen in Chapter 3, this exhaustive search becomes computationally prohibitive when searching in the much larger space of spatio-temporal tubes. Efficient search methods for spatio-temporal action localization have been proposed in the past, exploiting additivity structures of bag-of-word representations and linear classifiers, *e.g.* using branch-and-bound search or dynamic programming [Tran and Yuan, 2011, Yuan et al., 2009]. These methods, however, do not apply when the representation is non-additive, as is the case for the Fisher vector representation used in recent state-of-the-art action classification methods, due to its non-linear power and ℓ_2 normalizations. We have addressed this issue using approximate normalizations in Chapter 4, while others

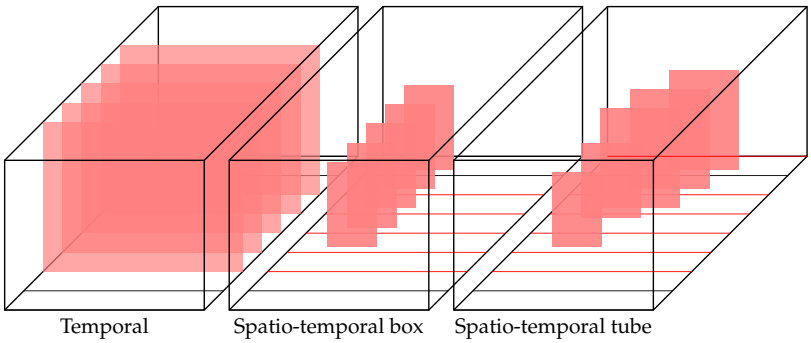


Figure 5.1 – Illustration of the three localization scenarios.

turned towards more efficient implementations [Li et al., 2013b, Van de Sande et al., 2014].

A more general technique to tackle this issue, which has recently surfaced in the 2D object recognition literature, is the use of generic class-independent detection proposals [Alexe et al., 2012, Endres and Hoiem, 2010, Manen et al., 2013, Uijlings et al., 2013, Krähenbühl and Koltun, 2014, Zitnick and Dollár, 2014]. These methods produce image-dependent but unsupervised and class-independent tentative object bounding boxes, which are then assessed by the detector. This enables the use of more computationally expensive features, that would be too expensive to use in sliding window approaches. Recent state-of-the-art object detectors based on this approach use various representations, *e.g.* Fisher vectors [Cinbis et al., 2013], max-pooling regionlets [Wang and Schmid, 2013], and convolutional networks [Girshick et al., 2014].

In this chapter we explore how we can generate video tube proposals for spatio-temporal action detection. We build on the recent approach of Manen et al. [2013] that uses a randomized superpixel merging procedure to obtain object proposals. We extend their approach to the spatio-temporal domain, using supervoxels as the units that will be merged into video tube proposals. We introduce spatial, temporal and spatio-temporal pairwise supervoxel features that are used to learn a classifier that guides the random merging process. Our proposals are based on a novel hierarchical spatio-temporal segmentation method (HST).We experimentally evaluate our detection proposals, in combination with the new supervoxel method as well as existing ones. This evaluation shows that the HST supervoxels lead to more accurate proposals when compared to using existing state-of-the-art supervoxel methods.

Below, we first review related work in more detail in Section 5.1. Then, in Section 5.2 we present our supervoxel method, and in Section 5.3

our tube proposal method. In Section 5.4 we present our experimental evaluation results, and we conclude in Section 5.5.

5.1 Related work

In this section we discuss the most relevant related work on supervoxels, and efficient detection methods based on object proposals and other techniques.

5.1.1 Supervoxel methods

Instead of a complete survey of supervoxel methods, we concentrate here on the approaches most related to our work. The recent evaluation by Xu and Corso [2012] compares five different methods [Corso et al., 2008, Felzenszwalb and Huttenlocher, 2004, Fowlkes et al., 2004, Grundmann et al., 2010, Paris and Durand, 2007] to segment videos into supervoxels. They identify GBH [Grundmann et al., 2010] and SWA [Corso et al., 2008] as the most effective supervoxel methods according to several generic and application independent criteria. SWA is a hierarchical segmentation method that solves normalized cuts at each level. At the finest levels, it defines similarities from voxel intensity differences, while at higher levels it uses aggregate features which are computed over regions merged at earlier levels. GBH is a hierarchical extension of the graph-based method of Felzenszwalb and Huttenlocher [2004]. A streaming version of GBH was introduced in [Xu et al., 2012], which performs similar to GBH, but at a fraction of the cost by using overlapping temporal windows of the video to optimize the segmentation. GBH, similar to SWA, also uses aggregate features (such as color histograms) to define similarities once an initial segmentation is performed based on intensity differences.

While SWA and GBH directly work on the 3D space-time voxel graph, the recent VideoSEEDS approach of Van den Bergh et al. [2013] shows that supervoxels of similar quality can be obtained by propagating 2D superpixels computed over individual frames in a streaming manner. In our own work we take a similar approach, in which we take per-frame SLIC superpixels [Achanta et al., 2012] as the starting point, and merge them spatially and temporally to form supervoxels. The advantage of starting from per-frame superpixels is that the graphs that are used to form larger supervoxels are much smaller than those based on individual voxels. The superpixels themselves are also efficient to obtain since they are extracted independently across frames.

Since objects can appear at different scales, a single segmentation of a video into supervoxels does typically not succeed in accurately capturing all objects and either leads to under or over segmentation. Xu et al. [2013] recently proposed a supervoxel hierarchy flattening approach that selects a slice through such a hierarchy that can maximize a variety of unsupervised or supervised criteria. In this manner the segmentation scale can be locally adapted to the content. Our work is related, in the sense that we also aim to use (hierarchical) supervoxel segmentation to find regions that correspond to objects. Unlike Xu et al. [2013], however, we do not restrict ourselves to finding a single segmentation of the video, and instead allow for overlap between different detection hypotheses.

5.1.2 Object proposals for detection in video

Several spatio-temporal action detection approaches have been developed based on ideas originally developed for efficient object detection in still images. Yuan et al. [2009] proposed an efficient branch-and-bound search method to locate actions in space-time cuboids based on efficient subwindow search [Lampert et al., 2009a]. Search over space-time cuboids is, however, not desirable since the accuracy of the spatial localization will be compromised as the object of interest undergoes large motion. Tran and Yuan [2011] proposed an efficient method for spatio-temporal action detection, which is based on dynamic programming to search over the space of tubes that connect still-image bounding boxes that are scored prior to the spatio-temporal search. Building the trellis, however, is computationally expensive since it requires per frame a sliding-window based scoring of all considered bounding boxes across different scales and aspect ratios if the tube size is allowed to vary over time. Moreover, the efficiency of such approaches relies on the additive structure of the score for a video cuboid or tube. This prevents the use of state-of-the-art feature pooling techniques that involve non-additive elements, including max-pooling [Wang and Schmid, 2013], power and ℓ_2 normalization for Fisher vector representations [Perronnin et al., 2010] or second-order pooling [Carreira et al., 2012].

Recently, several authors have proposed efficient implementations [Li et al., 2013b, Van de Sande et al., 2014], while we have introduced approximate normalizations in Chapter 4 to efficiently use the Fisher vector representation. Our approximate normalizations, as well as the fast local area independent representation (FLAIR) of Van de Sande et al. [2014] assume rectangular regions in order to exploit integral images, but this assumption does not hold for spatio-temporal tubes. A technique

that is more general and applies to arbitrary representations is the use of generic class-independent proposals [Alexe et al., 2012, Endres and Hoiem, 2010, Manen et al., 2013, Uijlings et al., 2013], which have recently surfaced in the context of 2D object localization. These methods rely on low-level segmentation cues to generate in the order of several hundreds to thousands of object proposals per image, which cover most of the objects. Once the detection problem is reduced to assessing a relatively modest number of object hypotheses, we can use stronger representations that would otherwise have been prohibitively costly if employed in a sliding window detector, see the recent state-of-the-art results in *e.g.* [Cinbis et al., 2013, Girshick et al., 2014, Wang et al., 2013c]. Here we just discuss two of the most effective object proposal methods. Uijlings et al. [2013] generate proposals by performing a hierarchical clustering of superpixels. Each node in the segmentation hierarchy produces a proposal given by the bounding box of the merged superpixels. The approach of Manen et al. [2013] similarly agglomerates superpixels, but does so in a randomized manner. In Section 5.3 we show how this technique can be adapted for space-time detection proposals based on supervoxel segmentation.

Van den Bergh et al. [2013] proposed a video objectness method based on tracking windows that align well with supervoxel boundaries. The tracking is based on the evolution of the supervoxels inside the tracked window. Unlike [Manen et al., 2013, Uijlings et al., 2013] and our work, however, their method is inherently dependent on the scale of the supervoxels that produce the boundaries which define the objectness measure.

In parallel to our work, Jain et al. [2014] developed an extension of the hierarchical clustering method of Uijlings et al. [2013] to the video domain to obtain object proposals. The most notable difference with our work is that they compute their initial supervoxels from an "independent motion evidence" map. This map estimates for each pixel in each frame the likelihood that its motion is different from the dominant motion. While this approach is effective to segment out objects that are in motion w.r.t. the background, it does not provide a mechanism to recover objects that are static in the scene. Furthermore, estimating the dominant motion is often error prone in real world videos.

Finally, several recent methods address the related but different problem of motion segmentation in video [Ma and Latecki, 2012, Papazoglou and Ferrari, 2013, Zhang et al., 2013, Fragkiadaki et al., 2015]. Their goal is to produce a pixel-wise segmentation of the dominant moving object in videos. Unlike the methods discussed above, they produce a single estimate of the dominant object, which is assumed to be at least partially in motion. Both [Ma and Latecki, 2012] and [Zhang et al., 2013] are based on

linking still-image object proposals from [Endres and Hoiem, 2010]. They refine the window-based solution using a pixel-wise MRF. Papazoglou and Ferrari [2013] proposed a method using motion boundaries to estimate the outline of the object of interest, and refine these estimates using an object appearance model. Instead of relying on object proposal bounding boxes per frame, they rely on per-frame superpixel segmentation as the base units over which the energy function is defined. Fragkiadaki et al. [2015] start with object proposals for each frame, which they rank using a moving objectness score and then extend temporarily the most promising ones, based on dense optical flow. In our experiments we compare to the results of Papazoglou and Ferrari [2013] on the YouTube Objects dataset.

5.2 Hierarchical supervoxels by spatio-temporal merging

Our supervoxel approach starts from superpixels as basic building blocks, and aggregates spatially and temporally connected superpixels using hierarchical clustering. In Section 5.2.1 we detail the superpixel graph construction and the definition of the edge costs. Then, in Section 5.2.2 we present the hierarchical clustering approach which includes a novel penalty term that prevents merging physically disconnected objects.

5.2.1 Construction of the superpixel graph

We use SLIC [Achanta et al., 2012] to independently segment each video frame into N superpixels. SLIC superpixels have been shown to accurately follow occlusion boundaries [Lu et al., 2013], and are efficient to extract. For each superpixel n, we compute its mean color $\mu(n)$ in Lab space, a color histogram $h_{\text{col}}(n)$ using ten bins per channel, and a flow histogram $h_{\text{flow}}(n)$ that uses nine orientation bins. We construct a graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$, where $\mathcal{S}$ is the set of superpixels, and $\mathcal{E}$ is the set of edges between them. In Section 5.4.1 we detail how we set the parameters of the graph weights using a small set of training images.

Spatial neighbor connections. Spatial connection edges are created per frame for each pair of neighboring superpixels n and m, and their weight $w^{\text{sp}}(n,m)$ is given by the weighted sum of distances based on several cues that we detail below:

$$\begin{aligned} w^{\text{sp}}(n,m) = {} & \alpha_{\mu} d_{\mu}(n,m) + \alpha_{\text{col}} d_{\text{col}}(n,m) + \alpha_{\text{flow}} d_{\text{flow}}(n,m) \\ & + \alpha_{\text{mb}} d_{\text{mb}}(n,m) + \alpha_{\text{edge}} d_{\text{edge}}(n,m), \end{aligned} \tag{5.1}$$

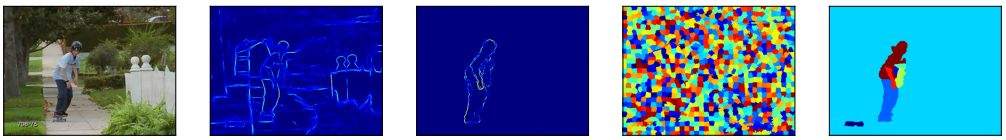


Figure 5.2 – Illustration of our supervoxel construction. From left to right: video frame, detected edges, flow boundaries, superpixels, and hierarchical clustering result at the level with eight supervoxels.

where $d_\mu(n,m) = \min(\|\mu(n) - \mu(m)\|, 30)$ is the robust thresholded-distance between the color means [Pele and Werman, 2009], $d_{\text{col}}(n,m)$ and $d_{\text{flow}}(n,m)$ are chi-squared distances between the color and flow histograms. In our implementation we use the LDOF optical flow method of Brox and Malik [2011].

The last two terms, $d_{\text{mb}}(n,m)$ and $d_{\text{edge}}(n,m)$, are geodesic distances between the superpixels centroids, efficiently computed using the distance transform [Weber et al., 2008]. We use the norm of the gradient of the flow and the output of the recent structured edge detector [Dollár and Zitnick, 2013] respectively, to define their geodesic pixel-wise cost. In practice, it means that if two superpixels are separated by an edge—either intensity-based or motion-based—the distance between them will increase proportionally to the edge strength. See Figure 5.2 for an illustration of these two distance terms.

Second-order spatial connections. We also add second-order spatial edges to connect neighbors of neighbors. The rationale behind this setting is to be robust to small occlusions. Imagine, *e.g.*, the case of a lamp post in front of a tree: we would like the two parts of the tree to be connected together before they are merged with the lamp post. In this case we drop the geodesic distances, since they are affected by occlusions, and add a constant penalty α_{2hop} instead:

$$w^{\text{2hop}}(n,m) = \alpha_\mu d_\mu(n,m) + \alpha_{\text{col}} d_{\text{col}}(n,m) + \alpha_{\text{flow}} d_{\text{flow}}(n,m) + \alpha_{\text{2hop}} \quad (5.2)$$

Whenever a second-order neighbor can be reached through an intermediate neighbor k with a smaller distance, *i.e.* when $w^{\text{sp}}(n,k) + w^{\text{sp}}(k,m) \leq w^{\text{2hop}}(n,m)$, we remove the second-order edge between n and m. This avoids spurious connections between physically disconnected regions, and also significantly reduces the number of edges in the graph.

Temporal neighbor connections. Temporal connections are naturally introduced using optical flow. We connect each superpixel with its neighbor in the next frame indicated by the flow. Because the flow is sometimes

noisy, we enforce a one-to-one correspondence in the temporal connectivity. More precisely, for each superpixel at frame t, we compute its best match match in frame $t+1$ according to flow and pixel-wise color difference. We run this procedure in the opposite temporal direction as well, and keep only reciprocal connections. For two temporally connected superpixels n and m, we set the edge weight to:

$$w^t(n,m) = \alpha^t_\mu d_\mu(n,m) + \alpha^t_{\text{col}} d_{\text{col}}(n,m) + \alpha^t_{\text{flow}} d_{\text{flow}}(n,m). \tag{5.3}$$

This is similar to the spatial edge weight, but excludes the motion and contour boundaries as we do not have a temporal counterpart for them.

5.2.2 Hierarchical clustering

Once the superpixel graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$ is constructed, we run a hierarchical clustering with average linkage [Arbeláez et al., 2011]. For two clusters $A \subset \mathcal{S}$ and $B \subset \mathcal{S}$, we denote the set of edges connecting them as $\mathcal{B}(A,B) = \{(n,m) \mid n \in A, m \in B, (n,m) \in \mathcal{E}\}$. By construction, $\mathcal{B}(A,B)$ only contains edges at the boundary between A and B, or slightly deeper for the second-order connections. We define the distance between A and B as

$$w(A,B) = \frac{1}{|\mathcal{B}(A,B)|} \sum_{(n,m)\in\mathcal{B}(A,B)} w(n,m). \tag{5.4}$$

This corresponds to measuring the distance between two clusters as the average edge weight along their common boundary. Because the graph is sparse, the clustering can be computed efficiently: if the number of connections per superpixel is independent of the number of superpixels—as is the case in practice—the complexity is linear in the number of superpixels.

While such a clustering approach gives good results for image segmentation [Arbeláez et al., 2011], its temporal extension tends to group clusters corresponding to different physical objects that are only accidentally connected in a small number of frames, see Figure 5.3. We propose a simple solution to solve this issue. We add a penalty α_{dis} acting as a virtual edge for each frame where the two clusters are present but not in contact:

$$\tilde{w}(A,B) = \frac{c(A,B)w(A,B) + s(A,B)\alpha_{\text{dis}}}{c(A,B) + s(A,B)}, \tag{5.5}$$

where $c(A,B)$ is the number of frames where A and B are connected by direct or second-order spatial connections, and $s(A,B)$ is the number of

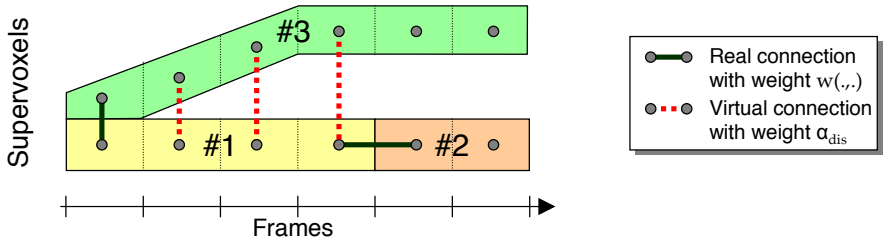


Figure 5.3 – Illustration of several supervoxels (SV) during the hierarchical clustering. While SV#1 and SV#2 can be merged without penalty, merging SV#1 and SV#3 will trigger a penalty α_{dis} in the form of a virtual edge added in all frames where both are present but not in contact.

frames where both are present but not connected. In practice, we set α_{dis} to the cost of the last merge of a hierarchical clustering performed preliminarily without temporal penalty, which corresponds to the weight of the hardest merge.

For an illustration of the supervoxel clustering process see the two right-most panels of Figure 5.2 which show the SLIC superpixels, and the hierarchical clustering result obtained from them. Figure 5.4 shows examples of the segmentation method at various levels of the hierarchy. The resulting supervoxels are heterogeneous in size: the humans are segmented into small supervoxels, while the background into larger ones. The difference in the granularity is caused by the features we use: motion and appearance vary more for the humans than for the background. Choosing a finer level of the segmentation hierarchy (the images on the right) further accentuates further the bimodal size distribution—humans are oversegmented more than the background.

The overall complexity of our method is linear in the number of frames. In practice, about 99% of the computational cost is devoted to computing LDOF optical flow [Brox and Malik, 2011]. Concretely, for a video of 55 frames with resolution 400×720 pixels, computing the flow with LDOF takes 13.8 minutes (about 15s/fr), computing the SLIC superpixels 9.7s, computing superpixels connection weights 7.8s, and performing hierarchical clustering 1.7s (all times are given for a single core @3.6GHz). Our method can benefit from the existing GPU implementation of LDOF [Sundaram et al., 2010], as well as a trivial parallelization over frames for the per-frame feature pipeline to compute SLIC, edges, and flow.

Figure 5.4 – Examples of segmentation at various levels of the hierarchy.

5.3 Spatio-temporal object detection proposals

In this section we describe how to produce spatio-temporal object detection proposals based on a given supervoxel segmentation.

5.3.1 Randomized supervoxel agglomeration

We extend the region growing method of Manen et al. [2013], which is a randomized form of Prim's maximum spanning tree algorithm. It starts from a random seed superpixel, and iteratively adds nodes that are connected to the ones that are already selected, as shown in Figure 5.5. Instead of adding the node with the maximum edge, as in Prim's algorithm, edges are sampled with probability proportional to the edge weight. The edge weight w_{nm} that connects two superpixels n and m is given by a logistic discriminant classifier that linearly combines several pairwise features over superpixels, and predicts whether two superpixels belong to the same object or not.

At each merging step t a random stopping criterion is evaluated. The stopping probability is given as $(1 - w_{nm} + s(a_t))/2$, which is the average of two terms. The first, $(1 - w_{mn})$, is the probability (as given by the classifier) that the sampled edge connects superpixels that belong to different objects. This term avoids growing the proposal across object boundaries. The second term, $s(a_t)$ gives the fraction of objects in the training dataset that is smaller than the size a_t of the current proposal. This term ensures that the size distribution of the proposals roughly reflects the size distribution of objects on the training set. The sampling process can be repeated to produce any desired number of detection proposals.

To apply this method for spatio-temporal proposals, we consider a graph over the supervoxels, with connections between all supervoxels that are connected by a regular 3D 6-connected graph over the individual voxels. To ensure that we sample proposals that last for the full video duration, we continue the merging process as long as the full duration

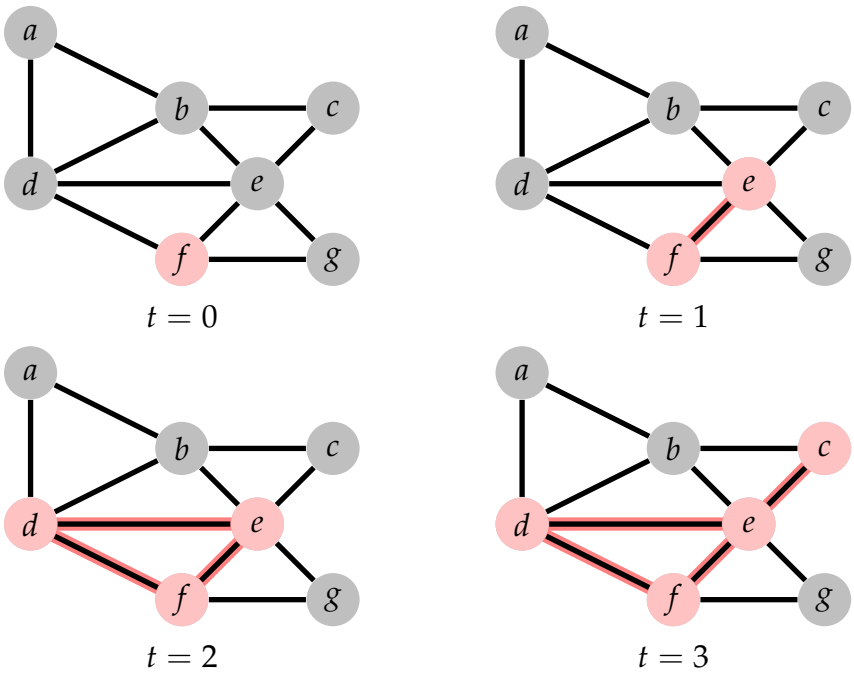


Figure 5.5 – Steps of the randomized supervoxel proposal algorithm. Each node corresponds to a supervoxel. The edges indicate a neighbouring connection between two supervoxels. At each step the algorithm selects one node that connects with the existing agglomeration.

of the video is not covered, and use the stopping criterion only after that point. Once a merged set of supervoxels is obtained, we produce a space-time tube by taking in each frame the bounding box of the selected supervoxels in that frame. Examples of resulting space-time tubes are shown in Figure 5.6.

5.3.2 Learning supervoxel similarities

We train a logistic discriminant model to map a collection of pairwise features to a confidence value that two supervoxels should be merged. To this end we generate a training set of supervoxel pairs from a collection of training videos. Each supervoxel is labeled as positive if it is contained for at least 60% inside a ground truth object, and negative otherwise. We then collect pairs of neighboring supervoxels that are either both positive, which leads to a positive pair, or for which one is positive and the other is negative, which leads to a negative pair. Neighboring supervoxel pairs that are both negative are not used for training.

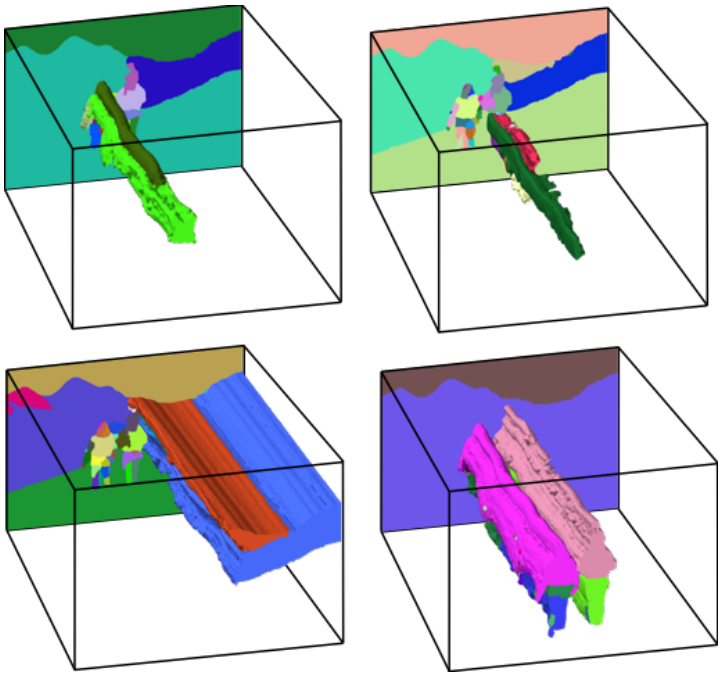


Figure 5.6 – Examples of spatio-temporal proposals.

We use eight different pairwise features between supervoxels, and list them below.

1. **Color feature.** We use the chi-squared distance between supervoxel color histograms $f_{\text{color}}(n,m) = d_{\chi^2}(h_{\text{col}}(n), h_{\text{col}}(m))$. We use the same color histogram as used for our supervoxels, *i.e.* using ten bins per channel in the Lab space.
2. **Flow feature.** We measure chi-squared distances between histograms of optical flow $f_{\text{flow}}(n,m) = d_{\chi^2}(h_{\text{flow}}(n), h_{\text{flow}}(m))$. Here we also use the same histograms as before, *i.e.* using LDOF [Brox and Malik, 2011] and nine orientation bins.
3. **Size feature.** The size feature favors merging small supervoxels first, and is defined as the sum of the volumes of the two supervoxels: $f_{\text{size}}(n,m) = a_n + a_m$. The volumes a_n and a_m are normalized by dividing over the volume of the full video.
4. **Fill feature.** The fill feature that favors merging supervoxels that form a compact region, and measures to which degree the two supervoxels fill their bounding box: $f_{\text{fill}}(n,m) = (a_n + a_m)/b_{nm}$, where b_{nm} is the volume of the 3D bounding box of the two supervoxels, again normalized by the video volume.

5. **Spatial size feature.** This feature only considers the spatial extent of the supervoxels, and not their duration: $f_{\text{size2D}}(n,m) = \frac{1}{|t_m \cup t_n|} \sum_{t \in (t_m \cup t_n)} a_n^t + a_m^t$, where t_n is a set of frames in which supervoxel n exists, while a_n^t denotes the area of the supervoxel in frame t, normalized by the frame area.
6. **Spatial fill feature.** Similarly, the fill feature can be made duration invariant by averaging over time: $f_{\text{fill2D}}(n,m) = \frac{1}{|t_m \cup t_n|} \sum_{t \in (t_m \cup t_n)} (a_n^t + a_m^t)/b_{nm}^t$, where b_{nm}^t gives the area of the bounding box in frame t.
7. **Temporal size feature.** In analogy to the spatial size feature, we also consider the joint duration $f_{\text{duration}}(n,m) = |t_m \cup t_n|/T$, where T is the duration of the video.
8. **Temporal overlap feature.** We measure to what extent the supervoxels last over the same period of time by intersection over union: $f_{\text{overlap}}(n,m) = |t_m \cap t_n| / |t_m \cup t_n|$.

The color (1), size (3) and fill (4) features are similar to those used by Manen et al. [2013] and Uijlings et al. [2013] for still-image object detection proposals. Besides these features, we also include features based on optical flow, as well as size and fill features that consider separately the spatial and temporal extent of the supervoxels (2, 5, 6, 7, 8). We do not include the motion boundary and edge features of Section 5.2, since these are not defined for neighboring supervoxels that have no temporal overlap.

In our experiments we use the same set of eight features regardless of the underlying supervoxel segmentation method, but we do train specific weights for each segmentation method to account for their different characteristics.

5.4 Experimental evaluation results

Before presenting our experimental results, we first describe the experimental setup, datasets, and evaluation protocols in Section 5.4.1. We then evaluate our supervoxel algorithm in Section 5.4.2, and our spatio-temporal detection proposals in Section 5.4.3.

5.4.1 Experimental setup

Supervoxel segmentation. We evaluate our supervoxel method on the Xiph.org benchmark of Chen and Corso [2010], using the following two of the evaluation measures proposed in [Xu and Corso, 2012]. The *3D segmentation accuracy* averages over all ground truth segments g the

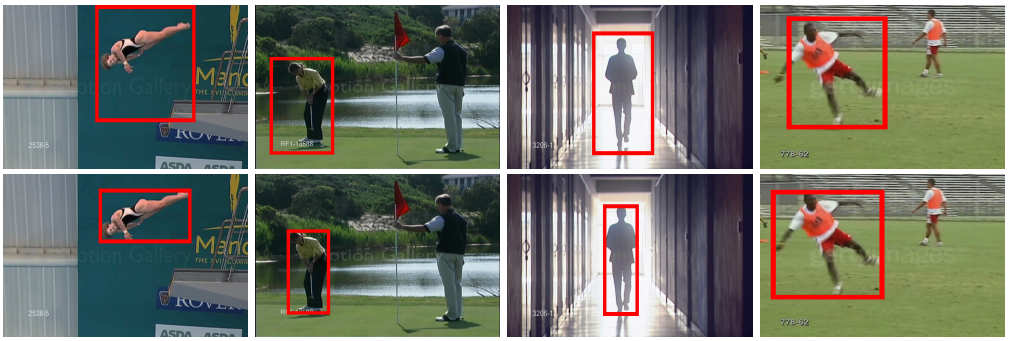


Figure 5.7 – Examples of annotations: on top showing the existing annotations for UCF Sports, on bottom, our re-annotations. The previous annotations were usually too loose (first three columns), but sometimes they were just imprecise (last column).

following accuracy: the volume of g covered by supervoxels that have more than 50% of their volume inside g, divided by the volume of g. The *3D undersegmentation error* averages the following error over all ground truth segments: the total volume of supervoxels that intersect g minus the volume of g, divided by the volume of g.

Spatio-temporal detection proposals. The first dataset we use is UCF Sports [Rodriguez et al., 2008], which consists of 150 videos of 10 sports: diving, golf, kicking, lifting, horse riding, running, skating, swinging, high bar, and walking. We use five videos of each class for training our supervoxel similarities, and to select other hyperparameters such as the granularity of the base segmentation level. The remaining 100 videos are used for testing. Since the original ground-truth data is not very precise, we re-annotated the objects more accurately every 20 frames. These annotations, as well as the train-test division, are all publicly available on our project webpage. [1]

The second dataset we consider is the YouTube Objects dataset [Prest et al., 2012a]. It contains a total of 1407 video shots divided over ten object categories. Each video contains one dominant object, for which a bounding box annotation is available in only one frame.

Similar to the 2D object proposal evaluation in [Uijlings et al., 2013], we measure performance using the best average overlap (BAO) of proposals with ground truth actions and objects. The BAO of a ground truth object v is given by the proposal p in the set of proposals P_v for v that maximizes the average overlap with the ground truth bounding boxes across all

1. See http://lear.inrialpes.fr/~oneata/3Dproposals.

annotated frames. More formally:

$$\text{BAO}(v) = \max_{p \in P_v} \frac{1}{|T_v|} \sum_{t \in T_v} \text{Overlap}(p_t, b_v^t), \tag{5.6}$$

where T_v is the set of frames for object v with ground-truth annotation, and b_v^t denotes the bounding box of v in frame t, and p^t is the bounding box of the proposal in that frame. We measure the per-frame overlap in the usual intersection-over-union sense. Compared to the segmentation metrics, which match many super-voxels to a single groundtruth segment, the proposal performance matches a single object to the groundtruth.

Based on the BAO we compute the mean BAO (mBAO) across all ground truth actions/objects. We also consider the correct localization (CorLoc) rate, as in [Papazoglou and Ferrari, 2013], which measures the fraction of objects for which the BAO is above 50%.

5.4.2 Experimental evaluation of supervoxel segmentation

Setting of supervoxel parameters. For our supervoxel method we set the number of SLIC superpixels to $N = 1,000$ per frame. We set the parameters of the hierarchical superpixel merging weights using a 9D grid search over their values, and evaluate the performance using a subset of 10 videos from the UCF Sports training set. For a given set of parameters, we generate the segmentation hierarchy, and for each node n in the hierarchy we evaluate the precision and recall w.r.t. the ground-truth object bounding box in all annotated frames. Precision $p(n)$ is defined as the fraction of the supervoxel that is inside the ground-truth box, and recall $r(n)$ the fraction of the ground-truth box that is inside the supervoxel. We then compute the maximum score of the product of recall and precision $F(n) = p(n)r(n)$ across all supervoxels in the hierarchy, and take the average of $\max_n F(n)$ across all annotated frames in all videos.

In experiments on the Xiph.org benchmark we do not use boundary features, since the resolution of 240×160 of the videos in this data set is too small to obtain accurate boundary estimates.

Supervoxel segmentation evaluation results. We compare our approach to GBH [Grundmann et al., 2010] and SWA [Corso et al., 2008], as well as the GB [Felzenszwalb and Huttenlocher, 2004], Nyström [Fowlkes et al., 2004], and meanshift [Paris and Durand, 2007] methods as evaluated in [Xu and Corso, 2012]. We used the publicly available implementation

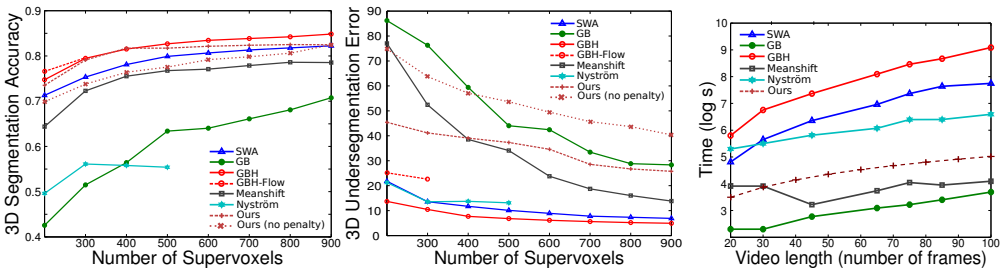


Figure 5.8 – Comparison of our and state-of-the-art supervoxel methods on the Xiph.org benchmark.

Figure 5.9 – Supervoxel comparison on videos of UCF Sports: video frames (top), GBH-Flow (middle), ours (bottom). For each video both methods are set to produce the same number of supervoxels.

in LIBSVX. [2] For GBH we also used the online processing service which also uses optical flow features that are not included in the SVX implementation. [3]

For our own method we also evaluate the effect of the penalty term α_{dis} that penalizes spatially distant supervoxel clusters.

The evaluation results are presented in Figure 5.8. In terms of segmentation accuracy (left) our method is comparable to the best methods: GBH and GBH-Flow. For the undersegmentation error (middle) our method gives significantly worse results than the best results obtained with GBH. The discrepancy of the evaluation results across these measures is due to the fact that our method tends to produce larger supervoxels, as well as many tiny supervoxels that consist of single isolated superpixels. Figure 5.9 illustrates this in comparison to GBH-Flow, where both methods

2. See http://www.cse.buffalo.edu/~jcorso/r/supervoxels.
3. See http://www.cc.gatech.edu/cpl/projects/videosegmentation.

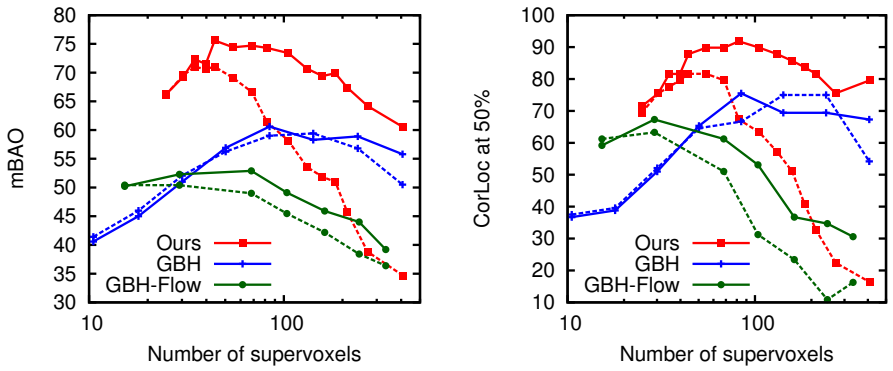


Figure 5.10 – Evaluation in terms of mBAO (left) and 50% CorLoc (right) of our, GBH, and GBH-Flow supervoxels on the UCF Sports train set using 1,000 proposals as a function of the supervoxel granularity. We show results with (solid) and without (dashed) the full-duration constraint.

are set to produce the same number of supervoxels for each video. Our method seems to produce less supervoxels due to isolated superpixels, which constitute more than 50% of the supervoxels. The undersegmentation error suffers from this, since a large supervoxel that overlaps a ground truth segment by a small fraction can deteriorate the error significantly.

Our method improves in both evaluation measures with the addition of the penalty term α_{dis} for spatially disconnected components, demonstrating its effectiveness. We therefore include it in all further experiments.

We compare the run times of the different supervoxel methods in the right panel of Figure 5.8. We do not include the GBH-Flow method here, since we ran it over the online service which does not allow us to evaluate its run time. As compared to GBH, the top performing method, our method runs one to two orders of magnitudes faster. Compared to the fastest method, GB, our method is only about 4 times slower, but we perform better in terms of 3D segmentation accuracy and similarly in terms of 3D undersegmentation error.

5.4.3 Evaluation of spatio-tempoal detection proposals

Video tube proposals for UCF Sports. In our first experiment we consider the performance using supervoxels from GBH, GBH-Flow, and our method with different granularities, i.e. different numbers of extracted supervoxels. In Figure 5.10 we compare the performance on the training set of the UCF Sports dataset. We consider the performance using 1,000

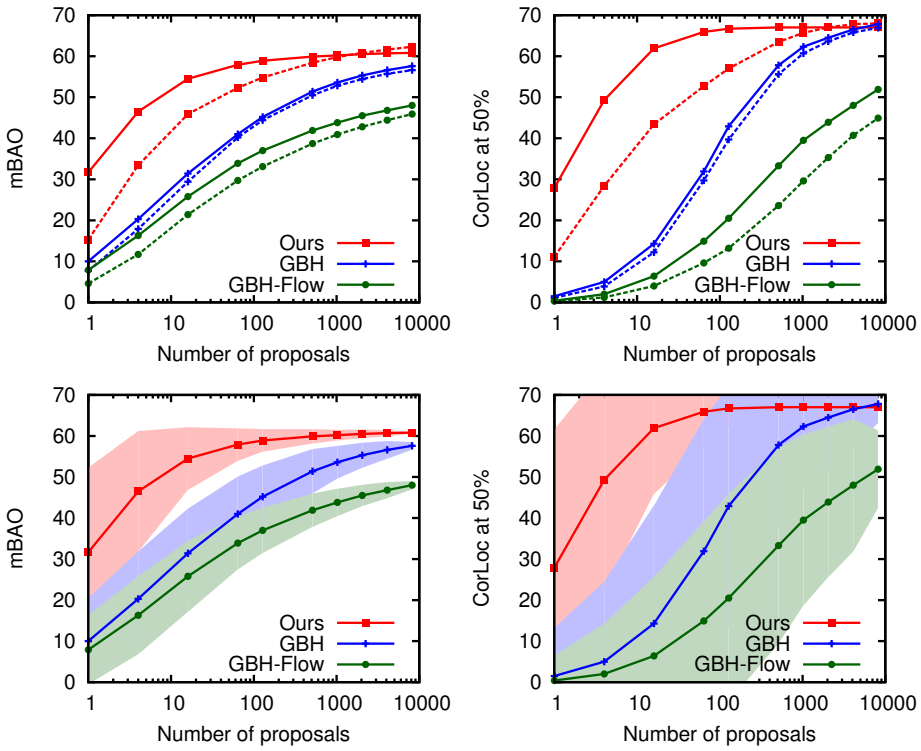


Figure 5.11 – UCF Sports test set performance against the number of proposals for our, GBH and GBH-Flow supervoxels. We show results with (solid) and without (dashed) the full-duration constraint. In the bottom plots, the filled area indicates the standard deviation of the performance given multiple samples of proposals.

proposals for different granularities. We train the logistic discriminant model for each segmentation and granularity. The results show that our supervoxels lead to substantially better proposals. The full-duration temporal constraint, which accepts proposals only if they span the full duration of the video, improves results for all methods. It is particularly important for our supervoxels when using finer segmentations, probably because it helps to deal with very short supervoxels that are frequent in our approach. Based on the mBAO performance we choose for each supervoxel method the optimal granularity, which will be used on the test set.

The results for the UCF Sports test set are given in Figure 5.11. For both evaluation measures, we need far fewer proposals for a given level of performance using our supervoxels, as compared to using GBH or GBH-

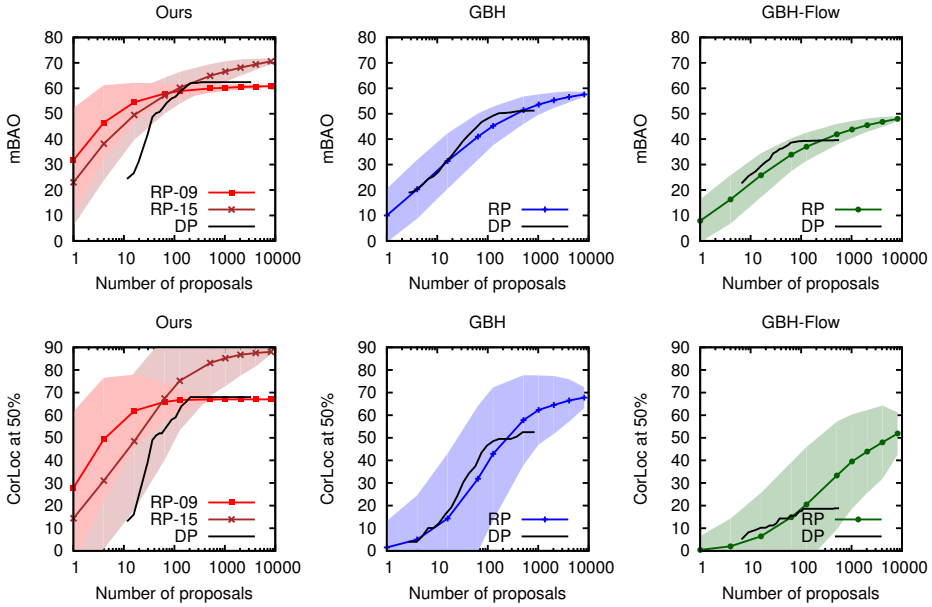


Figure 5.12 – UCF Sports test set performance against the number of proposals for the direct proposals baseline (DP) and the random Prim (RP) For our segmentation method we have included two levels of the segmentation: the best obtained on the train set (RP-09) and a finer level (RP-15).

Flow supervoxels. Also in this case the full-duration temporal constraint benefits the performance, particularly so when generating few proposals using our supervoxels. For the following experiments we enforce the full-duration temporal constraint for the proposals.

The supervoxel agglomeration algorithm is a randomized procedure, so it produces different results for different runs. To measure the variance in performance across runs, we perform a sampling procedure: we first generate $N = 10000$ proposals, then for a desired number of proposals M, we sample M proposals out of N without replacement. We repeat the sampling procedure 100 times and average the results. The bottom plots from Figure 5.11 show the mean and one standard deviation of the measures as a function of the number of proposals $M = \{1, 4, \ldots, 8196\}$. As expected there the result variance is large when generating few proposals and it decreases with the number of proposals. The CorLoc measure has a larger variance than mBAO, because the discrete nature of the CorLoc metric.

Next, we compare the randomized merging algorithm to a baseline

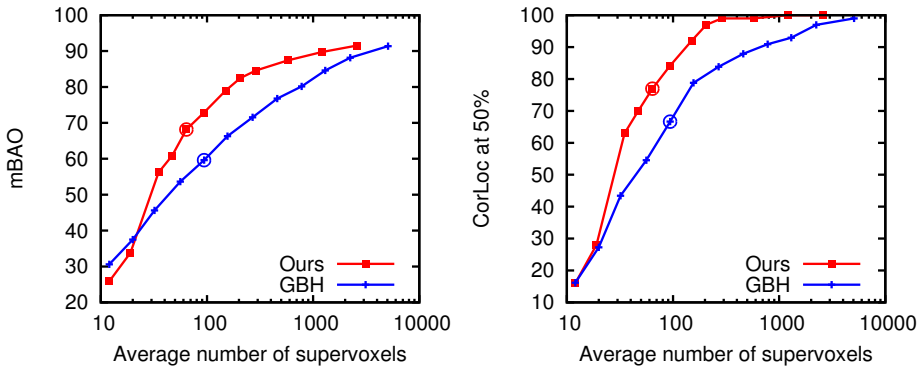


Figure 5.13 – Best achievable performance for GBH and our segmentations. The circles indicate the segmentation levels we cross-validated on the training set.

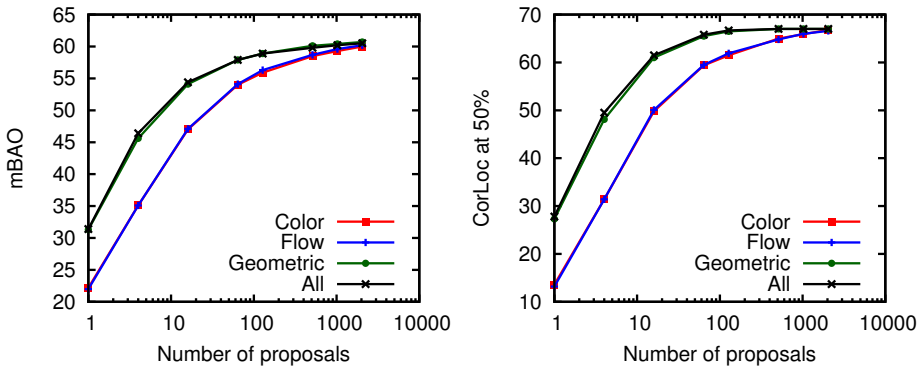


Figure 5.14 – Evaluating the features for our segmentation using the random Prim algorithm with the temporal constraint.

that generates proposals directly from a segmentation method. For a given segmentation, it converts each supervoxel into a tube proposal by enclosing it into a bounding box for each frames it appears in. We vary the number of proposals by changing the level of the segmentation: we start with the proposals generated from the coarsest segmentation and incrementally add proposals obtained at finer levels. We call this method *direct proposals* (DP). Figure 5.12 show the results for DP and the randomized merging algorithm, *random Prim* (RP), for the three segmentation methods. DP obtains good recall when using around 100 proposals, but after that point the performance stagnates, as the supervoxels become too small to correctly cover the object of interest. For the GBH and GBH-Flow

segmentations, RP outperforms DP as we generate more proposals. On the other hand, using our segmentation, RP performs better when using fewer proposals, but it also plateaus after 100 proposals. This behaviour happens because our segmentation is coarse and RP is unable to generate new tubes. For this reason, we experiment with a finer level (RP-15 in the figure). At this level our method performs better in the limit. Surprisingly, it also improves the previous results that used the granularity cross-validated on the UCF Sports train set (RP-09).

Figure 5.13 shows an upper bound on the performance for GBH and our segmentations. We find the best achievable performance by selecting the supervoxels such that they cover the groundtruth the best. More precisely, we first select all the supervoxels that are fully contained by the groundtruth, then we add supervoxels one by one, starting with those that are overlapping most with the groundtruth, until the average overlap between the groundtruth and the constructed tube starts decreasing; at that moment we stop the algorithm. The results show that as the segmentation grows finer, the best attainable performance goes up; this confirms that theoretically we can achieve perfect performance if the we have a fine enough segmentation. The results also indicate that our segmentation method is better for the proposal generation task than GBH, because it obtains a better possible recall given a number of supervoxels. For GBH we obtain an mBAO of 57.9% by using 8,196 proposals, while the upper bound on the mBAO for the same level is 59.6% (indicated by the circled point in Figure 5.13); for our segmentation method we obtain an mBAO of 60.8% by using 8,196 proposals, while the upper bound on the mBAO for the same level is 68.2%. These differences can be explained by various factors: not generating enough proposals, enforcing the full-duration temporal constraint, or allowing disconnected tubes in the upper bound. The 50% CorLoc measure has an elbow point at around 200 supervoxels for our segmentation; that point corresponds to level 15, which we previously used in the previous experiment shown in Figure 5.12.

In Figure 5.14, we investigate the importance of the different features in the randomized merging algorithm. We compare four variants of feature combinations: color, flow, geometric (the combination of features 3–8 from Section 5.3.2) and all the features For these experiments we use only our segmentation. We observe that the geometric features are the most important ones and they perform as well as using the entire feature combination. Having good features is of particular importance when we want to generate only a few proposals. To find which of the geometric features is dominant, we include the weights learnt on the geometric features in Table 5.1. According to these results, the randomized merging

Size	Fill	Spatial size	Spatial fill	Temporal size	Temporal overlap
−1.69	−0.92	−1.73	−1.22	−0.57	4.25

Table 5.1 – Learnt weights for the combination of geometric features.

favours the aggregation of temporarily overlapping supervoxels and of small supervoxels.

Video Tube Proposals for YouTube Objects. We now evaluate our approach on the YouTube Objects dataset. We use the randomized merging algorithm at two levels of our segmentation, at level 9 (RP-09) and at level 15 (RP-15). For comparison, we consider two previous method: the video object segmentation method of [Papazoglou and Ferrari, 2013], and the weakly supervised learning method of Prest et al. [2012a]. The video object segmentation approach of Papazoglou and Ferrari [2013] is unsupervised, and only outputs a single segmentation per shot. Prest et al. [2012a] used the method of Brox and Malik [2010] to output coherent motion segments, to which they fit spatio-temporal tubes. This leads to between 3 and 15 tubes per shot. They then use weakly supervised training to automatically select the tube corresponding to the object of interest, exploiting class labels given at the video level. We report their results, and the result for the best tube among the proposals from [Brox and Malik, 2010].

Figure 5.15 compares the aforementioned methods to ours when using between one and 8,196 proposals. When using 16 proposals, comparable to the number produced by Brox and Malik [2010], we obtain 49.6% CorLoc using the segmentation at level 9 (RP-09), a result that is almost 15% points above the 34.8% of Brox and Malik [2010]. As compared to the video object segmentation method of Papazoglou and Ferrari [2013], our method obtains similar results when using 16 proposals, but we further improve when more proposals are used. Prest et al. [2012a] obtain similar results to ours with a single proposal, but their method is class-dependent. As for the UCF-Sport, we observe that a finer segmentation gives better results if we use more proposals.

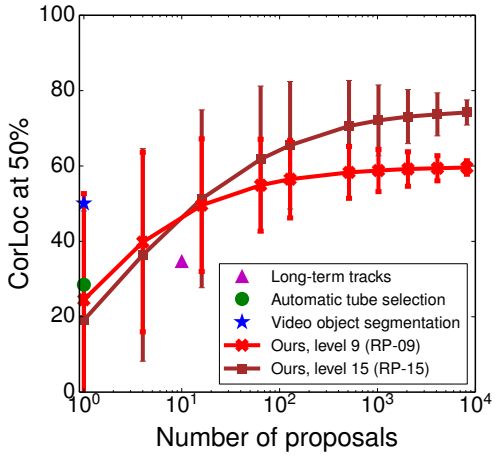


Figure 5.15 – Performance on the YouTube Objects dataset (50% CorLoc), as a function of the number of proposals. We show results for two levels of our supervoxel hierarchy (depicted as lines) and compare to other related methods (depicted as points): the long term-tracks of Brox and Malik [2010], the automatic tube selection of Prest et al. [2012a], the video object segmentation of Papazoglou and Ferrari [2013].

5.4.4 Discussion

Our experiments to suggest that the supervoxel evaluation measures of Xu and Corso [2012] are not directly indicative of the performance on proposal generation task. In its hierarchical clustering process our method quickly agglomerates large segments for objects, while also retaining a set of small fragments. In contrast, other methods, such as GBH, steadily produce larger segments from smaller ones, which leads to more homogeneously sized supervoxels. See Figure 5.9 for an example of our and GBH supervoxels at a given frame, and Figure 5.16 for the distribution of supervoxel durations. It seems that our more heterogeneous size distribution is advantageous for proposal generation, since good proposals can be made by merging a few large supervoxels. For spatio-temporal segmentation as measured by the supervoxel benchmark metrics, however, this unbalanced size distribution is sub-optimal since it is more likely to be imprecise at the object boundaries. Recent supervoxel evaluation metrics proposed by Galasso et al. [2013], which also assess temporal consistency and hierarchies, might be more related to the properties that are important for proposal generation.

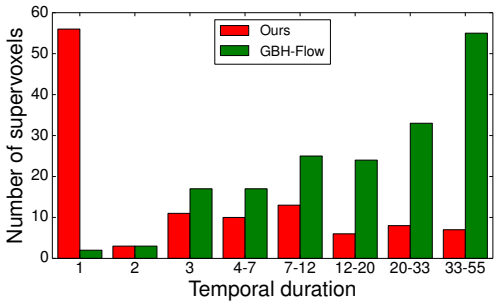


Figure 5.16 – Distribution of the supervoxel durations obtained on four video sequences from the UCF Sports dataset. Note the logarithmic binning of the duration, which is measured in frames. Both methods are set to produce a similar number of supervoxels per video.

5.5 Conclusion

In this chapter we have introduced a proposal algorithm, which can be used for the more general task of spatio-temporal localization. Our system has two main components.

First, we have presented a new supervoxel method, that performs a hierarchical clustering of superpixels in a graph with spatial and temporal connections. Experimental results demonstrate that our supervoxel method is efficient, and leads to state-of-the-art 3D segmentation accuracy. Its 3D undersegmentation error is significantly worse than the state-of-the-art, which is due to the fact that up to 50% of the final supervoxels consist of isolated superpixels. We believe that it is possible to significantly improve the undersegmentation error by agglomerating the isolated superpixels as a post-processing step.

Second, we have adapted the randomized Prim object proposal method to the spatio-temporal domain. To this end we have introduced a set of new pairwise supervoxel features, which are used to learn a similarity measure that favors grouping supervoxels that belong to the same physical object. Our results show that using the supervoxels leads to better proposals than using existing state-of-the-art supervoxel methods.

Chapter 6

Conclusions

Contents

In this thesis we have focused on three tasks that stem from the challenging problem of video understanding: action recognition, event recognition and action localization. Our contributions aim at two main goals: (i) build a robust, task-independent video representation; (ii) improve the efficiency of the methods, particularly for the localization task.

This chapter is organized as follows: Section 6.1 summarizes the contributions of the thesis and puts them in the context of our goals; Section 6.2 gives directions for further research inspired from the work done in the thesis.

6.1 Summary of contributions

Below we restate the contributions of the thesis and relate them to the goals we presented in the introduction.

In Chapter 3 we have compared local feature methods, encoding techniques and different ways of incorporating weak spatio-temporal information. We have found that the video representation based on the improved dense trajectories and the Fisher vector encoding obtains the best results. The excellent performance was confirmed across multiple datasets and tasks: action recognition on six popular benchmark datasets, event recognition in over 1,000 hours of video, action localization in feature-length movies. We also showed that our video representation surpasses more

sophisticated and highly-specialized models and, hence, it constitutes a robust and task-independent representation.

In Section 3.4, we have investigated how the action localization performance is affected by short windows that are retained by the non maxima suppression (NMS) algorithm. We propose a direct approach to mitigate this problem: re-score the windows by their duration before the NMS step. The re-scoring can be interpreted as setting a prior on longer windows. Empirical results show that this method brings significant improvements in the localization performance.

In Section 3.6.1, we show that the Fisher vector (FV) encoding is more efficient than its more popular counterpart, the bag of words (BOW) encoding. The computation cost of both BOW and FV grows linearly with the number of clusters, but FV can achieve better performance than BOW given the same number of clusters. In practice we have found FV suitable for large-scale datasets, such as the TRECVID MED dataset.

In Chapter 4 we improve the efficiency of FV in the context of action localization. We observed that if the scoring function was additive in the features, then we could localize the action by operating directly on the scored features. For example, temporal localization could be solved by using the efficient maximum sub-array search. But the FV normalizations are non-linear and, consequently, they break the additive property of the scoring function. Our contribution is an approximation that linearizes the normalizations for each of the k components of the Fisher vector, corresponding to the K Gaussian clusters. We show promising results, as we are able to obtain a speed-up of about an order of magnitude at a very little cost in performance.

In Chapter 5 we have improved the efficiency of localization by reducing the number of candidate windows. For spatio-temporal localization, exhaustive search over the space of tubes is infeasible. So we employ a proposal algorithm, the random Prim method, to group together supervoxels based on their appearance and geometric similarity. We show that only a few hundred proposals are sufficient to cover more than 60% of the ground-truth windows with at least 50% overlap. In Section 5.2 we compare the supervoxels produced by the graph-based hierarchical segmentation (GBH; Grundmann et al., 2010) and a novel method based on hierarchical clustering and features such as color, flow, edges, motion boundaries. We show that the new method is faster and performs better for the task of proposal generation than GBH. Given the segmentation, our proposal algorithm is generally fast, taking around a couple of minutes for each video.

6.2 Future research perspectives

In the following we describe how each of our three contributions can be further extended or combined.

Video representation The field of video understanding is progressing at a rapid pace, with numerous action recognition methods being regularly proposed. Our video representation constitutes a solid baseline for these recent methods. But we also believe that our pipeline is mature enough to be a building block for other, higher-level applications and real-life products. Content-based video indexing and wearable devices are examples of applications that could benefit from a robust video representation. However, in order to make the step towards real-word applications, more engineering efforts are required. First, an easy-to-use interface is mandatory if we want our representation to deployed on multiple platforms (web, mobile). Second, we should aim to further improve the speed of the pipeline. The computational cost is critical for real-world applications: wearable systems usually require real-time processing and low memory consumption; indexing applications deal with very large quantities of data. Currently, the speed of our pipeline is limited by the optical flow computation. Thus, it would be worthwhile exploring how faster optical flow methods, *e.g.*, [Tao et al., 2012], impact the final recognition performance.

Another direction of research could be towards a better understanding of what lies behind the strong performance. Visualization is one way to deepen our understanding, and it is currently successfully applied for convolutional neural network features [Zeiler and Fergus, 2014, Zhou et al., 2014]. We would like to investigate what are the local features or the dictionary components that are the most relevant for the classification task. We can get intuition in this direction by scoring local features or the dictionary centroids; similar work was done for images [Marszałek et al., 2009, Gandhi et al., 2013].

Approximating normalization of Fisher vectors We have applied our approximated FV normalization for the task of temporal action localization. The same idea is, however, useful whenever the bottleneck lies in the aggregation of the FV features. The most direct extensions are object detection in images and sub-volume action localization in videos. For these tasks branch-and-bound (BB) formulations have already been proposed [Lampert et al., 2009a, Yuan et al., 2009], hence, we can adapt those models by incorporating our bounding function, which is based on the approximated FV normalization. Even if not straightforward, we believe that it is possible to give a similar BB formulation for spatio-temporal

localization. The problem lies in the irregularity of the state search space: a window has a more complex representation than just the two opposite corners, as it was the case for the previous tasks. Assuming that our video is represented as super-voxels connected in a graph, the goal of spatio-temporal localization is to find the subgraph which maximizes the classification score. We represent the subgraph as a vector of binary values, one for each node of the full graph, denoting the inclusion or exclusion. At each iteration of the BB algorithm we branch over one of the binary variables, thus reducing the state space by half. BB has already been applied for graph partitioning and similar ideas might be exploited.

The Codemaps method Li et al. [2013b] proposes an exact way of speeding-up the ℓ_2 normalization. The idea is to store a matrix of pre-computed inner products between the FVs corresponding to the slices. Based on those quantities we can efficiently compute the ℓ_2 norm of an arbitrary region by summing up their corresponding partial ℓ_2 norms. As we have already seen in our results, and showed also by Cinbis et al. [2013], the ℓ_2 normalization is crucial for good localization performance. The Codemaps method induces an additional memory cost (storing the matrix of partial ℓ_2 norms), which can be significant if we are interested in fine localization (*i.e.*, we need to store a large number of slices). We could reduce the cost by assuming a maximum size for our windows. In this case, we will not need to save all the pairwise inner-products, but only for those units that are withing the maximum size. This induces significant sparsity in the matrix, making the use of Codemaps possible in practice.

The intuition of our normalization is based on the work of Cinbis et al. [2012]. They were the first to take a step towards a more principled motivation of the power normalization. We would like to encourage however further theoretical investigation in this direction. We believe that this could enable further contributions and ideas.

Spatio-temporal proposals Proposals for videos are still largely unexplored. An obvious direction of research would be to draw more inspiration from the image proposal techniques and investigate their use for generating spatio-temporal proposals. We believe that the community could benefit from an extensive evaluation of the individual components of a general proposal method. A general proposal method can be decomposed into the three parts: segmentation, features, aggregation. In the following we give some pointer on what could be explored on each of the three directions.

Segmentation There is an abundance of video segmentation methods, *e.g.*, [Corso et al., 2008, Grundmann et al., 2010, Xu et al., 2012,

Van den Bergh et al., 2013, Xu et al., 2013]. Most of those have been benchmarked in the context of segmentation [Xu and Corso, 2012], using metrics like the segmentation accuracy or under-segmentation error. However, as we have already seen in our experiments, Section 5.4, the segmentation performance is not particularly indicative of the recall performance on the proposal generation task. Thus, we suggest evaluating the segmentation algorithms for the task at hand.

Features In our work, as well as in related studies [Papazoglou and Ferrari, 2013, Jain et al., 2014], the proposals algorithm defines the similarity between pairs of super-voxels based on various appearance-based and geometrical features. These features are the input of a binary function—the similarity function—, but there exist features that are specific to unary functions. Examples of this type of features are those defined by Alexe et al. [2012] in the objectness measure: saliency, colour contrast, edge density, location. Some unary potential have also been proposed for videos. For example, Van den Bergh et al. [2013] proposes an objectness measure reminiscent of the straddling score [Alexe et al., 2012], which favors those proposals that fit tightly the bounding box; the objecteness is extended to videos by averaging over the temporal domain. We can derive unary features that favor the moving regions, based on previous work: (i) Papazoglou and Ferrari [2013] use inside-outside maps on motion boundaries to identify which pixels are inside the moving objects; (ii) Jain et al. [2014] propose independent motion evidence of that captures the residual motion in small sub-volumes. The unary potentials can be directly incorporated into the aggregation process, but we could use them to seed the proposal algorithm or to score and rank the obtained proposals. These uses should improve the recall at a fixed number of proposals.

Aggregation The way in which the super-voxels are aggregated into larger units influences the type of proposals we obtain. In our case we grow the proposals in a randomized manner starting from a random seed supervoxel. The probabilistic nature of our method ensures a diverse set of proposals. In theory, we can achieve perfect recall given a fine enough segmentation and a large enough number of runs. By contrast, the hierarchical aggregation [Van de Sande et al., 2011, Jain et al., 2014] is a deterministic method and it might be affected by merging wrong segments early on in the building process. These hierarchical approaches achieve variety by either changing the base segmentations [Van de Sande et al., 2011] or the

features [Jain et al., 2014]. Another way of obtaining proposals from a base segmentation is by using graph cut algorithms. For videos, Papazoglou and Ferrari [2013] use graph cut to obtain a single proposal per video, but models that generate more proposals could be inspired from the work on images [Carreira and Sminchisescu, 2012, Krähenbühl and Koltun, 2014].

Another direction to explore is the amount of supervision involved in the proposal generation process. Similar to [Manen et al., 2013], our method uses supervision to learn a weight combination for the different features. Others possible uses of supervision are proposal scoring [Carreira and Sminchisescu, 2012, Alexe et al., 2012] and seed placement Krähenbühl and Koltun [2014]. If our final goal is action localization, we can assume that we have access to classifiers for each of the action categories. So, we can incorporate more supervision into the proposal generation process by using the classifier score as a feature. This method produces class-specific proposals. This idea could benefit from our work on approximated normalization, because computing scores on groups of super-voxels can induce a significant computational cost.

In our work we have seen that by enforcing a full duration constraint of the spatio-temporal tubes we can improve results. However, this constraint is too strict sometimes: it happens that the object is occluded or disappears from the scene. We could favor temporarily long proposals in a soft way. For example, we could adapt the probability distribution (from which we draw select a new super-voxel to be added) based on the current shape of the proposal. If the proposal is not long enough, favor the supervoxels that increase the duration of the tube.

Publications

This thesis has led to several publications summarized below.

International conferences

- D. Oneata, J. Verbeek, C. Schmid. Action and event recognition with Fisher vectors on a compact feature set. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013.
- D. Oneata, J. Verbeek, and C. Schmid. Efficient action localization with approximately normalized Fisher vectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

Journal submissions

- H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. Submitted (minor revision) to the *International Journal of Computer Vision (IJCV)*, 2015.

Other publications

- D. Oneata, M. Douze, J. Revaud, J. Schwenninger, D. Potapov, H. Wang, Z. Harchaoui, J. Verbeek, C. Schmid, R. Aly, K. Mcguiness, S. Chen, N. O'Connor, K. Chatfield, O. Parkhi, R. Arandjelovic, A. Zisserman, F. Basura, and T. Tuytelaars. AXES at TRECVID 2012: KIS, INS, and MED. In *TRECVID Workshop*, 2012.

- R. Aly, R. Arandjelovic, K. Chatfield, M. Douze, B. Fernando, Z. Harchaoui, K. Mcguiness, N. O'Connor, D. Oneata, O. Parkhi, D. Potapov, J. Revaud, C. Schmid, J.-L. Schwenninger, D. Scott, T. Tuytelaars, J. Verbeek, H. Wang, and A. Zisserman. The AXES submissions at TRECVID 2013. In *TRECVID Workshop*, 2013.
- M. Douze, D. Oneata, M. Paulin, C. Leray, N. Chesneau, D. Potapov, J. Verbeek, K. Alahari, Z. Harchaoui, L. Lamel, J.-L. Gauvain, C. A. Schmidt, and C. Schmid. The INRIA-LIM-VocR and AXES submissions to TRECVID 2014 Multimedia Event Detection, 2014.
- D. Oneata, J. Verbeek, and C. Schmid. The LEAR submission at Thumos 2014. In *ECCV International Workshop and Competition on Action Recognition with a Large Number of Classes*, 2014.
- H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. *ArXiv e-prints*, 2015.

Bibliography

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34 (11):2274–2282, 2012. Cited on pages 89 and 92.

J. Aggarwal and Q. Cai. Human motion analysis: A review. *CVIU*, 73(3): 428–440, 1999. Cited on page 3.

J. Aggarwal and M. Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 43(3):1–43, Apr. 2011. Cited on page 9.

B. Alexe, T. Deselares, and V. Ferrari. Measuring the objectness of image windows. *PAMI*, 34(11):2189–2202, 2012. Cited on pages 25, 64, 88, 91, 115, and 116.

J. Almazan, A. Gordo, A. Fornés, and E. Valveny. Handwritten word spotting with corrected attributes. In *ICCV*, 2013. Cited on page 34.

R. Aly, R. Arandjelovic, K. Chatfield, M. Douze, B. Fernando, Z. Harchaoui, K. Mcguiness, N. O'Connor, D. Oneata, O. Parkhi, D. Potapov, J. Revaud, C. Schmid, J.-L. Schwenninger, D. Scott, T. Tuytelaars, J. Verbeek, H. Wang, and A. Zisserman. The AXES submissions at TRECVID 2013. In *TRECVID Workshop*, 2013. Cited on page 18.

S. An, P. Peursum, W. Liu, and S. Venkatesh. Efficient algorithms for subwindow search in object detection and localization. In *CVPR*, 2009. Cited on page 63.

R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012. Cited on page 30.

R. Arandjelović and A. Zisserman. All about VLAD. In *CVPR*, 2013. Cited on page 65.

P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011. Cited on page 94.

F. R. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *NIPS*, 2009. Cited on page 19.

N. Ballas, Y. Yang, Z.-Z. Lan, B. Delezoide, F. Prêteux, and A. Hauptmann. Space-time robust video representation for action recognition. In *ICCV*, 2013. Cited on pages 51, 52, and 53.

A. Barbu, A. Bridge, Z. Burchill, D. Coroian, S. Dickinson, S. Fidler, A. Michaux, S. Mussman, S. Narayanaswamy, D. Salvi, L. Schmidt, J. Shangguan, J. S. Mark, J. Waggoner, S. Wang, J. Wei, Y. Yin, and Z. Zhang. Video in sentences out. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, 2012. Cited on page 20.

H. Bay, A. Ess, T. Tuytelaars, and L. van Gool. SURF: Speeded up robust features. *CVIU*, 110(3):346–359, 2008. Cited on pages 18 and 28.

P. Beaudet. Rotationally invariant image operators. In *ICPR*, 1978. Cited on page 11.

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957. Cited on page 63.

H. Bilen, V. Namboodiri, and L. van Gool. Object and action classification with latent window parameters. *International Journal of Computer Vision*, 106(3):237–251, 2014. Cited on page 21.

M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *ICCV*, 2005. Cited on page 10.

M. Blaschko and C. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008. Cited on page 25.

A. F. Bobick. Movement, activity and action: The role of knowledge in the perception of motion. *Philosophical Transactions of the Royal Society of London*, 352(1358):1257–1265, 1997. Cited on page 20.

A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *CIVR*, 2007. Cited on page 21.

Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010a. Cited on page 14.

Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, 2010b. Cited on page 14.

M. Brand. Shadow puppetry. In *CVPR*, 1999. Cited on page 3.

W. Brendel and S. Todorovic. Learning spatio-temporal graphs of human activities. In *ICCV*, 2011. Cited on page 22.

T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. Cited on pages 12, 108, and 109.

T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *PAMI*, 2011. Cited on pages 93, 95, and 98.

Z. Cai, L. Wang, X. Peng, and Y. Qiao. Multi-view super vector for action recognition. In *CVPR*, 2014. Cited on page 15.

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

L. W. Campbell and A. F. Bobick. Recognition of human body motion using phase space constraints. In *ICCV*, 1995. Cited on page 2.

L. Cao, Z. Liu, and T. Huang. Cross-dataset action detection. In *CVPR*, 2010. Cited on page 24.

L. Cao, Y. Mu, A. Natsev, S.-F. Chang, G. Hua, and J. Smith. Scene aligned pooling for complex video recognition. In *ECCV*, 2012. Cited on page 22.

J. Carreira and C. Sminchisescu. CPMC: Automatic object segmentation using constrained parametric min-cuts. *PAMI*, 34(7):1312–1328, 2012. Cited on pages 26 and 116.

J. Carreira, R. Caseiroa, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012. Cited on page 90.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–27, 2011. Cited on page 36.

K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: An evaluation of recent feature encoding methods. In *BMVC*, 2011. Cited on pages 15, 27, 33, and 79.

A. Chen and J. Corso. Propagating multi-class pixel labels throughout video frames. In *Proceedings of Western New York Image Processing Workshop*, 2010. Cited on page 99.

Q. Chen, Z. Song, R. Feris, A. Datta, L. Cao, Z. Huang, and S. Yan. Efficient maximum appearance search for large-scale object detection. In *CVPR*, 2013. Cited on pages 62, 63, and 64.

G. Cheng, Y. Wan, A. N. Saudagar, K. Namuduri, and B. P. Buckles. Advances in human action recognition: A survey. *ArXiv e-prints*, 2015. Cited on page 9.

M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*, pages 3286–3293, 2014. Cited on page 25.

K. Church and W. Gale. Poisson mixtures. *Natural Language Engineering*, 1(02):163–190, 1995. Cited on page 65.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014. Cited on page 34.

R. Cinbis, J. Verbeek, and C. Schmid. Image categorization using Fisher kernels of non-iid image models. In *CVPR*, 2012. Cited on pages 15, 65, 66, and 114.

R. Cinbis, J. Verbeek, and C. Schmid. Segmentation driven object detection with Fisher vectors. In *ICCV*, 2013. Cited on pages 34, 64, 88, 91, and 114.

S. Clinchant, J.-M. Renders, and G. Csurka. Trans-media pseudo-relevance feedback methods in multimedia retrieval. In *Advances in Multilingual and Multimodal Information Retrieval*, 2008. Cited on page 142.

J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. Yuille. Efficient multilevel brain tumor segmentation with integrated Bayesian model classification. *IEEE Transactions on Medical Imaging*, 27(5):629–640, 2008. Cited on pages 89, 101, and 114.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995. Cited on page 36.

G. Csurka and F. Perronnin. An efficient approach to semantic segmentation. *IJCV*, 95(2):198–212, 2011. Cited on page 64.

G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, 2004. Cited on pages 13 and 32.

O. Cula and K. Dana. Compact representation of bidirectional texture functions. In *CVPR*, 2001. Cited on page 32.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. Cited on pages 11, 28, and 29.

T. Darrell and A. Pentland. Space-time gestures. In *CVPR*, 1993. Cited on page 3.

P. Das, C. Xu, R. Doell, and J. Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *CVPR*, 2013. Cited on page 20.

P. Dollár and C. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. Cited on page 93.

P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *IEEE Workshop Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005. Cited on pages 11 and 28.

O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009. Cited on pages 22, 23, 27, 43, 44, 55, 56, 82, and 83.

I. Endres and D. Hoiem. Category independent object proposals. In *ECCV*, 2010. Cited on pages 88, 91, and 92.

M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge 2009 results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop, 2009. Cited on page 32.

M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge 2010 results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop, 2010. Cited on page 32.

M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge 2011 results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop, 2011. Cited on page 32.

C. Fanti, L. Zelnik-Manor, and P. Perona. Hybrid models for human motion recognition. In *CVPR*, 2005. Cited on page 11.

A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. Cited on page 20.

G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the Scandinavian Conference on Image Analysis*, 2003. Cited on pages 12, 29, and 31.

P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. Cited on pages 89 and 101.

P. Felzenszwalb, R. Grishick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 32(9): 1627–1645, 2010. Cited on pages 23, 24, and 31.

V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *CVPR*, 2008. Cited on page 31.

M. A. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, C-22(1):67–92, 1973. Cited on page 23.

C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *PAMI*, 26(2):214–225, 2004. Cited on pages 89 and 101.

K. Fragkiadaki, P. Arbelaez, P. Felsen, and J. Malik. Learning to segment moving objects in videos. In *CVPR*, 2015. Cited on pages 91 and 92.

A. Gaidon, Z. Harchaoui, and C. Schmid. Actom sequence models for efficient action detection. In *CVPR*, 2011. Cited on pages 22, 23, 44, 56, 82, and 83.

A. Gaidon, Z. Harchaoui, and C. Schmid. Activity representation with motion hierarchies. *IJCV*, 107(3):219–238, 2013. Cited on pages 12, 51, and 52.

F. Galasso, N. Nagaraja, T. Cardenas, T. Brox, and B.Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, 2013. Cited on page 109.

A. Gandhi, K. Alahari, and C. Jawahar. Decomposing bag of words histograms. In *ICCV*, 2013. Cited on page 113.

D. M. Gavrila. The visual analysis of human movement: A survey. *CVIU*, 73(1):82–98, 1999. Cited on pages 3 and 9.

E. Gavves, B. Fernando, C. Snoek, A. Smeulders, and T. Tuytelaars. Fine-grained categorization by alignments. In *ICCV*, Sydney, Australia, 2013. Cited on page 34.

P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009. Cited on page 19.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. Cited on pages 88 and 91.

R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In *CVPR*, 2015. Cited on page 9.

M. Gönen and E. Alpaydın. Multiple kernel learning algorithms. *JMLR*, 12:2211–2268, 2011. Cited on page 19.

L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *PAMI*, 29(12):2247–2253, 2007. Cited on page 11.

M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010. Cited on pages 89, 101, 112, and 114.

A. Gupta, A. Kembhavi, and L. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *PAMI*, 31(10):1775–1789, 2009. Cited on page 22.

A. Habibian, K. E. van de Sande, and C. G. Snoek. Recommendations for video event recognition using concept vocabularies. In *ACM Multimedia*, 2013. Cited on page 20.

C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988. Cited on page 11.

H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009. Cited on page 63.

D. Hogg. Model-based vision: A program to see a walking person. *IVC*, 1 (1):5–20, 1983. Cited on page 3.

S. J. Hwang, K. Grauman, and F. Sha. Semantic kernel forests from multiple taxonomies. In *NIPS*, pages 1718–1726, 2012. Cited on page 19.

S. Intille and A. Bobick. Representation and visual recognition of complex, multi-agent actions using belief networks. Technical Report 454, MIT, 1998. Cited on page 20.

H. Izadinia and M. Shah. Recognizing complex events using large margin joint low-level event model. In *ECCV*, 2012. Cited on page 22.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999. Cited on pages 15, 33, and 34.

M. Jain, H. Jégou, and P. Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013. Cited on pages 50, 51, 52, 80, and 81.

M. Jain, J. van Gemert, P. Bouthemy, H. Jégou, and C. Snoek. Action localization with tubelets from motion. In *CVPR*, 2014. Cited on pages 22, 26, 91, 115, and 116.

H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *CVPR*, 2009. Cited on page 65.

H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *PAMI*, 34(9): 1704–1716, 2012. Cited on pages 13, 15, 34, 65, 66, 79, and 81.

S. Ji, W. Xu, M. Yang, and K. Yu. 3D convolutional neural networks for human action recognition. *PAMI*, 35(1):221–231, 2013. Cited on page 17.

Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org`, 2013. Cited on page 142.

Y.-G. Jiang, X. Zeng, G. Ye, S. Bhattacharya, D. Ellis, M. Shah, and S.-F. Chang. Columbia-UCF TRECVID 2010 multimedia event detection: Combining multiple modalities, contextual concepts, and temporal matching. In *TRECVID Workshop*, 2010. Cited on pages 18 and 32.

Y.-G. Jiang, Q. Dai, X. Xue, W. Liu, and C.-W. Ngo. Trajectory-based modeling of human actions with motion reference points. In *ECCV*, 2012. Cited on pages 50, 51, and 52.

Y.-G. Jiang, S. Bhattacharya, S.-F. Chang, and M. Shah. High-level event recognition in unconstrained videos. *International Journal of Multimedia Information Retrieval*, 2(2):73–101, 2013a. Cited on page 9.

Y.-G. Jiang, J. Liu, A. Roshan Zamir, I. Laptev, M. Piccardi, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/ICCV13-Action-Workshop/, 2013b. Cited on pages 43 and 53.

Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14, 2014. Cited on page 141.

G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics*, 14(2):201–211, 1973. Cited on page 11.

B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290:91–97, 1981. Cited on page 32.

S. Karaman, L. Seidenari, A. D. Bagdanov, and A. Del Bimbo. L1-regularized logistic regression stacking and transductive CRF smoothing for action recognition in video. In *ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013. Cited on pages 52 and 53.

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. Cited on page 17.

S. M. Katz. Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, 2(01):15–59, 1996. Cited on page 65.

I. Kim, S. Oh, A. Vahdat, K. Cannons, A. Perera, and G. Mori. Segmental multi-way local pooling for video recognition. In *ACM Multimedia*, 2013. Cited on page 59.

A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3D-gradients. In *BMVC*, 2008. Cited on pages 11, 23, and 28.

A. Kläser, M. Marszałek, C. Schmid, and A. Zisserman. Human focused action localization in video. In *ECCV Workshop on Sign, Gesture, and Activity*, 2010. Cited on pages 22, 24, 44, and 56.

B. Klein, G. Lev, G. Sadeh, and L. Wolf. Fisher vectors derived from hybrid Gaussian-Laplacian mixture models for image annotation. *ArXiv e-prints*, 2014. Cited on page 15.

A. Kojima, T. Tamura, and K. Fukunaga. Natural language description of human activities from video images based on concept hierarchy of actions. *IJCV*, 50(2):171–184, Nov. 2002. Cited on page 20.

P. Krähenbühl and V. Koltun. Geodesic object proposals. In *ECCV*, 2014. Cited on pages 26, 88, and 116.

J. Krapac, J. Verbeek, and F. Jurie. Modeling spatial layout with Fisher vectors for image categorization. In *ICCV*, 2011. Cited on pages 21 and 35.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. Cited on page 17.

H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011. Cited on pages 42 and 51.

L. Lamel. Multilingual speech processing activities in Quaero: Application to multimedia search in unstructured data. In *Human Language Technologies - The Baltic Perspective*, 2012. Cited on page 143.

L. Lamel and J.-L. Gauvain. Speech processing for audio indexing. In *Advances in Natural Language Processing*, 2008. Cited on page 143.

C. Lampert, M. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *PAMI*, 31(12): 2129–2142, 2009a. Cited on pages 24, 62, 63, 76, 90, and 113.

C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009b. Cited on page 20.

T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *ICCV*, 2011. Cited on pages 22, 23, and 24.

I. Laptev. On space-time interest points. *IJCV*, 64(2/3):107–123, 2005. Cited on pages 11 and 28.

I. Laptev and P. Pérez. Retrieving actions in movies. In *ICCV*, 2007. Cited on pages 27, 43, 44, 55, 56, 82, and 83.

I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. Cited on pages 11, 21, 29, 35, and 52.

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. Cited on pages 21 and 81.

Q. Le, W. Zou, S. Yeung, and A. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011. Cited on page 17.

T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43(1):29–44, 2001. Cited on page 32.

W. Li, Q. Yu, A. Divakaran, and N. Vasconcelos. Dynamic pooling for complex event recognition. In *ICCV*, 2013a. Cited on pages 21, 51, 52, 58, and 59.

Z. Li, E. Gavves, K. van de Sande, C. Snoek, and A. Smeulders. Codemaps: Segment classify and search objects locally. In *ICCV*, 2013b. Cited on pages 34, 64, 88, 90, and 114.

J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos. In *CVPR*, 2009. Cited on pages 11 and 52.

J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *CVPR*, 2011. Cited on page 20.

L. Liu, C. Shen, L. Wang, A. van den Hengel, and C.Wang. Encoding high dimensional local features by sparse coding based Fisher vectors. In *NIPS*, 2014. Cited on page 15.

D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. Cited on pages 18, 28, and 142.

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. Cited on page 92.

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 7, pages 674–679. Morgan Kaufmann, San Mateo, CA, USA, 1981. Cited on page 12.

S. Ma, J. Zhang, N. Ikizler-Cinbis, and S. Sclaroff. Action recognition and localization by hierarchical space-time segments. In *ICCV*, 2013a. Cited on pages 51 and 52.

T. Ma and L. Latecki. Maximum weight cliques with mutex constraints for video object segmentation. In *CVPR*, 2012. Cited on page 91.

Z. Ma, Y. Yang, Z. Xu, S. Yan, N. Sebe, and A. Hauptmann. Complex event detection via multi-source video attributes. In *CVPR*, 2013b. Cited on page 21.

J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In *ICCV*, 1999. Cited on page 32.

S. Manen, M. Guillaumin, and L. van Gool. Prime object proposals with randomized Prim's algorithm. In *ICCV*, 2013. Cited on pages 7, 25, 26, 64, 88, 91, 96, 99, and 116.

D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London: Biological Sciences*, 200(1140):269–294, 1978. Cited on page 3.

M. Marszałek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, 2009. Cited on pages 11, 41, 42, and 113.

S. Mathe and C. Sminchisescu. Dynamic eye movement datasets and learnt saliency models for visual action recognition. In *ECCV*, 2012. Cited on pages 50 and 52.

P. Matikainen, M. Hebert, and R. Sukthankar. Representing pairwise spatial and temporal relations for action recognition. In *ECCV*, 2010. Cited on page 22.

P. K. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *ICCV Workshop on Video-Oriented Object and Event Classification*, 2009. Cited on pages 11 and 12.

M. Mazloom, E. Gavves, and C. Snoek. Conceptlets: Selective semantics for classifying video events. *IEEE Transactions on Multimedia*, 16(8): 2214–2228, 2014. Cited on page 21.

S. McCann and D. Lowe. Spatially local coding for object recognition. In *ACCV*, 2012. Cited on page 35.

M. Merler, B. Huang, L. Xie, G. Hua, and A. Natsev. Semantic model vectors for complex video event recognition. *IEEE Transactions on Multimedia*, 14(1):88–101, Feb 2012. Cited on page 20.

R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *ICCV*, 2009. Cited on page 11.

T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *CVIU*, 81(3):231–268, 2001. Cited on page 10.

T. B. Moeslund, A. Hilton, and V. Krger. A survey of advances in vision-based human motion capture and analysis. *CVIU*, 104(2–3):90–126, 2006. Cited on pages 3 and 9.

O. R. Murthy and R. Goecke. Ordered trajectories for large scale human action recognition. In *ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013a. Cited on page 53.

O. R. Murthy and R. Goecke. Combined ordered and improved trajectories for large scale human action recognition. In *ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013b. Cited on pages 52 and 53.

G. K. Myers, R. Nallapati, J. van Hout, S. Pancoast, R. Nevatia, C. Sun, A. Habibian, D. Koelma, K. van de Sande, A. Smeulders, and C. Snoek. Evaluating multimedia features and fusion for example-based event detection. *Machine Vision and Applications*, 25(1):17–32, 2014. Cited on page 19.

P. Natarajan, P. Natarajan, V. Manohar, S. Wu, S. Tsakalidis, S. N. Vitaladevuni, X. Zhuang, R. Prasad, G. Ye, D. Liu, I. Jhuo, S. Chang, H. Izadinia, I. Saleemi, M. Shah, B. White, T. Yeh, and L. Davis. BBN VISER TRECVID 2011 multimedia event detection system. In *TRECVID Workshop*, 2011. Cited on page 32.

P. Natarajan, S. Wu, S. Vitaladevuni, X. Zhuang, S. Tsakalidis, U. Park, R. Prasad, and P. Natarajan. Multimodal feature fusion for robust event detection in web videos. In *CVPR*, 2012. Cited on page 18.

J. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010. Cited on pages 11, 22, 23, 42, 43, and 51.

S. Niyogi and E. Adelson. Analyzing and recognizing walking figures in XYT. In *CVPR*, 1994. Cited on page 2.

A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001. Cited on page 18.

D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with Fisher vectors on a compact feature set. In *ICCV*, 2013. Cited on pages 7, 79, 80, 81, 82, and 83.

D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In *ECCV*, 2014a. Cited on page 7.

D. Oneata, J. Verbeek, and C. Schmid. Efficient action localization with approximately normalized Fisher vectors. In *CVPR*, 2014b. Cited on page 7.

P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, B. Shaw, W. Kraaij, A. Smeaton, and G. Quénot. TRECVID 2012 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID*, 2012. Cited on pages 28 and 44.

A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013. Cited on pages 91, 92, 101, 108, 109, 115, and 116.

S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. In *CVPR*, 2007. Cited on pages 89 and 101.

A. Patron-Perez, M. Marszalek, A. Zisserman, and I. Reid. High Five: Recognising human interactions in TV shows. In *BMVC*, 2010. Cited on pages 11, 43, and 51.

O. Pele and M. Werman. Fast and robust earth mover's distances. In *ICCV*, 2009. Cited on page 93.

X. Peng, L. Wang, Z. Cai, Y. Qiao, and Q. Peng. Hybrid super vector with improved dense trajectories for action recognition. In *ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013. Cited on page 52.

X. Peng, Y. Qiao, and Q. Peng. Boosting VLAD with supervised dictionary learning and high-order statistics. In *ECCV*, 2014. Cited on page 16.

X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *ArXiv e-prints*, 2014. Cited on pages 15 and 19.

X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked Fisher vectors. In *ECCV*, 2014. Cited on page 16.

F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. Cited on pages 34 and 64.

F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010. Cited on pages 14, 36, 62, 65, 66, and 90.

P. J. Phillips and A. J. O'Toole. Comparison of human and computer performance across face recognition experiments. *IVC*, 32(1):74–85, 2014. Cited on page 1.

R. Poppe. A survey on vision-based human action recognition. *IVC*, 28 (6):976–990, 2010. Cited on page 9.

A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, 2012a. Cited on pages 100, 108, and 109.

A. Prest, C. Schmid, and V. Ferrari. Weakly supervised learning of interactions between humans and objects. *PAMI*, 34(3):601–614, 2012b. Cited on page 31.

A. Prest, V. Ferrari, and C. Schmid. Explicit modeling of human-object interactions in realistic videos. *PAMI*, 35(4):835–848, 2013. Cited on page 22.

L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., 1993. Cited on page 18.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. Cited on page 3.

C. Rao, A. Yilmaz, and M. Shah. View-invariant representation and recognition of actions. *IJCV*, 50(2):203–226, 2002. Cited on page 11.

K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013. Cited on pages 43 and 52.

L. G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963. Cited on page 3.

M. Rodriguez, J. Ahmed, and M. Shah. Action MACH: A spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008. Cited on page 100.

K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image Understanding*, 59(1):94–115, 1994. Cited on page 3.

M. Rohrbach, Q. Wei, I. Titov, S. Thater, M. Pinkal, and B. Schiele. Translating video content to natural language descriptions. In *ICCV*, 2013. Cited on page 20.

C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004. Cited on page 26.

S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. Cited on page 20.

H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978. Cited on page 3.

J. Sánchez, F. Perronnin, and T. de Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 33(16): 2216–2223, 2012. Cited on page 36.

J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013. Cited on pages 13, 15, 27, 35, 36, 65, and 141.

P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *IJCV*, 80(1):72–91, 2008. Cited on page 11.

M. Sapienza, F. Cuzzolin, and P. Torr. Learning discriminative space-time actions from weakly labelled videos. In *BMVC*, 2012. Cited on page 22.

S. Satkin and M. Hebert. Modeling the temporal extent of actions. In *ECCV*, 2010. Cited on page 22.

C. Schmid. Constructing models for content-based image retrieval. In *CVPR*, 2001. Cited on page 32.

C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: a local SVM approach. In *ICPR*, 2004. Cited on page 10.

P. Scovanner, S. Ali, and M. Shah. A 3-dimensional SIFT descriptor and its application to action recognition. In *ACM Multimedia*, 2007. Cited on page 28.

T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, 2005. Cited on page 14.

E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *CVPR*, 2007. Cited on page 18.

Z. Shi, T. Hospedales, and T. Xiang. Bayesian joint topic modelling for weakly supervised object localisation. In *ICCV*, 2013. Cited on page 52.

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. Cited on page 17.

K. Simonyan, O. Parkhi, A. Vedaldi, and A. Zisserman. Fisher vector faces in the wild. In *BMVC*, 2013a. Cited on page 34.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Fisher networks for large-scale image classification. In *NIPS*, 2013b. Cited on page 16.

J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. Cited on page 32.

Y. Song, L. Goncalves, and P. Perona. Unsupervised learning of human motion. *PAMI*, 25(7):814–827, 2003. Cited on page 11.

K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. Technical Report CRCV-TR-12-01, University of Central Florida, 2012. Cited on pages 43 and 53.

T. E. Starner and A. Pentland. Visual recognition of american sign language using hidden Markov models. In *International Symposium on Computer Vision*, 1995. Cited on page 4.

C. Sun and R. Nevatia. ACTIVE: activity concept transitions in video event classification. In *ICCV*, 2013. Cited on page 15.

J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In *CVPR*, 2009. Cited on page 11.

N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *ECCV*, 2010. Cited on page 95.

V. Sydorov, M. Sakurada, and C. Lampert. Deep Fisher kernels: End to end learning of the Fisher kernel GMM parameters. In *CVPR*, 2014. Cited on page 16.

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. Cited on page 1.

K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012. Cited on pages 21, 22, 44, and 59.

K. Tang, B. Yao, L. Fei-Fei, and D. Koller. Combining the right features for complex event recognition. In *ICCV*, 2013. Cited on pages 19, 59, and 60.

M. Tao, J. Bai, P. Kohli, and S. Paris. SimpleFlow: A non-iterative, sublinear optical flow algorithm. In *Computer Graphics Forum*, 2012. Cited on page 113.

E. Taralova, F. de la Torre, and M. Hebert. Motion words for videos. In *ECCV*, 2014. Cited on page 16.

G. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *ECCV*, 2010. Cited on page 17.

Y. Tian, R. Sukthankar, and M. Shah. Spatio-temporal deformable part models for action detection. In *CVPR*, 2013. Cited on pages 22 and 23.

D. Tran and J. Yuan. Optimal spatio-temporal path discovery for video event detection. In *CVPR*, 2011. Cited on pages 22, 24, 87, and 90.

D. Tran and J. Yuan. Max-margin structured output regression for spatio-temporal action localization. In *NIPS*, pages 350–358, 2012. Cited on pages 22 and 25.

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: Generic features for video analysis. *ArXiv e-prints*, 2014. Cited on page 17.

H. Uemura, S. Ishikawa, and K. Mikolajczyk. Feature tracking and motion compensation for action recognition. In *BMVC*, 2008. Cited on page 12.

J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013. Cited on pages 64, 88, 91, 99, and 100.

A. Vahdat and G. Mori. Handling uncertain tags in visual recognition. In *ICCV*, 2013. Cited on page 59.

A. Vahdat, K. Cannons, G. Mori, S. Oh, and I. Kim. Compositional models for video event detection: A multiple kernel learning latent variable approach. In *ICCV*, 2013. Cited on pages 21, 58, and 59.

K. van de Sande, T. Gevers, and C. Snoek. Evaluating color descriptors for object and scene recognition. *PAMI*, 32(9):1582–1596, 2010. Cited on page 18.

K. van de Sande, J. Uijlings, T. Gevers, and A. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, 2011. Cited on pages 25, 26, and 115.

K. van de Sande, C. Snoek, and A. Smeulders. Fisher and VLAD with FLAIR. In *CVPR*, 2014. Cited on pages 88 and 90.

M. van den Bergh, G. Roig, X. Boix, S. Manen, and L. van Gool. Online video SEEDS for temporal window objectness. In *ICCV*, 2013. Cited on pages 89, 91, and 115.

L. van der Maaten. Learning discriminative Fisher kernels. In *ICML*, 2011. Cited on page 16.

J. van Gemert, C. Veenman, A. Smeulders, and J.-M. Geusebroek. Visual word ambiguity. *PAMI*, 32(7):1271–1283, 2010. Cited on pages 13 and 33.

M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *ICCV*, 2002. Cited on page 32.

A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009. Cited on page 63.

P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 57(2): 137–154, 2004. Cited on pages 24, 62, and 63.

F. Wang, Z. Suny, D. Zhang, and C. Ngo. Semantic indexing and multimedia event detection: ECNU at TRECVID 2012. In *TRECVID Workshop*, 2012a. Cited on page 32.

H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. Cited on pages 12, 29, 30, 31, 53, 80, 81, 88, 90, and 141.

H. Wang, M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009. Cited on page 12.

H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, 2013a. Cited on pages 12, 28, 29, 30, 45, 50, 51, 53, 79, 80, and 81.

J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. Cited on page 13.

L. Wang, Y. Qiao, X. Tang, et al. Mining motion atoms and phrases for complex action recognition. In *ICCV*, pages 2680–2687, 2013b. Cited on pages 51, 52, and 53.

L. Wang, Y. Qiao, and X. Tang. Latent hierarchical model of temporal structure for complex activity classification. *IEEE Transactions on Image Processing*, 23(2):810–822, Feb 2014a. Cited on page 22.

L. Wang, Y. Qiao, and X. Tang. Video action detection with relational dynamic-poselets. In *ECCV*, 2014b. Cited on page 22.

X. Wang, L. Wang, and Y. Qiao. A comparative study of encoding, pooling and normalization methods for action recognition. In *ACCV*, 2012b. Cited on page 15.

X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *ICCV*, 2013c. Cited on page 91.

O. Weber, Y. Devir, A. Bronstein, M. Bronstein, and R. Kimmel. Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics*, 2008. Cited on page 93.

D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *CVIU*, 115(2): 224–241, 2011. Cited on page 9.

G. Willems, T. Tuytelaars, and L. van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, 2008. Cited on page 11.

G. Willems, J. Becker, T. Tuytelaars, and L. van Gool. Exemplar-based action recognition in video. In *BMVC*, 2009. Cited on page 28.

A. Wilson and A. Bobick. Learning visual behavior for gesture analysis. In *International Symposium on Computer Vision*, 1995. Cited on page 4.

J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005. Cited on page 66.

J. Wu, Y. Zhang, and W. Lin. Towards good practices for action video encoding. In *CVPR*, 2014. Cited on page 15.

S. Wu, O. Oreifej, and M. Shah. Action recognition in videos acquired by a moving camera using motion decomposition of Lagrangian particle trajectories. In *ICCV*, 2011. Cited on page 12.

C. Xu and J. Corso. Evaluation of super-voxel methods for early video processing. In *CVPR*, 2012. Cited on pages 26, 89, 99, 101, 109, and 115.

C. Xu, C. Xiong, and J. Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012. Cited on pages 89 and 114.

C. Xu, S. Whitt, and J. Corso. Flattening supervoxel hierarchies by the uniform entropy slice. In *ICCV*, 2013. Cited on pages 90 and 115.

J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden Markov model. In *CVPR*, 1992. Cited on pages 2, 3, and 4.

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009. Cited on page 13.

X. Yang and Y.-L. Tian. Action recognition using super sparse coding vector with spatio-temporal awareness. In *ECCV*, 2014. Cited on pages 16 and 21.

L. Yeffet and L. Wolf. Local trinary patterns for human action recognition. In *ICCV*, 2009. Cited on page 28.

G. Yu, J. Yuan, and Z. Liu. Propagative Hough voting for human activity recognition. In *ECCV*, 2012. Cited on page 52.

J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, 2009. Cited on pages 22, 24, 63, 87, 90, and 113.

J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *ArXiv e-prints*, 2015. Cited on page 17.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. Cited on page 113.

D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*, 2013. Cited on page 91.

J. Zheng, Z. Jiang, R. Chellappa, and J. P. Phillips. Submodular attribute selection for action recognition in video. In *NIPS*, 2014. Cited on page 21.

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene CNNs. *ArXiv e-prints*, 2014. Cited on page 113.

X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010. Cited on page 13.

J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu. Action recognition with actons. In *ICCV*, 2013. Cited on pages 50, 51, and 52.

C. Zitnick and P. Dollár. Edge boxes: locating object proposals from edges. In *ECCV*, 2014. Cited on pages 25 and 88.

Appendix A

Participation to THUMOS 2014

In this chapter we describe our entry in the THUMOS Challenge 2014. The goal of the THUMOS Challenge is to evaluate action recognition approaches in realistic conditions. In particular the test data consists of untrimmed videos, where the action may be short compared to the video length, and multiple instances can be present in each video. We have ranked second for the classification task and first for the temporal localization task. For full details on the definition of the challenge, task, datasets, and results, we refer to the challenge website [Jiang et al., 2014].

Below, we describe our systems for classification and localization in Section A.1, and present experimental results in Section A.2.

A.1 System description

We first describe our classification system to recognize untrimmed action videos in Section A.1.1. The localization system presented in Section A.1.2 is similar, but trained to recognize temporally cropped actions instead of complete untrimmed videos. The localization system also exploits the classification scores obtained for complete videos as a contextual feature.

A.1.1 Classification

For our classification system we build upon the video representation that we developed in Chapter 3: the Fisher vector (FV; Sánchez et al., 2013) encoding of improved dense trajectory features [Wang and Schmid, 2013]. We choose to use a vocabulary of size 256, rescale the videos to be at most 320 pixels wide, and skip every second frame when decoding the video.

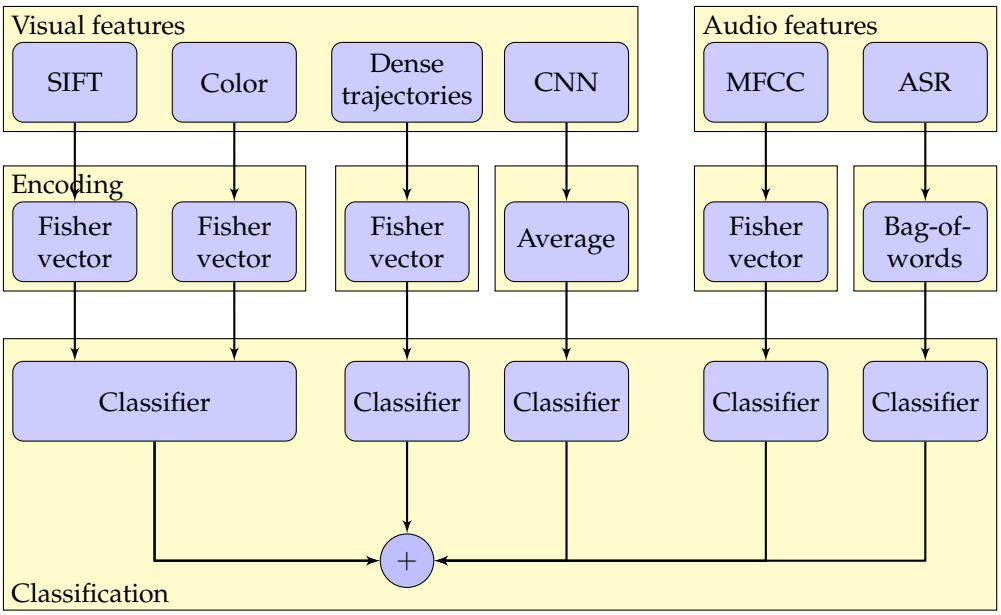


Figure A.1 – The classification system. We illustrate the types of features, which we extract for each video, and how we encode them and combine their classification scores.

Feature extraction. Compared to the experiments in Chapter 3, we complement the motion-based features with several new features (see Figure A.1 for a summary illustration of the classification system). We add static visual appearance information through the following features:

1. **SIFT:** we extract SIFT features [Lowe, 2004] on a dense multi-scale grid, and encode these in a FV using a vocabulary of size 1024. We extract SIFT on one frame out of 60, and aggregate all descriptors in a single FV.

2. **Color:** we extract color features based on local mean and variance of the color channels [Clinchant et al., 2008] every 60-th frame, and encode them in a single FV with a vocabulary size 1024.

3. **CNN:** we extract a 4K dimensional feature using a convolutional network trained on the ImageNet 2010 Challenge data. We use the CAFFE implementation [Jia, 2013], and retain the layer six activations after applying the linear rectification (which clips negative values to zero). We also experimented with using layer seven or eight, but found worse performance. We extract CNN features in every 10-th frame, and average them into a single feature vector.

In addition to the visual features, we also extract features from the audio stream:

1. **MFCC:** we down-sample the original audio track to 16 kHz with 16 bit resolution and then compute Mel-frequency cepstral coefficients (MFCC) with a window size of 25 ms and a step-size of 10 ms, keeping the first 12 coefficients of the final cosine transformation plus the energy of the signal. We enhance the MFCCs with their first and second order derivatives. The MFCC features are then aggregated into a FV with a vocabulary size of 256.
2. **ASR:** for ASR we used state-of-the art speech transcription systems available for 16 languages [Lamel, 2012, Lamel and Gauvain, 2008]. The files were processed by first performing speaker diarization, followed by language identification (LID) and then transcription. The system for identified language was used if the LID confidence score was above 0.7, else an English system as used. The vast majority of documents were in English, with a number in Spanish, German, Russian, French as well as a few in 8 other languages. Therefore, we only used the English transcripts, and represent them using a bag-of-word encoding of 110K words.

Classifier training. To train the action classification models, we train SVM classifiers in a 1-vs-rest approach. We perform early fusion to the dense trajectory features, by concatenating FVs for the MHB, HOG, and HOF channels. Similarly we early fuse the two local image features: SIFT and color. We, then, learn a per-class late-fusion of the SVM classifiers trained on the early fusion channels and the CNN, MFCC, and ASR features.

We also investigated the effect of using different parts of the training data. The *Train* part consists of 13,320 trimmed action clips across the 101 action classes. The *Validation* part consists of 1,010 untrimmed videos across the 101 action classes (10 per class), which are representative for the test videos. Finally, the *Background* part consists of 2,500 untrimmed videos not corresponding to any of the action classes.

A.1.2 Localization

To assess our performance we split the 1010 videos from the *Validation* split into two equal parts; we used one of them as train split and the other one as test.

For the temporal action localization task we only use the dense trajectory features, since the remaining features are more likely to capture

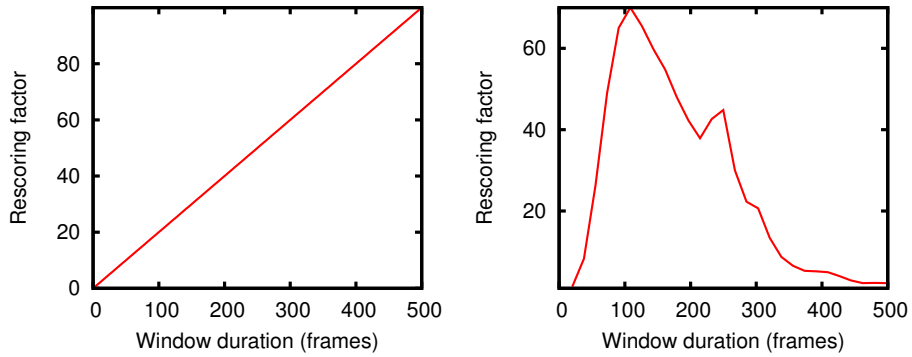


Figure A.2 – Two methods for window rescoring: clip duration (left) and class-specific action duration (right).

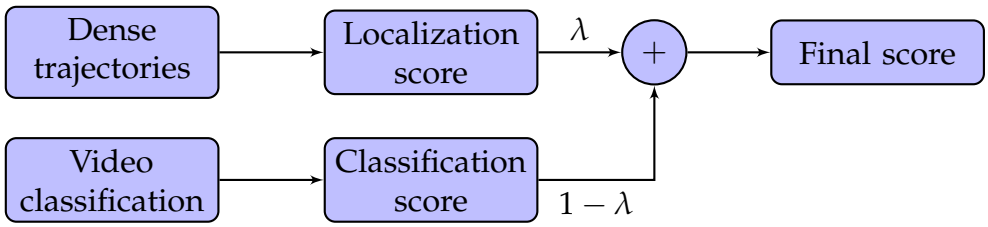


Figure A.3 – We incorporate the classification score into the localization system by taking a weighted average between the classification and localization scores. We obtain the classification score for each video as shown in Figure A.1.

contextual information rather than information that can be used for precise action localization.

We train 1-vs-rest SVM classifiers, albeit using only trimmed action examples from the *Train* and *Validation* sets as positives. As negatives we use (i) all examples from other classes of the *Train* part of the data, (ii) all untrimmed videos in the *Background* part of the data, (iii) all untrimmed videos of other classes in the *Validation* part of the data, and (iv) all trimmed examples of other classes in the *Validation* part of the data. In addition we performed one round of hard-negative mining on the *Validation* set, based on a preliminary version of the detector, and used these as additional negatives.

For testing we use temporal detection windows with a duration of 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, and 150 frames, which we slide with a stride of 10 frames over the video. After scoring the windows,

Feature	mAP
MBH	52.02 ± 2.4
HOF	50.38 ± 1.9
HOG	48.79 ± 2.3
CNN	48.42 ± 2.0
Color	37.36 ± 1.7
SIFT	37.17 ± 1.8
ASR	20.77 ± 1.0
MFCC	18.97 ± 1.5

Table A.1 – Evaluation of individual features for the classification task.

we apply non-maximum suppression to enforce that non of the retained windows are overlapping. We consider a window to be a correct match if its intersection over union with a groundtruth window exceeds 50%.

As in Chapter 3, we re-score the detection windows by multiplying the localization score by the duration of the window (see Figure A.2a). This avoids a bias towards detecting too small video fragments. In addition, we experimented with a class-specific duration prior, estimated from the training data (see Figure A.2b).

Finally, we combine the window's localization score with the video's classification score for the same action class (see Figure A.3). This pulls-in additional contextual information from the complete video that is not available in the temporal window features. We take a weighted average of these scores; the weight is determined using the *Validation* set.

A.2 Results

In this section we present experimental results obtained on the *Validation* set.

A.2.1 Classification results

For the classification task we split the *Validation* set into 30 train/test folds. For each training fold we select 7 samples from each class, with the

Fusion	mAP
Early fusion (EF)	
EF1: MBH + HOF + HOG	64.35 ± 2.3
EF2: SIFT + Color	45.78 ± 2.3
Late fusion (LF)	
LF1: EF1 + EF2	69.62 ± 2.18
LF2: EF1 + EF2 + CNN	71.06 ± 2.00
LF3: EF1 + EF2 + CNN + MFCC	73.65 ± 1.90
LF4: EF1 + EF2 + CNN + ASR	76.26 ± 1.85
LF5: EF1 + EF2 + CNN + MFCC + ASR	77.84 ± 1.70

Table A.2 – Evaluation of combinations of features for the classification task.

test fold containing the remaining 3 samples. We report the mean and the standard deviation of the mAP score across these 30 folds.

Table A.1 presents an evaluation of the individual features. The results show that the visual features are the strongest, in particular the motion features. Table A.2 shows an evaluation of various combinations of features. Combining features significantly improves the results, *e.g.* from 52.02% mAP for MBH, to 64.35% for MBH + HOF + HOG. When combining all features, we obtain 77.84% mAP. Interestingly, the high-level ASR feature brings more than 4% mAP improvement when all other features are already included. Table A.3 shows the top ten classes which benefited most from the inclusion of audio features (MFCC and ASR). We notice that these classes are related to music or musical instruments (*band marching*, *playing dhol*, *blowing candles*), or they contain instructional snippets (*apply lipstick*, *tennis swing*, *blow dry hair*), or they are characterized by commentary (*cricket shot*, *parallel bars*).

Next, we evaluate the effect of using different parts of the training data and test on the held-out part of the validation set, see above description of the cross-validation procedure. The results in Table A.4 clearly show the importance of using both the trimmed (in *Train*) and untrimmed (in *Validation*) examples; untrimmed videos are important since these are representative of the test set, and the trimmed examples are important because they are roughly 10 times more of them. The videos in the *Background* set were not useful, probably because there are enough negative

Class name	Absolute gain from audio features
Band marching	+32.7
Cricket shot	+32.5
Apply lipstick	+31.9
Tennis swing	+31.2
Brushing teeth	+30.7
Blow dry hair	+27.6
Playing dhol	+25.6
Parallel bars	+24.2
Blowing candles	+22.1
Playing sitar	+19.0

Table A.3 – The top ten classes whose performance was improved the most by the use of audio features. The classes are sorted in decreasing order by the mAP difference between visual-audio classifier and visual-only classifier.

Validation	Y		Y	Y		Y
Train		Y	Y		Y	Y
Background				Y	Y	Y
LF5 mAP	70.40 $\pm$ 1.6	68.74 $\pm$ 2.2	77.84 $\pm$ 1.7	67.94 $\pm$ 1.9	67.90 $\pm$ 2.2	77.70 $\pm$ 1.8

Table A.4 – Evaluation of different parts of the training data for the classification task.

samples across the *Train* and *Validation* dataset. In conclusion, we used the *Train* and full *Validation* sets in our submitted classification results.

We have submitted two runs for the classification task. The primary run (Run #1) was the full-fledged system, which includes all the channels, visual and audio; this run corresponds to LF5 in Table A.2. For the secondary run (Run #2) we have used a visual-only system, corresponding to LF3 in Table A.2. As in our validation experiments, including the audio features improves the performance significantly: from 63.6% mAP for Run #2 to 67.2% mAP for Run #1 on the test set.

System	Rescoring	Remarks	mAP
L1	clip duration	K=64, MBH	12.56
L2	clip duration	K=64, MBH+HOF+HOG	14.58
L3	clip duration	K=256, MBH+HOF+HOG	19.17
L3+C, $\lambda = 0.2$	clip duration	Run #3	21.63
L3+C, $\lambda = 0.2$	class specific prior, *Train+Val.*		21.57
L3+C, $\lambda = 0.25$	class specific prior, *Validation*	Run #1	26.57
L3+C*, $\lambda = 0.25$	class specific prior, *Validation*	Run #2, C* visual-only	26.52
L3	class specific prior, *Validation*		24.43

Table A.5 – Evaluation of action localization using the localization (L) and classification (C) system. The combined score is a weighted average which weights the localization score by λ and the classification score by $(1 - \lambda)$.

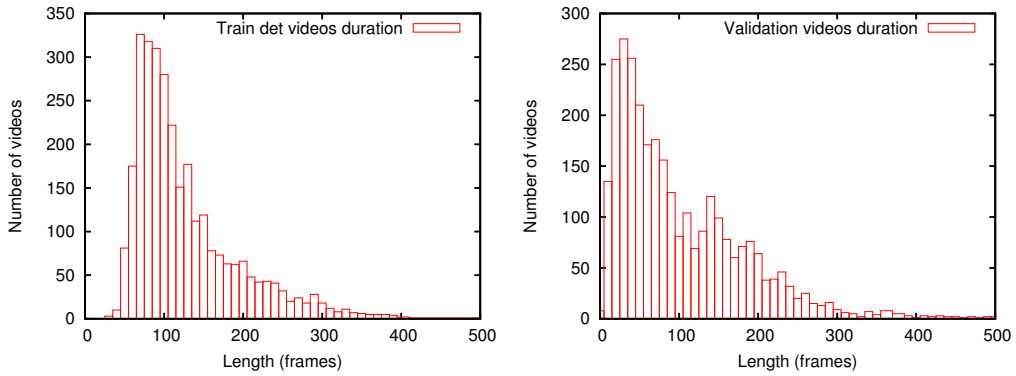


Figure A.4 – Duration histograms of positive action instances across the 20 classes used for localization for the *Train* (left) and *Validation* (right) part of the data.

A.2.2 Localization results

For our localization system we have to compute features and scores for many temporal windows, and this is much more costly than the classification of entire videos. Therefore, we first evaluated the effect of using only MBH or all three trajectory features, and the impact of using a smaller vocabulary of size 64 *vs*. using the one of size 256 used for classification. In these experiments we use the rescored non-maxima suppression technique, which we introduced in Chapter 3. This means we rescore the windows by their duration. The first three rows of Table A.5 show that

the performance drops significantly if we use a smaller vocabulary, or use only MBH features. Therefore, we keep all trajectory features and the vocabulary of size 256 in all remaining experiments.

In the remaining experiments in Table A.5 we consider the benefit of including the classification score as a contextual feature to improve the localization performance. The trade-off between the classification and localization score is determined cross-validation. The classification and localization scores are first normalized to be zero-mean and unit-variance so that the scores are comparable, and the combination weight has a natural interpretation. In the first experiment (row 4) we combine the best detector L3 (with mAP 19.17%) with the classification model using all our channels, which leads to an improved mAP of 21.63%. This is the system submitted as Run #3.

Instead of rescoring with the clip duration, we also considered rescoring with a class-specific prior on the duration (obtained using a histogram estimate). This leads to a similar performance of 21.57% mAP.

We observed a difference in the duration distribution of positive action instances in the *Train* and *Validation* part of the data, see Figure A.4. This difference is explained by different annotation protocols and teams used to annotate these parts of the data. Therefore, we also considered using a prior estimate based on the validation data only. This leads to a significantly improved localization mAP of 26.57%. This is the system we submitted as Run #1.

Finally, submitted Run #2 is similar to Run #1, but is a vision-only run that excludes the MFCC and ASR audio features in the classification model. The system corresponding to the Run #2 obtains a performance of 26.52% mAP on our test split. Interestingly, in this case the audio features do not have a signifiant impact. To verify that the localization still benefits from the classifier when using the stronger prior, we also include a last run that uses this prior without the classification score (last row). This leads to a reduction in performance to 24.43%, showing that global video context is useful in the localization task, even when using the strong prior on duration.

We present in Table A.6 the results from the competition for the three runs we have described above. Surprisingly, our primary run (Run #1) is outperformed by the other two runs at any evaluation overlap, but the performance differences are small.

	Overlap				
	0.1	0.2	0.3	0.4	0.5
Run #1	34.11	32.28	28.18	20.83	13.91
Run #2	34.39	32.64	**28.48**	**20.94**	13.85
Run #3	**36.57**	**33.60**	26.99	20.75	**14.36**

Table A.6 – Final results for the localization task for our three runs as a function of the evaluation overlap (how much a window has to overlap with the groundtruth in order to be considered a match).

Figure A.5 – Snapshots from the top four ranked test videos for the three easiest classes for the classification task.

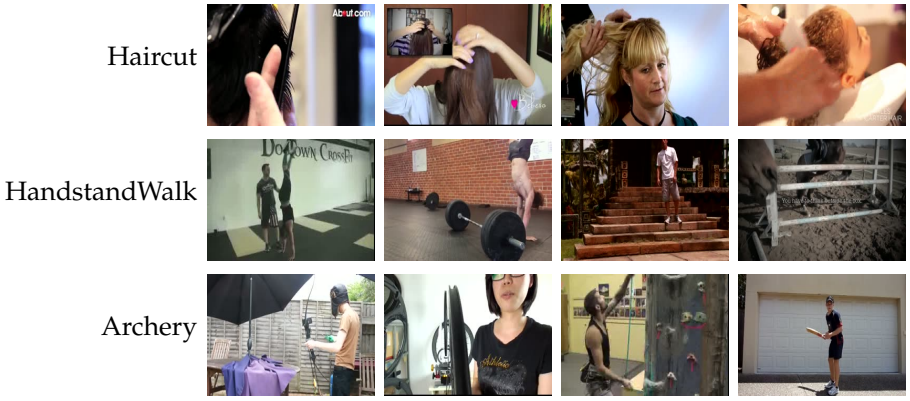


Figure A.6 – Snapshots from the top four ranked test videos for the three hardest classes for the classification task.

A.3 Conclusion

We have described our submission to the THUMOS 2014 Challenge, and presented an experimental evaluation of its components. Our main findings are as follows: (i) Additional visual and audio features significantly improve over a system based on dense trajectory features only (as we used in our winning entry in the 2013 THUMOS Challenge). This improved our results from 64.35% mAP to 77.84% mAP in our evaluation. (ii) For action classification in untrimmed videos it is beneficial to include representative untrimmed training videos in addition to trimmed action examples. This improved our results from 68.74% mAP to 77.84% mAP in our classification experiments. (iii) For action localization in untrimmed videos it is beneficial to use global video features, which we included in the form of the video classification scores. This improved our results from 19.17% mAP to 21.63% mAP in our localization experiments. We conclude by showing examples of our results, figures A.5 and A.6, on the test data for the classification task for the three easiest and hardest classes (according to the results on the *Validation* set).