

## 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

**Nicolas CHESNEAU**

Thèse dirigée par **Cordelia SCHMID**  
et coencadrée par **KartEEK ALAHARI** et **Grégory ROGEZ**

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

## Learning to Recognize Actions with Weak Supervision

Reconnaissance d'actions de manière faiblement supervisée

Thèse soutenue publiquement le **23 février 2018**,  
devant le jury composé de :

**Dr. François Brémont**

Inria Sophia Antipolis, Examineur

**Pr. Christian Wolf**

INSA Lyon, Lyon, France, Rapporteur

**Pr. Jordi González**

CVC, Barcelone, Espagne, Rapporteur

**Dr. Cordelia Schmid**

Inria Grenoble, Montbonnot, France, Directeur de thèse

**Dr. KartEEK Alahari**

Inria Grenoble, Montbonnot, France, Co-Encadrant de thèse

**Dr. Grégory Rogez**

Inria Grenoble, Montbonnot, France, Co-Encadrant de thèse







## Abstract

With the rapid growth of digital video content, automatic video understanding has become an increasingly important task. Video understanding spans many possible applications such as web-video content analysis, autonomous vehicles, human-machine interfaces (e.g., Kinect or video surveillance). This thesis makes contributions addressing two major tasks in video understanding: webly-supervised action detection and human action localization.

Webly-supervised action recognition aims to learn actions from video content on the internet, with no additional supervision. We propose a novel approach in this context, which leverages the synergy between visual data (video) and the associated textual metadata, to learn event classifiers with no manual annotations. Specifically, we first download a video dataset making use of queries constructed automatically from textual description of events, prune irrelevant videos (i.e., videos where the event of interest does not occur) with text and video data, and then learn the corresponding event-specific classifiers. We show the importance of both the main steps of our method, i.e., query generation and data pruning, with quantitative results. We evaluate this approach in the challenging setting where no manually annotated training set is available, i.e., EK0 in the TrecVid challenge, and show state-of-the-art results on MED 2011 and 2013 datasets.

In the second part of the thesis, we focus on human action localization, which involves recognizing actions that can occur in a video, such as “drinking” or “phoning”, as well as their spatial and temporal extent. We propose a new person-centric framework for action localization that tracks people in videos and extracts full-body human tubes, i.e., spatio-temporal regions localizing actions, even in the case of occlusions or truncations. The motivation is two-fold. First, it allows us to handle occlusions and camera viewpoint changes when localizing people, as it infers full-body localization. Second, it provides a better reference grid for extracting action information than standard human tubes, i.e., tubes which frame visible parts only. This is achieved by training a novel human part detector that scores visible parts while regressing full-body bounding boxes, even when they lie outside the frame. The core of our method is a convolutional neural network which learns part proposals specific to certain body parts. These are then combined to detect people robustly in each frame. Our tracking algorithm connects the image detections temporally to extract full-body human tubes. We evaluate our new tube extraction method on a recent challenging dataset, DALY, showing state-of-the-art results.

**Keywords:** action recognition, action localization, weakly supervision, text and visual representation, convolutional neural network, video understanding

## Résumé

L'accroissement rapide des données numériques vidéographiques fait de la compréhension automatique des vidéos un enjeu de plus en plus important. Comprendre de manière automatique une vidéo recouvre de nombreuses applications, parmi lesquelles l'analyse du contenu vidéo sur le web, les véhicules autonomes, les interfaces homme-machine. Cette thèse présente des contributions dans deux problèmes majeurs pour la compréhension automatique des vidéos : la détection d'actions supervisée par des données web, et la localisation d'actions humaines.

La détection d'actions supervisées par des données web a pour objectif d'apprendre à reconnaître des actions dans des contenus vidéos sur Internet, sans aucune autre supervision. Nous proposons une approche originale dans ce contexte, qui s'appuie sur la synergie entre les données visuelles (les vidéos) et leur description textuelle associée, et ce dans le but d'apprendre des classifieurs pour les événements sans aucune supervision. Plus précisément, nous téléchargeons dans un premier temps une base de données vidéos à partir de requêtes construites automatiquement en s'appuyant sur la description textuelle des événements, puis nous enlevons les vidéos téléchargées pour un événement, et dans laquelle celui-ci n'apparaît pas. Enfin, un classifieur est appris pour chaque événement. Nous montrons l'importance des deux étapes principales, c'est-à-dire la créations des requêtes et l'étape de suppression des vidéos, par des résultats quantitatifs. Notre approche est évaluée dans des conditions difficiles, où aucune annotation manuelle n'est disponible, dénotées EK0 dans les challenges TrecVid. Nous obtenons l'état de l'art sur les bases de données MED 2011 et 2013.

Dans la seconde partie de notre thèse, nous nous concentrons sur la localisation des actions humaines, ce qui implique de reconnaître à la fois les actions se déroulant dans la vidéo, comme par exemple "boire" ou "téléphoner", et leur étendues spatio-temporelles. Nous proposons une nouvelle méthode centrée sur la personne, traquant celle-ci dans les vidéos pour en extraire des tubes encadrant le corps entier, même en cas d'occultations ou dissimulations partielles. Deux raisons motivent notre approche. La première est qu'elle permet de gérer les occultations et les changements de points de vue de la caméra durant l'étape de localisation des personnes, car celle-ci estime la position du corps entier à chaque frame. La seconde est que notre approche fournit une meilleure grille de référence que les tubes humains standards (c'est-à-dire les tubes qui n'encadrent que les parties visibles) pour extraire de l'information sur l'action. Le coeur de notre méthode est un réseau de neurones convolutionnel qui apprend à générer des propositions de parties du corps humain. Notre algorithme de tracking connecte

les détections temporellement pour extraire des tubes encadrant le corps entier. Nous évaluons notre nouvelle méthode d'extraction de tubes sur une base de données difficile, DALY, et atteignons l'état de l'art.

**Mots-clefs :** reconnaissance d'actions, localisation d'actions, supervision par des données web, représentations textuelle et visuelle, réseau de neurones convolutionnel, compréhension vidéo.

## Acknowledgements

First of all, I would like to thank my advisors, Doctor Cordelia Schmid, Doctor Karteek Alahari, and Doctor Grégory Rogez for their invaluable guidance. Cordelia's experience, vision, and scientific intuition have been extremely precious during all these years. Furthermore, Cordelia has always been very supportive and patient. Karteek's openness, his ability to stand back and give a different perspective, have made me pushed past my boundaries. Grégory's openness, contagious enthusiasm, scientific intuition and rigor, have helped me a lot to formulate, propose novel ideas, and allowed me to technically and scientifically progress throughout all projects. I am also grateful to all the colleagues I met over the past few years. They are too many to be exhaustively cited here. Many thanks go to my jury members – Doctor François Brémond, Professor Christian Wolf, Professor Jordi González – for agreeing to evaluate my work.

My special thanks go to Mattis Paulin, Philippe Weinzaepfel, Xavier Martin, Jakob Verbeek, Julien Mairal, Ghislain Durif, Jerome Revaud, Dan Oneata, Shreyas Saxena, Vicky Kalogeiton, Danila Potapov, Hongzhou Lin, Thomas Lucas and to my office mates Piotr Koniusz, Anoop Cherian, Daan Wynen, Vladyslav Sydorov, Dexiong Chen for the valuable and daily discussions we had and their support during these years. I would also like to thank Nathalie Gillot who helped me in all administrative tasks. I finally cannot express how grateful I am to my family and friends for their tireless and unconditional support.



# Contents

<b>Contents</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Goals . . . . .	2
1.2 Context . . . . .	4
1.3 Contributions . . . . .	9
<b>I LEARNING ACTIONS FROM WEB DATA</b>	<b>13</b>
<b>2 Related Work</b>	<b>14</b>
2.1 Text classification . . . . .	14
2.2 Action recognition . . . . .	19
2.3 Zero-shot and webly-supervised learning . . . . .	23
<b>3 Learning from Web Videos for Action Classification</b>	<b>26</b>
3.1 Introduction . . . . .	26
3.2 An overview of text classification . . . . .	29
3.3 Learning from Web Videos . . . . .	38
3.4 Experiments . . . . .	45
3.5 Summary . . . . .	57
<b>II ACTION LOCALIZATION IN REAL-WORLD VIDEOS : A PERSON-CENTRIC APPROACH</b>	<b>59</b>
<b>4 Related work</b>	<b>60</b>
4.1 Object detection . . . . .	60
4.2 Amodal completion . . . . .	62
4.3 Action localization . . . . .	64
4.4 Metric . . . . .	69

<b>5</b>	<b>Detecting parts for action localization</b>	<b>71</b>
5.1	Introduction . . . . .	72
5.2	Method: From Parts to Tubes . . . . .	73
5.3	Experiments . . . . .	80
5.4	Summary . . . . .	95
<b>6</b>	<b>Conclusion</b>	<b>97</b>
6.1	Summary of contributions . . . . .	97
6.2	Perspectives for future research . . . . .	99
<b>A</b>	<b>Text classification on Reuters-21758</b>	<b>104</b>
A.1	Per-class results for pre-processing steps and features . . . . .	104
A.2	RBF kernel results on Reuters-21578 . . . . .	107
A.3	Variation of the number Gaussians and PCA dimensions for Word2Vec features . . . . .	109
	<b>Bibliography</b>	<b>110</b>



# Chapter 1

## Introduction

### Contents

---

1.1	Goals . . . . .	2
1.2	Context . . . . .	4
1.2.1	Action classification . . . . .	4
1.2.2	Zero-shot and webly-supervised techniques . . . . .	5
1.2.3	Spatio-temporal action localization . . . . .	8
1.3	Contributions . . . . .	9

---

Human action recognition and localization in videos is one of the most active fields in computer vision. Automatic video understanding offers a broad range of potential applications ranging from surveillance to auto-annotation of movies, TV footage, sport-video analysis, robotics and autonomous vehicles, social network content analysis (e.g., YouTube, Facebook, Dailymotion, Hangout, Vine), human-machine interfaces (e.g., Kinect). Many of these applications require analyzing human actions. For instance, recognizing, localizing, and predicting actions is a key component for video-surveillance. The Kinect system recognizes human poses and motion. Autonomous vehicles need to detect pedestrians and predict their future movement.

Most of these applications were made possible by the development of video content sharing. Watching a video on a mobile phone or a computer is nowadays a common activity. YouTube was the most visited website in 2016, ahead of Google and Facebook.<sup>1</sup> For the latter, video content has become the most popular and shared media, and is the first factor of growth. The amount of video data to be treated is thus considerably increasing. 300 hours of video are uploaded to YouTube ev-

---

1. [https://en.Wikipedia.org/wiki/List\\_of\\_most\\_popular\\_websites](https://en.Wikipedia.org/wiki/List_of_most_popular_websites)

ery minute, and 3.25 billion hours of video are watched each month. Consequently, designing automatic tools to analyze and understand this content, in particular recognizing and localizing human actions in these videos, has become a critical issue.

Besides the need for automatic algorithms, this rapid increase makes web data the largest source of information. Automatically leveraging it for a better video understanding is an open and interesting problem. Furthermore, web videos are often human centered, making them good candidates for a training dataset. [Laptev \[2013\]](#) shows that about 35% of screen pixels in movies, TV programs, and YouTube videos belong to people. Our work focuses on human actions, in particular recognizing and localizing them in real-world videos, e.g., YouTube videos.

## 1.1 Goals

Action classification aims to assign an action label to a video. Besides the labeling, action localization aims to localize both spatially and temporally the action in the video. This thesis addresses two important problems in this context. The first task consists of recognizing human actions using an action classifier trained with no human supervision. The second task involves localizing human actions, i.e., estimating when and where the action occurs in video frames.

**Human action recognition in real-world videos.** We aim to recognize actions in real-world videos. This problem faces multiple challenges such as designing a representation that is both robust to intra-class variability, and sufficiently discriminative in order to avoid inter-class confusion.

Intra-class variability points out that actions from the same event may significantly differ in terms of appearance and motion. For instance, for the *changing vehicle tire* event, a vehicle can be a bike or a car, as defined in the TrecVid11 challenge [[Over et al., 2010](#)]. Processes and tools involved in both sub-events, i.e., *changing a bike tire* and *changing a car tire*, are different and must be taken into account in an effective action classification system. [Figure 1.1](#) shows different videos of the event. We observe that visual content and semantic environment of a *changing vehicle tire* video differ greatly from one another.

Inter-class confusion points out that two actions may be very similar in terms of visual contents and human motions. For example, [Figure 1.1](#) shows videos from two TrecVid actions: *changing vehicle tire* and *getting the vehicle unstuck*. The two actions can share some objects (e.g., a car), a



Figure 1.1 – An example of inter-class confusion and intra-class variation. Sample frames from two TrecVid11 events: *changing vehicle tire* (top and bottom left) and *unstuck vehicle* (bottom right).

place (e.g., a road), and similar motions (e.g., squatting in front of a tire). Making an action classification system robust to inter-class confusion is thus hard to design.

Given the above mentioned challenges, a lot of manually-annotated training data is required. We, thus, want to reduce the human supervision. The motivation is two-fold. First, as Internet data are nowadays the largest available source of information, we want to automatically use web videos as training data. Second, a fully-supervised model, i.e., which requires annotating each video with an action label, can be extended to many action classes only at the cost of an important annotation effort. We thus want to design a system which requires no human annotation (zero-shot learning).

**Human action localization in real-world videos.** Besides classification, we aim to localize human actions in real-world videos. The task is challenging because people can be occluded by other objects or truncated by image boundaries. Their bodies can undergo deformations over time in the video. Figure 1.2 shows some examples of these challenges for the event *cleaning window* from the DALY dataset [Weinzaepfel et al., 2016]. Besides these difficulties, visible human body parts change over time, making it hard to design a robust tracker. For instance, video 1 in Figure 1.2 shows the upper body of the person performing the action, then the legs, and finally his right arm. We, thus, want to design an human action localization model which is robust to occlusions, viewpoint changes,



Figure 1.2 – Samples from three videos from the DALY dataset [Weinzaepfel et al., 2016]. Camera viewpoints and visible body parts change in a video.

and deformations.

## 1.2 Context

Automatic video understanding is becoming a crucial task in computer vision. We first describe the context for action classification (see Section 1.2.1). Then, we briefly review zero-shot and webly-supervised techniques adapted to action classification in Section 1.2.2. Finally, we describe techniques for localizing actions in Section 1.2.3.

### 1.2.1 Action classification

For action classification, the popular pipeline consists in extracting spatio-temporal descriptors in a video, aggregating them into a single global vector, which is then used to classify this video with an action label. Spatio-temporal local descriptors are often an extension of 2D de-

scriptors which have been successfully crafted for image processing tasks like object detection. These features describe the appearance (e.g., 3D-HOG [Kläser et al., 2008]) or the motion (e.g., HOF [Dalal et al., 2006]) of a very delimited spatio-temporal volume of the video. Thanks to their robustness, spatio-temporal descriptors have been successfully applied in action recognition tasks.

The set of extracted spatio-temporal video descriptors is then embedded into a global vector by a bag-of-features procedure [Sivic and Zisserman, 2003, Lazebnik et al., 2006] which assigns for each of them its closest “visual word”. In other words, the global vector can be interpreted as an histogram of visual cues. This original model does not consider spatio-temporal relations between descriptors. Extensions have been proposed to structure the information contained in the global vector [Lazebnik et al., 2006], by splitting a spatio-temporal cube into bins where the bag-of-features procedure is computed independently per bin and then aggregated into the final vector. This technique, called Spatial Pyramid Matching (SPM), improves the performance of methods based on visual words by a significant margin. After aggregating the set of descriptors, machine learning tools learn to classify videos from features. In computer vision, one of the most popular technique is SVM [Vapnik, 1998].

In the past few years, deep-learning approaches have enabled a breakthrough in computer vision. For action recognition, several recent methods integrate the temporal dimension and motion into a CNN. The most natural approach is to extend 2D-filters to 3D-filters in order to incorporate the temporal dimension [Ji et al., 2013, Karpathy et al., 2014, Tran et al., 2015, Carreira and Zisserman, 2017]. An other approach is to build a two-stream network with 3D filters [Simonyan and Zisserman, 2014b, Feichtenhofer et al., 2016, Carreira and Zisserman, 2017]: one stream for RGB images, the another one for optical flow images. Such architecture combines appearance and motion information for classifying actions. Finally, a third approach builds recurrent network (RNN, e.g., a stack of LSTM layers) upon a frame-based CNN and enables to learn long-term correlations between frames [Donahue et al., 2015]. With large-scale datasets as a requirement for training, CNNs make the need for training data even more crucial than before.

## 1.2.2 Zero-shot and webly-supervised techniques

Zero-shot techniques [Niebles et al., 2008, Wang and Chen, 2016, Lampert et al., 2014, Palatucci et al., 2009, Mensink et al., 2014, Habibian

et al., 2014, Hussein et al., 2017] attempt to recognize objects whose instances may not have been seen during training, i.e., without any training data, but with some kind of annotation. A learning step is still required, but zero-shot learning aims at generalizing the training for new tasks or new classes. For example, if a system is very efficient at distinguishing dogs from cats, one would also want to recognize rabbits from dog and cat classes without annotating a thousand of additional rabbit images. The most popular way to overcome the lack of data is to map the image feature space to a semantic space which describes an image in terms of visual attributes [Mensink et al., 2014, Habibiian et al., 2014, Hussein et al., 2017, Wang and Chen, 2016], e.g., the object is furry or the object has long ears. The semantic space is directly linked to the class space, either by user annotation or by using a knowledge base like Wordnet [Miller, 1995]. For actions, visual attributes can be objects associated to a verb [Mensink et al., 2014, Habibiian et al., 2014] or a mixture of semantically close actions [Hussein et al., 2017, Wang and Chen, 2016]. Although zero-shot techniques have shown promising results, they still require a dataset to learn a model. Moreover, the semantic environment for actions face intra-class variation challenges, making a direct mapping hard to design. For example, the events *changing car tire* and *change bike tire* involve different objects, movements, and places. Actions must somehow be analyzed in order to better take into account all these variations.

More recently, webly-supervised approaches [Song et al., 2010, Leung et al., 2011, Wang et al., 2010, Ye et al., 2015, Chen et al., 2014a, Liu et al., 2013, Gan et al., 2016a,b, Ikizler-Cinbis et al., 2009, Chen et al., 2013, Duan et al., 2012, Niu et al., 2015, Nguyen et al., 2016] use web content to learn a task without additional annotation. Web content can be images or videos. Web data is downloaded by a set of queries and is used to learn a pre-designed model. A popular approach is to learn from web content some visual concepts [Chen et al., 2014a, Liu et al., 2013, Gan et al., 2016a,b, Nguyen et al., 2016], or semantic attributes, adopting a similar approach to zero-shot learning technique as mentioned above. Although these approaches have made notable progress, they fall short in one or more of the following ways: (i) lack of ability to exploit rich cues present in textual description of events [Ikizler-Cinbis et al., 2009, Niu et al., 2015, Duan et al., 2012, Gan et al., 2016a], (ii) reliance on some form of manual annotation [Niu et al., 2015, Duan et al., 2012, Chen et al., 2013], (iii) limited to using image data [Ikizler-Cinbis et al., 2009, Singh et al., 2015, Gan et al., 2016a] or micro-videos (i.e., video shorter than 5 seconds [Nguyen et al., 2016]). Moreover, all these recent approaches use web data with-



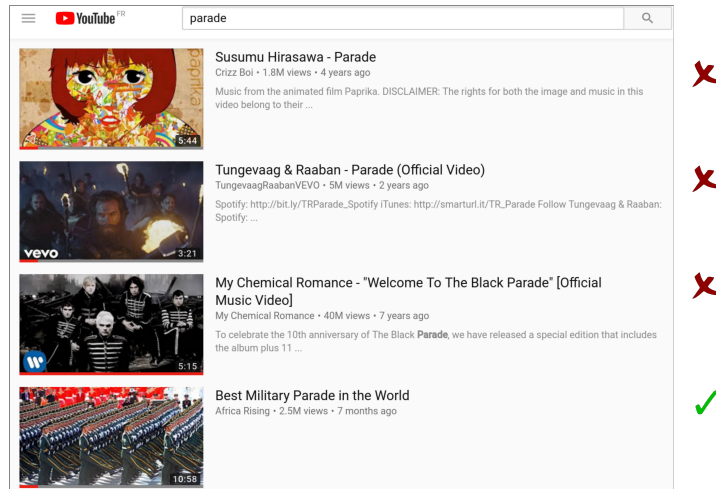


Figure 1.3 – Overview of an action query on YouTube. The query is “parade”, one of the ten TrecVid11 events [Over et al., 2010]. Textual information (metadata) are shown and provide a reliable source of information for action recognition.

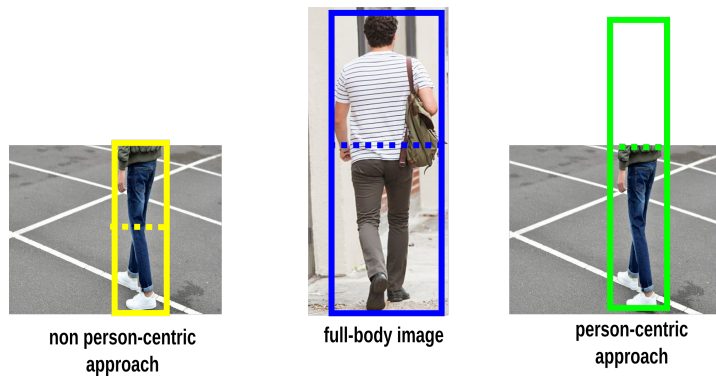


Figure 1.4 – Illustration of a person-centric approach and comparison with a non-person-centric approach.

out pruning it, making the training dataset very noisy. Figure 1.3 shows an example for the event *parade*. Music clip videos whose title contain the word “parade” are mixed with videos where a real parade is shown. YouTube video ranking is based on a combination of criteria including popularity, publicity boosting, user preference and originality. For action recognition, videos which do not show any occurrence of the action must somehow be removed from the downloaded data.

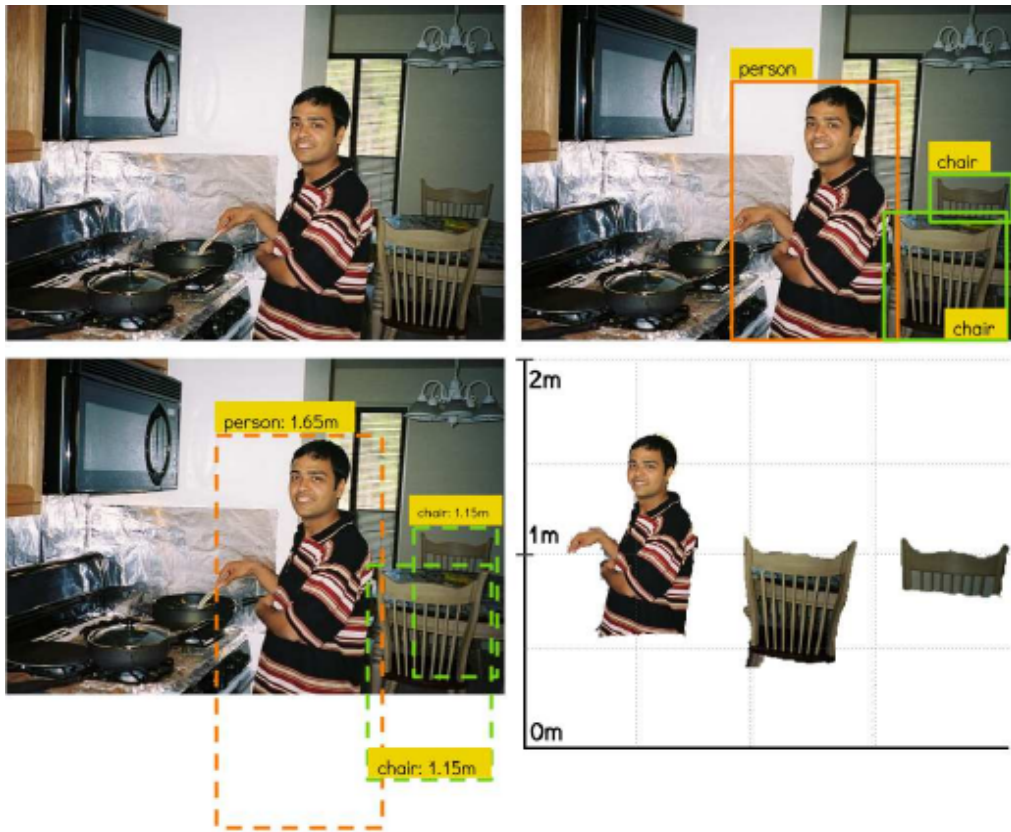


Figure 1.5 – Illustration of the amodal completion method which infers the vertical size of object (Illustration courtesy of [Kar et al., 2015]).

### 1.2.3 Spatio-temporal action localization

For action localization, three approaches have been proposed during the last years: frame-based approaches [Yuan et al., 2009], human-track based approaches [Lan et al., 2011] and tubelet-based approaches [Jain et al., 2014]. The first frame-based models were either based on cuboids [Laptev and Pérez, 2007, Cao et al., 2010, Yuan et al., 2009] or on figure-centric models [Kläser et al., 2010, Lan et al., 2011]. Cuboids, i.e., fixed human positions over frames, can not generalize to the case of moving actors or moving camera. Figure-centric models leverage a human detector [Kläser et al., 2010] or treat the actor position as a latent variable [Lan et al., 2011]. As actions are usually people-centered, human-track based approaches aim to detect, then track people through time, and infer actions. In [Moeslund et al., 2006], a four-step human action recognition system is proposed. The first stage detects a person visible in the first



frame of a video. Second, the detected person is tracked throughout the video. Third, the human track is used to model the tracked person, using for instance a pose representation. This modeled feature is finally used for classification. More recently, other approaches, called tubelet approaches, have been proposed, based on extensions of successful methods for object detection in images, such as part-based models or proposals. For instance, [Tian et al. \[2013\]](#) extends the deformable parts model proposed in [[Felzenszwalb et al., 2010](#)] to videos. Proposals have been extended to actions in videos, for instance based on clustering supervoxels [[Jain et al., 2014](#), [Oneata et al., 2014](#)] or trajectories [[van Gemert et al., 2015](#), [Marian Puscas et al., 2015](#)]. [Kalogeiton et al. \[2017a\]](#) extends the single shot multibox detector (SSD) to generate action tubelets. In [[Peng and Schmid, 2016](#)], a two-stream network combines spatial proposals and motion proposals. As in classification task, deep-learning approaches provides “all-in-one” methods where people simultaneously are tracked and actions are scored. All these methods detect visible parts of human people, without taking into account the full-body. Figure 1.4 shows the importance of estimating full-body location. If we split the bounding box in two halves, e.g., as done in Spatial Pyramid Matching (SPM, see Section 1.2.1), leg movements are included in the bottom part of the blue bounding box and the green bounding box, whereas the yellow square splits the legs into two different parts, making correspondences less efficient and thus diluting information.

This type of modality, i.e., inferring from visible parts the location of the full-body, is called amodal completion. Such approach has been proposed in [[Kar et al., 2015](#)], showing that inferring the full-size of an object helps for scene understanding, see Figure 1.5 for an illustration. For action localization, no amodal completion work has been yet proposed, although estimating the full-body localization defines a better reference frame for features (spatio-temporal grid is more adapted as person-centric).

### 1.3 Contributions

This PhD presents two contributions: (i) how to learn actions from web medias without any supervision, (ii) a person-centric approach for action localization.

**Learning actions from web media without any supervision.** We propose a novel webly-supervised approach which leverages the synergy

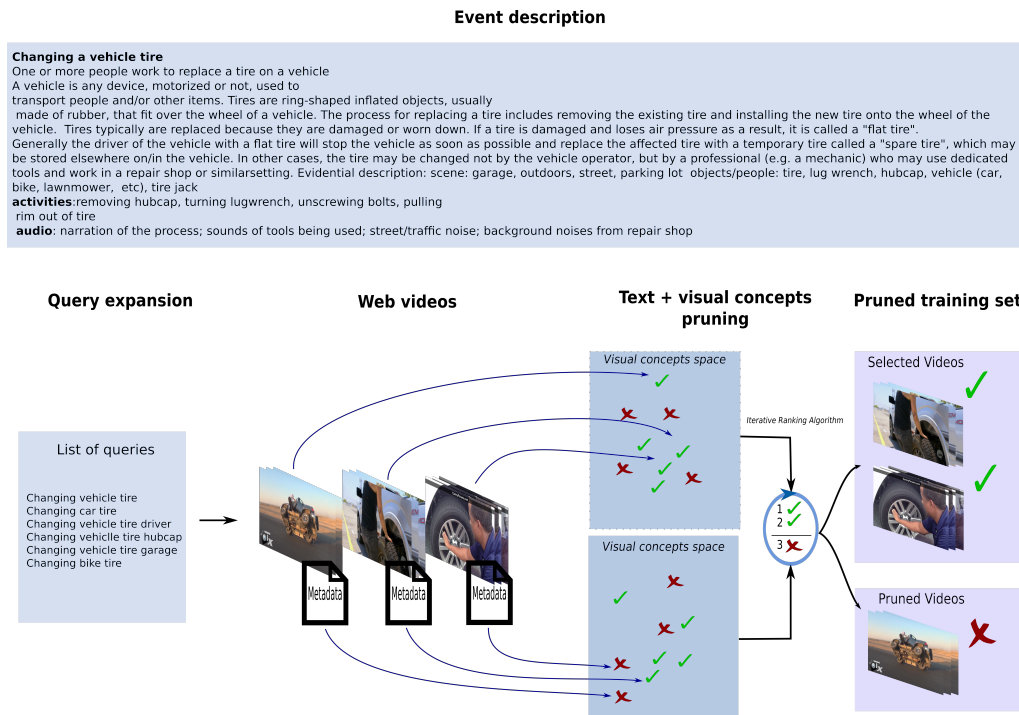


Figure 1.6 – Overview: Given a textual description of an event (“Event description”), relevant queries are automatically generated (“Query generation”) to collect an initial training set (“Web videos”). Text metadata and visual concepts extracted from these videos are used to select the relevant ones automatically (“Text + visual concept pruning”), and to build a training set for event classification (“Pruned training set”).

between visual video data and the associated textual metadata, to learn event classifiers with no manual annotation. Specifically, we first collect a video dataset with queries constructed automatically from textual description of events, prune irrelevant videos with text and video data, and then learn the corresponding event classifiers. Figure 1.6 gives an overview of our method. The query expansion step is designed to tackle intra-class variation and inter-class confusion problems. For an event (e.g., *changing vehicle tire*), fine-grained queries are created with different strategies involved, e.g., replacing a word by a more specific one (“changing a bike tire”), or adding a clue word (“changing a vehicle tire with a jack”).

The “text + visual concept” pruning step selects videos that both share textual and visual similarities. We evaluate this approach in the challeng-

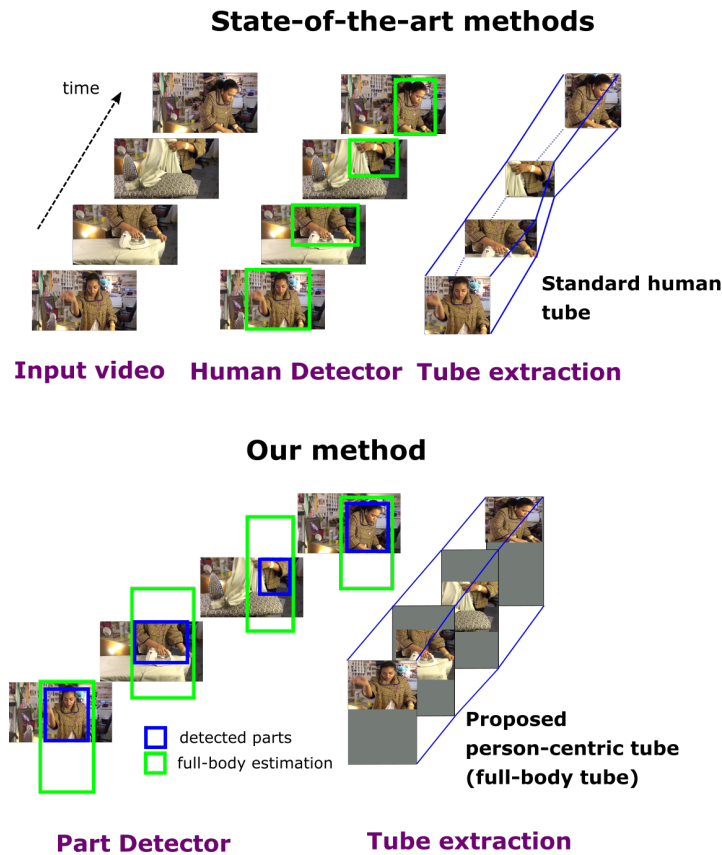


Figure 1.7 – Overview of our person-centric tube approach. Given an input video, classic approaches detect visible parts of a person, which then is used to build through time a standard human tube. Our approach also detects visible parts, and infers from them location of the full-body. Full-body inferences are then used to create robust person-centric tube.

ing setting where no manually annotated training set is available, i.e., EK0 in the TrecVid challenge, and show state-of-the-art results on MED 2011 and 2013 datasets. This work is presented in Part I and is to appear in Transactions on Circuits and Systems for Video Technology [Chesneau et al., 2017a].

**Action Localization in real-world videos: a person-centric approach.** We propose a new person-centric framework for action localization that tracks people in videos and extracts full-body human tubes. The motivation is two-fold. First, it permits to handle occlusions and camera viewpoint changes when localizing people, by detecting human visible

parts while inferring full-body localization. Second, it provides a better reference grid for extracting action information. The intuition behind this is that establishing spatial correspondences between frames of the same sequence, but from different camera viewpoints or with partially occluded bodies, is more efficient with a person-centric approach than a non-person centric approach (see Figure 1.4). Our method trains a novel human part detector that scores visible parts while regressing full-body bounding boxes, even when they fall outside the frame. The core of our detector is a convolutional neural network which learns part proposals specific to certain body parts. Before training, the part classes are automatically defined from the set of training proposals, by clustering the relative position of parts in full-body bounding boxes. Then, our part detector learns to detect simultaneously full-body and parts, while regressing full-body bounding boxes for all classes. Part detections are then used to create robust full-body human tubes in videos. Our tracking algorithm connects the part detections temporally by learning their appearance through frames, and combines them to extract full-body human tubes. Figure 1.7 shows an example of our tubes. We apply our new tube extraction method to the problem of human action localization, on the popular JHMDB dataset [Jhuang et al., 2013b], and a very recent challenging dataset DALY (Daily Action Localization in YouTube, see [Weinzaepfel et al., 2016]), showing state-of-the-art results. We prove that inferring the full-body location in video helps for recognizing and locating actions. This work is presented in Part II and was published in BMVC [Chesneau et al., 2017b].

# **Part I**

## **LEARNING ACTIONS FROM WEB DATA**

In this part, we propose a novel webly-supervised approach which leverages the synergy between visual video data and the associated textual metadata, to learn event classifiers with no manual annotation.

# Chapter 2

## Related Work

### Contents

---

2.1	Text classification . . . . .	14
2.1.1	Pre-processing step . . . . .	15
2.1.2	Text features . . . . .	17
2.2	Action recognition . . . . .	19
2.2.1	Local feature-based methods . . . . .	19
2.2.2	Deep learning approaches . . . . .	21
2.3	Zero-shot and webly-supervised learning . . . . .	23

---

This chapter describes the related work on text and video classification, and zero-shot learning techniques. We start by reviewing text classification in Section 2.1. Text content was the first media where the need of classifying documents arose in computer science, and many of these methods have been adapted to image and video classification. We then present an overview of action recognition in Section 2.2. Finally, we review related work on zero-shot and webly-supervised techniques in Section 2.3.

### 2.1 Text classification

For text classification, the standard learning approach is three-fold. First, each document of the corpus is converted into an interpretable format, typically a set of stemmed words, see Section 2.1.1. Document features are then extracted (Section 2.1.2). In the final step, a classifier is learned, e.g., using support vector machine [Vapnik, 1998].

### 2.1.1 Pre-processing step

Before extracting features, a pre-processing step is required to transform corpus documents into a list of words or terms. For some features (e.g., tf-idf, bigrams, see Section 2.1.2 for more details), a dictionary is also built. The importance of preprocessing text was studied in [Uysal and Gunal, 2014]. Several steps are analyzed and combined: tokenization (procedure of splitting a text into a list of words), lowercase conversion (i.e., converting each character into lowercase format), stemming words (i.e., obtaining the word root form of word derived forms), stop words removal (i.e., removing from the list of root form words basic terms, such like 'be', 'in'...). Only the tokenization operation is absolutely necessary during the preprocessing step, as it creates the list of terms. The three other operations, i.e., lowercase conversion, stemming word, stop words removal, can be added to the tokenization. Some studies [Méndez et al., 2005, Pomikálek and Řehůřek, 2007] claim that adding these three steps does not improve performance for text classification, but other work [Uysal and Gunal, 2014] shows that a good combination of the four operations can improve global accuracy. Preprocessing text remains a handcrafted operation whose efficiency depends on the corpus domain and language.

During tokenization, every document is transformed into a list of terms. Typically, the tokenization is carried out considering only alphabetic or alphanumeric characters that are delimited by non-alphanumeric characters (e.g. punctuation, whitespace). For example, the text "Good muffins cost \$3.88 in New York. Please buy me ...two of them.Thanks." is converted into the following list: "[ 'Good', 'muffins', 'cost', '\$', '3.88', 'in', 'New', 'York', '.', 'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.' ]". In TREC contest, Tomlinson [2003] pointed out that allowing a token to contain both alphabetical and numerical characters was slightly better than separating them for medical information retrieval. In contrast, Pirkola and Leppänen [2003], Crangle et al. [2004] chose to separate alphabetical and numerical strings in order to handle hyphenation of the alphabetical and the numerical parts in a gene name. Pirkola and Leppänen [2003] also imposed proximity search to ensure that the separated components were close together in the retrieved documents. Nevill-Manning and Oughtred [2003] allowed the following special characters to be part of a token provided that they were not the first or the last character of the token: '( ', ')', '[ ', ']', '0', '-', ',', and '/'.

After the tokenization, the stemming operation is an important step for text classification. Stemming is the term used in linguistic morphol-

ogy and information retrieval to describe the process for reducing inflected (or sometimes derived) words to their word stem. The Porter Stemming Algorithm (also called “Porter stemmer” [Porter, 1980]) was built on the assumption that a stem dictionary is not provided or available and that the purpose of the task is to improve information retrieval performance (not as a linguistic exercise). The program is given an explicit list of suffixes, and, with each suffix, the criterion under which it may be removed from a word to obtain a valid stem. Terms with a common stem will usually have similar meanings, for example: “connect”, “connected”, “connecting”, “connection”, “connects”. Each of these terms are grouped into their stem: “connect”.

As an alternative to the Porter algorithm, WordNet morphological function [Harabagiu et al., 1999] uses a two-step approach to convert a word into an entry of the WordNet database. The first step is the search of an inflected form, based on exception list files (one for each syntactic category). The exception lists contain the stemming operation for strings or words that are not regular. The second step is to stem the word based on its syntactic category, in order to find a form of the word that is in WordNet. Each word is thus associated with a syntactic category, e.g., noun or verb. A grammatical analysis can be performed to tag each term according to its syntactic category.

In addition to the stemming, a grammatical analysis, i.e., tagging each term according to its syntactic category, can be performed to leverage ambiguity between homonyms for derived forms. For example, the word “leaves” is transformed into “leave” if the tag is *verb*, but is transformed into “leaf” if the tag is *noun*. Without a grammatical analysis, words are stemmed with its default syntactic form. In Ratnaparkhi [1996], a statistical model is trained from a corpus annotated with part-of-speech tags and predicts tags to previously unseen text. The probability assigned to a tagged sequence of words  $x = \langle \mathbf{t}, \mathbf{w} \rangle$ , where  $\mathbf{w}$  is the sequence of terms or words, and  $\mathbf{t}$  is the sequence of tags (e.g, noun), is:

$$P(\mathbf{t}, \mathbf{w}) = \prod_i P(t_i | t_{i-1}, w_i) \quad (2.1)$$

The tag at position  $i$  depends only on the preceding tags. In Toutanova et al. [2003], a max-entropy based model is built:

$$P_\lambda(t_0 | t_{-1}, w_0) = \frac{\exp(\lambda_{\langle t_0, t_{-1} \rangle} + \lambda_{\langle t_0, w_0 \rangle})}{\sum_{t'_0} \exp(\lambda_{\langle t'_0, t_{-1} \rangle})} \quad (2.2)$$



As in (2.1), the tag  $t_0$  depends only on the previous tag  $t_{-1}$  in (2.2). As the following tag  $t_{+1}$  also carries useful information about the current tag  $t_0$ , replicated models  $P_\lambda(t_0|t_{-1}, t_{+1}, w_0)$  of that in (2.2) are used to tag words of a sentence. The software POS Tagger<sup>1</sup> is based on this bidirectional dependency network.

### 2.1.2 Text features

Once the documents are transformed into a list of terms, text features can be computed. One of the most popular methods to compute an interpretable feature from set of words for text retrieval is the bag-of-words approach. An early reference to this approach can be found in [Harris, 1954]. Since then, many works have used bag-of-words procedure for text retrieval [Luhn, 1957, Meadow, 1992, Blair and Maron, 1985]. This approach transforms a set of words into a histogram of word occurrences, with a dictionary of  $d$  words. The dictionary can be a classic one, like Oxford dictionary [dic, 2007] or Collins [Hanks, 1986], but in most cases it is computed from the corpus of documents. The document is thus represented by a  $d$ -dimensional feature, with the number of entries  $d$ , or words, in the dictionary. The values are the number of word occurrences. A vector normalization, e.g., a  $L_2$ -normalization or a normalization by the number of total words of the document, is also applied.

Vector Space Model (VSM) feature can be viewed as an extension of the bag-of-words and was first used in the SMART Information Retrieval System [Salton, 1971]. Since then, a lot of literature has emerged. In [Salton et al., 1975], the authors study the importance of term frequency (tf) and inverse document frequency (idf) in a corpus to compute similarity between documents. Tf-idf is a weighting approach and the product of two statistics: tf (term frequency) and idf (inverse document frequency). Tf measures the occurrence of a term (or a word) in a document, as in a bag-of-words procedure. Idf measures the proportion of documents in which the word appears in the corpus. Typically, a word contained in a few documents should be more discriminant than a word contained in all documents. For information retrieval, a document can be compared with a query by computing their cosine-similarity :

$$sim(t_j, q) = \frac{t_j \cdot q}{\|t_j\| \|q\|}, \quad (2.3)$$

---

1. <https://nlp.stanford.edu/software/tagger.shtml>

where  $q, t_j$  are tf-idf vectors of the query, and the  $j$ -th document, respectively. If the vectors are normalized, the similarity score simply becomes:

$$\text{sim}(t_j, q) = t_j \cdot q. \quad (2.4)$$

For information retrieval, a matrix  $D \in \mathbb{R}^{N \times d}$  containing the normalized features of all the  $N$  documents is built (the  $j$ -th row in this matrix corresponds to  $t_j$ ). Similarities of all the documents are computed as:

$$\text{sim}(D, q) = D \cdot q. \quad (2.5)$$

Documents are then sorted with respect to their score. An alternative to tf-idf weighting scheme was proposed by Niwa and Nitta [1994] by incorporating semantics similarity at word level. A word-similarity matrix  $S \in \mathbb{R}^{d \times N}$  is built and incorporated into the similarity equation 2.5:

$$\text{sim}(D, q) = D \cdot S \cdot q, \quad (2.6)$$

where  $S_{w_1, w_2}$  denotes the similarity between two words. Niwa and Nitta [1994] define  $S_{w_1, w_2}$  as the mutual information:

$$S_{w_1, w_2} = \log^+ \frac{P(w_1|w_2)}{P(w_1)}, \quad (2.7)$$

where  $P(w)$  is the occurrence density of word  $w$  in the whole corpus and the conditional probability  $P(w_1|w_2)$  is the density of  $w_1$  in a neighborhood of word  $w_2$ . Two words are thus considered to be similar if they appear frequently and closely in documents.

Bigrams [Tan et al., 2002] is a VSM technique where entries in the dictionary are either words or a sequence of two words. The aim is to refine the corpus. For example, one may wish to integrate the term “new york” in addition of “new” and “york”. Bigram features can be viewed as an extension of tf-idf features.

More recently, a concept-based retrieval model was introduced in [Shehata et al., 2013], which consists of conceptual ontological graph (COG) representation and conception-based weighting scheme. The COG representation captures the semantic structure of each term within a sentence. Then, all the terms are placed in the COG representation according to their contribution to the meaning of the sentence. The concept-based weighting analyzes terms at the sentence and document levels, unlike tf-idf representation which analyzes the contribution of a term only at the document level.

Introduced by [Lodhi et al. \[2002\]](#), string subsequence kernels (SSK), is a different approach that considers documents simply as symbol sequence. The approach does not use any domain knowledge, in the sense that it considers the document just as long sequences, but is capable of comparing texts by computing a similarity kernel based on common character string. The feature space in this case is generated by the set of all (non-contiguous) sub-strings of  $k$ -symbols. The more sub-strings two documents have in common, the more similar they are considered. No preprocessing step is required for the kernel computation.

Word2Vec descriptors were first introduced in [Mikolov et al. \[2013a\]](#). Word2Vec is a group of related models that are used to produce word embeddings. Word representations are computed using neural networks and explicitly encode many linguistic regularities and patterns. The training objective of the skip-gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document. After the preprocessing step, each word of a corpus document is encoded into a 300-dimensional Word2Vec vector. Aggregations techniques like Fisher vector (see Section 2.2) can be applied to represent the document.

## 2.2 Action recognition

### 2.2.1 Local feature-based methods

The standard approach to address the action recognition problem is to extract features from videos, learn a classifier for each event with a training set, and then evaluate it on the test set. Features extraction has been widely studied in the past ten years, resulting in tremendous progress in computer vision. Several methods [[Laptev, 2005](#), [Wang et al., 2013](#)] are based on local features: a video is represented as a collection of descriptors, representing small volumes or sequences of image patches. In contrast to global representation, local features are robust under a variety of video settings thanks to the absence of a strict assumption on the global structure of the action.

**Spatio-temporal features.** In [Laptev \[2005\]](#), an extension of local image features, or interest points, is built and used for a compact representation of video data. The neighborhood of spatio-temporal interest points is described by a set of spatio-temporal Gaussian derivatives normalized with respect to the scale. In [Dalal and Triggs \[2005\]](#), histogram of ori-

ented gradients (HOG) descriptor is described for human detection in images. For each cell of a dense grid, an histogram of gradient intensity and direction is computed and normalized with respect to local contrast. HOG descriptors have been extended to video content by Kläser et al. [2008]. The quantification step is done with a regular polyhedron. HOG captures static appearance information, whereas histograms of optical flow descriptors (HOF) focus on the local motion information by using optical flow maps [Laptev et al., 2008]. In Dalal et al. [2006], a new motion descriptor, called Motion Boundary Histograms (MBH), is built by computing histograms of oriented gradients on horizontal and vertical differential optical flow.

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

Consequently, many approaches use an aggregation of short-term trajectories (around 15 frames), often referred to as tracklets [Matikainen et al., 2009, Wang et al., 2013]. In Wang et al. [2013], trajectories are extracted from a dense sampling grid of video frames at several scales. Points within a uniform area are removed, and other points are tracked with respect to a local analysis of the optical flow. The coordinates of trajectories are stacked and normalized, creating a fixed-length descriptor. Besides trajectory-based descriptors, HOG, HOF, MBH descriptors are computed along them in the video. An improved version of dense trajectories [Wang and Schmid, 2013] removes camera motion from optical flow, by matching SURF patches [Bay et al., 2006] along the video. The proportion of trajectories extracted within a human body is more important when removing the outliers, making the process more efficient for action recognition.

**Aggregation.** The representation of a video is finally done by aggregating the set of descriptors into a global vector. Originally designed for text retrieval, the bag-of-words method (see Section 2.1), was adapted for video processing [Sivic and Zisserman, 2003]. During pre-processing step, a “dictionary” of  $d$  visual words is created by performing a k-means clustering over a large set of spatio-temporal descriptors. Then, each descriptor extracted from the image is associated to a visual word by a

nearest search over the visual words, i.e., the cluster. The feature is a  $d$ -dimensional vector and represents the number of occurrences of each visual word. [Csurka et al. \[2004\]](#) successfully apply this approach with a bag-of-keypoints aggregation for object categorization in images.

Improved aggregation techniques have also been proposed over the years. [Van Gemert et al. \[2010\]](#) propose a soft assignment scheme instead of the hard assignment in bag-of-words procedure. Several techniques like VLAD [[Jegou et al., 2012](#)], super vector coding [[Zhou et al., 2010](#)], and Fisher vector [[Perronnin et al., 2010](#)] can be viewed as an extension for higher-level statistics of the traditional bag-of-words embedding. VLAD and super vector coding represents the mean of the descriptors assigned to the same visual word. Fisher vector includes second-order statistics, i.e., the variance of the assigned descriptors. Since the dimensionality of the Fisher vector is high (typically  $2d(m + 1)$  where  $m$  is the dimensionality of the descriptors,  $d$  the number of Gaussians, i.e., the number of visual words), a PCA is often required to reduce the dimensionality  $m$  of descriptors before the Fisher vector embedding.

These aggregation techniques do not model any geometric relation between feature points. Several extensions incorporating spatio-temporal relations between local features have been proposed. For instance, [Laptev et al. \[2008\]](#) extend the spatial pyramid [[Lazebnik et al., 2006](#)] to videos, i.e., the temporal dimension: the aggregation is computed for several video volumes and then concatenated. In the same spirit, [Gaidon et al. \[2013\]](#) concatenate bag-of-words representations for three sub-actions, i.e., an action is split into three sub-actions. Another example is to pool the features over supervoxels as proposed by [[Taralova et al., 2014](#)] or [Peng et al. \[2014\]](#).

## 2.2.2 Deep learning approaches

In the past few years, many works extend to video the recent success of deep convolutional neural networks (CNN) in image processing. Indeed, deep learning methods have obtained tremendous results in computer visions tasks such as image classification [[Krizhevsky et al., 2012](#), [Simonyan and Zisserman, 2014a](#)] and object detection [[Felzenszwalb et al., 2010](#)]. [Krizhevsky et al. \[2012\]](#) proposed a new CNN architecture called AlexNet with five convolutional layers, three fully-connected layers, a softmax layer which enables to output a probability distribution over a set of classes. A new ReLU function  $f(x) = \max(0, x)$  is proposed, which makes the network convergence faster. In 2012, the network achieved a top-5 error of 15.3%, more than 10.8% ahead of the runner up. Since

then, deeper networks have been proposed. In the same spirit, VGG networks [Simonyan and Zisserman, 2014a] uses  $3 \times 3$  convolution filters. In He et al. [2016], a novel learning scheme, called residual learning, avoids vanishing gradients by directly linking the input and the output of a layer. Thanks to this technique, deeper architectures can be learned.

The challenge of adapting deep architectures to the action recognition task has been extensively studied in the past few years. The most natural approach is to extend 2D-filters to 3D-filters in order to incorporate the temporal dimension [Ji et al., 2013, Karpathy et al., 2014, Tran et al., 2015]. In [Ji et al., 2013], seven frames of  $60 \times 40$  resolution are considered as inputs to the 3D-CNN model. Tran et al. [2015] prove that using  $3 \times 3 \times 3$  filter improves performance of 3D models in action classification. Karpathy et al. [2014] compare several methods to merge frame-level information into a global network. Although these methods show interesting results, the restriction of the number of frames in the input layer, mostly due to computational reasons, do not allow 3D-CNN to capture long-term dependencies, which are crucial for complex action recognition tasks. Another approach is to leverage the Recurrent Neural Network (RNN, see [Donahue et al., 2015, Baccouche et al., 2011]). Long Short-Term Memory (LSTM) network is one the most popular RNN implementations and was first proposed by Hochreiter and Schmidhuber [1997]. An LSTM unit remembers values for either long or short time periods. By not using activation functions, the stored value is not iteratively modified, and the gradient does not tend to vanish when trained with backpropagation through time. For action recognition, an LSTM layer is built upon frame-level AlexNet CNNs [Donahue et al., 2015]. LSTM layer enables to learn long-term correlations between frames. An other approach is to build a two-stream network for action recognition [Simonyan and Zisserman, 2014b]. A spatial stream-CNN takes as input a single frame and describes its static appearance. A temporal stream takes as input several frames of optical flows and captures motion information. The two stream scores are then merged into a class score fusion. The method shows that combining static and motion information is useful for action recognition.

Although all these methods show a significant gain in performance for the action recognition task, they require large quantities of data to train models. For instance, AlexNet-CNN was trained on the ImageNet dataset with over 14 millions images. Some techniques, called zero-shot techniques, aim to learn classifiers or models with a limited amount of annotated data.



## 2.3 Zero-shot and webly-supervised learning

The traditional setup for event classification is where a dataset of labeled videos is provided to train models [Laptev and Pérez, 2007, Ke et al., 2005, Laptev et al., 2008, Liu et al., 2009, Wang et al., 2011, Karpathy et al., 2014, Simonyan and Zisserman, 2014b, Xu et al., 2015, Gan et al., 2015]. Several innovative approaches have been proposed to address the availability of limited or no (zero-example) training data for event classification. Niebles et al. [2008] represented events, in particular those performed by humans, with topic models, which were learned with probabilistic latent semantic analysis and latent Dirichlet allocation. This framework was however evaluated on a limited set of sequences, and it is unclear if it would generalize well to the unconstrained setting we consider in this thesis. Other approaches like [Ikizler-Cinbis et al., 2009, Chen et al., 2013, Duan et al., 2012, Niu et al., 2015, Gan et al., 2016a] have used web resources to collect a training set. For example, Ikizler-Cinbis et al. [2009], Chen et al. [2013], Niu et al. [2015] learn a classifier with an initial training set, and then use it to collect additional samples from the web, with Google, Bing and YouTube search. Duan et al. [2012] proposed a transfer learning scheme on videos collected from YouTube. Although these approaches have made notable progress, they fall short in one or more of the following ways: (i) lack of ability to exploit rich cues present in textual description of events [Ikizler-Cinbis et al., 2009, Niu et al., 2015, Duan et al., 2012, Gan et al., 2016a], (ii) reliance on some form of manual annotation [Niu et al., 2015, Duan et al., 2012, Chen et al., 2013], (iii) limited to using image data [Ikizler-Cinbis et al., 2009, Singh et al., 2015, Gan et al., 2016a]. We address these limitations in this thesis with an approach exploiting additional cues in text metadata, in the challenging setting where no manual annotation is available for large TrecVid datasets, i.e., TrecVid EK0.

One way to address some of the limitations discussed above is by using text as additional information [Song et al., 2010, Leung et al., 2011, Wang et al., 2010, Ye et al., 2015], inspired by early methods for video segmentation [Hauptmann and Smith, 1995] and video summarization [Smith and Kanade, 1997]. Such techniques were seldom deployed on a large scale, and were ahead of their time. Song et al. [2010] adapt classifiers learned on labeled text documents to videos, by treating them as weak classifiers in a boosting framework. A strong video classifier is then learned by combining the weak responses with a classifier trained on labeled videos. Another boosting approach [Leung et al., 2011] combined text metadata and video feature classifiers in a Multiple Instance Learn-



Figure 2.1 – An example of between-class confusion. Sample frames from two events: *changing vehicle tire* (top) and *unstuck vehicle* (bottom). These events are very close in terms of semantic content and motion, and are nearly impossible to distinguish from images alone.

ing (MIL) framework, but relied on a training set of videos, annotated by experts and amateur human labelers. Similarly, the method in [Wang et al., 2010] requires a manually curated initial training set to extract additional data from webpages and related videos. While these methods demonstrated the benefits of using text, they still require video annotations, unlike our method presented in Chapter 3, where no video is manually labeled.

An alternative way to use text in combination with visual features is to define a set of concepts, with individual words or short phrases, that describe events, and are simpler to learn. Works such as [Chen et al., 2014a, Liu et al., 2013, Gan et al., 2016a,b] learn a model to detect events with such concepts. Improvements to this scheme include modeling a pair of words or n-grams to not only exploit the co-occurrences between words [Mensink et al., 2014], but also disambiguate among the multiple meanings represented by individual words [Divvala et al., 2014]. Works in this paradigm, where events are represented as a collection of concept responses, are increasingly leveraging weakly annotated data from the internet [Chen et al., 2014a, Wu et al., 2014]. The method in Ye et al.



[2015] builds a list of frequent words in the text metadata, which represent concepts, to prune videos downloaded from YouTube. Singh et al. [2015] present a similar approach, where an initial set of concepts is extracted from the textual description of an event, which is then pruned to ultimately obtain an image dataset for training visual concepts. Despite promising results, this method solely relies on images to differentiate between events. Consider two example events from the TrecVid event retrieval challenge: *changing vehicle tire* and *unstuck vehicle*, see Figure 2.1. It is nearly impossible to distinguish between these events simply from images. Jiang et al. [2015b,a] use video data instead of images, and a self-paced learning scheme to train concept detectors, but still require a small set of reliably-annotated video samples. A very recent work [Hussein et al., 2017] simultaneously learns a CNN for video content embedding and a multi-layer-perceptron for textual description of the event. Videos can be ranked according to a previously unseen new event by computing the distance between the textual representation and video representation projected in the same latent space. As in Jiang et al. [2015b,a], a learning step is still required to learn how to bridge textual and video content.

# Chapter 3

## Learning from Web Videos for Action Classification

### Contents

---

3.1	Introduction . . . . .	26
3.2	An overview of text classification . . . . .	29
3.2.1	Methods . . . . .	30
3.2.2	Dataset . . . . .	33
3.2.3	Implementation details . . . . .	34
3.2.4	Results . . . . .	35
3.2.5	Summary . . . . .	38
3.3	Learning from Web Videos . . . . .	38
3.3.1	Textual query generation . . . . .	38
3.3.2	Pruning with text and visual classification . . . . .	41
3.3.3	Video description and classifier . . . . .	44
3.4	Experiments . . . . .	45
3.4.1	Datasets and metric . . . . .	45
3.4.2	Implementation details . . . . .	46
3.4.3	Results on TrecVid 2011 . . . . .	48
3.4.4	Results on TrecVid 2013 . . . . .	54
3.4.5	Comparison to the state of the art . . . . .	55
3.5	Summary . . . . .	57

---

### 3.1 Introduction

In this chapter, we describe our webly-supervised approach for action classification [[Chesneau et al., 2017a](#)]. The standard approach to ad-

Birthday party

An individual celebrates a birthday with other people.

A birthday in this context is the anniversary of a person's birth

Less commonly, the term "birthday" can be used to refer to the anniversary of an organization's establishment, but a celebration for an organization does not satisfy the event definition. A birthday celebration is a gathering of people who have been invited by the host or hosts to come to a set location (often a private home, sometimes a restaurant, bar, nightclub, park, or other public venue) to celebrate the birth of the person(s) whose birthday it is (the birthday celebrant(s)). Birthday parties, as with other parties/celebrations, will typically feature an assortment of food and beverages. Birthday parties are often accompanied by colorful decorations, such as balloons and streamers, and some people may wear cone-shaped "birthday hats". The decorations may include signs or banners displaying a message for the birthday celebrant. Often, especially for children's parties, guests will bring gifts and/or gifts wrapped in shiny colorful paper or bags, which will be opened by the birthday celebrant(s), or by the parents/brothers if the birthday celebrants are too young to open the gifts by themselves. A cake (or sometimes cupcakes or other food items) with lit candles, called the "birthday cake," is often served. The song "Happy Birthday to You" may be sung by the guests while the birthday cake with lit candles is carried to a table or counter where the birthday celebrant(s) are seated. The birthday celebrant(s) then blow out the candles, usually after the song is finished, and the guests then clap and cheer. Birthday parties may also involve games or other organized group activities.

Exemplar description: scene: indoors (a home, a restaurant) or outdoors (backyard, park); day or night object(s): people; decorations (balloons, streamers, conical hats, etc.); birthday cake (often with candles); birthday celebrant; guests; gifts activities: singing, blowing out candles on cake.



Figure 3.1 – Given a textual description of a category (left), here *birthday party*, the goal is to rank a set of (test) videos to find the relevant videos of the category (right). Our goal in this work is to learn a classifier without any manually annotated training videos.

dress this classification problem is three-fold: (i) features from videos are extracted, (ii) a classifier is learned for each event with a training dataset of videos, (iii) evaluation on the test set [Laptev and Pérez, 2007, Ke et al., 2005, Laptev et al., 2008, Liu et al., 2009, Wang et al., 2011, Simonyan and Zisserman, 2014b, Karpathy et al., 2014, Xu et al., 2015, Gan et al., 2015]. While methods in this paradigm vary in terms of feature representation, from spatio-temporal or volumetric models [Laptev and Pérez, 2007, Laptev et al., 2008, Liu et al., 2009, Ke et al., 2005] to dense trajectories [Wang et al., 2011], and then to features learned with convolutional neural networks [Simonyan and Zisserman, 2014b, Xu et al., 2015, Karpathy et al., 2014, Gan et al., 2015], they all rely on manually annotated training videos. This makes it difficult to scale them up to data collections with a large number of classes, given the lack of reliable, sufficiently large public video training sets.

In the past few years, several approaches have been proposed to overcome the need for fully-annotated training data for event classification [Niu et al., 2015, Niebles et al., 2008, Ikizler-Cinbis et al., 2009, Duan et al., 2012, Chen et al., 2013, Singh et al., 2015]. Some of these methods build a training set incrementally, by first learning a classifier from an initial dataset and then using it to retrieve additional samples from the web [Ikizler-Cinbis et al., 2009]. An alternative to this strategy is to learn multiple

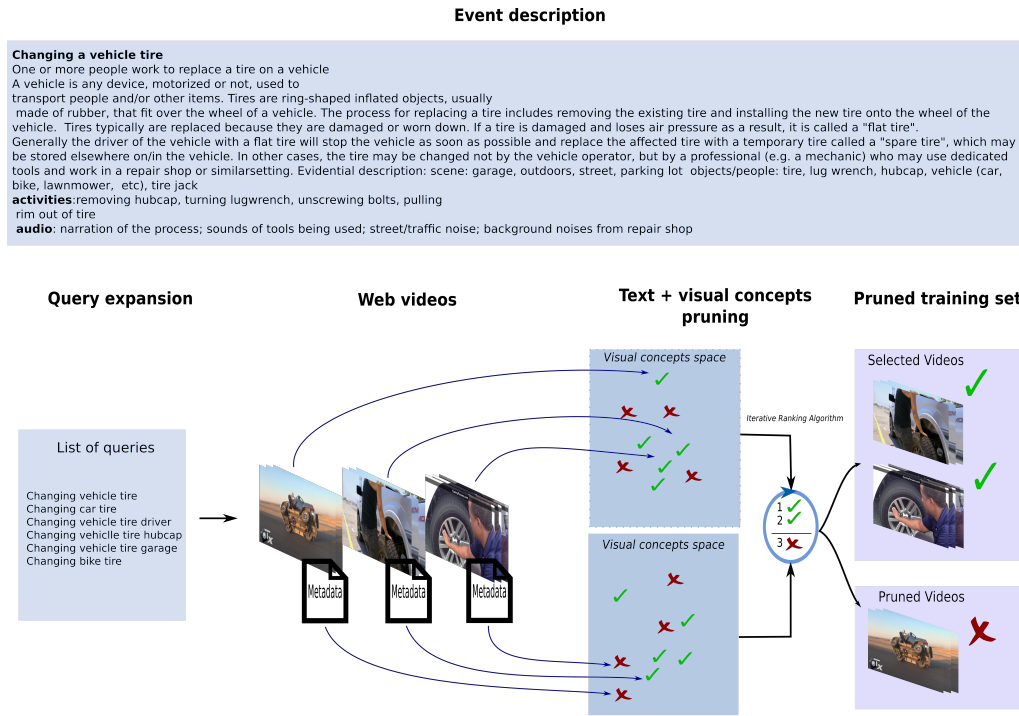


Figure 3.2 – Overview: Given the description of an event (“Event description”), relevant queries are automatically generated (“Query generation”) to collect an initial training set (“Web videos”). Text metadata and visual concepts extracted from these videos are used to select the relevant ones automatically (“Text + visual concepts pruning”), and build a training set for event classification (“Pruned training set”).

classifiers, and combine them with learned weights [Chen et al., 2013] or a multiple instance learning (MIL) framework [Niu et al., 2015]. Although these approaches showed interesting results, they do not exploit the rich cues present in metadata (in the form of text) associated with image or video content on the web [Ikizler-Cinbis et al., 2009, Niu et al., 2015, Duan et al., 2012], or are limited to using only image data [Singh et al., 2015]. This chapter focuses on addressing such limitations of zero-example event classification methods, wherein no manually-curated video training data is available to learn the models, see Figure 3.1.

The core of the proposed approach is the synergy between text and visual content (see Section 3.2 for an overview of text classification). It begins by analyzing the given textual description of each event, which consists of event name (*Birthday party* in Figure 3.1), a one-phrase def-

inition (“An individual celebrates birthday with other people.”) and a short description (“A birthday in this context...”) [Over et al., 2013], to automatically extract a set of queries (see Section 3.3.1). To this end, we use natural language processing techniques to extract keywords relevant to the event, and perform query expansion to further enrich the initial query based on the event name. We then query YouTube to collect an initial training set. This set is automatically pruned, with our novel algorithm, using text and vision-based features to retain only the most relevant video content (see Section 3.3.2). The text features are chosen according to their performance in a text classification task (see Section 3.2). Each selected video is then represented with state-of-the-art convolutional neural network (CNN) features [Krizhevsky et al., 2012, Simonyan and Zisserman, 2014a], together with dense trajectories [Wang et al., 2011], to learn event classifiers (see Section 3.3.3). We analyze the impact of the different steps in our algorithm, namely, query generation, expansion, and pruning on the TrecVid 2011 test set of 31,820 videos, and then compare to state-of-the-art methods [Singh et al., 2015, Jiang et al., 2015a,b] on the TrecVid MED 2013 EK0 dataset [Over et al., 2013], which contains 24,957 test videos. We show that our method achieves the best performance, with a significant improvement of more than 30% mean average precision (mAP) over recent results [Jiang et al., 2015a] on this dataset (see Section 3.4).

## 3.2 An overview of text classification

The goal of this overview is to compare several text pre-processing approaches, features and to find the combination which gives the best result for text classification. The aim of text classification, an extensively researched topic [McCallum and Nigam, 1998, Joachims, 1998, Nigam et al., 2000, Lodhi et al., 2002, Baker and McCallum, 1998], is to assign a label to each document of a corpus. For example, given a set of newspaper articles, we want to automatically classify them as *sport* or *economic* articles. The standard approach consists of three steps. First, a pre-processing step converts each corpus document into a list of words, which are then tagged with their corresponding syntactic categories. Second, features are extracted. Then, a classifier is learned on the training corpus, with techniques such as support vector machine (SVM), by minimizing the classification error.

We evaluate several methods for each of these three steps (see Section 3.2.1). For the preprocessing step, the impact of grammatical anal-

ysis is studied, and two stemming functions are compared: WordNet morphological function [Harabagiu et al., 1999], and Porter stemming algorithm [Porter, 1980]. Several state-of-the art features are combined with pre-processing steps: tf-idf features, bag-of-words features, bigram features, and string kernel (SSK), Word2Vec+FV. These features are previously described in Section 2.1.2, and more details are given in Section 3.2.1. For the learning phase, the use of radial basis function (RBF), linear, polynomial, chi-square kernels is analyzed. We performed these experiments on the Reuters-21578 dataset (Section 3.2.4).

### 3.2.1 Methods

**Pre-processing step.** We present three methods of preprocessing a corpus of raw documents into lists of words: the “POS+WordNet”, “WordNet”, “Porter” methods. As discussed in Section 2.1.1, no consensus emerged from different works [Uysal and Gunal, 2014, Méndez et al., 2005, Pomikálek and Řehůřek, 2007], except that lowercase conversion improves performance for text classification methods. We evaluate the impact of the two main operations during the pre-processing step: the stemming function, which reduces a word to its stem, and grammar tagging, which associates each word with its syntactic form. Results are discussed in Section 3.2.4.

The “POS+WordNet” method starts by building a dictionary by first tokenizing each document into words. The tagging operation is performed by the Stanford part-of-speech (POS) tagger [Toutanova et al., 2003]. Each character is then transformed into lowercase. Words are stemmed with the WordNet morphological function [Harabagiu et al., 1999], using their POS tags. Only nouns, adjectives, adverbs, and verbs are considered. Each distinct stemmed word (4296 in total) is considered as an entry of the dictionary.

The “WordNet” method estimates the influence of grammar analysis performed by Stanford Part-Of-Speech Tagger [Toutanova and Manning, 2000]. The method is the same as the “POS+WordNet” method, except the tagging operation is removed. Words are thus stemmed without the syntactic form as input. If the word is ambiguous (i.e, if the word spelling with no context can be for example either a verb or a noun), it is considered as a noun. The result is a 5470-word dictionary.

The “Porter” method estimates the influence of the stemming function. The stemming step is performed by the Porter stemming algorithm [Porter, 1980], instead of WordNet morphological function in the

two other methods. The result is a 4676-word dictionary.

**Features.** After the pre-processing step, features are extracted from sets of words. In bag-of-words method, the document feature is computed by counting the number of occurrences of each term of the dictionary in the document. For ff-idf (term frequency-inverse document frequency) features, we compute tf and idf for each word in the dictionary and compose them into two vectors. The tf-idf feature vector is the element-wise product of these vectors. This combination of tf and idf vectors diminishes the importance of words that occur frequently in the corpus, as they are not relevant for distinguishing documents, and on the other hand, increases the influence of rare words. Sublinear *term frequency* scaling [Paltoglou and Thelwall, 2010] is applied as follows:

$$\text{tf}_s = \begin{cases} 1 + \log(\text{tf}) & \text{if } \text{tf} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

Adding sublinear *term frequency* dampens the importance of terms that have a high frequency. The intuition behind this is that the difference of information provided by a document which contains one occurrence of a certain word and a document containing two occurrences is more important, or relevant, than the difference between a 1000-occurrence document and a 1001-occurrence document. In contrast to tf-idf features, bigrams are computed by taking into account two consecutive words as a term, in addition to single words. A specific dictionary with two-word entries are built.

For Word2Vec+FV features, each word of the document is embedded into its Word2Vec descriptor. Then, the set of Word2Vec descriptors (one for each word) is aggregated into a Fisher vector using a Gaussian mixture model (GMM). GMM parameters are evaluated from the training corpus, i.e., from the set of words contained in the training samples.

**Learning.** The use of SVM for text classification was introduced by Joachims [1998], in combination with tf-idf features. SVM is a supervised learning model. Given training features  $x_i \in \mathbb{R}^d, i = 1, \dots, l$  and a label  $y \in \mathbb{R}^l$  such that  $y_i \in \{1, -1\}$  (we consider only the case with two labels, i.e., the label represented by 1, and the label represented by  $-1$ ), we want to find the maximum-margin hyperplane that divides the group of points  $x_i$  for which  $y_i = 1$  from the group of points for which  $y_i = -1$ . The optimization problem is defined so that the distance between the hyperplane and the nearest point  $x_i$  from either group is maximized. The distance between a point and the hyperplane is given by its orthogonal projection on



the hyperplan:  $\frac{y_i(w \cdot x_i - b)}{\|w\|}$ . SVM aims at maximizing  $\frac{2}{\|w\|}$  and thus minimizing  $\|w\|$ . Finally, the optimization problem can be formulated as follows:

$$\begin{aligned} & \underset{w}{\text{minimize}} \quad \|w\|, \\ & \text{subject to} \quad y_i(w \cdot x_i - b) \geq 1, \quad i = 1, \dots, n. \end{aligned} \quad (3.2)$$

Non-linear kernels permits to embed features in a higher dimensional space, making the optimization problem linearly separable. For example, RBF kernel embeds the corpus features into an infinite-dimensional Euclidean space. C-SVC formulation [Chang and Lin, 2011] solves the following primal optimization problem.

$$\begin{aligned} & \underset{w, b, \zeta}{\text{minimize}} \quad \frac{1}{2} w^T w + C \sum_{i=1}^l \zeta \beta, \\ & \text{subject to} \quad y_i(w^T \Phi(x_i) + b) \geq 1 - \zeta \beta, \\ & \quad \quad \quad \zeta \beta \geq 0, \quad i = 1 \dots l, \end{aligned} \quad (3.3)$$

and where the kernel is defined by:

$$K(x_1, x_2) \equiv \Phi(x_1)^T \Phi(x_2). \quad (3.4)$$

For the linear case, the kernel is simply a dot product:

$$K(x_1, x_2) = x_1^T x_2. \quad (3.5)$$

The RBF kernel is defined as  $K(x_1, x_2) = \exp^{-\gamma \|x_1 - x_2\|^2}$ . The degree- $d$  polynomial kernel is defined as  $K(x_1, x_2) = (x_1^T x_2 + c)^d$ .

Subsequence string kernel (SSK) leverages the power of feature kernel embedding by building a similarity function based on character subsequence sharing. A subsequence is defined as any ordered sequence of  $n$  characters occurring in the text, though not necessarily contiguous. More formally, string  $u$  is a subsequence of string  $s$ , if there exists indices  $\mathbf{i} = (i_1, \dots, i_{|u|})$ , with  $1 \leq i_1 \leq \dots \leq i_{|u|} \leq |s|$ , such that  $u_j = s_{i_j}$  for  $j = 1, \dots, |u|$ . The subsequence  $u$  of  $s$  is then written as  $u = s[\mathbf{i}]$ . We denote by  $\sum^n$  the set of all finite strings of length  $n$ . The feature mapping  $\phi$  for a string  $s$  is given by defining the  $u$  coordinate  $\phi_u(s)$  for each  $u \in \sum^n$ :

$$\phi_u(s) = \sum_{\mathbf{i}: u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}, \quad (3.6)$$

where  $l(\mathbf{i})$  is the length of the subsequence  $u$  in  $s$ , i.e., by  $l(\mathbf{i}) = i_{|u|} - i_1 + 1$ , and  $\lambda$  a parameter set between 0 and 1. The kernel is an inner product in



Method	SVM Kernel	Accuracy (%)
tf-idf	Linear	98.43
tf-idf	RBF	98.52
tf-idf	poly	98.61
bow	Linear	98.89
bow	RBF	98.71
bow	poly	98.71
tf-idf [Kaestner, 2013]	Linear	98.25
tf-idf [Kaestner, 2013]	RBF	97.02
if-idf [Kaestner, 2013]	poly	97.78

Table 3.1 – Comparison with Kaestner [2013]

the feature space generated by all subsequences of length  $n$  :

$$K_n(s, t) = \sum_{u \in \Sigma^n} \langle \phi_u(s), \phi_u(t) \rangle = \sum_{u \in \Sigma^n} \sum_{i: u=s[i]} \sum_{j: u=t[j]} \lambda^{l(i)+l(j)} \quad (3.7)$$

Similarity of two documents  $d_k$  and  $d_l$  can be estimated using (3.7). If the two documents contain the same string of size  $n$ , then this string will contribute the most to the similarity estimation. This kernel is used in the SVM formulation in (3.3) during the training phase.

### 3.2.2 Dataset

We perform experiments on Reuters-21578 in order to evaluate and compare our three pre-processing steps combined with the most popular text features. The Reuters dataset contains articles from Reuters news agency. We use Reuters-21578, the newer version of the corpus [Lewis, 1997]. Reuters-21578 is one of the most commonly used datasets for text classification during the last two decades. It was first compiled by David Lewis in 1987 and is publicly available.<sup>1</sup> We used the Modified Apte ("ModeApte") split. It consists of 9603 training documents, 3299 test documents, and 8676 unused documents. The training set was reduced to 7775 documents as a result of pruning the unlabelled ones, i.e., those with no TOPICS category. There are 135 topics in this category, with 118 of them occurring at least once in the training set and test documents. Of the 118 potential categories, only the most frequent 10 are used, while

1. <http://www.research.att.com/lewis>

keeping all documents as negative samples. The top-10 categories are: *earn, acq, money-fx, grain, crude, trade, interest, ship, wheat, corn*.

In order to compare with [Kaestner, 2013], we use a second split described in [Kaestner, 2013]. The set of documents that are assigned exclusively to the two most frequent classes of the database are selected. The resulting dataset contains 6,215 documents, assigned to *earn* (2,840 training documents and 1,083 test documents with 398,706 words) and to *acq* (1,596 training documents and 696 test documents with 334,554 words).

### 3.2.3 Implementation details

**Preprocessing - Dictionary creation.** For the creation of the three dictionaries (one for each pre-processing method), words composed of three or fewer characters are ignored. Terms which appear less than ten times in the corpus and stop words, i.e., words which belong to a predefined list are removed. For example, the verb “be” in the list of root words is useless, because it appears too frequently in every document, and thus does not provide any information about the document. Words which appear in more than 90% of the documents in the total corpus are also removed. We use the nltk toolkit [Loper and Bird, 2002] for WordNet morphology function and Porter algorithm. The grammatical tagging is computed with the Stanford part-of-speech tagger.<sup>2</sup>

**Features extraction and learning.** We use the implementation of scikit-learn [Pedregosa et al., 2011] to compute the bag-of-words, tf-idf, bigram features. Dictionaries are given by the preprocessing step. The features are also individually L2-normalized.

SSK kernels are computed with the SHOGUN implementation [Sonnenburg et al., 2010], with standard parameters  $n = 5$  and  $\lambda = 0.5$ . For Word2Vec+FV features, we first compute the Word2Vec descriptors for every word of the corpus, using the implementation described in [Mikolov et al., 2013b]. The model is trained on a subset of the Google News dataset (about 100 billion words), and contains 300-dimensional vectors for three million words and sentences. The dimensionality of the descriptors is reduced to 250 by a principal component analysis (PCA). A GMM of 32 Gaussians is computed from the set of reduced descriptors. LibSVM [Chang and Lin, 2011] is used during the learning step. Several kernel are tested: linear, RBF, polynomial, chi-square. The regularization parameter (see (3.2)) and the positive-negative balance weights are cross-

2. <https://nlp.stanford.edu/software/tagger.shtml>

Method	SVM Kernel	Kernel parameters	Mean
tf-idf	Linear	–	86.9
tf-idf	RBF	$\gamma = 1$	87.21
tf-idf <a href="#">Joachims [1998]</a>	RBF	$\gamma = 1$	86.30
tf-idf	poly	$degree = 2$	86.37
tf-idf <a href="#">Joachims [1998]</a>	poly	$degree = 2$	85.10

Table 3.2 – Comparison with [Joachims \[1998\]](#). The means refers to the average precision-recall breakeven point.

validated.

### 3.2.4 Results

In order to validate our implementation, we compare the results of our pipeline with two reference papers [[Joachims, 1998](#), [Kaestner, 2013](#)]. In [[Kaestner, 2013](#)], accuracy is the evaluation metric, i.e., the percentage of documents correctly classified, while [[Joachims, 1998](#)] follows the traditional measure performance in text classification: mean precision-recall breakeven point, i.e., the measure where precision and recall are equal in the precision-recall curve.

Table 3.1 shows a comparison with [Kaestner \[2013\]](#). Results similar to those reported by [Kaestner \[2013\]](#) validate our pipeline. We also remark that the performance is very high (around 98%). Moreover, tf-idf and bag-of-words features give similar results, as do linear, RBF, and polynomial kernels. It leads us to the conclusion that accuracy and the proposed split is not challenging enough on Reuters-21578.

Table 3.2 shows a comparison [[Joachims, 1998](#)]. We select kernel parameters reported in [Joachims \[1998\]](#) with the highest breakeven point for RBF and polynomial kernels. Our implementation is comparable to [Joachims \[1998\]](#). Moreover, learning SVM classifiers with RBF kernels gives better results than using linear or polynomial kernels.

Table 3.3 shows the influence of the three preprocessing methods, described in Section 3.2.1, on the Reuters-21578 dataset. Two features are used: tf-idf, and bag-of-words features. For both features, the preprocessing method which uses WordNet morphological function and no grammar analysis produces the best results. Using grammatical analysis reduces the performance for text classification, aside from being time consuming. This reduction in performance shows that a good exploita-

Name	Stemming function	Grammar analysis	Features	mean breakeven
"POS+WordNet"	WordNet	Yes	tf-idf	86.9
"POS+WordNet"	WordNet	Yes	bow	86.07
"WordNet"	WordNet	No	tf-idf	88.07
"WordNet"	WordNet	No	bow	87.71
"Porter"	Porter	No	tf-idf	87.18
"Porter"	Porter	No	bow	86.79

Table 3.3 – Analysis of three preprocessing methods. We compare the Porter algorithm function and the Wordnet morphological functions. We also compare the impact of grammatical analysis. Two features are tested : bag-of-words (bow), and tf-idf features. Results are shown for a linear SVM

tion of text statistics, by considering text as a set of stem words, is the key factor for text classification. Higher-level representation, like a grammar representation, does not carry additional useful information for text classification.

The table also shows that WordNet morphological function performs better than Porter stemming algorithm. Wordnet morphological function takes advantage of the WordNet taxonomy to stem words, whereas Porter algorithm does not use additional dictionary. Based on these experiments, the following pre-processing steps are recommended for text classification: tokenization, lowercase conversion, no grammar analysis, stemming with WordNet morphological function, removing stop words and words occurring less than ten times in the corpus.

Table 3.4 compares the performance of different text features on the Reuters-21578 dataset. Tf-idf features produce better results than bag-of-words features, Word2Vec+FV, and SSK kernel. Adding two consecutive words as entries of dictionary for tf-idf features does not provide any gain (see single words and bigrams features in Table 3.4). Tf-idf features outperform Word2Vec+FV by 2.50%, showing that Word2Vec descriptors are less-adapted for text classification. This is likely due to the average number of words in a Reuters-21578 document being 200. A set of 200 descriptors is not large enough to leverage the effectiveness of the Fisher vector model. Tf-idf features outperforms SSK kernel by 7.76%, showing that considering text as a set of stem words is preferable for text classification than a raw sequence of characters. Words carry semantics information. As for the Word2Vec+FV features, the average length of document may have an impact on the SSK kernel performance, but this study is out of the scope of our work. Tf-idf features produce bet-

method	kernel	kernel parameters	mean
bow	Linear	–	86.07
bow	RBF	$\gamma = 1$	86.07
bow	poly	$degree = 2$	85.70
tf-idf	linear	–	86.90
tf-idf	RBF	$\gamma = 1$	87.21
tf-idf	poly	$degree = 2, c = 0$	86.37
bigram	linear	–	77.30
bigram	RBF	$\gamma = 1$	75.86
single word + bigram	linear	–	87.34
single word + bigram	RBF	$\gamma = 1$	86.95
Word2Vec+Fisher	RBF	$\gamma = 1$	84.71
SSK kernel	–	–	79.45

Table 3.4 – Comparison of bag-of-words, tf-idf, bigram, Word2Vec+FV features, and SSK kernel.

ter results than the bag-of-words, showing the importance of idf (inverse document frequency) vector, which highlights infrequent words being more discriminative for text classification than common words. Table 3.4 also shows that RBF kernel performs better than polynomial and linear kernels for tf-idf and bag-of-words features. Appendix A.1 show per-class results for all the tested features.

Appendix A.2 shows additional results on Reuters-21578, by varying the  $\gamma$  parameter for RBF kernel combined with tf-idf, bigram, and bag-of words features. The parameter range is set from 0.05 to 5. The best score is obtained for tf-idf features with a gamma parameter set to 0.6, leading to a mean breakeven point of 87.68. For bigram features, the best score is obtained with gamma parameter set to 0.2, resulting in a mean breakeven point of 87.64. Performance with tf-idf features is more stable compared to bigram features, when varying the gamma parameter. For example, with a gamma parameter set to 5, the mean breakeven point is 83.17 for tf-idf features, and 75.27 for bigram features. For Word2Vec feature, several PCA reductions and number K of Gaussians have been tested (see Appendix A.3 for more details and results).

### 3.2.5 Summary

The following pipeline achieves state-of-the-art results for text classification:

- a fast-to-compute preprocessing step: tokenization, lowercase conversion, WordNet morphological function stemming, removing stop words and those occurring less than ten times in the corpus.
- computing tf-idf features.
- using RBF kernels for the learning phase.

In the rest of the chapter, we will apply this pipeline to improve ranking of videos using metadata (for example Youtube metadata: title, tags, description), which is then used for an action recognition. These text features will be combined with visual videos features in Section 3.3.2.

## 3.3 Learning from Web Videos

Given a textual description of the approach, we propose a webly-supervised approach which generates queries (Section 3.3.1) to collect web data, and then prune this collection to account for noisy data (Section 3.3.2). Then, we represent each video with state-of-the-art convolutional neural network (CNN) features [Simonyan and Zisserman, 2014a] and dense trajectories [Wang et al., 2011], and learn event classifiers (Section 3.3.3).

### 3.3.1 Textual query generation

A straightforward way to generate queries is to use the title of the event, e.g., *birthday party*, *changing vehicle tire*. While this is a good starting point to retrieve relevant data from the web, it falls short on using the rich content in the description of the event effectively. For example, events such as *birthday party* involve various objects like cake, candle, and can be organized as a barbecue, tea party, ball, etc. In the case of the event *changing vehicle tire*, changing a car tire is very different to changing bicycle tire. Such content is typically available in the event description. In order to better exploit this rich text information, we design a query expansion strategy.

We start with the event description available in TrecVid data, which contains a definition—a short phrase of 5 to 10 words, and a short paragraph describing the event and the context in which the event occurs (see

the *Changing vehicle tire* example in Figure 3.2). A similar description can also be provided by a user when defining the list of events. The description presents scenes and objects potentially involved, activities that are likely to occur, and in some cases a textual description of audio in the event. We create queries specific to each event from all of this text.

The event title forms the reference query in our approach. It is used to automatically produce additional queries as described in the following. We first tag each word in the event description (i.e., title, and all the associated text) with the Stanford part of speech tagger (Stanford POSTagger) [Toutanova and Manning, 2000]. This assigns every word one of the 31 standard tags, such as noun-singular, adjective, adverb. Note that, although results in Section 3.2.4 suggest that grammatical tagging degrades performance in text classification, the task here is different, and we show in the experiments in Section 3.4.3 that such a tagging is necessary for our query expansion. We then compute a similarity between words in the event title and those in the remaining (longer) event description. For example, in the case of the event *changing vehicle tire*, we compute the similarity  $s$  between every word in the title  $tw$  (i.e., changing, vehicle, tire) and all the words  $w$  of the same tag type in the text description, individually. In this example, “changing” is compared to other verbs in the event description. The similarity  $s$  is given by:

$$s(tw, w) = \frac{1}{2}(\mathbf{x}_w \cdot \mathbf{x}_{tw} + \frac{1}{N} \sum_{\substack{i=1 \\ tw_i \neq tw}}^N \mathbf{x}_w \cdot \mathbf{x}_{tw_i}), \quad (3.8)$$

where  $\mathbf{x}_w$  and  $\mathbf{x}_{tw}$  are feature representations of words  $w$  and  $tw$  representations,  $N$  is the total number of words in the event title, with  $tw_i$  denoting the  $i$ th word from it. The feature representation is computed with Word2Vec Mikolov et al. [2013b], which is an embedding of a word into a vector space. More details of this feature computation are presented in Section 3.4.2.

Dot product between two feature vectors is a measure of the similarity between the two corresponding words, as used in Mikolov et al. [2013b]. The first term in (3.8) measures the similarity between a word in the event title,  $tw$ , and one of the words from the event description,  $w$ . The second term measures the similarity between  $w$  and the remaining words in the event title, denoted by  $tw_i$ . For the *changing vehicle tire* class, this would mean a high similarity to “tire” as well as “changing”, when finding words similar to “vehicle”. We then retain all the words of the same type with high similarity. To sum up, we use verbs, nouns



and adjectives as reference words in the event title, and generate a set of event-related words for each tag-type. In the case of *changing vehicle tire*, the set of relevant verbs associated with “changing” contains “replacing”, and the set of nouns related to “vehicle” contains “car”, “bike”. These sets of words are then used for query expansion based on the following three strategies determined by the structure of the event title.

(i) **Event title contains verb and object.** If a word from the relevant set is a hyponym of the reference word Miller [1995], a new query is created by substituting the reference word with the related word. Otherwise, the word is added to the original query. For example, since words like “car”, “bike” are hyponyms of vehicle, the query “changing vehicle tire” becomes “changing bike tire” and “changing car tire”. With “hubcap”, the new query is “changing vehicle tire + hubcap”. The intuition behind this expansion strategy is to adapt queries to intra-class variation.

(ii) **Event title without verb.** We first generate additional queries with the strategy described above, for the two other tag types, i.e., noun and adjective. Since action events are better described with verbs, we propose an additional strategy to compensate for the lack of verbs in the event title. We use the similarity measure (3.8) to find verbs related to words of other tag-types in the title. Each of these related verbs is individually added to the title to produce new queries. Consider the *birthday party* class, which has no verb in the event name, as an example. An automatically extracted verb related to this event, “celebrate”, is added to the event title to generate a new query “birthday party celebrate”.

(iii) **Event title with a single word.** We handle events such as *parade*, *parkour*, which contain a single word in the event title, separately. In order to avoid drifting from the original semantic meaning, which is likely to occur when we replace the only title word with those related to it, we propose adding each one to the title to generate new queries, instead of replacing. In the case of the reference query “parkour”, it is related semantically to “gymnastics”. However, replacing it with “gymnastics” as the new query results in a large number of generic videos of gymnastics, and not all of them belong to the *parkour* class. We avoid this with new queries targeted to events, i.e., “parkour gymnastics” in this case.

At the end of the query expansion step, we have a rich set of event-related queries (see examples in Table 3.5). These automatically generated queries allow for the creation of a new event dataset with web resources. In this work, we download videos and their corresponding



Table 3.5 – Query expansion results for six of the TrecVid13 categories.

Changing vehicle tire	Parkour	Cleaning an appliance
Changing vehicle tire		
Changing car tire	Parkour	Cleaning an appliance
Changing vehicle tire driver	Parkour parkour	Cleaning an appliance household
Changing vehicle tire wheel	Parkour skateboarding	Cleaning an appliance cooker
Changing vehicle tire lawnmower	Parkour snowboarding	wash an appliance
Changing vehicle tire hubcap	Parkour gymnastic	Cleaning an dishwasher
Changing vehicle tire garage	Parkour acrobatic	Cleaning an dryer
Changing vehicle tire person	Parkour outdoor	Cleaning an toaster
Changing bike tire	Parkour maneuver	Cleaning an refrigerator
replace vehicle tire		
Dog show	Rock climbing	Town hall meeting
Dog show		
Dog show show		Town hall meeting
Dog show kennel	Rock climbing	Town hall meeting village
Dog show leash	Rock climbing climb	Town hall meeting auditorium
Dog show obedience	Rock climbing climber	Town hall meeting community
Dog show breed	Rock climb	Town hall meeting discuss
Dog show handler	Rock climbing jump	Town hall meeting event
Dog show Frisbee	Rock climbing wall	Town hall meeting vote
chihuahua show		Town hall meeting discussion
sheepdog show		Town hall meeting attend

metadata from YouTube. Implementation details of our query generation method are presented in Section 3.4.2.

### 3.3.2 Pruning with text and visual classification

Given the variety of data available online, we observed that more than half of the videos are irrelevant to an event. For example, videos in which a person is talking about *changing a car tire* without actually doing it, or videos displaying a *parkour* in the video game *Minecraft*, or videos of red carpet arrivals for a celebrity *birthday party*, are available through YouTube, but are not relevant to learn the action event. In Figure 1.3, the highest ranking videos for the request “parade” are shown. Only a few of them can be used as positive samples for an action recognition dataset. To prune such irrelevant videos, we use text and visual concept features jointly. We begin by describing the representation of text and visual data with tf-idf and visual concept features respectively, and then present our pruning algorithm.

**Tf-idf features.** We represent text data with tf-idf (term frequency-inverse

document frequency) features [Hiemstra \[2000\]](#), which have achieved excellent performance for text classification, as detailed in Section 3.2. These features capture the importance of a word to a document in the corpus. In our case, the corpus is the set of YouTube text metadata of all the downloaded videos, the text associated with each video is the document, and the set of words occurring in the corpus is the dictionary. We follow the method described in Section 3.2.1.

**Visual concept features.** We compute visual features from video data to complement text features extracted from metadata. This helps leverage the visual similarities among videos depicting the same event. For example, in videos of the event *changing vehicle tire*, a tire is visible in at least a part of the video. To this end, we use state-of-the-art convolutional neural networks [[Simonyan and Zisserman, 2014a](#), [Krizhevsky et al., 2012](#)]. In particular, we use the VGG-16 network [[Simonyan and Zisserman, 2014a](#)] trained on ImageNet with 1000 classes. The last layer of this network (fc8) is a soft-max score indicating the presence of a class. We refer to these classes as visual concepts [[Binder et al., 2014](#)] as they encode semantic content based on the appearance in input images. A visual concept can be an object, a place, an animal, or a texture. We compute the visual concept scores for each image independently, and aggregate the number of activations, i.e., the number of non-zero probabilities, of each concept over the entire video.

**Pruning algorithm.** Given the text and visual representations of videos collected with our queries, we present a two-step approach to prune them. In the first step, we perform pruning with text data to select an initial set of relevant examples. As demonstrated in our experimental analysis, this step removes some of the irrelevant videos. To leverage the complementary cues in visual data, we perform a second pruning step with visual concept features.

The first step uses the given textual description of events, as it is the primary source of information about an event. For each event, all the videos downloaded are ranked with tf-idf features, by comparing them to the tf-idf of the event description. Specifically, we compute the dot product of event description and metadata tf-idf features. The top-ranked examples with this are videos which metadata is most similar to the event description, in terms of word statistics. These videos are considered as representative examples for the event. We take the top-ranked videos chosen with a threshold on the dot product matching score, determined empirically, to evaluate this text-only pruning step (see “txt prun.”

Table 3.6 – Influence of our pruning approaches on the TrecVid11 test set. We show the result using the initial training set (“no prun.”), and the training set pruned with text (“txt prun.”), text and visual features (“txt+vis. prun.”). The variant “Ref. query” is one where only videos obtained with the reference query are used to train the event classifiers. The percentage in the last column indicates the proportion of selected videos per class after the pruning

Category	Ref. query	Query expansion		
		no prun.	txt prun.	txt+vis. prun.
E006: <i>birthday party</i>	8.93	16.65	17.50	19.87 (57%)
E007: <i>changing vehicle tire</i>	48.60	54.67	51.45	55.58 (73%)
E008: <i>flash mob gathering</i>	21.62	29.27	38.29	42.36 (61%)
E009: <i>unstuck vehicle</i>	42.00	35.13	33.23	40.89 (39%)
E010: <i>grooming an animal</i>	13.35	11.21	14.19	15.08 (75%)
E011: <i>making a sandwich</i>	10.24	12.29	10.55	18.04 (66%)
E012: <i>parade</i>	15.03	38.13	36.81	45.37 (45%)
E013: <i>parkour</i>	22.93	28.77	37.83	30.67 (43%)
E014: <i>repairing an appliance</i>	23.09	30.87	32.00	36.57 (75%)
E015: <i>sewing project</i>	21.65	22.87	24.12	22.91 (84%)
mean	22.74	27.99	29.60	32.73 (60%)

in Table 3.6). Note that this threshold is independent of the event type, and its impact on the performance is analyzed in Section 3.4.

To also prune with visual data, we compute the mean visual concept feature vector of the top-20 videos ranked with text. In other words, we take the most reliable videos in terms of their textual description, and extract a representation of the occurrence of visual concepts in the event. We then measure the relevance of any video to an event as the dot product of this mean vector and the video’s visual concept feature. We re-rank all the video examples according to this similarity measure, and re-compute the mean vector with the new top-20 videos. This step is repeated several times, until the set of top-20 videos does not change, typically less than 50 iterations. This visual data pruning allows us to retrieve videos similar in visual content even if they are lacking in meta-data, for example, when it is not available for a video. On the other hand, it also prunes videos with good text description, but no relevant video content, e.g., a person talking about how to change a car tire, without

Table 3.7 – Evaluating dense trajectory (“dense traj.”) and CNN (“VGG-16”) features, and their combination (“Combined”) as a video representation on the 10 event classes (E00x) from the TrecVid 2011 test set. See text in Section 3.4.3 for details.

Category	dense traj.	VGG-16	Combined
E006: <i>birthday party</i>	14.51	19.07	19.87
E007: <i>changing vehicle tire</i>	31.80	49.62	55.58
E008: <i>flash mob gathering</i>	39.29	27.09	42.36
E009: <i>unstuck vehicle</i>	23.25	37.48	40.89
E010: <i>grooming an animal</i>	6.22	8.96	15.08
E011: <i>making a sandwich</i>	7.74	13.18	18.04
E012: <i>parade</i>	37.38	35.51	45.37
E013: <i>parkour</i>	24.87	21.14	30.67
E014: <i>repairing an appliance</i>	22.04	26.64	36.57
E015: <i>sewing project</i>	15.78	13.70	22.91
mean	22.29	25.23	32.73

actually demonstrating how it is done in the video clip.

### 3.3.3 Video description and classifier

At the end of the pruning stage, we have an automatically refined set of videos to learn the event classifiers. We use a combination of CNN [Simonian and Zisserman, 2014a] and dense trajectory features to represent these videos. Figure 3.3 shows an overview of the video description. The CNN features are computed per frame and mean-pooled over time. For the dense trajectory features, we compute HOG, HOF, MBH descriptors along each trajectory, and aggregate them into a Fisher vector, as described in Wang et al. [2011]. These two features are complementary—while CNN features describe appearance at the frame level, dense trajectories extract motion information in the video. For example, in a *parade* video, dense trajectories capture the movement of people walking in one direction, while CNN extracts visual attributes like people, flags, costumes, etc. Using these two features together, by concatenating them into a single vector, rather than separately, leads to a significant gain in performance. We demonstrate this empirically in Table 3.7 (see Section 3.4.3 for details). Given the set of automatically selected positive training video samples, we learn a one-vs-rest linear SVM for each class, with all videos

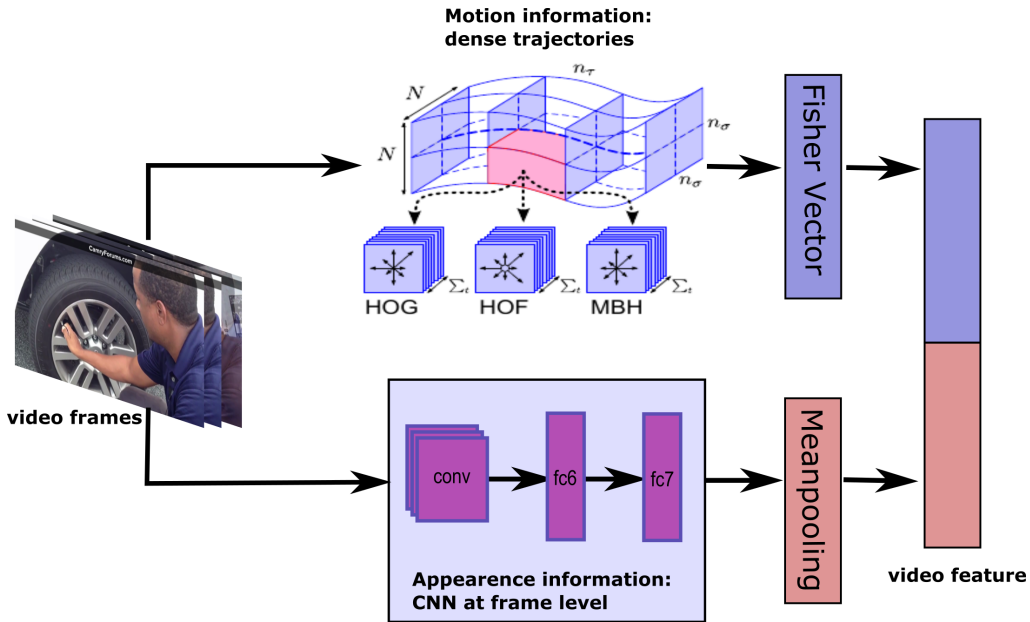


Figure 3.3 – Overview of our feature extraction technique. Dense trajectory and CNN features are combined to produce a spatio-temporal representation.

from the other classes as negative samples.

## 3.4 Experiments

### 3.4.1 Datasets and metric

We used videos from the TrecVid multimedia event detection task to evaluate our approach. In particular, we used test videos from the 2011 challenge to analyze the variants of our method: different video feature representations, using only the reference queries, and the complete approach with query expansion. We take the best variant from these (i.e., full method with query expansion) and evaluate it on the 2013 challenge videos. Note that none of the TrecVid training videos were used in our experiments, except for the analysis “Adding TrecVid11 training data to the pruned set” in Section 3.4.3. We follow the standard TrecVid evaluation protocol and report mean average precision (mAP).

The TrecVid 2011 (TrecVid11) dataset contains videos of 10 events: *birthday party, changing vehicle tire, flash mob gathering, unstuck vehicle,*

*grooming an animal, making a sandwich, parade, parkour, repairing an appliance, sewing project*, along with videos unrelated to any of these classes, i.e., *background* category. These classes are referred with labels E006 for *birthday party*, E007 for *changing vehicle tire*, ..., and E015 for *sewing project*. The test set contains 1244 videos of the 10 events, and 30,576 for the background class.

TrecVid 2013 (TrecVid13) contains 10 additional events: *attempting bike trick, cleaning an appliance, dog show, giving directions to location, marriage proposal, renovating home, rock climbing, town hall meeting, winning race without vehicle, working on metal crafts*. These classes are assigned labels E021 (for *attempting bike trick*) through E030 (for *working on metal crafts*). The test set has a total of 24,957 videos, among which 23,468 belong to the background class. We follow the EK0 challenge protocol for this dataset, where no training videos are available to learn the event classifiers.

For all of our experiments, we use the average precision (AP) metrics. Average precision is a measure that combines recall and precision for ranked retrieval results. It is defined as the mean of the precision scores after each relevant video is retrieved.

### 3.4.2 Implementation details

**Query generation and expansion.** During the query generation process (Section 3.3.1), a word from the event description is selected to create a new query if its similarity with a word from the event title, according to (3.8), is greater than 0.35. Table 3.5 shows a few examples of queries generated. For computational reasons, we limit the maximum number of queries to 10, including the reference query. If the query generation step produces more than 10 queries, we pick the top ones most similar to the event title, according to the measure (3.8). We download 150 videos for the reference query, and 50 each for other queries from YouTube, along with their corresponding metadata. This forms our initial training set.

For TV11 events, 3626 videos were downloaded with our query expansion. The pruning algorithm selects 2172 videos from this set. Note that the pruning step only changes the number of positive samples for each class, and all the videos downloaded from the other classes, pruned or not, form the negative exemplar set.

**Text data.** We use metadata fields “tags” and “description” available with YouTube videos to form the text component of our training data. We build a dictionary of words from all these text metadata files, by applying standard text processing techniques, such as stemming [Loper and Bird,



2002] and removing stop words. This results in a 8652-word dictionary for TrecVid 2011, and one with 8000 words for TrecVid 2013. We compute a 8652-dimensional (8000-dim for 2013) tf-idf representation for each text metadata document (as described in Section 3.3.2). Idf is computed over all categories jointly. We also apply sublinear tf scaling [Töpper et al., 2012], which replaces  $tf$  by  $tf_s$  (3.1). This feature is then L2-normalized after multiplying tf and idf. We use the same dictionary to compute the tf-idf representation of each TrecVid event description.

**Video data.** One of the issues with downloading videos from YouTube for events like *birthday party* is the presence of animated slideshows. Such videos are a collection of non-contiguous photos or title slides (e.g., “Happy Birthday!”) and lack any movement depicting an action. To remove these slideshows from our dataset, we measure the similarity between two consecutive frames with the L2-norm of the difference of their appearance features. If this norm is less than 0.15, the frames are considered as similar. If two consecutive frames  $i$  and  $i + 1$  are similar, and frames  $i + 1$  and  $i + 2$  are also similar, then the three frames  $i, i + 1, i + 2$  are grouped into a same set of similar frames. With this method, if a video contains  $N$  slides, its frames are split into  $N$  sets of similar frames.

**Visual concept and video features.** We use the VGG-16 network to compute these features. It is composed of 13 convolution (with ReLU), 5 max pooling and 3 fully-connected layers, and an additional soft-max layer. All the convolutional filters are of size  $3 \times 3$ . The network is trained with ImageNet on 1000 categories to extract our visual concept as well as video features as follows. We compute VGG-16 responses for one in every ten frames independently, normalize them with their L2-norm, and pool them temporally to get a video representation. Specifically, we use the 1000-dimensional output of the fc8 layer for the visual concept features. These correspond to ImageNet categories, which are objects (*lifeboat, convertible, bassoon*), places (*restaurant, home theater*), animals (*gazelle, sea lion*), or textures (*velvet*). For extracting video features to learn the classifiers (as described in Section 3.3.3), we use richer features from the fc6 layer.

In the “txt+vis. prun.” method, the textual ranking from “txt. prun.” method is used as initialization. For a frame  $i$ , we compute the VGG16-fc8 feature,  $fc8_i$ . For computational reasons, this feature is extracted every ten frames. We then build a matrix  $M_{fc8} \in \mathbf{R}^{1000 \times N}$ , where the  $i$ -th column is the  $fc8_i$  feature vector, and  $N$  is the number of frames. We then aggregate these frame-level features temporally to obtain the vi-

sual representation, with a L0-norm pooling operation in the temporal dimension. L0-norm pooling measures the number of times a concept is activated in the video, and performs better than L1-norm pooling as it captures finer details, whereas L1-norm is more sensitive to large activations. For example, for the *changing vehicle tire* event, the “tire” concept gives large values all over the video, reducing the importance of other concepts if L1-pooling is used. This produces a 1000-dimensional vector which describes the video in the visual concept space. This feature is then L2-normalized.

We reduce the dimension of the HOG, HOF, and MBH components of dense trajectory features by a factor of 2 with principal component analysis. We then perform power and L2 normalization, and use 256 Gaussian components for the mixture model.

**Classifier.** The SVM for video classification is implemented with LIBSVM [Chang and Lin, 2011]. Regularization and class imbalance parameters are set with 10-fold cross-validation. During cross-validation, the training set is split randomly into a validation set, and a smaller training set containing 75% of the training data, whilst maintaining the original ratio of positive and negative samples. Mean average precision on the validation set is computed for each choice of parameters, and the one with the best performance is chosen.

### 3.4.3 Results on TrecVid 2011

Our overall method, where classifiers are learned with CNN and dense trajectory features computed on the (two-step) pruned video set, achieves 32.73 mAP, as shown in Table 3.6. We use TrecVid 2011 as a test-bed to analyze several variants of our approach.

**Importance of query expansion.** Table 3.6 shows a comparison of variants based on the queries used to constitute the training data. We report results for query expansion (shown as in “Query expansion” in the table), where we download additional videos with our textual query expansion. We observe that query expansion significantly improves over the baseline reference query result (shown as “Ref. query” in the table), by over 5% in mAP score on average. For E012 (*parade*), query expansion improves AP from 15.03 to 38.13, due to the creation of accurate additional queries (e.g., “parade procession”, “parade march”, “parade commemoration”). Query expansion can affect the performance negatively in a few cases. For E009 (*unstuck vehicle*), we see a reduction from 42.00 to 35.13,



Table 3.8 – The 10 most relevant visual concepts for four events. These correspond to the concepts with the top-10 values in the mean visual concept vector, computed at the end of the pruning algorithm. See Section 3.3.2 for details.

<i>Changing vehicle tire</i>	<i>Unstuck vehicle</i>
1 - car wheel	1 - jeep, landrover
2 - minivan	2 - pickup, pickup truck
3 - car mirror	3 - tow truck, tow car, wrecker
4 - limousine, limo	4 - minivan
5 - crash helmet	5 - snowplow, snowplough
6 - disk brake, disc brake	6 - minibus
7 - vacuum, vacuum cleaner	7 - ambulance
8 - minibus	8 - golfcart, golf cart
9 - seat belt, seatbelt	9 - beach wagon, station wagon, wagon
10 - motor scooter, scooter	10 - car wheel
<i>Repairing an appliance</i>	<i>Grooming an animal</i>
1 - switch, electric switch, electrical switch	1 - hair spray
2 - washer, automatic washer, washing machine	2 - hand blower, hair dryer, hair drier
3 - refrigerator, icebox	3 - fur coat
4 - safe	4 - wig
5 - microwave, microwave oven	5 - Afghan hound, Afghan
6 - cash machine, cash dispenser	6 - swab, swob, mop
7 - dishwasher, dish washer,	7 - iron, smoothing iron
8 - loudspeaker, speaker, speaker unit	8 - cash machine, cash dispenser,
9 - iPod	9 - dishwasher, dish washer
10 - soap dispenser	10 - vacuum, vacuum cleaner

as expansion adds noisy videos due to the addition of generic verbs, e.g., “have”, “do”, to the initial query. This behavior is more of an exception than a rule. Pruning methods do compensate for most of this loss, with the final result of 40.89 for this event.

**Importance of pruning algorithm.** The method “txt prun.” in Table 3.6 is the variant where only text features are used to prune the initial set of videos created with our query expansion (see Section 3.3.2 for details). This further improves the mAP over the “no prun.” variant by 1.61%. The overall method, “txt+vis. prun.”, which uses text and visual features for pruning gives an additional gain of 3% in mAP score over text-only pruning. We performed an additional experiment by manually annotating videos for two classes: E006, E008. We built a positive set for the two classes by watching all the videos, and annotating relevant ones as positive samples. The results for this ground truth annotation are 22.13, 46.23 respectively. Our result of 19.87 and 42.36 for these classes, using

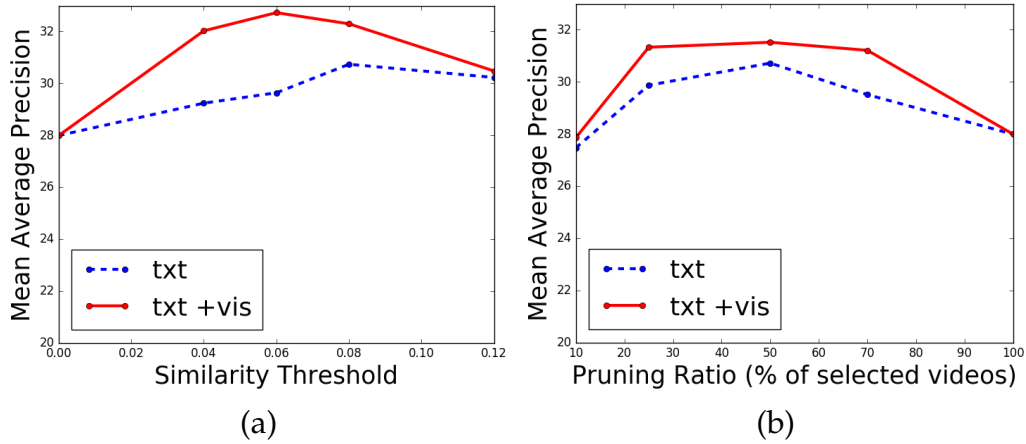


Figure 3.4 – Impact of (a) the similarity threshold and (b) the pruning ratio are shown as mean average precision on the TrecVid11 test set.

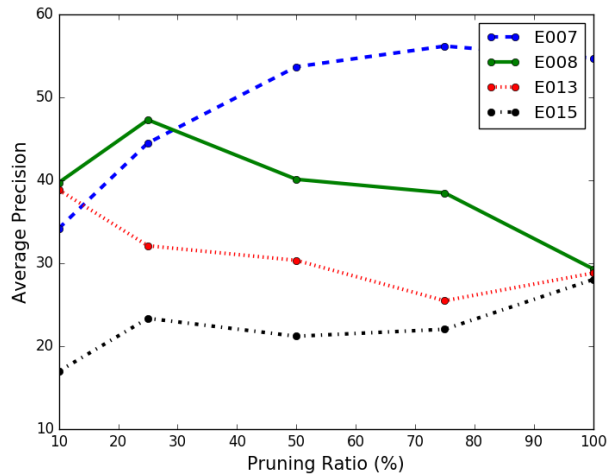


Figure 3.5 – Average precision for four classes with the “txt+vis. prun.” method using pruning ratio on the TrecVid11 test set. (Best viewed in pdf.)

no annotated data, is close to this upper bound, further highlighting its effectiveness.

We also analyze the 10 most relevant visual concepts for the four sample events in Table 3.8. They correspond to the concepts with the most important activations in the mean visual concept vector, which is com-

puted over the top-20 ranked videos at the end of our pruning algorithm. We observe that concepts are semantically related to the event. For example, “car wheel” is the most important concept for the event *Changing vehicle tire*, and the top-10 relevant concepts for the event *repairing an appliance* include appliances such as washer, refrigerator. While the events *changing vehicle tire* and *unstuck vehicle* can be visually similar (see examples in Figure 1.1), the relevant concepts help us distinguish them, e.g., “snowplow” is relevant only for *unstuck vehicle*.

**Combining appearance and motion features.** We analyze the performance of three visual feature representations: dense trajectories, VGG-16 fc6 responses, and a combination of the two features, in Table 3.7. The variant with VGG-16 features performs better than dense trajectories on average. Dense trajectories, however, show a better performance for four events, *flash mob gathering*, *parade*, *parkour*, *sewing project*, where motion plays an important role in representing them. Combining the two features, by stacking them into a single vector, outperforms using either of the two features individually by a large margin—an improvement of over 7% on average. The classifiers learn the relative importance of these two features automatically from the training data. Significant improvement is observed for all the TrecVid11 events due to the two representations being complementary. For example, for a *changing vehicle tire* video, dense trajectories capture the movement of a person manipulating the car jack, while VGG-16 extracts visual attributes like tire, garage, car. For a *parkour* video, dense trajectories capture the movement of a person performing activities, while VGG-16 extracts visual attributes like body shape, streets, parks. Thus, we use the combined feature vector in our full method.

**Impact of the similarity threshold.** The similarity threshold determines the number of videos pruned with tf-idf (see the details of pruning in Section 3.3.2). We analyze its impact by varying the threshold on TrecVid11 in Figure 3.4(a). We observe the best performance on this dataset for a threshold value of 0.06. We also followed an alternate strategy of using a pruning ratio, i.e., keeping  $r\%$  of the downloaded videos, to determine the refined set used for training. This is shown in Figure 3.4(b). Note that pruning ratio 100 and the similarity threshold 0 are equivalent to the “no prun.” method. On average, using a similarity threshold performs better than using a pruning ratio. In the latter method, the number of positive samples depends on the total number of downloaded videos, but many videos can be relevant for some events (see E007, E015 in Figure 3.5), and

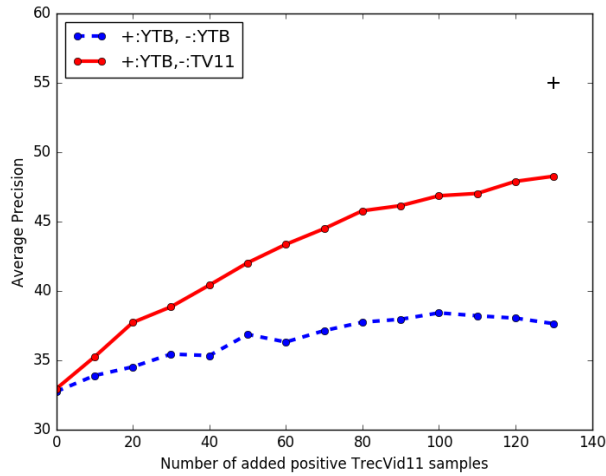


Figure 3.6 – Impact of adding TrecVid11 positive and negative samples. “+:YTB, -:YTB” is the pruned YouTube dataset to which TrecVid samples are added. “+:YTB, -:TV11” is the variant with pruned YouTube samples as positive and background TrecVid11 videos as negative samples, with progressive addition of TrecVid11 positives. The average precision of a method trained on the entire TrecVid11 training set is shown with a black cross.

irrelevant for other events (see E008, E013 in Figure 3.5). Using an absolute threshold to determine the pruned set, as in the similarity threshold strategy, maintains a uniform quality for this pruning approach over all the events. We observe that “txt+vis. prun.” outperforms “txt. prun.” in both (a) and (b) in Figure 3.4.

**Adding TrecVid11 training data to the pruned set.** We study the impact of introducing TrecVid11 training dataset into our (pruned) training set through two experiments, presented in Figure 3.6. First, we progressively add positive samples from TrecVid11 training dataset to our training set composed of pruned videos. This results in an improvement from 32.73 (the overall result in Table 3.6) to 38.39 shown in the dashed-line (blue) curve in the figure, for 100 positive TrecVid samples added per event. This is partly due to bias in the TrecVid11 dataset, wherein videos in the training and test sets are very similar. For example, videos in the training and test set<sup>3</sup> for the *making a sandwich* event show the same per-

3. Videos named HVC494077, HVC516890, HVC606263, HVC788253 from TrecVid11 training set and videos HVC220966, HVC358422, HVC122406, HVC648049,



Figure 3.7 – Sample frames from TrecVid11 videos labeled as “making sandwich”. Samples from training set (the two left columns) and from the test set (the two right columns) are shown.

son in the same kitchen, and from the same viewpoint, see Figure 3.7.

In the second experiment, we study the impact of using 9600 videos corresponding to “background” from the TrecVid11 training set. We follow the standard TrecVid protocol of using these videos as negative examples. We add positive samples from the TrecVid11 training set to a dataset composed of our positive pruned YouTube samples and TrecVid11 background events as negative exemplars. When no positive TrecVid11 samples are added, we obtain an mAP of 32.92 (the lowest point on the red solid-line curve in Figure 3.6), which is very similar to the result with our pruned dataset (32.73). Adding positives from TrecVid11 in this case, we observe that the improvement is more significant than in the first case (blue dashed-line in the figure). This is potentially due to domain adaptation issues, where TrecVid11 positive samples are visually more similar to the negative ones than our YouTube positive exemplars. Videos in the YouTube set are longer, have a higher resolution and more professional content (e.g., tutorial for changing vehicle tire or making a sandwich),

---

HVC218518 from the test set.

Table 3.9 – Per-category results for zero-shot learning on TrecVid MED2013 EK0 test set.

Category	Singh et al. [2015]	Gan et al. [2016a]	Query expansion		
			no prun.	txt prun.	txt+vis. prun.
E006	14.48	15.148	18.12	20.70	25.86
E007	41.37	39.60	60.78	57.94	66.39
E008	45.56	19.30	42.44	47.66	51.22
E009	53.71	36.80	47.66	49.15	59.12
E010	6.38	8.60	13.99	13.27	20.41
E011	11.56	15.10	16.15	19.79	16.91
E012	17.76	32.20	54.39	53.30	55.16
E013	7.86	12.90	45.84	57.03	58.33
E014	15.18	16.10	35.88	40.68	47.68
E015	3.23	29.20	24.19	31.70	31.79
E021	6.76	6.60	9.81	6.94	6.94
E022	3.11	2.10	16.07	14.76	19.91
E023	0.91	40.50	38.46	37.32	38.29
E024	0.55	1.60	10.28	10.06	13.15
E025	0.21	1.30	8.57	16.54	10.96
E026	3.63	3.90	5.02	5.18	2.86
E027	1.44	13.20	15.19	15.84	13.88
E028	0.95	10.50	22.79	29.86	35.96
E029	0.10	13.70	0.20	0.61	0.40
E030	0.82	2.90	0.45	1.25	0.60
mean	11.81	16.10	24.29	26.44	28.79

than the TrecVid11 examples. For comparison, we also show the performance (55.01 mAP) when learning the classifiers with the manually-annotated TrecVid11 training set (black cross in the plot). The difference between this and our final result of adding all TrecVid11 positives to the YouTube set (48.24 on the red solid-line curve in the figure) is also due to the domain adaptation problem between TrecVid and YouTube videos.

### 3.4.4 Results on TrecVid 2013

Having used TrecVid11 as a test-bed to evaluate all the variants of our method, we choose the best-performing method on it (i.e., the method using query expansion, combination of appearance and motion features, and a similarity threshold of 0.06) as our full method. Table 3.9 shows the results of our full method with text and visual feature pruning (“txt+vis. prun.” in the table). The method “txt prun.” shows an improvement of 2.20% in mAP score over “no prun.”. The full method “txt+vis. prun.”

Table 3.10 – Comparison to the state of the art for zero-shot learning on TrecVid MED 2013 EK0 test set. Video representations used by each method (denoted by “✓”), appearance features (“appearance”), motion features (“motion”), and speech or text features (“speech/text”), are also shown.

Method	appearance	motion	speech/text	mAP
Chen et al. [2014a]	✓	–	–	2.30
Jiang et al. [2014]	✓	–	✓	2.50
Wu et al. [2014]	✓	✓	–	6.12
Habibian et al. [2014]	–	✓	–	6.39
Singh et al. [2015]	✓	–	–	11.81
Jiang et al. [2015b]	✓	✓	✓	20.80
Jiang et al. [2015a]	✓	✓	✓	22.12
Gan et al. [2016a]	✓	✓	–	16.10
Ye et al. [2015]	✓	✓	–	8.86
Gan et al. [2016b]	✓	✓	–	16.70
Hussein et al. [2017]	✓	–	–	17.86
Ours: Query expansion: no prun.	✓	✓	–	24.29
Ours: Query expansion: txt prun.	✓	✓	–	26.44
Ours: Query expansion: txt+vis. prun.	✓	✓	–	28.79

gives an additional gain of 2.3% in mAP, over text-only pruning. We also report results from Singh et al. [2015], which is the only recent method providing per-class results on TrecVid13. We added per-class results provided by the authors of a very recent paper [Gan et al., 2016a]. Our method outperforms Gan et al. [2016a], Singh et al. [2015] by a large margin. For example, we obtain an average precision of 47.68 (16.10 for Gan et al. [2016a], 15.18 for Singh et al. [2015]) for the *repairing an appliance* event. Figure 3.8 shows a few qualitative results of our approach on this dataset. All the displayed videos contain visual concepts related to the event: tire, and car jack for *changing vehicle tire*, cake and candles for *birthday party*, an appliance for *repairing an appliance*, a group of people wearing the same uniform for *parade*. It shows the importance of visual concepts for pruning.

### 3.4.5 Comparison to the state of the art

We compare with several approaches [Jiang et al., 2015a, 2014, Chen et al., 2014a, Singh et al., 2015, Jiang et al., 2015b, Gan et al., 2016a, Wu et al., 2014, Habibian et al., 2014] on the TrecVid 2013 EK0 challenge







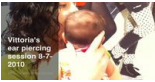

























Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6
<i>Birthday party</i>					
					
✓	✓	✓	✓	✗	✓
<i>Changing vehicle tire</i>					
					
✓	✓	✓	✓	✓	✓
<i>Grooming an animal</i>					
					
✓	✗	✓	✗	✗	✗
<i>Parade</i>					
					
✓	✓	✓	✓	✓	✓
<i>Repairing an appliance</i>					
					
✓	✓	✓	✓	✓	✓

Figure 3.8 – The top-6 videos for the categories: *Birthday Party*, *Changing vehicle tire*, *Grooming an animal*, *Parade*, *Parkour*, *Repairing an appliance*, *Dog Show*, *Town Hall Meeting*, from the TrecVid 2013 test set, obtained with our method using automatically-curated training data.



dataset. Here, we focused on methods which build a training set from internet data. As shown in Table 3.10, our results have a mean average precision of 28.79%, which is a significant gain of 6.5% over the state-of-the-art approach [Jiang et al., 2015a], which also curates a training set from the internet. We also compare with other recent methods: 20.80% [Jiang et al., 2015b], 16.7 [Gan et al., 2016b], 16.1 [Gan et al., 2016a], 11.89% [Singh et al., 2015], 8.86 [Ye et al., 2015], 6.39% [Habibian et al., 2014], 6.12% [Wu et al., 2014], 2.5% [Jiang et al., 2014], 2.3% [Chen et al., 2014a]. We outperform all these methods significantly, due to the following key differences. We use motion information effectively, in contrast to Wu et al. [2014], Chen et al. [2014a], Jiang et al. [2014], Singh et al. [2015], Gan et al. [2016a], Ye et al. [2015], relying on image data alone. Our query expansion method exploits critical elements of actions, e.g., related verbs, whereas Wu et al. [2014], Chen et al. [2014a], Jiang et al. [2014], Habibian et al. [2014], Singh et al. [2015], Jiang et al. [2015a], Ye et al. [2015], Gan et al. [2016b,a], Jiang et al. [2015b] are limited to queries which are not as rich.

### 3.5 Summary

This work introduces a novel approach for event classification, given only a textual description of the event. Our approach relies on textual query expansion specifically designed for actions, which allows us to collect significantly more videos than using only the event name. A pruning step creates a reliable training dataset of videos sharing semantic and visual content. We show state-of-the-art results on TrecVid MED 2011 and 2013 in the zero-shot learning framework.



## **Part II**

# **ACTION LOCALIZATION IN REAL-WORLD VIDEOS : A PERSON-CENTRIC APPROACH**

In this part, we describe a new person-centric framework for action localization that tracks people in videos and extracts full-body human tubes.

# Chapter 4

## Related work

### Contents

---

4.1	Object detection . . . . .	60
4.1.1	Faster R-CNN: a RPN-based detector . . . . .	61
4.1.2	Part detectors . . . . .	62
4.2	Amodal completion . . . . .	62
4.3	Action localization . . . . .	64
4.3.1	Earlier approaches . . . . .	64
4.3.2	Action tubes . . . . .	66
4.4	Metric . . . . .	69

---

This chapter presents the related work for action localization. In Section 4.1, we start by presenting the work on object detection, which is often used in action localization techniques, specifically in our approach. In Section 4.2, the amodal completion principle, which is the basis for our person-centric approach, is introduced, along with an overview of related work on neuroscience and computer vision. In Section 4.3, we review the related work on action localization. Finally, Section 4.4 presents the metric used for action localization.

### 4.1 Object detection

Object detection is one of the most important and challenging tasks in computer vision. Given an image, the aim is to localize and identify objects of interest, e.g., a table or a person. Object detection combines both classification and localization techniques. It often requires a large pool of training images with bounding box annotations. Many approaches have been proposed to improve detection performance. Some

of the most recent and popular object detectors are: R-CNN by Girshick et al. [2014], OverFeat by Sermanet et al. [2014], MultiBox by Erhan et al. [2014], SPP by He et al. [2015], DeepBox by Kuo et al. [2015], MR-CNN by Gidaris and Komodakis [2015], AttentionNet by Yoo et al. [2015], DenseBox by Huang et al. [2015], Fast R-CNN by Girshick [2015], DeepID Net by Ouyang et al. [2015], Faster R-CNN by Ren et al. [2015], NoC by Ren et al. [2016], YOLO by Redmon et al. [2016], R-FCN by Li et al. [2016], SSD by Liu et al. [2016], and YOLOv2 by Redmon and Farhadi [2016]. As we will see in the next chapter, our person-centric approach is based on a part detector, whose architecture comprises a Region Proposal Network (RPN) adapted for parts. Section 4.1.1 presents the most advanced object detector using an RPN: Faster-RCNN. In Section 4.1.2, we review related work on part detectors.

#### 4.1.1 Faster R-CNN: a RPN-based detector

In Ren et al. [2015], an object detector is built by introducing an RPN that shares full-image convolutional features with the detection network proposed by Girshick [2015], thus enabling region proposals with few additional computations. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. It is trained end-to-end to generate region proposals, which are used by Fast R-CNN [Girshick, 2015] for detection. To do so, at each location of an image determined by a sliding-window, a set of anchor boxes is applied at different scales to generate a proposal. The proposal is labeled as positive if it has high overlap with the groundtruth box, or it is set as a negative proposal. A random sampling over the set of anchors at different scales and location is performed to create a batch that is used as training data for a loss function :

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \quad (4.1)$$

where  $p_i$  is the estimated probability of the anchor indexed by  $i$  being an object. The groundtruth label  $p_i^*$  is 0 if the anchor is negative, and is 1 if it is positive.  $t_i$  is a vector representing the four parameterized coordinates of the predicted bounding box, and  $t_i^*$  is that of the groundtruth box associated with a positive anchor. The classification loss  $L_{cls}$  is a log loss function. For the regression loss,  $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ , where  $R$  is the smooth  $L1$ -loss function defined in [Girshick, 2015]. The outputs of RPN are  $p_i$  and  $t_i$ . The two terms are normalized by the size of the mini-

batch,  $N_{cls}$ , and the number of anchors location,  $N_{reg}$ . A joint optimization with the R-CNN enables the detector to generate relevant proposals and to classify them accordingly. Our work generalizes the Faster-RCNN approach by proposing an RPN which generates proposals centered on object parts and infers the full object location from each part.

### 4.1.2 Part detectors

The deformable part-based model [Felzenszwalb et al., 2010] was the most effective object detector from 2009 until the publication of R-CNN [Girshick et al., 2014]. Mixtures of multi-scale deformable models use a sliding window approach. A latent SVM is learned on partially labeled data in order to detect objects. The model is able to represent a high variety of classes, achieving good detection performance. Other works leverage part detection to detect the whole object in an image [Modolo and Ferrari, 2017, Chen et al., 2014b, Endres et al., 2013]. Endres et al. [2013] use a model which trains a large number of part detectors with a single positive exemplar. Part detectors are refined by selecting a subset of diverse and discriminative examples. In Chen et al. [2014b], the holistic object and its parts are detected separately, and a connected template rearranges all the previous detections. In Modolo and Ferrari [2017], semantic part-based models are learned from a set of web images. Part appearances and spatial arrangement are used to detect the whole object. All these works are focused on detecting visible parts of the object, i.e., object detectors do not try to infer the location of the full object in cases of occlusion or frame truncation, as done in our approach. However, the human brain naturally infers what the eyes can not see. For instance, if we see a person sitting behind a table, a classic detector will estimate the upper body as a person, whereas the human brain will infer a seated person occluded by the table. This type of modality is called amodal completion.

## 4.2 Amodal completion

In cognitive theory, amodal completion is the perception of a whole object when only parts of it affect the sensory receptors. For example, a chair will be perceived as a complete volumetric structure even if only the backrest activates retina cells (or camera sensors). Objects are perceived as possessing internal volume and hidden rear surfaces despite the fact that only the near surfaces are exposed to view. Similarly, the world is

perceived as a surrounding plenum, even though only part of it is in view at any time. Another example of amodal perception is a dog behind a picket fence. The dog is perceived as a single continuous object even if the fence partially occludes it. In [Bregman \[1990\]](#), an auditory analogue is mentioned: a melody interrupted by bursts of white noise is heard as a single melody continuing “behind” the bursts of noise.

Amodal completion has been studied for several decades by the neuroscience community. [Kanizsa \[1976\]](#) shows that contours which appear to us as clearly visible, and which are drawn by a combination of incomplete figures, are supplied by the visual system. [Murray et al. \[2004\]](#) compare modal completion (i.e., the perceptual filling-in, in which the completed regions of the image are perceptually similar to the rest of the image) and amodal completion mechanisms. A common initial mechanism is observed for both types of completion, followed by differential mechanisms. [Rauschenberger and Yantis \[2001\]](#) show that visual search has access to pre-completion representations, i.e., information before amodal completion has occurred, but only for a limited time that depends on the size of the occluded region. [Gerbino and Salmaso \[1987\]](#) show that a complete target is matched faster with an amodally completed figure than with a truncated one.

Very few works have been proposed by the computer vision community for leveraging amodal completion to solve computer vision tasks. [Kanizsa and Chambolle \[1997\]](#) claim that continuation of object boundaries plays a central role in the amodal completion process. Based on this work, [Masnou and Morel \[1998\]](#) propose a variational process to “complete” the object, based on principles described by [Kanizsa and Chambolle \[1997\]](#). In [Kar et al. \[2015\]](#), the task of amodal bounding box completion aims to infer the full extent of the object instances in the image. A probabilistic framework for learning category-specific object size distributions from available annotations is built, which permits to infer the true sizes of objects in novel images. Finally, a focal length prediction approach exploits scene recognition to overcome inherent scale ambiguities. Their quantitative results show that inferring the full location of occluded or out of frame objects helps to better understand the scenes. In our work, we explore amodal completion for human body, infers its location even outside the frames, and uses amodal information for action localization. Our person-centric approach localizes the full body through videos and defines a better reference frame for feature extraction.

## 4.3 Action localization

Action localization consists of recognizing actions as well as estimating where the action occurs in video frames. In Section 4.3.1, we review the earlier approaches for action localization, and more recent works based on action tube generation are presented in Section 4.3.2.

### 4.3.1 Earlier approaches

Initial attempts for temporal and spatio-temporal action localization are based on a sliding-window scheme and handcrafted descriptors [Laptev and Pérez, 2007, Cao et al., 2010, Yuan et al., 2009, Gaidon et al., 2013, Tian et al., 2013]. In Laptev and Pérez [2007], a cascade of weak action classifiers are learned, leveraging a combination of space-time features (motion and shape features), keyframes priming, and space-time interest point descriptors. In Cao et al. [2010], action detection and a model adaptation are combined into a maximum a posterior (MAP) estimation framework. By exploring the spatio-temporal coherence of actions, prior information can be obtained without supervision. A video sequence is embedded into a collection of spatial-temporal interest points (STIPs), where each STIP is represented by a feature vector. To model the probability of each STIP, a Gaussian mixture model (GMM) is used to represent the background distribution. Tian et al. [2013] extend a deformable part-based method to video by replacing the HOG filters with their 3D version, i.e., the HOG-3D descriptor (see Section 2.2.1) In Yuan et al. [2009], an action is also represented as a space-time object and is characterized by a collection of spatio-temporal interest points. A discriminative pattern matching called naive Bayes based mutual information maximization (NBMIM) is proposed for multi-class action categorization. Based on the naive Bayes assumption and by assuming the independence among the STIPs, we can evaluate the mutual information between a video clip  $V$  and a specific class  $c \in \mathcal{C}$  as:

$$MI(\mathcal{C} = c, V) = \sum_{d_q \in V} \frac{P(d_q | \mathcal{C} = c, )}{P(d_q)}, \quad (4.2)$$

where  $d_q$  is the representation feature of a STIP in a video sequence. In addition to NBMIM, a search algorithm is proposed to locate the optimal subvolume in the 3D video space for action detection. In Gaidon et al. [2013], a model based on a sequence of atomic action units, called “actoms”, is proposed. Actoms are built to be semantically meaningful and



characteristic for the action. Actom sequence model (ASM) represents an action as a sequence of histograms of actom-based visual features, which can be seen as a temporally structured extension of the bag-of-features. The idea behind ASM is that actions can be broken down into a sequence of micro-actions called actoms, and thus can be detected and localized more efficiently.

Other approaches, such as Lan et al. [2011], Kläser et al. [2010], rely on figure-centric models, wherein the location of the person performing the action is detected in some form. In Lan et al. [2011], the location of a person is treated as a latent variable when inferring the action performed. Inspired by latent SVM [Felzenszwalb et al., 2010], a scoring function measures the compatibility between a video  $I$ , an action label  $y$ , the bounding boxes  $L$  that localize the person performing the action in each frame of the video:  $f_\theta(L, y, I) = \max_{\mathbf{z}}(\theta^T \Phi(\mathbf{z}, L, y, \mathbf{I}))$ , where  $\theta$  are the model parameters, the matrix  $\mathbf{z}$  specifies the discriminant regions inside each bounding box, and  $\Phi(\mathbf{z}, L, y, \mathbf{I})$  is a feature vector defined on  $\mathbf{z}, L, y, \mathbf{I}$ . We denote  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  the chain-structured undirected graph represents the configurations of bounding boxes  $L$  in a video. The model parameters have three parts  $\theta = \alpha, \beta, \gamma$ , and  $\theta^T \Phi(\mathbf{z}, L, y, \mathbf{I})$  is defined as:

$$\begin{aligned} \theta^T \Phi(\mathbf{z}, L, y, I) = & \sum_{i \in \mathcal{V}} \alpha^T \phi(l_i, z_i, y, I_i) \\ & + \sum_{i, i+1 \in \mathcal{E}} \beta^T \psi(l_i, l_{i+1}, z_i, z_{i+1}, I_i, I_{i+1}) + \gamma^T \eta(y, I) \end{aligned} \quad (4.3)$$

where the unary potential  $\alpha^T \phi(l_i, z_i, y, I_i)$  measures, for the  $i$ -th frame  $I_i$ , the compatibility between the action label  $y$ , the configuration of the bounding box  $l_i$ , and the discriminative cells of the bounding box  $z_i$ . The pairwise potential  $\beta^T \psi(l_i, l_{i+1}, z_i, z_{i+1}, I_i, I_{i+1})$  measures the compatibility between two neighboring frames and assesses how likely they are to contain the same person. The global action potential  $\gamma^T \eta(y, I)$  is a global template model measuring the compatibility between the action label  $y$  and a global feature vector of the whole video. In Kläser et al. [2010], the upper body is explicitly detected and tracked. Actions are then represented by a descriptor computed on spatio-temporal window. The descriptor, called HOG-Track descriptor, extends the HOG image descriptor to spatio-temporal volumes, and goes beyond a rigid spatio-temporal cuboid, as it adjusts piecewise to the spatial extent of the track previously computed. An SVM with an RBF kernel is learned on the HOG-track features. Our approach is also based on human detections but is signif-

icantly more robust to large variations in pose and appearance, due to a part detection approach.

### 4.3.2 Action tubes

More recently, methods based on action proposals [Jain et al., 2014, van Gemert et al., 2015, Yu and Yuan, 2015, Oneata et al., 2014, Marian Puscas et al., 2015] have been employed to reduce the search complexity and improve the quality of tracks, referred to as “tubes”. In Jain et al. [2014], a sampling strategy produces sequences of bounding boxes, one for each frame, called tubelets. The action localization pipeline is split into four steps : 1) a super-voxel segmentation step, producing  $n$  super-voxels, to which we associate  $n$  tubelets, obtained as sequences of bounding boxes that encompass the super-voxel, 2) iterative generation of tubelets, where two super-voxels can be merged into a single one, resulting in a new tubelet, 3) descriptor computation where a tubelet is embedded into a BOW representation, 4) classification. In Oneata et al. [2014], action proposals are created by merging hierarchical supervoxels. A superpixel graph is constructed with spatial neighbours connections, second-order spatial connections (in order to be robust to small occlusions), temporal neighborhood connections. A hierarchical clustering is performed over the superpixel graph to segment the video both spatially and temporally. Supervoxels are agglomerated, and supervoxel similarities are learnt using color, flow, size, fill, spatial size, spatial fill, temporal size and temporal overlap features. In van Gemert et al. [2015], a localized SLINK algorithm (see [Sibson, 1973] for more details) is proposed to combine the benefits of tubelets [Jain et al., 2014] and prim3D [Oneata et al., 2014]. The similarities that seed the proposal generation are based on improved dense trajectory features, which are also used during the classification step. In Yu and Yuan [2015], a set of action proposals  $\mathbf{P}$  is extracted by maximizing the coverage of actionness scores in the video:  $\max_{\mathbf{P} \subset S} \sum_{b_t \in \mathbf{P}_i} w(b_t)$ , where  $S$  is a collection of proposal candidates that satisfies some constraints of consistency,  $\mathbf{p}_i$  is a proposal included into the subset  $\mathbf{P}$ ,  $b_t$  a bounding box in frame  $t$  of the video. The actionness of a bounding box  $w(b)$  is based on two parts: human detection score  $H(b)$  and motion score  $M(b)$ :  $w(b) = H(b) + \lambda M(b)$ .

In Marian Puscas et al. [2015], an unsupervised method for foreground object extraction in video is initialized by a selective search approach. Then, a tracking and refinement step is done using optical flow with a transductive learning approach. This paradigm has produced promising results, but these methods generate thousands of proposals even for a

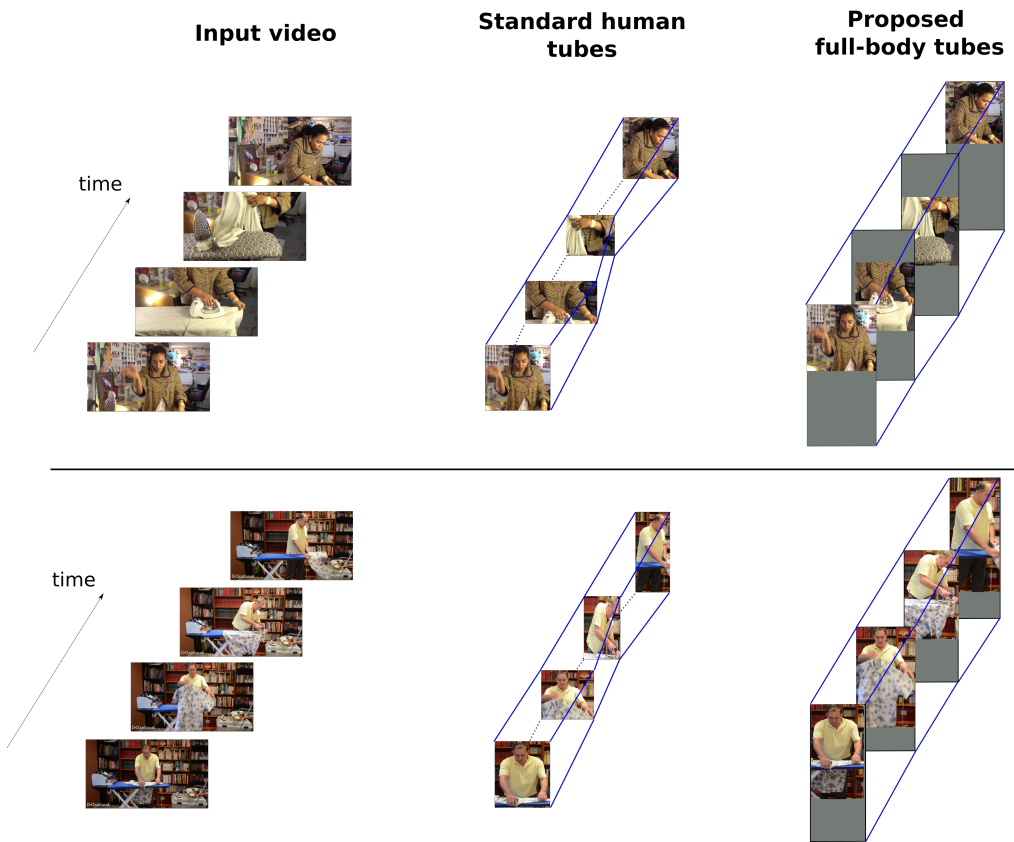


Figure 4.1 – Two example videos from the DALY dataset to illustrate the difference between our human tube extraction and previous methods (a state-of-the-art method [Weinzaepfel et al., 2016] is used for illustration here). Note that our tubes (shown on the right) take the occluded parts as well as parts beyond the image boundaries into account.

short video sequence, and are not scalable to large video datasets. Moreover, they do not take hidden parts and occlusions into account, and are very sensitive to viewpoint changes. In contrast, our method computes one tube for each person in the sequence, taking into account body parts that are occluded or truncated by image boundaries, thereby addressing the problem of amodal completion [Kar et al., 2015] in the context of action localization.

Recent work has leveraged the success of deep learning for human action localization [Gkioxari and Malik, 2015, Weinzaepfel et al., 2015, Peng and Schmid, 2016, Saha et al., 2016], by using state-of-the-art object detectors, like region proposal-based convolutional neural networks [Ren

et al., 2015]. Region-CNNs (R-CNNs) are trained with both appearance and motion cues in these methods to classify region proposals in individual frames. Human tubes are then obtained by combining class-specific detections with either temporal linking based on proximity [Gkioxari and Malik, 2015], or with a tracking-by-detection approach [Weinzaepfel et al., 2015]. State-of-the-art methods [Peng and Schmid, 2016, Saha et al., 2016] rely on an improved version of R-CNN, e.g., Faster R-CNN [Ren et al., 2015], trained on appearance and flow. In Peng and Schmid [2016], a two-stream Region-CNNs is built with two RPNs: an appearance region proposal network similar to RPN in Ren et al. [2015] and a motion region proposal network which generates flow proposals. The two sets of proposals are then merged. A multi-region scheme is added in the faster R-CNN model, which provides complementary information on body parts. The detected regions include full-body, half upper-body, half bottom-body part, and a larger region covering the full body and its neighborhood. In Saha et al. [2016], A two-stream RPN approach - one on RGB and another on optical-flow images - is proposed. For each stream, the RPN takes as input a video frame, generates a set of region proposals with action scores. Next, a Fast R-CNN [Girshick, 2015] detection network takes as input the original video frame and a subset of the region proposals generated by the RPN, and outputs a detection box and a classification score for each input proposal, indicating the probability of an action class being present within the box. The two scores - one for RGB stream, one for the optical flow stream - are merged, improving action detection accuracy. After fusing the set of detections over the entire video, sequences are linked to create a single action tube.

All these methods make extensive use of bounding box annotations in every frame for training the network. Although this scheme is accurate for short videos, it is not scalable to long videos with viewpoint changes and close-ups, such as the examples shown in Figure 4.1. Our method automatically determines the best part to track and infers the global localization of the person from the part alone. The merging step of this inference for each part refines the final bounding box proposed for the frame.

Recent work in Weinzaepfel et al. [2016] is also related to our approach. It extracts human tubes with a Faster R-CNN based detector. These tubes are then used to localize and detect actions by combining dense trajectories, appearance and motion CNN classifiers. However, this method is also limited to tubes which frame only visible parts of people, and as a result loses spatial correspondences between frames, thus impacting feature extraction. In contrast, our method tracks the entire

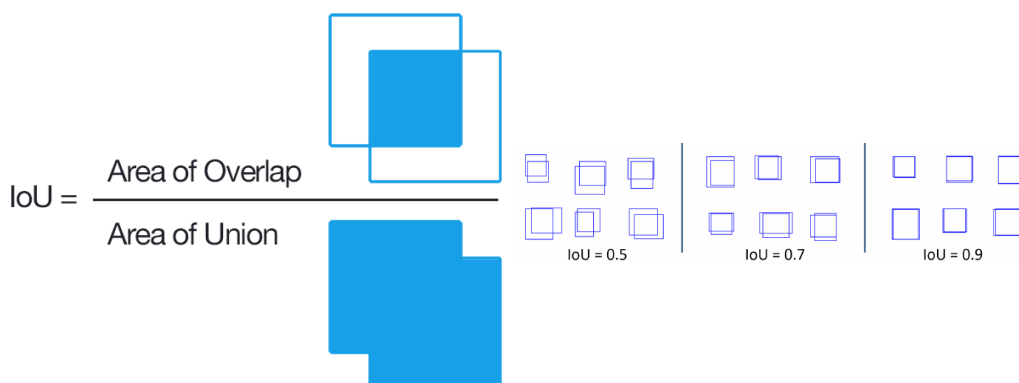


Figure 4.2 – Illustration of the metric “Intersection Over Union”. Figure courtesy: <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

person during the full sequence, making it robust to partial occlusions, and preserves spatial information for feature extraction and action classification. We establish new state-of-the-art results on the challenging DALY dataset proposed in [Weinzaepfel et al. \[2016\]](#), which consists of 31 hours of YouTube videos, with spatial and temporal annotations for 10 everyday human actions.

## 4.4 Metric

The Intersection over Union measure (IoU) is commonly used to evaluate the accuracy of an object detector [[Everingham et al., 2010](#)]. IoU between two boxes is the area of the overlap of the two boxes divided by the area of their union. A predicted bounding box is considered as correct if its IoU with a groundtruth box is higher than a threshold. The higher the threshold is, the more challenging and accurate is the detection task. Usually the threshold is set to 0.5. An illustration of the metric is shown in [Figure 4.2](#).

The standard metric consists in computing the mean Average-Precision (mAP) at a given threshold  $t$ . A detection is correct if the IoU with the groundtruth is over a threshold  $t$ . As in object detection, duplicate detections are considered as wrong. Precision is then computed for each class, with a threshold  $t = 50\%$  or a more challenging threshold  $t = 70\%$ .

For tube comparison, we use the recall of human tube extractions. The Recall curve shows the proportion of tubes correctly detected with

respect to the IoU threshold  $t$ . The area under curve (AUC) indicates the proportion of area “covered” under the curve.

# Chapter 5

## Detecting parts for action localization

### Contents

---

5.1	Introduction . . . . .	72
5.2	Method: From Parts to Tubes . . . . .	73
5.2.1	Detecting Parts . . . . .	74
5.2.2	Building full-body tubes . . . . .	76
5.2.3	Action localization . . . . .	79
5.3	Experiments . . . . .	80
5.3.1	Datasets . . . . .	80
5.3.2	Implementation details . . . . .	82
5.3.3	Results . . . . .	86
5.4	Summary . . . . .	95

---

In this chapter, we describe a new framework for action localization that tracks people in videos and extracts full-body human tubes, i.e., spatio-temporal regions localizing actions, even in the case of occlusions or truncations. This is achieved by training a novel human part detector that scores visible parts while regressing full-body bounding boxes. The core of our method is a convolutional neural network which learns part proposals specific to certain body parts. These are then combined to detect people robustly in each frame. Our tracking algorithm connects the image detections temporally to extract full-body human tubes. We apply our new tube extraction method to the problem of human action localization, on the popular JHMDB dataset, and on a very recent challenging dataset DALY (Daily Action Localization in YouTube), showing state-of-the-art results.



## 5.1 Introduction

Action localization in videos comprises recognizing the action as well as locating where and when it takes place in the sequence. A popular method for achieving this is to track the person(s) of interest during the sequence, extract image features in the resulting “human tube” i.e., the sequence of bounding boxes framing a person, and recognize the action occurring inside the tube. Such a method performs well when people are fully visible, and when correspondences can be established between tubes extracted from different videos. This hypothesis does not hold for most real-world scenarios, e.g., in YouTube videos, where occlusions and truncations at image boundaries are common, making action recognition more challenging.

State-of-the-art tracking algorithms [Hare et al., 2011] estimate a bounding box around the visible parts of a person, resulting in non-homogeneous tubes that can cover parts of the human body, the full-body or a mix of both in cases of close-up or moving cameras. For example, in Figure 4.1, a standard human tube extraction method frames the upper-body of the woman ironing, then the hands and arms, and finally the upper-body again. We posit that extracting full-body human tubes, even in case of occlusions or truncations, should help establish better correspondences and extract more discriminative features, improving action localization performance in complex scenarios.

The intuition behind our approach is that a bounding box corresponding to the full human body can often be inferred even if only parts of the person are visible—scene context and body pose constraints (feasible kinematic configurations) help estimate where the occluded or truncated body parts are (see the examples shown in our extracted tubes in Figure 4.1). To exploit such cues, we propose to train a human part detector that scores the visible parts but also regresses a full-body bounding box. We present a new tracking algorithm that simultaneously tracks several parts of a person-of-interest and combines the corresponding full-body bounding boxes inferred (regressed) from these parts to reliably localize the full-body. We demonstrate that our novel tube extraction approach outperforms state-of-the-art algorithms for action detection and localization [Weinzaepfel et al., 2016, 2015].

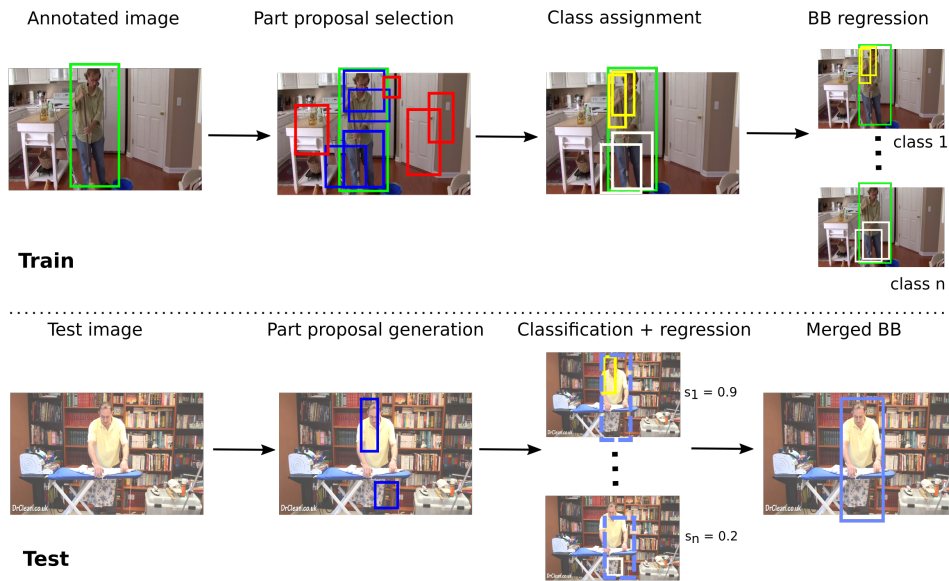


Figure 5.1 – Overview of our part detector. During training, the detector learns to select relevant part proposals in the annotated images. These proposals are assigned a specific class label and employed to regress the target full-body bounding box (BB). At test time, the detector generates part proposals, which are then classified and used to regress full-body bounding boxes. These are merged to estimate the final bounding box (Merged BB).

## 5.2 Method: From Parts to Tubes

We propose a new framework for action localization that tracks people in videos and extracts full-body human tubes, even in case of occlusions or truncations. This is achieved by training a novel human part detector that scores visible parts while regressing full-body bounding boxes. Figure 5.1 shows an overview of our detection architecture that is detailed in Section 5.2.1. The training phase begins with the selection of part proposals that overlap with groundtruth bounding boxes. These part proposals are then assigned a particular class label based on their height-width ratio and location with respect to the bounding box. Finally, a class-specific regressor is trained to infer the full-body bounding box from these parts. At test time, given an image, we first generate part proposals which are scored, and then use them to regress full-body bounding boxes, see example in Figure 5.1. We then merge these bounding boxes using a new tracking algorithm, detailed in Section 5.2.2. This

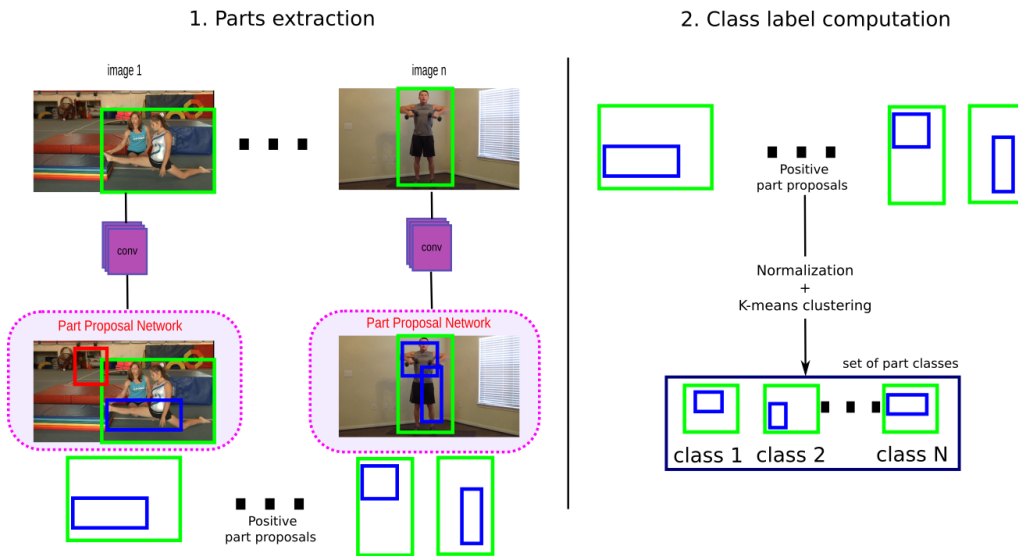


Figure 5.2 – Overview of our part classes creation. Part classes are defined with respect to the aspect ratio and location within the groundtruth box

algorithm simultaneously tracks several parts of a person-of-interest and combines the bounding boxes inferred from these parts to construct a full-body human tube. Two examples are given in Figure 4.1. Our tube extraction approach is then employed for action localization as detailed in Section 5.2.3.

### 5.2.1 Detecting Parts

Inspired by recent advances in deep learning for computer vision problems, we propose a CNN-based human part detector which is end-to-end trainable. Given a database of images annotated with full-body 2D human poses, we first define a set of human parts from this training data. Then, the learning phase trains our detector to: (1) generate relevant human part proposals through a region proposal network (RPN) [Ren et al., 2015], (2) classify and score them, and (3) regress the corresponding full-body bounding boxes.

**Part definition.** In this work, a human part is defined as a rectangular region covering a small area of the full-body bounding box. It is represented by its location within the bounding box (e.g., the “upper right” part), an aspect ratio (e.g., a “square” or a “rectangular” part), and a

scale. A set of parts is first extracted from the training set using an RPN. The parts must be big enough to contain relevant information and relatively small to ensure that multiple different parts will cover the bounding box. To ensure having small parts, we consider only regions overlapping a groundtruth box with an IoU score below a certain threshold. Each part is then represented by a four-dimensional vector containing its 2D location with respect to the center of the bounding box (normalized by bounding box height and width) and its normalized height and width, expressed as percentages of the height and width of the box. The four values are between 0 and 1. A K-means clustering is finally performed on the set of vectors. The centroids of the resulting  $K$  clusters define our set of part classes. We augment this set of classes with an additional full-body class. An illustration of our parts definition method is shown in Figure 5.2.

**Part proposal selection.** As in [Ren et al., 2015], our architecture uses an RPN to find a set of rectangular candidate regions. During the learning phase, this set is split into positive (blue boxes in Figure 5.1) and negative proposals (red boxes in Figure 5.1). We consider a proposal as positive if it is contained in the groundtruth box and has a fixed number of connected body keypoints, with two keypoints being connected if they are directly linked in the human skeleton (e.g., head and shoulders).

**Class-specific regression.** For the class assignment stage, proposals that have a large IoU score with the groundtruth box are labeled as full-body proposals, while others are assigned to the closest part class, which is obtained by choosing the corresponding centroid with the minimum  $\ell_2$ -distance. Note that the groundtruth bounding box is considered as a positive full-body proposal to include at least one positive exemplar for each class. The regression step then learns to regress the 2D image coordinates of the full-body bounding box. This is done independently for each class to ensure class-specific regression. The goal here is to localize the rest of the body from a single part. In the example in Figure 5.1 (top row), two proposal boxes each are assigned to the classes corresponding to the upper-left area (shown in yellow) and the legs area (in white). The full-body regression target is shown in green. We maintain a fixed ratio between part proposals and full-body exemplars in the training batches.

**Test time.** Our detector first generates relevant part proposals (blue boxes in the lower part of Figure 5.1). A full-body bounding box is regressed from each of these proposals (dashed boxes in the figure). These re-

gressed boxes corresponding to different part classes can be merged to produce a single full-body bounding box in a frame with a weighted average, where the weights are the classification scores. In Figure 5.1, the yellow box with a higher score (0.9) has a greater influence than the white box (0.2) in the final merged bounding box. This produces reasonable detections in several cases, but we present a more robust approach which leverages all the candidate boxes for building tubes.

## 5.2.2 Building full-body tubes

Given the detected parts, and the corresponding full-body bounding boxes, the next task is to build full-body tubes. We perform this by tracking all the parts detected in each frame, to associate them temporally in successive frames, and then use them jointly to localize the person(s) performing an action. To this end, we extend the tracking algorithm in [Weinzaepfel et al., 2015], which is limited to tracking the person as a whole and can not handle challenging cases where the person is occluded, as demonstrated in the experimental results (see Section 5.3.3). For clarity, we first explain how to track a single part from frame  $t$  to the frame  $t + 1$ , and then describe how to extend the tracking method to several parts.

**Initialization and tracking the first part.** Given a part box  $b_f$  at frame  $t$  of class  $c$  of our part detector, we employ a sliding window approach to track this part in the next frame. To this end, we combine a human part detector  $H_c$  (with a score function  $s_{H_c}$ ) and an instance-level detector  $I_c$  (with corresponding score function  $s_{I_c}$ ). The instance-level tracker learns human part appearance in the initialization frame  $t$ , by running a linear SVM on features from the last fully-connected layer  $fc7$  of our part detector. The tracker is updated every frame with the corresponding chosen box, in order to handle appearance changes over time. We provide further details of these steps in the following.

To initialize our tracking algorithm, we find the box  $b^*$  from frame  $t$  with the highest score among all the part classes and all the frames. We denote the feature maps for the box  $b^*$ :  $fc7^{b^*} = fc7(b^*, t)$ . Assuming the box  $b^*$  is of class  $c$ , we denote  $\mathbf{B}^*$  the corresponding full-body (regressed) bounding box as:

$$\mathbf{B}^* = f_{reg}^c(fc7^{b^*}, b^*). \quad (5.1)$$

Here,  $f_{reg}^c$  is the class-specific regressor for class  $c$  of our part detector.  $B^*$  is the initialized bounding box of our tube in frame  $t$ .

An instance-level detector is learned using the box  $b^*$  and features from the last fully connected layer denoted by  $fc7$ . A linear SVM is learned using as positive  $\mathcal{P}_c$  the feature from the set  $\mathcal{P}_c = \{b^*\}$  (the set is initialized with the bounding box  $b^*$ ) and as negatives the features from detected parts at frame  $t$  whose overlap with  $b^*$  is low. The instance-level detector basically learns the body part appearance. For instance, if the tracked part is the torso, the color information of the t-shirt and the appearance of the neck are represented by the instance-level detector, and then are used during tracking. Note that the SVM is specified to the class  $c$ .

Instance-level detector  $I_c$  and human part detector  $H_c$  are used to track the part  $b^*$  in the next frame. In frame  $t + 1$ , we perform a sliding window search within the neighborhood of the tracked box  $b^*$ :  $\mathcal{N}(b^*)$ . More precisely, we use the box  $b^*$ , translated with a vertical and horizontal shift of  $0, \pm\sigma, \pm2\sigma, \pm3\sigma, \pm4\sigma$  where  $\sigma$  is the stride of the network, and similarly for rescaled versions of the box by a factor  $0, \pm10\%, \pm20\%$ . We select the top scoring box according to a combined score, including the generic part detector score  $s_{H_c}$  and the instance-level detector  $s_{I_c}$ , i.e., the SVM score of learned appearance:

$$b_{t+1} = \arg \max_{b \in \mathcal{N}(b^*)} (s_c(b)), \quad (5.2)$$

where:

$$s_c(b) = s_{H_c}(b) + s_{I_c}(b). \quad (5.3)$$

The top scoring box  $b_{t+1}$  is then added to  $\mathcal{P}_c$ , and the instance-level detector is updated. The full-body box  $B_{t+1}$  is obtained from our network:  $B_{t+1} = f_{reg}^c(fc7^{b_{t+1}}, b_{t+1})$ , and becomes part of the tube for frame  $t + 1$ . If we limit our tracker to a single part, we then proceed with our tracking until the end of the sequence.

**Tracking several parts.** Limiting the tracker to a single body type, which may become occluded in some of the frames, is prone to losing the person being tracked. To address this, we detect other parts included in the full-body box  $B_{t+1}$ . For example, if the first tracked part is the torso of a person, a second part could be the legs also detected by our parts detector. We select the parts which fall into the tube at frame  $t + 1$ ,  $B_{t+1}$ , and whose part score is above a certain threshold  $th$ . We denote  $\{b_{t+1}^{c_i}\}$  the set of the parts to track. In our previous example,  $\{b_{t+1}^{c_i}\}$  could

be the set comprising the tracked part corresponding to the torso,  $b_{t+1}^{c_1}$ , and the new detected part corresponding to the legs,  $b_{t+1}^{c_2}$ . Note that for each tracked class  $c_i$ , only one part is associated and thus tracked. Each part is then tracked independently. An instance-level detector is learned and updated for every tracked class. Following Equation 5.2, the part in frame  $t + 2$  for class  $c_i$  is:

$$b_{t+2}^{c_i} = \arg \max_{b \in \mathcal{N}(b_{t+1}^{c_i})} (s_{c_i}(b)). \quad (5.4)$$

The full-body box  $B_{t+2}^{c_i}$  is estimated following Equation 5.1. In our example, two full-body boxes are computed:  $B_{t+2}^{c_1}$  is the regressed bounding corresponding to the torso part  $c_1$ ,  $B_{t+2}^{c_2}$  is the regressed bounding corresponding to the leg part  $c_2$ . The final box  $B_{t+2}$  of the tube is computed by merging the regressed boxes of different tracked parts using a weighted sum:

$$B_{t+2} = \sum_{\{c_i\}} w_{c_i} B_{t+2}^{c_i}, \quad (5.5)$$

where:

$$w_{c_i} = \frac{s_{c_i}(b_{t+2}^{c_i})}{\sum_{c_i} s_{c_i}(b_{t+2}^{c_i})}. \quad (5.6)$$

A part with a high global score  $s_{c_i}$  will contribute more to the computation of the tube at  $t + 2$   $B_{t+2}$  than a part with a low score. In our example, if the global score of the torso part is high (e.g.,  $s_{c_1} = 1.8$ ) and the global score of the leg part is low (e.g.,  $s_{c_2} = 0.5$ ), then the weight of the regressed full-body box from the torso part  $B_{t+2}^{c_1}$  is:  $\frac{1.8}{0.5+1.8} = 0.78$  and the weight of the full-body box regressed from the leg part  $B_{t+2}^{c_2}$  is 0.22.

At frame  $t + 2$ , tracked parts can be removed from the set  $\{b_{t+2}^{c_i}\}$  if their global score is below a certain threshold. New parts to be tracked are added following the same rule as before. A new set of part boxes  $\{b_{t+2}^{c_i}\}$  is thus created and tracked in the next frame. This step prevents the tracker from being impacted by occlusions, removing parts which were visible and tracked during the sequence and are now occluded. In the other way, if parts which were occluded become visible, they are added to the set of tracked parts. Note that a part can be visible and tracked, become occluded and thus not tracked, then visible and tracked again. The instance-level detector remains the same during the entire sequence and is not updated during the occlusion. We iterate until the end of video.



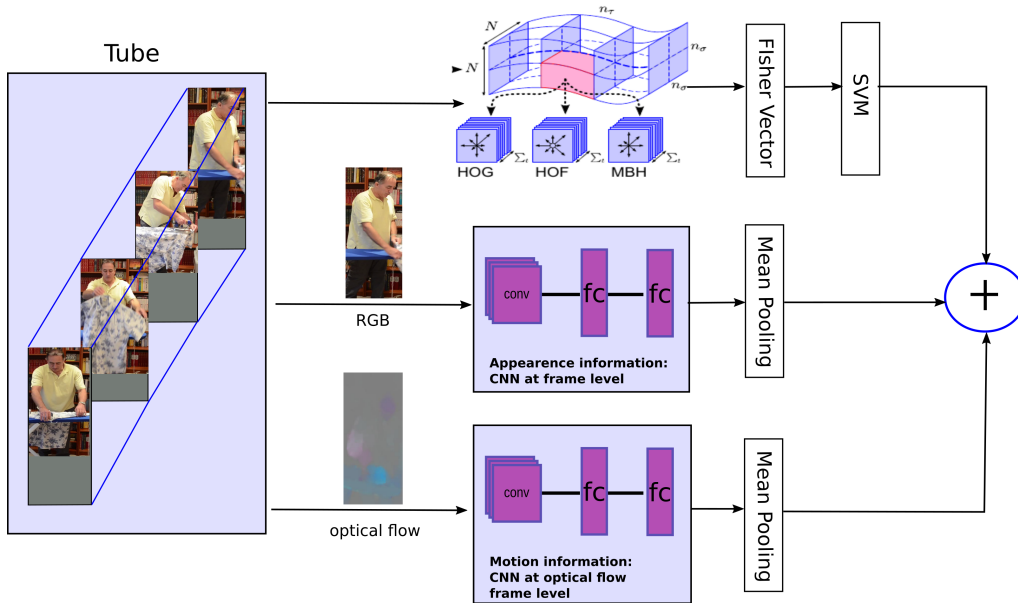


Figure 5.3 – Overview of our tube scoring method. Dense trajectories, appearance CNN, and motion CNN are extracted from the tube. A fusion strategy outputs a global action score for the tube

### 5.2.3 Action localization

The final step of our approach is to localize actions from the extracted full-body tubes, see Figure 5.3 for an overview. We achieve this by representing tubes with features and then learning an SVM for recognizing actions. We use dense trajectories, appearance and motion CNNs as features.

Dense trajectories are used with the four popular descriptors: HOG, HOF, MBHx, MBHy. For each tube, we build a Fisher vector per descriptor type, using only the trajectories that start inside the track. Each of the four Fisher Vectors is then independently power-normalized and L2-normalized [Sanchez et al., 2013]. A tube is finally described by the concatenation of the four normalized Fisher vectors. We also perform a spatial pyramid matching (SPM) on the tube.

Appearance and motion CNNs are based on the R-CNN architecture proposed in [Girshick et al., 2014]. A probability output from softmax layer and a fully-connected regression layer are associated for each event of the dataset (10 for DALY, 20 for JHMDB). Given a region which is resized to  $227 \times 227$  pixels, a spatial-CNN operates on RGB channels and

captures the static appearance of the actor and the scene, while a motion-CNN takes as input optical flow and captures motion patterns.

A human tube is considered as positive if the temporal intersection over union with the annotated frames is above a certain threshold. The temporal IoU is defined as the average per-frame IoU. For this step, our tubes are cropped to frame boundaries to be comparable with groundtruth annotations. For both RGB and flow CNN, a R-CNN network is trained on the respective dataset (i.e., JHMDB and DALY) as follows: region proposals in a frame whose IoU with our estimated bounding box is above a threshold are labeled as positives for the class of the tube. Action classifiers are learned for dense trajectories and then combined with the score of RGB and flow CNNs using a late-fusion strategy. These steps are described in detail in Section 5.3.2.

## 5.3 Experiments

### 5.3.1 Datasets

**Test data.** For action localization, we evaluate our method on two action recognition datasets, namely JHMDB [Jhuang et al., 2013a] and DALY [Weinzaepfel et al., 2016]. JHMDB is a standard action recognition database used in Saha et al. [2016], Peng and Schmid [2016], Weinzaepfel et al. [2015], Gkioxari and Malik [2015], Weinzaepfel et al. [2016]. It is a subset of the larger HMDB51 dataset collected from digitized movies and YouTube videos. It contains 928 videos covering 21 action classes involving a single person in action: *brush hair, catch, clap, climb stairs, golf, jump, kick ball, pick, pour, pull-up, push, run, shoot ball, shoot bow, shoot gun, sit, stand, swing baseball, throw, walk, wave*. Only the person in action is annotated in each clip. The first and the last frames roughly correspond to the beginning and the end of an action, i.e. no untrimmed videos). In total, it results in 36-55 clips per action class with each clip containing 15-40 frames. In summary, there are 31,838 annotated frames in total.

DALY is a more challenging large-scale action localization dataset consisting of 31 hours of YouTube videos (3.3M frames, 3.6k instances) with spatial and temporal annotations for 10 everyday human actions. The action categories are *applying make up on lips, brushing teeth, cleaning floor, cleaning windows, drinking, folding textile, ironing, phoning, playing harmonica* and *taking photos/videos*. Each video lasts between 1 and 20 minutes with an average duration of 3min 45s. The dataset is split into

31 training videos and 20 test videos for each class. Temporal annotation of the 10 actions results in 3637 instances in total. Actions are short (8s on average) with some classes having very brief instances (e.g., *drinking*) or somewhat longer (e.g., *brushing teeth*). The videos are untrimmed: 75% of the frame do not contain any action. For each instance, the spatial extent is provided, i.e., a bounding box around the actor for 5 frames, regularly sampled over time, with a maximum of one frame per second for short instances.

We also use the LSP dataset [Johnson and Everingham, 2010] for analyzing our full-body box generation method. LSP contains 2000 pose-annotated images of people doing sports, collected from Flickr : *athletics, badminton, baseball, gymnastics, parkour, soccer, tennis, volleyball*. The images have been scaled such that the most prominent person is roughly 150 pixels in length. Each image is also annotated with 14 joint locations. Left and right joints are consistently labeled from a person-centric viewpoint.

**Training data.** For training our human part detector, we use the MPII human pose dataset [Andriluka et al., 2014]. The dataset includes around 17K images containing over 40K people with annotated body joints. The images were systematically collected using an established taxonomy of everyday human activities. Overall, the dataset covers 400 human activities and each image is provided with an activity label, which we do not use in this work. For training our human part detector, we use the whole dataset of around 17K images, with each scene containing at least one person, often occluded or truncated at frame boundaries. As in Weinzaepfel et al. [2016], we compute a groundtruth bounding box for each person by taking the box containing all the annotated body keypoints with a fixed additional margin of 20 pixels. To obtain full-body bounding boxes for a dataset where only visible keypoints are annotated, we employ a method which infers the full pose of the person in case of occlusions and truncations, see Figure 5.4 for an illustration. We first extract the incomplete 2D pose from MPII training set, and normalize it (keypoint values are between  $[-1, 1]$ ). We then employ a nearest neighbor search on the annotated keypoints to complete missing annotations and recover complete full-body 2D poses. As done in Rogez et al. [2017], we generate a large set (8M) of human 2D poses by projecting 3D poses from the CMU Motion Capture dataset on multiple random camera views. 2D full-body poses are also normalized between  $[-1, 1]$ . Then, for each incomplete 2D pose in the MPII training set, a search is performed on the annotated 2D joints to estimate the closest match, i.e., full-body 2D pose, that is later

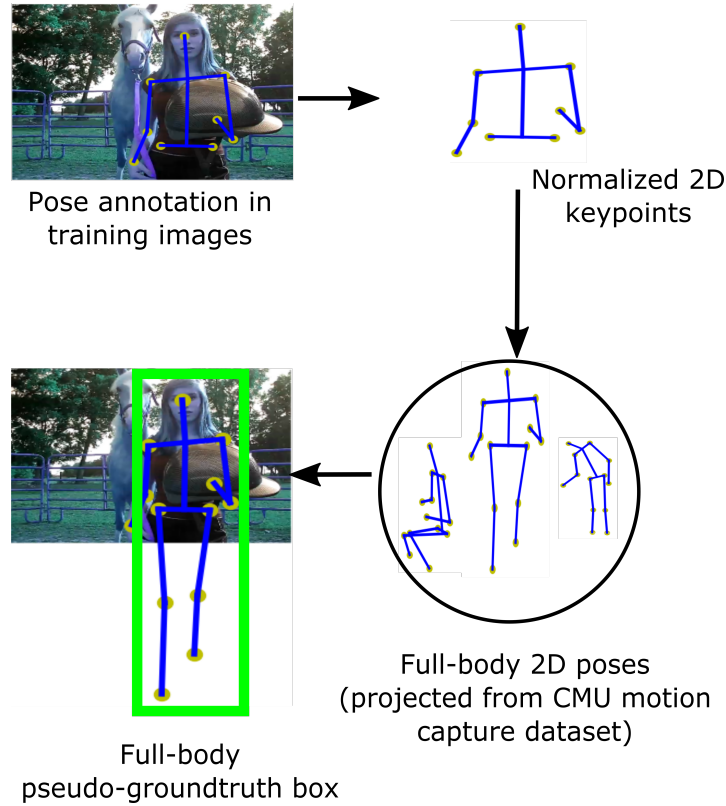


Figure 5.4 – Our full-body pseudo-groundtruth generation. A nearest neighbor search is employed to complete the missing annotations.

employed to estimate a full-body bounding box. More specifically,  $l_2$  distance between incomplete and complete skeletons is evaluated only on annotated keypoints. The closest match is then realigned with respect to the MPII skeleton, and is used as a pseudo-groundtruth to extract a full-body box.

### 5.3.2 Implementation details

The implementation of our part detector is based on Faster-RCNN [Ren et al., 2015] with VGG16 layers [Simonyan and Zisserman, 2014a]. The number of classes is set to 21: 20 human parts and the full body class.

**Parts definition.** In Figure 5.5, we show the 20 normalized part centroids computed by our method, i.e., the result of the k-mean clustering performed on the set of part proposals, which are extracted on the MPII-

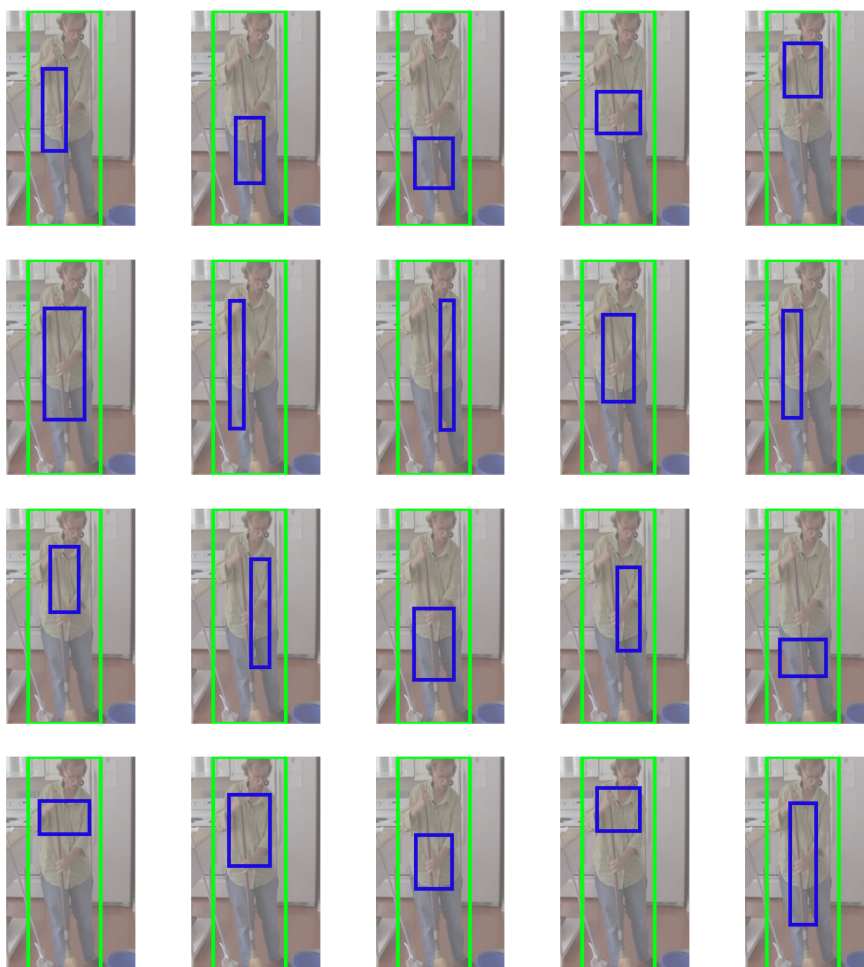


Figure 5.5 – Results of our automatic part definition. Bounding boxes corresponding to part centroids are shown in blue. Groundtruth boxes are in green.

dataset by our part detector. The detector has pre-trained weights on ImageNet [Deng et al., 2009]. The centroids cover the whole area of the normalized groundtruth. We observe a left-right symmetry distribution, and bottom-up symmetry distribution. For instance, the third part in the top row presents a symmetry with the fifth part (top row).

**Part detector.** The detector is trained on the MPII dataset. The connections between joints are defined following the standard human skeleton (e.g., head connected to shoulders, shoulders to elbows). For the part

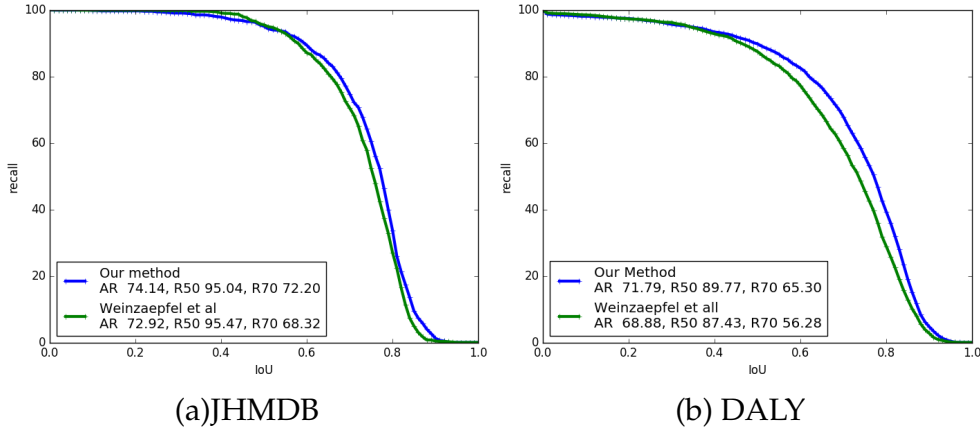


Figure 5.6 – Recall of tubes extraction method on DALY et JHMDB, and comparison with state-of-the art method[Weinzaepfel et al., 2016].

proposal selection stage, we consider a proposal as positive if it overlaps the groundtruth box and contains exactly three connected body keypoints. We tested with different numbers of keypoints and found that three was an optimum number to maximize human detection rate on DALY (with recall@0.5). For the class assignment stage, proposals with IoU more than 0.55 are labeled as full-body proposals, while those with IoU between 0.1 and 0.55 are assigned to the class of the closest part.

**Training.** Initialization of the network is done with ImageNet pretrained weights. The number of iterations is set to 180K, the learning rate to 0.001, the momentum to 0.9, the gamma parameter to 0.1, i.e., the learning rate is divided by 10 at every learning step (@ 100K, 150K, 170K iterations). We use batches of 128 proposals (32 positive and 96 background). We define the ratio of full-body proposals over the entire set of proposals in a batch as:

$$FBR = \frac{N_{fb}}{N_{fb} + N_p} * 100 \quad (5.7)$$

where  $N_{fb}$  is the number of full-body proposals and  $N_p$  is the number of part proposals in the batch. By fixing  $FBR$  at 9%, we add the constraint to have approximately 10 times more part proposals than full-body exemplars.

**Human tubes.** For the DALY dataset, tubes are computed using every

fifth frame for computational reasons. During the tracking procedure, a box is removed if its combined score  $s_c$  (defined in Section 5.2.2) is less than 1. A new part is added to the tube if it has a detector score of 0.25.

**Action localization: Positive and negative tubes.** Five annotated frames per sequence are used for computing the temporal IoU with our tubes. A human tube is considered as positive if the temporal IoU with the five annotated frames is above 0.5.

**Action localization: Improved dense trajectories.** The dimensions of the four descriptors (HOG, HOF, MBHx and MBHy) are reduced by a factor of 2 using PCA and a codebook of 256 Gaussians. Each tube is represented by a vector of 102,400 dimensions. For SPM, we split the tubes into three vertical bins, and compute, for each of the three bins, a Fisher vector in the same way as the full tube. The four Fisher vectors (one for the full tube, three for each of the vertical bins) are then stacked into a single feature of 409,600 dimensions. Note that since our full-body tubes can be partially and temporally out of the video frame, some bins may have no trajectories starting inside them. For example, a video showing only the legs generates some tubes where the upper body part is outside the video frame, and thus has no video representation. This bin is then represented by a zero-vector.

**Action localization: Appearance and motion CNNs.** For motion CNNs, the optical flow signal is transformed into a 3-dimensional image by stacking the x and y components and the magnitude of the flow. Each image intensity is then multiplied by 16 and converted to the closest integer between 0 and 255. We use LDOF [Brox et al., 2004] to estimate the optical flows. When training appearance and motion CNNs, the region proposals whose IoU with our estimated bounding box is above 0.5 are labeled as positives for the class of the tube. The region proposals with no class label are considered as negative and are labeled as background. The number of iterations is set to 70K, the learning rate to 0.001, the momentum to 0.9, the gamma parameter to 0.1, i.e., the learning rate is divided by 2 at 30K iterations. We use batches of 128 proposals (32 positive and 96 background).

**Testing.** For each tube extracted from test videos, three scores are computed, one for each feature (improved dense trajectories, appearance and motions CNNs). For dense trajectories, a linear SVM classifier is learned for each class of the dataset. Scores are scaled between 0 and 1 using a



Features	Method	DALY		JHMDB	
		meanAP@0.5	meanAP@0.7	meanAP@0.5	meanAP@0.7
Dense Trajectories	Ours	58.97	31.35	64.91	46.45
	[Weinzaepfel et al., 2016]	53.21	21.57	60.11	41.39
Appearance & motion CNNs	Ours	63.51	38.21	61.81	46.12
	[Weinzaepfel et al., 2016]	61.12	28.37	64.08	49.22
	[Gkioxari and Malik, 2015]	-	-	53.30	-
	[Weinzaepfel et al., 2015]	-	-	60.70	-
	[Peng and Schmid, 2016]	-	-	73.10	-
	[Saha et al., 2016]	-	-	71.50	-
Combination	Ours	67.79	39.05	68.85	49.10
	[Weinzaepfel et al., 2016]	64.56	29.31	65.80	49.54

Table 5.1 – Comparison to the state of the art with mAP@0.5 and mAP@0.7 measures on DALY and JHMDB datasets. We report results for the fully-supervised variant of Weinzaepfel et al. [2016].

sigmoid. For appearance and motions CNNs, scores for each class are computed at frame level. The score of a tube is the mean of all the frame-level scores.

### 5.3.3 Results

In Table 5.1, our method shows an improvement over Weinzaepfel et al. [2016] on both DALY and JHMDB, of 3.21% and 3.05% respectively for mAP@0.5. A larger gain is obtained with mAP@0.7 on DALY (9.74% for “Combination”), showing that our method is more accurate for action localization in videos. All the results of Weinzaepfel et al. [2016] were obtained directly from the authors. For AP@0.5, per event results emphasize the role of detecting and tracking multiple parts (see Table 5.2). Compared to Weinzaepfel et al. [2016], we significantly improve the performance for actions such as *Applying make up on lips* with 81.91% (vs 68.18% for Weinzaepfel et al. [2016]), *Brushing teeth* with 68.64% (vs 57.61%). Videos of these actions are often close-up views, where the body is not fully visible during all or part of the sequence. This makes computing feature correspondences between frames more difficult for methods such as Weinzaepfel et al. [2016] which do not estimate the full-body bounding box. The difference between the two methods is even more important for AP@0.7: 49.27% (vs 2.62%) for *Applying make up on lips*, 28.62% (vs 20.58%) for *Brushing teeth*. Our human tubes estimate the position of the full body and infer the location of non-visible parts. This provides a canonical region of support for aggregating local descriptors which belong to the same parts of the body. Although the method in Peng and Schmid [2016] shows better results on JHMDB, our method has the ad-

Classes	[Weinzaepfel et al., 2016]		Ours	
	AP@0.5	AP@0.7	AP@0.5	AP@0.7
ApplyingMakeUpOnLips	68.18	2.62	81.91	49.27
BrushingTeeth	57.61	20.58	68.64	28.62
CleaningFloor	88.54	72.56	86.13	66.85
CleaningWindows	77.37	35.25	78.13	53.05
Drinking	44.10	13.99	36.77	24.86
FoldingTextile	58.90	35.35	60.77	15.94
Ironing	78.28	39.38	82.52	29.68
Phoning	52.06	25.05	63.41	34.19
PlayingHarmonica	68.36	26.93	70.18	58.12
TakingPhotosOrVideos	52.19	21.36	49.42	29.95
Mean	64.56	29.31	67.79	39.05

Table 5.2 – Per-event results on the DALY dataset of our method and Weinzaepfel et al. [2016]. The results of our method correspond to the one using five parts for human tracking (see Section 5.2.2).

vantage of being scalable to larger datasets and longer videos. It can also be applied in a weakly-supervised way.

**Part detector.** Figure 5.7 shows a selection of qualitative results for our part detector. Blue boxes are detected parts, green boxes are full-body boxes inferred from the parts. When only the upper-body is visible, our detector detects upper part and infers the location of the legs (see for instance the first row of Figure 5.7). In some challenging situations, like people occluded by textile, our detector recognizes the visible parts, and infers the full-body location, showing the effectiveness of the method. In the bottom-left image, our detector detects legs and estimates, from this part, the location of the full-body.

**Tubes extraction.** Figure 5.6 shows the recall of our method and Weinzaepfel et al. [2016], on JHMDB and DALY datasets. Our method improves tube extraction method in Weinzaepfel et al. [2016] by 3.78% for R@0.7 on JHMDB. The area under curve (AUC) is 74.14% for our method and 72.92% for [Weinzaepfel et al., 2016]. On the DALY dataset, our tube extraction method improves the state-of-the art method [Weinzaepfel et al., 2016] by 9.02% for R@0.7 and 2.34% for R@0.5. On both the datasets, our tube extraction performs significantly better for a high recall, showing the pertinence of our approach. Note that our tubes are designed to frame the full body, although recall is computed on the groundtruth given by the dataset, i.e., visible parts in frames. This annotation bias is studied below, in “Analysis of annotations” in the results section 5.3.3. Figures 5.1 and 5.8 show a selection of qualitative results. Although Figure 5.1 shows standing persons, people seated are also well-detected.

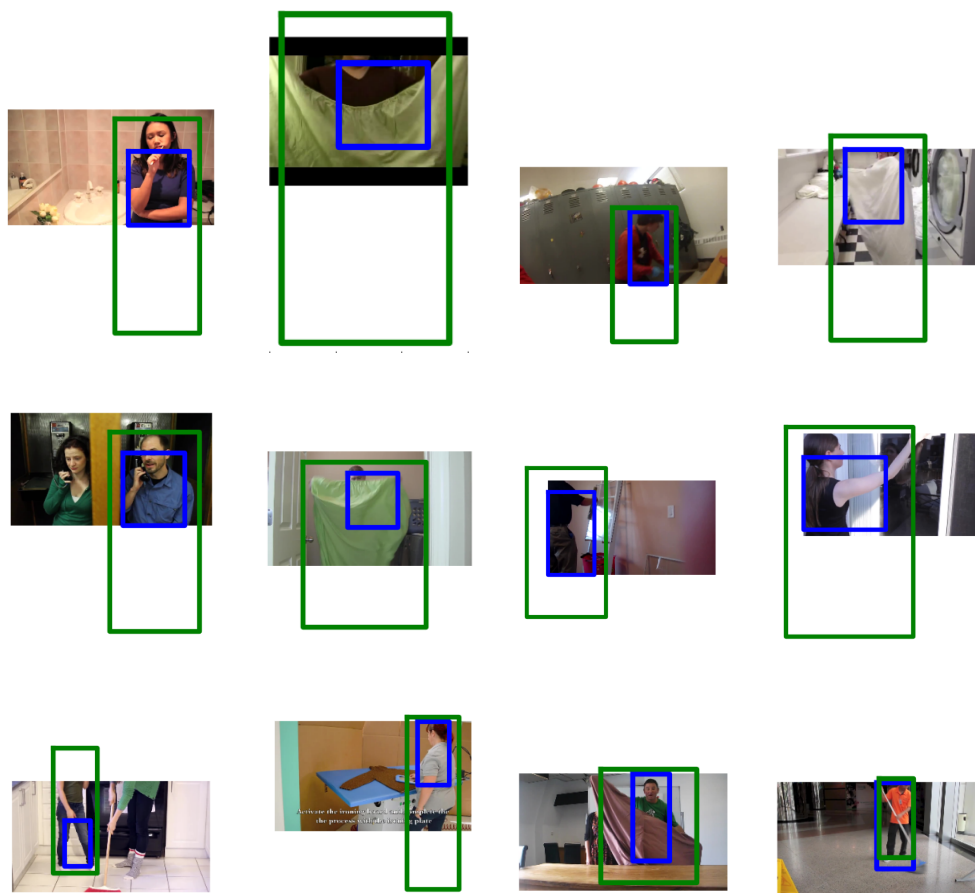


Figure 5.7 – Qualitative results of our part detectors on DALY frames. Blues boxes are detected parts, green boxes are full-body boxes inferred from displayed detected parts.

For example, for the *Playing Harmonica* event in DALY, which contains videos of people sitting (34 examples) and standing (16), we observe a significant improvement: over 1.8% and 31% for AP@0.5 and AP@0.7 respectively. Figure 5.8 compares our tubes with those extracted by [Weinzaepfel et al. \[2016\]](#), showing that our method better handles close-up views and occlusions.

**Influence of part trackers.** Figure 5.9 shows the mean average precision of our method when varying the number of parts being tracked

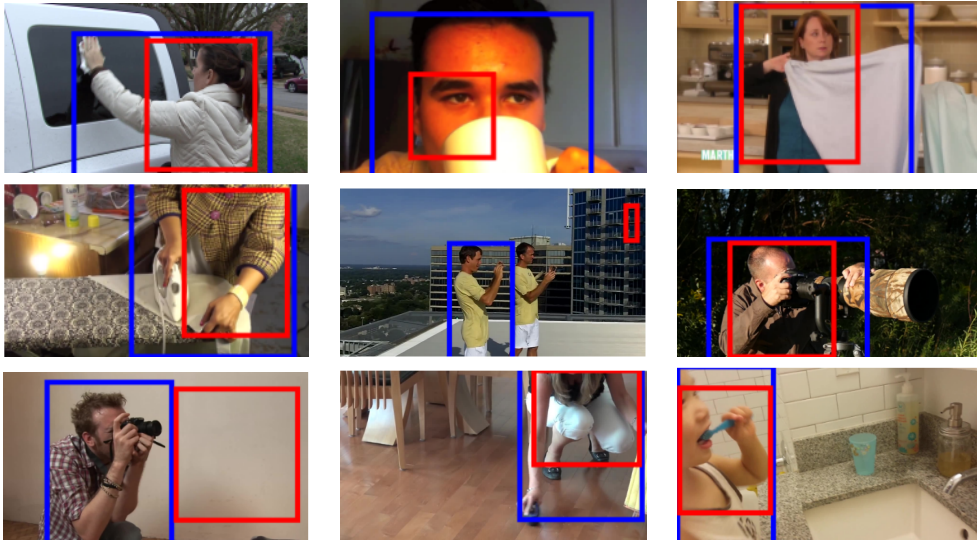


Figure 5.8 – Qualitative results. The visible part of our full-body tubes is shown in blue. For comparison, the tubes of the state-of-the-art method [Weinzaepfel et al., 2016] are in red. Here, we show a sample frame from nine example videos corresponding to *Cleaning windows*, *Drinking*, *Folding Textile*, *Ironing*, *Taking photography or videos*, *Cleaning floor*, *Brushing teeth* events of DALY.

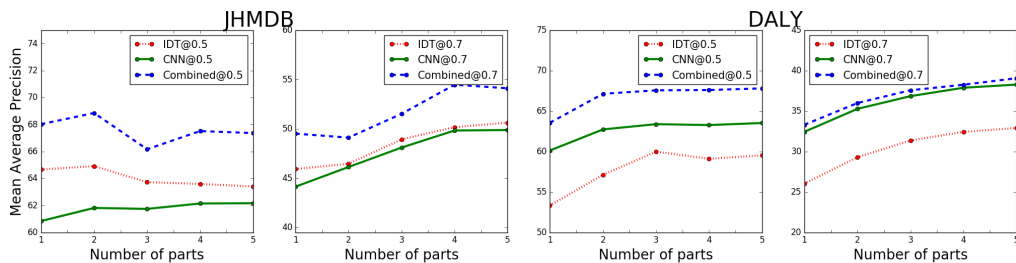


Figure 5.9 – mAP@0.5 and mAP@0.7 results on JHMDB and DALY datasets with respect to the number of parts used for building human tubes.

when building our tubes. The gain of adding parts is particularly significant with AP@0.7 for JHMDB. For AP@0.5, two-part tracking gives the best results because videos are short and viewpoint changes are limited. For AP@0.7, tracking a maximum of four parts improves average precision significantly. On average, we obtain an mAP of 54.46%, with

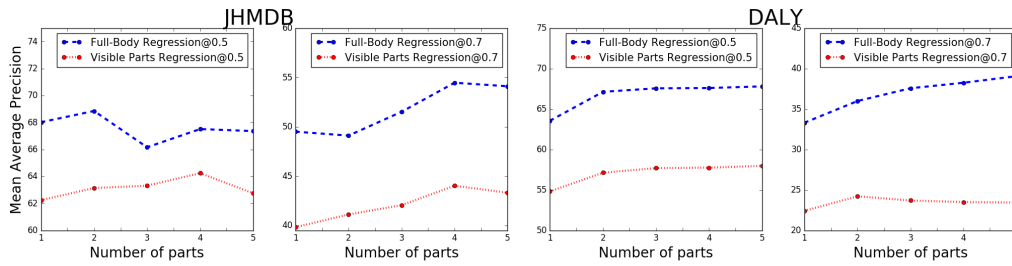


Figure 5.10 – Comparison of our method (trained with full-body boxes) and a variant that only uses visible regions. mAP@0.5 and mAP@0.7 results on JHMDB and DALY datasets with respect to the number of parts used for tube generation are shown.

an improvement of 4.92% over [Weinzaepfel et al. \[2016\]](#). The results for a few specific actions highlight the effectiveness of our tracking. For example, the *wave* action has an average precision of 32.49% with 1-part tracking, and 49.92% when tracking 4 parts. Videos of this action contain two different points of view: a “full-body” point of view, and a “torso” point of view, making the use of full-body tubes relevant and effective. A similar observation can be made with the *climb stairs* action (“full body” and “legs” points of view), with a gain of 8.27% (50.80% with single-part tracking, 59.07% with 4 parts), and also for the *throw* action, with a significant number of upper-body and head videos (18.74 % with single-part tracking, 45.90% with 4). On the contrary, the *walk* action shows better results with a single-part tracker (67.56%, compared to 64.59%) because the body of the person walking is fully visible in all the videos, and using the full-body class suffices. On the DALY dataset, a five-part tracker gives the best results. This is partly due to DALY being a much more challenging dataset than JHMDB.

**Influence of fully-body tubes.** Figure 5.10 compares the performance of part detectors that regress to full-body (including occluded or truncated regions) with those that regress only to visible regions (i.e., non full-body). Building non full-body tubes decreases the performance, for example, from 68.85% to 63.14% on JHMDB, from 67.79% to 57.97% on DALY for AP@0.5. It confirms our idea that building full-body tubes instead of the standard ones are well-adapted for action localization and classification, and can: (1) establish better feature correspondences, and (2) better exploit techniques such as spatial pyramid matching (SPM) for recognition tasks. Additional experiments show that SPM is more effec-

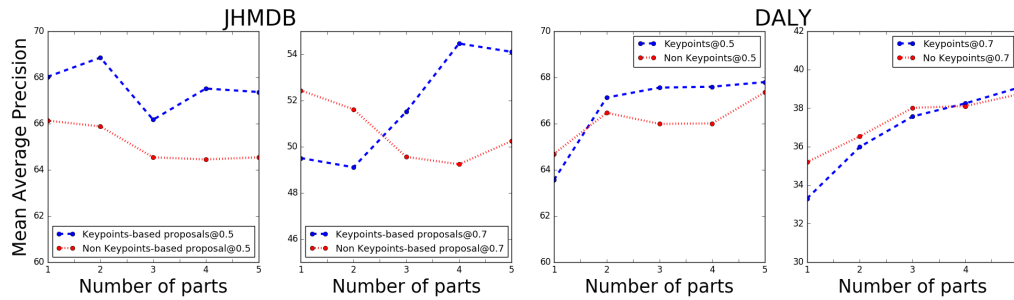


Figure 5.11 – Influence of keypoint based proposal generation.

tive with dense tracks when considering our full-body tubes (+3% mAP) vs cropped and mis-aligned tubes from [Weinzaepfel et al. \[2016\]](#) (+1%). In essence, such an “amodal completion” defines a better reference frame for features (spatio-temporal grid is more adapted as person-centric), and results in better performance in the context of action localization.

**Influence of keypoints.** The results in Figure 5.11 highlight the importance of keypoint based proposal generation. We compare our full method, which uses keypoints for selecting parts proposals (refer Section 5.2.1) with a variant that considers a proposal as positive if its overlap with ground truth is in the range of 0.2 and 0.6, i.e., without using keypoints. The performance of this no-keypoint variant is lower than our full method for mAp@0.5: 65.87% vs 68.85% on JHMDB, and 67.35% vs 67.79% on DALY.

**Influence of the number of keypoints.** We study the influence of the number of keypoints used to select positive proposals during the training phase of our parts detector, in Figure 5.12. On the JHMDB dataset, part detector trained with positive proposals which contain 3 connected keypoints performs better than part detector based on 2 and 4 keypoints (for mAp@0.5: 68.85% for 3-keypoint based detector, 65.49% for 2-keypoint based detector, 65.13% for 4-keypoint based detector). It indicates that the mean area covered by three connected keypoints is optimal for learning a parts detector. For instance, the 2 keypoints-based detector outputs smaller parts than 3 keypoints-based detector. On the DALY dataset, the 3-keypoint based detector also performs better than the 2-keypoint based detector (67.79% vs. 65.30%).

**Influence of fixing FBR during the training phase.** Figure 5.13 shows



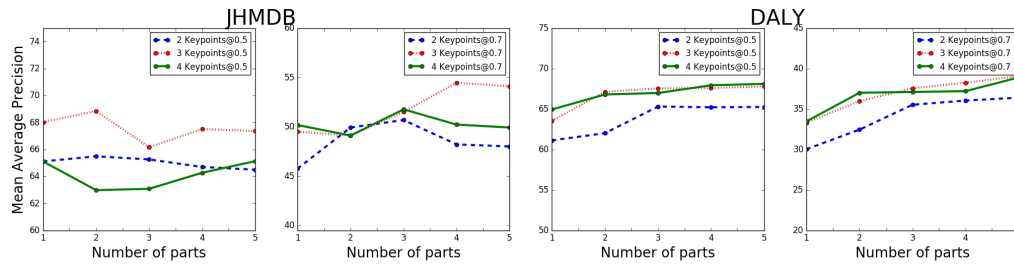


Figure 5.12 – Comparison of our method (trained with full-body boxes) and a variant that only uses 2 keypoints and 4 keypoints to generate proposals during training. mAP@0.5 and mAP@0.7 results on JHMDB and DALY datasets with respect to the number of parts used for tube generation are shown.

the evolution of FBR when augmenting the number of iterations during training. Here, the FBR is not fixed. During the first 10K iterations, FBR is equal to 27%. FBR increases to 31% in the range 40-50k, 36% in the range 100-110k, and 42% in the range 120-140k. This result shows that the detector learns to generate more and more full-body proposals, to the detriment of the 20 part class proposals. Figure 5.14 shows the influence of fixing the FBR to 9% for action localization. A network trained with a fixed FBR performs better than a network trained without a constant FBR: 68.85% vs. 67.02% on JHMDB dataset for mAP@0.5, 67.79% vs. 65.79% on DALY dataset for mAP@0.5. Learning with a constant FBR enables the network to both learn how to simultaneously detect full body and body parts. With no constraint, the network will focus on full-body detections only.

**Analysis of full-body box generation.** We simulated partially-occluded human poses on the LSP dataset for this analysis. Given a full pose, we successively remove the lowest keypoint in the human pose/skeleton, then the two lowest keypoints, and so on. We then estimate the full-body box with our method for each of these simulated incomplete poses, i.e, the box which frames the full estimated pose. The effectiveness of this estimation is measured by comparing it with the groundtruth box. We do this experiment by removing successively the highest, the left-most, and the right-most keypoints. Results are shown in Figure 5.15. Full-body boxes estimated with 9 out of the 13 keypoints (which corresponds to missing legs for the “lowest” experiment, and missing head and shoulders for the “highest” experiment) gives a mean IoU over 0.7 with the



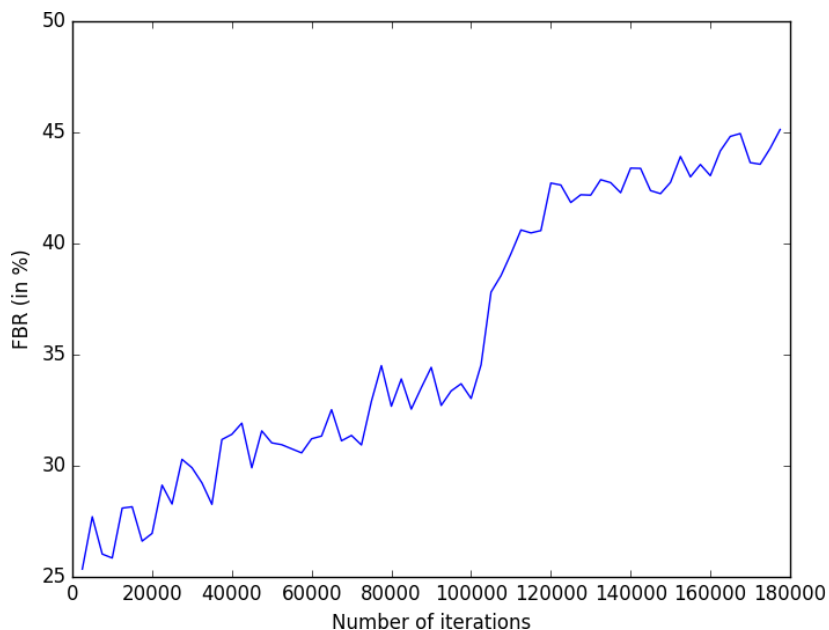


Figure 5.13 – FBR with respect to the number of iterations during training. If the FBR is not fixed, the full-body class becomes more and more important, to the detriment of part classes.

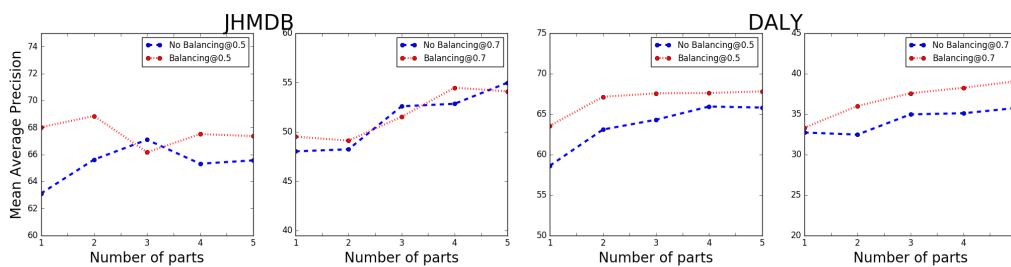


Figure 5.14 – Comparison of our method (trained with full-body boxes) and a variant that do not use a constant FBR during the training phase.

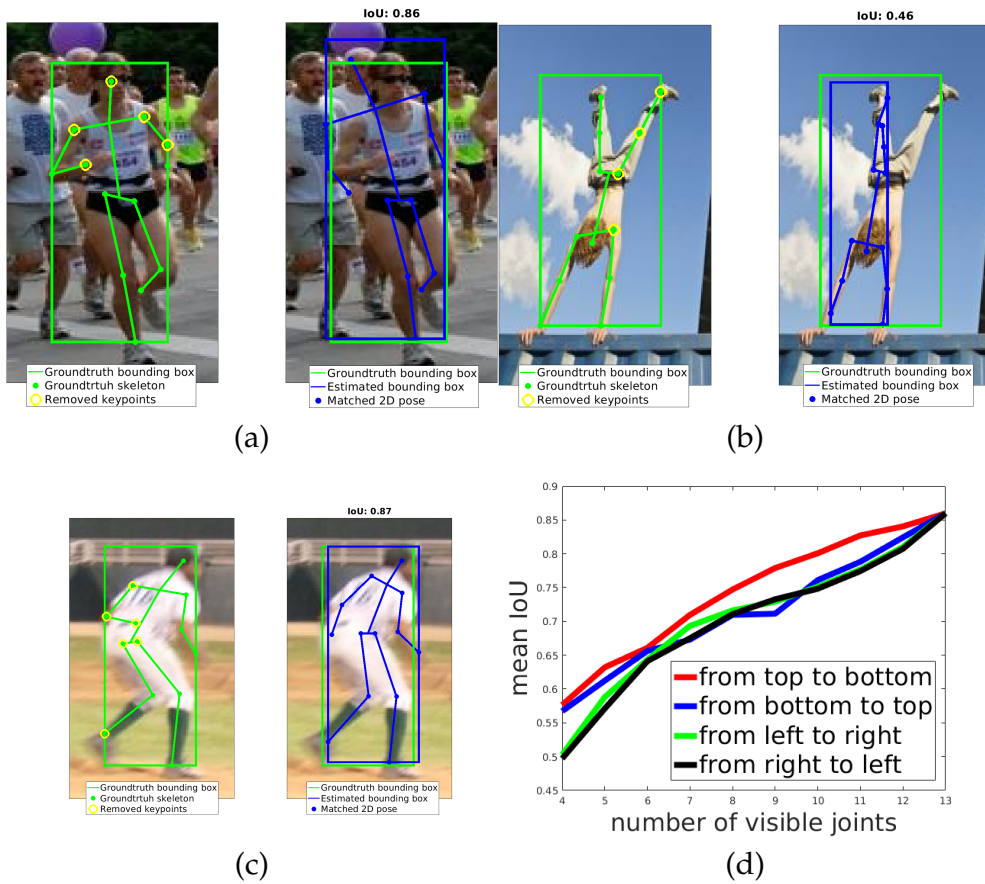


Figure 5.15 – Numerical validation of our full-body box generation method. The curves show the mean IoU of our estimated bounding box when removing keypoints from top to bottom (a), left to right (b), right to left (c).

groundtruth in all the four cases. With only 4 out of 13 keypoints, the IoU remains relatively high (between 0.5 and 0.6).

**Analysis of annotations.** Although our method shows state-of-the-art results on action localization, it suffers from an annotation bias. Our human tubes frame the full body, including hidden parts. For the Ironing event in DALY, legs are frequently hidden by ironing boards (see Figure 5.16 with three examples of DALY videos where the ironing event occurs), and the annotations are focused on visible parts of the body, i.e., the torso and arms. Consequently, the IoU between our tubes and annotations suffers from the fact that they do not cover the same parts

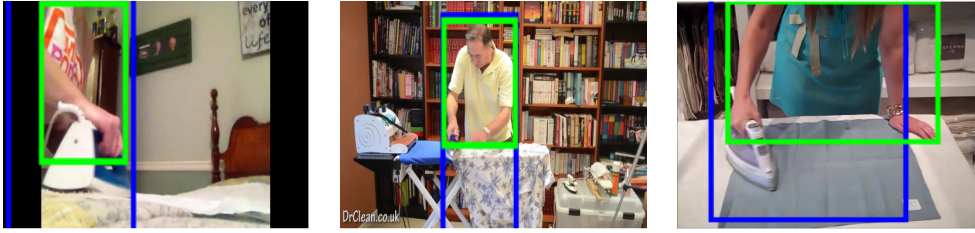


Figure 5.16 – Illustration of annotation bias for three samples from the *Ironing* event. The annotated groundtruth box is in green, our tracking box in blue. Annotations are focused on visible parts, whereas our tracker frames the full-body.

of the actors. To estimate the impact of this annotation bias, we re-annotated the groundtruth bounding boxes in all *Ironing* sequences from DALY, taking into account the hidden body parts. We then computed the AP@0.5 and AP@0.7 with dense trajectories and CNNs. With these new annotations, we obtain an average precision of 85.2% (45.55% for AP@0.7), whereas average precision with original annotations is 82.52% (29.68% for AP@0.7). The method in [Weinzaepfel et al. \[2016\]](#) obtains an average precision of 78.28% (39.38% for AP@0.7) with the original annotations. The experiment shows that the classical way of annotating people in computer vision datasets, i.e., annotating only visible parts, is not ideal to correctly evaluate our full-body tube extractor. However, our results show significant improvements in action localization in a semi-supervised way. Although we train our part detector for people detection, it can be extended to all objects. The main point is to have a training dataset with annotations for the full object, i.e., taking into account hidden parts. Our part detector can also be used with tracking by detection algorithms [[Saha et al., 2016](#)].

## 5.4 Summary

We proposed a novel full-body tube extraction method based on a new body part detector. This detector is specific to body parts, but regresses to full-body bounding boxes, thus localizing the person(s) in a video. Our tube extraction method tracks several human parts through time, handling occlusions, view point changes, and localizes the full body in any of these challenging scenarios. We showed that using our full-

body tubes significantly improves action localization compared to methods focusing on tubes built from visible parts only, with state-of-the-art results on the new challenging DALY dataset. We also highlighted the importance of generating proposals with keypoints during the training, and the optimal number of connected keypoints to be used. Finally, we discussed the annotation bias in action recognition datasets, showing that we significantly improve the performance by re-annotating *ironing* videos on DALY dataset. The new *ironing* annotations, which frame the full-body boxes, result in a boost of 16% for R@0.7.

# Chapter 6

## Conclusion

### Contents

---

6.1	Summary of contributions . . . . .	97
6.1.1	Learning action classification from web videos	97
6.1.2	Human action localization . . . . .	98
6.2	Perspectives for future research . . . . .	99
6.2.1	Learning action classification from web videos	99
6.2.2	Human action localization . . . . .	101

---

In this thesis, we focus on two tasks related to video understanding: learning actions from web videos and human action localization in real-world videos. This chapter is organized as follows. We summarize the contributions of the thesis on learning from web videos in Section 6.1.1, and on human action localization in Section 6.1.2. We conclude the dissertation by proposing directions for future research in Section 6.2.

### 6.1 Summary of contributions

#### 6.1.1 Learning action classification from web videos

In Part I, we present a novel approach for action classification, given only a textual description of the event. Our approach relies on textual query expansion specifically designed for actions, which allows us to collect significantly more videos than using only the event name. We show that our query expansion significantly improves the performance of our approach, and permits to adapt to intra-class variation.

**Query expansion.** Given an event name and a textual description, we propose a query expansion, which allows us to download a large pool of relevant web videos, covering intra-class variation. Our query expansion extracts from the textual description words associated to the event. These words are combined with the original query, i.e., the name of the event. At the end of the query expansion, a rich set of event-related queries is available, which allows for the creation of the new event dataset with web resources. We show that the query expansion step improves action classification performance on both TrecVid11 and TrecVid13 datasets.

**Pruning algorithm.** We propose a new two-step approach to prune the set of YouTube videos collected with our queries. In the first step, we perform pruning with text data to select an initial set of relevant examples. Tf-idf features are used, as we show that, combined with a dedicated pre-processing step, they show state-of-the-art results for text classification. A second step prunes data based on visual clues and similarities. The pruning step creates a reliable training dataset of videos sharing semantic and visual content. We show that combining textual and video information improves the performance for event classification. We show state-of-the-art results on TrecVid MED 2011 and 2013 in the zero-shot learning framework.

### 6.1.2 Human action localization

In Part II, we propose a novel full-body tube extraction method based on a new body part detector. This detector is specific to body parts, but regresses to full-body bounding boxes, thus localizes the person(s) in a video. To this end, we build, from the annotated keypoints of training images, pseudo-groundtruth full-body bounding boxes, even in the case where they fall outside the frame. Then, parts are defined without supervision from the set of training part proposals. During training, we show that learning our network with a constant full-body ratio (FBR, i.e. the proportion of full body proposals in the positive proposals) enables to detect simultaneously full body and parts. To the best of our knowledge, amodal completion has never been used in localizing actions in videos. Based on our novel part detector, our tube extraction method tracks several human parts through time, handling occlusions, viewpoint changes, and localizes the full body in any of these challenging scenarios. We show that using our full-body tubes significantly improves action localization compared to methods focusing on tubes built from visible parts

only, with state-of-the art results on the new challenging DALY dataset. More specifically, we show that using amodal completion information improves the accuracy of an action localization method.

## 6.2 Perspectives for future research

### 6.2.1 Learning action classification from web videos

**Textual and visual common semantic space.** Our webly approach can benefit from recent advances in zero-shot learning techniques. [Hussein et al. \[2017\]](#) use an approach similar to our webly-supervised method, i.e., given a text describing a novel event, the goal is to find relevant videos. First, both text and video contents are separately embedded into a dedicated CNN. A unified semantic space and metric are then learned on top of CNN features. For an unseen class during test time, textual description is first embedded into the semantic space. Video content is then sorted with respect to their distance with the event’s textual embedding. This method and our work can be efficiently combined. The training set created by our approach can provide relevant content for iteratively learning the metric space described in [[Hussein et al., 2017](#)]. For example, given video exemplars of events *removing drywall* and *fit wall tiles*, exemplars of the new event *renovate home* are detected based on the probability distribution over the predefined events. Our webly-supervised approach would provide relevant samples for *removing drywall* and *fit wall times*, which then would be used to learn the metric space defined in [[Hussein et al., 2017](#)]. Moreover, the semantic space metric can be iteratively refined by providing samples for other categories. For instance, we can compare samples sorted by the algorithm with samples sorted by our pruning algorithm for the event *renovate home*, and adjusting the metric accordingly. The semantic space will be adapted to intra-class variation and inter-class confusion by forcing videos of the same event to be closed in the semantic space, e.g., one “changing car tire” video and one “changing bike tire” video. Our pruning algorithm can also benefit from the metric space by using CNN features from [[Hussein et al., 2017](#)]. Our method can go further for event recognition by proposing an end-to-end learning algorithm, which leverages web textual and web visual content simultaneously, by first pruning relevant videos, and then using textual and visual content to train iteratively an embedding metric space.



**The semantics of an action.** One of the most exciting challenges in action recognition is to define the semantic meaning of an action. An action is ambiguous. For instance, if two persons are talking and walking in the corridor, some people would annotate the sequence as “two persons are walking”, whereas other people would consider “two persons are talking” as the correct label. Moreover, some actions are more human attention focused than other ones. For instance, if the two persons are walking in the corridor and arguing, everybody would define the sequence as “two persons are arguing”. Understanding the nature of an action would be a tremendous breakthrough in action recognition. Computer vision will benefit from the future advances in Natural Language Processing (NLP). Bridges between the two communities are relatively easy to build since a unified framework, i.e., deep-learning methods, shows state-of-the-art results in both domains. For instance, image captioning has shown very promising results by using both NLP and image processing techniques. For future work, a Generative Adversarial Network (GAN) for actions may be considered. GAN is a deep-learning technique originally designed for images [Goodfellow et al., 2014]. The network learns to produce an image of a particular object, e.g., bird. Some works extend the method to video content [Vondrick et al., 2016], but we are still far from generating realistic videos. An action-GAN would generate action videos indistinguishable from web action videos pruned by our system. GAN videos would follow the probabilistic distribution of web videos. Thus, our automatic training dataset generation method would be extended to create an unsupervised method of video generation.

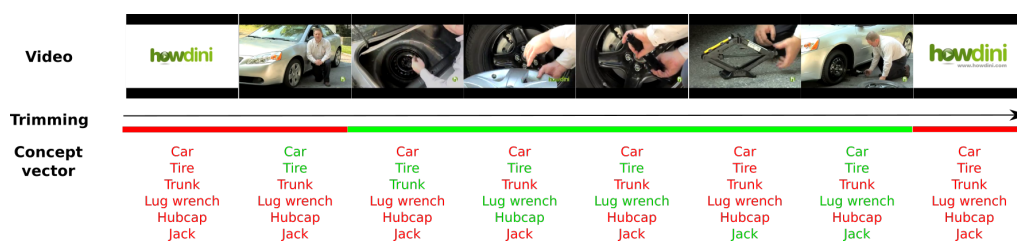


Figure 6.1 – Illustration of our work adapted to temporal trimming. While an event is represented by a single concept vector (mean vector) in [Chesneau et al., 2017a], an ordered set of concept vectors will extend our work to temporal localization.

**Temporal trimming with the pruning algorithm.** In our work we have not considered to trim the downloaded videos. Trimming videos is a

crucial step since video content often shows several actions in addition to the specific events used to retrieve the videos, e.g., an introduction where a person gives a description of the action or the problem, a person performing the action of interest, and a conclusion. Our work can be adapted to temporal localization. Our pruning algorithm extracts semantic content similarities from a set of deep-learning video features, which are obtained by a mean-pooling of frame-level features. An event is then represented as a mean vector of pooled features. In a frame level approach, an ordered set of mean vectors would represent an event. Long-Short Term Memory (LSTM) can be used to describe the temporal structure of an action, by looking for long-term correlations. A temporal trimming function would be built upon the LSTM structure. LSTM has been successfully applied to video captioning [Yamato et al., 1992, Brand et al., 1997]. An event can thus be viewed as an ordered sequence of semantic content, see Figure 6.1 for an illustration. The pruning algorithm would also prune from relevant videos the irrelevant sequences of frames in an unsupervised way.

## 6.2.2 Human action localization

**Temporal localization.** In our work, we have not considered the task of trimming actions in videos, i.e., temporally localizing actions. To this end, Weinzaepfel et al. [2016] threshold action scores. Below the threshold, Weinzaepfel et al. [2016] consider that no action is occurring in the sequence. Experimental results, in particular on the DALY dataset, show that many failure cases are due to inaccurate temporal detections: in particular, long actions tend to be split into multiple small detections. The action descriptors employed in our work, i.e., dense trajectories and appearance and motion CNNs, are not discriminative enough for temporal detection. Moreover, for dense trajectories, the Fisher vector aggregates trajectories of a certain length. One way to improve temporal detection would be to use the spatial pyramid match method extended to the temporal dimension, as done in [Oneata et al., 2013]. The perspective described in Section 6.2.1 could also be applied here, i.e., using a LSTM to trim videos with a frame-based approach. Many recent works [Buch et al., 2017, Gao et al., 2017, Yang et al., 2017] propose some directions for this challenging task. Hou et al. [2017] propose to revisit the classic idea of modeling an action as a sequence of sub-actions, called actoms in Gaidon et al. [2013]. A new approach is to anticipate both temporally [Gao et al., 2017] and spatially [Yang et al., 2017] the occurring actions.

Gao et al. [2017] propose to capture time correlations in videos by leveraging the efficiency of a LSTM. Yang et al. [2017] build a Location Anticipation Network (LAN) in order to predict the bounding boxes in the future. LAN is trained by comparing two frames separated by a constant time step. In Buch et al. [2017], two memory modules are built on top of visual encoder and output temporal action detections. Memory and anticipation layers show promising results. Our work could be efficiently extended by adding an additional time-specific layer on top of our part detector. For instance, we could leverage the accuracy of our part detector with an anticipation layer in order to predict the full-body location in the future. We would also have to design a network which can be used on large-scale datasets like DALY.

**Tube network and action proposal.** Our tube extraction method is built upon our part detector and combines human part appearance information and detection. An end-to-end learning method for tube proposal, based on our part detector, could efficiently extend our work. An amodal human tube proposal network would take as input a video and would extract full-body human tubes with action scores, see Figure 6.2 as a proposal for the network. Saha et al. [2016] show that an action proposal network is well-adapted for action localization. The amodal human tube proposal network would be designed to be scalable to large-scale datasets like DALY, whose videos last at average more than three minutes.

**Going further with amodal completion for actions.** We show that inferring the full body box within video frames improves performance for action localization. We can extend our work by building a part detector for every type of object. We would have to tackle the challenge of choosing the correct number of part classes for an object. For instance, we may not need 20 parts for a chair. This is important because the number of classes would dramatically increase with the number of detected objects. The network would need an appropriate training dataset, with full-object annotations. We believe that our work combined with depth estimation will lead to better scene understanding in 3D [Xia et al., 2012, Ni et al., 2013].

**Action localization and objects.** Understanding action context is still an open problem in the computer vision community. Actions are performed on purpose. Sigurdsson et al. [2017] propose to detect and localize actions with additional cues: objects of interest, e.g., a cup for *Drinking a cup*

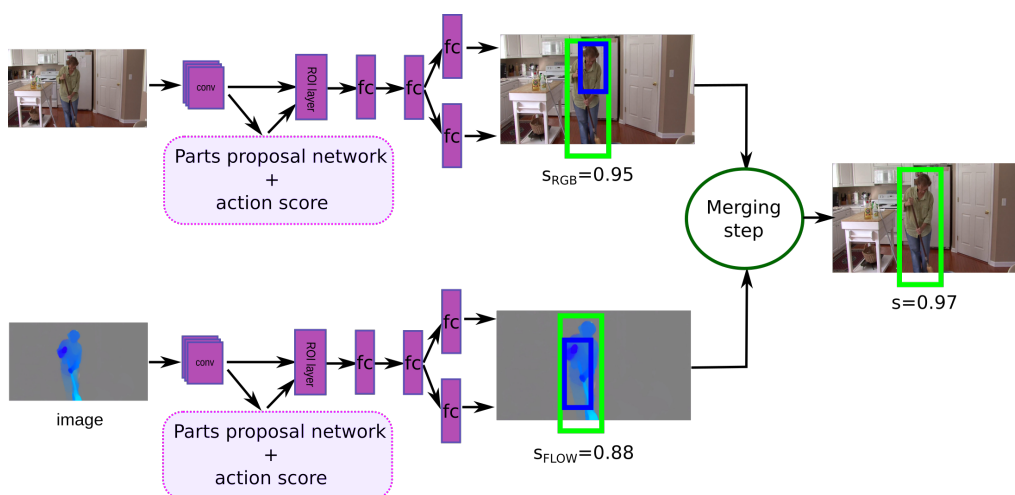


Figure 6.2 – Overview of an amodal tube proposal network.

of tea, the scene where the action is performed, e.g., a dining room. We can go further in action localization, by incorporating context information, interactions between people and objects. Kalogeiton et al. [2017b] show promising results by combining actions and active subjects, e.g., a person, a car, an animal. Using audio and speech information would also help to understand the purpose of an action, and temporally trim them into sub-sequences.

# **Appendix A**

## **Text classification on Reuters-21758**

### **A.1 Per-class results for pre-processing steps and features**

Name	Stemming function	Grammar analysis	Features	earn	acq	money-fx	grain	crude	trade	interest	ship	wheat	corn	mean
"POS+WordNet"	WordNet	Yes	tf-idf	97.88	95.41	78.21	94.63	86.77	80.34	78.63	85.39	84.51	87.5	86.90
"POS+WordNet"	WordNet	Yes	bow	98.07	96.24	78.21	89.93	85.19	75.21	79.39	86.52	84.51	87.5	86.07
"WordNet"	WordNet	No	tf-idf	98.53	96.94	79.33	93.29	86.77	80.34	83.21	85.39	85.92	91.07	88.07
"WordNet"	WordNet	No	bow	97.89	96.38	83.8	87.92	87.3	81.2	80.92	86.52	85.92	89.29	87.71
"Porter"	Porter	No	tf-idf	98.24	96.38	81.12	89.83	87.15	80.87	79.12	84.87	85.09	89.20	87.18
"Porter"	Porter	No	bow	98.16	96.38	82.12	89.93	86.24	79.49	78.63	85.39	85.92	85.71	86.79

Table A.1 – Analysis of three preprocessing methods. We compare the Porter algorithm function and the Wordnet morphological functions. We also compare the impact of grammatical analysis. Two features are tested : bag-of-words (bow), and tf-idf features. Results are shown for a linear SVM.

method	kernel	kernel parameters	earn	acq	money-fx	grain	crude	trade	interest	ship	wheat	corn	mean
bow	Linear	-	98.07	96.24	78.21	89.93	85.19	75.21	79.39	86.52	84.51	87.5	86.07
bow	RBF	$\gamma = 1$	98.16	96.52	79.89	87.92	88.89	80.34	78.63	86.52	81.69	82.14	86.07
bow	poly	$degree = 2$	98.07	95.97	78.77	87.25	87.83	78.63	83.21	83.15	80.28	83.93	85.70
tf-idf	linear	-	97.88	95.41	78.21	94.63	86.77	80.34	78.63	85.39	84.51	87.5	86.9
tf-idf	RBF	$\gamma = 1$	97.98	95.83	75.42	93.96	87.83	79.49	80.15	87.64	84.51	89.29	87.21
tf-idf	poly	$degree = 2, c = 0$	98.07	96.38	72.07	92.62	87.83	79.49	78.63	82.02	87.32	89.29	86.37
bigram	linear	-	97.15	93.05	79.89	76.51	77.78	71.79	77.86	55.06	76.06	67.86	77.30
bigram	RBF	$\gamma = 1$	96.96	92.91	79.33	74.5	77.25	70.09	74.05	58.43	69.01	66.07	75.86
single word + bigram	linear	-	98.25	96.38	80.45	91.95	86.24	79.49	77.86	87.64	85.92	89.29	87.34
single word + bigram	RBF	$\gamma = 1$	98.25	96.38	76.54	89.93	86.77	81.2	79.39	87.64	85.92	87.5	86.95
Word2Vec+Fisher	RBF	$\gamma = 1$	98.16	96.11	77.65	85.23	87.83	78.63	78.63	85.39	77.46	71.43	83.65
SSK kernel	-	-	93.88	62.50	82.01	97.79	79.87	77.10	78.77	75.28	72.65	74.65	79.45

Table A.2 – Comparison of bag-of-words, tf-idf, bigram, Word2Vec+FV features, and SSK kernel.



## A.2 RBF kernel results on Reuters-21578

$\gamma$ value	earn	acq	money-fx	grain	crude	trade	interest	ship	wheat	corn	mean
0.2	97.98	95.69	77.65	94.63	86.77	80.34	77.86	85.39	85.92	87.5	86.97
0.6	97.98	95.55	76.54	94.63	87.83	81.2	80.92	86.52	84.51	91.07	87.67
0.8	97.98	95.83	75.42	93.96	87.83	79.49	80.15	86.52	85.92	91.07	87.41
1	97.98	95.83	75.42	93.96	87.83	79.49	80.15	87.64	84.51	89.29	87.21
1.2	98.07	95.69	74.86	92.62	87.3	80.34	80.15	86.52	84.51	91.07	87.11
1.4	97.98	95.55	74.86	91.95	87.3	82.05	79.39	85.39	84.51	89.29	86.82
1.6	97.98	95.55	74.3	92.62	87.3	82.91	79.39	85.39	84.51	91.07	87.10
1.8	98.07	96.11	77.65	91.95	86.24	79.49	80.15	85.39	84.51	89.29	86.88
2	97.98	95.97	75.42	91.28	86.24	80.34	77.86	84.27	84.51	89.29	86.31
5	96.87	94.99	73.18	84.56	86.24	81.2	79.39	82.02	74.65	78.57	83.16

Table A.3 – Influence of gamma parameter for RBF kernel with tf-idf features.

$\gamma$ value	earn	acq	money-fx	grain	crude	trade	interest	ship	wheat	corn	mean
0.05	98.34	96.66	80.45	91.95	85.71	79.49	77.86	87.64	85.92	89.29	87.33
0.1	98.16	96.24	79.89	91.95	86.24	79.49	77.86	88.76	87.32	89.29	87.52
0.2	98.34	96.94	77.65	91.95	88.89	81.2	78.63	87.64	85.92	89.29	87.64
0.6	98.34	96.52	77.09	91.28	88.89	81.2	78.63	87.64	87.32	89.29	87.62
0.8	98.25	96.38	76.54	90.6	88.89	82.05	78.63	87.64	85.92	87.5	87.24
1	98.25	96.38	76.54	89.93	86.77	81.2	78.63	87.64	85.92	87.5	86.87
1.2	98.16	96.11	80.45	89.26	86.77	82.05	78.63	87.64	85.92	85.71	87.07
1.4	98.16	95.83	81.01	88.59	85.71	82.05	78.63	86.52	84.51	87.5	86.85
1.6	98.16	95.83	79.89	87.25	85.71	80.34	79.39	84.27	83.1	85.71	85.96
1.8	98.16	95.55	80.45	87.25	86.24	79.49	79.39	84.27	83.1	85.71	85.96
2	98.07	95.83	78.21	87.25	86.24	78.63	78.63	84.27	83.1	83.93	85.41
5	97.61	95.13	75.98	83.78	86.24	0.0	77.27	80.9	71.83	83.93	75.26

Table A.4 – Influence of gamma parameter for RBF kernel with bigrams features.

$\gamma$ value	earn	acq	money-fx	grain	crude	trade	interest	ship	wheat	corn	mean
0.05	97.98	95.83	81.56	90.6	86.77	78.63	81.68	86.52	84.51	87.5	87.15
0.1	98.34	95.55	82.68	88.59	85.19	79.49	80.15	87.64	84.51	85.71	86.78
0.2	98.07	96.11	82.68	88.59	86.77	78.63	80.92	86.52	81.69	89.29	86.92
0.6	98.25	95.97	81.56	87.92	88.36	79.49	80.92	87.64	80.28	85.71	86.61
0.8	98.25	95.97	81.56	87.92	89.42	79.49	82.44	86.52	80.28	83.93	86.57
1	98.16	96.52	79.89	87.92	88.89	80.34	78.63	86.52	81.69	82.14	86.07
1.2	98.16	96.24	79.95	87.92	88.89	79.49	78.63	85.39	81.69	82.14	86.02
1.4	98.16	96.11	80.45	87.92	88.89	79.49	78.63	85.39	81.69	83.93	86.06
1.6	98.16	95.97	78.21	86.58	88.36	78.63	78.63	84.27	81.69	82.14	85.26
1.8	98.16	95.55	77.65	85.91	87.83	78.63	81.68	83.15	80.28	80.36	84.92
2	98.25	95.41	77.65	85.14	88.36	78.63	81.68	83.15	80.28	78.57	84.71
5	96.6	91.52	73.89	81.21	84.66	0.0	75.57	78.65	74.65	73.21	72.99

Table A.5 – Influence of gamma parameter for RBF kernel with bag-of-words features.

$\gamma$ value	earn	acq	money-fx	grain	crude	trade	interest	ship	wheat	corn	mean
0.05	97.42	95.69	79.33	85.23	87.3	79.49	75.57	87.64	81.69	75.0	84.43
0.1	97.24	95.69	79.33	86.58	88.89	80.34	77.1	88.76	78.87	73.21	84.60
0.2	97.79	95.97	79.33	86.58	88.36	79.49	76.34	86.52	80.28	78.57	84.92
0.4	97.88	95.83	79.89	87.25	87.83	78.63	77.1	85.39	80.28	75.0	84.50
0.6	98.07	95.97	78.77	84.56	88.89	77.78	76.34	87.64	80.28	73.21	84.15
0.8	98.07	96.11	78.21	85.91	88.89	78.63	79.39	87.64	78.87	71.43	84.31
1	98.16	96.11	77.65	85.23	87.83	78.63	78.63	85.39	77.46	71.43	83.65
1.2	98.16	95.83	77.65	84.56	88.36	78.63	77.1	86.52	77.46	71.43	83.57
1.4	98.16	95.69	78.77	83.89	87.83	77.78	77.86	83.15	77.46	71.43	83.20
1.6	98.16	95.69	75.98	82.55	86.77	75.21	77.86	85.39	77.46	71.43	82.65
1.8	98.25	95.69	76.54	81.88	85.71	76.07	77.86	82.02	77.46	69.64	82.11
2	97.98	95.55	76.54	81.88	84.66	76.07	77.1	84.27	74.65	67.86	81.65
5	96.87	91.38	73.74	80.54	85.19	73.5	73.28	78.65	69.01	57.14	77.93

Table A.6 – Influence of gamma parameter (RBF kernel) for Word2Vec Fisher Vector.

### A.3 Variation of the number Gaussians and PCA dimensions for Word2Vec features

K (GMM)	PCA	earn	acq	money-fx	grain	crude	trade	interest	ship	wheat	corn	mean
32	10	96.32	93.6	70.39	77.85	82.54	64.1	74.05	61.8	69.01	58.93	74.86
32	20	96.96	94.58	73.74	77.18	85.19	70.09	72.52	75.28	69.01	57.14	77.17
32	30	97.15	95.41	78.21	76.51	83.6	76.92	77.1	77.53	70.42	57.14	79.00
32	50	96.69	95.41	74.86	85.91	87.3	74.36	80.15	82.02	77.46	76.79	83.01
32	100	98.07	95.41	75.98	81.21	87.3	72.65	79.39	82.02	74.65	71.43	81.81
32	150	98.16	95.69	77.09	83.89	87.83	76.92	75.57	85.39	78.87	75.0	83.44
32	200	97.52	95.55	77.65	84.56	87.83	80.34	77.86	85.39	78.87	76.79	84.24
32	250	97.61	95.27	77.65	85.91	88.89	77.78	77.86	87.64	81.69	76.79	84.71
64	10	97.06	94.99	74.86	78.52	83.6	67.52	76.34	68.54	71.83	58.93	77.22
64	20	97.15	95.13	75.98	77.18	87.83	78.63	71.76	75.28	67.61	55.36	78.20
64	30	97.15	94.85	76.54	85.23	86.24	76.92	74.81	79.78	77.46	67.86	81.68
64	50	97.24	95.41	77.65	79.87	85.71	76.07	75.57	78.65	70.42	69.64	80.62
64	100	97.52	94.3	76.54	77.18	84.66	75.21	74.81	74.16	67.61	62.5	78.45
64	150	97.7	95.41	78.21	85.23	87.3	76.07	75.57	83.15	77.46	69.64	82.58
64	200	97.06	95.41	79.33	86.58	87.3	78.63	78.63	77.53	77.46	78.57	83.65
64	250	97.98	94.58	72.07	79.87	87.3	75.21	76.34	82.02	66.2	60.71	79.22
86	10	97.24	93.74	75.42	79.19	84.13	66.67	75.57	59.55	67.61	62.5	76.17
86	20	97.24	95.69	77.65	85.91	87.3	74.36	74.81	73.03	71.83	64.29	80.21
86	30	97.88	95.97	71.51	79.19	79.08	69.23	74.81	73.03	70.42	57.14	76.82
86	50	97.06	95.13	73.74	71.81	82.01	76.07	70.99	75.28	64.79	55.36	76.22
86	100	97.42	95.69	75.98	82.55	88.36	71.79	79.39	82.02	76.06	69.64	81.89
86	150	96.6	95.55	77.65	85.23	87.3	69.23	74.05	83.15	80.28	76.79	82.58
86	200	97.79	94.85	73.18	84.56	87.83	75.21	74.05	85.39	77.46	71.43	82.18
86	250	97.61	94.02	75.98	79.87	87.83	75.21	70.99	83.15	80.28	57.14	80.21
100	10	97.61	94.16	72.63	79.87	86.77	68.38	77.86	61.8	69.01	57.14	76.52
100	20	97.06	94.71	77.65	86.58	87.83	72.65	71.76	74.16	80.28	75.0	81.77
100	30	96.32	95.55	77.09	82.55	87.83	74.36	74.05	76.4	80.28	75.0	81.94
100	50	97.61	95.69	78.21	81.88	86.77	71.79	74.05	80.9	71.83	62.5	80.12
100	100	97.7	95.13	72.62	79.87	82.01	74.36	75.57	77.53	73.24	57.14	78.52
100	150	96.78	95.55	75.42	83.89	86.77	77.78	74.05	85.39	73.24	69.64	81.85
100	200	97.06	94.58	77.65	83.22	88.36	76.92	75.57	84.27	78.87	76.79	83.33
100	250	97.98	95.41	75.42	85.23	84.66	78.63	75.57	79.78	80.28	76.79	82.98

Table A.7 – Influence of numbers of Gaussian and PCA for Word2Vec Fisher Vector.

# Bibliography

- Oxford english dictionary online, 2007. [17](#)
- M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014. [81](#)
- M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, 2011. [22](#)
- L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. In *SIGIR*, 1998. [29](#)
- H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *ECCV*, 2006. [20](#)
- A. Binder, W. Samek, K.-R. Müller, and M. Kawanabe. Machine learning for visual concept recognition and ranking for images. In *Towards the Internet of Services: The THESEUS Research Program*. 2014. [42](#)
- D. C. Blair and M. E. Maron. An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Com. ACM*, 1985. [17](#)
- M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *CVPR*, 1997. [101](#)
- A. Bregman. Auditory scene analysis: The perceptual organization of sound. In *MIT Press*, 1990. [63](#)
- T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. [20](#)
- T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004. [85](#)

- S. Buch, V. Escorcia, B. Ghanem, L. Fei-Fei, and J. C. Nieble. End-to-end, single-stream temporal action detection in untrimmed videos. *BMVC*, 2017. [101](#), [102](#)
- L. Cao, Z. Liu, and T. Huang. Cross-dataset action detection. In *CVPR*, 2010. [8](#), [64](#)
- J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *CVPR*, 2017. [5](#)
- C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011. [32](#), [34](#), [48](#)
- J. Chen, Y. Cui, G. Ye, D. Liu, and S.-F. Chang. Event-driven semantic concept discovery by exploiting weakly tagged internet images. In *ICMR*, 2014a. [6](#), [24](#), [55](#), [57](#)
- L. Chen, L. Duan, and D. Xu. Event recognition in videos by learning from heterogeneous web sources. In *CVPR*, 2013. [6](#), [23](#), [27](#), [28](#)
- X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014b. [62](#)
- N. Chesneau, K. Alahari, and C. Schmid. Learning from web events for event classification. *TCSVT*, 2017a. [11](#), [26](#), [100](#)
- N. Chesneau, G. Rogez, K. Alahari, and C. Schmid. Detecting parts for action localization. *BMVC*, 2017b. [12](#)
- C. E. Crangle, A. Zbyslaw, J. M. Cherry, and E. L. Hong. Concept extraction and synonymy management for biomedical information retrieval. In *TREC*, 2004. [15](#)
- 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*. Prague, 2004. [21](#)
- N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. [19](#)
- N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006. [5](#), [20](#)

- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 83
- S. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014. 24
- J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 5, 22
- L. Duan, D. Xu, I. W.-H. Tsang, and J. Luo. Visual event recognition in videos by learning from web data. *PAMI*, 2012. 6, 23, 27, 28
- I. Endres, K. J. Shih, J. Jiaa, and D. Hoiem. Learning collections of part models for object recognition. In *CVPR*, 2013. 62
- D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, 2014. 61
- M. Everingham, L. VanGool, C. Williams, W. J., and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge 2010. *IJCV*, 2010. 69
- C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 5
- P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2010. 9, 21, 62, 65
- A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal Localization of Actions with Actoms. *PAMI*, 2013. 21, 64, 101
- C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann. Devnet: A deep event network for multimedia event detection and evidence recounting. In *CVPR*, 2015. 23, 27
- C. Gan, C. Sun, L. Duan, and B. Gong. Webly-supervised video recognition by mutually voting for relevant web images and web video frames. In *ECCV*, 2016a. 6, 23, 24, 54, 55, 57
- C. Gan, T. Yao, K. Yang, Y. Yang, and T. Mei. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. In *CVPR*, 2016b. 6, 24, 55, 57
- J. Gao, Z. Yang, and R. Nevatia. Red: Reinforced encoder-decoder networks for action anticipation. *BMVC*, 2017. 101, 102

- W. Gerbino and D. Salmaso. The effect of amodal completion on visual matching. *Acta psychologica*, 1987. [63](#)
- S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware CNN model. In *ICCV*, 2015. [61](#)
- R. Girshick. Fast R-CNN. *ICCV*, 2015. [61](#), [68](#)
- R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. [61](#), [62](#), [79](#)
- G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. [67](#), [68](#), [80](#), [86](#)
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. [100](#)
- A. Habibian, T. E. J. Mensink, and C. G. M. Snoek. Composite concept discovery for zero-shot video event detection. In *ICMR*, 2014. [5](#), [6](#), [55](#), [57](#)
- A. Habibian, T. Mensink, and C. G. M. Snoek. Video2vec embeddings recognize events when examples are scarce. *PAMI*, 2017.
- P. Hanks. Collins dictionary of the English language. *Collins*, 1986. [17](#)
- S. M. Harabagiu, G. A. Miller, and D. I. Moldovan. In *SIGLEX*, 1999. [16](#), [30](#)
- S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. [72](#)
- Z. S. Harris. Distributional structure. *Word*, 1954. [17](#)
- A. G. Hauptmann and M. A. Smith. Text, speech and vision for video segmentation: The informedia project. In *AAAI Fall Symposium*, 1995. [23](#)
- K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *PAMI*, 2015. [61](#)
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016. [22](#)



- D. Hiemstra. A probabilistic justification for using tf-idf term weighting in information retrieval. *International Journal on Digital Libraries*, 2000. [42](#)
- S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997. [22](#)
- R. Hou, R. Suthankar, and M. Shah. Real-time temporal action localization in untrimmed videos by sub-action discovery. *BMVC*, 2017. [101](#)
- L. Huang, Y. Yang, Y. Deng, and Y. Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015. [61](#)
- N. Hussein, E. Gavves, and A. W. Smeulders. Unified embedding and metric learning for zero-exemplar event detection. *CVPR*, 2017. [6](#), [25](#), [55](#), [99](#)
- N. Ikizler-Cinbis, R. Cinbis, and S. Sclaroff. Learning actions from the web. In *ICCV*, 2009. [6](#), [23](#), [27](#), [28](#)
- M. Jain, J. van Gemert, H. Jégou, P. Bouthemy, and C. Snoek. Action localization by tubelets from motion. In *CVPR*, 2014. [8](#), [9](#), [66](#)
- H. Jegou, F. Perronnin, M. Douze, J. Sánchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *PAMI*, 2012. [21](#)
- H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013a. [80](#)
- H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013b. [12](#)
- S. Ji, W. Xu, M. Yang, and K. Yu. 3D convolutional neural networks for human action recognition. *PAMI*, 2013. [5](#), [22](#)
- L. Jiang, T. Mitamura, S.-I. Yu, and A. G. Hauptmann. Zero-example event search using multimodal pseudo relevance feedback. In *ICMR*, 2014. [55](#), [57](#)
- L. Jiang, S.-I. Yu, D. Meng, T. Mitamura, and A. G. Hauptmann. Bridging the ultimate semantic gap: A semantic search engine for internet videos. In *ICMR*, 2015a. [25](#), [29](#), [55](#), [57](#)

- L. Jiang, S.-I. Yu, D. Meng, Y. Yang, T. Mitamura, and A. G. Hauptmann. Fast and accurate content-based semantic search in 100M internet videos. In *ACMM*, 2015b. [25](#), [29](#), [55](#), [57](#)
- T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML*, 1998. [29](#), [31](#), [35](#)
- S. Johnson and M. Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *BMVC*, 2010. [81](#)
- C. A. A. Kaestner. Support vector machines and kernel functions for text processing. *Revista de Informática Teórica e Aplicada*, 2013. [33](#), [34](#), [35](#)
- V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action Tubelet Detector for Spatio-Temporal Action Localization. In *ICCV*, Venice, Italy, 2017a. [9](#)
- V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Joint learning of object and action detectors. In *ICCV*, 2017b. [103](#)
- G. Kanizsa. Subjective contours. *Scientific American*, 1976. [63](#)
- G. Kanizsa and A. Chambolle. *La grammaire du voir: essais sur la perception*. Diderot Editeur arts et sciences, 1997. [63](#)
- A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Amodal completion and size constancy in natural scenes. *ICCV*, 2015. [8](#), [9](#), [63](#), [67](#)
- A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. [5](#), [22](#), [23](#), [27](#)
- Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *ICCV*, 2005. [23](#), [27](#)
- A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3D-gradients. In *BMVC*, 2008. [5](#), [20](#)
- A. Kläser, M. Marszalek, C. Schmid, and A. Zisserman. Human Focused Action Localization in Video. In *International Workshop on Sign, Gesture, and Activity*, 2010. [8](#), [65](#)
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. [21](#), [29](#), [42](#)

- W. Kuo, B. Hariharan, and J. Malik. Deepbox: Learning objectness with convolutional networks. In *ICCV*, 2015. 61
- C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *PAMI*, 2014. 5
- T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *ICCV*, 2011. 8, 65
- I. Laptev. On space-time interest points. *IJCV*, 2005. 19
- I. Laptev. *Modeling and visual recognition of human actions and interactions*. Habilitation à diriger des recherches (HDR), Ecole Normale Supérieure de Paris, 2013. 2
- I. Laptev and P. Pérez. Retrieving actions in movies. In *ICCV*, 2007. 8, 23, 27, 64
- I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 20, 21, 23, 27
- S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 5, 21
- T. Leung, Y. Song, and J. Zhang. Handling label noise in video classification via multiple instance learning. In *ICCV*, 2011. 6, 23
- D. D. Lewis. Reuters-21578 text categorization test collection, distribution 1.0. <http://www.research.att.com/~lewis/reuters21578.html>, 1997. 33
- 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. 20
- Y. Li, K. He, J. Sun, et al. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 61
- J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos “in the wild”. In *CVPR*, 2009. 23, 27
- J. Liu, Q. Yu, O. Javed, S. Ali, A. Tamrakar, A. Divakaran, H. Cheng, and H. Sawhney. Video event recognition using concept attributes. In *WACV*, 2013. 6, 24

- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. [61](#)
- H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *JMLR*, 2002. [19](#), [29](#)
- E. Loper and S. Bird. NLTK: The natural language toolkit. In *ACL Workshop*, 2002. [34](#), [46](#)
- H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM J. Res. Dev.*, 1957. [17](#)
- M. Marian Puscas, E. Sangineto, D. Culibrk, and N. Sebe. Unsupervised tube extraction using transductive learning and dense trajectories. In *ICCV*, 2015. [9](#), [66](#)
- S. Masnou and J.-M. Morel. Level lines based disocclusion. In *ICIP*, 1998. [63](#)
- P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *ICCV Workshops*, 2009. [20](#)
- A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI*, 1998. [29](#)
- C. T. Meadow. *Text Information Retrieval Systems*. Academic Press, Inc., 1992. [17](#)
- T. Mensink, E. Gavves, and C. Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *CVPR*, 2014. [5](#), [6](#), [24](#)
- T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *ICLR*, 2013a. [19](#)
- T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013b. [34](#), [39](#)
- G. A. Miller. Wordnet: A lexical database for English. *Com. ACM*, 1995. [6](#), [40](#)
- D. Modolo and V. Ferrari. Learning semantic part-based models from google images. *PAMI*, 2017. [62](#)

- T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *CVIU*, 2006. 8
- M. M. Murray, D. M. Foxe, D. C. Javitt, and J. J. Foxe. Setting boundaries: brain dynamics of modal and amodal illusory shape completion in humans. *Journal of Neuroscience*, 2004. 63
- J. R. Méndez, E. L. Iglesias, F. Fdez-riverola, F. Díaz, and J. M. Corchado. Tokenising, Stemming and Stopword Removal on Anti-spam Filtering Domain. In *AEPIA*. 2005. 15, 30
- A. D. C. G. Nevill-Manning and B. Oughtred. Partitioning a graph of sequences, structures and abstracts for information retrieval. In *TREC*, 2003. 15
- P. X. Nguyen, G. Rogez, C. Fowlkes, and D. Ramanan. The open world of micro-videos. *arXiv:1603.09439*, 2016. 6
- B. Ni, G. Wang, and P. Moulin. Rgbd-hudaact: A color-depth video database for human daily activity recognition. In *Consumer Depth Cameras for Computer Vision*. 2013. 102
- J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *IJCV*, 2008. 5, 23, 27
- K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 2000. 29
- L. Niu, W. Li, and D. Xu. Visual recognition by learning from web data: A weakly supervised domain generalization approach. In *CVPR*, 2015. 6, 23, 27, 28
- Y. Niwa and Y. Nitta. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. *COLING*, 1994. 18
- D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013. 101
- D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-Temporal Object Detection Proposals. In *ECCV*, 2014. 9, 66
- W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, and C.-C. Loy. Deepid-net: Deformable deep convolutional neural networks for object detection. In *CVPR*, 2015. 61

- P. Over, G. Awad, J. Fiscus, B. Antonishek, M. Michel, A. Smeaton, W. Kraaij, and G. Quénot. TRECVID 2010 – An overview of the goals, tasks, data, evaluation mechanisms, and metrics. TRECVID, 2010. [2](#), [7](#)
- P. Over, J. Fiscus, G. Sanders, M. Michel, G. Awad, A. F. Smeaton, W. Kraaij, and G. Quénot. TRECVID 2013 – An overview of the goals, tasks, data, evaluation mechanisms and metrics. TRECVID, 2013. [29](#)
- M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009. [5](#)
- G. Paltoglou and M. Thelwall. A study of information retrieval weighting schemes for sentiment analysis. In *ACL*, 2010. [31](#)
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *JMLR*, 2011. [34](#)
- X. Peng and C. Schmid. Multi-region two-stream R-CNN for action detection. In *ECCV*, 2016. [9](#), [67](#), [68](#), [80](#), [86](#)
- X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014. [21](#)
- F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. *ECCV*, 2010. [21](#)
- A. Pirkola and E. Leppänen. TREC 2003 genomics track experiments at UTA: Query expansion with predefined high frequency terms. In *TREC*, 2003. [15](#)
- J. Pomikálek and R. Řehůřek. The influence of preprocessing parameters on text categorization. *International Journal of Applied Science, Engineering and Technology*, 2007. [15](#), [30](#)
- M. F. Porter. An algorithm for suffix stripping. *Program*, 1980. [16](#), [30](#)
- A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging, 1996. [16](#)
- R. Rauschenberger and S. Yantis. Masking unveils pre-amodal completion representation in visual search. *Nature*, 2001. [63](#)
- J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016. [61](#)

- J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 61
- S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 61, 67, 68, 74, 75, 82
- S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *PAMI*, 2016. 61
- G. Rogez, P. Weinzaepfel, and C. Schmid. LCR-Net: Localization-Classification-Regression for Human Pose. In *CVPR*, 2017. 81
- S. Saha, G. Singh, M. Sapienza, P. Torr, and F. Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. In *BMVC*, 2016. 67, 68, 80, 86, 95, 102
- G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., 1971. 17
- G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Com. ACM*, 1975. 17
- J. Sanchez, F. Perronnin, T. E. J. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *IJCV*, 2013. 79
- P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. *IJCV*, 2008. 20
- P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using CNN. In *ICLR*, 2014. 61
- S. Shehata, F. Karray, and M. S. Kamel. An efficient concept-based retrieval model for enhancing text retrieval quality. *Knowledge and Information Systems*, 2013. 18
- R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *The computer journal*, 1973. 66
- G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. In *CVPR*, 2017. 102
- K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2014a. 21, 22, 29, 38, 42, 44, 82

- K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014b. [5](#), [22](#), [23](#), [27](#)
- B. Singh, X. Han, Z. Wu, V. I. Morariu, and L. S. Davis. Selecting relevant web trained concepts for automated event retrieval. In *ICCV*, 2015. [6](#), [23](#), [25](#), [27](#), [28](#), [29](#), [54](#), [55](#), [57](#)
- J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. [5](#), [20](#)
- M. Smith and T. Kanade. Video skimming and characterization through the combination of image and language understanding techniques. In *CVPR*, 1997. [23](#)
- Y. Song, M. Zhao, J. Yagnik, and X. Wu. Taxonomic classification for web-based videos. In *CVPR*, 2010. [6](#), [23](#)
- S. Sonnenburg, S. Henschel, C. Widmer, J. Behr, A. Zien, F. d. Bona, A. Binder, C. Gehl, and V. Franc. The shogun machine learning toolbox. *JMLR*, 2010. [34](#)
- C.-M. Tan, Y.-F. Wang, and C.-D. Lee. The use of bigrams to enhance text categorization. *Inf. Process. Manage.*, 2002. [18](#)
- E. H. Taralova, F. De la Torre, and M. Hebert. Motion words for videos. In *ECCV*, 2014. [21](#)
- Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *CVPR*, 2013. [9](#), [64](#)
- S. Tomlinson. Robust, web and genomic retrieval with hummingbird searchserver at TREC 2003. In *TREC*, 2003. [15](#)
- G. Töpper, M. Knuth, and H. Sack. DBpedia ontology enrichment for inconsistency detection. In *ICSS*, 2012. [47](#)
- K. Toutanova and C. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP*, 2000. [30](#), [39](#)
- K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*, 2003. [16](#), [30](#)
- D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015. [5](#), [22](#)



- A. K. Uysal and S. Gunal. The impact of preprocessing on text classification. *Information Processing and Management*, 2014. 15, 30
- J. van Gemert, M. Jain, E. Gati, and C. Snoek. APT: Action localization proposals from dense trajectories. In *BMVC*, 2015. 9, 66
- J. C. Van Gemert, C. J. Veenman, A. W. Smeulders, and J.-M. Geusebroek. Visual word ambiguity. *PAMI*, 2010. 21
- V. N. Vapnik. *Statistical learning theory*. Wiley New York, 1998. 5, 14
- C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NIPS*. 2016. 100
- H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 20
- H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin. Action Recognition by Dense Trajectories. In *CVPR*, 2011. 23, 27, 29, 38, 44
- H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 2013. 19, 20
- Q. Wang and K. Chen. Zero-shot visual recognition via bidirectional latent embedding. *arXiv preprint arXiv:1607.02104*, 2016. 5, 6
- Z. Wang, M. Zhao, Y. Song, S. Kumar, and B. Li. YouTubeCat: Learning to categorize wild web videos. In *CVPR*, 2010. 6, 23, 24
- P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, 2015. 67, 68, 72, 76, 80, 86
- P. Weinzaepfel, X. Martin, and C. Schmid. Human action localization with sparse spatial supervision. *arXiv*, 2016. 3, 4, 12, 67, 68, 69, 72, 80, 81, 84, 86, 87, 88, 89, 90, 91, 95, 101
- S. Wu, S. Bondugula, F. Luisier, X. Zhuang, and P. Natarajan. Zero-shot event detection using multi-modal fusion of weakly supervised concepts. In *CVPR*, 2014. 24, 55, 57
- L. Xia, C.-C. Chen, and J. Aggarwal. View invariant human action recognition using histograms of 3D joints. In *CVPR*, 2012. 102
- Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative CNN video representation for event detection. In *CVPR*, 2015. 23, 27

- J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *CVPR*, 1992. [101](#)
- Z. Yang, J. Gao, and R. Nevatia. Spatio-temporal action detection with cascade proposal and location anticipation. *BMVC*, 2017. [101](#), [102](#)
- G. Ye, Y. Li, H. Xu, D. Liu, and S.-F. Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *ACMM*, 2015. [6](#), [23](#), [24](#), [55](#), [57](#)
- D. Yoo, S. Park, J.-Y. Lee, A. S. Paek, and I. So Kweon. Attentionnet: Aggregating weak directions for accurate object detection. In *ICCV*, 2015. [61](#)
- G. Yu and J. Yuan. Fast action proposals for human action detection and search. In *CVPR*, 2015. [66](#)
- J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, 2009. [8](#), [64](#)
- X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010. [21](#)