Structured Models for Action Recognition in Real-word Videos

Adrien Gaidon

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

Adrien GAIDON

Thèse dirigée par **Cordelia SCHMID**

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

Modèles Structurés pour la Reconnaissance d'Actions dans des Vidéos Réalistes

Structured Models for Action Recognition in Real-world Videos

Thèse soutenue publiquement le **25 octobre 2012**,
devant le jury composé de :

Pr. Martial Hebert
Carnegie Mellon University, Pittsburgh, PA, USA, Rapporteur

Pr. Patrick Pérez
Technicolor Research & Innovation, Rennes, France, Rapporteur

Dr. Ivan Laptev
Inria Paris, France, Examinateur

Dr. Zaid Harchaoui
Inria Grenoble, France, Examinateur

Pr. Cordelia Schmid
Inria Grenoble, France, Directeur de thèse

Abstract

This dissertation introduces novel models to recognize broad action categories — like "opening a door" and "running" — in real-world video data such as movies and internet videos. In particular, we investigate how an action can be decomposed, what is its discriminative structure, and how to use this information to accurately represent video content. The main challenge we address lies in how to build models of actions that are simultaneously information-rich — in order to correctly differentiate between different action categories — and robust to the large variations in actors, actions, and videos present in real-world data. We design three robust models capturing both the content of and the relations between action parts. Our approach consists in organizing collections of robust local features into structured action representations, for which we propose efficient kernels. Even if they share the same underlying principles, our methods differ in terms of the type of problem they address and the structural information they rely on. First, we propose to model a simple action as a sequence of meaningful atomic temporal parts, termed "actoms", which are obtained by manual annotation. Our action model represents the temporal structure of actions as a sequence of histograms of actom-anchored visual features. We show how to learn a flexible model of an action's temporal structure in order to automatically localize actions in long unsegmented videos. Second, we extend our ideas to the spatio-temporal structure of more complex activities. We describe a large-scale unsupervised learning algorithm used to hierarchically decompose the motion content of videos. We leverage the resulting tree-structured decompositions to build hierarchical action models, and provide an action kernel between unordered binary trees of arbitrary sizes. Third, we directly compare models of the structure instead of structuring action models. We view short-duration actions as high-dimensional time-series, and propose to model an action's temporal dynamics with its auto-correlation. We design an efficient kernel to compare the temporal dependencies between two actions, and show that it provides useful complementary information to the state-of-the-art unstructured models for action classification.

In all three cases, we conducted thorough experiments on real-world videos from challenging benchmarks used by the action recognition community. We show that our methods outperform the related state of the art, thus highlighting that using structure information allows for more accurate and robust action recognition in real-world videos.

Keywords:

Action Recognition, Video Analysis, Computer Vision, Machine Learning.

Résumé

Cette thèse décrit de nouveaux modèles pour la reconnaissance de catégories d'actions comme "ouvrir une porte" ou "courir" dans des vidéos réalistes telles que les films. Nous nous intéressons tout particulièrement aux propriétés structurelles des actions : comment les décomposer, quelle en est la structure caractéristique et comment utiliser cette information afin de représenter le contenu d'une vidéo. La difficulté principale à laquelle nos modèles s'attellent réside dans la satisfaction simultanée de deux contraintes antagonistes. D'une part, nous devons précisément modéliser les aspects discriminants d'une action afin de pouvoir clairement identifier les différences entre catégories. D'autre part, nos représentations doivent être robustes en conditions réelles, c'est-à-dire dans des vidéos réalistes avec de nombreuses variations visuelles en termes d'acteurs, d'environnements et de points de vue.

Dans cette optique, nous proposons donc trois modèles précis et robustes à la fois, qui capturent les relations entre parties d'actions ainsi que leur contenu. Notre approche se base sur des caractéristiques locales — notamment les points d'intérêts spatio-temporels et le flot optique — et a pour objectif d'organiser l'ensemble des descripteurs locaux décrivant une vidéo. Nous proposons aussi des noyaux permettant de comparer efficacement les représentations structurées que nous introduisons. Bien que nos modèles se basent tous sur les principes mentionnés ci-dessus, ils diffèrent de par le type de problème traité et la structure sur laquelle ils reposent.

Premièrement, nous proposons de modéliser une action par une séquence de parties temporelles atomiques correspondant à une décomposition sémantique. De plus, nous décrivons comment apprendre un modèle flexible de la structure temporelle dans le but de localiser des actions dans des vidéos de longue durée. Deuxièmement, nous étendons nos idées à l'estimation et à la représentation de la structure spatio-temporelle d'activités plus complexes. Nous décrivons un algorithme d'apprentissage non supervisé permettant de dégager automatiquement une décomposition hiérarchique du contenu dynamique d'une vidéo. Nous utilisons la structure arborescente qui en résulte pour modéliser une action de manière hiérarchique. Troisièmement, au lieu de comparer des modèles structurés, nous explorons une autre alternative : directement comparer des modèles de structure. Pour cela, nous représentons des actions de courte durée comme des séries temporelles en haute dimension et étudions comment la dynamique temporelle d'une action peut être utilisée pour améliorer les performances des modèles non structurés formant l'état de l'art en reconnaissance d'actions. Dans ce but, nous proposons un noyau calculant de manière efficace la similarité entre les dépendances temporelles respectives de deux actions.

Nos trois approches et leurs assertions sont à chaque fois validées par des expériences poussées sur des bases de données publiques parmi les plus difficiles en reconnaissance d'actions. Nos résultats sont significativement meilleurs que ceux de l'état de l'art, illustrant ainsi à quel point la structure des actions est importante afin de bâtir des modèles précis et robustes pour la reconnaissance d'actions dans des vidéos réalistes.

Mots-clés :
Reconnaissance d'Actions, Analyse Vidéo, Vision par Ordinateur, Apprentissage Statistique.

Acknowledgments

First of all, I would like to thank my advisors, Professor Cordelia Schmid and Doctor Zaid Harchaoui, for their invaluable guidance. Cordelia's experience, vision, and scientific intuition have been extremely precious throughout all our projects. Furthermore, Cordelia has always been very supportive and patient. Zaid's openness, enthusiasm, and scientific culture have allowed me to consistently progress both technically and scientifically during all these years. Both of them allowed me to push past my boundaries and develop my own ideas. I would also like to thank Doctor Marcin Marszałek, who is a co-author on my first paper, and the one who convinced me to pursue a PhD both in the LEAR team and on action recognition. Special thanks go also to Professor Andrew Zisserman for kindly inviting me to the University of Oxford. Finally, many thanks go to my jury members — Professor Martial Hebert, Professor Patrick Pérez, and Doctor Ivan Laptev — for agreeing to evaluate my work and asking me many insightful questions throughout our interactions. I am also grateful to Microsoft and the joint Microsoft Research - Inria center in Paris, which partially funded my research. I would like to thank Jean Ponce, Andrew Blake, Josef Sivic, and Ivan Laptev for all of our interesting encounters. Ivan, in particular, has been like a scientific mentor whose remarkable work motivated and inspired me. I would also like to thank Anne Pasteur and Martine Thirion for being both very helpful and friendly. Although this is unconventional, I would like to express my gratitude towards the Open Source community for the great scientific ecosystem around Python (the best programming language, ever, period), which allowed me to address many technological challenges in the best conditions. My colleagues at LEAR were one of the reasons why Grenoble was such a great working environment. I would like to thank especially Jakob Verbeek, Matthijs Douze, Hervé Jégou, Rémi Ronfard, and Roger Mohr. My work also allowed me to make a lot of amazing friends that supported me in more ways than one: Gokberk Cinbis, Matthieu Guillaumin, Hedi Harzallah, Alexander Kläser (incredible jazz guitar player and inspiration for my rock band's name), Josip Krapac (The Kläsers' very own Dr. Matt Destruction), Diane Larlus, Jörg Liebelt, Thomas Mensink, Alessandro Prest (The Kläsers' very own Nicko McBrain), Arnau Ramisa, Gaurav Sharma, Heng Wang, and Oksana Yakhnenko. I also cannot express how grateful I am for all the support I continuously received from all of my friends (they will recognize themselves), my parents Muriel and Didier, my baby brother Clément, and the rest of my family. Finally, the biggest thanks go to my wife, Chacha, for all her support, help, and understanding. The love and joy she brings me every day makes life taste better, and anything seems possible with her beside me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

Does a horse lift all four feet completely off the ground during the gallop? The answer to this question is often considered as the first step towards the development of video. In 1872, the photographer Eadweard Muybridge was commissioned by Leland Stanford to give a definitive answer to this long debated question. At this pace — around 58 kilometers per hour, or 36 miles per hour — a horse's hooves move too fast for the human eye to break down the action: a scientific proof using photographs was required. After several preliminary experiments, Muybridge brought an affirmative answer to the question through his famous study called "Sallie Gardner at a Gallop" (*cf.* Figure 1.1). The main challenge was to create a set-up able to take a sequence of pictures capturing the details of fast motions. Muybridge managed to take a series of photos on June 19, 1878 at Stanford's Palo Alto stock farm by placing 24 cameras in a line along the track. The shutter of each camera was triggered by a trip wire as the horse passed.

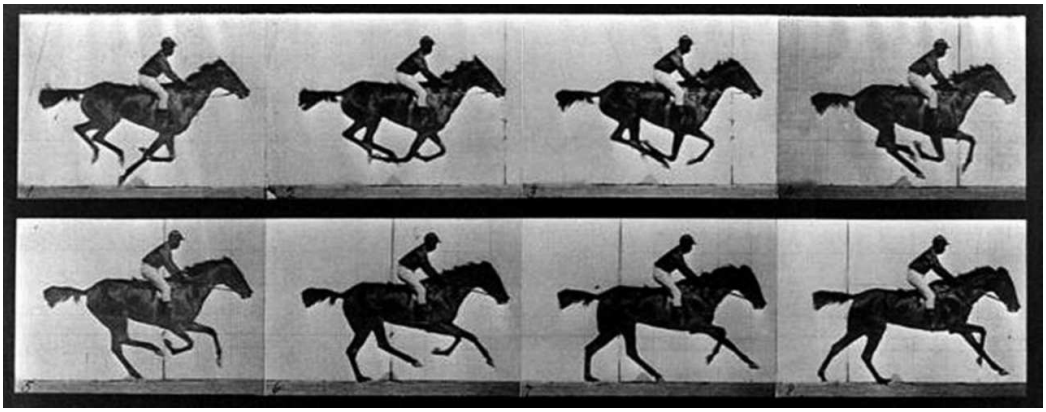


Figure 1.1: *Muybridge's "Sallie Gardner at a Gallop" study. It proved that, at one point in time, all four hooves of a galloping horse are off the ground.*

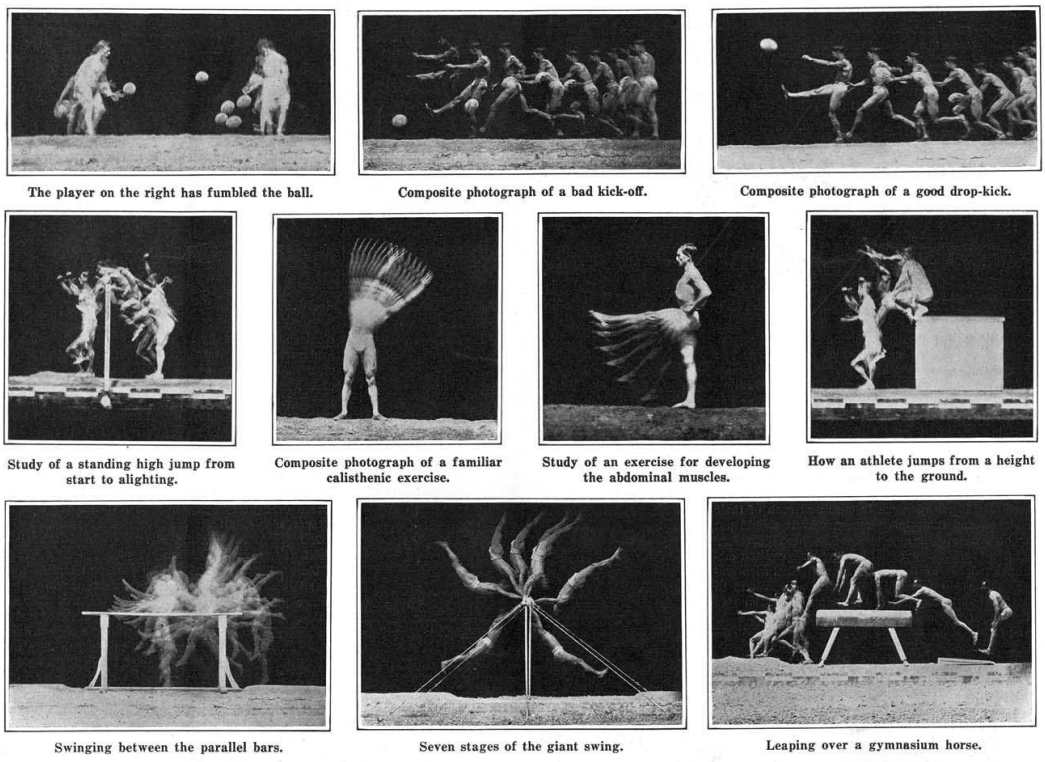


Figure 1.2: *"The Human Body in Action", Etienne-Jules Marey, 1914.*

This work was a prelude to the development of visual motion analysis, which consists in scientifically and precisely investigating the visual properties of dynamical systems, *e.g.*, fluids and humans. Muybridge's work showed that the technological evolution of photography allowed to take sequences of pictures to display a real-world dynamic scene. Image sequences — such as Etienne-Jules Marey's chronographs in Figure 1.2 — allow the scientist to analyse complex motions, *e.g.*, actions, by looking at the temporal evolution of the participating actors and objects.

The development of video analysis is inseparable from the improvement of recording and projection devices. A plethora of apparatus showing movement through sequences of drawings have existed for a long time. They rely on the optical illusion — discovered in 1912 by Max Wertheimer — called "Beta movement", whereby a succession of still images are combined by the brain to form the perceptual impression of motion. The Zoetrope — meaning "wheel of life" in Greek — is an example of a popular device producing the illusion of motion from the rapid succession of still pictures fixed on the interior of a spinning cylinder with vertical slits acting as a shutter.

Early video projectors combined the same ideas with the improvements of photography — credited to Daguerre and Niépce (1839). For instance,

Muybridge invented the first moving picture projector — the Zoopraxiscope (1879) — in order to showcase his stop-action photographies. It relied on rapidly projecting a succession of images from rotating glass disks. Shortly after, Auguste and Louis Lumière patented the Cinématographe — a film camera, projector, and developer — in 1895 and started producing some of the first movies ever made.

Since then, video has become one of the most popular visual media for communication and entertainment. In 2010, 441 billion videos were accessible online, which corresponds to approximately 10^{21} pixels[1]. In 2011, the Youtube website had more than one trillion views and over 800 million unique visitors watched over 3 billion hours of videos per month. Content creation is also booming as one hour of new video content is uploaded to Youtube every second[2]. By the end of 2012, video will account for over 50% of internet traffic, with a 48% compound annual growth rate[3].

Video spans a wide range of sources and applications: video surveillance, industrial control, robotics (*e.g.*, autonomous vehicles), building automation and domotic, human computer interaction systems, gaming (*e.g.*, Microsoft's Kinect device for the Xbox gaming console), internet content platforms (*e.g.*, Youtube and DailyMotion), TV shows, and movies. The common goal of these applications is to convey or capture some visual information about a real-world scene over some period of time.

The main challenge currently faced by users of these technologies is how to find the meaningful content they are looking for. Due to the overwhelming quantity of videos, answering this question calls for tools that can automatically analyze, index, and organize video content.

Our work focuses on recognizing *actions*, one of the important visual information present in video data. Computer vision has recently made significant progress in action recognition — especially for specific tasks such as video-surveillance. Nevertheless, the current state-of-the-art methods are still far from being able to understand every possible action in every type of video. This can be measured by the low performance of the systems involved in the Multimedia Event Detection (MED) task of TRECVID [Smeaton et al. 2006] — a challenging automatic video interpretation competition. For the 2011 edition [Over et al. 2011], the submitted systems had an average of 6% false alarms with 54% missed detections on actions like "making a sandwich" and "grooming an animal".

[1] http://www.comscore.com
[2] http://www.youtube.com/t/press_statistics
[3] http://www.cisco.com

Figure 1.3: *Examples of actions and videos from the "Hollywood 2" dataset [Marszalek et al. 2009] used in our experiments.*

1.1 Goals

The goal of our work is to recognize generic actions performed in real-world videos (*cf.* Figure 1.3 for an illustration). We aim at designing robust models that can represent a broad range of action categories, while accounting for the visual variability of uncontrolled video conditions.

Action recognition under uncontrolled conditions faces numerous difficulties inherent to the complexity of both actions and videos. Several challenges are related to **intra-class variability**: actions in the same category may significantly differ in terms of both appearance and motion content. The sources of variation include noise, lighting conditions, scale and shape of the participating entities, background clutter, wide range of viewpoints, occlusions, fast, complex, and articulated movements, motion blur, camera motion, variability in duration, and execution style. These variations make it difficult for action models to rely on simplifying assumptions such as the full body of the actor being always visible. We account for this variability by using robust local features and by capturing essential structural properties of an action category. Another set of challenges results from **between-class confusion**: there are often subtle distinctions between actions of different classes. For instance,

the running and walking actions share similar motions and discriminating between these categories calls for careful motion analysis. Accurately modeling the specificities of an action is especially important in uncontrolled conditions, where a recognition algorithm has to distinguish between the actions of interest and all other possible events — some observed during learning, some not. Again, we tackle this issue by using structure information to build and compare more detailed action models.

Action recognition encompasses two main tasks: classification and localization. On the one hand, classification aims at assigning a semantic label to an entire video focused on an action. On the other hand, the purpose of localization is to provide both a semantic label and its spatio-temporal extent in videos that may contain more than just one action. In practice, these two problems are closely related, as localization methods often consist in applying a classifier at multiple candidate locations to determine the most likely ones. In both cases, we perform *category-level* recognition, and study *supervised learning* algorithms to tackle these problems: given a list of action categories, we learn a model using a set of video examples for each category.

As stated previously, we assume that an action is characterized by spatio-temporal relationships between sub-events. We claim that capturing this structural information — and in particular the temporal structures — is especially important for action recognition in order to achieve robustness under uncontrolled video conditions and to disambiguate between similar categories. Our main goal is to learn better action models by discovering and using this structure. The major difficulty lies in how to identify, represent, and integrate in a learning framework the relevant structural aspects of actions. We explore different possibilities: sequences of atomic actions, hierarchical motion decompositions, and temporal dependencies between frames.

1.2 Context

Automatic video understanding is a fundamental goal of computer vision. As mentioned previously, actions convey important information needed to understand a dynamic visual scene such as the one depicted in Figure 1.4. Even though the actions of some characters may be inferred from this single image, it is more difficult to determine what the person in the middle is doing. The nature of the objects and the pose of the actors are difficult to understand and — by themselves — not sufficient to determine what actions are being performed: using motion information over a range of frames is necessary. Furthermore, analysing the relations between these movements is necessary to recognize this scene as a Kung-Fu fight.

Figure 1.4: *Frame from Jackie Chan's movie "The Legend of Drunken Master". This frame alone is not enough to understand the content of the scene. Only after looking at the temporal evolution of the different actors can we find out that this is a fighting scene where the character in the center of the frame is being attacked while drinking wine from a bottle.*

For computers to be capable of such a high-level interpretation, they need models to represent the various inter-related motions caused by the temporal evolution of the participating entities, *e.g.*, objects and body parts, over the course of an action. For instance, the Motion History Images (MHI) of Bobick and Davis [2001] is an early and influential model used to represent an action as the temporal evolution of a silhouette (*cf.* Figure 1.5). This approach and other closely related ones — *e.g.*, the spatio-temporal shapes of Gorelick et al. [2007] or the Motion History Volumes (MHV) of Weinland et al. [2006] — are, however, not robust and limited to constrained scenarii, such as video-surveillance [Hu et al. 2004], or optical human-computer interfaces [Shotton et al. 2011]. They, indeed, rely on information that is difficult to obtain under uncontrolled video conditions, *e.g.*, background subtraction, silhouette extraction, or parametric models of the human body.

Recently, the computer vision community has investigated more challenging videos. Uncontrolled video data is characterized by a great variability and the lack of available prior knowledge applicable to (i) the recording process (*e.g.*, the viewpoint, the camera motion, the video quality, and the resolution), (ii) the scene (*e.g.*, indoors or outdoors, crowded or not, moving and cluttered backgrounds, occlusions, lighting conditions), and (iii) the actions performed (*e.g.*, fast or slow motions, interactions). It spans a large spectrum of videos coming from a wide range of potential sources including amateur videos, news reports, sports broadcasts, TV shows, and movies. There is also an expanding variety of action categories present in the related action recognition benchmarks: simple actions (*e.g.*, running), more complex ones (*e.g.*, getting out of a car), interactions (*e.g.*, kissing), and activities (*e.g.*, eating).

Figure 1.5: *The motion energy images (MEI) and motion history images (MHI) of Bobick and Davis [2001].*

In order to address the aforementioned issues, several researchers [Chomat and Crowley 1999, Dollár et al. 2005, Laptev 2005, Schüldt et al. 2004, Zelnik-Manor and Irani 2001] proposed to use local features. These are in general spatio-temporal extensions of the 2D detectors and descriptors successfully used in images for object recognition (see for instance the results of the PASCAL VOC challenges [Everingham et al. 2010]). Local video features such as spatio-temporal interest points [Laptev 2005] describe the appearance (*e.g.*, with histograms of oriented gradients [Dalal and Triggs 2005]) or motion (*e.g.*, with histograms of optical flow [Laptev et al. 2008]) of local spatio-temporal volumes. Thanks to the robustness of local descriptors and their lack of global assumptions (*e.g.*, about geometrical relations), local features have been successfully applied to challenging action recognition tasks in real-world videos.

An action is then described by the collection of local features found in the corresponding video. The bag-of-features (*cf.* Section 2.2.3) is a popular model used to represent the set of local spatio-temporal descriptors. It consists in quantizing these local features by assigning each one of them to the closest "visual word" — a prototypical feature — and then building the histogram counting the occurrences of each visual word in the video (*cf.* Figure 1.6 for an illustration). As this orderless model ignores the relations between local features, several extensions have been proposed to incorporate some structure information — *e.g.*, with spatio-temporal pyramids [Laptev et al. 2008].

Action models based on local features are often high-dimensional (typically on the order of 10^4 dimensions). Furthermore, representations like the bag-of-features are data-driven and not easily interpretable by humans. Therefore, recognition approaches based on local features cannot be implemented in practice using manually defined rules. The machine learning field provides principled tools that allow to overcome these difficulties. Thanks to the large amount of available video data, supervised machine learning algorithms can automatically learn complex action models from annotated training examples.

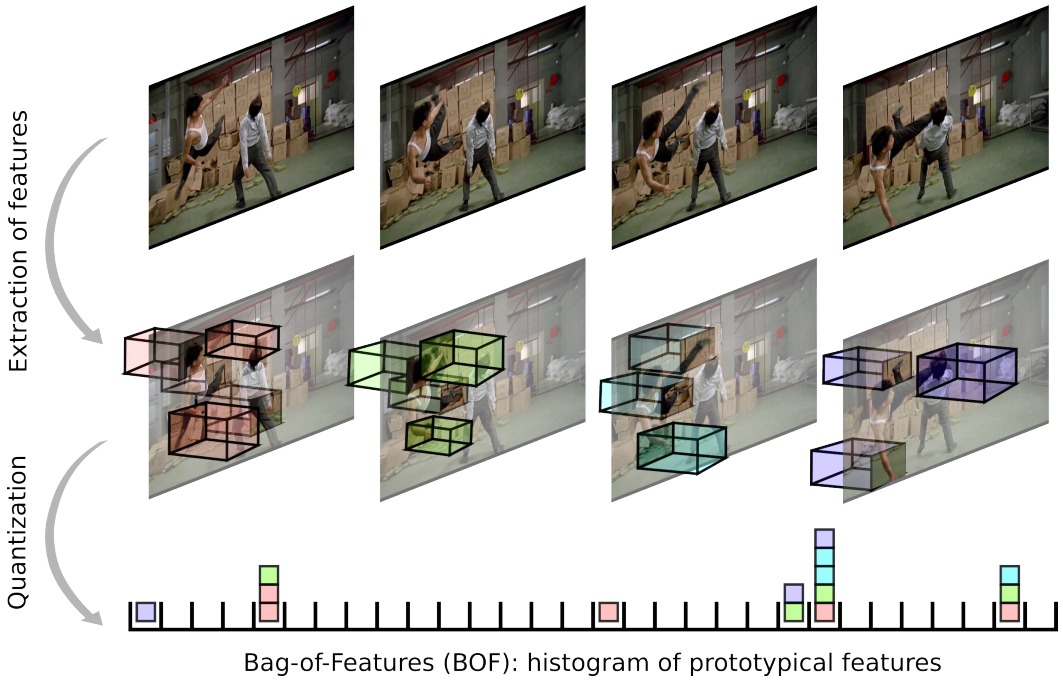


Figure 1.6: *Bag-of-features: a robust model that aggregates local spatio-temporal features by quantizing them on a vocabulary of prototypical features called "visual words" by analogy with the "bag of words" text model.*

Annotations — *i.e.*, semantic labels such as "this video contains someone falling down" — can be acquired by different means, but are often obtained manually. The power of machine learning techniques like Support Vector Machines [Schölkopf and Smola 2002] allows to generalise to any video the knowledge acquired from a few training ones.

Successful action recognition applications using machine learning algorithms have already emerged — see for instance the development of automatic video-surveillance or Microsoft's Kinect sensor for the Xbox gaming console [Shotton et al. 2011]. Nevertheless, at the time of writing (2012) and to the best of our knowledge, a mainstream application of action recognition "in the wild" (*e.g.*, on Youtube or with autonomous robots) does not exist yet. This highlights the large amount of research that is still required in order to understand the fundamental properties of actions and how to represent them under real-world video conditions.

In this context, we believe it is necessary to build on both the early structured action models and on the recent orderless representations in order to progress towards better action models. Our goal is to find the good level of details that will allow action models to be simultaneously more precise than bag-of-features and more robust than silhouette-based representations.

1.3 Contributions

Our work relies on local features and aims at structuring their aggregation to learn better high-level models of actions. We introduce both sequential representations for simple actions and hierarchical representations for more complex activities. We propose techniques to decompose — on frames, temporal parts, and mid-level motion components — the global spatio-temporal structure of actions. We exploit these structured representations by designing adapted kernels (similarity measures), which are then integrated in non-linear classifiers. We provide experimental evidence validating our solutions concerning the following questions:

- what are good *units* to decompose actions on;
- what is the interesting structure of actions;
- how to extract and represent structure in a robust manner;
- how to learn and compare structured action models.

Our contributions are described in the following paragraphs.

- To localize simple actions, *e.g.*, opening a door, in long unsegmented videos, we introduce a temporally structured extension of bag-of-features: the Actom Sequence Model (ASM). We describe an action as a sequence of atomic action units, termed *actoms*, which are semantically meaningful temporal parts that are characteristic for the action. ASM represents the temporal structure of actions as a sequence of histograms of actom-anchored visual features (*cf.* Figure 1.7). Training requires the annotation of actoms for action examples. At test time, actoms are detected automatically based on a non-parametric model, which acts as a prior on an action's temporal structure. We propose a sliding central frame detection approach: every N frames, we evaluate the probability of an action being centered at a particular frame t by marginalizing over our temporal prior:

$$\mathbf{P}(\text{action at } t) = \sum_{j=1}^{s} f_{\text{ASM}}(t, \hat{\mathbf{\Delta}}_\mathbf{j})\ \mathbf{P}(\hat{\mathbf{\Delta}}_\mathbf{j}) \tag{1.1}$$

  where $\hat{\mathbf{\Delta}}_\mathbf{j}$ is the $j-th$ candidate sequence of actoms in our learned temporal model, $\mathbf{P}(\hat{\mathbf{\Delta}}_\mathbf{j})$ is its estimated prior probability, and $f_{\text{ASM}}(t, \hat{\mathbf{\Delta}}_\mathbf{j})$ is the posterior probability returned by our ASM classifier. We present experimental results on the challenging Coffee and Cigarettes [Laptev and

Pérez 2007] and DLSBP [Duchenne et al. 2009] datasets. We also adapt our approach to a classification by detection set-up, and demonstrate its applicability on the Hollywood 2 dataset [Marszalek et al. 2009]. We show that our ASM method outperforms the current state of the art in temporal action detection, as well as baselines that detect actions with a sliding window method combined with a bag-of-features. This work was published in [Gaidon et al. 2011a] and is presented in Chapter 3.

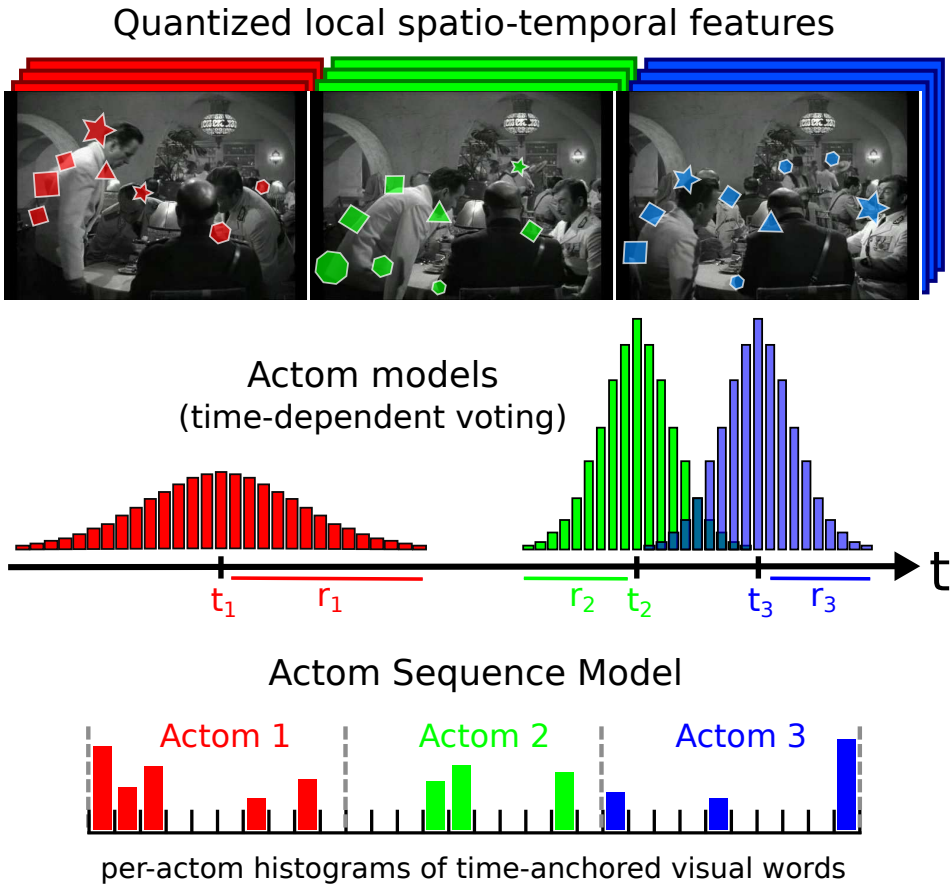


Figure 1.7: *Illustration on three actoms for the "sitting down" action of our Actom Sequence Model (ASM): succession of temporal parts modeled by the aggregation of time-anchored local features.*

- Although simple actions can be represented as sequences of short temporal parts, longer activities, *e.g.*, pole vaulting, are composed of a variable number of sub-events connected by complex spatio-temporal relations. In Chapter 4, we learn how to automatically represent activities as a hierarchy of mid-level motion components. This hierarchy is a data-driven decomposition that is specific to each video. We introduce a spectral divisive clustering algorithm to efficiently extract a hierarchy over a large

number of tracklets (*i.e.*, local trajectories, *cf.* Figure 1.8). We use this structure to represent a video as an unordered binary tree. This tree is modeled by nested histograms of local motion features. We provide an efficient positive definite kernel that computes the structural and visual similarity of two hierarchical decompositions by relying on models of their parent-child relations:

$$k(\mathcal{T}_1, \mathcal{T}_2) = \frac{1}{|\mathcal{E}_1||\mathcal{E}_2|} \cdot \sum_{\substack{e_1 \in \mathcal{E}_1 \\ e_2 \in \mathcal{E}_2}} h(e_1, e_2) \tag{1.2}$$

where $\mathcal{T}_i$ is our tree model with edges $\mathcal{E}_i$ and $h(e_1, e_2)$ denotes the similarity between the histogram models of two edges e_1 and e_2. We present experimental results on three recent challenging benchmarks: the complex activities from the Olympics Sports dataset [Niebles et al. 2010], the human-human interactions of the High Five dataset [Patron-Perez et al. 2010], and the large set of simpler actions from the HMDB dataset [Kuehne et al. 2011]. We show that per-video hierarchies provide additional information for activity recognition. Our approach improves over unstructured activity models, baselines using other motion decomposition algorithms, and the state of the art. This work was published in [Gaidon et al. 2012] and is presented in Chapter 4.

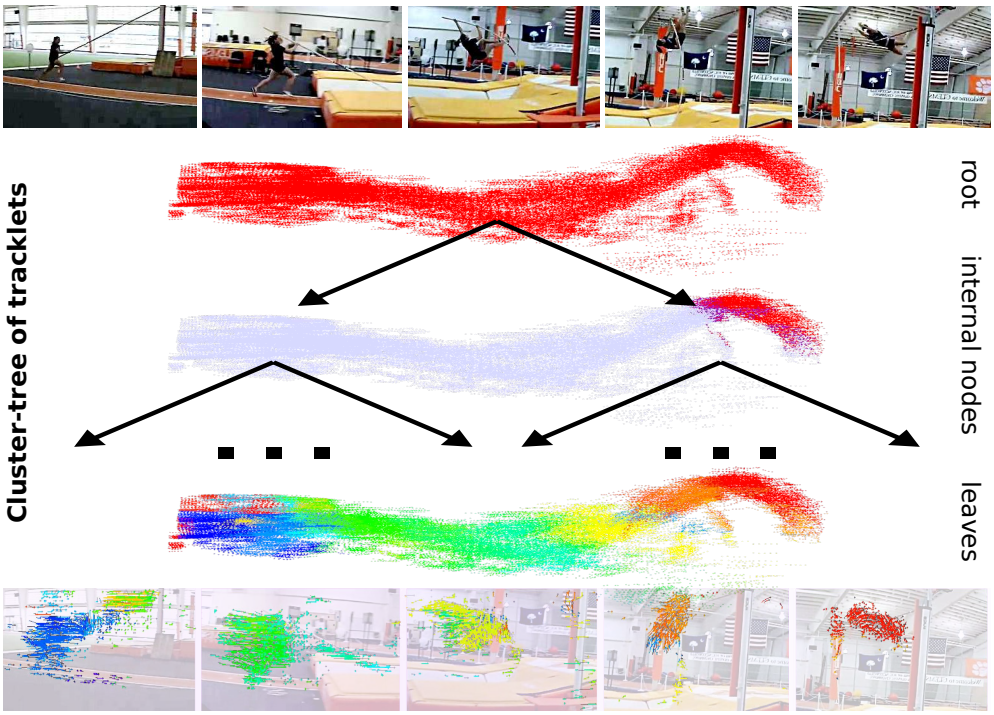


Figure 1.8: *Illustration on a pole vaulting activity of our cluster-tree of tracklets: automatically learned hierarchical motion decomposition for complex activity recognition.*

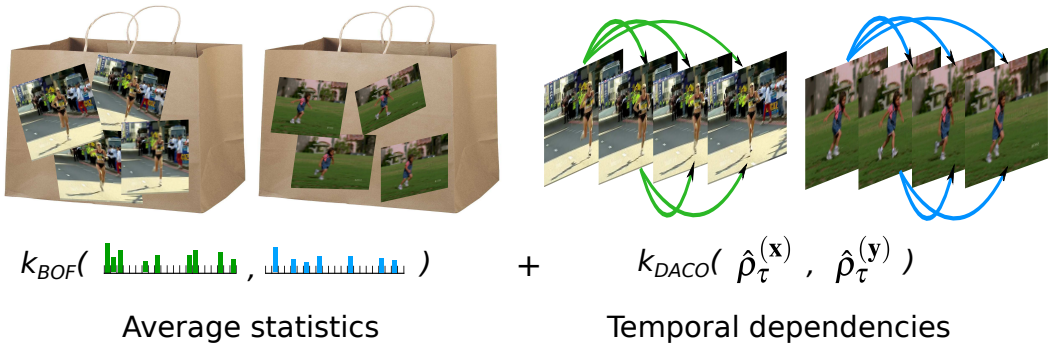


Figure 1.9: *Combination of the orderless bag-of-features (left) with temporal dependency information (right) represented through auto-correlation operators in the DACO kernel.*

- Finally, we propose to complement representations based on average statistics — *e.g.*, BOF or the two aforementioned approaches — with a model of the temporal dependencies between frames: the auto-correlation operator. We address the problem of action recognition by describing actions as time series of per-frame models and introduce a kernel to compare their dynamical aspects (*cf.* Figure 1.9). The two previous contributions focus on the global structure of actions to organize sets of local features into high-level content representations reflecting this structure. This part of our work investigates a different way to use structure information: instead of comparing *structured video content*, we compare *the structure of video contents*. Our main contributions are the following. First, we provide a principled kernel to compare the dynamics and temporal structure of actions by computing the Distance between their respective Auto-Correlation Operators (DACO):

$$k_{\mathrm{DACO}}(\{\mathbf{x_t}\}_{t=1:T}\ ,\ \{\mathbf{y_{t'}}\}_{t'=1:T'})^2 = \exp\left(-\frac{1}{2\sigma^2}\left\|\hat{\rho}_\tau^{(\mathbf{y})} - \hat{\rho}_\tau^{(\mathbf{x})}\right\|_{HS}^2\right) \quad (1.3)$$

  where $\hat{\rho}_\tau^{(\mathbf{x})}$ and $\hat{\rho}_\tau^{(\mathbf{y})}$ denote the auto-correlation operators of the time series $\{\mathbf{x_t}\}_{t=1:T}$ and $\{\mathbf{y_{t'}}\}_{t'=1:T'}$ respectively. Second, we derive a practical formulation to compute DACO in any feature space deriving from a base kernel between frames. Third, we report experimental results on recent action recognition datasets showing that DACO provides useful complementary information to the average distribution of frames, as used in state-of-the-art models based on the bag-of-features representation. This work was published in [Gaidon et al. 2011b] and is presented in Chapter 5.

Chapter 2

Related Work

Contents

In this chapter, we first provide a broad overview of early structured action models based on *global representations* (Section 2.1), which include sequences of frame descriptors and models of the spatio-temporal volume spanned by an action. We then describe methods based on *local features* (Section 2.2), which model actions as collections of spatio-temporal patches or local trajectories represented with robust appearance and motion descriptors. We finally review the existing *structured models* (Section 2.3) that relate most closely to ours. These approaches aim at organizing collections of local descriptors, for instance by modeling their co-occurrence relations or by automatically grouping them into mid-level parts.

2.1 Global representations

Global models represent an action using appearance and motion descriptors either of the whole body structure or of a region of interest tightly enclosing an actor. These representations typically depend on silhouette extraction or grid structured representations of the region spanned by the body over the execution of an action. In contrast to early gait analysis methods using parametric models of the human body (*e.g.*, the locations and angles of body joints, *cf.* [Moeslund et al. 2006] for a review), global models are more widely applicable, as they do not rely on the identification and tracking of individual body parts. We first describe action models based on sequences of global representations. We then review volumetric models that describe the shape or motion content of the video volume spanned by an action in space-time.

2.1.1 Sequential approaches

Inspired by techniques successfully applied on speech recognition problems, several works [Brand et al. 1997, Starner and Pentland 1995, Wilson and Bobick 1995, Yamato et al. 1992] represent actions as sequences of states modeled by latent variables. They rely on dynamic probabilistic graphical models — such as Hidden Markov Models (HMM) [Rabiner and Schafer 2007] — to learn temporal transitions between these hidden states. An action is represented by a generative model over sequences of per-frame feature vectors, and recognition is performed based on the likelihood of an image sequence with respect to this model. For instance, Yamato et al. [1992] recognize tennis actions using HMMs on grid representations of silhouettes.

Since then, several approaches [Chen and Agarwal 2011, Hoai et al. 2011, Kulkarni et al. 2010, Laxton et al. 2007, Lv and Nevatia 2007, Oliver et al. 2000, Shi et al. 2011, Zeng and Ji 2010] attempt to recognize actions in more challenging settings. In general, they use a Viterbi-like inference algorithm to temporally segment videos. For instance, Chen and Agarwal [2011] adapt speech recognition tools to model actions with a space-time-frequency representation of scores of boosted spatio-temporal interest point detectors. Their approach requires the localization and continuous tracking of the entire human body (not individual body parts) in order to compute a figure-centric descriptor. Dealing with long-term interactions with objects, Laxton et al. [2007] use Dynamic Bayesian Networks (DBN) with manually designed per-activity hierarchies of predefined contextual cues and object detectors. Some other sequential approaches [Hoai et al. 2011, Shi et al. 2011] are related to HMMs, but instead learn a discriminative model from manually segmented training videos in order to perform action segmentation.

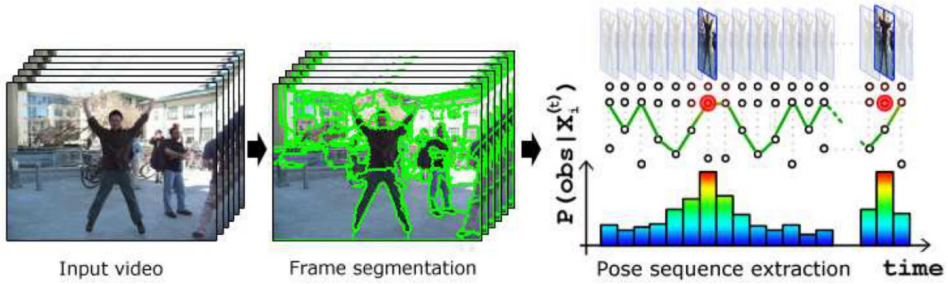


Figure 2.1: *Actions as time series of poses from [Brendel and Todorovic 2010].*

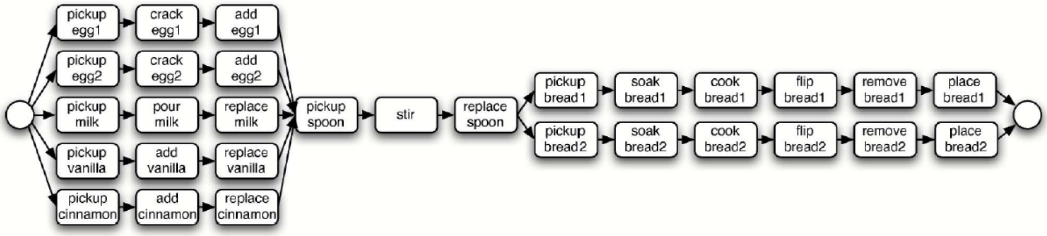


Figure 2.2: *A hand-crafted Dynamic Bayesian Network from [Laxton et al. 2007] modeling the "cooking French toast" activity.*

Also operating on sequential representations, exemplar-based techniques [Darrell and Pentland 1993, Gavrila and Davis 1995, Niyogi and Adelson 1994, Veeraraghavan et al. 2006] directly compute an alignment score between an action and some template sequences. In general, these methods require less training data and provide more flexibility, as they can handle non-linear speed variations with the Dynamic Time Warping (DTW) algorithm [Sakoe and Chiba 1978]. For instance, Darrell and Pentland [1993] use DTW on a sequence of scores obtained by correlating a video with a set of per-view templates. More recently, Brendel and Todorovic [2010] recognize actions as time series of a few snapshots of body parts obtained by video segmentation (*cf.* Figure 2.1). They use a template-based classification approach by aligning a video with training examples using DTW.

Despite their efficiency on constrained video data and their encoding of temporal aspects, sequential methods suffer from some limitations. For instance, they cannot represent actions involving concurrent sub-events, *e.g.*, human interactions. Some extensions of HMMs, such as coupled HMMs [Oliver et al. 2000] or more generic Dynamic Bayesian Networks [Laxton et al. 2007], address this issue. Their complex domain-specific structure, however, needs to be manually specified by experts (see for instance Figure 2.2). In addition, spatio-temporal action localization requires higher-level probabilistic models, *e.g.*, by combining multiple HMMs [Brand and Kettnaker 2000, Lv and Nevatia 2007]. These models need a large amount of training examples in order to account for all events that might occur, including non-actions.

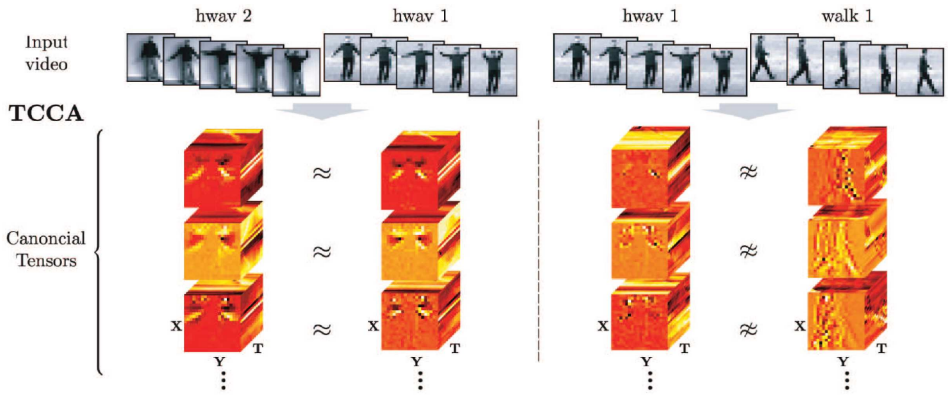


Figure 2.3: *Tensor CCA on video volumes [Kim and Cipolla 2009].*

2.1.2 Volumetric approaches

In contrast to sequential recognition techniques, volumetric methods view actions as 3D (X-Y-T) objects in a spatio-temporal video volume, thus treating space and time in a unified manner. They extract global action models that capture structure information in the form of spatio-temporal shapes. Volumetric approaches mostly rely on template matching for recognition and are successful under restricted video conditions [Gorelick et al. 2007, Schüldt et al. 2004], such as simple static backgrounds, actors with full body always visible, and restricted viewpoint variations and camera motions.

Some models operate directly on the videos themselves. For instance, Kim and Cipolla [2009] directly compare video volumes using Tensor Canonical Correlation Analysis (*cf.* Figure 2.3). Alternatively, several approaches [Blank et al. 2005, Bobick and Davis 2001, Gorelick et al. 2007, Rodriguez et al. 2008] rely on silhouettes in order to obtain spatio-temporal templates. For instance, Bobick and Davis [2001] introduce motion history images (MHI), which are temporal templates representing the evolution of motion over time. Blank et al. [2005] and Gorelick et al. [2007] use background subtraction techniques to model actions as space-time shapes. Rodriguez et al. [2008] learn spatio-temporal cuboid templates and perform template matching via correlation using the generalized Fourier transform. Silhouettes provide useful information for action recognition, but their use is limited to simple or controlled video conditions. They are, indeed, difficult to obtain in the presence of complex dynamic backgrounds, and do not account for self-occlusions.

Other volumetric approaches [Efros et al. 2003, Ke et al. 2010, Polana and Nelson 1994, Schindler and Van Gool 2008, Shechtman and Irani 2005] focus on optical flow to obtain global action templates. For instance, Polana and Nelson [1994] represent periodic actions with spatio-temporal grids of optical flow. Shechtman and Irani [2005] use the correlation between motion flows.

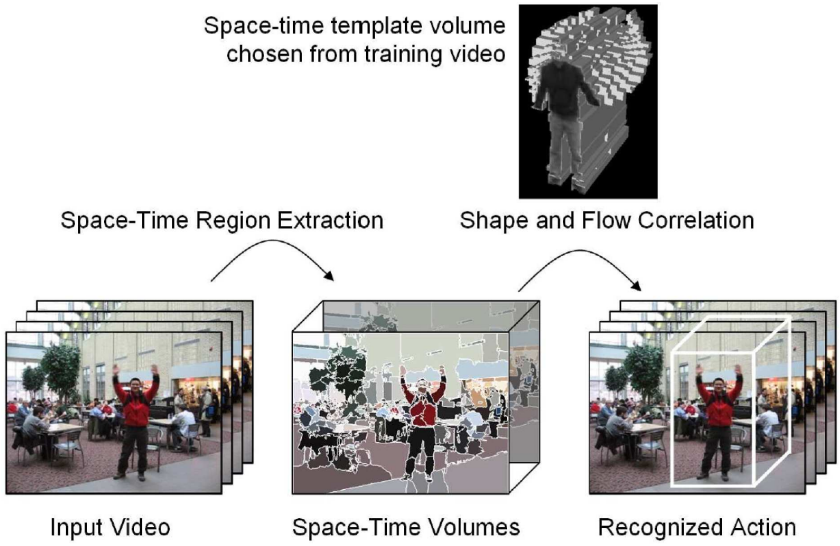


Figure 2.4: *Action localization with space-time templates from [Ke et al. 2010].*

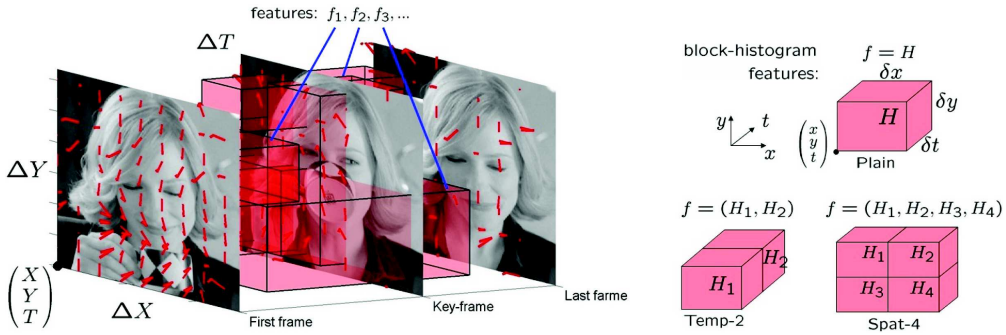


Figure 2.5: *Spatio-temporal descriptors of appearance (Histogram of Oriented Gradients) and motion (Histogram of Optical Flow) with different spatial and temporal subdivisions from [Laptev and Pérez 2007].*

Efros et al. [2003] compute blurred optical flow features inside tracks of soccer players, and compare actions to templates by frame-wise alignment. Combining appearance and flow, Schindler and Van Gool [2008] model actions as the concatenation of per-frame shape and motion features. More recently, Ke et al. [2010] over-segment videos and use optical flow and volumetric features to match space-time shapes representing action templates (see Figure 2.4).

Finally, several researchers have investigated the combination of both appearance and motion information to represent the content of a video volume. For instance, Laptev and Pérez [2007] propose a boosted action classifier over multiple motion and shape features (see Figure 2.5). Relying on spatio-temporal gradients and a generic human detector, Kläser et al. [2010b] introduce the HOG-Track descriptor to represent a segment of a human track.

A spatio-temporal volume spanned by a human is discretized with a spatio-temporal grid, and each cell of this grid is represented using a HOG3D descriptor [Klaser et al. 2008]. These methods have shown promising results on challenging video data such as movies.

Volumetric approaches rely on a similarity measure between video volumes, and typically localize actions by matching sub-volumes with a set of candidate templates. For instance, Ke et al. [2010] use a sliding-window approach with part-based matching relying on pictorial structures, and Kläser et al. [2010b] localize actions using a sliding window approach restricted to human tracks. Note that sequential approaches can also be applied in a similar sliding window manner, such as in [Darrell and Pentland 1993] with DTW and in [Wilson and Bobick 1999] with HMMs.

The aforementioned global approaches are particularly efficient when used to compare spatio-temporally aligned videos. As they focus on global structure, they are, however, not robust to occlusions (*e.g.*, truncated actors), significant viewpoint changes, and duration variations. Global approaches also assume the contiguity of the video volume spanned by an action. Consequently, they are not adapted to actions composed of spatio-temporally separated events such as interactions involving multiple actors (*e.g.*, fighting). Finally, global representations are often restricted to constrained video scenarii, because they rely on accurate human localization or background subtraction, *e.g.*, for silhouette extraction. This type of information is, indeed, difficult to obtain in the presence of camera motion, dynamic and cluttered backgrounds, lighting changes, and poor video quality.

2.2 Local features

Pioneered by Chomat and Crowley [1999], Dollár et al. [2005], Laptev and Lindeberg [2003], Schüldt et al. [2004], Zelnik-Manor and Irani [2001], action recognition approaches based on local features represent videos as collections of local spatio-temporal regions. They rely on the automatic extraction of small video volumes, or sequences of image patches, directly from the pixel values or the optical flow field. Local features make no assumptions about the global structure of actions, and yield representations that are remarkably robust under uncontrolled video conditions. These methods have demonstrated competitive performance on a wide range of challenging data, including movies [Laptev et al. 2008, Marszalek et al. 2009], TV shows [Gaidon et al. 2009, Patron-Perez et al. 2010], Youtube clips [Ikizler-Cinbis et al. 2009, Liu et al. 2009], and sports broadcasts [Niebles et al. 2010, Rodriguez et al. 2008]. See [Wang et al. 2009] for a recent evaluation.

We first describe the different types of local features. We then review the two main families of recognition techniques based on local features: *local classifiers*, which deal with features individually, *e.g.*, using voting strategies, and *global approaches*, which aggregate features over video sub-volumes, *e.g.*, using a bag-of-features representation.

2.2.1 Types of local features

Local spatio-temporal features

Local spatio-temporal features are an efficient method to extract and represent the visual information from local video volumes. Indeed, they are easy to extract, as they often rely on well-known image interest point detectors generalized to the 3D spatio-temporal domain, and require neither video segmentation, nor human or object detection. For instance, Laptev [2005] proposes an extension of the Harris cornerness criterion [Harris and Stephens 1988] to detect spatio-temporal interest points (see Figure 2.6). Other spatio-temporal feature detectors include the cuboids of Dollár et al. [2005] and Willems et al. [2008]'s space-time extension of the Hessian blob detector [Beaudet 1978].

Spatio-temporal features are then described using robust local descriptors. For instance, Laptev et al. [2008] describe both local appearance and local motion in a small spatio-temporal volume using Histograms of Oriented Gradients (HOG) [Dalal and Triggs 2005] and Histograms of Optical Flow (HOF). Klaser et al. [2008] also propose the extension of HOG to the space-time domain.

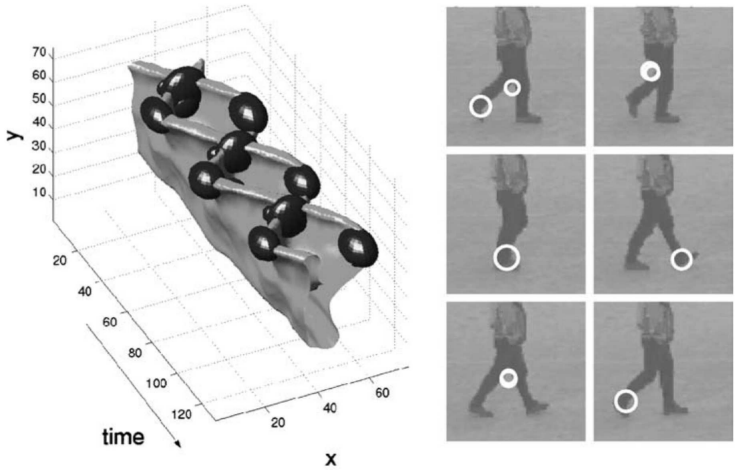


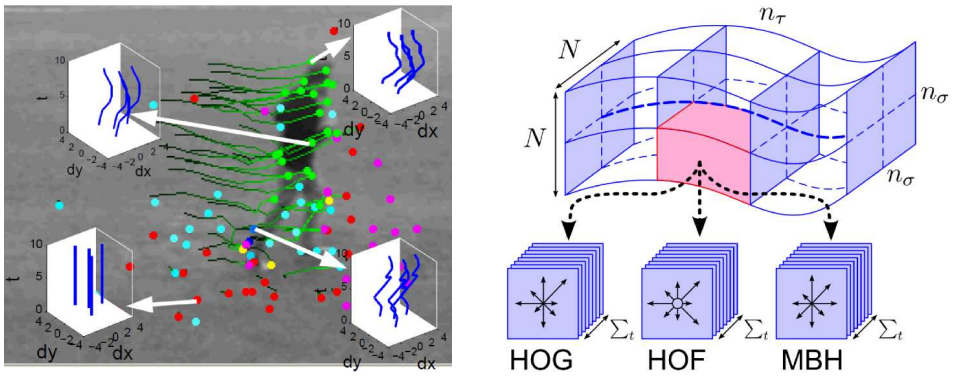
Figure 2.6: *Spatio-temporal interest points of a walking action [Laptev 2005]. The upside-down level surface of a leg pattern (left) shows that the detected interest points (the ellipsoids) correspond to corners in space-time.*

Point trajectories

The temporal evolution of spatial coordinates of tracked local image regions is another intuitive and efficient way to characterize the local motion content associated to actions. Approaches based on long-term spatio-temporal trajectories, *e.g.*, [Ali et al. 2007, Campbell and Bobick 1995, Junejo et al. 2010, Messing et al. 2009, Parameswaran and Chellappa 2006, Rao et al. 2002, Sheikh et al. 2005], have been able to automatically recognize a variety of actions from trajectory and velocity information only. However, tracking pixels and their underlying parts across many frames is a challenging problem for complex articulated movements of humans in real-world videos — essentially due to lighting changes, motion blur, compression artifacts, and occlusions. Furthermore, trajectory-based approaches often rely on the accurate detection or segmentation of body parts or objects, or require manually annotated training data that are expensive to obtain.

Tracklets — local point trajectories, corresponding to pixels tracked across a fixed small number of frames — combine useful aspects of both trajectories and local spatio-temporal features. Similar to trajectories, tracklets differentiate the time domain from the spatial one. They provide a structured and information-rich characterization of the motion of underlying parts. Similar to local features, tracklets can be easily and reliably extracted [Wang et al. 2011], for instance by simple interpolation in a motion field computed using mature optical flow extraction algorithms [Farnebäck 2003, Lucas and Kanade 1981]. Their short duration also limits drifting problems, *i.e.*, trajectories deviating from the underlying tracked object. This is in contrast to long-range trajectories [Brox and Malik 2010, Lezama et al. 2011, Sand and Teller 2008], which are expensive to compute and face difficulties in capturing fast and articulated motions due to their sensitivity to occlusions.

The modeling approaches used with local spatio-temporal features have been successfully adapted to tracklets, thanks to their limited spatio-temporal range and fixed duration. For instance, Matikainen et al. [2009] quantize trajectory snippets of key points over a vocabulary of prototypical tracks (see Figure 2.7a). In addition, the appearance and motion information of video volumes surrounding a trajectory can be represented with the aforementioned robust local descriptors (see Figure 2.7b). This yields state-of-the-art results as shown in [Wang et al. 2011]. Similarly to local spatio-temporal features, tracklets only describe local information. Furthermore, tracklet-based approaches often sample a large number of features obtained by dense sampling [Wang et al. 2011], which makes them more computationally expensive than sparse features like spatio-temporal interest points.



(a) (b)

Figure 2.7: *Trajectories as local features: (a) spatial interest point trajectories quantized over a vocabulary of prototypical "trajectons" [Matikainen et al. 2009]; (b) dense "tracklets" represented with track-aligned local appearance and motion descriptors [Wang et al. 2011].*

2.2.2 Local classifiers

Amongst methods processing features individually, voting-based approaches [Gilbert et al. 2010, Mikolajczyk and Uemura 2008, Oikonomopoulos et al. 2009, Willems et al. 2009, Yao et al. 2010, Yuan et al. 2011] aim at measuring the importance of each feature for a particular action. They detect actions by extracting local features, each of which casts a vote for a particular action. For instance, Willems et al. [2009] propose to prune quantized local features using an exemplar-based voting approach (*cf.* Figure 2.8). Yuan et al. [2011] detect spatio-temporal interest points that cast votes based on their point-wise mutual information with the action category. They, then, use a spatio-temporal branch and bound algorithm to efficiently localize actions.

Other approaches [Liu et al. 2009, Nowozin et al. 2007] propose to select the most relevant local features. For instance, Nowozin et al. [2007] use a sequence of individual local features assigned to a fixed number of uniformly sized temporal bins. Liu et al. [2009] prune local spatio-temporal features using the PageRank algorithm [Page et al. 1999], and combine static and motion features with an AdaBoost classifier [Freund and Schapire 1995].

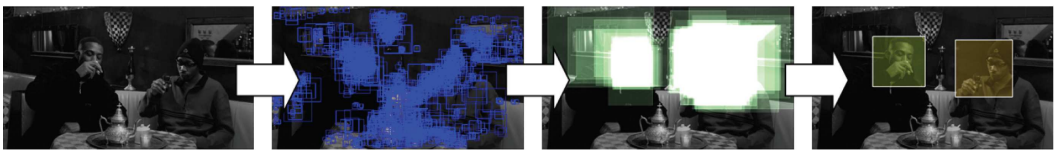


Figure 2.8: *Voting approach of Willems et al. [2009]. From left to right: (i) a video, (ii) spatio-temporal features, (iii) location hypotheses generated from the features, (iv) detections obtained by clustering the hypotheses.*

Selecting local features allows to efficiently localize the action in space-time. Voting approaches can also detect multiple co-occurring actions and simple multi-actor interactions. However, these methods assume that each local feature can provide enough information to recognize an action. Therefore, they are not discriminative enough to differentiate between complex actions sharing common motion or appearance primitives, *e.g.*, sport activities.

2.2.3 Bag-of-features

An alternative family of models uses the global distribution of features over a video volume. In particular, one of the most common and efficient representation is the bag-of-features (BOF). Originally designed to represent textual documents as word frequencies, it was successfully adapted to model images as collections of local patches [Csurka et al. 2004, Cula and Dana 2001, Sivic and Zisserman 2003] (see Figure 2.9) , and videos as bags of local spatio-temporal features [Dollár et al. 2005, Laptev et al. 2008, Niebles et al. 2008, Schüldt et al. 2004, Wang et al. 2011]. In the BOF model, a vocabulary of prototype features — called "visual words" — is obtained by clustering, and a video is represented as a histogram of occurrences of local features quantized over this visual word vocabulary (see Figure 1.6 for an illustration). Statistical learning methods like Support Vector Machines (SVM) [Schölkopf and Smola 2002] can then be applied to learn a BOF classifier. Localization can be performed by applying this classifier in a sliding window manner [Duchenne et al. 2009]. Due to its efficiency and popularity, we systematically compare our methods to this BOF approach, which we consider as a baseline to improve upon.

Note that the BOF model discards the global spatio-temporal information inherent to actions: it is an *orderless* representation. Therefore, it is not designed to discriminate between actions characterized by their structure — *e.g.*, opening and closing a door — and can cause numerous high score false alarms during localization.

Figure 2.9: *The bag-of-features image model (courtesy of Fei-Fei Li)*

2.3 Structuring collections of local features

Several works address the aforementioned shortcomings of the BOF representation. We first describe methods computing statistics on the spatial and/or temporal relations between local features. We then review models that use groups of features as mid-level building blocks for structured action models.

2.3.1 Modeling relations between local features

As stated before, the BOF model disregards the position of the local features to build robust action models. This, however, is done at the expense of discriminative power. In order to alleviate this well-known issue, a significant body of work has focused on augmenting BOF with the relations between the positions of co-occurring features, either at a global level — *i.e.*, in an entire video — or at a more local one — *i.e.*, between neighboring features.

Global relations

Spatio-temporal pyramids [Laptev et al. 2008] are a popular example of using the positions of the features to globally structure an action representation. Extending spatial pyramids [Lazebnik et al. 2006] to the spatio-temporal domain, Laptev et al. [2008] propose to divide a video into cells (sub-volumes) using a coarse spatio-temporal grid. Each cell is described by the aggregation of its local features. The full video is then described by the ordered concatenation of the per-cell BOF models, leading to a globally structured representation. The main limitation of this approach lies in the rigidity and a priori definition of the coarse spatio-temporal layout, assumed to be shared by all videos: the actors and actions must be roughly aligned in the X-Y-T video space such that cells are in correspondence. Multiple grids can be combined — at the expense of added computational complexity — but the authors showed that, ultimately, the efficiency of this technique depends on the videos and actions, while yielding only moderate improvements on average.

Wong et al. [2007] propose a more flexible way to encode the global relations between local features by extending the Implicit Shape Model (ISM) of Leibe et al. [2005] to the space-time domain. Inspired by [Fergus et al. 2005, Leibe et al. 2005], they propose to combine probabilistic Latent Semantic Analysis (pLSA) [Hofmann 2001] with an ISM in order to model both the appearance of local space-time cuboids [Dollár et al. 2005] and their relative position with respect to the center of the action. The EM algorithm is used to learn the pLSA parameters from a large co-occurrence matrix between visual words and quantized relative positions. Note that the center of the action

must be known in order to convert the absolute positions of local features to relative ones. If not available in advance, then this central location can be inferred automatically using the probabilistic Hough voting scheme from Leibe et al. [2005]. This, however, strongly degrades performance with respect to the fully supervised case, *i.e.*, with manually annotated centers. Even in this scenario, discriminative methods, *e.g.*, SVMs, clearly outperform this generative model — as reported in [Wong et al. 2007].

Our Actom Sequence Model — presented in Chapter 3 — builds on these approaches — especially the spatio-temporal pyramids of Laptev et al. [2008] — in order to integrate in a discriminative learning framework a flexible model of the global temporal relations between local features. Instead of using a fixed video subdivision, we propose to learn an action-specific decomposition from manual annotations of an action's most meaningful temporal parts.

Local relations

In contrast to the aforementioned global approaches, some methods [Gilbert et al. 2010, Kovashka and Grauman 2010, Oikonomopoulos et al. 2009, Savarese et al. 2008] focus on modeling the local relations between a feature and its spatio-temporal neighbors. In general, these methods explicitly count co-occurrences, and either build compound descriptors out of a feature and its neighborhood [Gilbert et al. 2010, Kovashka and Grauman 2010], or concatenate the usual *content BOF* with a *relations BOF* [Matikainen et al. 2010, Savarese et al. 2008]. For instance, Gilbert et al. [2010] use data mining techniques to find frequent 2D corners of an action. They group a feature with its neighbors to build compound features. They then iterate this process by mining frequently co-occurring compound features to build higher-level compound features. Recognition is performed using a voting scheme on the local hierarchical structures of 2D corner neighborhoods.

Kovashka and Grauman [2010] use a similar hierarchical process. They iteratively build increasingly complex compound features by vector-quantizing the cumulative co-occurrence histograms between a feature and its neighbors (*cf.* Figure 2.10a). A video is represented by the set of histograms of quantized features over the various levels of neighborhoods. Recognition is performed by a SVM and multiple neighborhood sizes and hierarchy levels are combined through Multiple Kernel Learning [Bach et al. 2004].

Separately modeling relations, Savarese et al. [2008] propose to describe the local neighborhood of a feature using spatio-temporal correlograms. These are the concatenation of co-occurrence histograms between visual words in a spatio-temporal neighborhood. These correlograms are then vector-quantized on a k-means-computed codebook of "correlatons", and classification is per-

formed using pLSA on histograms of correlatons. Matikainen et al. [2010] also explicitly model pairwise relations between features using a Naive-Bayes model, which assumes conditional independence of features when knowing the action class (*cf.* Figure 2.10b). They avoid the performance issues of generative models by using the log-likelihoods with respect to their per-class probabilistic co-occurrence model as features fed to a discriminative classifier.

These co-occurrence based methods face several limitations. First, counting co-occurrences between local features yields very sparse models with a vast number of parameters, which leads to overfitting and prevents these methods from generalizing. It is possible to mitigate this issue by relying on approximations and simplifications of the considered relations — *e.g.*, Matikainen et al. [2010] rely on a coarse orientation discretization. This, however, induces strong quantization effects and restricts the discriminative power of those representations. In addition, only relations between spatio-temporally close features can be captured due to the combinatorial nature of the problem.

Finally, all the aforementioned works have conducted experiments suggesting that co-occurrence models and compound features *by themselves* are performing significantly worse than the simple BOF model. Nevertheless, they have also shown that the relations between features provide complementary information that, when used in conjunction with BOF, results in performance improvements.

Our DACO kernel between time series of frames — presented in Chapter 5 — builds on these results. However, instead of relying on co-occurrences between features, it accounts for an action's dynamics, *i.e.*, the relations between its frames over time. We show that the auto-correlation between frames provides a model of the temporal dependencies, which complements the average statistics captured by BOF.

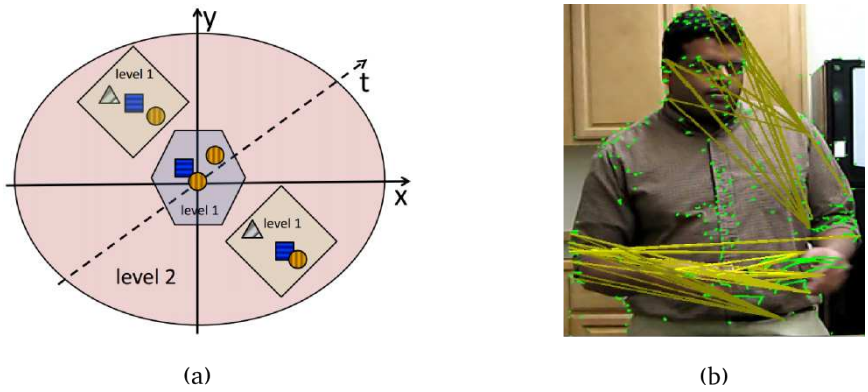


Figure 2.10: *Approaches modeling relations between local features: (a) hierarchical compound features from local co-occurrence histograms [Kovashka and Grauman 2010], (b) explicit models of pairwise relations between trajectories [Matikainen et al. 2010].*

2.3.2 Structuring groups of local features

Instead of directly modeling relations between local features, some models rely on the relations between *groups* of local features. Contrary to human-centric models, these groups — or parts — do not necessarily map to semantically meaningful entities, *e.g.*, a body part, but correspond to a data driven organization of the video content. They aim at robustly encoding the global structure of an action by studying the relations between mid-level descriptors of groups of local features.

In particular, the design of probabilistic part-based models is a popular direction of research to formalize high-level structural aspects of actions. For instance, Niebles and Fei-Fei [2007] model an action category by a constellation of parts (*cf.* Figure 2.11). Each part is associated to a distribution over positions and appearance of its constitutive local features. The content of a part is then represented as a BOF. The part layer of the generative model in Figure 2.11 encodes the geometrical relations between parts. This mixture model is learned using the EM algorithm and inference is done by maximizing an approximation of the likelihood for efficiency reasons.

Wang and Mori [2011] propose another probabilistic part-based model that encodes the spatial pairwise relationships between features in a frame. Their model is based on the hidden conditional random field (HCRF) [Quattoni et al. 2007]. These approaches, however, are limited to frame-by-frame recognition and only model spatial relations inside the same frame.

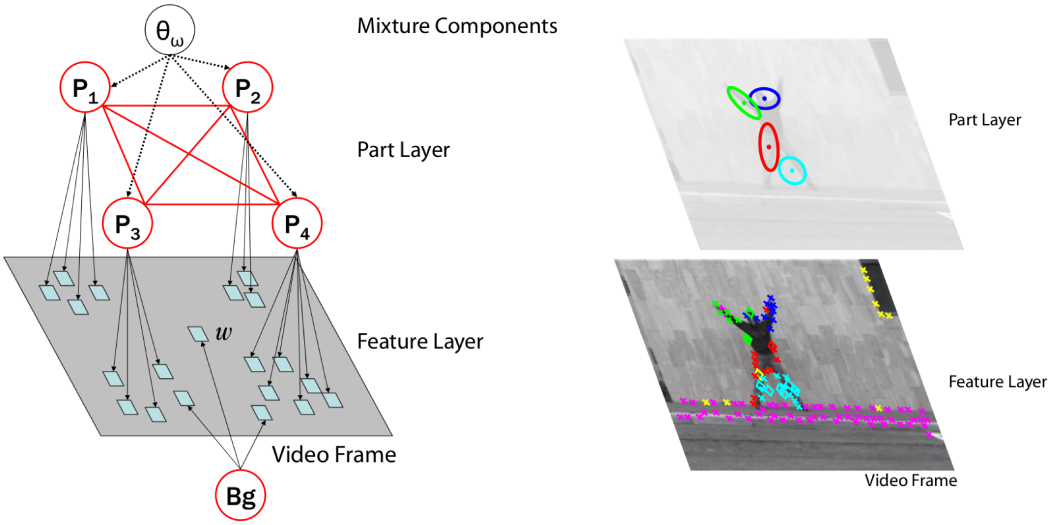


Figure 2.11: *Geometric part-based model of Niebles and Fei-Fei [2007] where an action category is modeled by a constellation of bag-of-features.*

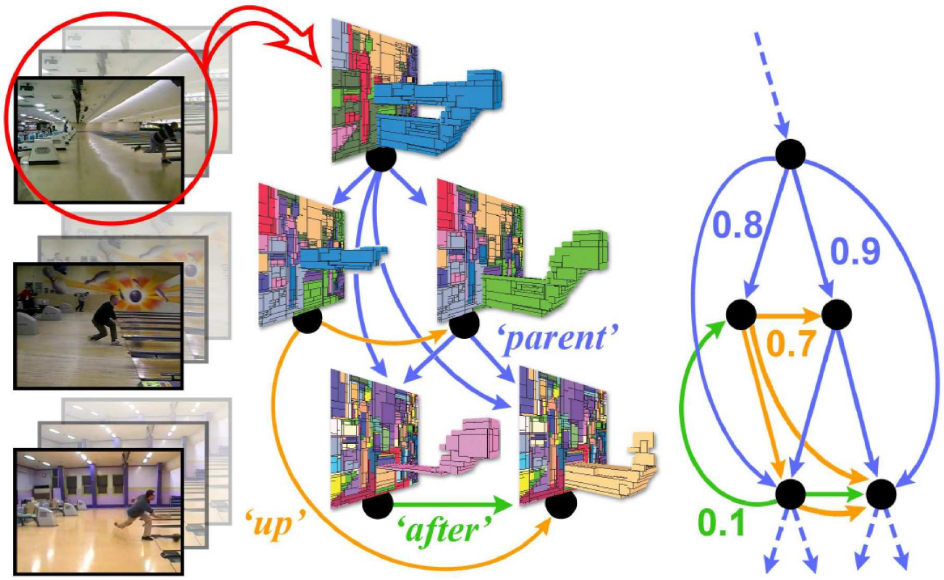


Figure 2.12: *Graphical activity model of Brendel and Todorovic [2011] obtained by hierarchical video segmentation.*

Brendel and Todorovic [2011] propose a more sophisticated graphical model which relies on hierarchical video segmentation (*cf.* Figure 2.12). They represent a video using a graph of spatial, temporal, and inclusion relations between spatio-temporal parts. A category is modeled by an average graph — containing thousands of nodes — learned by a robust least squares algorithm to account for spurious parts. Recognition is done by estimating the likelihood of a video with respect to each action graph. The resulting complex inference procedure involves approximately solving a NP-hard Quadratic Assignment Problem (QAP) to find a subgraph isomorphism between the video's graph and an action's model. This powerful generative approach highlights the importance of representing hierarchal structure. However, Brendel and Todorovic [2011] make a strong spatio-temporal consistency assumption about the data. Indeed, as they search for a subgraph isomorphism between an action instance and the graphical model of a category, their approach assumes that all action instances of a category share the same parts in the same geometrical and temporal configurations. This method is also restricted to model spatio-temporal "tubes". Their video segmentation, indeed, defines parts as continuous motions of color-consistent objects.

More discriminative approaches make use of the latent SVM of Felzenszwalb et al. [2010]. For instance, Niebles et al. [2010] discover temporal parts and learn a SVM classifier per video segment at temporal locations considered as latent variables. They use a loose hierarchical structure that is adapted to long duration activities by introducing a temporal displacement penaliza-

tion factor in segment comparisons. In addition to discovering discriminative temporal segments, Tang et al. [2012] model the transitions between hidden states and their durations using a semi-Markov Model [Hongeng and Nevatia 2003], whose parameters are learned using a latent SVM.

Extending the previous approaches to handle not only temporal segments but spatio-temporal parts, Raptis et al. [2012] use clusters of long-term point trajectories [Brox and Malik 2010] as candidate parts. They also learn a latent SVM model, where latent variables determine which clusters are the most discriminative. MAP inference over the latent variables involves solving a complex subgraph matching problem. This method relies on bounding boxes for training and assumes that all actions from a category share the same a priori fixed number of important parts.

Similar to the aforementioned models, our hierarchical action representation presented in Chapter 4 relies on modeling the content and relations between groups of local features. We, however, distance ourselves from existing work by directly comparing per video hierarchies instead of looking for a single global model of a category. Furthermore, both the content *and the structure* of our activity descriptors are specific to each video. This allows for models with a variable number of parts, which can, therefore, better model intra-class variability.

Chapter 3

Action Localization with Actom Sequence Models

Contents

3.1 Introduction

Although early action recognition experiments in simple videos have shown promising results [Blank et al. 2005, Schüldt et al. 2004], recent efforts have tried to address more challenging video sources like movies [Laptev et al. 2008]. In this chapter, we target such sources and address the problem of *temporal action localization*: finding *if and when* an action is performed in a database of long and *unsegmented* videos. In particular, we focus on searching for actions of a few seconds, like sitting down, in several hours of real-world videos.

For such realistic data, many state-of-the-art action models are based on the popular bag-of-features (BOF) [Dollár et al. 2005, Duchenne et al. 2009, Laptev et al. 2008, Niebles et al. 2008, Satkin and Hebert 2010, Schüldt et al. 2004, Wang et al. 2011], which suffers from several limitations in the context of action localization. First, a BOF ignores the temporal ordering of the frames. Therefore, we propose to address the BOF's *orderless* nature by introducing a *sequential action model* that enforces a *soft ordering* between *meaningful temporal parts*. Our robust structured model can represent actions with only approximately ordered or partially concurrent sub-events of different durations. Second, BOF-based recognition methods assume that test videos are presented in the same *temporally segmented* fashion as the training examples, *i.e.*, that they strictly contain only one action. Instead, we provide an algorithm to *learn the global temporal structure of actions*, which allows for efficient action localization in *unsegmented* videos.

Our approach is based on a simple observation: a large number of actions can be naturally defined in terms of a composition of simpler temporal parts. For instance, Figure 3.1 illustrates that the displayed actions are easy to recognize given a short sequential description. Obtaining such a decomposition is challenging, and its components are clearly action-specific.

Figure 3.1: *Examples of actom annotations for two actions.*

In this work, we propose to model an action as a small sequence of key atomic action units, which we refer to as *actoms*. These action atoms are an intermediate layer between *motion primitives* — *e.g.*, spatio-temporal interest points — and *actions*. Composed of local video features, actoms are specific to each action class, and obtained by manual annotation, though only at training time. These annotations have the same cost as specifying start and end frames of actions, while being richer and more consistent across action instances.

Closest references

Our approach extends BOF and localizes actions with a sliding central frame technique. The most similar approaches are [Duchenne et al. 2009, Satkin and Hebert 2010], which rely on multi-scale heuristics with manually defined window sizes. In contrast, our algorithm leverages a learned generative model of an action's temporal structure.

We also improve upon existing temporally structured extensions of BOF. Laptev et al. [2008] combine multiple BOF models extracted for different, manually selected, and rigid spatio-temporal grids. Multiple coarse grids are combined in a multi-channel Gaussian kernel. This approach improves over the standard BOF representation, but the temporal structure of actions is fixed and not explicitly modeled. On the contrary, we learn a temporal structure adapted to the action, and show that, compared to a rigid grid, this results in more significant gains in localization performance.

Related to our work, Niebles et al. [2010] discover motion parts based on the latent model from Felzenszwalb et al. [2009]. They learn a SVM classifier per motion segment at fixed temporal locations, whereas we do not rely on an intermediate recognition step, and use our temporal decomposition to classify actions. In addition, they use a loose hierarchical structure that is tailored to the classification of long duration activities, *e.g.*, "triple-jump", but not adapted to short actions as illustrated by their results.

Our method is similar in spirit to the state-of-the-art approaches for facial expression recognition from videos. Facial expression recognition can be performed using label information defined by the Facial Action Coding System (FACS) [Ekman and Friesen 1978], which segments facial expressions into predefined "action units", complemented with temporal annotations such as "onset", "peak", and "offset". Most approaches, however, only use peak frames for classification [Cohn and Kanade 2006], except [Simon et al. 2010]. Furthermore, as the complexity of generic human actions makes the construction of universal action units impractical, we investigate user-annotated, action-specific training actoms.

Finally, we differ from popular sequential probabilistic models like Hidden Markov Models (HMM) [Rabiner and Schafer 2007]. Indeed, instead of using

a generative model to measure the likelihood of transitions between latent states, we adopt a non-parametric discriminative approach that jointly models the content of temporally ordered parts without independence assumptions.

Outline

First, in Section 3.2, we introduce actoms and our temporally structured representation of actions in videos, called the *Actom Sequence Model* (ASM). Second, in Section 3.3, we describe our ASM classifier, and propose a simple yet efficient algorithm to learn a *generative model of the temporal structure of an action*. Third, in Section 3.4, we show how to use these models to perform temporal action localization with a *sliding central frame approach*. In addition to localizing actions, our approach can also return the most likely actoms (*cf.* Figure 3.12). In Section 3.5, we show that our method outperforms the state of the art on two challenging benchmarks for action localization: the Coffee and Cigarettes [Laptev and Pérez 2007] and DLSBP [Duchenne et al. 2009] datasets. We also demonstrate the applicability of our approach in a classification by localization setup on a larger set of actions from the Hollywood 2 dataset [Marszalek et al. 2009]. Finally, we investigate and quantify the importance of the different components of our method.

3.2 Actions as sequences of actoms

An action is decomposed into a few *temporally ordered* and *category-specific actoms*. An actom is a short atomic action, identified by its central temporal location around which discriminative visual information is present. It is represented by a temporally weighted aggregation of local features, which are described in Section 3.2.1. We model an action as a sequence of actoms by concatenating the per-actom representations in temporal order. We refer to our sparse sequential model as the Actom Sequence Model (ASM), which we define in Section 3.2.2. Finally, we describe the process used to acquire training actom annotations in Section 3.2.3.

3.2.1 Local visual information in actoms

Following [Duchenne et al. 2009, Laptev et al. 2008], we extract sparse space-time features [Laptev 2005] to represent video content. They provide a good compromise between representation power and sparsity, which allows for accurate models with a limited memory footprint — a necessary property for action localization in long video streams. We use a multi-scale space-time extension of the Harris operator to detect spatio-temporal interest points

(STIPs) [Laptev 2011]. They are represented with a concatenation of histograms of oriented gradients (HOG) and optical flow (HOF).

Once a set of local features has been extracted, we quantize them using a visual vocabulary of size v. In our experiments, we cluster a subset of 10^6 features randomly sampled from the training videos. Similar to [Duchenne et al. 2009], we use the k-means algorithm with a number of clusters set to $v = 1000$ for our localization experiments, while, similar to [Wang et al. 2009], we use $v = 4000$ for our classification-by-localization experiments. We then assign each feature to the closest visual word.

3.2.2 The Actom Sequence Model

We define the time-span of an actom with a *radius* around its temporal location. We propose an adaptive radius that depends on the relative position of the actom in the video sequence. The adaptive radius r_i, for the actom at temporal location t_i, in the sequence of a actom locations $(t_1, \cdots, t_a)$, is parametrized by the *overlap ratio* ρ between adjacent actoms:

$$r_i = \frac{\delta_i}{2-\rho} \quad , \quad \delta_i = \begin{cases} t_2 - t_1 & \text{if} \quad i = 1 \\ t_a - t_{a-1} & \text{if} \quad i = a \\ \min(t_i - t_{i-1}, t_{i+1} - t_i) & \text{if} \quad 1 < i < a \end{cases} \tag{3.1}$$

where ρ ranges between 0 and 1, and δ_i is the distance to the closest actom. This defines a symmetric neighborhood around the temporal location specific to each actom of an action. Visual features are computed only within the forward and backward time range defined by the actom's radius. They are, then, accumulated in per-actom histograms of visual words — *cf.* Figure 3.2.

Our model only assumes a weak temporal ordering of the actoms. In addition, defining the actom's time-span relatively to its closest neighbor has multiple advantages. On the one hand, it allows adjacent actoms to overlap and share features, while enforcing a soft temporal ordering. This makes the model robust to inaccurate temporal actom localizations and to partial orderings between concurrent sub-events. On the other hand, it also allows for gaps between actoms and can, therefore, represent discontinuous actions. Furthermore, an adaptive time-span makes the model naturally robust to variable action duration and speed.

We also introduce a *temporally weighted assignment scheme*. We propose to aggregate temporally weighted contributions of per-actom features. Each feature at temporal location t in the vicinity of the i^{th} actom, *i.e.*, if $|t - t_i| \leq r_i$, is weighted by its temporal distance to the actom:

$$w_i(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(t-t_i)^2}{2\sigma^2}\right) \tag{3.2}$$

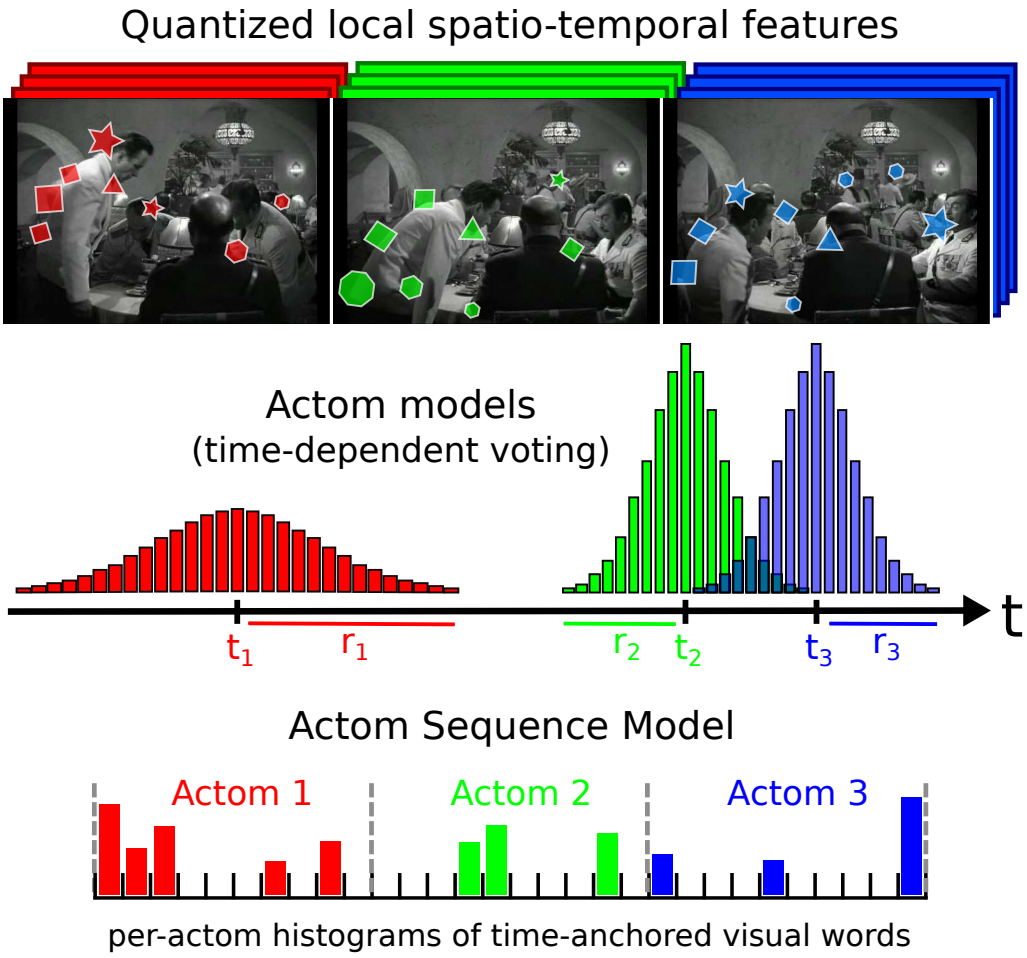


Figure 3.2: *Construction of our ASM action model using actom-based annotations and a temporal weighting scheme for aggregating local features in a sparse temporally structured bag-of-features.*

Hence, features further from an actom's center vote with a smaller importance. See the actom models depicted in Figure 3.2 for an illustration. This scheme offers an intuitive way to tune the bandwidth σ of the weighting window using the Chebyshev inequality. For a random variable X of mean μ and finite standard deviation σ, we know that $\mathbf{P}(|X - \mu| \geq k\sigma) \leq 1/k^2$, for any $k > 0$. Rewriting this with $X = t$, $\mu = t_i$ and $r_i = k\sigma$, we obtain:

$$\mathbf{P}(|t - t_i| < r_i) \leq p, \quad p = 1 - \frac{\sigma^2}{r_i^2} \tag{3.3}$$

The probability p is the "peakyness" of our soft-assignment scheme and replaces σ as a hyper-parameter of our model. It allows to encode a prior on the amount of probability mass of features falling in an actom's time range.

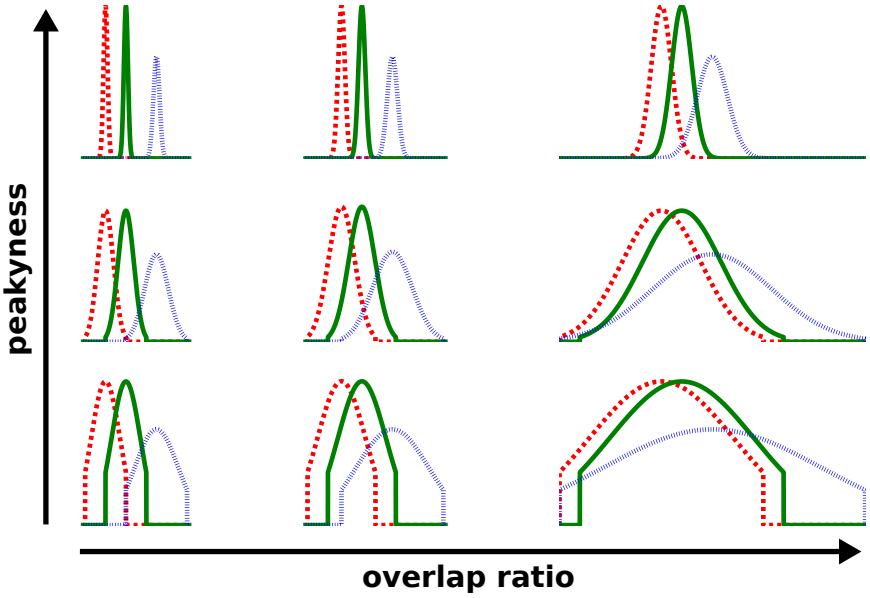


Figure 3.3: *Illustration of the actom models in ASM ($a = 3$) for the same actom locations, but different ASM hyper-parameters. This shows the influence of the overlap (ρ) and peakyness (p) on the per-frame weights. ASM encompasses a continuum of models, ranging from sequences of "soft" key-frames (upper left), to BOF-like models (lower right).*

See Figure 3.3 for an illustration of the influence of the overlap ρ and peakyness p on the actom-specific per-frame weights. In our experiments, we set the parameters ρ and p per-class by maximizing localization performance on a held out, unsegmented, validation video. We found that this hyper-parameter optimization scheme yielded better results than cross-validation on the segmented training clips. Cross-validation, indeed, only maximizes window classification performance, which is an easier problem than localization, as the training videos are already segmented.

To summarize, we derive our ASM model from a sequence of a actom locations by (i) computing visual features only in the actoms's time-spans, parametrized by the ρ parameter (Eq. 3.1), (ii) computing the feature contributions to per-actom temporally weighted histograms (Eq. 3.2), and (iii) appending these histograms into a temporally ordered sequence, *i.e.*, our ASM representation of videos $\mathbf{x} = (x_{1,1}, \cdots, x_{1,v}, \cdots, x_{a,1}, \cdots, x_{a,v})$, where:

$$x_{i,j} = \sum_{t=t_i-r_i}^{t_i+r_i} w_i(t) c_j(t) \tag{3.4}$$

In Equation 3.4, $x_{i,j}$ is the weighted sum of the number $c_j(t)$ of local features detected at frame t and assigned to visual word j, over the i^{th} actom's time-span $[t_i - r_i, t_i + r_i]$. The ASM vector $\mathbf{x}$ is then L_1-normalized.

3.2.3 Actom annotations

We now present how to obtain actom annotations and the advantages of this supervision over annotating beginning and ending frames of actions.

Obtaining annotations

An actom annotation is a time stamp in the corresponding video. This temporal location is selected such that its neighboring frames contain visual information that is representative of a part of the action. The number of actoms is fixed depending on the action category. We observed that only a few actoms are necessary to unambiguously recognize an action from their sequence (*cf.* examples in Figure 3.1). We use three actoms per action example in our experiments. Note that only positive training examples are manually annotated. In general, this corresponds to a small fraction of the training data — most action recognition benchmarks use in the order of 100 action examples per category. The initial noisy set of candidate training clips can be automatically obtained by using external data, *e.g.*, with simple textual queries on movie transcripts [Gaidon et al. 2009].

During the annotation process, semantic consistency in the choice of actoms across different video clips is necessary: the i^{th} actom of an action should have a single interpretation, *e.g.*, "recipient containing liquid coming into contact with lips" for the drinking action. This is ensured by giving precise guidelines to annotators, and by making multiple annotators label or correct each example. Note, however, that our approach is robust to the violation of this assumption, *i.e.*, we can still model actions, even when an actom has a few possible meanings across training examples.

After all the training examples are annotated once, we perform a simple outlier detection step using the temporal structure model described in Section 3.3.2. First, we learn a model of the temporal structure and estimate the likelihood of each annotation according to this model. We, then, resubmit for annotation the inconsistently annotated examples, *i.e.*, those below a likelihood threshold (we use 2%). After these samples are re-annotated, we update the model of the temporal structure and re-estimate the likelihood of each annotation. We iterate this process up to three times.

Practical observations

Consistent actom annotations are easier to obtain than precise action boundaries. For instance, it is unclear whether a person walking towards a door before opening it is a part of the action "Open Door". In contrast, the time at which the door opens can be unambiguously determined. Duchenne et al.

[2009] and Satkin and Hebert [2010] observed that temporal boundaries of actions are not precisely defined in practice. Furthermore, they show that inaccurate boundary annotations significantly degrade the recognition performance. Therefore, they propose to improve the quality of annotated action clips by automatically cropping their temporal boundaries in a discriminative manner. They, however, only model the temporal extent of actions, not their temporal structure. On the contrary, actom annotations are well defined as a few frames of precise atomic events. Consequently, annotating actoms leads to smaller annotation variability. Figure 3.4 quantitatively illustrates this claim. It shows that the ground truth annotations for the action "sitting down" have a smaller duration variance when actoms are annotated instead of beginning and ending frames. We also observed that the average annotation time per action is comparable for both of these annotation types — we measured between 10 and 30 seconds per action. In addition, an actom is a visual phenomenon that is deemed semantically relevant by annotators, and not an automatically learned part of the action. It is, therefore, always possible to interpret a predicted actom sequence. Moreover, we show that it leads to discriminative representations for action recognition.

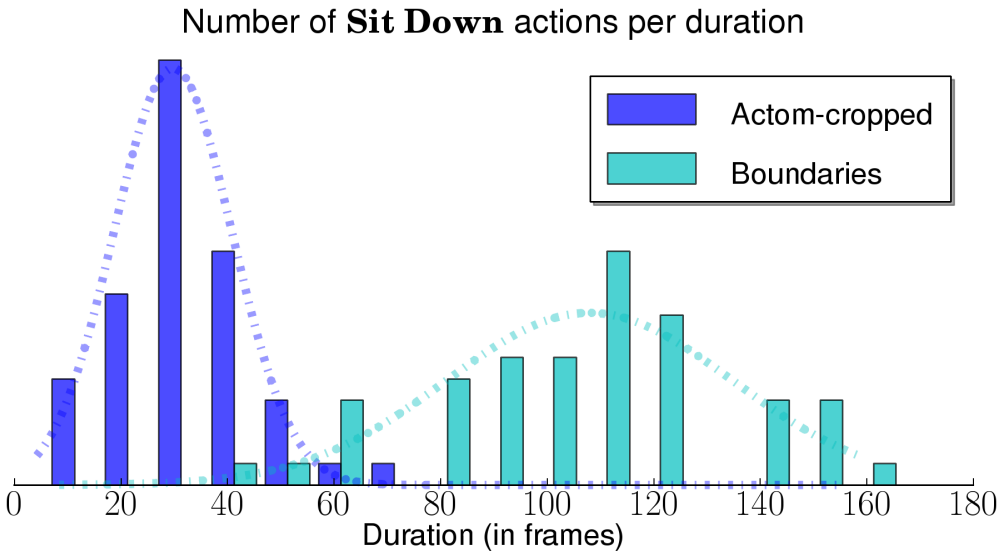


Figure 3.4: *Frequencies of action durations obtained from manual annotations for the action "Sit Down". "Boundaries" depict the duration of ground truth annotations from [Duchenne et al. 2009], obtained by labeling beginning and end frames of the action. "Actom-cropped" represents the time interval between the first and the last actom.*

3.3 Temporal action detector learning

In the following, we detail the training part of our localization algorithm. First, we give details on the action classifier operating on our ASM representation (Section 3.3.1). Then, we describe how we learn a generative model of an action's temporal structure in order to sample likely actom candidates at test time (Section 3.3.2). Finally, we show how to obtain negative training examples (Section 3.3.3).

3.3.1 ASM classifier

Our localization method is similar to the sliding-window approach. It consists in applying a binary classifier at multiple temporal locations throughout the video, in order to determine the probability that the queried action is being performed at a particular moment. We use a Support Vector Machine (SVM) [Schölkopf and Smola 2002] trained to discriminate between the action of interest and all other visual content. As ASM is a histogram-based representation, we can use a non-linear SVM with the χ^2 or the intersection kernel [Hein and Bousquet 2005, Laptev et al. 2008]. For efficiency reasons, we choose to use the intersection kernel [Maji et al. 2008]. It is defined for any $x = (x_1, \ldots, x_v)$ and $x' = (x'_1, \ldots, x'_v)$ as $K(x, x') = \sum_{j=1}^{v} \min(x_j, x'_j)$. Note that, in our case, using a non-linear SVM does not prohibitively impact the localization speed, because the size of our training set is small.

In this set-up, the negative class spans all types of events except the action of interest. Therefore, more negative training examples than positive ones are necessary. We use a SVM with class-balancing [Lin et al. 2002] to account for this imbalance between the positive and negative classes. Assume we have a set of labeled training examples $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \{-1, 1\}$, where $\mathcal{X}$ is the space of ASM models. Let n_+, resp. n_-, denote the number of positive, resp. negative, examples, and $n = n_+ + n_-$ the total number of examples. The binary SVM classifier with class-balancing minimizes the regularized cost function:

$$\frac{1}{n} \sum_{i=1}^{n} L(y_i) \ell(y_i, f(x_i)) + \lambda \|w\|^2_{\mathcal{H}} \tag{3.5}$$

with $f(x_i) = w^T \phi(x_i) + b$, $w \in \mathcal{H}$, $\mathcal{H}$ the feature space associated with the kernel K, $\phi : \mathcal{X} \to \mathcal{H}$ the corresponding feature map, $\ell(y, f) = \max(0, 1 - yf)$ the hinge loss, $L(+1) = 1/n_+$, $L(-1) = 1/n_-$, and λ a regularization parameter. In order to return probability estimates, we fit a sigmoid function to the decision function f learned by the SVM [Lin et al. 2007, Platt 2000]. Our ASM classifier evaluates the posterior probability of an action being performed, *knowing its actoms*.

3.3.2 Generative model of temporal structure

For unseen videos, we do not know the temporal locations of the actoms. Therefore, we learn a generative model of the temporal structure allowing to sample likely actom candidates at test time. The temporal structure we estimate is the distribution of inter-actom spacings from the training sequences: $\{\mathbf{\Delta}_i = (t_{i,2}-t_{i,1}, \ldots, t_{i,a}-t_{i,a-1}), i = 1 \ldots n_+\}$, where a is the number of actoms of the action category and n_+ is the number of positive training examples.

In practice, we have only few actom annotations, typically $n_+ \leq 100$, which, in addition, can significantly differ from one another. Therefore, using histograms to model the actom spacings yields a too sparse estimate with many empty bins. Instead, we make the assumption that there is an underlying smooth distribution, which we estimate via non-parametric Kernel Density Estimation (KDE) [Rosenblatt 1956, Wasserman 2004]. This makes the assumption that there is a *continuum of execution styles* for the action of interest, and it allows to correctly interpolate unseen, but likely, temporal structures. We use KDE with Gaussian kernels whose bandwidth h is automatically set using Scott's factor [Scott 1992]: $h = n_+^{-\frac{1}{a+4}}$. We obtain a continuous distribution $\mathcal{D}$ over inter-actom distances $\mathbf{\Delta} = (t_2 - t_1, \ldots, t_a - t_{a-1})$:

$$\mathcal{D} \sim \frac{1}{n_+ h^{a-1}\sqrt{2\pi}} \sum_{i=1}^{n_+} \exp\left(-\frac{||\mathbf{\Delta} - \mathbf{\Delta}_i||^2}{2h^2}\right). \tag{3.6}$$

As we deal with discrete time steps (frames), we discretize this distribution in the following way. First, we sample 10^4 points, randomly generated from our estimated density $\mathcal{D}$. Second, we quantize these samples by clustering them with k-means. This yields a set of s centroids $\{\hat{\mathbf{\Delta}}_j \ , \ j = 1 \cdots s\}$ and their associated Voronoi cells that partition the space of likely temporal structures. Third, we compute histograms by counting the fraction $\hat{p}_j$ of the random samples drawn from $\mathcal{D}$ that belong to each cell j. This results in the discrete multi-variate distribution:

$$\hat{\mathcal{D}} = \{(\hat{\mathbf{\Delta}}_j, \hat{p}_j) \ , \quad j = 1 \cdots s\}, \quad \hat{p}_j = \mathbf{P}(\hat{\mathbf{\Delta}}_j). \tag{3.7}$$

Finally, we truncate the support of $\hat{\mathcal{D}}$ by removing structures with a probability smaller than 2% (outliers), and re-normalize the probability estimates. Figure 3.5 gives an example of the distribution $\hat{\mathcal{D}}$ learned for the "smoking" action. Note that s corresponds to the size of the support of $\hat{\mathcal{D}}$, *i.e.*, the number of likely candidate actom spacings. This parameter controls a trade-off between the coarseness of the model of the temporal structure, and its computational complexity. We used $s = 10$ for all actions in our experiments.

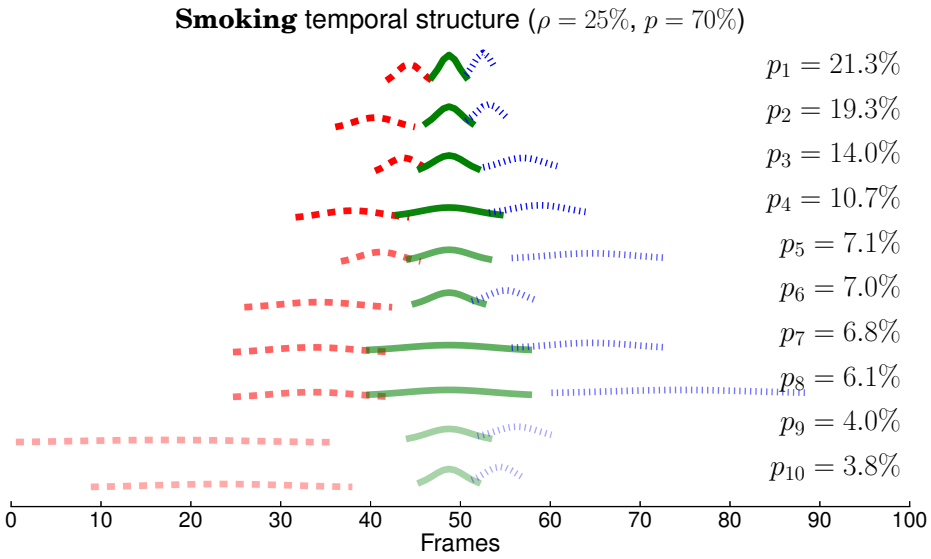


Figure 3.5: *Temporal structure learned for "smoking" from the Coffee & Cigarettes dataset. The candidate actom models* $(\hat{\mathbf{\Delta}}_j, p_j) \in \hat{\mathcal{D}}$ *are sorted by their estimated prior probability* p_j.

3.3.3 Negative training examples

Our localization method relies on a binary classifier discriminating between an action of interest and *all other* events. To obtain negative examples, we randomly sample clips from the unlabeled part of the training database, and filter out those intersecting annotated training positives. To be consistent with the localization stage at test time, we randomly sample actoms according to the learned temporal structure $\hat{\mathcal{D}}$. There are, however, several practical issues associated with the extraction of such random negatives. First, the number of random negatives needed to learn a good detector is not a priori obvious. Second, the unlabeled part of the database from which these negatives are sampled might still contain an unknown number of positives that were not annotated. Indeed, the automatic techniques used to help in the acquisition of positive examples (*e.g.*, [Gaidon et al. 2009, Laptev et al. 2008]) might miss many action examples. Therefore, randomly sampled windows have a non-negligible chance of containing the action of interest, and our negative examples might contain a significant number of false negatives compared to the number of true positives. Note that this problem also rules out the possibility to mine so-called "hard negative" examples for a re-training stage [Dalal and Triggs 2005]. Consequently, instead of generating as many negative windows as possible, we observed that sampling a number of negatives that is less than ten times the number of training positives yields a better performance — *cf.* Section 3.5.6 for more details. Furthermore, this significantly reduces the running time of our kernel-based classifier.

3.4 Localization with actoms

In this section, we describe our temporal localization approach (Section 3.4.1), some post-processing steps (Section 3.4.2), and a strategy for action classification in approximatively pre-segmented videos (Section 3.4.3).

3.4.1 Sliding central frame localization

To localize actions in a test sequence, we apply our ASM classifier in a sliding window manner. However, instead of sliding a temporal window of fixed scale, we shift the *temporal location of the middle actom* t_m, where $m = \lfloor a/2 \rfloor$, and a is the number of actoms for the action category. We use a temporal shift of 5 frames in our experiments. Given a central actom location t_m, we compute the probability of the action occurring at t_m by marginalizing over our generative model of inter-actom spacings $\hat{\mathcal{D}}$:

$$\begin{aligned} \mathbf{P}(\text{action at } t_m) &= \sum_{j=1}^{s} \mathbf{P}(\text{action at } t_m \mid \hat{\mathbf{\Delta}}_\mathbf{j})\ \mathbf{P}(\hat{\mathbf{\Delta}}_\mathbf{j}) \\ &= \sum_{j=1}^{s} f_{\text{ASM}}(\hat{t}_{j,1}, \cdots, t_m, \cdots, \hat{t}_{j,a})\ \hat{p}_j \end{aligned} \tag{3.8}$$

where f_{ASM} is the a posteriori probability estimate returned by our SVM classifier trained on ASM models (Eq. 3.5). See Figure 3.6 for an illustration.

Alternatively, taking the maximum a posteriori allows to not only localize an action, but also its most likely temporal structure. We have experimentally observed that, for the goal of temporal localization, marginalizing yields more stable results than just taking the best candidate actoms. The temporal structures in $\hat{\mathcal{D}}$ are indeed related. This is a consequence of our assumption on smoothly varying styles of execution (*cf.* Figure 3.5). Therefore, the redundancy in $\hat{\mathcal{D}}$ makes marginalizing over actom candidates robust to inaccurate actom placements.

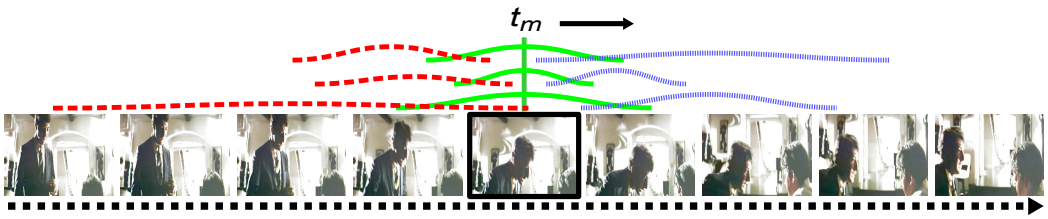


Figure 3.6: *Sliding central frame temporal localization. The probability of an action being performed at frame t_m is evaluated by marginalizing over all actom candidates learned with our model of the temporal structure.*

Note that Equation 3.8 provides a way to perform multi-scale localization differently than the usual multi-scale sampling heuristic. Both the sequential structure and the duration of the action is modeled by $\hat{\mathcal{D}}$, whereas traditional sliding-window approaches, *e.g.*, [Laptev and Pérez 2007], manually specify a fixed set of window sizes.

3.4.2 Post-processing

Our algorithm estimates a probability of localization every N^{th} frame. In order to localize the action, we also return an estimate of the temporal extent of each localization, *i.e.*, a temporal window surrounding each high-scoring frame. As we deal with short actions with limited duration variation, we choose to use a single, fixed, action-specific duration for our temporal windows. We use a temporal scale such that a localization window contains all frames used in the computation of the score of its central frame. As we marginalize over $\hat{\mathcal{D}}$, this defines a single scale per action category, which only depends on the largest actom spacings in $\hat{\mathcal{D}}$. In practice, we obtained the constant window sizes of 95 frames for "drinking" and "smoking" in the Coffee and Cigarettes dataset, 85 frames for "opening a door" and 65 frames for "sitting down" in DLSBP. In addition, as the temporal shift between two localizations can be small in practice, we use a non-maxima suppression algorithm to remove overlapping localization windows. We recursively (i) find the maximum of the scores, and (ii) delete overlapping windows with lower scores. Windows are considered as overlapping if the Jaccard coefficient — the intersection over the union of the frames — is larger than 20%.

3.4.3 Classification by localization

Although designed for temporal localization, our method is also applicable to action classification. In both cases, the training data and learning algorithms are the same. The test data, however, differs. For localization, we process continuous streams of frames. In contrast, unseen data for classification come in the form of pre-segmented video clips.

The classification goal is to tell whether or not the action is performed in an unseen video clip, independently of *when* it is performed. Consequently, after applying our sliding central frame approach to label every N^{th} frame of a new test clip, we pool all localization scores to provide a global decision for the entire clip. Similarly to Satkin and Hebert [2010], we found that max-pooling — *i.e.*, taking the best localization score as classification score — yields good results in our set up. Indeed, marginalizing over actom candidates limits the number and the score of spurious false localizations, thanks to the redundancy in the learned temporal structure.

3.5 Experimental evaluation

This section presents experimental results comparing our ASM-based approach with BOF-based alternatives and the state of the art. We, first, introduce the datasets used in our experiments (Section 3.5.1). We, then, describe how we measure the localization performance (Section 3.5.2), and what baseline localization methods we compare to (Section 3.5.3). Our localization results are reported in Section 3.5.4, while our classification results are reported in Section 3.5.5. Finally, we quantify and discuss the influence of the parameters of our method (Section 3.5.6).

3.5.1 Datasets

We use two challenging movie datasets for action localization: "Coffee and Cigarettes" [Laptev and Pérez 2007] and "DLSBP" [Duchenne et al. 2009]. Note that they are currently among the largest public benchmarks for action localization in realistic settings. There are, indeed, only few action localization datasets, due to the high cost incurred by the manual annotation of the full test sequences. We also use the "Hollywood 2" dataset [Marszalek et al. 2009] for our classification by localization experiments. These datasets are provided with annotations in the form of temporal boundaries delimiting actions.

Coffee and Cigarettes [Laptev and Pérez 2007]. This dataset is designed for the localization of two action categories in movie data: "drinking" and "smoking" (*cf.* Figure 3.7). Initially, Laptev and Pérez [2007] evaluated action localization for the drinking action only. Evaluation for the smoking category was later added by Kläser et al. [2010b]. The training sets contain 106 drinking and 78 smoking clips: 41 drinking and 70 smoking examples from six short stories — with different actors and settings — of the "Coffee and Cigarettes" movie, 32 drinking and 8 smoking clips from the movie "Sea of love", and 33 drinking examples recorded in a laboratory setting. The test set for the drinking action consists of two short stories (36,000 frames) containing 38 drinking actions. The test set for the smoking action consists of three short stories (32,000 frames) containing 42 smoking actions. There is no overlap between the training and test sets, in terms of both scenes and actors.

Figure 3.7: *Actions from the Coffee and Cigarettes dataset*

DLSBP [Duchenne et al. 2009]. Named after its authors, this dataset consists of two action categories: "Open Door" and "Sit Down" (*cf.* Figure 3.8). The training sets include 38 "Open Door" and 51 "Sit Down" examples extracted from 15 movies. Three movies are used as test set (440,000 frames), containing a total of 91 "Open Door" and 86 "Sit Down" actions. This dataset is more challenging than Coffee and Cigarettes, because the test data is larger by one order of magnitude, the actions are less frequent, and the video sources are more varied. Note that the chance level for localization, *i.e.*, the probability of randomly finding the positives, is of approximatively 0.1% for the Coffee and Cigarettes dataset, and 0.01% for the DLSBP dataset.

Figure 3.8: *Actions from the DLSBP dataset*

Hollywood 2 [Marszalek et al. 2009]. This classification dataset consists of 1707 video clips — 823 for training, 884 for testing — extracted from 69 Hollywood movies. There are 12 categories: answering a phone, driving a car, eating, fighting, getting out of a car, hand shaking, hugging, kissing, running, sitting down, sitting up, and standing up (*cf.* Figure 3.9 for some examples). Note that we use uniformly sampled actoms for the "fighting" category, as it could not be annotated with consistent actoms. Indeed, fighting *scenes* in movies are not short, sequentially defined actions, but involve various actions in no particular order, *e.g.*, punching and kicking.

Figure 3.9: *Some of the actions from the "Hollywood 2" dataset*

3.5.2 Evaluation criteria

For temporal localization, we use two evaluation criteria to determine if a test window is matching a ground truth action (*cf.* Figure 3.10). We first consider the most commonly used criterion [Duchenne et al. 2009, Kläser et al. 2010b, Laptev and Pérez 2007], referred to as OV20: a window matches a ground truth action if the Jaccard coefficient (intersection over union) is more than 20%. We use the original ground truth start and end frame annotations provided by the dataset authors. This criterion, however, does not guarantee that a localization will contain enough of the action to be judged relevant by a user. For instance, a localization relevant according to OV20 may contain a person walking towards a door, but not the door opening itself.

Therefore, in addition to OV20, we introduce a more precise matching criterion based on ground truth actom annotations. Referred to as OVAA, for "overlap all actoms", it states that a test window matches a ground truth test action only if it contains *the central frames of all ground truth actoms*. The OVAA criterion stems from the definition of actoms as the minimal set of sub-events needed to recognize an action. Hence, a correct localization must, by definition, contain all actoms. In consequence, we also annotate actoms for the positive test examples to assess ground truth according to OVAA. These annotations are not used at test time. Note that a single window covering the entire test sequence will always match the ground truth according to the OVAA criterion. This bias, however, is not present in our comparisons as all methods have comparable window sizes of approximatively 100 frames or less.

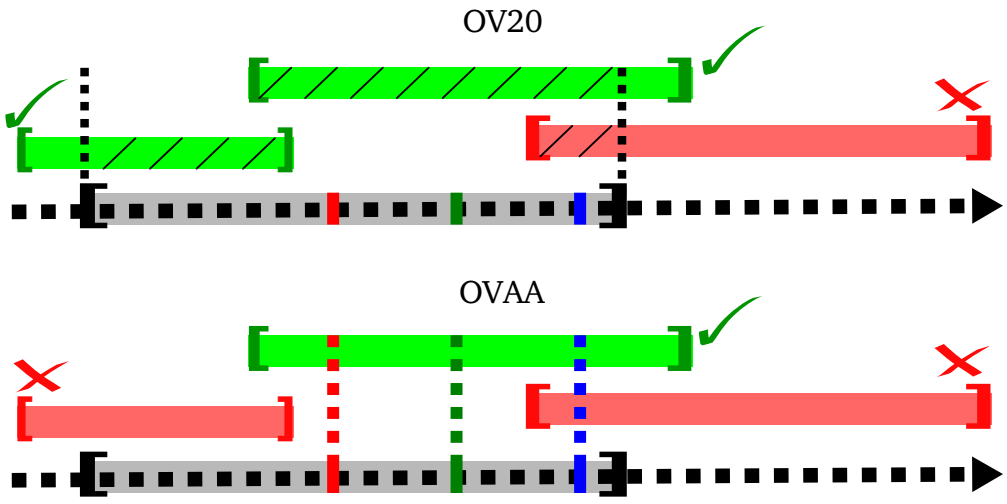


Figure 3.10: *Overlap criteria used for localization OV20 (intersection over union of at least 20% with the ground truth) and OVAA (a match needs to contain all ground truth actom frames).*

We use both criteria in our evaluation, because they provide complementary insights. If after non-maxima suppression there are multiple windows matching the same ground truth action, we only consider the one with the maximal score as a true positive, while the other localizations are considered as false positives. This is similar to the evaluation of object detection, *e.g.*, in the Pascal VOC challenge [Everingham et al. 2010]. Note that, for classification by localization, no matching criterion is required as we return one score for each test video. In all cases, we follow the state of the art and measure performance in terms of precision and recall by computing the Average Precision (AP).

3.5.3 Bag-of-features baselines

We compare our approach to two baseline methods: the standard bag-of-features (BOF), and its extension with a regular temporal grid. To make the results comparable, we use the same visual features, vocabularies, and kernel as the ones used for our ASM model. In addition, for the action localization experiments, we crop the original annotations of the positive training samples around the training actoms, which we further extend by a small offset — half the inter-actom distances for each sequence — in order to use an amount of frames comparable to what ASM uses. A similar, but automatic, cropping step was shown to improve performance by Satkin and Hebert [2010]. Furthermore, we use the same random training negative samples as the ones used by our ASM approach. This allows for a fair comparison between ASM and BOF-based methods.

At test time, BOF-based sliding window approaches require the *a priori* definition of multiple temporal scales. We learn the scales from the positive training examples using a generative model similar to the one used for actoms (*cf.* Section 3.3.2). Like for our sliding central frame method, we use a step-size of 5 frames for the sliding window approach in all of our experiments. We, finally, apply a non-maxima suppression post-processing step similar to the one described in Section 3.3.2, and commonly used in the literature, *e.g.*, in [Kläser et al. 2010b].

As mentioned previously, in addition to the global BOF baseline, we evaluate its extension with regular temporal grids [Laptev et al. 2008]. We use a fixed grid of three equally sized temporal bins, which in practice gave good results, and is consistent with our number of actoms. First, the video is cut in three parts of equal duration: beginning, middle, and end. A BOF is then computed for each part, and the three histograms are concatenated. This method is referred to as "BOF T3" in the following.

3.5.4 Localization results

We report temporal localization results in Table 3.1 for the Coffee and Cigarettes dataset, and in Table 3.2 for the DLSBP dataset. We compare our method (ASM), two baselines (BOF and BOF T3), and recent state-of-the-art results. Where possible, we report the mean and standard deviation of the performance over five independent runs with different random negative training samples. Figure 3.11 shows frames of the top five results for "drinking" and "open door" obtained with our method. Some examples of automatically localized actoms with our ASM method are depicted in Figure 3.12. In the following, we discuss how our ASM model compares to both our bag-of-features baselines and the state of the art.

Figure 3.11: *Frames of the top 5 actions localized with ASM for "Drinking" (top row) and "Open Door" (bottom row, #2 is the only false positive).*

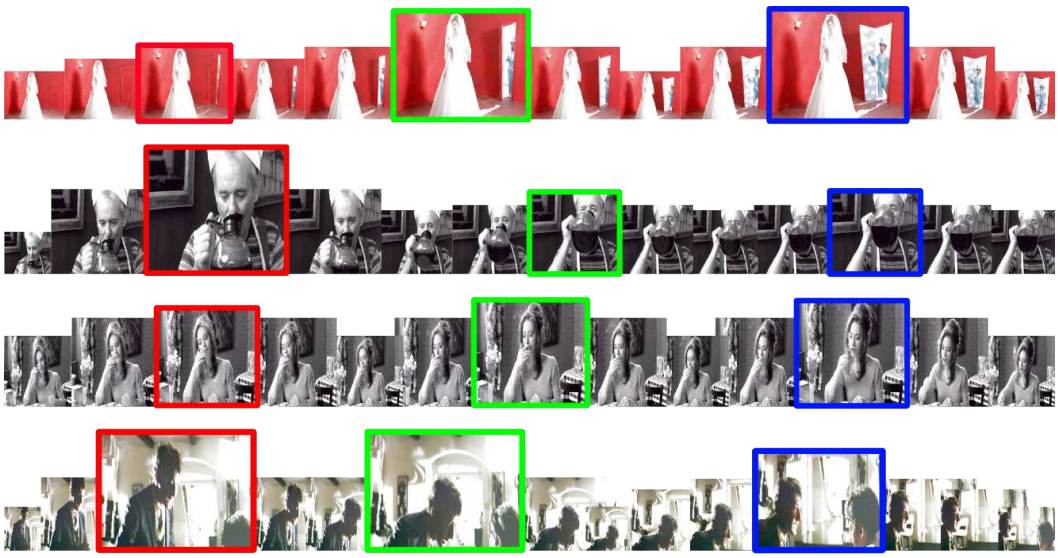


Figure 3.12: *Automatically localized frames of actoms for 4 action sequences (from top to bottom): opening a door, drinking, smoking, and sitting down.*

Method	"Drinking"	"Smoking"
matching criterion: OV20		
DLSBP [Duchenne et al. 2009]	40	N.A.
LP-T [Laptev and Pérez 2007]	49	N.A.
KMSZ-T [Kläser et al. 2010b]	59	33
BOF	36 (±1)	17 (±2)
BOF T3	44 (±2)	20 (±3)
ASM	**63** (±3)	**40** (±4)
matching criterion: OVAA		
BOF	10 (±3)	1 (±0)
BOF T3	21 (±4)	3 (±1)
ASM	**62** (±3)	**27** (±3)

Table 3.1: *Action localization results in Average Precision (in %) on the Coffee and Cigarettes dataset. ASM refers to our method.*

Method	"Open Door"	"Sit Down"
matching criterion: OV20		
DLSBP [Duchenne et al. 2009]	14	14
BOF	8 (±3)	14 (±3)
BOF T3	8 (±1)	17 (±3)
ASM	**14** (±3)	**22** (±2)
matching criterion: OVAA		
BOF	4 (±1)	3 (±1)
BOF T3	4 (±1)	6 (±2)
ASM	**11** (±3)	**19** (±1)

Table 3.2: *Action localization results in Average Precision (in %) on the DLSBP dataset. ASM refers to our method.*

Comparison to bag-of-features

We perform better than BOF according to both evaluation criteria. The improvement is significant with OV20: +27% for "Drinking", +23% for "Smoking", +6% for "Open Door", and +8% for "Sit Down". BOF is also less precise than our approach. Indeed, the performance of BOF drops when changing the matching criterion from OV20 to the more restrictive OVAA, *e.g.*, −26% for "Drinking". In contrast, our ASM model is more accurately localizing all action components, and the relative gap in performance with respect to the baseline increases even more when changing from OV20 to OVAA, *e.g.*, from +27% to +52% for "Drinking", and from +8% to +16% for "Sit Down".

Rigid *v.s.* adaptive temporal structure

The flexible temporal structure modeled by ASM allows for more discriminative models than BOF T3. Using the fixed temporally structured extension of BOF increases performance, but is outperformed by our model on all actions. This confirms that the variable temporal structure of actions needs to be represented with a flexible model that can adapt to different durations, speeds, and interruptions.

Comparison to the state of the art

The method of Laptev and Pérez [2007] is trained for spatio-temporal localization with stronger supervision in the form of spatio-temporally localized actions. We compare to the mapping of their spatio-temporal localization results to the temporal domain as reported in [Duchenne et al. 2009], *cf.* row "LP-T" in table 3.1. Similarly, Kläser et al. [2010b] learn from spatio-temporally localized training examples. The mapping of their results to the temporal domain are reported in the "KMSZ-T" row of table 3.1. On the DLSBP dataset, we compare to the original "ground truth" results of the authors in [Duchenne et al. 2009]. They use a similar set-up to our BOF baseline. The differences between their approach and our BOF baseline lies mostly in the negative training samples and, to a lesser extent, in the visual vocabulary. Our experiments show that ASM outperforms these state-of-the-art approaches, for all actions of the two datasets. Our method even outperforms methods trained with more complex supervision like bounding boxes, *e.g.*, +14% with respect to LP-T [Laptev and Pérez 2007], or human tracks, *e.g.*, +4% and +7% with respect to KMSZ-T [Kläser et al. 2010b]. This shows that modeling the temporal structure of actions is crucial for performance.

3.5.5 Classification-by-localization results

Figure 3.13 contains the per class classification by localization results on the "Hollywood 2" dataset. As mentioned previously, the BOF baselines are using the same sliding window approach as in the previous localization results.

On average over all classes, ASM improves by +11% over both BOF baselines, which perform comparably (BOF T3 only marginally improves by +0.1% with respect to BOF). The improvement yielded by ASM is noticeable on the classes with a clear sequential nature such as "Answer Phone", "Hug Person", or "Sit Down". Interestingly, ASM always improves performance, even when BOF T3 yields worse results than the orderless BOF, *e.g.*, for "Hand Shake" and "Stand Up". Once again, these results show that a *flexible model of the temporal structure* is required in order to recognize real-world actions.

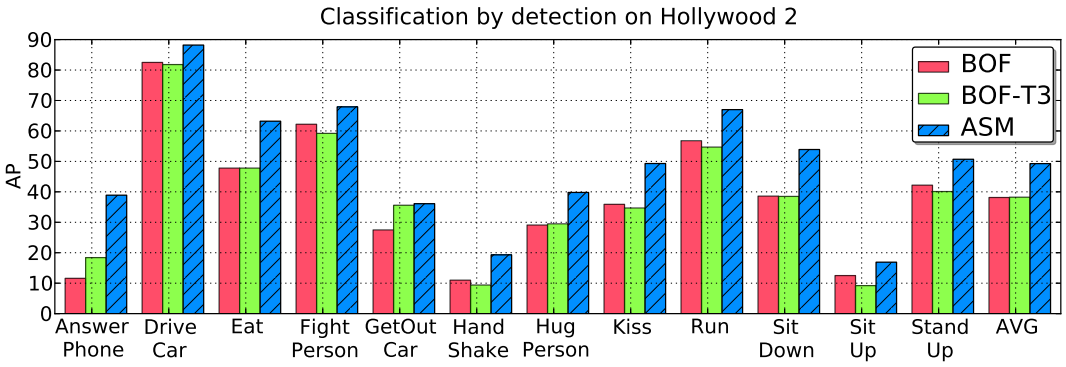


Figure 3.13: *Classification by localization results in Average Precision (AP) on the Hollywood 2 dataset [Marszalek et al. 2009]. BOF and BOF-T3 are sliding-window approaches using BOF and its temporally structured extension. Our approach is ASM. AVG contains the average performance over all classes (BOF:* 38%*, BOF-T3:* 38%*, ASM:* 49%*).*

We also evaluate baseline classification methods similar to [Laptev et al. 2008], where a single model is computed over the entire duration of each test video. On average over all classes, we obtained approximately the same results of 45% AP for three different models: BOF, BOF T3, and ASM with uniformly spread actoms and $\rho = p = 75\%$. Note that the similar performance of these three global models shows that the benefits of ASM do not only lie in its use of soft-voting. In comparison, ASM with classification by localization achieves 49% AP. This +4% gain is less significant than for temporal localization, because classification of pre-segmented videos is an easier problem in this benchmark. Indeed, global models use context information, whereas the more local representations used for localization focus only on the action, at training time as well as at test time. As shown by the dataset authors [Marszalek et al. 2009], using context improves action recognition. Our experiments confirm this, as classifying using a global model of the entire video improves by +9% (45%) over the BOF classification by localization results (36%).

3.5.6 Parameter study

We measured the impact of the different components of our approach: (i) the sliding central frame localization method compared to the sliding window technique, (ii) manual actom annotations compared to uniformly spread ones, (iii) the ASM parameters, (iv) the number of candidate temporal structures learned, and (v) the number of training negatives.

Sliding central frame

First, we found that our sliding central frame approach outperforms the sliding window one. Therefore, marginalizing over a generative model of the temporal

structure is preferable to the commonly used scale sampling heuristics. This can be observed in Table 3.3, where we report the localization results using BOF models in conjunction with our sliding central frame approach.

In this case, we adopt the same method as described in Section 3.4.1: a prior on the action duration is learned with the algorithm from Section 3.3.2, and localization is performed by marginalizing over this univariate distribution on temporal extents. In contrast, the sliding window approach also uses multiple scales learned from the training data, but it does not marginalize over a generative model of these scales. Note also that ASM still outperforms BOF baselines with a sliding central frame.

	C&C		DLSBP	
	OV20	OVAA	OV20	OVAA
BOF (s-win)	27.5	5.0	11.0	3.5
BOF (s-cfr)	35.5	21.5	12.5	9.0
BOF T3 (s-win)	32.0	12.0	12.5	5.0
BOF T3 (s-cfr)	37.0	26.5	14.0	9.5
ASM (s-cfr)	51.5	44.5	18.0	15.0

Table 3.3: *Average localization performance of sliding window (s-win) and sliding central frame (s-cfr) on Coffee and Cigarettes (C&C) and DLSBP.*

Manual training actoms

Second, we computed the localization results using our ASM approach with training actoms spread uniformly between the manually annotated temporal boundaries. We observed that localization results are significantly worse than when using manually annotated training actoms. Indeed, ASM with these uniform actoms yields results similar to BOF T3 with the sliding central frame approach. This shows that temporal boundaries do not provide enough information to model the temporal aspects of an action.

ASM parameters

Third, we studied the impact of the ASM parameters on performance. In table 3.4, we report localization results for ASM with learned parameters — *cf.* an example in Figure 3.5 — and for different parameter configurations — corresponding to the four corners in Figure 3.3. These results show that learning action-specific ASM overlap and peakyness parameters yields the most accurate models, resulting in increased localization performance. Note that the learned parameters change from one action to another. For instance, the learned parameters for "Smoking" are $\rho = 25\%$ and $p = 70\%$, denoting

clearly separated actoms, whereas for "Sit Down" we obtain $\rho = 120\%$ and $p = 50\%$, denoting actoms sharing a significant amount of frames.

ASM parameters		C&C		DLSBP	
ρ	p	OV20	OVAA	OV20	OVAA
low	high	40.3	34.5	11.4	9.0
high	low	39.0	30.9	12.5	10.0
high	high	45.5	34.8	15.0	11.2
low	low	49.8	42.8	15.1	11.9
learned		51.5	44.5	18.0	15.0

Table 3.4: *Impact of the ASM parameters: ρ (overlap) and p (peakyness). Average of the localization results on Coffee and Cigarettes (C&C) and DLSBP.*

Temporal model complexity

In addition, we studied the impact of the complexity of the temporal structure model — measured by the support size s of $\hat{\mathcal{D}}$, *cf.* Eq. 3.7 — on the localization performance. This parameter controls a trade-off between the precision of the model and, as we marginalize over this distribution, the computational complexity at test time. We found that $s = 10$ candidate actom structures yields a good compromise for most classes — *cf.* Figure 3.14 for an illustration using the "Smoking" action. On the one hand, if $s < 5$, then the model is too simple and the performance gap between the OV20 and OVAA results is large. On the other hand, if $s > 15$, then results are equivalent to $5 \leq s \leq 15$ but at a higher computational cost. Note that this parameter has the strongest impact on localization performance after the ASM hyper-parameters.

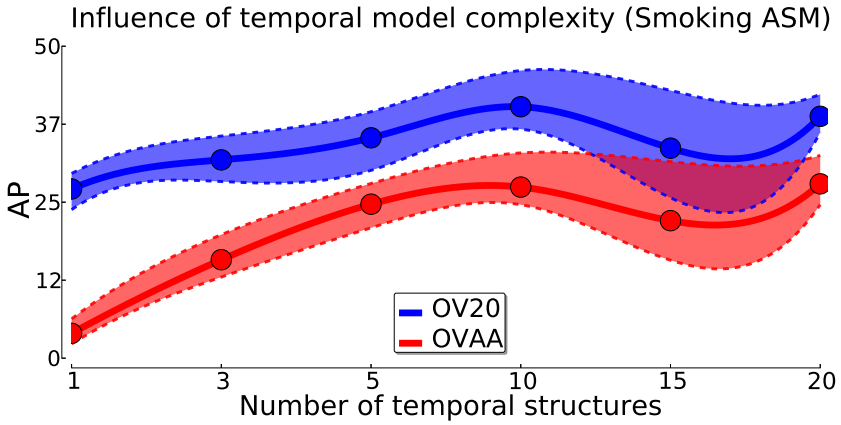


Figure 3.14: *Minimum, average, and maximum localization performance for action "Open Door" for varying sizes of the support of $\hat{\mathcal{D}}$ (number of candidate temporal structures).*

Training negatives

Finally, we measured the localization performance as a function of the number of random training negatives used (*cf.* Figure 3.15 for the "Drinking" action). We found that between one and ten times the number of positives was sufficient to reach satisfactory performance for all classes. We sampled twice more negatives than positives for the Coffee and Cigarettes dataset, and eight times more for the DLSBP dataset. This choice allows to maintain a high true positive over false negative ratio, while limiting the imbalance factor between classes and speeding up localization.

Sampling only few training negatives, however, yields unstable results, as illustrated by the large standard deviation reported in our experiments. This instability can be controlled using a simple bagging approach — *i.e.*, averaging classifiers over different negative training sets as suggested in [Mordelet and Vert 2010] — at the expense of an increase in computational complexity.

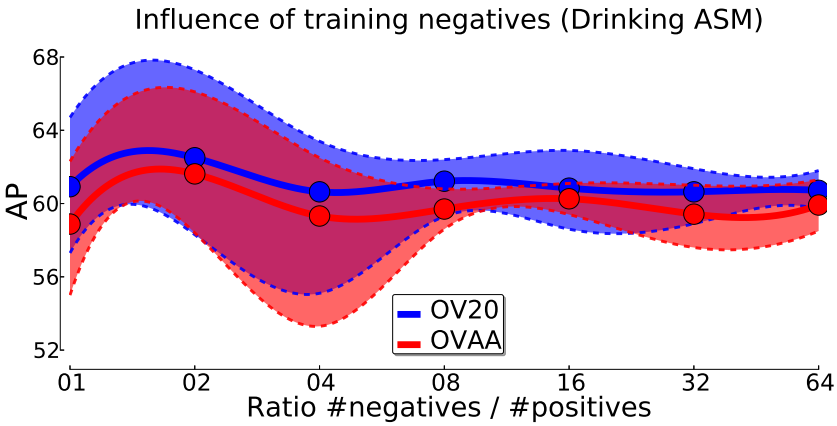


Figure 3.15: *Minimum, average, and maximum localization performance for action "Drinking" for varying numbers of random training negatives. The x-axis is in log-scale.*

3.6 Conclusion

In this chapter, we introduced the *Actom Sequence Model* (ASM). This model describes an action with a temporal sequence of *actoms*, *i.e.*, meaningful temporal parts that are characteristic of the action. It is *discriminative*, as it represents an action by several components instead of one average representation as in the bag-of-features. It is *flexible*, as our temporal representation allows for varying temporal speed of an action as well as interruptions within the action. Experimental results show that our approach outperforms the bag-of-features as well as its extension with a fixed temporal grid. Furthermore, ASM improves over the state of the art, including more sophisticated models using spatial localization.

Chapter 4

Activities as Cluster-trees of Tracklets

Contents

4.1 Introduction

Video content often relates to humans and their actions. Simple actions, *e.g.*, running, rarely convey enough meaning to interpret a scene, but instead form the building blocks of more complex activities, *e.g.*, pole vaulting. In this chapter, we aim at recognizing such *high-level activities*, which are spatio-temporal patterns composed of several related movements of actors, body parts, and objects. Automatically identifying those parts and exploiting both their contents and their relations is a challenging problem that is important for the recognition of complex activities. Therefore, we introduce an unsupervised approach to *hierarchically decompose the complex motion content of an activity*, and an efficient algorithm to compare two *tree-structured videos*.

Our method consists in, first, extracting *dense tracklets* [Wang et al. 2011], which are short-term trajectories of densely sampled points that describe most of the interesting motion contained in a video. We, then, decompose the motion content of a video into a *hierarchy of data-driven parts* by using hierarchical clustering on the set of tracklets. Our main contribution is a *hierarchical divisive clustering* algorithm based on recursive bi-partitioning of an approximate multi-modal spectral embedding of tracklets. We use the resulting structure, called *cluster-tree* [Duda et al. 1995] (*cf.* Figure 4.1), to model a video as an unordered binary tree, called *BOF-tree*, which we represent by nested histograms of local motion features.

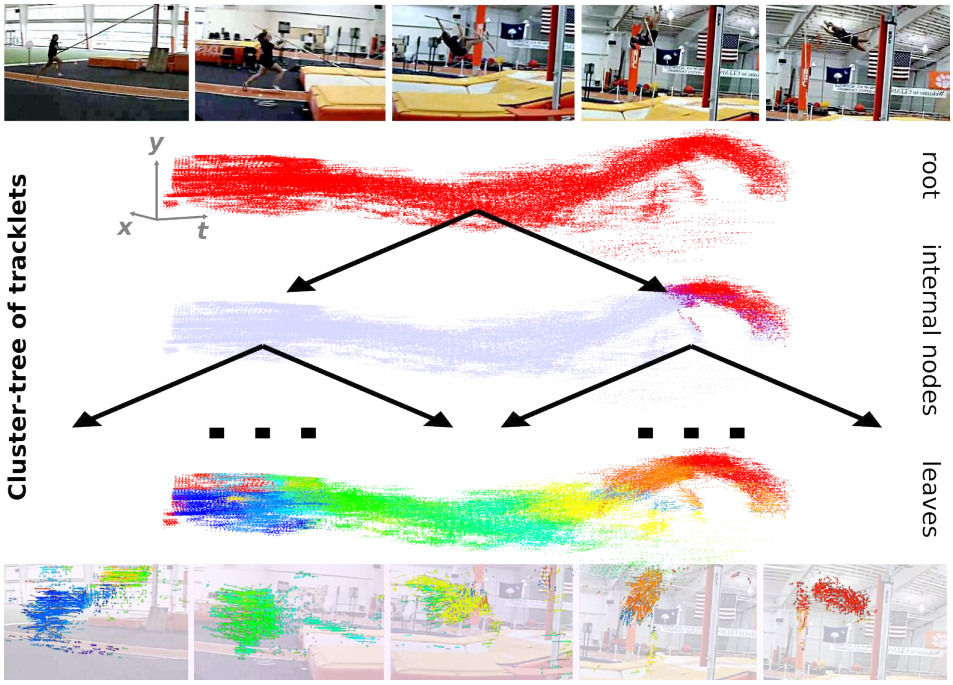


Figure 4.1: *Example of a hierarchical motion decomposition obtained with our recursive bi-partitioning algorithm on dense tracklets.*

Using this structured information presents several challenges. First, BOF-trees have a variable number of nodes (motion components), and a structure that is specific to each video. Second, there is no natural left-to-right ordering of the children of the same parent node. Therefore, we introduce a positive definite kernel on variable-size, unordered binary trees. It consists in efficiently comparing all sub-trees by approximating them through simple edge models, and leveraging the additive structure of BOF-trees. We, then, use this kernel with a Support Vector Machine (SVM) [Schölkopf and Smola 2002] to learn powerful non-linear activity classifiers.

Closest references

Our approach is weakly supervised: it relies neither on part annotations, nor on a predefined action decomposition as in Chapter 3. In addition, we use neither video segmentation techniques [Brendel and Todorovic 2011, Brox and Malik 2010, Grundmann et al. 2008, Lezama et al. 2011], nor pre-trained object detectors [Prest et al. 2012]. We argue that *clusters of tracklets* provide a compromise between, on the one hand, detailed information available from segmentation, and, on the other hand, higher-level attributes, body part localizations, or object detections. Although our tracklets are of a short duration, the clusters obtained with our hierarchical decomposition capture complex and long-term motions of spatio-temporal parts, as well as their relations.

Our method is based on the Nyström approximation for spectral clustering [Fowlkes et al. 2004]. However, in contrast to the k-means algorithm used in [Fowlkes et al. 2004], we do not require the number of parts to be globally fixed and known a priori, and our hierarchical decomposition provides useful structural information relating the motion parts together, instead of unrelated clusters. Furthermore, our *divisive* method can scale to videos with hundreds of thousands of tracklets, whereas bottom-up agglomerative algorithms have a computational complexity that is at least quadratic in the number of points [Fradet et al. 2009, Hastie et al. 2008]. As we deal with tracklets, another difference with existing trajectory clustering approaches [Brox and Malik 2010, Fradet et al. 2009, Lezama et al. 2011] is that we do not restrict trajectory comparisons to their common time span.

Related to our work, Niebles and Fei-Fei [2007] propose to represent an action as a constellation of parts represented by BOFs over shape and motion features. They model a category using a probabilistic mixture of a fixed number of parts. They classify actions in controlled video conditions by maximizing the likelihood with respect to their generative model. Brendel and Todorovic [2011] propose another related generative model. They use a more general graphical model learned from hierarchical video segmentations.

Their approach, however, assumes that all actions of the same category share strong geometrical and temporal part relationships. Furthermore, their video segmentation algorithm — and therefore the resulting action models — can only account for smooth motions of color-consistent parts forming continuous spatio-temporal "tubes". We do not make these restrictive assumptions on parts. Furthermore, we estimate and compare each video's specific structure, and learn a discriminative model instead of matching to a global action template. In our experiments, we show that we outperform their method.

Discriminative alternatives to these generative models are often based on the popular deformable part model of Felzenszwalb et al. [2010]. For instance, Liu et al. [2011] combine manually predefined attributes with data-driven ones obtained by clustering local features. They use a latent SVM [Felzenszwalb et al. 2010] to learn the importance of each part. Wang and Mori [2011] use tracking and a Hidden Conditional Random Field (HCRF) to learn a discriminative model of latent parts for frame-by-frame recognition. Closest to our work, Raptis et al. [2012] extract clusters of long-term trajectories and learn a latent model over a fixed number of parts. Their approach has a cubic time complexity in the number of trajectories, relies on bounding box annotations, and uses only a fixed small subset of clusters for all videos. Furthermore, they explicitly model pairwise relationships between clusters using the trajectory, whereas we use the full hierarchical structure resulting from our clustering. In contrast to all the aforementioned discriminative approaches, we do not assume that actions share a fixed number of parts common to all training instances. Instead, each video has its own decomposition structure, and all parts — including their relationships — are used in our video comparisons. In addition, as we do not rely on latent parts, we do not need to solve a complex inference problem for each test video.

We also differ from methods using global hierarchies over local features [Mikolajczyk and Uemura 2008, Reddy et al. 2009] or shape-motion prototypes [Jiang et al. 2012]. These methods use global tree structures to speed up the computation of a matching score, where each feature casts a vote for an action category. In our approach, we account for all local features jointly, and leverage instance-level relationships without explicitly modeling neighborhoods and co-occurrences over the whole dataset [Gilbert et al. 2010, Kovashka and Grauman 2010, Matikainen et al. 2010].

Outline

Section 4.2 describes the tracklets, and introduces our hierarchical divisive multi-modal clustering scheme. Section 4.3 details our second contribution: our hierarchical model for the motion content of a video, the BOF-Tree, and our kernel for such BOF-Trees. In Section 4.4, we evaluate our method on the complex activities from the "Olympic Sports" dataset [Niebles et al. 2010] and on the human-human interactions from the "High Five" dataset [Patron-Perez et al. 2010]. We show that our approach outperforms BOF-based baselines, decompositions obtained with other clustering algorithms, and the state of the art. We also show that our approach can be successfully applied on a large set of simple actions, the HMDB dataset [Kuehne et al. 2011], although the performance gains are smaller.

4.2 Clustering dense tracklets

In this section, we first describe how we extract dense local point trajectories, called *tracklets* (*cf.* Figure 4.2) using dense optical flow fields. We then address the problem of efficiently and accurately clustering a video composed of a large number of tracklets in order to obtain a hierarchical decomposition of its motion components. We describe the two steps of our clustering algorithm: a non-linear projection of the tracklets on a multi-modal spectral embedding using the Nyström approximation, followed by a hierarchical divisive clustering algorithm. Note that the algorithm described in this section is applied to a single video at a time and does not require any other external data, such as other videos containing the same action.

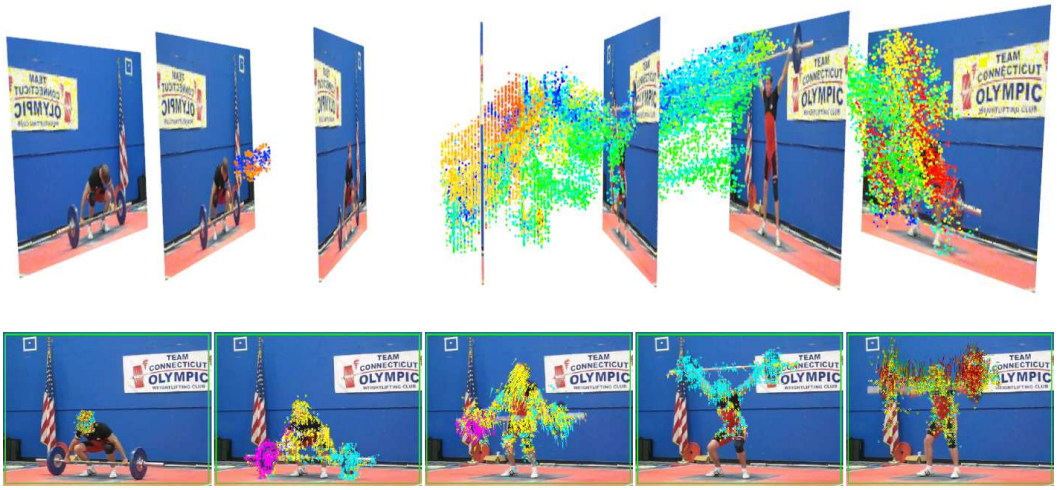


Figure 4.2: *Tracklets of a weightlifting activity. Colors correspond to the most detailed clusters obtained with our hierarchical motion decomposition.*

4.2.1 Extracting dense tracklets

In order to track densely sampled points, we follow the approach described in [Wang et al. 2011]. First, we compute the dense optical flow field of the video using Farnebäck's algorithm [Farnebäck 2003] as implemented in the OpenCV library [Bradski and Kaehler 2008]. This iterative multi-scale approach is based on approximating the neighborhood of pixels by quadratic polynomials using polynomial expansion.

Second, we sample the pixels to be tracked on a dense spatial grid. Similar to [Wang et al. 2011], we observed that a sampling step-size of 5 pixels along both the x-axis and the y-axis is a good compromise between density and speed given the dimensions of our videos — between 50 and 1000 frames for resolutions close to 640×480. As points in homogeneous regions cannot be reliably tracked, we filter them out following the criterion of Shi and Tomasi [1994]: if the smallest eigenvalue of the autocorrelation matrix of a point is lower than a threshold, it is removed from the set of points to be tracked.

Third, we use a multi-scale video representation over 8 spatial scales, spaced by a factor $1/\sqrt{2}$. We do not use multiple temporal scales — *i.e.*, we track points from frame t to $t+1$ — as we are only interested in *local* point trajectories. Each point $P_t = (x_t, y_t)$ at frame t is robustly tracked in each scale by median filtering in the dense optical flow field $w = (u_t, v_t)$:

$$P_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (M * w)|_{\bar{x}_t, \bar{y}_t}, \tag{4.1}$$

where M is the median filtering kernel and $(\bar{x}_t, \bar{y}_t)$ is the rounded position of (x_t, y_t). Note that this interpolation step is efficient once the dense optical flow field is extracted. This approach, therefore, allows to efficiently track points on a dense spatial grid.

The new frame $t+1$ may then contain good regions void of points to be subsequently tracked. Indeed, points can be lost and new ones can appear due to a change in the scene. Therefore, and in order to maintain density, we fill these regions by adding new points re-sampled on the dense spatial grid.

Point trajectories interpolated from an optical flow field are often deviating from the underlying tracked pixel after a certain number of frames. In order to limit this drifting problem, Wang et al. [2011] propose to track points only across a fixed small number of frames L. In their experiments, they show that using tracklets of duration $L = 15$ frames can yield state-of-the-art action recognition results. We observed that, in our case, shorter tracklets yields better results and use $L = 5$ frames in our experiments. Indeed, when dealing with fast motions, the displacements of the underlying pixels are large, motion blur causes point tracks to drift faster, and self-occlusions make it often impossible to track interesting body parts for more than a few frames. This

is especially frequent in sports videos, *e.g.*, when an athlete spins on himself before throwing a discus. Furthermore, although such short duration tracklets convey less information, *clusters* of tracklets can encode mid-level information on par with variable-length longer-term trajectories. Finally, errors in long-term trajectories due to drift cannot be easily recovered from at later stages. On the contrary, errors in short-term tracklets amount to noise that can be appropriately handled via some simple post-processing steps, as described in the following paragraphs.

First, tracklets with static trajectories are deleted. They indeed convey no motion information and are often associated either with a static textured background (*e.g.*, trees) or with noise. We also handle the other extremal case by removing tracklets containing large and sudden displacements.

Finally, we filter out tracklets that are in low-density regions. In more details, isolated but reliably tracked points — *e.g.*, due to block-like compression artifacts — are considered as outliers and removed. We use a simple sliding window outlier estimation technique that has proven effective in our situation. For each tracklet $\mathbf{P} = (P_t)_{t=F-L+1,\cdots,F}$ ending at frame F, we first estimate its approximate spatial k-nearest neighbors $\mathcal{N}_{k,r}(\mathbf{P})$ *restricted* to tracklets ending between frames $F - r$ and $F + r$. In our experiments, we used $k = 30$ neighbors, $r = 5$ frames, and the spatial distance between tracklets is estimated between their respective average positions $\bar{P} = (\bar{x}, \bar{y}) = \frac{1}{L}\sum_{t=1}^{L} P_t$. We then compute the local sparsity $s_{\mathbf{P}}$ of a tracklet as the mean spatial distance to its neighbors in the temporal window $[F - r, F + r]$:

$$s_{\mathbf{P}} = \frac{1}{k} \sum_{\mathbf{P}' \in \mathcal{N}_{k,r}(\mathbf{P})} \|\bar{P} - \bar{P}'\| \tag{4.2}$$

We consider a tracklet with local sparsity $s_{\mathbf{P}}$ as an outlier if $s_{\mathbf{P}} > \bar{s} + \bar{\sigma}_s$, where $(\bar{s}, \bar{\sigma}_s)$ is the mean and standard deviation of the spatial distance to the approximate spatial k-nearest neighbors of all tracklets in the entire video, irrespective of temporal position. The approximate k-nearest neighbors are computed *via* kd-trees, which are efficiently built in the 2-D space of average spatial positions of tracklets.

4.2.2 Tracklet descriptors for intra-video clustering

Once the tracklets have been extracted, we model them using multiple features describing both their spatio-temporal position and shape. We use the following descriptors to represent trajectory information:

- $\mathbf{x} = (x_1, \cdots, x_L)$, x positions over time,
- $\mathbf{y} = (y_1, \cdots, y_L)$, y positions over time,

- $\mathbf{z} = (t, \cdots, t+L-1)$, temporal positions,
- $\mathbf{v_x} = (x_{k+1} - x_k)_{k=1:L-1}$, velocities along x-axis,
- $\mathbf{v_y} = (y_{k+1} - y_k)_{k=1:L-1}$, velocities along y-axis.

Note that as our goal is to cluster tracklets to decompose the motion components *inside* a video, our descriptors need not be particularly robust as all comparisons are *intra-video* during this clustering stage. We also separate the trajectory information along the different spatio-temporal dimensions in order to allow for dimension-specific normalizations in the later stages of our algorithm. Another noteworthy point is the seemingly redundant use of both trajectory and velocity descriptors. This redundancy is important in practice, because velocity information tends to be unstable as taking derivatives along the trajectory amplifies high frequency noise that results from small inaccuracies of the point tracking algorithm.

Our approach is, however, not specific to the set of features chosen to represent a tracklet. The only pre-requisite is the availability of a similarity function specific to each feature channel. We use Gaussian RBF kernels $k(f, f') = exp(-\gamma d(f, f')^2)$, where the distance $d(f, f')$ is the Euclidean distance. The γ parameter of each kernel is automatically fixed to $\gamma = 1/(2\bar{d})$, where $\bar{d}$ is an estimate of the median of the distances between the corresponding tracklet features. This normalization ensures that all kernels are comparable across the different feature channels.

4.2.3 Multi-modal spectral embedding of tracklets

Once the features are extracted, the first step of our clustering algorithm is to project tracklets onto a low-dimensional space. This embedding relies on spectral properties of a similarity matrix between tracklets.

4.2.3.1 Similarity between tracklets

In our case, as we have multiple features representing different useful characteristics of tracklets, we use as similarity the *product of the per-feature similarities*. As each feature-specific similarity is positive-definite, the product similarity is also a positive-definite kernel. It corresponds to a feature space that is the tensor product of the feature spaces induced by the individual kernels. Using this product kernel has the advantage of entailing an "and" effect, *i.e.*, tracklets close according to this similarity are close with respect to *all* the features used. This behavior is desirable as all our feature spaces convey meaningful information representing our priors on tracklets, such as "similar tracklets should be close *and* have similar shapes".

In contrast, using an "early fusion" approach — *i.e.*, concatenating the descriptors into a single vector or summing the per-feature kernels — has an "or" effect: tracklets are close according to *at least one* feature. In addition, early fusion mixes heterogeneous descriptors with various scales into a long vector, for which Euclidean distances are not adequate. Therefore, central grouping methods like k-means are not adapted to our set-up, as they search for clusters with a Gaussian distribution in the Cartesian product of the input spaces. These shortcomings are addressed by resorting to a spectral clustering method, which operates on an embedding of the data points.

4.2.3.2 Spectral embedding

The spectral embedding of our tracklets is based on projections onto the leading eigenvectors of the graph Laplacian corresponding to a similarity matrix between tracklets. Let $W \in \mathbb{R}^{N \times N}$ be a symmetric similarity matrix between N tracklets. This matrix can be viewed as the weighted adjacency matrix of a graph, where the nodes are the tracklets and the edges are weighted by the pairwise affinity between tracklets. The normalized Laplacian of this graph is defined as $\mathcal{L} = I - D^{-1/2} W D^{-1/2}$, where D is the diagonal matrix of row-sums of W. Thresholding the eigenvector of $\mathcal{L}$ with the second smallest eigenvalue yields an approximate solution to the NP-Hard Normalized Cut (NCut) bi-partitioning problem [Shi and Malik 2000]. Note that the leading eigenvalue-eigenvector pair of $\mathcal{L}$ is $(0, D^{1/2}\mathbf{1})$.

In order to obtain a predetermined number of clusters, one can either apply this technique recursively on the next leading eigenvectors, or use a distortion-minimization clustering algorithm — *e.g.*, k-means — on an embedding consisting of the projections on multiple leading eigenvectors.

For N data points, the N_E-dimensional spectral embedding ($N_E \ll N$) is obtained by computing the $N \times (N_E + 1)$ matrix V of the $N_E + 1$ leading eigenvectors and the $(N_E + 1) \times (N_E + 1)$ diagonal matrix Λ of eigenvalues of the system:

$$\left(D^{-1/2} W D^{-1/2}\right) V = V \Lambda. \tag{4.3}$$

The j^{th} embedding coordinate of the i^{th} tracklet is given by:

$$E_{i,j} = \frac{V_{i,j+1}}{\sqrt{D_{i,i}}}, \quad i = 1, \cdots, N, \ j = 1, \cdots, N_E, \tag{4.4}$$

where the eigenvectors are sorted by ascending eigenvalue. Thus, each tracklet is associated with a row of E and a clustering algorithm can be applied on the rows to partition the data.

However, the spectral embedding cannot be directly computed this way in our case. Indeed, for N tracklets, solving equation 4.3 requires the computation and storage of a $N \times N$ similarity matrix. In addition, its computational complexity is in $O(N^3)$ which seems to prohibit its use in our setup, where the number of data points can go up to 10^6.

A mechanism to overcome this complexity is to use a sparse affinity matrix — *e.g.*, by thresholding its entries — and efficient sparse eigensolvers such as the Lanczos method [Shi and Malik 1998]. With our dense tracklets, the affinity matrix is not sparse and thresholding its entries still requires the evaluation of all similarities. Thresholding has also side-effects that are not well understood [Fowlkes et al. 2004]. For instance, thresholding a kernel matrix is known to not preserve its positive-definiteness. Therefore, we choose to use the Nyström method in order to efficiently compute an approximate spectral embedding.

4.2.3.3 Nyström approximation

The Nyström method [Nyström 1930] is a technique for finding numerical approximations to eigenfunction problems and has been introduced in the context of spectral clustering in [Fowlkes et al. 2004]. It allows to extend an eigenvector computed for a subset of points to any arbitrary point, while requiring only a subset of the columns (or rows) of the similarity matrix W (see Figure 4.3 for an illustration). As observed by Williams and Seeger [2001], it is related to kernel PCA [Schölkopf et al. 1998].

Let A be the $n \times n$ similarity matrix between a subset of n randomly sampled tracklets, with $n \ll N$, and let B be the $n \times (N - n)$ similarity matrix between this subset and all other tracklets. With no loss of generality, we assume the samples sorted such that we can rewrite the full $N \times N$ similarity matrix W as:

$$W = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \tag{4.5}$$

with $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times (N-n)}$ and $C \in \mathbb{R}^{(N-n) \times (N-n)}$, C being in general too expensive to compute as $N \gg n$. Note that in our case W and A are positive-definite.

Let $A = U_A \Lambda_A U_A^T$ be the eigendecomposition of A. The matrix form of the Nyström extension is $B^T U_A \Lambda_A^{-1}$. The approximate eigenvectors $\hat{U}$ of W given by the Nyström method are:

$$\hat{U} = \begin{bmatrix} U_A \\ B^T U_A \Lambda_A^{-1} \end{bmatrix} \tag{4.6}$$

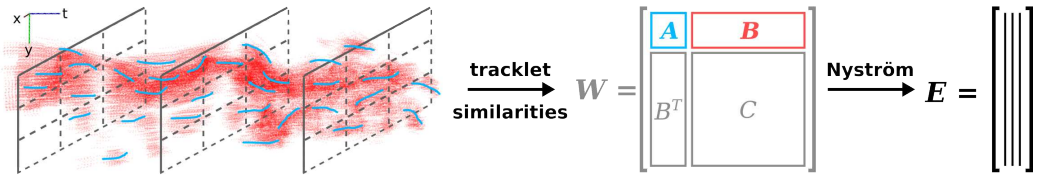


Figure 4.3: *Illustration of the Nyström approximation. We first sample $n \ll N$ tracklets by randomly choosing p points in each cell of a regular spatio-temporal grid. We then compute the similarity between the selected tracklets (in blue), yielding matrix A, as well as with all other $N - n$ points (in red), yielding matrix B. The Nyström method allows to efficiently compute the approximate leading eigenvectors $\hat{U}$ of the large $N \times N$ matrix W of all similarities using only the small $n \times N$ slice of it, composed of blocks A and B.*

Note that $\hat{U}$ are also the eigenvectors of the approximation $\hat{W}$ of W:

$$\hat{W} = \hat{U}\Lambda_A\hat{U}^T = \begin{bmatrix} A & B \\ B^T & B^TA^{-1}B \end{bmatrix} \tag{4.7}$$

It shows that the Nyström method implicitly approximates C using $B^TA^{-1}B$. Thus, the quality of the approximation can be measured by the norm of the Schur complement $\|C - B^TA^{-1}B\|$, which measures how well C is spanned by the rows of B. Consequently, the subset of rows of W must be chosen such that the corresponding tracklets span all underlying "true" clusters to be discovered. Therefore, to ensure good diversity and coverage, we randomly subsample p tracklets per cell of a spatio-temporal grid over the whole video (*cf.* Figure 4.3). The grid granularity and p can be adapted dynamically to the total number of tracklets and the available computational resources. Similarly to Fowlkes et al. [2004], we observed that good results can be obtained with less than 1% of the total number of points.

As observed in [Fowlkes et al. 2004], the columns of $\hat{U}$ are not orthogonal. However, as A is positive-definite in our case, we can apply the "one-shot" technique of Fowlkes et al. [2004] and compute the orthogonalized approximate eigenvectors of W in one step as follows. Let $S = A + A^{-1/2}BB^TA^{-1/2}$, where $A^{-1/2}$ is a pseudo-inverse of a symmetric positive definite square root of A. Let $S = U_S\Lambda_SU_S^T$ be the eigendecomposition of S. As shown in [Fowlkes et al. 2004], $\hat{W}$ is diagonalized by $\hat{V}$ and Λ_S, where:

$$\hat{V} = \begin{bmatrix} A & B \end{bmatrix}^T A^{-1/2}U_S\Lambda_S^{-1/2} \tag{4.8}$$

Note that in the context of NCut, A and B must be normalized by the row-sums $\hat{d}$ of $\hat{W}$ beforehand. This can be done without computing the expensive $B^T A^{-1} B$ block of $\hat{W}$ by:

$$A_{i,j} \leftarrow \frac{A_{i,j}}{\sqrt{\hat{d}_i \hat{d}_j}}, \quad B_{k,l} \leftarrow \frac{B_{k,l}}{\sqrt{\hat{d}_k \hat{d}_{l+n}}}, \qquad (4.9)$$
$$i, j, k = 1, \cdots, n, \quad l = 1, \cdots, N - n$$

where

$$\hat{d} = \hat{W} \mathbf{1}_N = \begin{bmatrix} a_r + b_r \\ b_c + B^T A^{-1} b_r \end{bmatrix} \qquad (4.10)$$

with $\mathbf{1}_N$ the column vector of N ones, $a_r, b_r \in \mathbb{R}^n$ the row-sums of A and B, and $b_c \in \mathbb{R}^{N-n}$ the column-sums of B. After normalizing A and B (eq. 4.9, 4.10) and computing $\hat{V}$ (eq. 4.8), the spectral embedding E is:

$$E_{i,j} = \frac{\hat{V}_{i,j+1}}{\hat{V}_{i,1}}, \quad i = 1, \cdots, N, \; j = 1, \cdots, N_E, \qquad (4.11)$$

As we are in a large scale, realistic set-up, special care must be taken with respect to numerical stability. Indeed, even though A is positive-definite, it is in general ill-conditioned. Therefore, we use a robust pseudo-inverse algorithm to compute A^{-1} by thresholding the lower-end of its spectrum. We also follow some useful guidelines from [Foster et al. 2009]. Note that the overall computational cost of this approximate spectral embedding is in $O(n^2 N)$ time and $O(nN)$ space, which is a large improvement over the $O(N^3)$ time, $O(N^2)$ space complexity of the exact method.

4.2.4 Hierarchical divisive clustering

Using the computed spectral embedding, we cluster tracklets via an efficient hierarchical divisive algorithm. The pseudo-code of our approach is given in algorithms 1 and 2. We adopt a top-down approach that consists in recursively bi-partitioning the set of tracklets (*cf.* Figure 4.1). We split a set of tracklets in two by thresholding along an eigenvector, *i.e.*, along a dimension of the spectral embedding. As the second smallest eigenvector is the real-valued solution to the Normalized-Cut (NCut) problem, it is composed of two clearly separated ranges of values that indicate the optimal partition of the tracklets. This argument is also valid for the next leading eigenvectors and, theoretically, one can optimally sub-partition the data by recursively thresholding one eigenvector after the other. However, several practical problems need to be taken into account while following such an approach.

Algorithm 1 spectral_division

Input: features of N tracklets (*cf.* section 4.2.2)
Output: the cluster-tree (hierarchical set of nodes)
Compute the spectral embedding
Compute the slice $[A \quad B]$ of W (*cf.* fig. 4.3)
Normalize A and B (eq. 4.9, 4.10)
Compute the embedding E (eq. 4.8, 4.11)
Initialize priority queue over nodes to be split
to_split ← empty priority queue of nodes
Push the root of the cluster-tree on *to_split*
# Recursively split nodes according to priority
while *to_split* is not empty **do**
  node ← pop highest priority node from *to_split*
  $left$, $right$ = **find_best_split**(*node*) # *cf.* algorithm 2
  # Add children to the queue
  if $left$ and $right$ are not empty **then**
    Push left on *to_split*
    Push right on *to_split*
  end if
end while

First, the optimal bi-partition is often unclear in realistic conditions and the eigenvectors tend to reflect this ambiguity by having smooth variations. Therefore, which threshold to use is in general unclear when looking only at eigenvector values. Furthermore, thresholding a smoothly varying eigenvector can lead to an unstable partition, as a small change in the threshold significantly modifies the partition.

The first technique we use to address this stability problem is based on a criterion similar to the one mentioned in [Shi and Malik 2000]. If the variations of an eigenvector are below a small threshold, then we do not attempt to split along this eigenvector. We used 10^{-10} as threshold in our experiments. More importantly, we propose to circumvent the difficulties posed by smooth eigenvectors by finding the threshold that maximizes a model selection score based on spatio-temporal information (*cf.* Algorithm 2). We experimented with multiple spatio-temporal consistency criteria, such as inertia or label agreement amongst neighbors, but the one that yielded the best results in our experiments is the *connectedness* measure described in the following.

Algorithm 2 find_best_split

Inputs: *node*: set of tracklets, E: embedding
Parameters: m & M: min & max leaf sizes
Output: $left$, $right$: nodes of best split found
# Determine if we cannot split further
if $|node| \leq 2m$ **then**
  return $\emptyset, \emptyset$ # node is a leaf
end if
# Determine score to improve upon
if $|node| \leq M$ **then**
  $best\_score \leftarrow score(node)$
else
  $best\_score \leftarrow -\infty$ # force the split
end if
# Greedily search for the best split
$left,\ right \leftarrow \emptyset,\ \emptyset$
for $j = 1$ **to** N_E **do**
  for e in candidate thresholds **do**
    $score\_l \leftarrow connectedness(\{i \in node\ ;\ E_{i,j} < e\})$
    $score\_r \leftarrow connectedness(\{i \in node\ ;\ E_{i,j} \geq e\})$
    $split\_score \leftarrow min(score\_l, score\_r)$
    if $split\_score > best\_score$ **then**
      $best\_score \leftarrow split\_score$ # best split so far
      $left\ \leftarrow \{i \in node\ ;\ E_{i,j} < e\}$
      $right \leftarrow \{i \in node\ :\ E_{i,j} \geq e\}$
    end if
  end for
  # Check the stopping criterion
  if an improving split was found **then**
    if $j > 1$ **or** split was not forced **then**
      return $left, right$ # the best split found
    end if
  end if
end for
return $left, right$

Connectedness score of a node

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a spatio-temporal k-nearest neighbor graph over the tracklets $\mathcal{V}$ in a video: $(v_i, v_j) \in \mathcal{E}$ if and only if $v_i, v_j \in \mathcal{V}$ are k-nearest neighbors according to their spatio-temporal positions. We define the connectedness score $c(\mathcal{V}')$ of a subset of tracklets $\mathcal{V}' \subset \mathcal{V}$ forming a node in the cluster-tree as: $c(\mathcal{V}') = |\mathcal{C}(\mathcal{V}')|^{-1}$, where $\mathcal{C}(\mathcal{V}')$ is the set of connected components of the subgraph $\mathcal{G}(\mathcal{V}')$ of $\mathcal{G}$.

Maximizing this connectedness score has a strong advantage over the minimization of distortion-based measures such as inertia: it does not constrain the shape of the clusters. Furthermore, it does not enforce "tube-like" clusters, but only *encourages* spatio-temporal contiguity of the tracklets in order to avoid clearly spatio-temporally disjoint parts in the same node. Connectedness is also not biased with respect to node size. In addition, it can be efficiently obtained along the course of our divisive algorithm by pre-computing once — during an initialization step — the full spatio-temporal adjacency matrix of all tracklets in the video. We compute this sparse matrix using *kd-trees* on the average spatio-temporal position of tracklets to determine the graph of approximate k-nearest neighbors. As this is fast to compute, including for hundreds of thousands of tracklets, we select the smallest k such that the entire video has exactly one spatio-temporally connected component. This yields approximatively $k = 10$ neighbors on average over all videos. Once the neighborhood graph is built, computing the number of connected components of a subgraph is done in linear time by a depth-first search.

This connectedness score is specific to a node in the cluster-tree, and when deciding whether a split improves performance, we compare the score of the parent node with the lowest score of its children resulting from a candidate split (*cf.* Algorithm 2). The score of a node is also employed to determine the order used to choose the nodes to split. We use a priority queue to split nodes with the *lowest* score first, as they are those with the maximum expected gain. As outlined in Algorithm 1, at each iteration, we retrieve the highest priority (lowest score) node, attempt to split it, and, if it is not a leaf, we push its children to the priority queue. We observed that, in general, this choice of priority leads to a depth-first construction of the cluster-tree, with occasional jumps to higher nodes (*i.e.*, backtracking). Ties in priority are handled by picking the largest node, then the node closer to the root.

Note that we do not use the NCut value as score for two reasons. First, as we compute an approximate spectral embedding using the Nyström method, we do not have access to the full similarity matrix W, and computing the NCut value would therefore be prohibitively expensive. Second, smooth variations in an eigenvector yields smooth variations in the NCut value, which, consequently, cannot be used reliably to find a good split.

The second issue faced when adopting a recursive thresholding approach, is that there is an accumulation of approximation errors that makes higher eigenvectors less reliable. We, therefore, use this natural ordering of the eigenvectors in order to find a good split. We adopt a greedy strategy depicted in Algorithm 2. We first try candidate thresholds along the leading eigenvector. In our experiments, we use nine adaptive thresholds corresponding to the 10^{th}, 20^{th}, ..., 90^{th} percentile of the considered eigenvector values, in order to avoid severe imbalance between nodes at the same depth in the cluster-tree. If the best split along this eigenvector improves with respect to the score of the parent node that we try to split, then the split is registered with this eigenvector-threshold pair. Otherwise, we iteratively try to split along the next leading eigenvector, until we reached the last one. We observed that less than 20 eigenvectors are generally used. For each node, we start again from the leading eigenvector, as suggested in [Shi and Malik 2000], because it is the most reliable one, and we observed that, in practice, the leading eigenvectors might be decomposed over several approximately flat plateaus.

The only hyper-parameters of our divisive algorithm are the minimum and maximum leaf sizes: m and M. They are employed in order to avoid "degenerate" — *i.e.*, too large or too small — leaves, a frequent problem faced by central grouping methods such as k-means. If a node to be split contains less than $2m$ tracklets, we automatically mark it as a leaf. If a node is larger than M, we force the split by using the best eigenvector-threshold pair, even if the scores of the children nodes are lower than the score of the parent (*cf.* Algorithm 2). We used $m = 200$ and $M = 2000$ in our experiments.

4.3 Classification of cluster-trees

In this section, we first explain how we represent each node in the cluster-tree as a histogram of quantized tracklet features. We then introduce a tree kernel to efficiently compare cluster-trees for activity recognition.

4.3.1 Tracklet descriptors for classification

As described in Section 4.2.2, we use simple discriminative tracklet descriptors in the video-specific clustering stage, because all tracklet comparisons remain inside the same video. In the classification stage of our method, however, we compare the content of different videos. Therefore, recognition requires more robust features in order to build models that can handle the large intra-class variability of activities in real-world videos. Consequently, we choose to represent a tracklet using Motion Boundary Histograms (MBH) [Dalal et al. 2006,

Wang et al. 2011]. It consists of two histograms quantifying the gradients of the horizontal and vertical components of the optical flow. As shown in [Wang et al. 2011] for the bag-of-features model, using MBH to describe tracklets allows for better action recognition than when using trajectory information. As MBH relies on the derivatives of the optical flow, it suppresses constant motion information and focuses on representing motion boundaries, *i.e.*, local changes of orientation in the motion field. Therefore, it can be robust to a certain amount of camera translation, but not to more complex movements such as rotations. In addition, MBH allows to robustly handle the noise in the spatial derivatives of the optical flow *via* quantization.

Note that we found that MBH is not adapted as a clustering feature for over-segmentation. Its robustness, indeed, can yield large clusters containing multiple distinct objects moving in the same manner, *e.g.*, soccer players running on a soccer field, which, in spite of their different spatio-temporal locations, cannot be separated according to MBH features.

Contrary to the clustering stage, we only use one descriptor here. Results presented in [Wang et al. 2011] have shown that further moderate classification improvements can be achieved by combining MBH with trajectory, HOG, and HOF information, at the expense of significantly increased computational cost. We use the same parameters as in [Wang et al. 2011] to efficiently compute tracklet-aligned MBH descriptors directly from the dense optical flow field used to obtain the tracklets.

4.3.2 BOF-Tree: tree of nested bag-of-features

As a node is a motion component composed of a diverse set of tracklets, we propose to efficiently encode this diversity with a bag-of-features (BOF). Note that we could also represent a node's content with the average of its tracklets. However, this has several limitations in our situation. First, there is a potentially large number of points per node, which makes a node qualitatively different from a "super-tracklet", especially at a low depth in the cluster-tree. Second, as our clustering algorithm uses different features than the ones used for classification, there is, in general, a high variability of tracklet descriptors in a node, *i.e.*, a large dispersion around the centroid. In addition, the spatio-temporal consistency of the nodes is not strictly enforced. Thus, the global shape of our nodes may not be discriminative, due to holes and noise caused by our approximate clustering algorithm. In contrast, the BOF representation has proven to be a reliable way to describe entire videos as well as smaller temporal action units [Gaidon et al. 2011a]. We adopt the common pipeline described in the following.

From the set of MBH descriptors and the cluster-tree obtained by our spectral divisive algorithm, we extract a hierarchical representation of a video called *BOF-tree*. Its structure is the same as its corresponding cluster-tree. In addition, each node in the BOF-tree is modeled by a bag-of-features. See Figure 4.4 for an illustration. We first pre-compute a vocabulary of track-aligned MBH features on a random subset of the training features. We use an on-line k-means algorithm [Sculley 2010] with $k = 4000$ to cluster 10^6 tracklets randomly sampled from the training videos. We then quantify all MBH features by assigning them to the closest "visual word" (centroid) in the learned vocabulary. Finally, each node is represented by a BOF, *i.e.*, its histogram of occurrences of visual words. Although we use a high-dimensional representation for better accuracy, this does not pose computational or memory problems. In practice, per-node histograms are indeed sparse, thus can be represented efficiently. Modeling a node with a BOF discards the "intermediate geometry" at the node level — *i.e.*, between neighboring tracklets — while local geometry is still captured by the local tracklet features, and global spatio-temporal structure information is captured by the cluster-tree.

As a consequence of the clustering, the left-to-right order in the BOF-tree does not have any geometrical interpretation (the cluster-tree is *unordered*). The only semantic relation directly captured by our tree structure is the inclusion relation derived from the cluster-tree: the two children of a node correspond to a bi-partition of the tracklets of this parent node. This induces an additive property on the nodes of a BOF-tree: the BOF of a node in a BOF-tree is the sum of its children's BOFs (*cf.* Figure 4.4). We now show how to use this property to efficiently compare BOF-trees, while leveraging the hierarchical structure information between nodes.

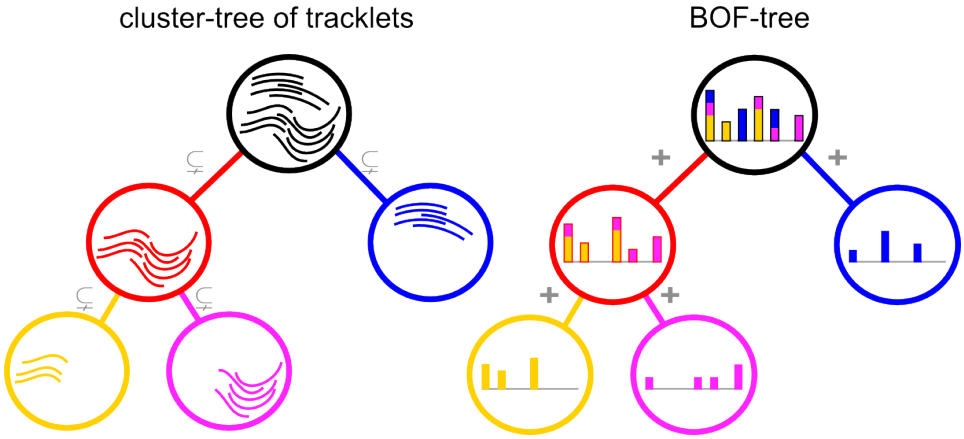


Figure 4.4: *(a) A cluster-tree of tracklets, where edges between nodes represent a strict inclusion. (b) Its corresponding BOF-tree, where the BOF of a node is the sum of its children's BOFs.*

4.3.3 Kernel between BOF-trees

On the one hand, existing kernels on variable-size trees are inspired by string kernels, *i.e.*, measure structural similarity by counting the number of common sub-structures [Shawe-Taylor and Cristianini 2004]. In BOF-trees, however, these approaches are not directly applicable as siblings are unordered. In addition, our goal is to use the structure information to disambiguate comparisons between the *content* of the motion component hierarchies, not to directly compare the tree structures. On the other hand, summing over all pairwise node comparisons results in a valid kernel comparing the contents of BOF-Trees (which we use as a baseline, *cf.* Section 4.4.3), but it ignores their structure. Instead, we propose the "All Tree Edge Pairs" (ATEP) kernel, which consists in comparing the *edges* of BOF-trees.

Let $\mathcal{T}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{T}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ be two BOF-trees, defined from their set of vertices (nodes) $\mathcal{V}_i$ and directed edges (parent-child relations) $\mathcal{E}_i$. Each node $v \in \mathcal{V}_i$ is represented by a BOF, noted $b[v]$, over its constitutive tracklets. We model a directed edge $e = (v_p, v_c) \in \mathcal{E}_i$ by the *concatenation* $b[e] = (b[v_p], b[v_c])$ of the BOF of the child node v_c with the BOF of its parent node v_p. Let h be a kernel between BOF. We use the intersection kernel [Maji et al. 2008] between L_1-normalized histograms:

$$h(x, x') = \sum_j \min\left(\frac{x_j}{\|x\|_1}, \frac{x'_j}{\|x'\|_1}\right) \tag{4.12}$$

Let $r_i \in \mathcal{V}_i$ be the root of $\mathcal{T}_i$, $i \in \{1, 2\}$. Our ATEP kernel is defined as:

$$k(\mathcal{T}_1, \mathcal{T}_2) = w_r \cdot h(b[r_1], b[r_2]) + \frac{1 - w_r}{|\mathcal{E}_1||\mathcal{E}_2|} \cdot \sum_{\substack{e_1 \in \mathcal{E}_1 \\ e_2 \in \mathcal{E}_2}} h(b[e_1], b[e_2]) \tag{4.13}$$

As the roots have no parents, they are handled separately in this kernel: $w_r \in [0, 1]$ is a cross-validated parameter encoding a prior on the importance of the root-to-root comparisons. Note that the case $w_r = 1$ corresponds to the standard global BOF model with the intersection kernel.

This kernel relies on hierarchical relations: a node only depends on its parent. It can be seen as a similarity between all sub-trees of two BOF-trees. Let a *direct family* $(v, s(v), p(v))$ denote, respectively, a non-root node $v \in \mathcal{V}_i \backslash r_i$, its only sibling $s(v)$, and its parent $p(v)$. The additive property of BOF-trees is formulated as $b[v] + b[s(v)] = b[p(v)]$. Therefore, $(b[v], b[p(v)])$ completely characterizes a direct family. In addition, the BOF $b[v]$ is the sum of all the BOFs of its descendants. Consequently, $(b[v], b[p(v)])$ can be seen as an approximation of the content of the sub-tree rooted at $p(v)$. Therefore, the ATEP kernel efficiently compares all sub-trees by using only one level

of hierarchy at a time to compare two motion components. Note that if the node kernel h is positive definite, then our ATEP kernel is also positive definite (it is a sum of positive definite kernels). We can, therefore, use it directly in conjunction with a SVM classifier [Schölkopf and Smola 2002]. For multi-class classification, we adopt the One *v.s.* Rest approach.

4.4 Experiments

In this section, we report the results obtained with our approach and compare it to both the state of the art and several baselines.

4.4.1 Activity datasets

We evaluate our method on two publicly available benchmarks: the High Five dataset [Patron-Perez et al. 2010] and the Olympic Sports dataset [Niebles et al. 2010]. Both of these datasets focus on complex activities instead of more commonly investigated short actions such as "running".

The **High Five** dataset [Patron-Perez et al. 2010] consists of 300 video clips collected from 20 different TV shows (*cf.* Figure 4.5). The activity categories are four human-human interactions: hand shakes, high fives, hugs, and kisses. Each activity is performed in 50 different clips, the remaining 100 "negative" clips containing other actions. These activities are of short duration and involve a simple combination of atomic actions. As the dataset authors propose a model relying on head orientations, annotations for discrete head orientation and upper body localization of actors are available. We do not use this additional supervision in our experiments. Evaluation on this benchmark is conducted like in [Patron-Perez et al. 2010], *i.e.*, by computing the recognition performance in Average Precision using a 2-fold cross-validation with fixed folds provided with the dataset.

Figure 4.5: *Frames from the High Five dataset [Patron-Perez et al. 2010]*

The **Olympic Sports** dataset [Niebles et al. 2010] contains 783 Youtube videos of athletes practicing 16 different sport activities such as springboard diving and weight lifting (*cf.* Figure 4.6). This dataset contains longer video sequences of fast and complex articulated human motions, possibly involving interactions with objects (*e.g.*, a javelin). It presents several challenges, such as cluttered backgrounds, low quality videos, compression artifacts, and subtle distinctions between some categories (*e.g.*, triple jump and long jump). In addition, the activities are composed of multiple simpler actions (*e.g.*, running and jumping) that can be shared across categories, which is a great challenge for part-based recognition. Different actions might, indeed, contain the same parts, but arranged in a different manner. This justifies the need to leverage the spatio-temporal structure of activities in order to better distinguish them.

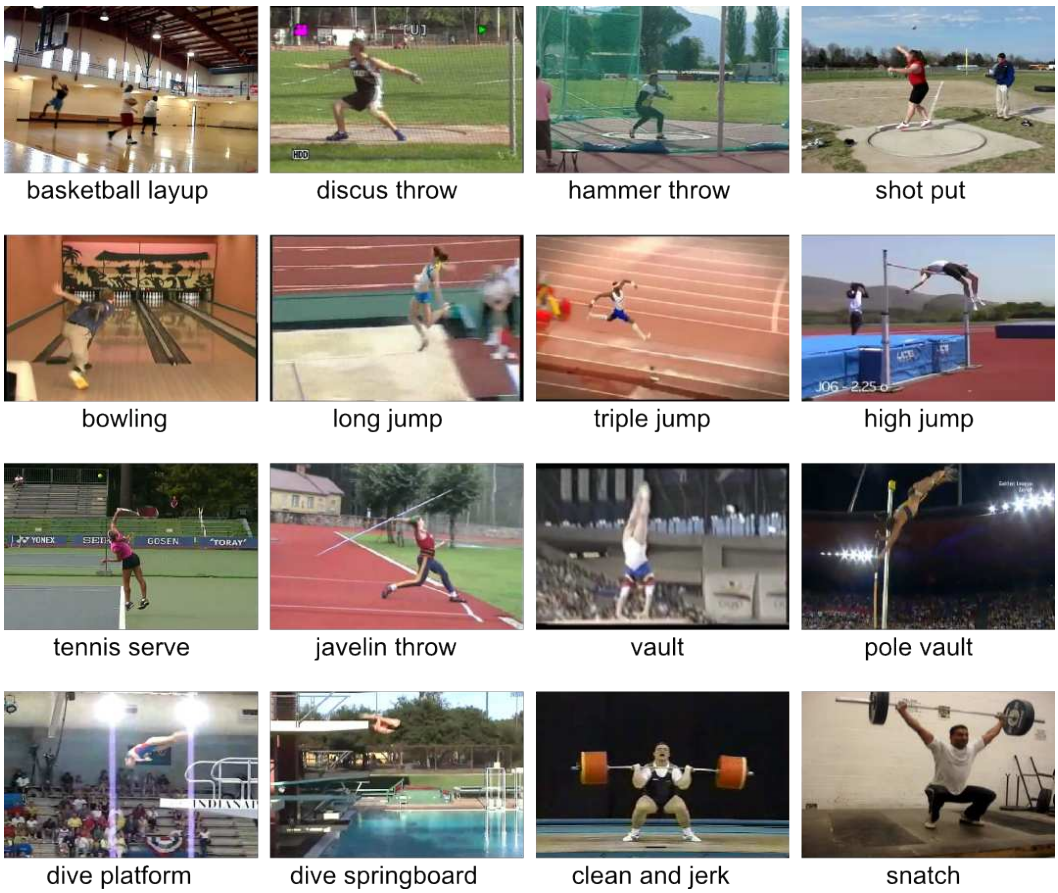


Figure 4.6: *Activities of the Olympic Sports dataset [Niebles et al. 2010]*

4.4.2 Camera motion stabilization

Although motion stabilization is not necessary with our approach, it allows to remove the camera motion bias of a dataset. Some video sources — and sports videos in particular — tend to be filmed in a manner that is strongly correlated with the action performed. For instance, in the stabilized frames of Figure 4.7 (bottom row), we can see the rapid downward tilt of the camera as it follows the landing of the jumper.

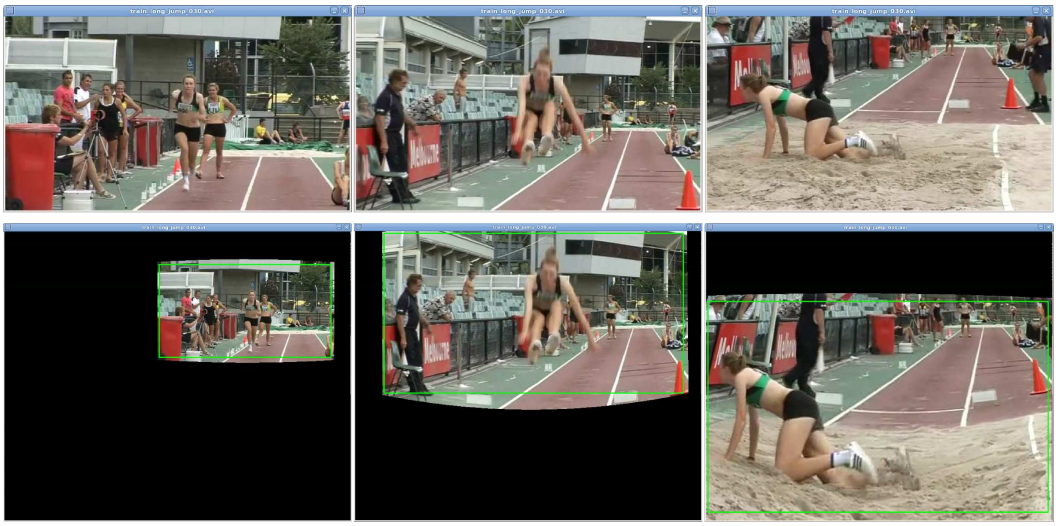


Figure 4.7: *An action of the Olympic Sport [Niebles et al. 2010] dataset, before (top row) and after (bottom row) camera motion stabilization.*

Such camera motion biases are consistently present in the Olympic Sports dataset and characterize to some extent the action performed. When using dense tracklets, camera motion is directly encoded by the features. Furthermore, the background tracklets will share similar motion statistics as they mostly correspond to the movement of the camera. As can be seen in figure 4.8, in the non-stabilized case (top row), the camera motion yields many consistent tracklets on the background, while in the second case most motion is on the moving objects (the jumper and the referee). Thus, in the non-stabilized case, we observed that our clustering approach may result in hierarchical decompositions, of which discriminative, large sub-trees are formed by background tracklets. Therefore, we report results on stabilized video for the Olympic Sports dataset. That way, our evaluation is independent of any (positive or negative) camera motion bias and only quantifies the efficiency of our action model. Note that the High Five clips do not suffer from camera motion bias. We, therefore, present results on the original non-stabilized video sequences for this benchmark.

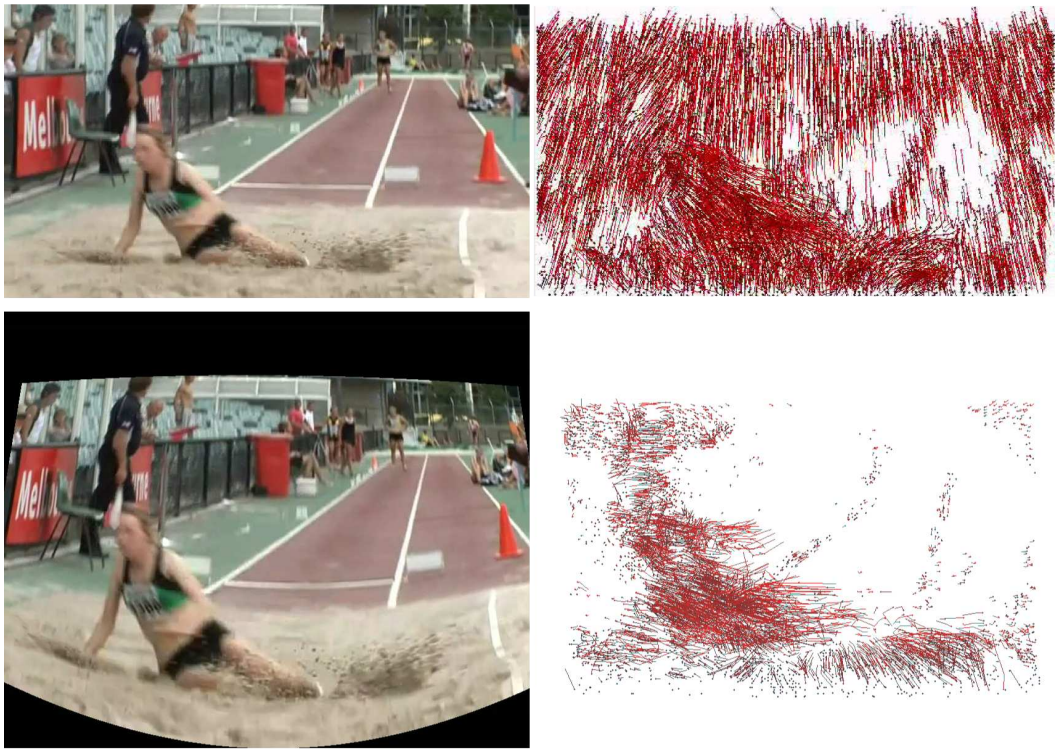


Figure 4.8: *Tracklets from a video (top) and from its stabilized version (bottom).*

Camera motion stabilization under uncontrolled video conditions is a difficult task. Although efficient approaches exist for simple transformations, *e.g.*, phase correlation [De Castro and Morandi 1987] for translations, only few can address all kinds of complex, time-varying camera motions, such as camera shake in amateur videos, or out-of-plane rotations coupled with zooms in movies. Here, we follow [Ikizler-Cinbis and Sclaroff 2010, Kuehne et al. 2011], *i.e.*, we explicitly model the camera motion to generate stabilized videos. The optical flow in these videos corresponds to the full flow in the original video, from which the estimated background flow was subtracted (*cf.* Figure 4.8).

We use simple and efficient image-stitching techniques to compute the approximate motion of a background plane. First, we extract salient 2D Harris interest points in frames which we represent with SIFT [Lowe 2004] descriptors. We, then, robustly match the interest points using RANSAC [Fischler and Bolles 1981] and estimate the transformation between all pairs of adjacent frames. Finally, we produce a stabilized video where all frames are warped according to the estimated transformation (*cf.* Figure 4.7). Note that the trajectories of points on the warped frame boundaries are filtered out and not included in our models.

4.4.3 Baselines

We first compare to two simple baselines using a global BOF model over the entire video [Laptev et al. 2008, Wang et al. 2011]. The approach of Laptev et al. [2008] uses sparse local Spatio-Temporal Interest Points (STIPS) [Laptev 2005] described by a concatenation of histogram of oriented gradients [Dalal and Triggs 2005] and optical flow, denoted "HOG/HOF". The approach of Wang et al. [2011] uses dense tracklets represented by MBH descriptors. Note that this method corresponds to using only the model of the root node of our BOF-trees. In both cases, we use our own implementation of the method with the same vocabulary construction and size as mentioned previously.

In addition to these unstructured baselines, we compare our approach with decompositions obtained by alternative clustering methods. First, we compare to two standard "flat" clustering algorithms that produce a set of unrelated clusters: k-means and spectral clustering. Due to the large amount of tracklets per video, we adopt an efficient on-line variant of k-means [Sculley 2010]. For spectral clustering, we adopt the approach of Fowlkes et al. [2004]: we partition tracklets with (on-line) k-means on the same approximate spectral embedding used by our method. These two algorithms take as input a fixed number of clusters. However, different videos contain different numbers of distinct motion components. Therefore, we use a number of clusters that depends linearly on the number of tracklets, such that the smallest videos have at least 2 clusters, whereas the largest ones have at most 1000 clusters.

We also compare our approach to a closely related baseline yielding a cluster-tree. Its steps are all similar to our method except for one: we replace the spectral divisive thresholding algorithm of Section 4.2.4 (noted SDT) by a recursive application of k-means. This amounts to splitting a node with (on-line) k-means ($k = 2$) on the spectral embedding. The algorithm is noted "SDKM" for Spectral Divisive K-Means. Like with our approach, we classify the resulting cluster-trees using a SVM with our BOF-tree kernel.

Note that in order to compare unrelated sets of clusters, we use a natural simplification of our ATEP kernel consisting in averaging all pairwise cluster comparisons between two videos. This baseline kernel is noted as "ACP" for "All Cluster Pairs". It can be applied on the clusters resulting from the "flat" clustering algorithms, the leaves of a BOF-tree, or on all nodes of a BOF-tree (including internal ones) in which case the hierarchical relations are ignored.

Table 4.1 contains a summary of all notations used to describe the methods and kernels we compare in our experiments.

Tree-structured activity models	
SDT tree	BOF-Tree (Section 4.3.2) obtained by Spectral Divisive Thresholding on tracklets (Section 4.2.4)
SDKM tree	BOF-Tree (Section 4.3.2) obtained by bi-partitioning K-Means on tracklets (Section 4.4.3)
Flat sets of bag-of-features	
SDT leaves	leaves of the SDT BOF-Trees
SDKM leaves	leaves of the SDKM BOF-Trees
spectral	clusters obtained by spectral clustering (Section 4.4.3)
kmeans	clusters obtained by k-means (Section 4.4.3)
Unstructured baselines	
BOF tracklets	bag-of-features with all tracklets as in [Wang et al. 2011], equal to the root of a BOF-tree
BOF STIPS	bag-of-features with spatio-temporal interest points as in [Laptev et al. 2008]
Kernels on part-based activity models	
ATEP	All Tree Edge Pairs kernel, mean of all pairwise edge comparisons, specific to BOF-Trees (Section 4.3.3)
ACP	All Cluster Pairs kernel, mean of all pairwise cluster comparisons, applicable to BOF-Trees and flat decompositions (Section 4.4.3)

Table 4.1: *The methods we implemented and compared in our experiments.*

4.4.4 Activity recognition results

Table 4.2 reports performance comparisons between our unstructured and structured baselines, the state of the art, and our method.

SDT tree & ATEP	**85.0**
SDKM tree & ATEP	77.8
SDT tree & ACP	79.3
SDKM tree & ACP	70.6
SDT leaves & ACP	77.9
SDKM leaves & ACP	72.2
spectral & ACP	71.7
kmeans & ACP	70.8
BOF tracklets	76.6
BOF STIPS	61.3
Brendel and Todorovic [2011]	77.3
Niebles et al. [2010]	72.1

(a) Olympics Sports (Accuracy in %)

SDT tree & ATEP	**56.5**
SDKM tree & ATEP	55.3
SDT tree & ACP	51.3
SDKM tree & ACP	46.5
SDT leaves & ACP	47.0
SDKM leaves & ACP	45.3
spectral & ACP	48.9
kmeans & ACP	50.1
BOF tracklets [Wang et al. 2011]	53.5
BOF STIPS [Laptev et al. 2008]	36.9
Patron-Perez et al. [2010]	32.8

(b) High Five (AP in %)

Table 4.2: *Performance comparison on the Olympics Sports [Niebles et al. 2010] and High Five [Patron-Perez et al. 2010] datasets.*

First, we found that our hierarchical ATEP kernel on SDT BOF-trees in conjunction with a SVM outperforms the state of the art on both datasets, including latent part models [Niebles et al. 2010] (+12.9%), complex graphical models resulting from video segmentation [Brendel and Todorovic 2011] (+7.7%, *cf.* Figure 4.9 for per-class details), and interaction-specific structured learning [Patron-Perez et al. 2010] (+23.7%).

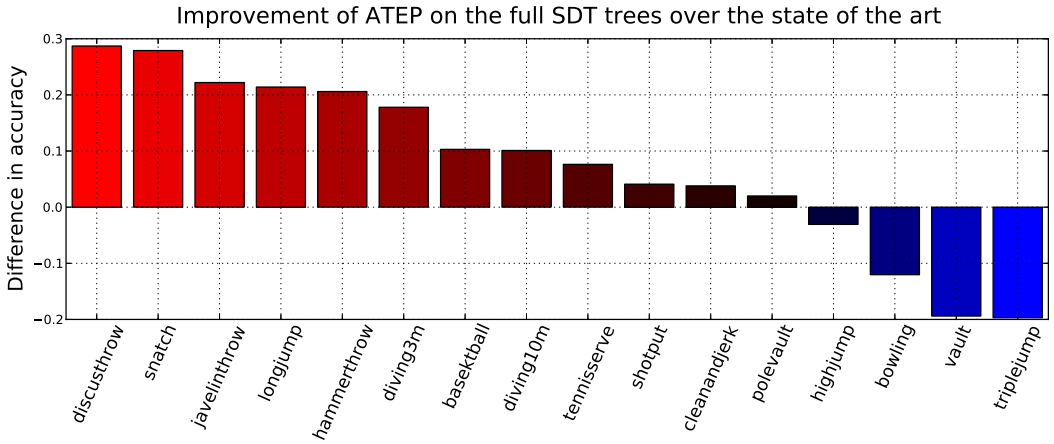


Figure 4.9: *Performance improvement of our method over the method from [Brendel and Todorovic 2011] on the Olympic Sports dataset.*

Second, our approach improves over the BOF baselines. This confirms the importance of leveraging structure information to recognize complex activities. Note, however, that only our method yields clear performance improvements, whereas other structured baselines are less accurate. This shows that decomposing activities is as challenging as it is critical for performance.

Third, using hierarchical relations between motion components with our ATEP kernel consistently improves over the "flat" baselines, which rely on unrelated sets of clusters, *e.g.*, the leaves of BOF-trees or clusters obtained by k-means. We also observe a similar improvement when comparing the performance of our ATEP kernel with its unstructured counterpart (ACP) on the full BOF-trees (*cf.* Figure 4.10 for per-class details on the Olympic Sports dataset). This confirms that our method captures useful structure information, and that the improvement does not result only from the decomposition into parts, but also from the comparison of their relations.

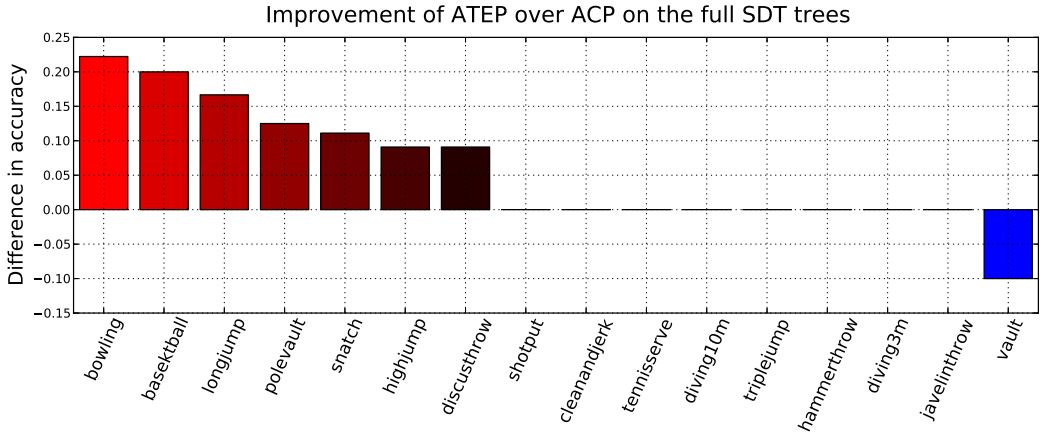


Figure 4.10: *Performance improvement obtained by using the hierarchical relations of the SDT BOF-trees on the Olympic Sports dataset.*

Finally, table 4.2 also shows that the BOF-trees produced by our SDT algorithm yield more powerful models than the SDKM ones obtained by bi-partitioning k-means (+7.2% on Olympic Sports, +1.2% on High Five).

Discussion on the Olympic Sports results

Figure 4.11 is the confusion matrix of our method on Olympic Sports. We can see, for instance, that "triple jump" is often predicted as the similarly structured "long jump" (the only difference between the two categories is the middle of the activity). In addition, "high jump" is often wrongly predicted as "javelin throw" due to their similar structure (running, then hopping, followed by a fast upwards motion). Another frequent confusion is between the "tennis serve" and "discus throw" actions, which both involve a long preparation time with only few movements, followed by a short and fast motion.

As can be observed in Figure 4.10, our hierarchical method yields the largest improvements on the "bowling" and "basketball layup" classes. Both of these activities involve a clear spatio-temporal structure that is captured by our cluster-trees. Furthermore, they contain large motions (*e.g.*, running) as well as fast and short motion parts that are characteristic for the category (*e.g.*, the layup gesture, or the arm movement when throwing the bowling ball). Our model can separate these components, whereas averaging models like BOF will be dominated by the largest clusters of tracklets, thus losing the discriminative information present in the motion parts with less tracklets.

The "vault" (gymnastics) activity is the only category where our method degrades performance with respect to its unstructured counterpart (ACP kernel on SDT BOF-Trees, *cf.* Figure 4.10). It is often confused with another difficult category: "triple jump" (*cf.* Figure 4.11). After analyzing the results, we observed that this confusion results from common artifacts due to incorrect stabilization of the fast traveling motions of the cameras filming both the "vault" and "triple jump" categories.

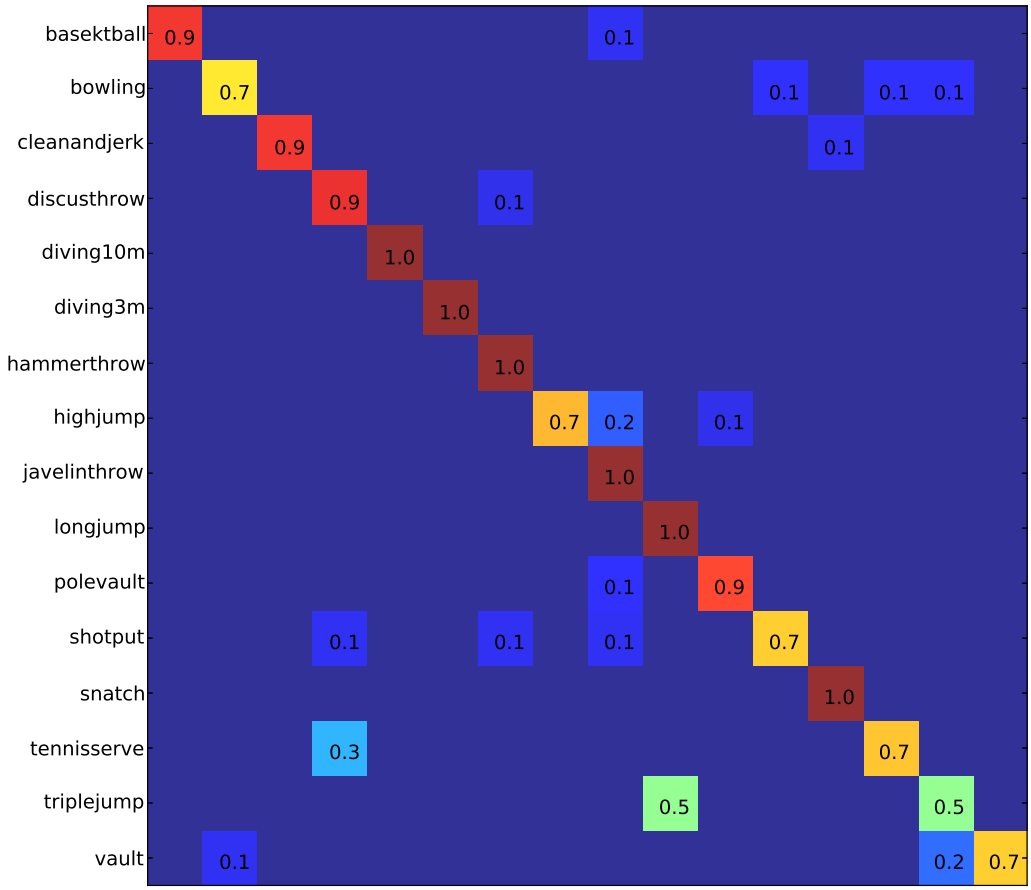


Figure 4.11: *Confusion matrix obtained with our method (ATEP on SDT BOF-Trees) on the Olympic Sports dataset.*

Discussion on the High Five results

Table 4.2 also shows that the performance improvements on the High Five dataset are less significant than on the Olympic Sports benchmark. This can be explained by several factors. First, the interactions contained in the High Five dataset are shorter and are, therefore, decomposed over smaller trees with less structure. Second, all interaction categories are close and share both common motion parts and relations between them (some are visible in Figure 4.5). Note that the per-class improvements on the High Five dataset are all roughly equal. Third, the High Five dataset contains additional random negative videos — both in the training and test splits — that have no particular motion structure, and, thus, increase the noise level. When removing these random negatives from the test samples, our method reaches 66% in mean Average Precision, which represents a +27% improvement over the results of Patron-Perez et al. [2010] in the same settings (39% mean AP).

4.4.5 Results for simpler actions

In addition to the previously described experiments on activity recognition, we investigated the performance of our method on simpler actions from the HMDB dataset [Kuehne et al. 2011] (*cf.* Figure 4.12 for frames extracted from some clips of the database).

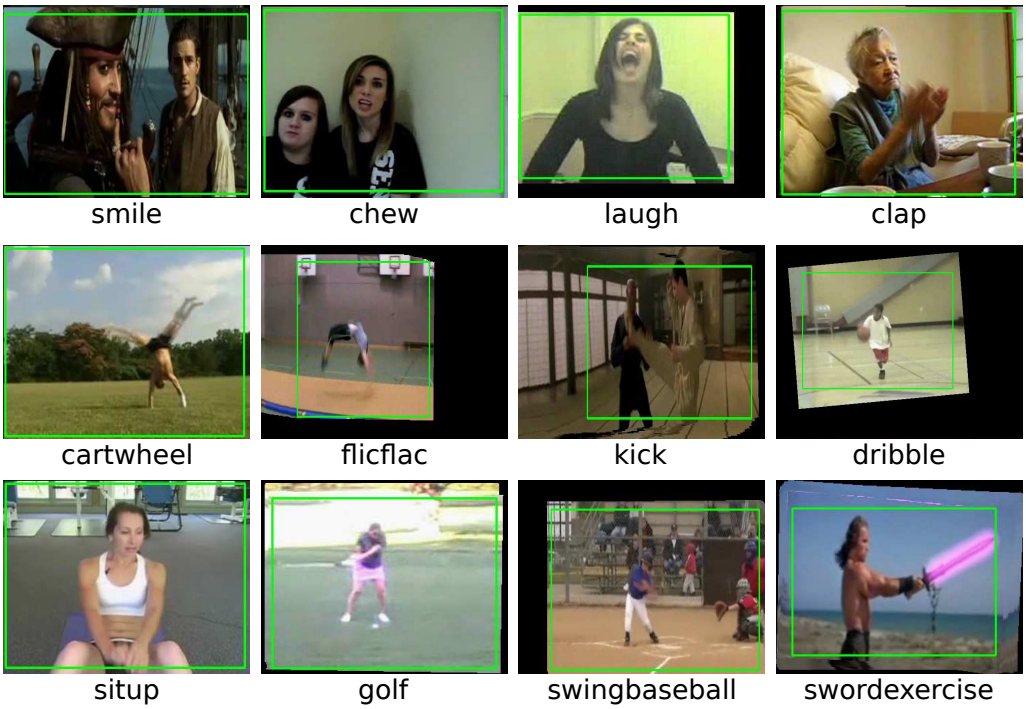


Figure 4.12: *Stabilized videos of the HMDB dataset [Kuehne et al. 2011]*

This benchmark is composed of 6849 clips from different real-world sources such as Youtube videos, TV footage, and movies. It contains 51 action categories that entail general facial actions (*e.g.*, smile, laugh, talk), facial actions with object manipulation (*e.g.*, smoke, drink), general body movements (*e.g.*, cartwheel, climb, walk, wave), interactions with objects (*e.g.*, dribble, golf), and person-person interactions (*e.g.*, hug, kiss, punch). Each action is performed in at least 101 clips. Performance is evaluated by the mean accuracy over all classes averaged over three fixed different train/test splits provided by the authors. HMDB is a challenging dataset: the current state of the art [Sadanand and Corso 2012] is of 26.9% accuracy only. This highlights the difficulties of this dataset, which include poor video conditions (only 17% of high quality clips), partially visible actors (in 44% of the videos), camera motions, and a large variety of viewpoints. In addition, another difficulty is caused by the presence of close pairs of classes such as chewing *v.s.* talking, shooting a gun *v.s.* shooting a bow, fencing *v.s.* sword exercise.

We use this dataset in addition to the previous ones because of their complementarity. On the one hand, the actions in the Olympic Sports dataset are longer, more complex activities with 230 frames per action on average. They are composed of multiple simpler actions (*e.g.*, running and jumping) that can be shared across categories, which is a great challenge for part-based recognition and justifies the need to leverage the spatio-temporal structure of actions. On the other hand, HMDB actions — some of which are parts of the more complex Olympic Sports activities — tend to be both simpler and shorter with 90 frames per action on average, but they are more numerous and varied. Note that we use the on-line available stabilized version of HMDB.

Our method (ATEP kernel on SDT BOF-Trees) obtains an average classification accuracy of 41.3%, which corresponds to a +14.4% improvement over the state of the art. This improvement is in large part due to our use of tracklets combined with the MBH descriptor. Indeed, the performance of our BOF baseline — *i.e.*, using only the root of our BOF-trees — reaches 38.9% accuracy. Figure 4.13 contains the confusion matrix for our method. The classes are clustered by confusion, according to a 1D projection obtained by Locally Linear Embedding (LLE) [Roweis and Saul 2000] on the rows of the confusion matrix. The actions most often confused are the facial actions visible in the top left corner of the confusion matrix: chew, smile, talk, and laugh. They all involve only few tracklets depicting consistent movements of the head and facial muscles that are too simple and too small to decompose. Other common confusions include smoke *v.s.* eat *v.s.* drink, draw sword *v.s.* sword exercise, and actions at the bottom-right corner of the confusion matrix (swing baseball, throw, jump, kick, kick ball, and fall floor), which generally involve a single, fast, uni-directional, translation movement.

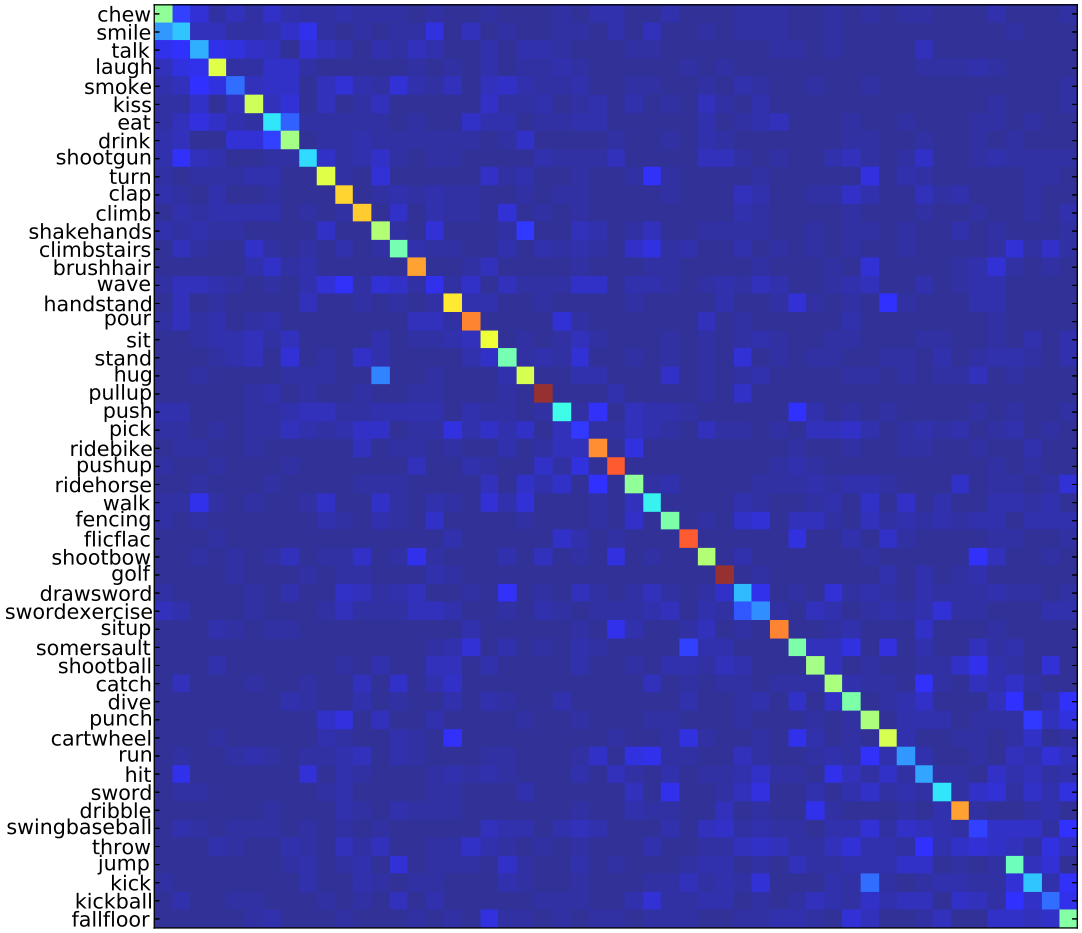


Figure 4.13: *Confusion matrix obtained with our method (ATEP on SDT BOF-Trees) on the HMDB dataset [Kuehne et al. 2011].*

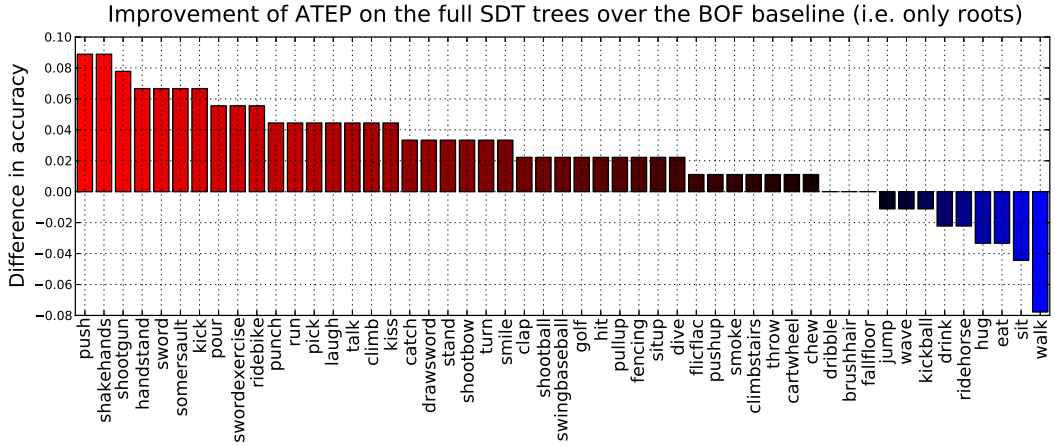


Figure 4.14: *Performance improvement of our method over the BOF baseline on the HMDB dataset [Kuehne et al. 2011].*

Figure 4.14 contains the per-class performance differences between our method and the BOF baseline. We can observe that, although the improvement is limited on average over all categories (+2.4%), the majority of the actions clearly benefit from our hierarchical decomposition. For instance, the "push" category is significantly improved (+9%), because it consists in the interaction between a human and an object (a cart). Similarly, our method improves the recognition of the human-human interaction "shake hands" (+9%), and other interactions, *e.g.*, "shoot gun", "pour", and "kiss".

Some actions, on the other hand, are negatively impacted by our method. However, as we cross-validate the per-class root weight w_r in ATEP (Equation 4.13), and as the case $w_r = 1$ corresponds to the BOF baseline, it should not be possible, in theory, for our method to perform worse than BOF. In practice, however, the limited training set size does not guarantee that the best parameters on the test set are those with maximum cross-validation performance. The performance degradation observed for some classes suggests that there is not a clear and stable value for the best root weight parameter of these actions. In particular, the performance on the "walk" action is smaller (−8%) with our model than with the BOF baseline. Indeed, the motions of the legs are sometimes either too close to each other to be separated, or are not visible. Furthermore, videos of this category often involve a wide array of contexts, most of which contain the unrelated motions of other objects or actors. These "distractors" result in spurious nodes in our cluster-trees, and negatively affect our kernel, as we rely on the full cluster-tree structure. Note that we did not observe such performance degradation on the activities of the Olympic Sports and High Five datasets: our ATEP kernel on SDT BOF-Trees performed equally or better than the BOF baseline on all activities.

4.5 Conclusion

We introduced an efficient hierarchical clustering algorithm on short duration point trajectories (tracklets). Our approach relies on recursive bi-partitioning using a spatio-temporal connectedness criterion to threshold the projection of tracklets on a multi-modal spectral embedding obtained with the Nyström approximation. A video is structured as a tree of nested motion components. Each one of these data-driven parts is represented by a bag-of-features over local motion descriptors in order to form our BOF-Tree activity model. We also proposed a kernel on unordered binary trees, which can accurately compare activities with a variable number of hierarchically ordered parts. This kernel uses an additive property of our motion decomposition to efficiently compare sub-trees and leverages structure information to refine part comparisons.

Our experimental results show that the combination of our clustering algorithm and our tree kernel outperforms unstructured bag-of-features baselines, video decompositions with other clustering techniques, and the state of the art — including approaches based on latent parts, video segmentation, and structured learning. Furthermore, we observed that our method can be successfully applied to short actions as long as they have a characteristic motion decomposition structure. We observed, however, that the performance improvements with our method are more significant on complex activities involving the spatio-temporal composition of several short actions.

Chapter 5

Time Series Kernels for Action Recognition

Contents

5.1 Introduction

Action recognition in realistic videos has been previously addressed using kernel methods like SVMs. Most existing approaches, however, extract fixed-length vector representations of video volumes (*e.g.*, a bag-of-features of an entire video or of each part like in previous chapters), and compare them using generic kernels on vectors. In contrast, in this chapter we model *actions as time series* of per-frame representations, and propose a *kernel specifically tailored for the purpose of action recognition*, which hinges upon distances between models of action dynamics. Dynamic aspects alone are likely to be insufficient to describe some types of actions, especially when the scene or the nature of objects involved are discriminative. Therefore, our goal is to show that our kernel on dynamics *complements* orderless aggregation statistics (*cf.* Figure 5.1), which have proved efficient in this context.

We propose to encode the statistical dependency information of an action example by using its *auto-correlation operator*, *i.e.*, the cross-correlation in a Reproducing Kernel Hilbert Space (RKHS)[Schölkopf and Smola 2002] of the signal of frames with a temporally shifted version of itself. The auto-correlation operator contains information pertaining to the temporal dependencies between frames and the temporal structure of actions, as it depends on the ordering of the frames. Hence, we propose to *compare the dynamics of two actions by computing the distance between their respective auto-correlations.* This distance is defined as the Hilbert-Schmidt norm of the difference between auto-correlations and we call the associated Gaussian RBF kernel the *Difference between Auto-Correlation Operators* (DACO) kernel (see figure 5.2 for an illustration). Note that this is different from the cross-correlation between

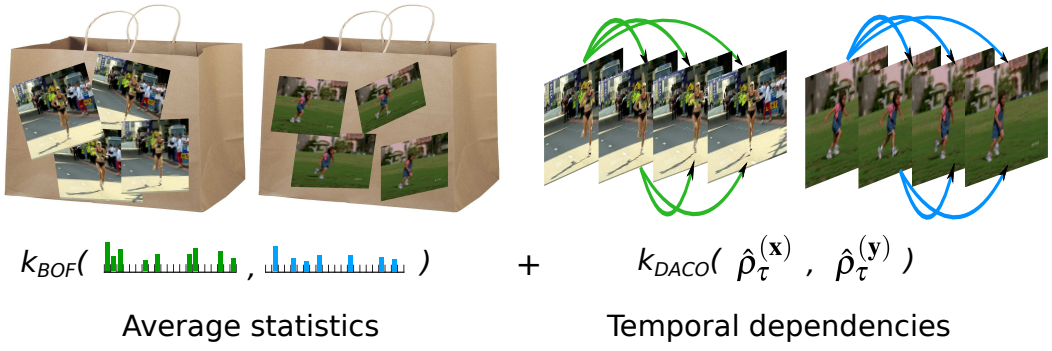


Figure 5.1: *Our goal is to complement the robustness of the orderless bag-of-features model of actions (left) with temporal dependency information (right) captured by our kernel on dynamics.*

video volumes, which measures dependencies between frames of two different videos and is not suited to compare different actions with strongly related motions (*e.g.*, running and walking).

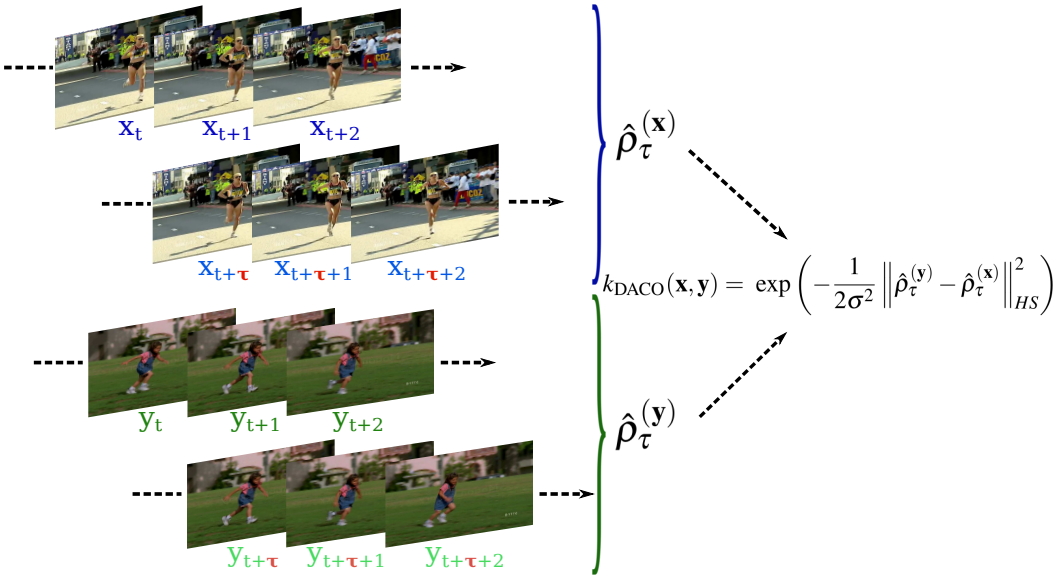


Figure 5.2: *Computation of our DACO kernel. For two actions represented as time series of frames, $\mathbf{x} = (x_1, \ldots, x_T)$ and $\mathbf{y} = (y_1, \ldots, y_{T'})$, the kernel compares their dynamics by using the difference between their auto-correlations $\hat{\rho}_\tau^{(\mathbf{x})}$ and $\hat{\rho}_\tau^{(\mathbf{y})}$, with a lag of τ frames.*

Closest references

Though good results were achieved with models aggregating statistics of local features over the entire duration of an action (*cf.* [Wang et al. 2009] for a recent evaluation), these action models can be enhanced by treating the time dimension differently from the spatial ones. For instance, Niebles et al. [2010] propose a latent model of temporal parts for long duration activities, and Gaidon et al. [2011a] model the temporal structure and ordering constraints of actions as sequences of "actoms". These methods only encode the coarse temporal structure, and, therefore, fail to capture dynamic aspects like the temporal dependencies between frames. Another limitation of volumetric approaches is that the temporal granularity is not taken into account. The visual difference between two frames is, indeed, much more pronounced than the difference between two pixels, as fast discriminative motions often occur in less than ten frames.

One of the simplest ways to represent actions and their temporal structure is just to concatenate, in the temporal order, per-frame feature vectors. Schindler and Van Gool [2008] show that such a simple technique can yield good results in simple video conditions. However, their approach assumes that videos are synchronized in time beforehand. Consequently, it is not robust to significant variations in action speed.

A more sophisticated approach based on chaos theory is used by Basharat and Shah [2009]. They model repetitive human actions and dynamic textures as nonlinear dynamical systems. They use "strange attractors" to represent the dynamics of time series for action and dynamic texture synthesis, yet do not provide a way to compare two series of observations.

Other approaches, inspired by speech and gesture recognition, represent actions as sequences of states [Brendel and Todorovic 2010, Jung et al. 2008, Kulkarni et al. 2010, Nowozin et al. 2007], or use dynamic probabilistic graphical models [Brand et al. 1997, Laxton et al. 2007, Yamato et al. 1992, Zeng and Ji 2010] to model the temporal aspects of the videos. These methods, however, only measure alignments between videos. Hence, they are not robust to temporal occlusions, and significant duration variations. Furthermore, they generally involve a difficult intermediate recognition step, for instance by labeling each frame.

Both alignment-based approaches — computing a matching score between videos — and aggregation-based techniques — discarding most temporal aspects by averaging over frames — do not take into account characteristic dynamic information, such as repeating patterns, or the relationships between frames. In contrast to these previous works, we represent videos directly as time series of frames and propose to compare actions through models of their key dynamic aspects.

Note that our DACO kernel differs from existing kernels on time series. Cuturi et al. [2007] proposed a kernel based on Dynamic Time Warping, with applications to speech recognition tasks. However, this kernel does not compare the dynamics but measures alignments between time series. Another example of time series kernel is given by Lu et al. [2008], with applications to synchronized EEG segments. However, their similarity between two time series corresponds to similarity between temporal regularity, in the spirit of the functional data analysis framework [Ramsay and Silverman 2005].

Outline

We introduce our time series representation of videos and our novel DACO kernel in Section 5.2. We give a practical formulation that can operate on any type of frame model, including high-dimensional ones like BOF, in Section 5.3.

DACO only requires a kernel function on frames (*e.g.*, the intersection kernel between histograms), and computes auto-correlation operators in the feature space induced by this kernel. Section 5.4 contains experimental results on recent action recognition datasets, showing that a simple linear combination of our DACO kernel and an aggregation-based kernel can improve recognition performance. In addition, we show the superiority of DACO over alternative kernels on time series.

5.2 Kernel on auto-correlation operators

Let $\mathcal{X}$ a space of frame representations (*e.g.*, histograms of visual words). Let a video $\mathbf{x} = (x_t)_{t=1\cdots T}$ of duration T frames represented as a time series where $x_t \in \mathcal{X}$. We define the space of videos $\mathcal{S} = \bigcup_{i>0} \mathcal{X}^i$. Our goal is to design a kernel $k_S : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ adapted to compare actions. In Section 5.2.1, we quickly recall the main properties of the auto-correlation, which is the base mathematical component of our approach. In Section 5.2.2, we give the definition of our auto-correlation-based kernel and some details.

5.2.1 Auto-correlation operators

We model the dynamics of an actions $\mathbf{x} = (x_t)_{t=1,\cdots,T}$ (a time series of frames) with its corresponding *auto-correlation operator* $\hat{\rho}_\tau^{(\mathbf{x})}$, *i.e.*, the series's sampled auto-covariance, $\hat{\Sigma}_\tau^{(\mathbf{x})}$, normalized by its sampled covariance, $\hat{\Sigma}^{(\mathbf{x})}$:

$$\hat{\rho}_\tau^{(\mathbf{x})} = \left(\hat{\Sigma}^{(\mathbf{x})} + \gamma \mathbf{I}\right)^{-1} \hat{\Sigma}_\tau^{(\mathbf{x})} \tag{5.1}$$

where γ is a regularization parameter, and τ is a time lag in frames. The auto-covariance of a time series is simply defined as the cross-covariance between the series and a temporally shifted version of itself. For a time series (X_t) with mean $E[X_t] = \mu_t$, the *auto-covariance at time t and lag τ* is defined by

$$\begin{aligned} \Sigma^{(x)}(t, t+\tau) &= E\left[(X_t - \mu_t) \otimes (X_{t+\tau} - \mu_{t+\tau})\right] \\ &= E\left[X_t \otimes X_{t+\tau}\right] - \mu_t \otimes \mu_{t+\tau} \end{aligned} \tag{5.2}$$

where $\otimes$ denotes the tensor product operation.

Note that Equation 5.1 makes the assumption that all time series are wide sense stationary, *i.e.*, that first and second order moments do not vary with time. We show in our experiments (Section 5.4) that this approximation yields reasonable results in practice for short actions.

We note the mean $\mu = \mu_t$ and the auto-covariance $\Sigma_\tau^{(x)} = \Sigma^{(x)}(t, t+\tau)$, which only depends on the lag τ. Their sampled versions estimated from the observation of the frames are noted $\hat{\mu}$ and $\hat{\Sigma}_\tau^{(\mathbf{x})}$.

The auto-covariance contains information pertaining to the temporal dependencies between frames, *e.g.*, repeating patterns. It is a special case of cross-covariance, which has some interesting statistical properties. The Hilbert-Schmidt Independence Criterion (HSIC) [Gretton et al. 2005] between two random variables $X \sim \mathcal{P}_\mathcal{X}$, $Y \sim \mathcal{P}_\mathcal{Y}$ is a statistical dependency measure using the squared Hilbert-Schmidt norm of the cross-covariance operator between X and Y in some kernel-induced feature space: $\|\Sigma^{(\mathbf{x},\mathbf{y})}\|^2_{HS}$. Gretton et al. [2005] prove that, under certain conditions, $\|\Sigma^{(\mathbf{x},\mathbf{y})}\|^2_{HS}$ is zero if and only if the random variables X and Y are independent. Furthermore, they show that it can be estimated using only kernel evaluations between *i.i.d.* sampled observations of X and Y.

However, contrary to Gretton et al. [2005], we are dealing with non-*i.i.d.* observations of frames, and different actions might be statistically dependent (*e.g.*, "walking" and "running"). Therefore, we propose to compare time series by computing the distance between their dynamics, modeled by their respective auto-correlation operators.

5.2.2 The DACO kernel

We define our *Difference between Auto-Correlation Operators* (DACO) kernel from the Hilbert-Schmidt norm of the difference between auto-correlations:

$$\begin{aligned} k_{\mathrm{DACO}}(\mathbf{x},\mathbf{y}) &= \exp\left(-\frac{1}{2\sigma^2} d_{\mathrm{DACO}}(\mathbf{x},\mathbf{y})^2\right) \\ \text{where} \quad d_{\mathrm{DACO}}(\mathbf{x},\mathbf{y}) &= \left\|\hat{\rho}_\tau^{(\mathbf{y})} - \hat{\rho}_\tau^{(\mathbf{x})}\right\|_{HS} \end{aligned} \tag{5.3}$$

The Hilbert-Schmidt norm, noted $\|\cdot\|_{HS}$, is simply the extension of the Frobenius matrix norm to any separable Hilbert space, and can be defined for any bounded operator A as:

$$\|A\|^2_{HS} = \mathrm{Tr}\left(A^* A\right) \tag{5.4}$$

where Tr denotes the trace function and A^* is the conjugate transpose of A. It derives from the Hilbert-Schmidt inner product:

$$\langle A, B\rangle_{HS} = \mathrm{Tr}\left(A^* B\right) \tag{5.5}$$

This allows us to decompose the DACO distance in three terms:

$$d_{\mathrm{DACO}}(\mathbf{x},\mathbf{y})^2 = \left\|\hat{\rho}_\tau^{(\mathbf{x})}\right\|_{HS} + \left\|\hat{\rho}_\tau^{(\mathbf{y})}\right\|_{HS} - 2\left\langle\hat{\rho}_\tau^{(\mathbf{y})}, \hat{\rho}_\tau^{(\mathbf{x})}\right\rangle_{HS} \tag{5.6}$$

As the norm of the auto-correlation operator measures the dependency between a series and a shifted version of itself, we can see from Equation 5.6 that the distance will tend to be smaller for time series with almost no temporal

structure (*e.g.*, for random sequences of frames), and bigger for actions with quasi-deterministic relationships between neighboring frames (*e.g.*, a constant translation movement). In addition, the inner product in Equation 5.6 shows that if the dynamics of the two series $\mathbf{x}$ and $\mathbf{y}$ are different, then the distance will be bigger. Therefore, actions with strong but different temporal structures will tend to have large DACO distances.

Consequently, DACO is well suited to compare actions characterized by their dynamics, but needs to be combined with another kernel in order to deal with actions with little temporal structure. This shows that combining DACO with an aggregation-based kernel allows to efficiently represent dynamics and orderless distribution aspects, which are both useful for action recognition.

5.2.3 The time-lag parameter

The main parameter of our DACO kernel is the lag τ in frames. For periodic actions, it is important that this parameter be different from a multiple of the period. Indeed, a periodic signal is always perfectly correlated with its version shifted by a period, and comparing two auto-correlations of two periodic signals with the same periodicity yields uninformative distances.

This problem can be avoided in practice by multiple techniques, for instance by detecting the period or by averaging the distances for multiple τ values. However, these methods are generally either expensive or unreliable in the absence of prior information, *e.g.*, for periodic actions with variable period durations.

The alternative solution that we found to work best in practice is to simply take a τ which is small enough with respect to the action duration, such that it cannot be a multiple of a period. As we deal with short actions that include fast motions and potentially drastic changes in a few frames, we chose a τ of *one frame* in our experiments. This has the other advantage to not "dilute" temporal relations between the signal and its shifted version, hence ensuring the DACO distances are meaningful thanks to the preservation of strong temporal structures.

Another advantage of using a small lag value is to limit border effects when estimating the sampled auto-correlation operator. We, indeed, deal with the recognition of short videos, which have only a finite number of observations per time series. Hence, we need either to truncate them, $\mathbf{x} = (x_1, \cdots, x_{T-\tau})$ and $\mathbf{x}^\tau = (x_{1+\tau}, \cdots, x_T)$, or to have a cyclic representation (modulo T). As the first solution is reducing the number of observations and making our representations dependent on the size of τ, we choose the latter (cyclic) one.

5.3 Practical formulation of DACO

In the following section, we give a practical formulation of the DACO distance that is obtained by kernelizing Equation 5.3, *i.e.*, expressing it using only inner products between frames, computed via a kernel function.

5.3.1 Kernelized DACO

Frame representations are in general high-dimensional (*e.g.*, several thousands of dimensions for BOF) and can be of a non-vector type (*e.g.*, graphs). Therefore, instead of making assumptions on the frame models, we only assume the availability of a symmetric positive-definite kernel between frames $k_F : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such as the intersection kernel between per-frame BOFs. We note $\mathcal{H}_F$ the feature space, and $\phi_F : \mathcal{X} \to \mathcal{H}_F$ the feature map associated with the kernel $k_F(x_t, y_{t'}) = \langle \phi_F(x_t), \phi_F(y_{t'}) \rangle_{\mathcal{H}_F}$ between two frames x_t and $y_{t'}$ of two series $\mathbf{x}$ and $\mathbf{y}$. $\mathcal{H}_F$ might be infinite-dimensional, for instance when the Gaussian RBF kernel is used. Yet, using the kernel trick, our kernel can be computed only by using kernel evaluations between frames.

In the following, we adopt notations similar to those of Shawe-Taylor and Cristianini [2004]. Corresponding to the video $\mathbf{x}$, we define the time series $\mathbf{X}$ of frames in the frame feature space $\mathcal{H}_F$ with $\mathbf{X} = [\phi_F(x_1) \ \cdots \ \phi_F(x_T)]$, where the column t of $\mathbf{X}$ is the projection $\phi_F(x_t)$ of frame x_t. For two time series $\mathbf{x} = (x_t)_{t=1 \cdots T}$, $\mathbf{y} = (y_{t'})_{t'=1 \cdots T'}$ and their representations $\mathbf{X}$ and $\mathbf{Y}$, the matrix $\mathbf{K}$ of kernel evaluations between frames of both series is noted:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}^{(\mathbf{x})} & \mathbf{K}^{(\mathbf{x},\mathbf{y})} \\ \mathbf{K}^{(\mathbf{y},\mathbf{x})} & \mathbf{K}^{(\mathbf{y})} \end{bmatrix} \tag{5.7}$$

$$\mathbf{K}^{(\mathbf{x})} = \mathbf{X}^T\mathbf{X} \ , \quad \mathbf{K}^{(\mathbf{y})} = \mathbf{Y}^T\mathbf{Y} \ , \quad \mathbf{K}^{(\mathbf{x},\mathbf{y})} = \mathbf{X}^T\mathbf{Y} = \left(\mathbf{K}^{(\mathbf{y},\mathbf{x})}\right)^T$$

Using our previous notations, we define the auto-covariance of action $\mathbf{x}$ at lag τ in the frame feature space $\mathcal{H}_F$ as:

$$\hat{\Sigma}_\tau^{(\mathbf{x})} = \frac{1}{T}\mathbf{X}\mathbf{X}_{+\tau}^T \quad \text{where} \quad \mathbf{X}_{+\tau} = [\phi_F(x_{1+\tau}) \ \cdots \ \phi_F(x_{T+\tau})] \tag{5.8}$$

$\mathbf{x}^\tau = (x_{1+\tau}, \cdots x_{T+\tau})$ is the shifted version of $\mathbf{x}$ and $\mathbf{X}_{+\tau}$ is the corresponding time series representation in $\mathcal{H}_F$. Additionally, we define the kernel matrix $\mathbf{K}^{(\mathbf{x}^\tau)}$ between frames of $\mathbf{x}^\tau$.

Note, that this formulation assumes that our actions have zero mean $\hat{\mu}_x = \frac{1}{T}\sum_{t=1}^{T} \phi_F(x_t) = 0$. This requires centering the frames in the feature space $\mathcal{H}_F$. As our computations use only kernel matrices, we directly center

them [Harchaoui et al. 2008, Shawe-Taylor and Cristianini 2004]:

$$\tilde{\mathbf{K}}^{(\mathbf{x})} = \tilde{\mathbf{X}}^T\tilde{\mathbf{X}} = \Pi_T\mathbf{K}^{(\mathbf{x})}\Pi_T, \quad \tilde{\mathbf{X}} = \mathbf{X}\Pi_T, \quad \Pi_T = \mathbf{I}_T - \frac{1}{T}\mathbf{1}_T\mathbf{1}_T^T \tag{5.9}$$

where $\mathbf{I}_T$ is the $T \times T$ identity matrix and $\mathbf{1}_T$ is the column vector of T ones. Π_T is called the centering matrix, $\tilde{\mathbf{X}}$ are the centered frames and $\tilde{\mathbf{K}}^{(\mathbf{x})}$ is the centered kernel matrix. In the following, we assume the series are centered in the feature space and use the notations $\mathbf{K}^{(\mathbf{x})}$ and $\mathbf{X}$ instead of $\tilde{\mathbf{K}}^{(\mathbf{x})}$ and $\tilde{\mathbf{X}}$.

We can compute the DACO kernel using only kernel matrices:

$$\begin{aligned} d_{\text{DACO}}(\mathbf{x},\mathbf{y})^2 &= \text{Tr}\left(\mathbf{N}^T\mathbf{K}\mathbf{N}\mathbf{K}_{+\tau}\right) \\ &= \text{Tr}\left(\mathbf{N}^{(\mathbf{x})^T}\mathbf{K}^{(\mathbf{x})}\mathbf{N}^{(\mathbf{x})}\mathbf{K}^{(\mathbf{x}^\tau)}\right) \\ &\quad +\text{Tr}\left(\mathbf{N}^{(\mathbf{y})^T}\mathbf{K}^{(\mathbf{y})}\mathbf{N}^{(\mathbf{y})}\mathbf{K}^{(\mathbf{y}^\tau)}\right) \\ &\quad -2\ \text{Tr}\left(\mathbf{N}^{(\mathbf{x})^T}\mathbf{K}^{(\mathbf{x},\mathbf{y})}\mathbf{N}^{(\mathbf{y})}\mathbf{K}^{(\mathbf{y}^\tau,\mathbf{x}^\tau)}\right) \end{aligned} \tag{5.10}$$

where $\mathbf{K}_{+\tau}$ is the $(T+T') \times (T+T')$ kernel matrix between all frames of the shifted series $\mathbf{x}^\tau$ and $\mathbf{y}^\tau$, and $\mathbf{N}$ is a matrix defined as:

$$\mathbf{N} = \begin{bmatrix} \mathbf{N}^{(\mathbf{x})} & \mathbf{0} \\ \mathbf{0} & -\mathbf{N}^{(\mathbf{y})} \end{bmatrix}, \quad \mathbf{N}^{(\mathbf{x})} = \frac{1}{\gamma^2 T}\left(\gamma\mathbf{I} - \left(T\mathbf{I} + \gamma^{-1}\mathbf{K}^{(\mathbf{x})}\right)^{-1}\mathbf{K}^{(\mathbf{x})}\right) \tag{5.11}$$

Proof. First, we recall that $\hat{\Sigma}^{(\mathbf{x})} = \frac{1}{T}\mathbf{X}\mathbf{X}^T$ and $\mathbf{K}^{(\mathbf{x})} = \mathbf{X}^T\mathbf{X}$. We use the Sherman-Morrison-Woodbury formula to develop the normalization factors:

$$\begin{aligned} \left(\hat{\Sigma}^{(\mathbf{x})} + \gamma\mathbf{I}\right)^{-1} &= \left(\gamma\mathbf{I} + \mathbf{X}\frac{1}{T}\mathbf{I}\mathbf{X}^T\right)^{-1} \\ &= \frac{1}{\gamma}\mathbf{I} - \frac{1}{\gamma}\mathbf{X}\left(T\mathbf{I} + \frac{1}{\gamma}\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\frac{1}{\gamma} \\ &= \frac{1}{\gamma^2}\left(\gamma\mathbf{I} - \mathbf{X}\mathbf{Q}^{(\mathbf{x})}\mathbf{X}^T\right) \qquad (5.12) \\ \text{where } \mathbf{Q}^{(\mathbf{x})} &= \left(T\mathbf{I} + \gamma^{-1}\mathbf{K}^{(\mathbf{x})}\right)^{-1} \qquad (5.13) \end{aligned}$$

Using the fact that $\hat{\Sigma}_\tau^{(\mathbf{x})} = \frac{1}{T}\mathbf{X}\mathbf{X}_{+\tau}^T$, Equation 5.12 allows us to re-write the auto-correlation as:

$$\begin{aligned} \left(\hat{\Sigma}^{(\mathbf{x})} + \gamma\mathbf{I}\right)^{-1}\hat{\Sigma}_\tau^{(\mathbf{x})} &= \mathbf{X}\mathbf{N}^{(\mathbf{x})}\mathbf{X}_{+\tau}^T \qquad (5.14) \\ \text{where } \mathbf{N}^{(\mathbf{x})} &= \frac{1}{\gamma^2 T}\left(\gamma\mathbf{I} - \mathbf{Q}^{(\mathbf{x})}\mathbf{K}^{(\mathbf{x})}\right) \qquad (5.15) \end{aligned}$$

We then compute the Hilbert-Schmidt norm of the difference between auto-correlation operators by expanding it from the Hilbert-Schmidt inner product (*cf.* Equation 5.6):

$$\begin{aligned} d_{\mathrm{DACO}}(\mathbf{x},\mathbf{y})^2 &= \left\|\mathbf{X}\mathbf{N}^{(\mathbf{x})}\mathbf{X}_{+\tau}^T\right\|_{HS}^2 + \left\|\mathbf{Y}\mathbf{N}^{(\mathbf{y})}\mathbf{Y}_{+\tau}^T\right\|_{HS}^2 \\ &\quad -2\left\langle \mathbf{X}\mathbf{N}^{(\mathbf{x})}\mathbf{X}_{+\tau}^T, \mathbf{Y}\mathbf{N}^{(\mathbf{y})}\mathbf{Y}_{+\tau}^T \right\rangle_{HS} \end{aligned} \tag{5.16}$$

The Hilbert-Schmidt norm of the auto-correlation operator is computed from the trace definition by:

$$\begin{aligned} \left\|\mathbf{X}\mathbf{N}^{(\mathbf{x})}\mathbf{X}_{+\tau}^T\right\|_{HS}^2 &= \mathrm{Tr}\left(\left(\mathbf{X}\mathbf{N}^{(\mathbf{x})}\mathbf{X}_{+\tau}^T\right)^T \mathbf{X}\mathbf{N}^{(\mathbf{x})}\mathbf{X}_{+\tau}^T\right) \\ &= \mathrm{Tr}\left(\mathbf{N}^{(\mathbf{x})^T}\mathbf{X}^T\mathbf{X}\mathbf{N}^{(\mathbf{x})}\mathbf{X}_{+\tau}^T\mathbf{X}_{+\tau}\right) \qquad (5.17) \\ &= \mathrm{Tr}\left(\mathbf{N}^{(\mathbf{x})^T}\mathbf{K}^{(\mathbf{x})}\mathbf{N}^{(\mathbf{x})}\mathbf{K}^{(\mathbf{x}^\tau)}\right) \qquad (5.18) \end{aligned}$$

where Equation 5.17 results from the fact that the trace of products is invariant to circular permutations. The Hilbert-Schmidt inner product in Equation 5.16 is obtained using the same approach. □

5.3.2 Advantages of the kernelized formulation

Our kernel is comparing time series using only between-frame kernel matrices as described in Equation 5.10. Therefore, its complexity depends only on the number of frames, instead of the number of dimensions of the frame features space. Consequently, this results in large speed-ups. Indeed, even when assuming the use of a linear kernel between frames, *i.e.*, that the frame feature space and the input space of finite dimension d coincide, the complexity of directly evaluating the DACO distance between two time series using Equation 5.3 is in $O(d^3)$. This is to expensive to be useful in practice, as frame representations are generally too high-dimensional, *e.g.*, $d = 4000$ in our experiments. However, computing the DACO distance using the kernelized expression in Equation 5.10 leads to a much smaller complexity in $O(T^3)$, where T is the duration of an action, typically of the order of 100 frames.

Furthermore, Junejo et al. [2010] observed that the diagonal blocks of $\mathbf{K}$ are matrices of "temporal self-similarities". A self-similarity matrix has some interesting properties for action recognition, namely its stability with respect to view point changes and its action specific structure. However, Junejo et al. [2010] propose to represent the structure of this matrix by viewing it as an image described using HOG features. Instead, we show that these temporal self-similarities are related to auto-correlation operators in the feature space associated with a frame kernel.

5.4 Experiments

In this section, we evaluate our approach using non-linear SVMs on standard video benchmarks for action classification. First, we introduce the datasets we use. Second, we detail how we model frames and the base kernel between these representations. Third, we describe how we combine a simple aggregation-based kernel with our DACO kernel. Finally, we give the classification results of our approach, and compare it to related and state-of-the-art methods.

5.4.1 Datasets

As dynamic aspects of actions might not be useful to classify every type of action, we investigate the use of our kernel on three state-of-the-art datasets.

The **KTH** dataset [Schüldt et al. 2004] is composed of six human action categories: three similar displacement ones (walking, jogging and running) and three others involving mostly arm motions (boxing, waving and hand-clapping). See Figure 5.3 for an illustration. Note that these actions are periodic. This dataset contains 2391 videos filmed with four different scenarios but with homogeneous and static backgrounds (in most sequences). We use the evaluation protocol of Schüldt et al. [2004]: accuracy averaged over all classes, for a fixed train and test split.

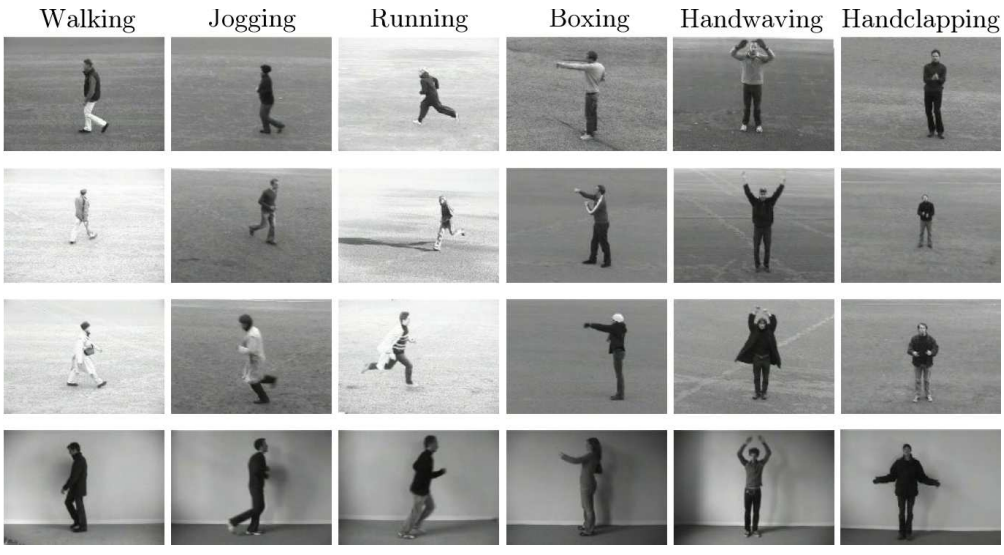


Figure 5.3: *Frames from the 6 action categories of the KTH dataset.*

The **UCF Sports** dataset [Rodriguez et al. 2008] contains ten human actions: bench swinging, diving, golf swinging, high-bar swinging, horse riding, kicking, running, skateboarding, walking, and weight-lifting (*cf.* Figure 5.4). It consists of 150 video samples that show a large intra-class variability. To increase the amount of samples, the dataset is extended with horizontally flipped versions of each sequence. Videos of this dataset are of high resolution and good quality. The evaluation metric is the average leave-one-out accuracy (without considering the flipped versions at test time).

Figure 5.4: *Frames from the 10 action categories of the UCF Sports dataset.*

The **Youtube** dataset [Liu et al. 2009] contains eleven action categories: basketball shooting, biking/cycling, diving, golf swinging, horse riding, soccer juggling, swinging, tennis swinging, trampoline jumping, volleyball spiking, and walking with a dog. See Figure 5.5 for an illustration. This dataset is challenging due to large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background and illumination conditions. Videos are of low resolution, contain shaky camera motion and actions are characterized not only by motion but also by the objects involved and the context in which they are performed. Performance is measured as in [Liu et al. 2009] using "leave-one-group-out" average accuracy.

For all experiments, predictions are obtained by learning non-linear SVMs with the standard "one-against-rest" multi-class approach.

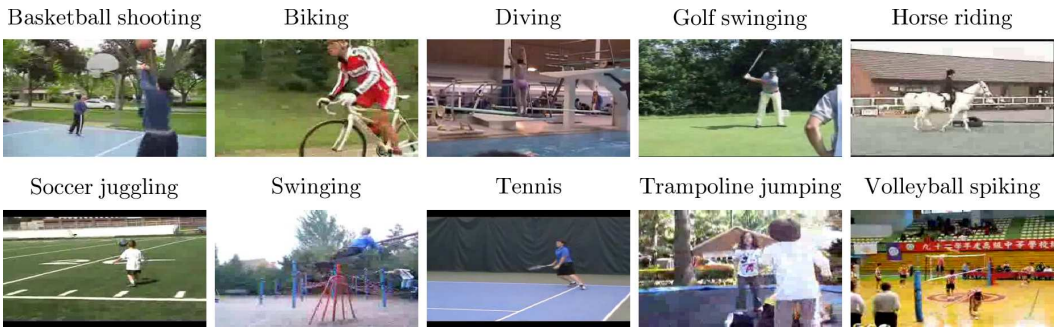


Figure 5.5: *Frames from 10 (out of 11) action categories of the Youtube dataset.*

5.4.2 Frame description and frame kernel

We use the state-of-the-art features recently proposed by Wang et al. [2011]. First, we compute densely sampled *tracklets* (local feature trajectories) from the dense optical flow field between frames. Then, we compute a concatenation of trajectory-aligned local descriptors for each feature track. We use the best descriptor reported in [Wang et al. 2011]: Motion Boundary Histograms (MBH [Dalal et al. 2006], quantized spatial derivatives of the horizontal and vertical components of the optical flow). We use the same track and descriptor parameters as the ones mentioned in [Wang et al. 2011], namely feature tracks of 15 frames, with a dense sampling stride of 5 pixels.

We, then, represent each frame by a BOF. We compute a dictionary of 4000 "visual words" obtained with k-means clustering on a subset of 100,000 randomly sampled features (separately for each dataset). Each tracklet is assigned to a visual word. Each frame is then modeled with the histogram of occurrences of visual words corresponding to tracks passing through this frame. This strategy implies that each tracklet votes for multiple frames. This has a temporal smoothing effect, and makes our frame representations depend on neighboring ones.

We obtain a video representation as a time series of per-frame BOF (*cf.* Figure 5.6). In the rare cases where the histogram of a frame is empty, we replace it by its linear interpolation obtained from neighboring frames. We use the intersection kernel between histograms [Maji et al. 2008] as base kernel between frames, as it provides a good compromise between accuracy and computational efficiency.

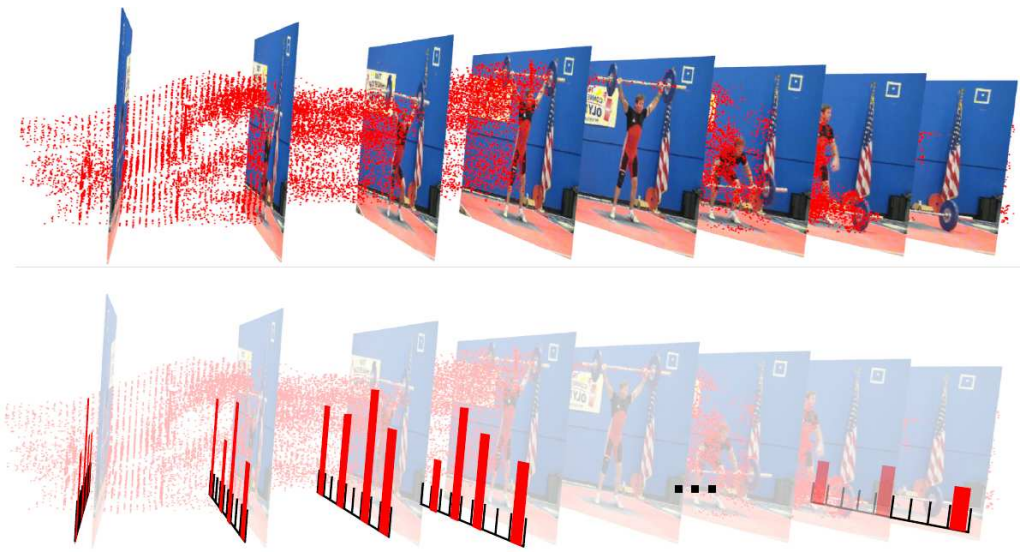


Figure 5.6: *Actions as time series of per-frame BOF over dense tracklets.*

5.4.3 Combination with an aggregation-based kernel

In order to show the complementarity of our kernel with traditionally used aggregation-based kernels, we combine DACO with the *Difference between Mean Elements* (DME) kernel:

$$k_{\text{DME}}(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2} d_{\text{DME}}(\mathbf{x},\mathbf{y})^2\right) ,$$

$$\text{where} \quad d_{\text{DME}}(\mathbf{x},\mathbf{y}) = \|\hat{\mu}_y - \hat{\mu}_x\|_{\mathcal{H}_F} \tag{5.19}$$

This kernel is simply using the difference between the means, $\hat{\mu}_x$ and $\hat{\mu}_y$, of the frames in the feature space $\mathcal{H}_F$. This is related to the traditional BOF approach consisting of aggregating local descriptors computed over the entire video sequence as in [Laptev et al. 2008, Wang et al. 2011], except that the aggregation is performed in the feature space. In practice, the DME kernel is computed using the same kernel matrix $\mathbf{K}$ (before centering) as DACO by:

$$d_{\text{DME}}(\mathbf{x},\mathbf{y})^2 = \mathbf{m}^T\mathbf{K}\mathbf{m} , \quad \mathbf{m} = \Big[\underbrace{-\frac{1}{T},\cdots,-\frac{1}{T}}_{T},\underbrace{\frac{1}{T'},\cdots,\frac{1}{T'}}_{T'}\Big]^T \tag{5.20}$$

In the following section, we report results for the DME kernel, the DACO kernel, and the linear combination of the two noted "DME + DACO". Note, that in the latter case, a single mixing weight between DME and DACO is cross-validated per dataset (instead of per-action).

5.4.4 Results

Table 5.1, Table 5.2, and Table 5.3 contain per-class and average classification accuracies on the three datasets described in Section 5.4.1. We compare our approach to our implementation of a bag-of-features (BOF) baseline over MBH descriptors of tracklets, the state of the art, and the related works.

First, the good performance of DACO on the simple KTH dataset confirms that comparing auto-correlation operators is a valid approach for action recognition. Indeed, both our DACO kernel and the DME+DACO combination achieve a performance above 93%, which is comparable to most of the recently published results. Our approach is only outperformed by Sun et al. [2011] on the KTH dataset. Interestingly, they also rely on self-similarity information, but at a local level. Note, however, that we outperform their approach by +3.4% on the UCF Sports.

Second, we can see that the simple combination of the aggregation-based DME kernel and our auto-correlation-based DACO kernel achieves comparable or better performance than the state of the art. We, indeed, improve by +0.2% on UCF Sports and by +3.7% on the Youtube dataset.

classes	DME	DACO	DME+DACO
boxing	100	100	**100**
handclapping	97.9	92.4	**97.9**
handwaving	**92.4**	91.7	91.7
jogging	**95.8**	92.4	94.4
running	80.6	84.0	**85.4**
walking	100	100	**100**

Schüldt et al. [2004]	71.7
Laptev et al. [2008]	91.8
Niebles et al. [2010]	91.3
Brendel and Todorovic [2010]	94.2
Sun et al. [2011]	**100**
Wang et al. [2011]	94.2
BOF	94.8
DME	94.4
DACO	93.4
DME+DACO	**94.9**

Table 5.1: *Accuracy on the KTH dataset [Schüldt et al. 2004].*

classes	DME	DACO	DME+DACO
bench	100	95.0	**100**
diving	100	100	**100**
golf	**94.4**	88.9	88.9
high bar	100	92.3	**100**
horse riding	75.0	**83.3**	75.0
kicking	85.0	100	**100**
running	53.8	69.2	**76.9**
skateboarding	66.7	50.0	**66.7**
walking	95.5	95.5	**95.5**
weightlifting	100	100	**100**

Rodriguez et al. [2008]	69.2
Wang et al. [2009]	85.6
Kläser et al. [2010a]	86.7
Kovashka and Grauman [2010]	87.27
Weinland et al. [2010]	90.1
Le et al. [2011]	86.5
Sun et al. [2011]	86.9
Wang et al. [2011]	88.2
BOF	86.5
DME	87.0
DACO	87.4
DME+DACO	**90.3**

Table 5.2: *Accuracy on the UCF Sports dataset [Rodriguez et al. 2008].*

classes	DME	DACO	DME+DACO
basketball	66.7	51.4	**72.5**
biking	**92.4**	82.1	91.7
diving	98.1	95.5	**98.1**
golf	**96.5**	95.1	95.1
horse riding	88.3	84.8	**90.4**
soccer	84.0	80.1	**84.6**
swinging	86.9	**87.6**	86.9
tennis	79.6	66.5	**82.0**
trampoline	92.4	89.9	**95.8**
volleyball	94.0	85.3	**94.0**
walking a dog	74.8	52.0	**75.6**

Liu et al. [2009]	71.2
Ikizler-Cinbis and Sclaroff [2010]	75.21
Brendel and Todorovic [2010]	77.8
Le et al. [2011]	75.8
Wang et al. [2011]	84.2
BOF	84.4
DME	86.7
DACO	79.1
DME+DACO	**87.9**

Table 5.3: *Accuracy on the Youtube dataset [Liu et al. 2009].*

Third, our experimental results suggest that the DME and DACO kernels complement each other for the purpose of action recognition. Their combination, indeed, improves performance with respect to just using one of the two kernels. Although the average improvement with respect to DME on the KTH dataset is small (+0.5%), it is more pronounced on the Youtube (+1.2%) and the UCF Sports (+3.3%) datasets. The actions that benefit the most from this combination are:

- the running actions from both the KTH (+4.8% *w.r.t.* DME, +1.4% *w.r.t.* DACO) and the UCF Sports (+23.1% *w.r.t.* DME, +7.7% *w.r.t.* DACO) datasets, which have discriminative dynamics (DACO alone outperforms DME by +3.4%, resp. +15.4%, on KTH, resp. UCF sports);
- the basketball shooting action from the Youtube dataset (+5.8% *w.r.t.* DME, +21.1% *w.r.t.* DACO), which is the most difficult category of Youtube (it has the lowest accuracy with DME), and, therefore, the largest margin for improvement.

Actions that are not improved by the combination of DME and DACO include:

- jogging from KTH (−1.4% *w.r.t.* DME), whose global temporal structure is similar to both the running and walking actions;
- the golfing actions (−5.5% *w.r.t.* DME on UCF Sports, −1.4% *w.r.t.* DME on Youtube), which are characterized by the fast local movement of a golf club instead of the temporal dependencies between frames.
- horse riding from UCF Sports (−8.3% *w.r.t.* DACO), which has a strong temporal structure (DACO alone outperforms DME by +8.3%);

Individually, the averaging DME kernel tends to outperform our dynamics-based DACO although not on all classes. Indeed, the average accuracies of DME and DACO on the UCF Sports are comparable (*cf.* Table 5.2). In addition, some actions are clearly better recognized based on their dynamics (*e.g.*, kicking and running). This can be qualitatively assessed by looking at Figure 5.7 and Figure 5.8. On the one hand, Figure 5.7 displays the kernel matrix $\mathbf{K}$ containing the similarities between the frames of two actions from the same kicking category. The diagonal blocks are the temporal self-similarities $\mathbf{K}^{(\mathbf{x})}$ and $\mathbf{K}^{(\mathbf{y})}$ of the two actions. They clearly share similar temporal dependency patterns. The DACO distance between these videos (0.86) captures this structure information and is smaller than the DME distance (1.31). On the other hand, Figure 5.8 also contains a kernel matrix $\mathbf{K}$, but for two different actions: walking and golf swing. In this case, the discrepancy between the two videos is better captured by DACO with a distance of 2.15, which is larger than the one obtained by DME (0.52).

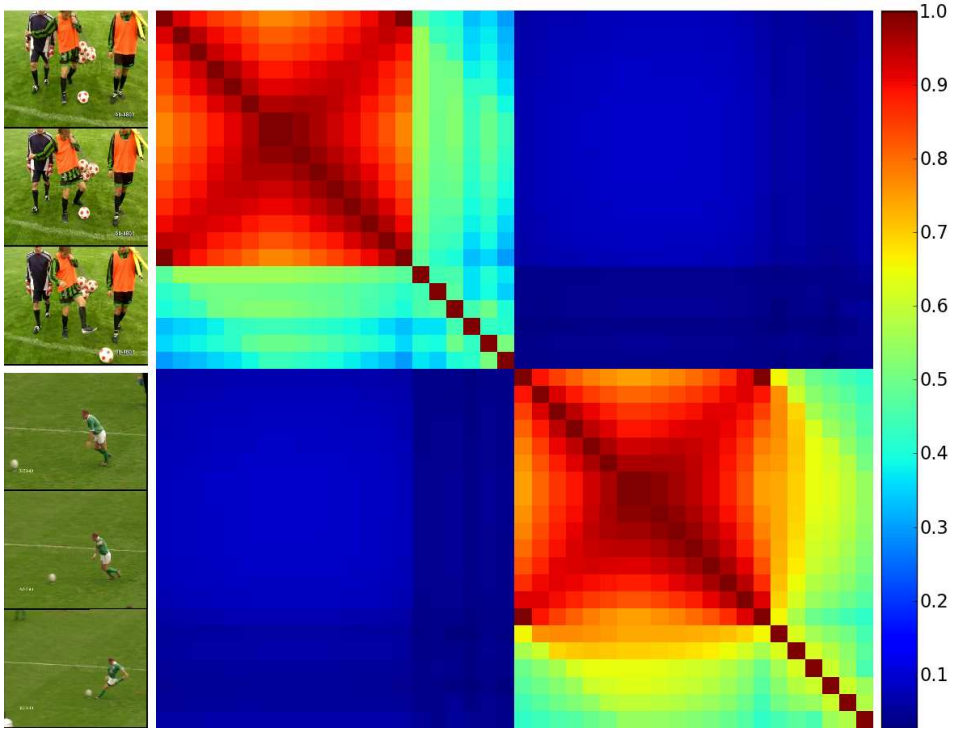


Figure 5.7: *Example of kernel matrix* $\mathbf{K}$ *(cf. Eq. 5.7) between the frames of two similar kicking actions with a small DACO distance (*0.86*).*

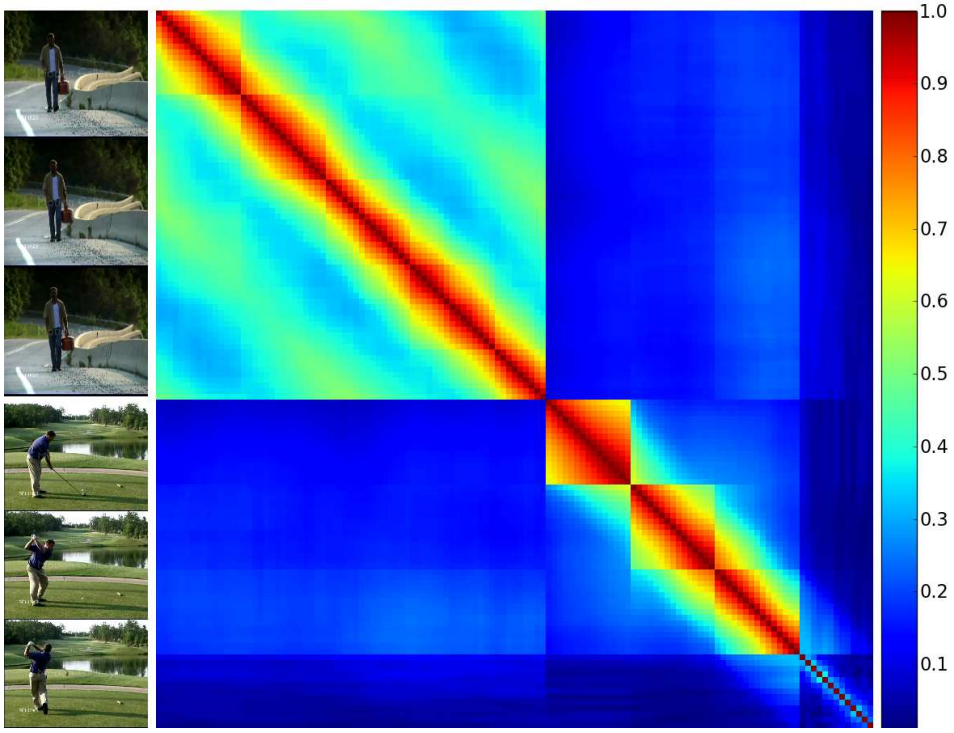


Figure 5.8: *Example of kernel matrix* $\mathbf{K}$ *(cf. Eq. 5.7) between the frames of a walking and golfing actions with a high DACO distance (*2.15*).*

However, most categories are better described by their average local motion statistics, especially on Youtube videos. The results of DACO on this dataset (-7.6% *w.r.t.* DME on average) could be explained by the longer duration of actions in this dataset (approximatively 160 frames on average) compared to the duration of actions in UCF Sports (around 60 frames on average). This suggests that DACO is more suited to short duration, fast actions and is explained by: (i) the small value of the lag we use ($\tau = 1$ frame), (ii) the difficulty to estimate long range temporal dependencies between frames of non-periodic actions, and (iii) longer actions are less likely to be stationary, as they contain more different motions. This suggests a possible improvement of our method by applying the DACO kernel in a temporally localized manner, in order to detect correlated components with a strong temporal structure.

Finally, it seems from our results that the mixing weight between DME and DACO should be optimized independently per binary classification problem in our one-against-rest strategy, instead of jointly over all categories. As stated previously, some actions are, indeed, better characterized by their dynamics, while others are better modeled using average statistics.

5.4.5 Comparison with other kernels on time series

We performed additional experiments with alternative time series kernels and compared their performance to DACO.

Difference between auto-covariances

First, we measured the importance of using the auto-*correlation* over just the auto-*covariance*. To that end, we compared DACO with a kernel using the Hilbert-Schmidt norm of the auto-covariance operators $\hat{\Sigma}_\tau^{(\mathbf{x})}$, instead of the auto-correlations $\hat{\rho}_\tau^{(\mathbf{x})} = \left(\hat{\Sigma}^{(\mathbf{x})} + \gamma \mathbf{I}\right)^{-1} \hat{\Sigma}_\tau^{(\mathbf{x})}$ (Equation 5.1), where $\hat{\Sigma}^{(\mathbf{x})}$ is the covariance operator. Using the same notations as in Section 5.3 and a similar proof to the one of Equation 5.10, we obtain:

$$\left\| \hat{\Sigma}_\tau^{(\mathbf{y})} - \hat{\Sigma}_\tau^{(\mathbf{x})} \right\|_{HS}^2 = \frac{1}{T^2} \mathrm{Tr}\left(\mathbf{K}^{(\mathbf{x})} \mathbf{K}^{(\mathbf{x}^\tau)}\right) + \frac{1}{T'^2} \mathrm{Tr}\left(\mathbf{K}^{(\mathbf{y})} \mathbf{K}^{(\mathbf{y}^\tau)}\right) - \frac{2}{TT'} \mathrm{Tr}\left(\mathbf{K}^{(\mathbf{x},\mathbf{y})} \mathbf{K}^{(\mathbf{y}^\tau,\mathbf{x}^\tau)}\right) \tag{5.21}$$

Note that this expression is similar to the one for DACO in Equation 5.10, but without the normalization matrix $\mathbf{N}$ (Equation 5.11). We found that replacing auto-correlations with auto-covariances yields a noticeable decrease in recognition performance for all actions and across all datasets: -2% on KTH, -6% on UCF Sports, -1% on Youtube. Therefore, normalizing by the covariance of the series allows for more meaningful inter-series comparisons.

Difference between auto-regressive models

Second, instead of using auto-correlation operators, we can directly model an action's temporal dependencies through an auto-regressive (AR) model in the RKHS $\mathcal{H}_F$ associated with the frame kernel:

$$\phi_F(x_t) = \sum_{i=1}^{p} W_i^{(\mathbf{x})} \phi_F(x_{t-i}) + u_t \tag{5.22}$$

where p is the order of the AR model, $W_i^{(\mathbf{x})}$ are operators in $\mathcal{H}_F$, and u_t is a white noise process. Note that this assumes that the action $\mathbf{x}$ is stationary and centered in the feature space (like with DACO). This model can be used to predict (linearly in the feature space $\mathcal{H}_F$) a frame x_t from the preceding p frames by using the *predictive codings* $W_i^{(\mathbf{x})}$. These operators can be obtained by fitting the AR model of $\mathbf{x}$ in Equation 5.22 using kernel ridge regression [Shawe-Taylor and Cristianini 2004], *i.e.*, by minimizing the following cost with respect to the predictive codings $W_i^{(\mathbf{x})}$:

$$\frac{1}{T-p} \sum_{t=p+1}^{T} \left\| \phi_F(x_t) - \sum_{i=1}^{p} W_i^{(\mathbf{x})} \phi_F(x_{t-i}) \right\|_{\mathcal{H}_F}^2 + \frac{\lambda}{p} \sum_{i=1}^{p} \left\| W_i^{(\mathbf{x})} \right\|_{\mathcal{H}_F}^2 \tag{5.23}$$

where λ is a regularization parameter. Solving the dual of this problem leads to the *dual predictive codings* $\mathbf{A}^{(\mathbf{x})}$:

$$\mathbf{A}^{(\mathbf{x})} = \mathbf{X} \left(\mathbf{S}^{(\mathbf{x})} + \lambda' \, \mathbf{I}_{T-p} \right)^{-1} \tag{5.24}$$

where $\lambda' = \lambda(T-p)p^{-1}$ and $\mathbf{S}^{(\mathbf{x})}$ is a kernel matrix between the sub-series of length p of action $\mathbf{x}$:

$$\{(x_{t-p}, \cdots, x_{t-1}) \ , \quad t = p+1, \cdots, T\} \, .$$

As the order p of the AR model is fixed, we use a simple frame alignment kernel, which yields:

$$\left(\mathbf{S}^{(\mathbf{x})}\right)_{t,t'} = \frac{1}{p} \sum_{k=1}^{p} k_F(x_{t-k}, x_{t'-k}) \tag{5.25}$$

where k_F is the previously defined frame kernel (Section 5.3). Note that the kernel matrix $\mathbf{S}^{(\mathbf{x})}$ defined in Equation 5.25 can be obtained from the between-frames kernel matrix $\mathbf{K}$ by simply convolving it with the identity $\mathbf{I}_p$.

We investigated an auto-regressive kernel based on the Hilbert-Schmidt distance between the dual predictive codings $\mathbf{A}^{(\mathbf{x})}$ and $\mathbf{A}^{(\mathbf{y})}$ of two actions $\mathbf{x}$ and $\mathbf{y}$. Directly from the definition of the Hilbert-Schmidt norm (Equation 5.4) and from Equation 5.24, we obtain:

$$d_{\mathrm{AR}}(\mathbf{x},\mathbf{y})^2 = \left\|\mathbf{A}^{(\mathbf{y})} - \mathbf{A}^{(\mathbf{x})}\right\|_{HS}^2 = Tr\left(\mathbf{L}^T\mathbf{K}_{|p}\mathbf{L}\right) \tag{5.26}$$

where $\mathbf{K}_{|p}$ is the kernel matrix between the frames of the series $\mathbf{x}$ and $\mathbf{y}$ truncated of their first p frames, and $\mathbf{L}$ is defined as:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}^{(\mathbf{x})} \\ -\mathbf{L}^{(\mathbf{y})} \end{bmatrix} \quad , \quad \mathbf{L}^{(\mathbf{x})} \;=\; \left(\mathbf{S}^{(\mathbf{x})} + \lambda'\,\mathbf{I}_{T-p}\right)^{-1} \tag{5.27}$$

This distance compares dynamics explicitly modeled by the dual AR coefficients obtained with Equation 5.24. Equation 5.26 shows that it is related to DACO — AR models, indeed, share strong connections with auto-correlations — and that it also relies only on the kernel matrix $\mathbf{K}$ of frame similarities. Nevertheless, their practical formulation differ, and we observed that, in most cases, DACO outperforms the auto-regressive kernel based on the distance in Equation 5.26. Indeed, although this kernel obtains the highest accuracy (93.5%) on the short actions of the UCF sports dataset, it only reaches 86.5% on KTH (-6.9% *w.r.t.* DACO) and 55.1% on Youtube (-24% *w.r.t.* DACO). These results suggest that DACO is more robust to the violation of the stationarity assumption, and, thus, applicable to a wider range of actions.

Global alignment kernel

Finally, we evaluated recognition results with the global alignment kernel of Cuturi [2011], which extends the well-known Dynamic Time Warping algorithm by computing the soft-max of all possible alignments. It obtained the average accuracy of 90.1% on the UCF Sports dataset, which is comparable to the DME+DACO combination. This good performance can be explained by the fact that actions of the same category tend to be of very similar duration and time-synchronized in this dataset. However, the good performance of this alignment kernel did not transfer to the other datasets where it obtained a much lower performance than DACO (70.2% on KTH, 57.6% on Youtube). Two factors can explain these results. First, the videos in these datasets are clearly not synchronized, and there is a significant intra-class duration variability. Second, some pairs of categories, *e.g.*, running and jogging in KTH, can be aligned in spite of their differences.

5.5 Conclusion

We introduced a new kernel on videos, the *Difference between Auto-Correlation Operators* (DACO) kernel, specifically designed for action recognition. It compares the dynamics of actions, represented as time series of frames, by using a distance between auto-correlations. It can be efficiently computed using only inner products between frames. We show that, even if not all actions exhibit characteristic temporal relationships between frames, the dynamic information extracted by auto-correlations can complement state-of-the-art distribution-based kernels that average visual information over frames.

Chapter 6
Conclusion

Our work investigated *structured collections of local spatio-temporal features* in order to build action models that are both accurate and robust to real-world video conditions. Our experiments suggest that recognition performance on challenging video sources can be improved by bridging the gap between rigidly structured global representations and orderless aggregations of local descriptors. This dissertation has presented several structured models adapted for the recognition of both simple actions and more complex activities in uncontrolled scenarii.

In the following, we summarize our key contributions and the main observations derived from our experiments (Section 6.1). We also indicate some potential research directions that we deem interesting for future work on action recognition in real-world videos (Section 6.2).

6.1 Summary of contributions

Action detection with Actom Sequence Models

Our first contribution is a *temporally structured extension of bag-of-features* for the task of *action localization* in long, real-world, unsegmented videos. We studied the decomposition of an action into a flexible sequence of meaningful temporal parts termed *actoms*, and proposed a robust representation called the *Actom Sequence Model*. We also introduced a flexible model of the temporal structure of an action. Relying on this generative model, we proposed a *sliding-central frame detection approach*, which is more principled and more accurate than traditional sliding window techniques. We validated our approach on movie data, showing that our method outperforms both unstructured and rigidly structured models, thanks to its flexibility and robustness. In addition, our experiments highlighted that marginalizing over a learned prior of the temporal structure significantly improves upon traditional multi-scale sampling heuristics, even in a typical sliding window context where the structure is simply the action's temporal extent.

Activities as cluster-trees of tracklets

Our second contribution pertains to the modeling of *complex activities* using *tree-structured video representations*. We proposed a large-scale divisive spectral clustering algorithm able to automatically organize a collection of dense *tracklets*, *i.e.*, local point trajectories. Our algorithm organizes the motion content of a video in a hierarchical representation, called *cluster-tree*. Contrary to existing methods, we neither select an arbitrarily fixed number of spatio-temporal parts (nodes in the cluster-tree), nor build a global model of a category such as its average cluster-tree. Instead, we use the full decomposition of a video to represent an action with nested histograms forming our *BOF-Tree* model. We introduced a kernel that can efficiently compare the content of two unordered binary trees of variable sizes by relying on the inclusion relations encoded by the tree's edges. We conducted experiments on challenging benchmarks for activity recognition and showed that our non-linear classifier on per-video hierarchies yields state-of-the-art results, improving upon structured learning methods and latent part models. We also showed that our clustering algorithm yields more discriminative decompositions than unrelated clusters obtained by k-means and spectral clustering, as well as decompositions from other hierarchical clustering algorithms.

Modeling action dynamics

Our third contribution consists in studying how action dynamics can complement the state-of-the-art models relying on average statistics about local features. We proposed to represent actions as time series of frame descriptors, and described a model of an action's temporal dependencies over time: the auto-correlation operator in a frame feature space. In addition to its statistical properties, we also motivated the choice of this operator as a model of action dynamics by showing its connection with temporal self-similarities. We formally derived this connection from the dual expression of the Hilbert-Schmidt distance between the auto-correlation operators of two actions. Furthermore, we relied on this dual formulation in order to provide an efficient kernel, called the Difference between Auto-Correlation Operators (DACO) kernel. It compares action dynamics by computing the similarity between the respective auto-correlation operators of two actions. In order to evaluate our approach, we conducted experiments on a variety of datasets, ranging from short simple periodic actions to complex ones in Youtube videos. In all cases, we showed that the performance of methods averaging over frames is improved by the combination with our kernel on dynamics. This emphasized that, for action recognition, comparing content structure (*e.g.*, an action's temporal dependencies) is complementary with comparing the contents themselves.

6.2 Perspectives for future research

In this section, we give some possible extensions of our work, which are suggested by our experiments and the recent progress of the computer vision and machine learning fields.

6.2.1 Extensions of ASM

A first improvement of our ASM model and of our sliding central frame approach consists in adapting them for the task of **spatio-temporal localization**, *i.e.*, returning a temporal window and a sequence of bounding boxes around the actors performing the action of interest. In order to localize when and where an action is performed, one could apply the temporal localization approach in Chapter 3 inside tracks obtained by pre-trained human and object detectors [Kläser et al. 2010a,b, Prest et al. 2011]. As mentioned above, instead of independently localizing first and then classifying, one could jointly model spatio-temporal structure and appearance, for instance with the deformable part model of Felzenszwalb et al. [2010]. One of the main difficulties, however, lies in how to incorporate geometrical aspects while maintaining the robustness of our models under real-world video conditions, especially when facing strong and time-varying viewpoint variations, or in the presence of occlusions. Poselets [Bourdev and Malik 2009] are an interesting approach to handle the aforementioned challenges in static images. These pre-trained detectors of local 3D poses have successfully been applied to action recognition in static images [Maji et al. 2011]. Building on these ideas and our work on actoms, one could design detectors of "actionlets", *i.e.*, iconic local dynamic poses, which encode more information than just motion, *e.g.*, the nature of the moving object or body part.

In Chapter 3, ASM models rely on a fixed number of actoms per category. This limitation needs to be removed and our approach extended in order to **handle sequence models with a variable number of parts**. Allowing action examples of the same category to have different numbers of actoms is, indeed, likely be necessary in order to model a wide range of actions, *e.g.*, due to temporal occlusions, intra-class variability, or differences between annotators on how to decompose the action. Comparing sequences with different numbers of actoms while preserving temporal ordering can be done by relying on dynamic programming strategies, *e.g.*, by adapting string kernels to compute an "actom edit distance", or with the Global Alignment Kernel [Cuturi et al. 2007]. One can also use the Fisher kernel [Jaakkola and Haussler 1999] with respect to a sequential generative model, *e.g.*, a HMM, in order to obtain a fixed-length representation based on a fixed number of

hidden states. In addition, estimating the variable-sized temporal structure of an action requires a more complex probability distribution than a histogram on inter-actom distances as done in Chapter 3. A straightforward extension of our approach would consist in using two temporal distributions instead: one for the number of temporal parts, and another conditional distribution on the actom placements knowing the number of actoms. However, this makes strong independence assumptions that may not hold in practice. Ideally, both the content, the number, and the positions of the actoms should be estimated jointly, instead of independently as done in Chapter 3. This is of particular importance when aiming to detect multiple actions simultaneously.

Weakly supervised training. In Chapter 3, we demonstrated that action models can be significantly improved by learning from a meaningful manually labeled decomposition instead of just a temporal window. This supervision has, however, a cost that needs to be minimized in order to scale up to a large number of action categories. In addition to semi-supervised learning approaches, a promising direction of research consists in automatically refining inaccurate annotations (also called weak supervision) coming from cheap or readily-available additional sources of information, *e.g.*, transcripts of movies and TV shows [Gaidon et al. 2009, Laptev et al. 2008]. Promising results have been obtained in order to refine temporal extent using discriminative methods [Duchenne et al. 2009, Satkin and Hebert 2010]. These approaches could be generalized to infer latent actom locations from rough temporal extent annotations. Note that the approach of Duchenne et al. [2009] shares several similarities with recent approaches for image co-segmentation. For instance, Joulin et al. [2012a] rely on multiple images and their global category labels as weak supervision for segmentation via discriminative clustering. Therefore, one could adapt the related weakly supervised learning algorithms (*e.g.*, [Joulin et al. 2012b]) in order to infer the latent spatio-temporal structure of actions from a set of loosely labeled videos.

6.2.2 Extensions of BOF-Trees

Our tree model of activities only accounts for inclusions between parts from a cluster-tree. This could be improved by **modeling more complex structural relations between motion parts**, *e.g.*, spatial, temporal, and hierarchical relations as in [Brendel and Todorovic 2011]. However, defining an overly complex latent structure leads to difficulties in learning and inference, which eventually results in sub-optimal performance. Indeed, the additional flexibility in the latent variables makes the parameter space too vast for it to be efficiently explored, and may lead to overfitting. Overcoming these issues — *e.g.*, by finding a good compromise between unconstrained structure learning

and fixed decompositions — will improve recognition, thanks to the discriminative learning of the model, while ensuring greater robustness to real-world videos by modeling the uncertainty in the structure of the decompositions.

Instead of only learning the parts, our decomposition method described in Chapter 4 could be improved by also **modeling the latent importance of each part** for a category. Our tree structure is, indeed, obtained without using the action category information. Furthermore, all nodes are treated equivalently in our ATEP kernel (Section 4.3.3), whereas some sub-trees might be unrelated to the action of interest (*e.g.*, background motion). So far, these extraneous sub-trees are treated as noise, which is handled by the robustness of our kernel thanks to its averaging effect. Our method could be improved by taking a more explicit approach and learning category-specific importance of sub-trees based on their content, structure, and position in the cluster-tree. Node-specific importance factors can, indeed, be directly integrated in our ATEP kernel as simple multiplicative weights in the edge-to-edge comparisons. These weights could be learned by treating the importance of each node as a latent variable, and using adequate inference mechanisms. As mentioned previously, jointly co-clustering actions from the same category or from all categories at the same time could be used to leverage category labels in order to obtain better activity decompositions. Note that such part-based weighting might also be applicable to actoms, as some temporal parts are more important than others (*e.g.*, the middle actom in our ASM models).

Finally, another extension consists in adapting our method to allow for **spatio-temporal localization within BOF-Trees**. The approach described in Chapter 4 is, indeed, specifically tailored for videos focusing on a single activity. When the video is only temporally segmented and contains several co-occurring activities, we can adapt our method to operate only on parts of the video by individually matching sub-trees instead of comparing the whole cluster-trees. In addition to existing efficient sub-tree matching algorithms (*cf.* [Valiente 2002]), we could also adopt a greedy coarse-to-fine matching approach in order to speed up computations. By relying on the additive property of BOF-Trees, we can indeed, approximate a sub-tree by its top-level (root, child) edge, as done in our ATEP kernel (Section 4.3.3). Therefore, we could match only the edges of BOF-Trees, and traverse the tree in a top-down fashion akin to a branch-and-bound strategy.

Applying our method to long video streams as those studied in Chapter 3 is, however, less straightforward. Indeed, as we build our tree using a divisive top-down strategy, our approach assumes that all tracklets are available at once. Therefore, it seems that our clustering algorithm must be preceded by an on-line temporal segmentation strategy — *e.g.*, using a shot detector for

edited videos such as movies, or a more generally applicable change point detection algorithm such as [Harchaoui et al. 2009] — in order to first find the roots of the cluster-trees to be built.

6.2.3 Extensions of DACO

Combining DACO with ASM. As suggested by our experiments in Chapter 5, our DACO kernel is particularly suited to short actions, and improves the performance of aggregation-based kernels. Therefore, we could improve the actom-to-actom comparisons in ASM by not only computing the similarity between actom contents, but also by comparing their dynamics using DACO. Indeed, the shorter duration of actoms means that they can be considered as almost stationary time-series of frames. Furthermore, the per-frame weights derived from the actom models can be directly integrated in DACO by weighting the projections of the per-frame BOFs in the RKHS associated with the frame kernel.

Short-time DACO. As our results on the Youtube dataset seem to indicate (*cf.* Section 5.4.4), the main limitation of our DACO kernel pertains to its stationarity assumption, which is often violated for long and complex actions. Therefore, a natural extension of our approach consists in computing a short-time version of DACO, *e.g.*, by successively convolving the frames with a temporal Gaussian window of a short duration. Shorter time series can indeed be considered as almost stationary, as there is simply less observations to notice a significant change in the distribution of frames. The problem with this approach, however, is that the dynamics of the series is modeled by a sequence of local auto-correlation operators instead of single global one. Consequently, another sequence kernel must be used in order to pool the local window-to-window comparisons obtained with DACO. Note that applying this idea to compute a windowed version of the auto-regressive kernel described in Section 5.4.5, results in an interesting representation of the video signal, which shares similarities with the linear predictive codings that are widely used in audio and speech processing.

Time series analysis of actions. Most of the tools built to analyze time series are designed for univariate data such as audio signals or the evolution of stock prices. Videos, however, rely on high-dimensional representations. Therefore, the analysis of their temporal properties — *e.g.*, periodicity — requires different techniques adapted to these complex multivariate signals. For instance, Cutler and Davis [2000], more than a decade before us, used time-frequency analysis on temporal self-similarities to detect periodic motion. In Chapter 5, we showed that the auto-correlation operator in a frame RKHS

is a mathematical object that can be used as a proxy to study the temporal properties of actions. We believe that this formal framework, although embryonic, can pave the way for more sophisticated analysis of actions by generalizing the numerous tools that already exist for such tasks in univariate settings such as in finance, or speech and audio processing. For instance, we could extend the notion of correlogram to build an "action correlogram" by using the Hilbert-Schmidt norm of the auto-correlation operator in order to measure the randomness of an action, *e.g.*, for abnormal behavior detection.

Publications

This thesis has led to several publications summarized below.

International conferences

- A. Gaidon, M. Marszalek, C. Schmid.

  Mining visual actions from movies

  Proceedings of the British Machine Vision Conference (BMVC), September 2009.

- A. Gaidon, Z. Harchaoui, C. Schmid.

  Actom Sequence Models for Efficient Action Detection

  Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2011.

- A. Gaidon, Z. Harchaoui, C. Schmid.

  A Time Series Kernel for Action Recognition

  Proceedings of the British Machine Vision Conference (BMVC), September 2011.

- A. Gaidon, Z. Harchaoui, C. Schmid.

  Recognizing Activities with Cluster-trees of Tracklets

  Proceedings of the British Machine Vision Conference (BMVC), September 2012.

Journal submissions

- A. Gaidon, Z. Harchaoui, C. Schmid.

  Action Detection with Actom Sequence Models

  Submitted (major revision) to the IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), April 2012.

Other publications

- H. Harzallah and C. Schmid and F. Jurie and A. Gaidon.

  Classification aided two stage localization

  PASCAL Visual Object Classes Challenge Workshop, in conjunction with ECCV, October 2008.

- M. Douze, A. Gaidon, H. Jégou, M. Marszalek, C. Schmid.

  INRIA-LEAR's video copy detection system

  Proceedings of the TREC Video Retrieval Evaluation (TRECVID Workshop), December 2010.

Annexe A

Résumé

Les quatre sabots d'un cheval au galop se retrouvent-ils à un moment donné tous en l'air ? La réponse à cette question est souvent considérée comme la première étape du développement de la vidéo. En 1872, le photographe Eadweard Muybridge fut chargé par Leland Stanford de donner une réponse définitive à cette question longtemps débattue. A cette allure — environ 58 kilomètres par heure, soit 36 miles par heure — les sabots d'un cheval sont trop rapides pour permettre à l'oeil humain de les suivre : une preuve scientifique basée sur la photographie était nécessaire pour répondre. Après plusieurs expériences préliminaires, Muybridge apporta une réponse affirmative à la question, avec sa célèbre étude "Sallie Gardner at a Gallop" (*cf.* Figure A.1). La principale difficulté fut d'élaborer une installation capable de prendre une succession d'images capturant les détails du mouvement malgré sa vitesse. Muybridge réussit à prendre une série de photographies le 19 juin 1878, à la ferme de Stanford à Palo Alto, en plaçant 24 appareils photo le long de la piste. L'obturateur de chaque appareil était déclenché par un fil de détente au passage du cheval.

Ce travail fut un prélude au développement de l'analyse visuelle du mouvement, qui consiste à étudier les propriétés visuelles des systèmes dynamiques, comme les fluides et les humains, de manière scientifique et précise. Le travail de Muybridge montra que l'évolution de la photographie permettait de

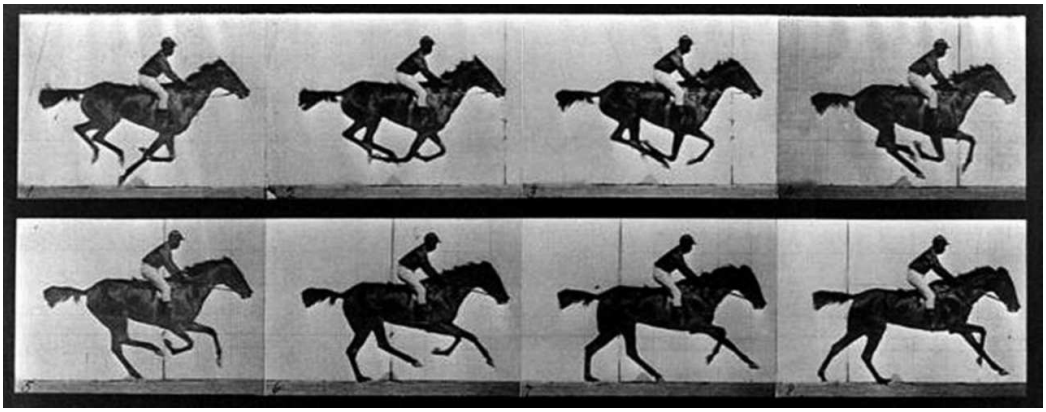


Fig. *A.1:* *Étude de Muybridge "Sallie Gardner at a Gallop" qui prouva qu'à un moment donné, les quatre sabots d'un cheval au galop sont tous en l'air.*

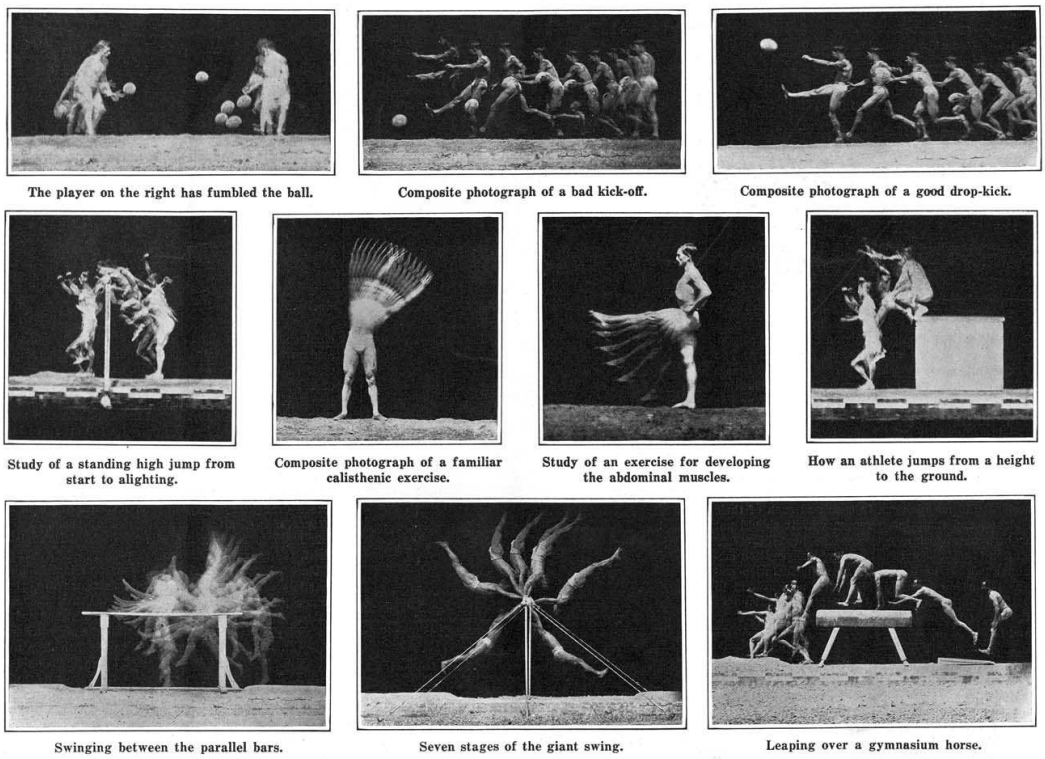


Fig. *A.2:* *"The Human Body in Action", Etienne-Jules Marey, 1914.*

prendre une succession d'images afin de de décrire une scène dynamique réaliste. Les séquences d'images — comme les chronographes d'Etienne-Jules Marey de la Figure A.2 — permettent aux scientifiques d'analyser les mouvements complexes, tels que les actions, en regardant l'évolution temporelle des acteurs et des objets participant.

Le développement de l'analyse vidéo est indissociable du progrès des appareils d'enregistrement et de projection. Pendant longtemps, un grand nombre d'appareils permettaient de montrer des mouvements à partir de séries de dessins. Ces appareils se basent sur l'illusion d'optique — découverte en 1912 par Max Wertheimer — appelée "Beta mouvement", par laquelle le cerveau combine une succession d'images fixes pour former une impression de mouvement. Le Zoetrope — "roue de la vie" en grec — est un exemple d'un dispositif populaire produisant l'illusion de mouvement à partir d'une succession rapide d'images placées à l'intérieur d'un cylindre tournant muni de fentes verticales jouant le rôle d'obturateur.

Les premiers vidéo-projecteurs combinaient les mêmes idées avec les progrès de la photographie — attribués à Daguerre and Niépce (1839). Par exemple, Muybridge inventa le premier projecteur de film — le Zoopraxiscope (1879) — dans le but de présenter ses photographies "stop-action". Cet

appareil reposait sur la projection rapide d'une succession d'images depuis des disques en verre tournants. Peu après, Auguste et Louis Lumière brevetèrent le Cinématographe — à la fois caméra, projecteur et révélateur de film — en 1895 et commencèrent à produire les tous premiers films jamais réalisés.

Depuis lors, la vidéo est devenue un des médias les plus populaires pour la communication et le divertissement. En 2010, 441 milliards de vidéos étaient accessibles en ligne, ce qui correspond approximativement à 10^{21} pixels[1].

En 2011, le site web YouTube avait plus de mille milliards de vues, et plus de 800 millions de visiteurs uniques regardaient plus de 3 milliards d'heures de vidéos par mois. La création de contenu est également en pleine explosion, avec une nouvelle heure de vidéo mise en ligne sur YouTube chaque seconde[2]. Fin 2012, la vidéo représentera plus de 50% du trafic internet, avec un taux de croissance annuelle de 48%[3].

La vidéo comprend un large éventail de sources et d'applications : la vidéosurveillance, le contrôle industriel, la robotique (comme les véhicules autonomes), la domotique, les systèmes d'interaction homme-machine, les jeux vidéo (comme le Kinect de Microsoft pour la console de jeux Xbox), les plateformes de contenu internet (comme YouTube et DailyMotion), les séries TV et les films.

Le but commun à toutes ces applications est de capturer et transmettre les informations visuelles d'une scène réaliste sur une certaine période de temps. Le principal challenge auquel les utilisateurs de ces technologies font actuellement face est de trouver le contenu qu'ils recherchent parmi la quantité massive de données disponibles. En raison de l'énorme quantité de vidéos existantes, répondre à cette question nécessite des outils pour automatiser l'analyse, l'indexation et l'organisation de leur contenu.

Notre travail se concentre sur la reconnaissance d' *actions*, l'une des informations visuelles les plus présentes dans les données vidéo. La vision par ordinateur a fait récemment des progrès significatifs dans le domaine de la reconnaissance d'actions — notamment pour des tâches particulières comme la vidéosurveillance. Néanmoins, les méthodes de l'état de l'art actuel sont encore loin de pouvoir comprendre toute action dans n'importe quel type de vidéo. Ceci peut être mesuré par la faible performance des systèmes impliqués dans la tâche Multimedia Event Detection (MED) de TRECVID [Smeaton et al. 2006] — une compétition d'interprétation automatique de vidéo. Pour l'édition de 2011 [Over et al. 2011], les systèmes soumis avaient une moyenne de 6% de faux positifs, avec 54% de détections manquées, sur des actions simples telles que "préparer un sandwich" et "brosser un animal".

[1] http://www.comscore.com
[2] http://www.youtube.com/t/press_statistics
[3] http://www.cisco.com

Fig. *A.3:* *Exemples d'actions tirées de vidéos de la base "Hollywood 2" [Marszalek et al. 2009] utilisé dans nos expériences.*

A.1 Objectifs

L'objectif de notre travail est de reconnaître des actions génériques effectuées dans des vidéos réalistes (*cf.* Figure A.3 pour une illustration). Nous avons pour but de concevoir des modèles robustes, capables de représenter une grande variété de catégories d'actions, tout en tenant compte de la variabilité visuelle de vidéos réalisées dans des conditions non contrôlées.

La reconnaissance d'actions en conditions non contrôlées fait face à de nombreuses difficultés inhérentes à la complexité des actions et des vidéos.

Plusieurs challenges sont liés à la **variabilité intra-classe** : les actions d'une même catégorie peuvent différer significativement en terme d'apparence et de mouvement. Les sources de variation incluent le bruit, les conditions d'éclairage, l'échelle et la forme des entités participantes, le fouillis de l'arrière-plan, le large éventail de points de vue possibles, les occlusions, les mouvements rapides, complexes et articulés, le flou du mouvement, le mouvement de la caméra, la variabilité de durée, et le style d'exécution. À cause de ces variations, les modèles d'action peuvent difficilement se reposer sur des hypothèses simplificatrices, comme par exemple supposer que la totalité du corps des acteurs est toujours entièrement visible. Nous tenons compte de cette variabilité

en utilisant des caractéristiques locales robustes, et en capturant des propriétés structurelles des actions qui sont essentielles à leur reconnaissance. Un autre ensemble de challenges résulte de la **confusion entre classes**. En effet, la distinction est souvent subtile entre des actions de classes différentes. Les actions courir et marcher partagent par exemple des mouvements similaires. Une analyse minutieuse du mouvement est donc nécessaire pour distinguer ces deux catégories. La modélisation précise des spécificités d'une action est particulièrement importante en conditions non contrôlées, où un algorithme de reconnaissance doit distinguer les actions intéressantes parmi tous les autres événements possibles. Une fois encore, nous nous attaquons à ce problème en utilisant la structure des actions pour construire et comparer des modèles d'action plus détaillés.

La reconnaissance d'actions comprend deux tâches principales : la classification et la localisation. D'une part, la classification a pour but d'assigner une étiquette sémantique à une vidéo focalisée sur une action. D'autre part, l'objectif de la localisation est de proposer une étiquette sémantique ainsi que son étendue spatio-temporelle, pour des vidéos pouvant contenir plus d'une action. En pratique, ces deux problèmes sont étroitement liés, dans la mesure où les méthodes de localisation consistent souvent à appliquer un classifieur à différentes positions afin de déterminer les plus probables. Dans les deux cas, nous reconnaissons des *catégories d'action*, et nous étudions des algorithmes d'*apprentissage supervisé* pour résoudre ces problèmes : étant donné une liste de catégories d'actions, nous apprenons un modèle en utilisant des exemples de vidéos correspondant à chaque catégorie.

Comme dit précédemment, nous supposons qu'une action est caractérisée par des relations spatio-temporelles entre sous-événements. Nous affirmons et vérifions expérimentalement qu'extraire cette information structurelle — et notamment les structures temporelles — est particulièrement important pour la reconnaissance d'action, afin d'être robuste aux conditions non contrôlées des vidéos, et de pouvoir lever l'ambiguïté entre des catégories similaires. Notre objectif principal est d'apprendre de meilleurs modèles d'action en découvrant et en utilisant cette structure. La difficulté majeure repose sur comment identifier, représenter et intégrer avec des techniques d'apprentissage statistique, les aspects structurels pertinents d'une action. Nous étudions différentes possibilités : des séquences d'actions atomiques, des décompositions hiérarchiques du mouvement, et des dépendances temporelles entre images.

FIG. *A.4: Image du film de Jackie Chan "The Legend of Drunken Master". Cette image seule ne suffit pas pour comprendre le contenu de la scène. Il est nécessaire de considérer l'évolution temporelle des différents acteurs pour pouvoir déterminer qu'il s'agit d'une scène de combat dans laquelle le personnage central est attaqué alors qu'il boit du vin à la bouteille.*

A.2 Contexte

La compréhension automatique de vidéos est un objectif fondamental de la vision par ordinateur. Comme mentionné précédemment, les actions contiennent des informations indispensables à la compréhension d'une scène visuelle dynamique telle que celle décrite dans la Figure A.4. En effet, même si les actions de certains personnages peuvent être déduites de cette seule image, il est plus difficile de déterminer ce que fait la personne au centre. La nature des objets et la pose des acteurs sont difficiles à comprendre et ne permettent pas — par eux-mêmes — de déterminer quelles actions sont réalisées : il est nécessaire d'utiliser l'information de mouvement sur une séquence d'images. De plus, il est nécessaire d'analyser les relations entre ces mouvements pour reconnaître qu'il s'agit d'une scène de combat de Kung-Fu.

Afin que les ordinateurs puissent faire preuve d'un tel niveau d'interprétation, ils ont besoin de modèles qui représentent les différents mouvements, leurs relations et l'évolution temporelle des entités participantes (objets, parties du corps) au cours d'une action. Par exemple, les Motion History Images (MHI) de Bobick and Davis [2001] sont un modèle précurseur utilisé pour représenter une action comme l'évolution temporelle d'une silhouette (*cf.* Figure A.5). Cependant, cette approche et ses extensions — comme les formes spatio-temporelles de Gorelick et al. [2007] ou les Motion History Volumes (MHV) de Weinland et al. [2006] — manquent de robustesse et sont limitées à des scenarii contraints, tels que la vidéosurveillance [Hu et al. 2004] ou les interfaces homme-machine [Shotton et al. 2011]. En effet, elles reposent sur des informations difficiles à obtenir pour des vidéos sous conditions non contrôlées,

Fig. *A.5:* *Les Motion Energy Images (MEI) et Motion History Images (MHI) de Bobick and Davis [2001].*

comme par exemple l'élimination d'arrière-plan, l'extraction de silhouettes ou l'estimation de modèles paramétriques du corps humain.

Depuis peu, la communauté de vision par ordinateur se penche désormais sur des vidéos plus difficiles. Les données vidéo non contrôlées sont caractérisées par une grande variabilité et l'absence de connaissance a priori sur (i) le processus d'enregistrement (comme le point de vue, le mouvement de la caméra, la qualité de la vidéo et la résolution), (ii) la scène (en intérieur ou en extérieur, bondée ou non, arrière-plan en mouvement ou encombré, occlusions, conditions d'éclairage), et (iii) les actions réalisées (mouvements rapides ou lents, interactions). Ces données englobent un large éventail de vidéos issues d'un grand nombre de sources potentielles, comprenant les vidéos amateur, les reportages, les diffusions sportives, les émissions télévisées et les films. Les catégories d'actions présentes dans les données de test pour la reconnaissance d'actions sont elles aussi de plus en plus variées : elles comprennent des actions simples (comme courir) et plus complexes (comme sortir de sa voiture), des interactions (comme embrasser) et des activités (comme se battre).

Afin de résoudre les problèmes mentionnés précédemment, plusieurs chercheurs [Chomat and Crowley 1999, Dollár et al. 2005, Laptev 2005, Schüldt et al. 2004, Zelnik-Manor and Irani 2001] ont proposé d'utiliser des caractéristiques visuelles locales. Ce sont en général des extensions spatio-temporelles de détecteurs de points d'intérêt en 2D et de descripteurs utilisés avec succès pour la reconnaissance d'objets dans des images (voir par exemple les résultats des compétitions PASCAL VOC [Everingham et al. 2010]). Les caractéristiques locales de vidéo, telles que les points d'intérêt spatio-temporels [Laptev 2005], décrivent l'apparence (comme avec les histogrammes de gradients orientés [Dalal and Triggs 2005]) ou le mouvement (comme avec les histogrammes de flot optique [Laptev et al. 2008]) de volumes spatio-temporels locaux. Grâce à la robustesse des descripteurs locaux et à leur absence d'hypothèses globales (sur les relations géométriques par exemple), les caractéristiques locales ont

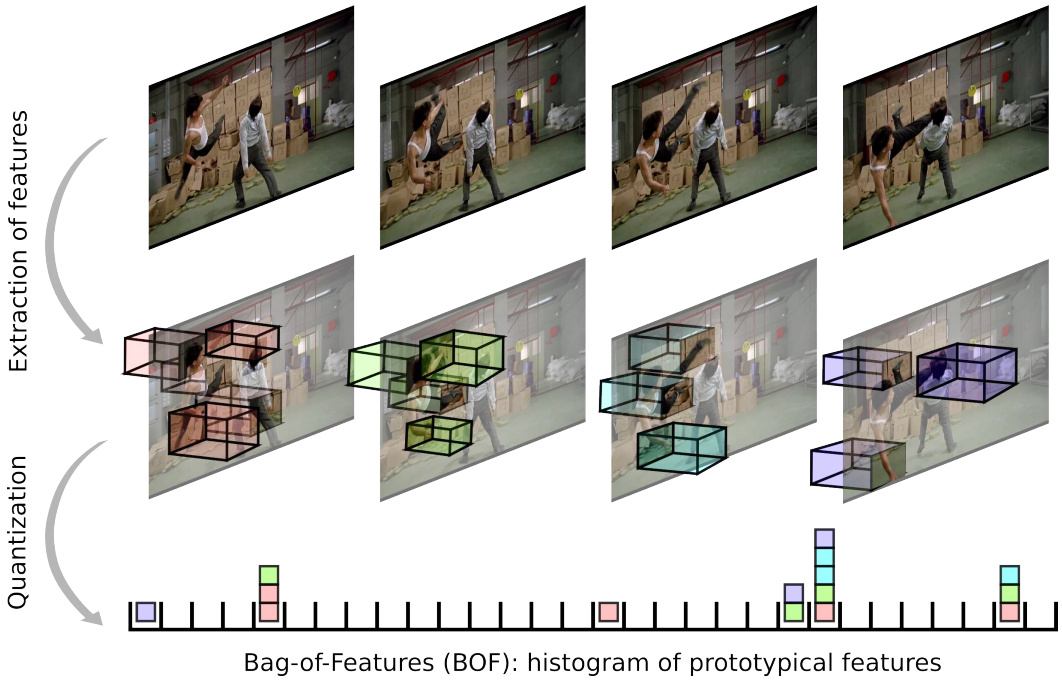


Fig. A.6: *Bag-of-features : un modèle robuste qui agrège les caractéristiques spatio-temporelles locales en les quantifiant sur un vocabulaire de caractéristiques prototypiques appelées "mots visuels", par analogie avec le modèle de texte par "sac de mots".*

été utilisées avec succès pour des tâches difficiles de reconnaissance d'actions dans des vidéos réalistes.

Après leur extraction, une action est ensuite décrite par une collection de caractéristiques locales trouvées dans la vidéo correspondante. Le bag-of-features (*cf.* Section 2.2.3) est un modèle populaire, utilisé pour représenter un ensemble de descripteurs spatio-temporels locaux. Il vise à quantifier ces caractéristiques locales en assignant à chacune d'entre elles le "mot visuel" — une caractéristique prototypique — le plus proche, et à construire ensuite l'histogramme constitué des occurrences de chaque mot visuel apparaissant dans la vidéo (*cf.* Figure A.6 pour une illustration). Comme ce modèle non structuré ignore les relations entre les caractéristiques locales, plusieurs extensions ont été proposées — comme par exemple les pyramides spatio-temporelles [Laptev et al. 2008] — afin d'incorporer de l'information de structure.

Les modèles d'action basés sur les caractéristiques locales sont souvent en grande dimension (typiquement de l'ordre de 10^4 dimensions). De plus, les représentations telles que bag-of-features sont apprises automatiquement à partir des données et difficilement interprétables. C'est pourquoi les approches basées sur des caractéristiques locales ne peuvent pas être implémentées en uti-

lisant des règles définies manuellement. L'apprentissage statistique propose un cadre théorique et pratique qui permet de surmonter ces difficultés. Grâce aux grandes quantités de données vidéo disponibles, les algorithmes d'apprentissage statistique supervisés peuvent automatiquement apprendre des modèles complexes d'action à partir de données d'entraînement annotées.

Les annotations — *i.e.*, étiquettes sémantiques telles que "cette vidéo contient quelqu'un en train de tomber" — peuvent être acquises de différentes manières, mais elles sont souvent obtenues manuellement. La puissance de techniques d'apprentissage statistique comme les Séparateurs à Vaste Marge (SVM) [Schölkopf and Smola 2002] permet de généraliser à n'importe quelle vidéo les connaissances acquises à partir de quelques données d'entraînement.

Des applications basées sur la reconnaissance d'actions à l'aide d'algorithmes d'apprentissage statistique ont déjà fait leur apparition — voir par exemple le développement de la vidéosurveillance automatique ou du capteur Kinect de Microsoft pour la console de jeux Xbox [Shotton et al. 2011]. Néanmois, à l'heure actuelle (2012), il n'existe toujours pas d'application grand public de la reconnaissance d'actions "in the wild" (comme sur YouTube ou avec des robots autonomes). Cela souligne le vaste travail qu'il reste à effectuer par la communauté scientifique, afin de comprendre les propriétés fondamentales des actions et comment les représenter dans des vidéos réalistes en conditions non contrôlées.

Dans ce contexte, nous pensons qu'il est nécessaire de se baser à la fois sur les modèles précurseurs structurés, et sur les représentations récentes non structurées, afin de construire de meilleurs modèles d'action. Notre objectif est de trouver le bon niveau de détails qui permettra aux modèles d'action d'être à la fois plus précis que bag-of-features et plus robustes que les représentations basées sur les silhouettes.

A.3 Contributions

Notre travail repose sur des caractéristiques locales et a pour but de structurer leur agrégation afin d'apprendre de meilleurs modèles d'action. Nous présentons à la fois des représentations séquentielles pour des actions simples, et des représentations hiérarchiques pour des activités plus complexes. Nous proposons des techniques de décomposition de la structure spatio-temporelle globale des actions par série d'images, par parties temporelles et par composantes de mouvement de niveau intermédiaire. Nous utilisons ces représentations structurées en concevant des noyaux (mesures de similarité), qui sont ensuite intégrés dans des classifieurs non linéaires.

Nous validons expérimentalement les approches que nous proposons afin de répondre aux questions suivantes :

- quelles sont les bonnes *unités* à considérer pour décomposer les actions ;
- quelle est la structure intéressante des actions ;
- comment extraire et représenter la structure de manière robuste ;
- comment apprendre et comparer les modèles structurés d'action.

Nos contributions sont décrites dans les paragraphes suivants.

- Dans le but de localiser des actions simples, comme ouvrir une porte, dans de longues vidéos non segmentées, nous introduisons une extension temporellement structurée de bag-of-features : le Actom Sequence Model (ASM). Nous décrivons une action comme une séquence d'unités atomiques d'action, appelées *actoms*, qui sont des parties temporelles sémantiquement significatives et caractéristiques de l'action. ASM représente la structure temporelle des actions comme une séquence d'histogrammes de caractéristiques visuelles ancrées sur les actoms (*cf.* Figure A.7). La phase d'apprentissage nécessite l'annotation des actoms pour les exemples d'action. A l'étape de test, les actoms sont détectés automatiquement à l'aide d'un modèle non paramétrique, qui agit comme une distribution a priori sur la structure temporelle d'une action. Nous proposons une approche de détection par image centrale glissante : toutes les N images, nous évaluons la probabilité qu'une action soit centrée à une image particulière t en marginalisant sur notre a priori temporel :

$$\mathbf{P}(\text{action at } t) = \sum_{j=1}^{s} f_{\text{ASM}}(t, \hat{\mathbf{\Delta}}_\mathbf{j})\ \mathbf{P}(\hat{\mathbf{\Delta}}_\mathbf{j}) \tag{A.1}$$

  où $\hat{\mathbf{\Delta}}_\mathbf{j}$ est la j ème séquence d'actoms candidate dans le modèle temporel que nous avons appris, $\mathbf{P}(\hat{\mathbf{\Delta}}_\mathbf{j})$ est sa probabilité a priori estimée à partir des données, et $f_{\text{ASM}}(t, \hat{\mathbf{\Delta}}_\mathbf{j})$ est la probabilité a posteriori retournée par notre classifieur ASM. Nous présentons des résultats expérimentaux obtenus sur les bases de données réalistes Coffee and Cigarettes [Laptev and Pérez 2007] et DLSBP [Duchenne et al. 2009]. Nous adaptons également notre approche au problème de classification par détection et démontrons son applicabilité sur la base de données Hollywood 2 [Marszalek et al. 2009]. Nous montrons que notre méthode ASM fait mieux que (i) l'actuel état de l'art en localisation temporelle d'action, ainsi que (ii) les méthodes classiques combinant bag-of-features avec une méthode de localisation par fenêtre glissante. Ce travail a été publié dans [Gaidon et al. 2011a].

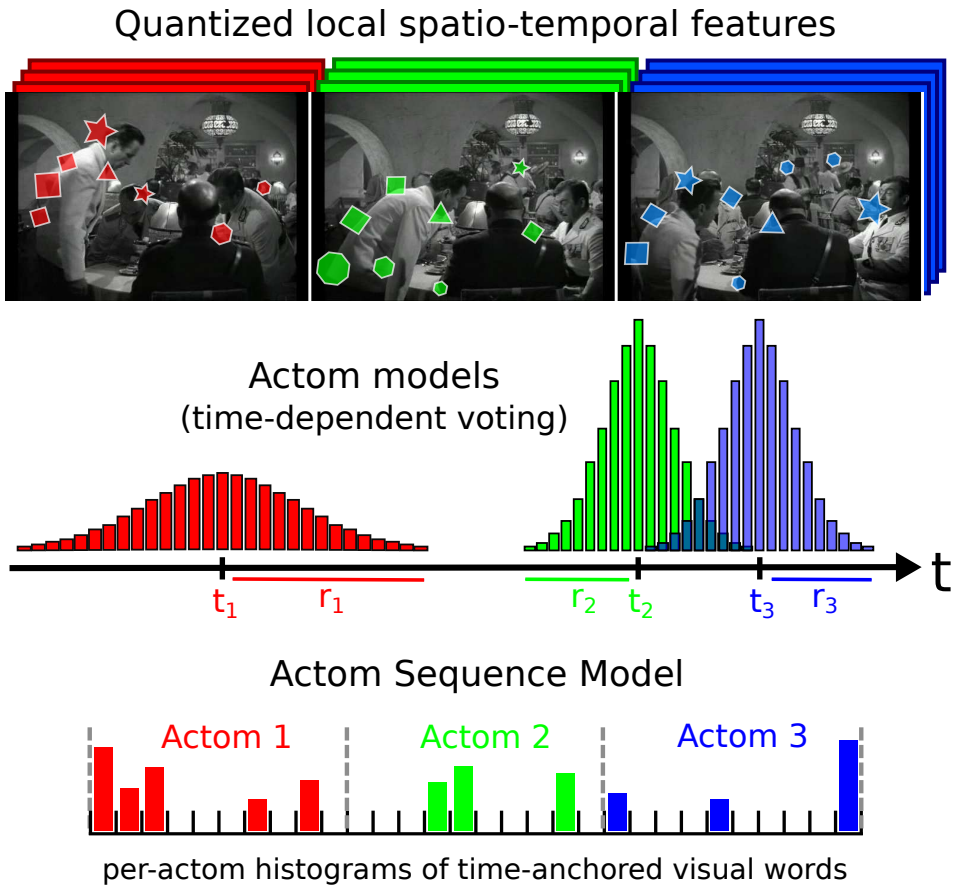


FIG. *A.7: Illustration de trois actoms pour l'action "s'asseoir" de notre Actom Sequence Model (ASM) : succession de parties temporelles modélisées par l'agrégation de caractéristiques locales ancrées dans le temps.*

– Les actions simples peuvent être représentées comme des séquences de parties temporelles de courte durée. Cependant des activités plus longues, comme le saut à la perche, sont composées d'un nombre variable de sous-événements spatio-temporellement interconnectés. Nous apprenons de manière automatique à réprésenter ces activités à l'aide de hiérarchies de mouvements. Ces structures arborescentes sont apprises directement des données, pour chaque vidéo indépendemment. Nous élaborons un algorithme de partitionnement divisif, s'appuyant sur des techniques de plongement spectral afin d'extraire efficacement la structure hiérarchique de vidéos contenant de grands nombres de trajectoires locales ("tracklets", *cf.* Figure A.8). Nous utilisons cette structure pour représenter une vidéo comme des arbres binaires non orientés. Ces arbres sont modélisés par des histogrammes emboîtés de caractéristiques locales du

mouvement. Nous proposons un noyau positif défini qui calcule efficacement la similarité structurelle et visuelle entre deux décompositions hiérarchiques. Ce noyau se base sur les relations père-fils :

$$k(\mathcal{T}_1, \mathcal{T}_2) = \frac{1}{|\mathcal{E}_1||\mathcal{E}_2|} \cdot \sum_{\substack{e_1 \in \mathcal{E}_1 \\ e_2 \in \mathcal{E}_2}} h(e_1, e_2) \tag{A.2}$$

où $\mathcal{T}_i$ est notre modèle arborescent ayant pour arrêtes $\mathcal{E}_i$ et où $h(e_1, e_2)$ dénote la similarité entre deux arrêtes modélisées par les histogrammes e_1 et e_2. Nous présentons des résultats expérimentaux sur trois bases de données récentes et difficiles : les activités complexes de Olympics Sports [Niebles et al. 2010], les intéractions entre personnes de High Five [Patron-Perez et al. 2010], et la base HMDB [Kuehne et al. 2011] contenant un plus grand nombre d'actions de structure simple. Nous montrons que des représentations hiérarchiques spécifiques à chaque vidéo donnent des informations additionnelles pour la reconnaissance d'actions. Notre approche offre des meilleures performances que les modèles d'activité non structurés, des méthodes utilisant d'autres algorithmes de décomposition du mouvement et l'état de l'art. Ce travail a été publié dans [Gaidon et al. 2012].

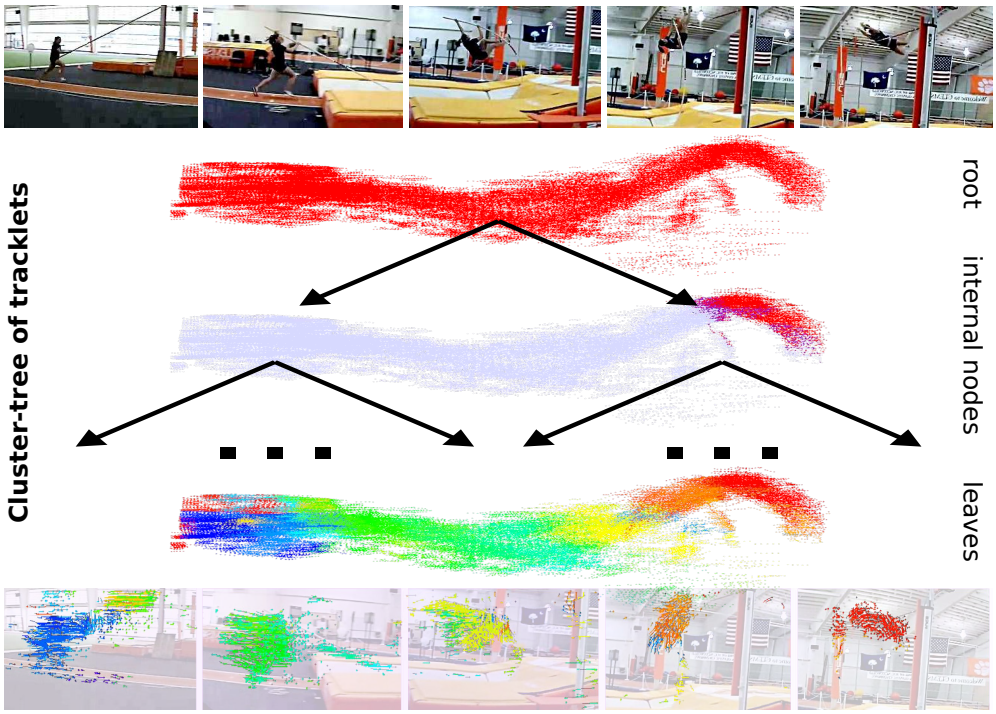


FIG. *A.8: Illustration de notre dendogramme de tracklets sur une activité de saut à la perche : décomposition hiérarchique du mouvement apprise automatiquement.*

– Pour finir, nous proposons de compléter les représentations traditionnelles basées sur la distribution moyenne des caractéristiques locales — comme BOF ou les deux approches mentionnées précédemment — avec un modèle des dépendances temporelles entre images : l'opérateur d'auto-corrélation. Nous nous attaquons au problème de reconnaissance d'actions en décrivant une vidéo comme une série temporelle de modèles par image. Nous proposons un noyau pour comparer la dynamique temporelle des actions (*cf.* Figure A.9). Les deux contributions détaillées précédemment se concentrent sur la structure globale des actions pour organiser des ensembles de caractéristiques locales en représentations structurées de haut niveau. Cette partie de notre travail explore une voie différente pour utiliser l'information de structure : au lieu de comparer *des représentations vidéo structurées*, nous comparons *des représentations de la structure* des vidéos. Nos contributions principales sont les suivantes. Tout d'abord, nous proposons un noyau théoriquement bien fondé permettant de comparer la dynamique et la structure temporelle des actions. Ce noyau s'appuie sur une distance entre opérateurs d'auto-corrélation (DACO : Difference of Auto-Correlation Operators) :

$$k_{\mathrm{DACO}}(\{\mathbf{x_t}\}_{t=1:T}\ ,\ \{\mathbf{y_{t'}}\}_{t'=1:T'})^2 = \exp\left(-\frac{1}{2\sigma^2}\left\|\hat{\rho}_\tau^{(\mathbf{y})} - \hat{\rho}_\tau^{(\mathbf{x})}\right\|_{HS}^2\right) \quad \text{(A.3)}$$

où $\hat{\rho}_\tau^{(\mathbf{x})}$ et $\hat{\rho}_\tau^{(\mathbf{y})}$ dénotent les opérateurs d'auto-corrélation des séries temporelles $\{\mathbf{x_t}\}_{t=1:T}$ et $\{\mathbf{y_{t'}}\}_{t'=1:T'}$. Ensuite, nous obtenons une formulation pratique pour calculer DACO dans n'importe quel espace de représentation dérivant d'un noyau de base entre images. Pour finir, nous rapportons des résultats expérimentaux sur des bases de données récentes de reconnaissance d'actions, montrant que DACO apporte une information complémentaire aux distributions moyennes de caractéristiques locales, comme utilisées dans les modèles de l'état de l'art basés sur bag-of-features. Ce travail a été publié dans [Gaidon et al. 2011b].

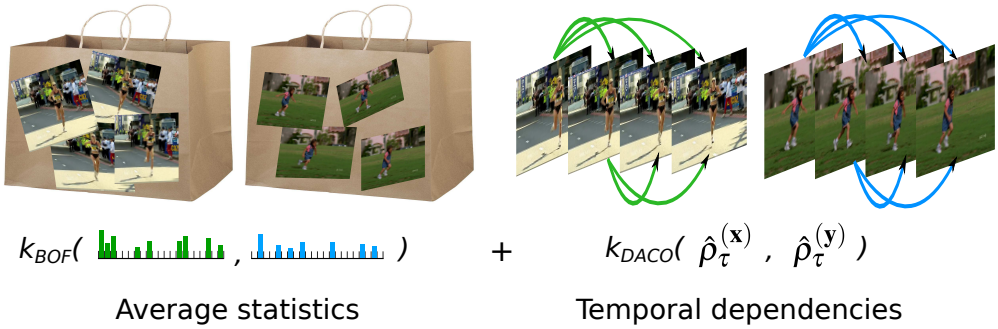


Fig. A.9: *Combinaison entre bag-of-features non structuré (à gauche) et dépendances temporelles (à droite) modélisées par des opérateurs d'auto-corrélation dans le noyau DACO.*

A.4 Perspectives

Dans cette section, nous donnons des extensions possibles à notre travail, suggérées par nos expériences et les récents progrès dans les domaines de la vision par ordinateur et de l'apprentissage statistique.

Extensions de notre approche de localisation

Une première amélioration de notre modèle ASM et de notre approche de localisation par image centrale glissante consiste à les adapter pour les tâches de **localisation spatio-temporelle**, c'est-à-dire retourner une fenêtre temporelle et une séquence de boîtes englobantes autour des acteurs réalisant l'action. Pour pouvoir localiser quand et où une action est réalisée, il est possible d'appliquer l'approche de localisation temporelle utilisée pour ASM à des trajectoires obtenues à l'aide de détecteurs d'humains et d'objets pré-entrainés [Kläser et al. 2010a,b, Prest et al. 2011]. Alternativement, au lieu de commencer par localiser indépendemment pour ensuite classifier, il est aussi possible modéliser conjointement la structure et l'apparence spatio-temporelles, avec par exemple le modèle par parties déformable de Felzenszwalb et al. [2010]. Cependant, l'une des principales difficultés est de trouver comment incorporer des aspects géométriques tout en maintenant la robustesse de nos modèles aux conditions vidéos réalistes, notamment en présence d'occlusions ou de grandes variations de points de vue. Les poselets [Bourdev and Malik 2009] constituent une approche intéressante pour gérer dans des images statiques les challenges cités précédemment. Ces détecteurs pré-entraînés de poses locales ont été appliqués avec succès à la reconnaissance d'actions dans des images statiques [Maji et al. 2011]. En se basant sur ces idées et sur notre travail sur les actoms, il est envisageable de concevoir des détecteurs de poses dynamiques locales emblématiques, qui contiendraient plus d'information que juste le mouvement, comme par exemple la nature des objets ou des parties du corps en mouvement.

Les modèles ASM reposent sur un nombre fixe d'actoms par catégorie. Cette limitation pourrait être levée afin que notre approche s'étende aux **modèles de séquence avec un nombre variable de parties**. En effet, il semble nécessaire de permettre à des exemples d'action d'une même catégorie d'avoir différents nombres d'actoms, pour pouvoir modéliser un large éventail d'actions, en raison par exemple des occlusions temporelles, de la variabilité intra-classe ou des différences entre les annotateurs sur la manière de décomposer une action. Il est possible de comparer des séquences avec différents nombres d'actoms tout en préservant l'ordre temporel en se basant sur des stratégies de programamtion dynamique, comme par exemple en

adaptant les noyaux sur chaînes de caractères au calcul d'une "distance d'édition d'actom" ou avec le Global Alignment Kernel [Cuturi et al. 2007]. Une autre piste potentielle consiste à utiliser le noyau Fisher [Jaakkola and Haussler 1999] associé à un modèle génératif séquentiel, comme un HMM, dans le but d'obtenir une représentation de longueur fixe basée sur un nombre fixe d'états latents. De plus, pour pouvoir estimer la structure temporelle de taille variable d'une action, cela nécessite une distribution de probabilité plus complexe qu'un histogramme de distances entre actoms comme utilisé dans ASM. Une extension directe de notre approche consisterait à utiliser à la place deux distributions temporelles : une pour le nombre de parties temporelles, et une autre distribution conditionnelle sur la position des actoms sachant le nombre d'actoms. Cependant, cela suppose de fortes hypothèses d'indépendance qui ne seraient pas nécessairement vérifiées en pratique. Idéalement, il faudrait estimer conjointement à la fois le contenu, le nombre et la position des actoms, au lieu de le faire indépendemment. Ceci est particulièrement important pour la localisation simultanée de plusieurs actions dans une même vidéo.

Apprentissage faiblement supervisé. Nous avons démontré que des modèles d'action peuvent être nettement améliorés en apprenant à partir d'une décomposition significative étiquetée manuellement, au lieu d'une simple fenêtre temporelle. Cette supervision a cependant un coût qui doit être minimisé dans le but de passer à l'échelle avec un grand nombre de catégories. En plus des approches d'apprentissage semi-supervisées, une direction prometteuse de recherche consiste à affiner automatiquement les annotations inexactes (également appelées supervision faible) venant de sources d'information bon marché ou facilement disponibles, comme des transcriptions de films et de séries TV [Gaidon et al. 2009, Laptev et al. 2008]. Des résultats prometteurs ont été obtenus dans le but d'affiner l'étendue temporelle en utilisant des méthodes discriminatives [Duchenne et al. 2009, Satkin and Hebert 2010]. Ces approches pourraient être généralisées afin de déduire les positions latentes des actoms depuis des annotations temporelles approximatives obtenues automatiquement. Notons que l'approche de Duchenne et al. [2009] partage plusieurs similarités avec les approches récentes de co-segmentation d'images. Par exemple, Joulin et al. [2012a] se base sur de multiples images et leurs étiquettes globales de catégories pour segmenter par un algorithme de partitionnement discriminatif. Il semblerait donc possible de pouvoir adapter ces algorithmes d'apprentissage faiblement supervisés (comme par exemple [Joulin et al. 2012b]) afin de trouver la structure spatio-temporelle latente des actions à partir d'un ensemble de vidéos faiblement annotées.

Extensions de nos modèles d'activités

Notre modèle arborescent d'activité modélise uniquement des relations d'inclusions entre parties du mouvement. Cette approche pourrait être augmentée par l'utilisation de relations supplémentaires — spatiales, temporelles et hiérarchiques — plus complexes, comme par exemple dans [Brendel and Todorovic 2011]. Néanmoins, la définition d'une structure latente trop complexe entraîne des difficultés d'apprentissage et d'inférence et, par conséquent, des performances sous-optimales. En effet, la flexibilité supplémentaire au niveau des variables latentes d'un tel modèle fait que l'espace de paramètres exploré est sous-contraint et trop vaste pour permettre une exploration efficace et éviter le sur-apprentissage. Surmonter ces difficultés — par exemple en trouvant des bonnes contraintes sur la structure à apprendre — permettrait d'améliorer les modèles d'actions, grâce à l'apprentissage discriminatif de leurs paramètres, tout en développant leur robustesse, grâce à la modélisation de l'incertitude sur leur structure.

En plus de découvrir automatiquement les différentes parties, notre méthode de décomposition pourrait être améliorée en modélisant également **l'importance latente de chaque partie** pour une catégorie. En effet, notre structure arborescente sur une action est obtenue sans utiliser l'information de son appartenance Ã une catégorie. De plus, tous les noeuds sont traités de manière équivalente dans notre noyau ATEP, alors que certains sous-arbres peuvent être sans rapport avec l'action (des mouvements à l'arrière plan par exemple). Pour l'instant, ces sous-arbres superflus sont considérés comme du bruit auquel notre noyau est robuste dû à son effet de moyenne. Notre méthode pourrait être améliorée par une approche plus explicite, en apprenant l'importance des sous-arbres, pour chaque catégorie, à partir de leur contenu, de leur structure et de leur position dans le dendogramme. Des facteurs d'importance d'un noeud peuvent en effet être directement intégrés dans notre noyau ATEP comme de simples poids multiplicatifs dans les comparaisons entre arêtes. Ces poids pourraient être appris en considérant l'importance de chaque noeud comme une variable latente et en utilisant des méchanismes d'inférence adéquats. Comme mentionné précédemment, la segmentation conjointe d'actions (d'une même catégorie ou de toutes les catégories à la fois) pourrait être utilisée pour tirer profit des étiquettes de catégorie, afin d'obtenir de meilleures décompositions d'activités. Notons qu'une telle pondération basée sur les parties pourrait également être applicable aux actoms puisque certaines parties temporelles sont plus importantes que d'autres a priori.

Enfin, une autre extension consiste à adapter notre méthode pour permettre la **localisation spatio-temporelle au sein des BOF-Trees**. En l'état actuel, notre approche est spécifiquement conçue pour des vidéos se

concentrant sur une seule activité. Lorsque la vidéo est seulement temporellement segmentée et contient plusieurs activités concurrentes, nous pouvons adapter notre méthode pour se restreindre à des sous-parties de la vidéo en mettant en correspondance des sous-arbres, au lieu de comparer la totalité de notre représentation hierarchique. En complément des algorithmes existants et efficaces d'appariement de sous-arbres, (*cf.* [Valiente 2002]), nous pourrions également adopter une approche d'appariement gloutonne "coarse-to-fine" afin d'accélérer les calculs. En s'appuyant sur la propriété d'additivité des BOF-trees, nous pouvons en effet approcher un sous-arbre par une de ses arêtes supérieures (père, fils), comme dans notre noyau ATEP. C'est pourquoi nous pourrions seulement appairer les arêtes des BOF-trees, et parcourir l'arbre de haut en bas de manière analogue à une stratégie par séparation et évaluation.

Appliquer notre méthode à des vidéos de longue durée comme celles étudiées/considérées par nos modèles ASM requiert une approche différente. En effet, la construction descendante de notre modèle arborescent suppose que toutes les tracklets soient disponibles simultanément. Par conséquent, notre algorithme de segmentation doit être précédé d'une phase de segmentation temporelle à la volée — par exemple en utilisant un détecteur de changement de plan pour des vidéos éditées telles que les films, ou plus généralement un algorithme de détection de point de changement comme [Harchaoui et al. 2009] — afin de dégager d'abord les racines des arbres à construire.

Extensions du modè par séries temporelles

Combinaison de DACO et de ASM. Nos résultats suggèrent que notre noyau DACO est particulièrement adapté aux actions courtes, et améliore la performance des noyaux par agrégation. Il semblerait donc possible d'améliorer les comparaisons entre actoms réalisées dans ASM en calculant non seulement la similarité entre le contenu des actoms, mais aussi en comparant leur dynamique avec DACO. En effet, la faible durée des actoms signifie qu'ils peuvent être considérés comme des séries temporelles quasi-stationnaires. De plus, les poids des images dérivés des modèles d'actom peuvent être directement intégrés dans DACO en les appliquant aux projections des BOFs de chaque image dans le RKHS associé au noyau entre images.

Version courte durée de DACO. Comme semblent le montrer nos résultats sur la base de données Youtube, la principale limitation de notre noyau DACO provient de son hypothèse de stationnarité, qui est souvent fausse dans le cas d'actions longues et complexes. C'est pourquoi une extension naturelle de notre approche consiste à calculer une version courte durée de DACO, par exemple en convoluant successivement les images avec une fenêtre temporelle Gaussienne de courte durée. Les séries temporelles de courte durée peuvent

en effet être consiérées comme quasi-stationnaires, puisqu'il y a tout simplement moins d'observations pour remarquer un changement significatif dans la distribution d'images. Cette approche pose cependant le problème suivant : la dynamique des séries est modélisée par une séquence d'opérateurs d'auto-corrélation locaux au lieu d'un seul global. Par conséquent, un autre noyau doit être utilisé afin de combiner les comparaisons locales de fenêtres obtenues avec DACO. Notons que l'application de cette idée au calcul d'une version fenêtrée du noyau d'auto-régression donne une représentation intéressante du signal vidéo, qui partage des similarités avec les codes linéaires prédictifs, très utilisés en traitement de signaux sonores et de la parole en particulier.

Analyse de séries temporelles d'actions. La plupart des outils construits pour analyser des séries temporelles sont conçus pour des données univariées, comme des signaux sonores ou l'évolution de prix sur un marché financier. Cependant, les vidéos reposent sur des représentations en haute dimension. C'est pourquoi l'analyse de leurs propriétés temporelles — comme la périodicité — nécessite des techniques différentes, adaptées à ces signaux complexes et multivariés. Par exemple, Cutler and Davis [2000] — plus d'une décennie avant nous — utilisa des techniques d'analyse temps-fréquence sur des auto-similarités temporelles afin de détecter des mouvements périodiques. Nous avons montré que l'opérateur d'auto-corrélation dans un RKHS d'image est un objet mathématique qui peut être utilisé pour étudier les propriétés temporelles des actions. Nous pensons que ce cadre formel, bien qu'embryonnaire, peut ouvrir la voie à une analyse plus sophistiquée des actions, en généralisant le grand nombre d'outils qui existent déjà pour des séries univariées, comme la finance ou le traitement de la parole et du son. Nous pourrions par exemple étendre la notion de corrélogramme et construire un "corrélogramme d'action" en utilisant la norme de Hilbert-Schmidt de l'opérateur d'auto-corrélation, afin de mesurer le caractère aléatoire d'une action, pour la détection de comportements anormaux notamment.

Bibliography

S. Ali, A. Basharat, and M. Shah. Chaotic invariants for human action recognition. In *ICCV*, 2007. (page 20.)

F.R. Bach, G.R.G. Lanckriet, and M.I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*, 2004. (page 24.)

A. Basharat and M. Shah. Time series prediction by chaotic modeling of nonlinear dynamical systems. In *CVPR*, 2009. (page 92.)

P.R. Beaudet. Rotationally invariant image operators. In *ICPR*, 1978. (page 19.)

M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as Space-Time Shapes. In *CVPR*, 2005. (pages 16 and 30.)

A. Bobick and J. Davis. The recognition of human movement using temporal templates. *PAMI*, 23(3):257–267, 2001. (pages 6, 7, 16, 126, and 127.)

L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009. (pages 113 and 134.)

G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008. (page 60.)

M. Brand and V. Kettnaker. Discovery and segmentation of activities in video. *PAMI*, 2000. (page 15.)

M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *CVPR*, 1997. (pages 14 and 92.)

W. Brendel and S. Todorovic. Activities as time series of human postures. In *ECCV*, 2010. (pages xiii, 15, 92, and 103.)

W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011. (pages xiii, 26, 27, 57, 80, 114, and 136.)

T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. (pages 20, 28, and 57.)

L. W. Campbell and A. F. Bobick. Recognition of human body motion using phase space constraints. In *ICCV*, 1995. (page 20.)

C. C. Chen and J. K. Agarwal. Modeling Human Activities as Speech. In *CVPR*, 2011. (page 14.)

O. Chomat and J. L. Crowley. Probabilistic recognition of activity using local appearance. In *CVPR*, 1999. (pages 7, 18, and 127.)

J. Cohn and T. Kanade. Use of automated facial image analysis for measurement of emotion expression. In *Handbook of emotion elicitation and assessment*. Oxford UP Series in Affective Science, 2006. (page 31.)

G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, 2004. (page 22.)

O.G. Cula and K.J. Dana. Compact representation of bidirectional texture functions. In *CVPR*, 2001. (page 22.)

R. Cutler and L.S. Davis. Robust real-time periodic motion detection, analysis, and applications. *PAMI*, 2000. (pages 116 and 138.)

M. Cuturi. Fast Global Alignment Kernels. ICML, 2011. (page 108.)

M. Cuturi, J. P. Vert, O. Birkenes, and T. Matsui. A Kernel for Time Series Based on Global Alignments. In *IEEE ICASSP*, 2007. (pages 92, 113, and 135.)

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. (pages 7, 19, 40, 78, and 127.)

N. Dalal, B. Triggs, and C. Schmid. Human Detection Using Oriented Histograms of Flow and Appearance. In *ECCV*, 2006. (pages 70 and 101.)

T. Darrell and A. Pentland. Space-time gestures. In *CVPR*, 1993. (pages 15 and 18.)

E. De Castro and C. Morandi. Registration of translated and rotated images using finite Fourier transforms. *PAMI*, 1987. (page 77.)

P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, pages 65–72, 2005. (pages 7, 18, 19, 22, 23, 30, and 127.)

O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009. (pages 10, 22, 30, 31, 32, 33, 36, 37, 43, 44, 45, 48, 49, 114, 130, and 135.)

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification and Scene Analysis*. 1995. (page 56.)

A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV*, 2003. (pages 16 and 17.)

P. Ekman and W. V. Friesen. *Facial Action Coding System*. Consulting Psychologists Press, 1978. (page 31.)

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010. (pages 7, 46, and 127.)

G. Farnebäck. Two-frame motion estimation based on polynomial expansion. *Image Analysis*, pages 363–370, 2003. (pages 20 and 60.)

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2009. (page 31.)

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2010. (pages 27, 58, 113, and 134.)

R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *CVPR*, 2005. (page 23.)

M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. (page 77.)

L. Foster, A. Waagen, N. Aijaz, M. Hurley, A. Luis, J. Rinsky, C. Satyavolu, M. J. Way, P. Gazis, and A. Srivastava. Stable and efficient gaussian process calculations. *JMLR*, 10:857–882, 2009. (page 66.)

C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nystrom method. *PAMI*, 26(2):214–225, 2004. (pages 57, 64, 65, and 78.)

M. Fradet, P. Robert, and P. Pérez. Clustering point trajectories with various life-spans. In *CVMP*, 2009. (page 57.)

Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995. (page 21.)

A. Gaidon, M. Marszałek, and C. Schmid. Mining visual actions from movies. In *BMVC*, 2009. (pages 18, 36, 40, 114, and 135.)

A. Gaidon, Z. Harchaoui, and C. Schmid. Actom Sequence Models for Efficient Action Detection. In *CVPR*, 2011a. (pages 10, 71, 91, and 130.)

A. Gaidon, Z. Harchaoui, and C. Schmid. A time series kernel for action recognition. In *BMVC*, 2011b. (pages 12 and 133.)

A. Gaidon, Z. Harchaoui, and C. Schmid. Recognizing activities with cluster-trees of tracklets. In *BMVC*, 2012. (pages 11 and 132.)

D. M. Gavrila and L. S. Davis. Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. In *International Workshop on Automatic Face and Gesture recognition*, 1995. (page 15.)

A. Gilbert, J. Illingworth, and R. Bowden. Action recognition using mined hierarchical compound features. *PAMI*, pages 883–897, 2010. (pages 21, 24, and 58.)

L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *PAMI*, pages 2247–2253, 2007. (pages 6, 16, and 126.)

A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *Algorithmic Learning Theory*, pages 63–77, 2005. (page 94.)

M. Grundmann, F. Meier, and I. Essa. 3D shape context and distance transform for action recognition. In *ICPR*, 2008. (page 57.)

Z. Harchaoui, F. Bach, and E. Moulines. Testing for homogeneity with kernel Fisher discriminant analysis. In *Adv. NIPS*, 2008. (page 97.)

Z. Harchaoui, F. Bach, and E. Moulines. Kernel change-point analysis. In *NIPS*, 2009. (pages 116 and 137.)

C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988. (page 19.)

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning (2nd edition)*. Springer, 2008. (page 57.)

M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *AISTATS*, 2005. (page 38.)

M. Hoai, Z. Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011. (page 14.)

T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196, 2001. (page 23.)

S. Hongeng and R. Nevatia. Large-scale event detection using semi-hidden markov models. In *ICCV*, 2003. (page 28.)

W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *Transactions on Systems, Man, and Cybernetics*, 34(3):334–352, 2004. (pages 6 and 126.)

N. Ikizler-Cinbis and S. Sclaroff. Object, scene and actions: Combining multiple features for human action recognition. In *ECCV*, 2010. (pages 77 and 103.)

N. Ikizler-Cinbis, R. G. Cinbis, and S. Sclaroff. Learning actions from the web. In *ICCV*, 2009. (page 18.)

T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. 1999. (pages 113 and 135.)

Z. Jiang, Z. Lin, and L. Davis. Recognizing Human Actions by Learning and Matching Shape-Motion Prototype Trees. *PAMI*, 2012. (page 58.)

A. Joulin, F. Bach, and J. Ponce. Multi-class cosegmentation. In *CVPR*, 2012a. (pages 114 and 135.)

A. Joulin, F. Bach, and J. Ponce. A convex relaxation for weakly supervised classifiers. In *ICML*, 2012b. (pages 114 and 135.)

I. N. Junejo, E. Dexter, I. Laptev, and P. Pérez. View-Independent Action Recognition from Temporal Self-Similarities. *PAMI*, 2010. (pages 20 and 98.)

S. H. Jung, Y. Guo, H. Sawhney, and R. Kumar. Action video retrieval based on atomic action vocabulary. In *MIR*, 2008. (page 92.)

Y. Ke, R. Sukthankar, and M. Hebert. Volumetric features for video event detection. *IJCV*, 2010. (pages xiii, 16, 17, and 18.)

T. K. Kim and R. Cipolla. Canonical correlation analysis of video volume tensors for action categorization and detection. *PAMI*, 2009. (pages xiii and 16.)

A. Klaser, M. Marszalek, and C. Schmid. A Spatio-Temporal Descriptor Based on 3D-Gradients. In *BMVC*, 2008. (pages 18 and 19.)

A. Kläser, M. Marszalek, I. Laptev, and C. Schmid. Will person detection help bag-of-features action recognition? Research report, INRIA, 2010a. (pages 103, 113, and 134.)

A. Kläser, M. Marszałek, C. Schmid, and A. Zisserman. Human Focused Action Localization in Video. In *SGA*, 2010b. (pages 17, 18, 43, 45, 46, 48, 49, 113, and 134.)

A. Kovashka and K. Grauman. Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In *CVPR*, 2010. (pages 24, 25, 58, and 103.)

H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. (pages xiv, 11, 59, 77, 83, 85, and 132.)

K. Kulkarni, E. Boyer, R. Horaud, and A. Kale. An unsupervised framework for action recognition using actemes. In *ACCV*, 2010. (pages 14 and 92.)

I. Laptev. On space-time interest points. *IJCV*, 64(2–3):107–123, 2005. (pages xiii, 7, 19, 32, 78, and 127.)

I. Laptev. STIP: Spatio-Temporal Interest Point library, 2011. URL `www.di.ens.fr/~laptev/interestpoints.html`. (page 33.)

I. Laptev and T. Lindeberg. Space-time interest points. 2003. (page 18.)

I. Laptev and P. Pérez. Retrieving actions in movies. In *ICCV*, 2007. (pages xiii, 9, 17, 32, 42, 43, 45, 48, 49, and 130.)

I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. (pages 7, 18, 19, 22, 23, 24, 30, 31, 32, 38, 40, 46, 50, 78, 79, 80, 102, 103, 114, 127, 128, and 135.)

B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *CVPR*, 2007. (pages xiii, 14, 15, and 92.)

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. (page 23.)

Q.V. Le, W.Y. Zou, S.Y. Yeung, and A.Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011. (page 103.)

B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *CVPR*, 2005. (pages 23 and 24.)

J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*, 2011. (pages 20 and 57.)

H. T. Lin, C. J. Lin, and C. Weng. A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, 38:267–276, 2007. (page 38.)

Y. Lin, G. Wahba, H. Zhang, , and Y. Lee. Statistical Properties and Adaptive Tuning of Support Vector Machines. *Machine Learning*, 4:115–136, 2002. (page 38.)

J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild". In *CVPR*, 2009. (pages 18, 21, 100, and 103.)

J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *CVPR*, 2011. (page 58.)

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. (page 77.)

Z. Lu, T. K. Leen, Y. Huang, and D. Erdogmus. A reproducing kernel Hilbert space framework for pairwise time series distances. In *ICML*, 2008. (page 92.)

B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI*, pages 674–679, 1981. (page 20.)

F. Lv and R. Nevatia. Single view human action recognition using key pose matching and Viterbi path searching. In *CVPR*, 2007. (pages 14 and 15.)

S. Maji, A. Berg, and J. Malik. Classification Using Intersection Kernel Support Vector Machines is efficient. In *CVPR*, 2008. (pages 38, 73, and 101.)

S. Maji, L. Bourdev, and J. Malik. Action recognition from a distributed representation of pose and appearance. In *CVPR*, 2011. (pages 113 and 134.)

M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, 2009. (pages 4, 10, 18, 32, 43, 44, 50, 124, and 130.)

P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 514–521. IEEE, 2009. (pages xiii, 20, and 21.)

P. Matikainen, M. Hebert, and R. Sukthankar. Representing pairwise spatial and temporal relations for action recognition. *ECCV*, 2010. (pages 24, 25, and 58.)

R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *ICCV*, 2009. (page 20.)

K. Mikolajczyk and H. Uemura. Action recognition with motion-appearance vocabulary forest. In *CVPR*, 2008. (pages 21 and 58.)

T. B. Moeslund, A. Hilton, and V. Kruger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126, 2006. (page 14.)

F. Mordelet and J.P. Vert. A bagging SVM to learn from positive and unlabeled examples. Technical report, INSERM, 2010. URL `http://hal.archives-ouvertes.fr/hal-00523336`. (page 53.)

J. C. Niebles and L. Fei-Fei. Hierarchical model of shape and appearance for human action classification. In *CVPR*, 2007. (pages xiii, 26, and 57.)

J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *IJCV*, 79(3):299–318, 2008. (pages 22 and 30.)

J. C. Niebles, Cw Chen, , and L. Fei-Fei. Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification. In *ECCV*, 2010. (pages xiv, 11, 18, 27, 31, 59, 74, 75, 76, 80, 91, 103, and 132.)

S. A. Niyogi and E. H. Adelson. Analyzing and recognizing walking figures in XYT. In *CVPR*, 1994. (page 15.)

S. Nowozin, G. Bakir, and K. Tsuda. Discriminative subsequence mining for action classification. In *ICCV*, 2007. (pages 21 and 92.)

E. J. Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54(1):185–204, 1930. (page 64.)

A. Oikonomopoulos, I. Patras, and M. Pantic. An implicit spatiotemporal shape model for human activity localization and recognition. In *CVPR Workshops*, 2009. (pages 21 and 24.)

N. M. Oliver, B. Rosario, and A. P. Pentland. A Bayesian computer vision system for modeling human interactions. *PAMI*, 2000. (pages 14 and 15.)

P. Over, G. Awad, M. Michel, J. Fiscus, W. Kraaij, A.F. Smeaton, and G. Quénot. Trecvid 2011 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID 2011*. NIST, USA, 2011. (pages 3 and 123.)

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web., 1999. (page 21.)

V. Parameswaran and R. Chellappa. View invariance for human action recognition. *International Journal of Computer Vision*, 66(1):83–101, 2006. (page 20.)

A. Patron-Perez, M. Marszałek, A. Zisserman, and I. D. Reid. High five: Recognising human interactions in TV shows. In *British Machine Vision Conference*, 2010. (pages xiv, 11, 18, 59, 74, 80, 83, and 132.)

J. Platt. Probabilistic outputs for support vector machines. *Bartlett P. Schoelkopf B. Schurmans D. Smola, AJ, editor, Advances in Large Margin Classifiers*, pages 61–74, 2000. (page 38.)

R. Polana and R. Nelson. Low level recognition of human motion. In *Proc. IEEE Workshop on Nonrigid and Articulate Motion*, pages 77–82, 1994. (page 16.)

A. Prest, V. Ferrari, and C. Schmid. Explicit modeling of human-object interactions in realistic videos. Research report, INRIA, 2011. URL `http://hal.inria.fr/inria-00626929`. (pages 113 and 134.)

A. Prest, V. Ferrari, and C. Schmid. Explicit modeling of human-object interactions in realistic videos. Pami, 2012. (page 57.)

A. Quattoni, S. Wang, L.P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *PAMI*, 29(10):1848–1852, 2007. (page 26.)

L. R. Rabiner and R. W. Schafer. Introduction to digital speech processing. *Foundations and trends in signal processing*, 2007. (pages 14 and 31.)

J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer, 2005. (page 92.)

C. Rao, A. Yilmaz, and M. Shah. View-invariant representation and recognition of actions. *International Journal of Computer Vision*, 50(2):203–226, 2002. (page 20.)

M. Raptis, I. Kokkinos, and S. Soatto. Discovering Discriminative Action Parts from Mid-Level Video Representations. In *CVPR*, 2012. (pages 28 and 58.)

K. K. Reddy, J. Liu, and M. Shah. Incremental action recognition using feature-tree. In *CVPR*, 2009. (page 58.)

M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008. (pages 16, 18, 100, and 103.)

M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 1956. (page 39.)

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. (page 84.)

S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. (page 84.)

H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Transactions on Acoustics, Speech and Signal Processing*, 1978. (page 15.)

P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *IJCV*, 80(1):72–91, 2008. (page 20.)

Scott Satkin and Martial Hebert. Modeling the Temporal Extent of Actions. In *ECCV*, 2010. (pages 30, 31, 37, 42, 46, 114, and 135.)

S. Savarese, A. DelPozo, J. C. Niebles, and L. Fei-Fei. Spatial-Temporal correlatons for unsupervised action classification. In *Motion and video Computing, 2008. WMVC 2008. IEEE Workshop on*, pages 1–8. IEEE, 2008. (page 24.)

K. Schindler and L. Van Gool. Action snippets: How many frames does human action recognition require. In *CVPR*, 2008. (pages 16, 17, and 92.)

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002. (pages 8, 22, 38, 57, 74, 90, and 129.)

B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998. (page 64.)

C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: a local SVM approach. In *ICPR*, 2004. (pages 7, 16, 18, 22, 30, 99, 103, and 127.)

D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*. Wiley, 1992. (page 39.)

D. Sculley. Web-scale k-means clustering. In *WWW*, 2010. (pages 72 and 78.)

J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge Univ Pr, 2004. (pages 73, 96, 97, and 107.)

E. Shechtman and M. Irani. Space-time behavior based correlation. In *CVPR*, 2005. (page 16.)

Y. Sheikh, M. Sheikh, and M. Shah. Exploring the space of a human action. In *ICCV*, 2005. (page 20.)

J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *ICCV*, pages 1154–1160. IEEE, 1998. (page 64.)

J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8): 888–905, 2000. (pages 63, 67, and 70.)

J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600. IEEE, 1994. (page 60.)

Q. Shi, L. Cheng, L. Wang, and A. Smola. Human action segmentation and recognition using discriminative semi-Markov models. *IJCV*, 2011. (page 14.)

J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011. (pages 6, 8, 126, and 129.)

Tomas Simon, Minh Nguyen, Fernando De la Torre, and Jeff Cohn. Action unit detection with segment-based SVM. In *CVPR*, 2010. (page 31.)

J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. (page 22.)

A.F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*. ACM Press, 2006. (pages 3 and 123.)

T. Starner and A. Pentland. Real-time American Sign Language recognition from video using Hidden Markov Models. In *International Symposium on Computer Vision*, 1995. (page 14.)

C. Sun, I. Junejo, and H. Foroosh. Action recognition using rank-1 approximation of joint self-similarity volume. In *ICCV*, 2011. (pages 102 and 103.)

K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. 2012. (page 28.)

G. Valiente. *Algorithms on Trees and Graphs*. Springer-Verlag, 2002. (pages 115 and 137.)

A. Veeraraghavan, R. Chellappa, and A. K. Roy-Chowdhury. The function space of an activity. In *CVPR*, 2006. (page 15.)

H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009. (pages 18, 33, 91, and 103.)

H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin. Action Recognition by Dense Trajectories. In *CVPR*, 2011. (pages xiii, 20, 21, 22, 30, 56, 60, 71, 78, 79, 80, 101, 102, and 103.)

Y. Wang and G. Mori. Hidden part models for human action recognition: Probabilistic vs. max-margin. *PAMI*, 2011. (pages 26 and 58.)

L. Wasserman. *All of statistics: a concise course in statistical inference*. Springer Verlag, 2004. (page 39.)

D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *CVIU*, 104(2):249–257, 2006. (pages 6 and 126.)

D. Weinland, M. Özuysal, and P. Fua. Making action recognition robust to occlusions and viewpoint changes. *ECCV*, 2010. (page 103.)

G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, 2008. (page 19.)

G. Willems, J. H. Becker, T. Tuytelaars, and L. Van Gool. Exemplar-based action recognition in video. In *BMVC*, 2009. (pages xiii and 21.)

C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *NIPS*, 2001. (page 64.)

A. D. Wilson and A. F. Bobick. Learning visual behavior for gesture analysis. In *International Symposium on Computer Vision*, 1995. (page 14.)

A. D. Wilson and A. F. Bobick. Parametric Hidden Markov Models for gesture recognition. *PAMI*, 1999. (page 18.)

S.F. Wong, T.K. Kim, and R. Cipolla. Learning motion categories using both semantic and structural information. In *CVPR*, 2007. (pages 23 and 24.)

J. Yamato, J. Ohaya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *CVPR*, 1992. (pages 14 and 92.)

A. Yao, J. Gall, and L. Van Gool. A hough transform-based voting framework for action recognition. In *CVPR*, 2010. (page 21.)

J. Yuan, Z. Liu, and Y. Wu. Discriminative video pattern search for efficient action detection. *PAMI*, 2011. (page 21.)

L. Zelnik-Manor and M. Irani. Event-based analysis of video. In *CVPR*, 2001. (pages 7, 18, and 127.)

Z. Zeng and Q. Ji. Knowledge based activity recognition with dynamic bayesian network. In *ECCV*, 2010. (pages 14 and 92.)