Université de Caen Basse-Normandie

Ecole doctorale S.I.M.E.M.

Thèse de doctorat

présentée et soutenue le : 19/12/2012

par

Cristina GARCIA CIFUENTES

pour obtenir le

Doctorat de l'Université de Caen Basse-Normandie

Spécialité : Informatique et Applications

preparée dans le cadre d'une cotutelle internationale de thèse
entre l'Université de Caen Basse-Normandie et
l'University College London (Royaume Uni)

Exploitation de la supervision faible pour l'analyse des vidéos

Directeur de thèse : Frédéric Jurie
Directeur en cotutelle: Gabriel Brostow

Jury

Laurent Heutte, Professeur des Universités, Université de Rouen (rapporteur)
Lourdes Agapito, Professeur, Queen Mary University of London (rapporteur)
Yann Gousseau, Professeur des Universités, Télécom ParisTech
Marc Sturzel, ingénieur de recherche, EADS Innovation Works
Gabriel Brostow, Assistant Professor, University College London (directeur de thèse)
Frédéric Jurie, Professeur des Universités, Université de Caen (directeur de thèse)

I, Cristina García Cifuentes, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

This research deals with the challenging task of video classification, with a particular focus on action recognition, which is essential for a comprehensive understanding of videos. In the typical scenario, there is a list of semantic categories to be modeled, and example clips are given together with their associated category label, indicating *which* action of interests happens in that clip. No information is given about *where* or *when* the action happens, even less about *why* the annotator considered the clip to belong to a sometimes ambiguous category.

Within the framework of the bag-of-words representation of videos, we explore how to leverage such weak labels from three points of view: (i) the use of coherent supervision from the earliest stages of the pipeline; (ii) the combination of heterogeneous features in nature and scale; and (iii) mid-level representations of videos based on regions, so as to increase the ability to discriminate relevant locations in the video.

For the quantization of local features, we propose and evaluate a novel form of supervision to train random forests which explicitly aims at the discriminative power of the resulting bags of words. We show that our forests are better than traditional ones at incorporating contextual elements during quantization, and draw attention to the risk of naive combination of features. We also show that mid-level representations carry complementary information that can improve classification. Moreover, we propose a novel application of video classification to tracking. We show that weak clip labels can be used to successfully classify videos into categories of dynamic models. In this way, we improve tracking by performing classification-based dynamic model selection.

Acknowledgements

I am most grateful to my family and friends for they love, patience and support.

I would also like to thank my supervisors for their technical guidance and their amiability, as well as all the people who took part in the draft reading and defense preparation and contributed useful suggestions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic recognition of actions in videos is a key goal in video analysis and understanding. It is receiving increasing attention from the computer vision community due to its appealing potential applications: video indexing and retrieval, visual surveillance, monitoring systems and human-computer interaction, among others.

A large amount of research has been reported on human action categorization, i.e. automatic vision-based grouping of videos into semantic categories corresponding to human actions, such as running, eating, fighting or shaking hands. Researchers are also interested in the localization of an action in a video, in action comparison, and in other related problems not necessarily involving humans, such as general event recognition and anomaly detection, which enlarges the range of potential applications, but also of specific difficulties to overcome.

Current research shows promising action recognition results on simplistic scenarios, e.g. [7], but only limited success in more realistic situations, e.g. [60]. How to perform robust action recognition in realistic unconstrained videos is still an open question, with implications in the design of appropriate features, video representations, appearance and motion models, and inference methods.

In addition to the understanding of human motion and behavior in realistic videos, it would be useful to accurately classify videos into other kinds of categories, not necessarily semantic. For example, categories that could facilitate further processing and understanding for a particular task, such as tracking or video-based 3D reconstruction. In that sense, reliable classification into task-specific categories could be a universal first step of many other vision systems.

1.1 Challenges

Similarly to the domain of object recognition in still images, an action recognition system should be robust against clutter and variations in viewpoint, spatial scale, and lighting conditions. There are other sources of intra-class variability specific to actions and videos: temporal length and execution rate of the action, subject clothing and general appearance, personal style in the performance, self-occlusion, distracting background motion, camera motion, etc. At the same time, in some situations, there can be low inter-class variability, e.g. actions performed by the same person or with the same type of camera motion may look very similar. Depending on the application, the action to recognize may be an interaction with

other people or objects in the environment.

Even if all these sources of uncertainty could be modeled, action recognition would still be a difficult problem due to the ambiguity of actions. Contrary to the field of the natural language processing –where there is a natural decomposition of speech into building units i.e. sentences, words, phonemes– there is no general agreement about the decomposition of actions into "basic" action units. In the same way, there are various definitions of other related terms such as gesture, activity, behavior and event, involving only vague quantitative notions about their complexity, and temporal or spatial extent. In practice, different approaches adopt the most convenient definition depending on the particular application.

The definition of the actions of interest is also application- or user-dependent. Any computer-based approach to recognition assumes that there is some manner of grouping actions that corresponds to human perceptual or semantic grouping. However, perception is often subjective and the semantics of actions involve qualitative and quantitative considerations, which are difficult to integrate in a single coherent output space of action categories. A particular choice of category taxonomy may seem arbitrary to another user or be useless in a different application –compare for instance ballet movements or olympic diving. This is not specific to actions; category taxonomies can be also be problematic in the domain of scene recognition [139]. Further, particular instances of actions may belong to multiple categories (e.g. composition of two actions happening at the same time) or lie in the fuzzy frontier between two or more of them (e.g. walking fast/running slow). Spatial and temporal localization are also ambiguous. The continuous nature of human actions makes it difficult to say where and when an action starts and ends, especially when there is overlap or a smooth transition between two actions. Due to all these particularities, the action recognition problem has been hard to define formally. They also lead to considerable difficulty in establishing the ground truth, essential to evaluate progress in the field.

1.2 Problem statement and contribution

The problem we address lies in the area of the description and classification of unconstrained video. The high number of degrees of freedom of the human body, the differences in the performance of actions by different people, the wide range of possible recording conditions, and the uncertainty due to the limitations of existing computer vision techniques lead to potentially very heterogeneous, sparse and high dimensional input data.

State-of-the-art approaches depend on a training phase using labeled data. This constitutes a bottleneck, as it is hard and expensive to get sensible ground truth in terms of time and human effort. In the typical scenario, there is a list of semantic categories to be modeled, and example clips are given together with their associated category label, indicating *which* action of interests happens in that clip. No information is given about *where* or *when* the action happens, even less about *why* the annotator considered the clip to belong to a sometimes ambiguous category.

Simple and robust representations like histograms of local features have been proved useful in the context of periodic actions, like walking or waving, but more elaborated representations of videos are needed in order to capture more complex and instantaneous spatiotemporal patterns, corresponding to actions like kissing, answering a phone or getting out of a car.

Still under-explored are alternative target categories, which could be less subjective, for example categories that serve some empirically measurable purpose.

Our proposed contribution is to improve video understanding from the points of view of the target categories and the descriptive power of video representations. We are aware of the availability of only weak annotations, and explore ways of leveraging them approaching the problem from three fronts: (i) the use of coherent supervision from the earliest stages of the pipeline; (ii) the combination of heterogeneous features in nature and scale; and (iii) mid-level representations of videos based on regions, so as to increase the ability to discriminate relevant locations in the video.

Our research revolves the bag-of-words representation of videos, which consist of discretizing (or quantizing) descriptors of local video patches and computing histograms of such quantized descriptors. Each chapter brings into question one aspect of this generic framework, which is described in Chapter 3.

For the quantization of local features, we are interested in ensembles of randomized trees, as they allow for the introduction of supervision in a very flexible way. In Chapter 4 we propose a novel form of supervision to train them, which explicitly aims at the discriminative power of the resulting bags of words, and evaluate it on challenging action recognition datasets.

Chapter 5 addresses the fusion of heterogeneous features, both in nature and scale, with a special focus on the combination of local and contextual features at quantization time. By means of an empirical example, we show that supervision can be a useful resource in this task, which may result disappointing if carried out naively.

The localization of most meaningful areas of the video for a particular classification task is not indicated by the weak labels. This motivates us to explore in Chapter 6 mid-level representations based on regions, which we obtain by randomly sampling regions and describing them with bags of words. We show that this intermediate representation carries complementary information that can improve classification.

Moreover, we propose in Chapter 7 a novel application of video classification to tracking. We show that weak clip labels can be used to successfully classify videos into categories of dynamic models. In this way, we improve tracking by performing classification-based dynamic model selection.

Chapter 2

Related Work

A possible definition of *action recognition* is to analyze a video clip so as to predict whether it contains one action among a list of predefined actions of interest, and *which*. "Does this clip contain the action of running?", or "which action is more likely happening in this video, running, walking or jumping?", are example questions that an action recognition system would aim at answering. In most of the work reviewed in this chapter, such actions of interest are performed by humans, e.g. running, walking, jumping, waving, kissing, hugging, shaking hands, answering a phone, getting out of a car, etc. Therefore, this computer vision task is very often referred to as *human action recognition*. However, as long as the human assumption is not hard-coded in the system, nothing prevents from using the approaches presented here to analyze the actions or motion of, e.g., animals. For example, if the application field is horse dressage, one could use such systems to recognize different kinds of horse movements, like different gaits, leg-yield, shoulder-in, etc.

In the typical supervised scenario, some example clips of the actions of interest are available "offline", i.e. at training time, so that statistical models of those actions can be learnt from them. Then, at test time, previously unseen clips –to which we may refer as *test* or *query* clips– are given to the system and the most likely action is to be predicted. The analysis of the query clips does not necessarily happen in real time.

The problem is simpler if the videos are already pre-segmented into clips containing a single action. It can be made harder if, for example, the videos contain more than one action, or multiple people performing different actions at the same time. Several datasets are being used to test and compare approaches and to assess progress in the field. The datasets vary with regards to the way they present the videos and the extra information available, e.g. whether there are training videos, whether they are pre-segmented temporally, whether people have been tracked and the bounding boxes are available or silhouettes are easy to extract, and so on. Somehow, datasets themselves define the scenarios to work on and the particular task to be achieved. We give an overview of datasets and methodology in Section 2.2.

The reported work on action recognition is varied from many different points of view: the nature of the extracted features, the action representation and its structure, the different inference and recognition techniques, the scenarios and target tasks, etc. The extracted features may capture shape and appearance information, motion information or both. The representation of the action may have different temporal

extents, from a single frame or small set of frames to the whole video. Considering the amount of information extracted we can distinguish between features describing the whole region of interest (ROI) and features describing only a sparse set of local patches. Another criterion to group action recognition approaches can be the way to deal with the temporal dimension, implicitly embedded in the representation or by using explicit dynamic models. There is also a wide range of choice of inference techniques for classification and matching.

Here we present the different approaches organized around the type of representation (holistic vs. sparse) in Sections 2.3 and 2.4 respectively, and the inference techniques used to perform recognition in Section 2.5.

Other tightly related tasks are *action localization* and *event detection*, which deal with long test videos and aim at saying *where* and/or *when* the action or event of interest is happening. We discuss those in Section 2.6.

2.1 Scope

This thesis focuses on action recognition from the stream of 2D mono-camera images directly, without intermediate 3D reconstruction of spatial locations. Therefore, this review excludes literature on motion capture and on action recognition approaches which derive from it. We are interested in solutions that are not specific to humans or articulated objects consisting of rigid parts, so we overlook approaches relying on body part tracking, skeletons, explicit pose estimation or other kinds of strong shape and motion assumptions. Human or pedestrian detection –which is a requirement of some of the action recognition approaches that will be described– are beyond the scope of this review, as well as research work that is specific to periodic motion (e.g. time-frequency analysis), and other specific application domains like facial expression recognition, sign language, etc.

The interested reader may find a review on motion capture and analysis in [78] and a list of more general action recognition reviews in [90].

2.2 Scenarios and methodology

A large family of approaches assumes that the region occupied by the subject of interest is known. In the most demanding cases, it is assumed that the subject's silhouette can be easily extracted, e.g. [146]. In other cases a bounding box may be enough, but then it is common to assume that there is no distracting background motion around the subject, e.g. [21]. Other simplifications include considering static camera [6], actions being readily segmented in time, or a small set of unambiguous actions to recognize [106]. The datasets traditionally used to evaluate and compare approaches are a good reflection of these assumptions. Recently there has been an effort to overcome these limitations, which has been followed by the publication of new datasets representing more and more realistic and unconstrained environments [67, 60].

Most current approaches rely on an existing training set of actions and statistical learning, but others can perform action recognition from a single example –typically those based on template matching, e.g. [110]. Works have appeared that focus on specific scenarios: crowded scenes [47], short-duration actions

[18], high/low resolution videos [6], etc. More generally, research has been directed towards recognition in unconstrained videos, so-called actions "in the wild". Our work is in line with these tendencies. There is also an increasing interest in the online and real-time scenarios, e.g. [145, 138, 94, 149].

The datasets illustrate these scenarios. It is common in the action recognition community to report performance on relatively simple datasets build by the same authors for a particular occasion. However there is an increasing interest in gathering more and more extensive datasets and making them publicly available, to facilitate the assessment of progress in the field.

There are two datasets that had become very popular and were the standard to compare different approaches at the begining of our research: the **KTH human motion** dataset [106] and the **Weizmann human action** dataset [6]. Both contain videos recorded under controlled conditions, showing several actors doing the same repetitive action during the whole clip, the performance style and duration being variable. They contain six and ten action categories, e.g. walking, jogging, running, boxing, hand waving, jumping, etc. In the Weizmann dataset, the foreground silhouettes are included. Background is relatively static and there is only slight camera motion. Recognition performance for these datasets has already saturated.

Another dataset that has been used for performance comparison is the **UCSD mice behavior** dataset [18] containing five actions: drinking, eating, exploring, grooming and sleeping. This dataset allows for the evaluation of the applicability of the approaches to discriminate subtle events and non-human actions.

Ke et al. [47] collect the **CMU** action dataset consisting of 20 minutes of crowded videos containing 110 events of interest (e.g. pick-up, hand wave, jumping jacks, etc). It is more challenging than the previous human action datasets, but still quite limited in extension.

Laptev et al. use movies to build extensive and realistic human action datasets: **Drinking&Smoking** [59], **Hollywood Human Action** dataset (HOHA) [60] and the sequel **Hollywood2** [73]. The later contains 12 categories, e.g. kissing hugging, getting out of a car, answering the phone, fighting, shaking hands, etc. Within the same trend, Rodriguez et al. [96] propose the **UCF sports** dataset, containing sequences of sport motions, e.g. diving, golf swinging, kicking, weight-lifting, horseback riding, running, skating, swinging a baseball bat and walking. Bounding boxes of the human figure are provided with the dataset. For most action classes, in these datasets, there is considerable variation in action performance, human appearance, camera motion, viewpoint, illumination and background.

Mikolajczyk and Uemura [77] acquire the **Multi-KTH** dataset, an extension of KTH containing the same action categories but performed simultaneously by several actors, with more complex backgrounds, occlusions and camera motion. In the same paper, they show the performance of their system on a sports dataset containing 17 sport categories.

Recently, more datasets obtained from sports footage, YouTube and home videos have been introduced and used by their respective authors, e.g. **UFC fighting** [145], **UCF YouTube** dataset of actions "in the wild" [67], and a dataset containing **social games** interactions [127]. A dataset containing several non-periodic actions and interactions happening simultaneously has also been introduced by Ryoo and Aggarwal [100].

Particular evaluation practices are associated to the different datasets. Techniques are evaluated on a test set, or by cross-validation over splits of the available training data, being typical leave-one-person-out cross-validation. Typical performance measures in the recognition task include accuracy, average precision (in the biclass case), and mean average precision (in the multiclass case).

2.3 Figure-centric representations

This section deals with approaches that focus on describing the zone occupied by the subject performing the action. They build action representations centered on the human figure, typically consisting of dense descriptions of appearance, motion or both. The captured information is rich, but it is assumed that the localization has been solved previously. Therefore, even if good performance have been reported in simple scenarios, there may be limitations in their applicability to unconstrained videos.

For example, some approaches assume that the sequence of body silhouettes is available. However foreground segmentation is hard in cluttered and dynamic environments. The region of interest (ROI) may also be given under the form of bounding boxes. In that case, the input is the stabilized sequence of cropped frames centered on the human figure. Even if a milder assumption, it would be unrealistic to consider that the actor can be reliably tracked in complex videos.

Many early solutions to the problem of action recognition are based on **template matching** and would correspond to this category. One representative example is the work by Davis and Bobick [17]. They combine silhouettes from consecutive frames into Motion Energy Images and Motion History Images. These are 2D images whose pixels' values indicate the presence and recency of motion respectively. These templates can be described and compared to others by computing their Hu moments. MHI have been used as basic features in more complex systems, e.g. [11]. Weinland et al. [133] extend them to 3D: they build Motion History Volumes from multiple synchronized cameras and compute viewpoint-invariant descriptors of these templates.

Efros et al. [21] take as input the stabilized cropped frames. For each frame, they computed a motion descriptor based the spatial arrangement of blurred optical flow, previously separated into sparse non-negative channels. The frame descriptor is to be matched via spatiotemporal cross correlation against the frames contained in a labeled dataset. This descriptor became quite popular and can be found with slight modifications in subsequent attempts to exploit motion information, as a building block of more complex representations, e.g. [24, 130, 63]. Interestingly, these blurred arrangements are reasonably discriminative, thus being a robust descriptor even if the flow cannot be estimated precisely. However, they are sensitive to distracting background motion.

Shechtman and Irani [110] avoid the explicit computation of flow and choose to describe motion using spatiotemporal gradients instead. They derive a correlation measure between two spatiotemporal volumes that accounts for consistency of their underlying motion fields. Given a query, i.e. a cropped video containing an action of interest, this correlation measure can be computed between the query and a target video at every position to perform action detection. This motion consistency measure builds on previous work [150] in which it is used to perform temporal segmentation of long video sequences into event-consistent sub-sequences. A recent approach by Seo and Milanfar [109] can be considered

a generalization of [110]. They use local steering kernels and Principal Components Analysis to build resemblance volumes.

Actions have also been regarded as 3D spatiotemporal shapes. Stacking consecutive silhouettes results in a spatiotemporal volume. Yilmaz and Shah [146] extract local geometric features of its surface, while Blank et al. [6] use the solution to the Poisson equation to extract saliency and orientation. Ke et al. [47] also consider actions as spatiotemporal shapes, but their approach needs no extraction of the sequence of silhouettes at inference time. Instead, they use color-based mean-shift to obtain a spatiotemporal over-segmentation of the whole video. In order to detect a given action, they perform volume matching between the query volume and the video segments. The matching measure includes both the shape correspondence and the motion consistency (like in [110]). This technique can be extended to matching by parts of the original template query, thus becoming suitable in crowded scenes with partial occlusion and overcoming one typical limitation of such holistic approaches.

A disadvantage of template-based methods is their rigidity and dependence on viewpoint, scale, etc. Typically they are unable to generalize from a collection of examples. Alternatively, **statistical learning** can be used to learn action categories. Examples of approaches using discriminative classifiers on the computed features can be found in [46, 43, 105, 24, 130, 36]. For example, Ke et al. [46] compute optical flow on the whole video and use "histogram videos" to efficiently compute Haar-like volumetric features on the components of the optical flow. They train a cascade supervised classifier and adopt a sliding-window approach to perform action detection.

The group of figure-centric approaches further includes those that build a vocabulary of key poses or prototypes and consider actions as sequences of those prototypes. Although these representations are potentially more expressive than, for example, motion history images or volumes, in practice their capability of discriminating subtle nuances may be limited by the granularity of set of prototypes. Weinland et al. [131] use probabilistic dynamic models to represent sequences of 3D hulls obtained from multiple synchronized cameras. They match their projections to the 2D observations. In a later work, they consider actions as sets 2D key poses, ignoring temporal dynamics [132]. A recent work [63] uses prototypes built by clustering frame descriptors, which include both shape information (silhouettes or likelihood maps) and motion information (similar to [21]).

Recently, there have been some attempts to learn flexible templates from several examples, overcoming the rigidity of traditional templates. This is the case of Rodriguez et al. [96]. Similarly in spirit to [110], they build a filter to detect actions. This filter – which is an extension to spatiotemporal volumes of the traditional MACH filter – has the advantage of capturing intra-class variations, and detection can be done very efficiently by analyzing the response in the frequency domain. Yao and Zhu [143] learn deformable templates from several examples using the pursuit algorithm. The use a generative model to describe some motion and shape primitives contained in a frame, and align and match actions using Dynamic Spatio-Temporal Warping.

2.4 Sparse representations

This section encompasses approaches that describe local video patches and/or the configuration of those patches. Compared to holistic figure-centric approaches, sparse local representations tend to be more robust to deformations, occlusions and changes in viewpoint, similarly to their analog in the object recognition domain. Their main requirement is to have a sufficient amount of relevant patches. Usually, it is useful to compensate for camera motion; otherwise, the background may generate distracting features.

By far, the most popular techniques are based on the local description of a collection of patches, which are typically extracted at selected interest points, but can also be obtained by random or dense sampling. In these techniques, the clip or frame is represented by some statistical model capturing to some extent the relationships among the extracted patches, the typical example being a the histogram of appearance of several types of patch.

In this section we also include approaches based on the analysis of the configuration of interest points, without any local description, as well as tracking-based approaches.

2.4.1 Spatiotemporal features

Motivated by the success of local viewpoint invariant features in the field of object recognition and image matching, Laptev and Lindeberg [56] propose to extend the notion of spatial **interest point** to the spatiotemporal (ST) domain. They present a 3D version of the Harris detector that detects ST corners. Dollár et al. [18] claim that ST corners are rare in practice, specially for more subtle actions than running or waving. They propose an alternative ST point detector based on Gabor filters. It detects spatially distinguishable regions undergoing complex motion and leads to denser sets of points. These two detectors have been extensively used in combination with ST descriptors in subsequent works, e.g. [20, 73] and [143, 67].

In parallel, a considerable amount of work has focused on the derivation of alternative ST point detectors based on different definitions of saliency [37, 87, 136]. Willems et al. [134] make a review of existing local ST detectors in terms of efficiency, automatic scale selection, and density, and propose their own alternative overcoming the typical limitations. Gilbert et al. [31] use 2D Harris corner detector to the planes x-y, x-t, y-t, thus providing ST information while obtaining much denser collections of points.

The ST detectors are usually used combined with rich **descriptors of the ST volume** at the corresponding location and scale. There is a wide choice of ST descriptors. The encoded information typically consists of the components or orientations of optical flow or 2D/3D gradients, expressed as a flat vector, using histograms or simply retaining mean value and variance. Some sort of dimensionality reduction such as Principal Components Analysis may be applied to the descriptor. Examples of ST descriptors are the histograms of oriented gradients (HOG) and histograms of optical flow (HOF) [60, 59, 73], histograms of 3D gradient orientations [50], extended SIFT [108], extended SURF [134], among others [57, 18, 125].

Raptis and Soatto [92] have recently proposed *tracklets*, which are extensions of the HOG/HOF descriptors with a more principled theoretical basis. Instead of describing a ST cuboid around a static

point, they track points and retain the gradient and flow information around them from their birth to their death. In posterior work, Wang et al. [124] also describe tracked points, and use a descriptor based on gradients of optical flow (motion boundary histograms, MBH), new in the domain of action recognition.

There have also been recent attempts to *learn* local ST features rather than using hand-crafted descriptors such as HOG or HOF, following approaches reminiscent of neural networks [62, 118].

As an alternative to ST features, some researchers have opted for extracting classical spatial features frame by frame and use them to recognize actions [77]. These so-called static features may also be used in combination with the the ST ones [67].

2.4.2 Vocabulary of visual words

The space of descriptors is very large and is usually compressed and discretized by building feature vocabularies. This is typically done by unsupervised clustering of features extracted from training videos. The most common technique to do so is k-means, e.g. [56, 136, 60, 20, 127, 125]. A non-planar vocabulary can be obtained by building feature trees, which allow for efficient feature matching [77, 94].

The granularity of the vocabulary has a great impact on the performance. The choice of the number of words in the vocabulary is non-trivial, as there is a trade-off between the representativeness of the words and their discriminative power, which is related to the loss of information due to the quantization. To minimize the impact of the granularity issue, there have been attempts to obtain more compact yet discriminative vocabularies by fusing words according to their mutual information [65, 67] or co-occurrence [108]. Liu et al. [67] show that their technique is consistently superior to k-means for different vocabulary cardinalities.

Even if becoming popular in other vision tasks, the use of efficient discriminative vocabularies as in [80] does not seem a common practice in the action recognition field. And yet some recognition approaches (e.g. [77, 94]) rely on the assumption that individual features are more or less specific to particular actions, i.e. actions are recognized according to the votes independently casted by the features.

2.4.3 Aggregation of features

By far, the most popular way of representing a clip is the **bag of features** (or bag of words) [106, 18, 136, 60, 84, 20, 125, 67, 73]. The bag of features is the histogram of all the visual words extracted from the clip, i.e. features which have been quantized according to some vocabulary. This representation inherits the advantages and drawbacks of its homolog in 2D images: they are simple and robust against deformations and noise, but disregarding all spatial and temporal information may limit its discriminative power in certain cases. As noticed by Hospedales et al. [35], they also suffer from a trade-off in determining the temporal window to collect the bag of words: short windows may split actions arbitrarily, while long windows risk overlooking actions of short duration. Duchenne et al. [20] explore this phenomenon empirically. They train classifiers on short video segments. They experiment with different lengths of the training segments containing the action of interest. They observe decreasing performance with increasing lengths, illustrating the importance of temporal localization. This issue has also been observed by Satkin and Hebert [102] who try to extract the most discriminative portion of a training video automatically to improve classification.

There are extensions and alternatives to this approach that try to incorporate some information about the **spatial and/or temporal configuration** of the extracted visual words. For example: spatiotemporal pyramid matching [65], concatenation of bags of features computed in spatiotemporal grids [60], enforcing temporal ordering using sequential representations [86, 127], encoding feature co-occurrences [103, 108] and correlograms [65], using constellations and other part-based models in space or space and time [83, 77, 136], etc. Gilbert et al. [31] group interest points hierarchically in spatiotemporal neighborhoods and detect frequent configurations using data mining. Ryoo and Aggarwal [100] model explicit temporal and spatial relationships among the extracted features and derive a spatiotemporal match kernel to compare sets of features from two different videos. In Section 6.1 we describe more publications aiming at describing neighborhoods of spatiotemporal features.

A general drawback of all these approaches based on vocabularies of features is that the vocabulary is usually expensive to compute, is fixed during the training phase and cannot grow with new incoming features. Even if the construction of a new vocabulary could be done efficiently, that would affect the features identities and any subsequent video representation, e.g. the new bag of features could not be computed from the previous one. Relying on the assumption that features alone can be sufficiently discriminative without any further modeling, Reddy et al. [94] present a method that does not require feature quantization. Features are organized in a tree that can be efficiently grown by adding features from new incoming videos. Recognition is performed by matching at the feature level and voting schemes. The vote casted by each feature depends on the labels associated to neighboring features in the tree.

As an alternative to rich descriptors and vocabularies, some researchers opt for much simpler descriptions of the ST interest points. For example, Gilbert et al. [31] describe interest points only by their scale and roughly quantized dominant orientation, resulting in a very low dimensional descriptor. Stronger and discriminative features are then obtained by hierarchical grouping those simple features within ST neighborhoods, and the application of data mining techniques.

Another way of avoiding the vocabulary and all the associated problems is to focus on the spatiotemporal distribution of interest points, ignoring their local description altogether. Oikonomopoulos et al. [87] match sets of unbalanced numbers of features through spatiotemporal dynamic warping, and use the obtained distance measure in a supervised classification framework. Bregonzio et al. [8] recognize actions by describing clouds of interest points at different temporal scales. Oshin et al. [88] use randomized ferns within the action boundaries to learn spatiotemporal distributions of points.

2.4.4 Tracking-based representations

Tracking-based approaches concentrate on describing the trajectories followed by the detected interest points, rather than their local appearance. Typically, these trajectories are obtained using the Kanade-Lucas-Tomasi tracker. Messing et al. [76] use the description of long-term trajectories as the observation in a generative mixture model. Appearance information can also be incorporated into their model. Matikainen et al. [74] build a vocabulary of short term trajectory patterns called *trajectons* and use the bag-of-words representation and supervised learning. Sun et al. [117] describe trajectories at three levels: local appearance, trajectory shape, and co-occurrence of trajectory types. Each level is represented

by a bag of features in a spatiotemporal grid, and all the resulting feature channels are combined in a Multiple-Kernel Support Vector Machine.

Other approaches are based on the analysis of trajectories of human body joints, but those are beyond the scope of this review as they assume a geometric model is available.

2.4.5 Dealing with distracting features

Sparse representations are often affected by camera motion. They are also affected by the detection of irrelevant and distracting features. The first issue can be partially solved by estimating the dominant homographies to compensate motion [120] or subtracting median motion [77]. Laptev and Lindeberg [58] propose features that are invariant to local velocity. The second issue can be avoided using region-of-interest (ROI) estimation techniques, e.g. [8]. Liu et al. [67] address both by pruning and mining static and motion features.

2.5 Inference

The inference method is normally tied to the chosen representation. They can be grouped according to the implicit or explicit modeling of temporal dynamics.

In the first group, we can find all those methods performing direct decisions, taking as input the representation of the global clip or frame-by-frame representations. Frame-by-frame decisions can be extended to clip decisions using simple voting schemes.

Nearest Neighbor classifiers are very popular [21, 56, 6, 94], as well as discriminative models like Support Vector Machines (SVM) [106, 43, 145, 117, 105, 60, 108, 73], Relevance Vector Machines (RVM) [87], boosting [24, 67, 59, 86], Conditional Random Fields (CRF) and extensions [130], etc. Generative models such as latent topic models can be also be used for unsupervised classification [84, 136]. The number of topics has to be fixed a priori. They have been used in a way that assumes that each obtained topic corresponds to an action, which is not necessarily true due to the total absence of supervision. Other models allow to infer the number of topics from the data (e.g. Hierarchical Dirichlet Processes [119] used in [129]), but they have not received much attention from the action recognition community until very recently, e.g. [64].

In the second group we can find techniques explicitly modeling the temporal dimension of actions. Some approaches consider actions as sequences of symbols that have to be mined from the data, aligned and matched, e.g. [11, 127]. Lin et al. [63] use Dynamic Time Warping to align and match sequences of prototypes. Yao and Zhu [143] perform Dynamic Spatio-Temporal Warping to align their sequences of deformable templates.

Niebles et al. [82] use the same latent-part-based model as Felzenszwalb et al. [25] for object recognition, only that in this case the latent information is the temporal position of the parts instead of their spatial position.

Graphical models and probabilistic dynamic models in general –both generative and discriminative– are very popular and intuitive tools for modeling temporal sequences. Some examples are stochastic grammars [152], Hidden Markov Models (HMM) and extensions [131, 22], CRF [111] and extensions

[126, 128], among others.

HMM are powerful generative models with latent state structure. They have been widely used to model human motion as a sequence of states corresponding to pose, e.g. the image location of body joints, or the parameters of some geometrical model of body parts. This can be applied to action recognition, but is beyond the scope of this review because of the structural assumptions. Their main drawback is that they assume conditional independence between observations, which makes it hard to model overlapping and long-range dependencies. CRF and extended models (e.g. Hidden CRF) avoid the independence assumption and allow for non-local dependencies between states and observations, but are not generative. All these probabilistic models involve assumptions about the probability distributions of the stochastic variables of the model, the development of inference and learning methods, and need large amounts of training data to learn all the involved parameters [1]. Non parametric approaches, e.g. [27], remain under-explored for action recognition.

2.6 Beyond action classification

The action recognition community is interested in other tasks complementary to pure action classification, such as action localization and detection of anomalies or particular events. Thus, some approaches have recently appeared that are oriented towards these goals.

As has already been mentioned, template matching or classifiers can be used to perform action localization through sliding-window techniques. Sliding windows are in general computationally expensive. Further, classifiers need to be trained on positive examples, i.e. bounding boxes of the good locations, implying a huge annotation effort, e.g. Oshin et al. [88].

The match kernel by Ryoo and Aggarwal [100] overcomes the computational cost limitation of sliding windows by allowing for direct partial matching of two video representations. In the case of Rodriguez et al. [96], their filter performs very efficient detection by analyzing its response in the frequency domain.

Lin et al. [63] use a joint likelihood model of actor location and action prototype, which they optimize at inference time using likelihood maps provided by an actor tracker and by sampling at several locations. However, they need figure-centric sequences at training time. The frame-by-frame spatial localization technique based on centroid voting used by Mikolajczyk and Uemura [77] needs no sliding window to infer location, but does require bounding boxes at training time.

More publications aiming at localizing actions in time or performing action search are [28, 33, 148]. Gaidon et al. [28] represent actions by the concatenating three bags of words, which they call *actoms*. These actoms are located at different key frames of the action and have different temporal lengths, which automatically determined depending on the distance to the next and previous actom. For training, they need to manually annotate the positive examples, consistently indicating the temporal location of the three key frames. They train an SVM classifier with the three concatenated histograms, using an intersection kernel. They also estimate a generative model for the distribution of the relative temporal locations of the actoms, resulting in a few candiate temporal sequences of actoms. In order to detect actions in test videos, they use a sliding-window approach and non-maxima suppresion. For a given

temporal position, the score is obtained by marginalization over the candidate sequences of actoms, each evaluated with the SVM.

In order to alleviate some of the annotation effort, Duchenne et al. [20] use weakly-supervised clustering of videos into temporal segments containing a particular action, and use those segments to train temporal action detectors. The weak supervision is obtained by means of movie script alignment, like in [60]. Ikizler-Cinbis et al. [38] learn action models from still images obtained by querying a web image search engine, thus being very flexible with regard to the possible action categories.Yao and Zhu [143] need only one manually annotated video to build a deformable template and use it to detect other training videos of the same class and iteratively update the template. Aware of the ambiguities in determining the spatial and temporal boundaries of an action, Hu et al. [36] use multiple-instance learning (MIL) to recognize and locate actions in space and time. Ambiguities in the labeling process are allowed, thus reducing the labeling labor without sacrificing performance. They label manually only rough head positions and the approximate frame where the action happens.

A little amount of approaches are able to perform localization with no localization annotation during the training phase, i.e. only weak supervision of class labels. The pruning technique to select relevant features used in [67] at training and testing time allow them to infer spatial localization by computing the centroid, assuming only one action of interest is happening at a given frame. Gilbert et al. [31] use data mining to learn discriminative patterns of interest points at training time and identify them efficiently at testing time, allowing them to perform localization even of multiple simultaneous actions.

A related task is event detection and retrieval. One recent example of approach addressing anomaly detection is [35]. They derive a new generative model that combines the advantages of probabilistic topic models (e.g. Latent Dirichlet Allocation) and dynamic Bayesian networks (e.g. HMM) and is rich enough to capture the typical co-occurrence relationships among visual events, thus allowing to detect online anomalies of different natures (not only rare events, but also rare combinations of them). Yet another application-specific technique is [127], in which they retrieve quasi-periodic patterns in an unsupervised way from unstructured videos.

Chapter 3

Framework

In view of the existing literature, we consider that the bag-of-words approach to video classification is a suitable baseline, with state-of-the-art performance, and yet very simple and well-understood. Here we present a description of this widespread approach. It basically aims at obtaining a global representation of the video (or image) through the accumulation of sparse local information into a fixed-sized description vector. We refer to this video description as a *video signature*.

The set of local features detected in a video may have different cardinality from one video to another. One advantage of the bag-of-words approach is that it takes this variable ensemble and transforms it into a signature with fixed size for any video. This makes it possible to use off-the-shelf machine-learning techniques for classification.

The pipeline can be divided into the following stages, as illustrated in Figure 3.1:

(i) Sampling spatiotemporal points, e.g. using an interest-point detector, sampling uniformly or randomly, more or less densely.

(ii) Computing a **descriptor** for each sampled point, e.g. related to the gradient or the optical flow in the neighboring pixels.

(iii) Discretizing or **quantizing** the descriptors using a vocabulary of visual words, which can be pre-computed using some example descriptors[1]. This step can also be called *coding* and the vocabulary be called *codebook*.

(iv) Summarizing the content of the video by aggregating the quantized descriptors. This step is sometimes called *pooling*, e.g. [140]. Typically, local features are accumulated into a global histogram of visual words.

The final video descriptor obtained this way is then the input of a **classifier**, which needs to be trained on some example pairs of input video signatures and output video categories.

The bag-of-words representation of videos inherits the advantages and drawbacks of its homolog

[1] Strictly speaking, one should distinguish between the *vocabulary construction*, i.e. the process of defining a vocabulary from training data, and the *quantization*, i.e. the process by which any descriptor is assigned to one or more words in the vocabulary. Regarding the later, we consider that a vocabulary is a *partition* of the descriptor space, and simply assign the descriptor to the word associated to the *part* in which in lays (roughly, the "nearest" word). This is called "hard" coding, but other schemes could be possible, e.g. "soft" assigning the descriptor to several nearby words, e.g. [68]. Our focus is how to obtain such partition, so in our text we may abuse the term *quantization/quantize* to refer to both.

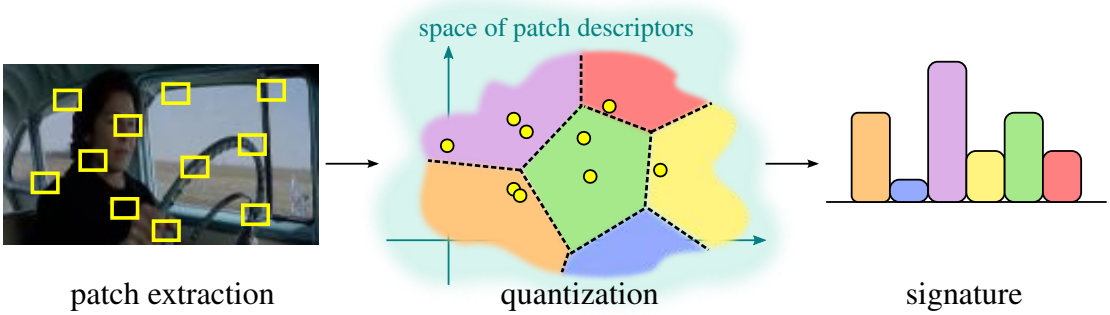


Figure 3.1: Bag-of-words representation of a video.

in 2D images: it is simple and robust against deformations and noise, but disregarding all spatial and temporal information may limit its discriminative power in certain cases. Some information about the spatiotemporal distribution of the words can be preserved by concatenating histograms computed over a spatiotemporal grid. In 2D images, spatial pyramids [61] are a well-established approach for classification.

The following sections describe our typical choices for local detectors and descriptors and experimental setup. The quantization step and generalizations of the bag-of-words framework are presented in Chapters 4, 5 and 6.

3.1 Spatiotemporal detectors and descriptors

3.1.1 STIP features

STIP stands for "Spatio-Temporal Interest Points". We mean by STIP features a certain combination of detector and descriptor used successfully for action recognition by Laptev et al. [60]. An implementation is available for download in the author's webpage[2] in the form of compiled binaries. The detector is based on a Harris detector extended to 3D corners, as described in [55]. The code does not select for scale; instead, interest point are detected at several spatial and temporal scales given as input. The descriptors of the 3D patches around the detected points can be Histograms of Oriented Gradients (HOG) or Histograms of Optical Flow (HOF). The lengths of these descriptors are 72 and 90 respectively. This implementation was updated in 2011 so that the 3D Harris detector could be replaced by dense regular sampling of points or to admit any set of detected points as input.

We use STIP features in our experiments in Chapters 4 and 5. In particular, we use the initial version of the detector (the only one available when we started this work) with the same default parameters an in the original publications, and use a concatenation of the HOG and HOF descriptors (HOG|HOF for short). In the fore-mentioned chapters, we may refer to STIP features or HOG|HOF features interchangeably.

[2] http://www.di.ens.fr/ laptev/download.html#stip

3.1.2 DenseTrack features or Dense Trajectories (DT)

These spatiotemporal features are presented by Wang et al. in their 2011 work [124]. The code is available for download and compilation in the author's website[3], as well as instructions on how to set all the parameters.

Initially, points are sampled following a regular grid. For each point, a 15-frame-long trajectory is computed by concatenating median-filtered optical flow. In the areas where the density of tracked points becomes low, new points are sampled. Each point trajectory defines a local (x, y, t)-tube (rather than a 3D patch like in the STIP features). The tubes are described by means of several types of descriptor. The trajectory itself (Traj) is described through the concatenation of the x and y displacements, the resulting descriptor having 30 dimensions for the default track length of 15 frames. Histograms of Oriented Gradients (HOG) and Histograms of Optical Flow (HOF) can be also computed, which have 96 and 108 dimensions respectively for the default parameter setting. Finally, they introduce the Motion Boundary Descriptor (MBH) in x and y, which are oriented gradients of the x and y components of the optical flow, of 96 dimensions each. In this work we concatenate both into a single MBH descriptor of 192 dimensions. We use these DT features in Chapters 6 and 7.

3.2 Vocabulary training

As detailed in the next chapter, out preferred method to build visual-word vocabularies is to use Extremely Randomized Trees [30] as in [80], which are called in this context ERC-Forests. We may also use the expression *random forests* to refer to this quantization technique. The number of training features that we use depends on memory availability; typically 10^6 for STIP descriptors, $2 \cdot 10^5$ or $4 \cdot 10^5$ for DT features and region descriptors of Chapter 6. More implementation details can be found in Section 4.5.

3.3 Datasets

3.3.1 The KTH human action dataset

This dataset was introduced in [106]. Compressed AVI files and ground-truth anotations per clip are publicly available[4] This dataset is representative of simple periodic actions on simple background and with very little camera motion. It is composed by videos of 25 subjects performing six different actions in four different scenarios –thus 600 videos– which can each be divided in four subsequences. We use this division into subsequence, and therefore work with around 2400 clips (in practice this number is lower because of missing data). The six actions are walking, jogging, running, boxing, hand waving and hand clapping, and the scenarios include indoors and outdoors, varying scale and different clothing. A natural partition of this dataset for training and testing purposed is based on the subjects. The authors originally propose to use subjects number 11-18 for training a supervised classifier, subjects 01, 04, 20, 21 and 23-25 for validation and choice of the classifier parameters, and subjects 22, 02, 03 and 05-10 for testing. Instead, we fuse the training and validation sets into a single training set and use 5-fold cross-validation to eventually pick any classifier parameters.

[3] http://lear.inrialpes.fr/people/wang/dense_trajectories

[4] http://www.nada.kth.se/cvap/actions/

Table 3.1: Composition of the Hollywood2 dataset.

	training	testing
AnswerPhone	66	64
DriveCar	85	102
Eat	40	33
FightPerson	54	70
GetOutCar	51	57
HandShake	32	45
HugPerson	64	66
Kiss	114	103
Run	135	141
SitDown	104	108
SitUp	24	37
StandUp	132	146
total	823	884

3.3.2 Hollywood2 dataset

This dataset was originally used by Marszalek et al. [73] and is publicly available[5]. It is composed by clips from 69 movies. There are 823 training and 884 testing clips. Movies are kept separate in the training a test sets, so that each movie is used only for training or testing. Each clip comes with one or more category labels out of 12 possible human actions, see Table 3.1. These challenging clips, contrarily to the first action datasets such as KTH [106], do not contain actions in controlled environments, but in realistic scenarios, with high variability, considerable clutter, etc. They have high resolution, so when extracting STIP features their resolution is typically halved, e.g. [125]. Note the disproportion in the amount of clips belonging to each category, which introduces an extra difficulty. Some classes are better represented and *a priori* "easier" to learn. This means that an "aggressive" training which focuses only on a small set of easier actions may result in better overall performance than a training trying to maximize the worst of the per-class performances. This observation is also made by Matikainen et al. [74], who compare both approaches in their work.

3.3.3 UCF YouTube Action dataset

First used in [67], this dataset is available in the UCF webpage[6]. In contains short home-made clips from YouTube covering 11 actions: basketball shooting, biking/cycling, diving, golf swinging, horse back riding, soccer juggling, swinging, tennis swinging, trampoline jumping, volleyball spiking, and walking with a dog. There is great variety in camera motion, viewpoint, appearance, scale etc. Image quality is much worse than in Hollywood2. Each clip of the dataset belongs to one action only. Each set of actions is divided into 25 groups. Each group may contain videos with the same actor or environment. Such grouping is useful in order to perform cross-validation in a more realistic way, avoiding that very similar videos are used for training and testing. However, we have observed some deficiencies in such partition, with different groups containing almost identical videos. There is room for improvement in that sense.

[5] http://www.di.ens.fr/ laptev/actions/hollywood2/

[6] http://vision.eecs.ucf.edu/datasetsActions.html#UCF%20YouTube%20Action%20Dataset

Each group contains at least four videos, and in some publications (e.g. [67]) only the first four videos of each group are used. We use instead all the clips, which makes a total of 1600 videos.

The original way of evaluating classification performance on this dataset is leave-one-group-out cross-validation. For a less exhaustive but faster evaluation, we divide the dataset into five partitions only. We call the first partition YouTube-CV1, which consists of groups 1 to 5 for testing and the rest for training. The second partition is YouTube-CV2 and consist of groups 6 to 10 for testing and the rest for training, and so on until YouTube-CV5. Instead of measuring multiclass classification accuracy as in [67, 125] we prefer to evaluate the per-class classifiers as detailed in Section 3.5.1.

3.4 Classifier

Our choice for two-class classifier is the Support Vector Machine (SVM) [10].

Given a labeled training set $\{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$ of video signatures $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ and their binary categories $y_i \in \mathcal{Y} = \{-1, 1\}$, one approach to classification is to find the function $f : \mathcal{X} \to \mathcal{Y}$ that solves

$$\min_{f \in \mathcal{F}} \quad \underbrace{\textstyle\sum_{i=1}^{n} l(y_i, f(\mathbf{x}_i))}_{\text{empirical error}} + \underbrace{\frac{\lambda}{2}\|f\|^2}_{regularization}, \tag{3.1}$$

where λ controls the strength of the regularization term, and the loss function l and function space $\mathcal{F}$ are *a priori* choices with theoretical and algorithmic consequences.[7]

Consider a positive definite kernel k on $\mathcal{X} \times \mathcal{X}$. There exists a mapping $\mathbf{\Phi} : \mathcal{X} \to \mathcal{H}$ such that the kernel in the input space is the inner product in the destination space, so

$$\forall (\mathbf{x}, \mathbf{x}') \in \mathcal{X}^2, \quad k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{\Phi}(\mathbf{x}), \mathbf{\Phi}(\mathbf{x}') \rangle_{\mathcal{H}}. \tag{3.2}$$

$\mathcal{H}$ can be instantiated as a function space by taking $\mathbf{\Phi}(\mathbf{x}) = k(\cdot, \mathbf{x})$ so that

$$f(\mathbf{x}) = \langle f, \mathbf{\Phi}(\mathbf{x}) \rangle = f^\top \mathbf{\Phi}(\mathbf{x}). \tag{3.3}$$

In that case, $\mathcal{H}$ is the reproducing kernel Hilbert space (RKHS) associated to the kernel k.

If we make that particular choice of space of functions, i.e. $\mathcal{F}$ being the RKHS of kernel k, and by the representer theorem, an equivalent problem to Eq. 3.1 is

$$\max_{\alpha \in \mathbb{R}^n} \quad \textstyle\sum_{i=1}^{n} l(y_i, (\mathbf{K}\alpha)_i) + \frac{\lambda}{2}\alpha^\top \mathbf{K}\alpha, \tag{3.4}$$

where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is the kernel matrix of the training examples, $\mathrm{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, and the relationship between Eq. 3.1 and Eq. 3.4 at the optimum is

$$f = \sum_{j=1}^{n} \alpha_j \mathbf{\Phi}(\mathbf{x}_j) = \sum_{j=1}^{n} \alpha_j k(\cdot, \mathbf{x}_j). \tag{3.5}$$

In other words, the classification of a test point $f(\mathbf{x})$ is roughly a linear combination of kernel evaluations between the test point and the training points $k(\mathbf{x}, \mathbf{x}_j)$, and the vector of optimal weights α is to be learnt.

[7] See the tutorial http://www.di.ens.fr/ fbach/INRIA_summer_school_2012_fbach.pdf for more details on this and following statements.

In particular, traditional SVM choose l to be the hinge loss and the kernel to be linear, so that the possible classification functions are hyperplanes in $\mathcal{X}$.

We use the LIBSVM implementation [13], which solves a dual problem efficiently thanks to the convexity of the hinge loss. It integrates a number of possible choices for the kernel function, and it also admits personalized kernel matrices. The inputs are the type and parameters of the kernel (or the custom kernel matrix) and a regularization parameter $C \propto \frac{1}{\lambda}$. It also allows for asymmetric losses

$$\min_{f \in \mathcal{F}} \quad C w_+ \sum_{\substack{i \\ y_i=1}} l(y_i, f(\mathbf{x}_i)) + C w_- \sum_{\substack{i \\ y_i=-1}} l(y_i, f(\mathbf{x}_i)) \quad + \|f\|^2 \tag{3.6}$$

through class weights w_+ and w_- (compare to eq, 3.1), which are useful when the training set has an unbalanced number of positive and negative examples.

We opt for natural balancing, and automatically set the class weights to be inversely proportional to the number of training items belonging to the positive and negative class, so that $w_+ n_+ = w_- n_-$. We use a default value of 100 for parameter C. Otherwise, we choose it through 5-fold cross-validation, taking classification accuracy as the performance measure.

In most of our experiments, we use a custom exponential χ^2 kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{d_{\chi^2}(\mathbf{x}, \mathbf{x}')}{A}\right), \tag{3.7}$$

where d_{χ^2} is the symmetric χ^2 distance

$$d_{\chi^2}(\mathbf{x}, \mathbf{x}') = 2 \sum_i \frac{(x_i - x'_i)^2}{x_i + x'_i}. \tag{3.8}$$

As in [125], we set the parameter A to be the average distance between pairs of training points.

In order to deal with multiple classes, we transform the problem into several two-class classification tasks in a one-vs-all manner, i.e. one classifier per class is trained in which one class at a time is considered positive and the rest negative.

3.5 Evaluation

3.5.1 VOC Average Precision

We evaluate two-class classification as a detection task. Consider a two-class classifier with a continuous output, allowing to rank elements from the most to the least likely to belong to the positive class. Putting a threshold on that output, one obtains a binary classification into positive and negative class. Varying this threshold permits to adjust the trade-off between the number of false alarms and non-detections.

The precision/recall curve describes this trade-off. It can be computed from the ranked outputs obtained on a test dataset. For a given rank, *recall* is the fraction of positive examples above the rank out of all the positive examples in the test set (i.e. the ratio between true detections and all positives). *Precision* is the fraction of positive examples above the rank, out of all the examples above the rank (i.e. the ratio between true detections and all detections).

The Average Precision (AP) is a value that summarizes the precision/recall curve. We compute it following PASCAL Visual Object Classes (VOC) Challenge procedure [23]. Eleven equally spaced

recall levels $r \in \{0, 0.1, ..., 1\}$ are taken. For each r, precision is interpolated by taking the maximum value of precision observed for any recalls equal or higher than r

$$p_{\text{interp}}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}), \tag{3.9}$$

where $p(\tilde{r})$ is the precision measured at recall $\tilde{r}$. AP is the average of the eleven interpolated values. We evaluate multiclass classification in terms of the *mean AP* obtained over all the classes.

3.5.2 Classification accuracy

For convenience, we use classification accuracy (percentage of correct predictions) when choosing parameters by cross-validation. This enables to make the folds arbitrarily small (e.g. leave one out), contrarily to Average Precision, which only makes sense for relatively large tests sets.

Chapter 4

Quantization of Local Features

An important step of the bag-of-words approach is the quantization of local descriptors, which can have a critical impact on the final performance. This chapter deals with this step. After a brief review of relevant existing techniques, we focus on ensembles of randomized trees, which allow us to introduce supervision in a very flexible way. We explain how this classification algorithm can be used in the bag-of-words framework as a substitute for k-means clustering. Furthermore, we propose and evaluate a new objective function to train randomized trees which is specifically designed to obtain discriminative video signatures. Although there is room for further improvement, we show that our new form of supervision can perform better than traditional forests. This is also confirmed by experiments in Chapter 5.

4.1 Quantization techniques in the literature

A quantization process maps elements of a potentially large continuous input space to elements of a smaller discrete space. For example, rounding real numbers to some unit of precision is a form of quantization. In signal processing, it is typically used to compress signals by approximating them with elements from a smaller set. The idea is to choose such elements so that the approximation error is minimized, taking advantage from the statistical properties of the signal. At the same time, the size of the output set is to be minimized too, or at least some constraints on it must be satisfied depending on the application, e.g. a desired bit rate to send the signal. There is a trade-off between the approximation error and the size of the output space, i.e. the quantity of resources used to approximate the signal. Intuitively, it is desirable to allocate more resources to approximate finely the elements of the input space that appear more often, whereas the rare elements are approximated coarsely, in order to minimize the overall error.

In the case of the bag-of-words representation, the input is the multidimensional space of local descriptors (or feature vectors) and the output the set of visual words. K-means clustering is the most widespread technique to build a vocabulary of visual words given some training descriptors. Given a target number of clusters k and an initial set of k centroids, this algorithm iteratively assigns each training element to the nearest centroid, then updates each centroid so that it best approximates its corresponding subset of elements, until convergence. The final result depends on the initialization and the typical metric is Euclidean distance. From this description it may seem that k-means finds a local minimum of the quantization error mentioned in the previous paragraph. This would be the case if the algorithm tried

to segment the data into groups with the similar number of elements. Instead, k-means assigns training elements to centroids based on distance only. The underlying assumption is that the data forms spherical clusters of similar size, which justifies the assignation step of each training item to the nearest centroid. This is in contrast to other clustering algorithms such as Expectation Maximization with a Gaussian mixture model, which can represent ellipsoidal clusters of different sizes and orientations.

The optimization of the approximation error is not such a big issue in the bag-of-words representation as in the compression of signals. Indeed, in our context the quantization is needed to obtain an intermediate representation of the image or video, i.e. a signature, which is later used as input of a supervised classification method. The performance of that classifier is the final objective. From this point of view, k-means clustering is unsupervised, as no information of the training categories is used to obtain the clusters.

Moreover, the trade-off between size of the vocabulary and final performance does not necessarily hold. A large vocabulary with very specific words representing only few descriptors of the input space may make the later learning task harder and harm final performance. In other words, some loss of information in the vector quantization process may be useful if it makes it easier for the classifier to learn and generalize. The trade-off would rather be between the representativeness of the words and their discriminative power. Ideally, we would like the words to carry some semantic meaning, as in the case original application of bag-of-words to categorize text and documents. Some authors have shown that a visual vocabulary can be transformed into another more compact yet discriminative one by fusing words according to their mutual information [65, 67] or co-occurrence [108]. In some computer vision applications, the size of the signatures may not even be an operational constraint.

The use of efficient discriminative vocabularies as in [80] is becoming more and more popular in many vision tasks. According to this technique, named Extremely Randomized Clustering Forest (ERC-Forest), a random forest is built from training descriptors and the obtained leaves are used as words. This is explained in more detail in Sections 4.2 and 4.3.

Random forests have also been applied to density estimation [15]. As such, they could be used for clustering, and therefore for unsupervised quantization.

Mu et al. [81] propose a technique called Random Locality Sensitive Vocabulary (RLSV) that addresses the computational limitations of k-means clustering when it comes to large training sets and high dimensionality. They defend the aggregation of different clusterings, as in random forests, as a means to enhance stability in high-dimensional spaces, and randomization as a means to more efficiently tackle large-scale problems. Instead of optimizing a global or local objective, their algorithm generates a sequence of random bi-partitions of the space of descriptors, based on hashing functions. Supervision is not needed. Any two samples belong to the same word if they lie on the same side for all bi-partitions. If B is the number of hashing functions, this corresponds to at most 2^B words, as not every partition intersects every other. Theory from the Local Sensitive Hashing domain guarantees that the probability that two descriptors collide in the same word is related to their pairwise distance. They combine several vocabularies generated this way.

Another method that uses random projections to generate vocabularies is [141], but this time in a supervised way. They encode features into *visual bits* optimized for each category. They integrate both the step of vocabulary generation and classifier training, by generating visual bits in a boosting-like manner.

4.2 The Decision Forest model

Random decision forests encompass a family of techniques that can be used to address several machine learning tasks. They became popular thanks to their success in classification and regression, but Criminisi et al. [15] present a unified model that also deals with density estimation, manifold learning, semi-supervised learning, etc.

A decision tree is a set of tests (or questions) that are applied to some input data in order to predict something about them. The tests are hierarchically organized: the first question is the same for all input objects, and subsequent questions depend on previous answers. The tree can be represented as a directed graph (Figure 4.1). The internal (*split*) nodes of the tree store the tests. Here we only consider binary tests, i.e. yes/no questions. The tree maps a given input object to one of the terminal (*leaf*) nodes. The ensemble of leaf nodes constitute therefore a partition of the input space into mutually exclusive regions.[1] Objects being mapped to the same leaf share some similarity, as they have satisfied the same tests. Each leaf stores a local predictor.

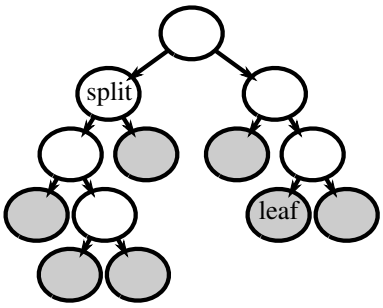


Figure 4.1: A decision tree can be represented by a directed graph. Its internal and terminal nodes are called *split* and *leaf* nodes respectively.

In the case of supervised classification and regression, training data are available with their corresponding target output, a.k.a. a *label*, which is discrete and unordered in classification, and continuous in regression. During the training stage, the dataset is used to build a function that will map new input data to their most probable label. The ability to map previously unseen data to their correct label is called *generalization*.

When decision trees are used for classification or regression, at *training* time the tests are chosen so as to split the data into subsets with similar labels. The chosen tests are stored in the nodes of the tree. At *test* time, a chain of tests is applied to a query datum starting from the root node, until a leaf is reached. Each question asked depends on previous answers, and tends to increase the certainty about the output label. The final decision on the query's label is based on the distribution of labels that was observed at training time in the reached leaf.

Decision forests combine the ideas of decision trees and ensemble methods. From the same training

[1] Tree-based quantization explained in the following sections is based on this property.

dataset, different trees can be generated by randomizing the test choice and/or the subset of data used to train each tree. It has been observed that aggregating such trees leads to better generalization, e.g. [30].

We formalize the Random Decision Forest model following mostly the notation in [15]. We choose the classification task to illustrate it (see also Figure 4.2).

4.2.1 Data point and features

A generic input object is denoted by a vector $\mathbf{x} = (x_1, x_2, ..., x_d) \in \mathcal{X}$, whose components x_i represent some attributes, called *features*. The *training set*, denoted by $\mathcal{S}_0$, is a collection of points that are available to build the forest. In the case of a classification task, the training set is labeled. Each training point is then denoted by a pair $(\mathbf{x}, c)$ where $c \in \mathcal{C}$ is the point's class label or category, with $\mathcal{C} = \{c_1, ..., c_K\}$.

4.2.2 Test or split functions

The test or split function at node j is a function of the feature vector and has a binary output,

$$h(\mathbf{x}, \boldsymbol{\theta}_j) : \mathcal{X} \times \mathcal{T} \longrightarrow \{0, 1\}, \tag{4.1}$$

where $\boldsymbol{\theta}_j \in \mathcal{T}$ denotes the parameters of the test function in the j-th node. A typical choice for test functions is the family of *axis aligned* functions, which applies a threshold to some component of the feature vector. The parameters of this family of tests are the component index and the value of the threshold.

At training time, the subset of training points $\mathcal{S}_j$ is available in this node. The parameters of the split function are chosen so as to optimize some objective function defined on $\mathcal{S}_j$ (see below). Once the function has been chosen, $\mathcal{S}_j$ is split into two subsets $\mathcal{S}_j^{\mathrm{L}}$ and $\mathcal{S}_j^{\mathrm{R}}$ corresponding to the left and right child nodes. This splitting process is repeated for each child node, unless the node satisfies some stopping condition: maximal depth of the tree allowed, too few training points, too low the score of the optimized objective function, etc.

Similarly, at test time the stored function is applied to a previously unseen point, which moves to the right or left child node depending on the output of this binary test, and this is repeated until a leaf is reached.

4.2.3 Objective or score function

During the training of node j, the parameters that optimize some objective function are selected and stored. When training decision forests, one way of obtaining diverse trees from the same training data consists of **randomizing** this optimization. Only a random subset of the parameter space $\mathcal{T}_j \subset \mathcal{T}$ is explored at node j. The amount of randomness is controlled by the ratio $\rho = |\mathcal{T}_j|/|\mathcal{T}|$. The resulting optimization problem is

$$\boldsymbol{\theta}_j^* = \arg\max_{\boldsymbol{\theta}_j \in \mathcal{T}_j} I_j. \tag{4.2}$$

The objective function takes values depending on the data in the node $\mathcal{S}_j$, and the way the data is split due to the value of $\boldsymbol{\theta}_j$. It is here denoted by I_j because it is typically based on *information gain*. In the classification setting, where each point is associated to a categorical label c, information gain is the

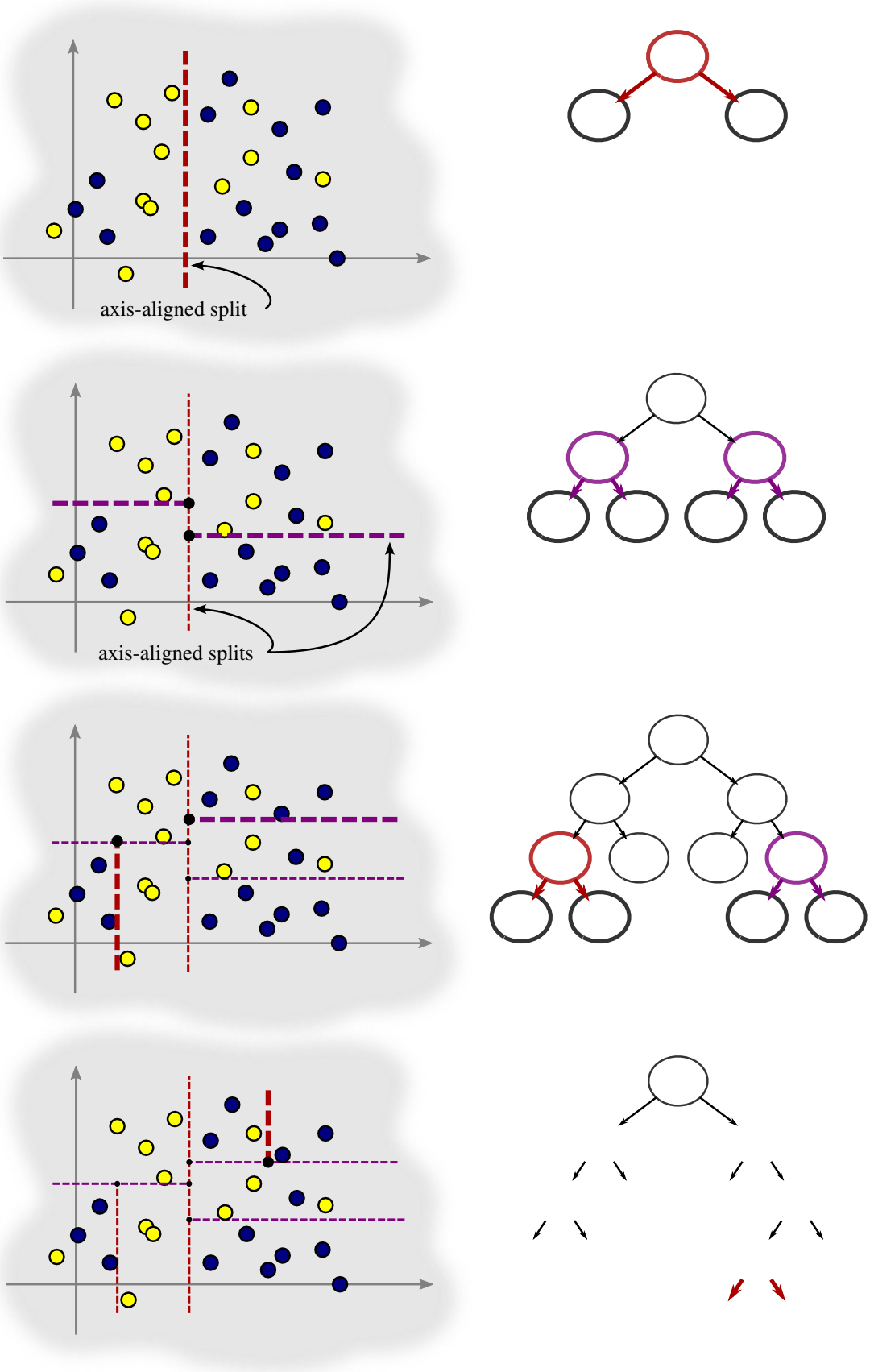


Figure 4.2: Training a decision tree with axis-aligned splits. For visualization purposes, the feature space $\mathcal{X}$ is represented with two dimensions only. Each data point has a class label, indicated by its color.

reduction in label *entropy* in the two child nodes compared to their parent node. It is defined in [15] as

$$I_j = H(\mathcal{S}_j) - \sum_{i \in \{\mathtt{L},\mathtt{R}\}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} H(\mathcal{S}_j^i), \tag{4.3}$$

where H is the entropy of c

$$H(\mathcal{S}) = -\sum_{c \in \mathcal{C}} p(c) \log p(c) \tag{4.4}$$

computed from $p(c)$, the *empirical* distribution of c in $\mathcal{S}$.

This measure builds on *information theory*. Consider the categorical label and the outcome of the binary test to be two random variables, C and T, taking values c and $t = h(\mathbf{x}, \boldsymbol{\theta}_j)$ respectively for each data point $\mathbf{x}$. The *mutual information* of two random variables measures their mutual dependence, i.e. how much knowing one of these variables reduces the uncertainty about the other. Mutual information of C and T can be written as

$$\begin{aligned}
I(C;T) &= \sum_{c \in \mathcal{C}} \sum_{t \in \{\mathtt{L},\mathtt{R}\}} p(c,t) \log \frac{p(c,t)}{p(c)p(t)} && (4.5)\\
&= \sum_{c \in \mathcal{C}} \sum_{t \in \{\mathtt{L},\mathtt{R}\}} p(c,t) \log \frac{p(c|t)}{p(c)} && (4.6)\\
&= -\sum_{c \in \mathcal{C}} \sum_{t \in \{\mathtt{L},\mathtt{R}\}} p(t|c)p(c) \log p(c) + \sum_{c \in \mathcal{C}} \sum_{t \in \{\mathtt{L},\mathtt{R}\}} p(c|t)p(t) \log p(c|t) && (4.7)\\
&= -\sum_{c \in \mathcal{C}} p(c) \log p(c) \sum_{t \in \{\mathtt{L},\mathtt{R}\}} p(t|c) + \sum_{t \in \{\mathtt{L},\mathtt{R}\}} p(t) \sum_{c \in \mathcal{C}} p(c|t) \log p(c|t) && (4.8)\\
&= H(C) - \sum_{t \in \{\mathtt{L},\mathtt{R}\}} p(t) H(C|T=t) && (4.9)\\
&= H(C) - H(C|T), && (4.10)
\end{aligned}$$

where $H(Z)$ denotes entropy of a discrete random variable Z and $p(z)$ its probability distribution:

$$H(Z) = -\sum_{z} p(z) \log p(z). \tag{4.11}$$

In a classification task, mutual information is indeed a meaningful measure: we want to design the split functions so that knowing the outcome of the test reduces the uncertainty about the label.

Going back to our formulation of the node's objective function, we see that mutual information (Eq. 4.9) is equivalent to Criminisi et al.'s definition of information gain (Eq. 4.3), replacing probability distributions by the empirical distributions extracted from the training points in $\mathcal{S}_j$. As said, this score measures the increase of certainty about a point label when we know to which side of the split, left or right, it corresponds. Intuitively, this objective function measures how well the test function splits the data into two subsets with homogeneous labels. When labels are mixed up in the resulting subsets, their entropy is high and the information gain is low.

Note that the entropies in each child node are weighted by the number of training points on each side. This means that very unbalanced splits (i.e. most points on one of the subsets) have low entropy gain, because the entropy in the most populated node is very similar to the entropy in the parent node (as

they contain mostly the same points) and the entropy in the other child has a very low weight. This score function is therefore implicitly discouraging very unbalanced splits.

Other entropy-based score functions are possible. For instance, the inventors of a variety of decision forest called Extremely Randomized Trees [30] suggest normalizing mutual information by the entropies of C and T. Their score function is given by

$$score(\mathcal{S}_j, \boldsymbol{\theta}_j) = \frac{2I(C;T)}{H(C) + H(T)}, \tag{4.12}$$

with the entropies computed from the empirical distributions of C and T in the set $\mathcal{S}_j$, e.g. $p(t) = |\mathcal{S}_j^t|/|\mathcal{S}_j|, \quad t \in \{\texttt{L}, \texttt{R}\}$. If we removed the normalization by the test entropy $H(T)$, the denominator would become independent of the split function and would have no influence in the optimization of Eq. 4.2. In that case, the solution $\boldsymbol{\theta}_j^*$ would be the same as using the score function based on information gain only.

4.2.4 Leaf and ensemble predictive models

At test time, a chain of tests is applied to a data point until it reaches a leaf. In a classification task, the leaf stores the empirical label distribution of the training points that reached it. This is the predictive model of one tree: to map each point to the empirical label distribution corresponding to the reached leaf, $p_t(c|\mathbf{x})$. In a forest, each data point reaches many leaves, one per tree. The predictive model of the ensemble of T trees is a combination of the predicted distributions, typically through averaging:

$$p(c|\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathbf{x}). \tag{4.13}$$

4.2.5 Main parameters

Assuming that the family of test functions has been chosen, as well as the objective function and input features, the remaining main parameters of this learning model are according to [15]:

- the amount of randomness when optimizing each node (ρ);
- the maximum allowed tree depth (D); and
- the number of trees (T).

Geurts et al. [30] present a particular case of the family of decision forests. They consider only axis-aligned test functions with a random threshold. In that case, the cardinality of the space of possible tests $\mathcal{T}$ is the dimension of the feature space d. In order to solve the optimization problem in Eq. 4.2, they only consider K candidate attributes randomly chosen among the d components of the feature vector. The amount of randomization is controlled by this parameter, and $\rho = K/d$. Rather than maximal depth, their smoothing parameter is minimum support $n_{\min}$, i.e. the minimum number of training points required to further split a node. The third parameter is the same, the number of trees or forest size, denoted by M in their work.

These parameters affect the forest's *predictive accuracy* or *generalization*, i.e. the ability to predict the right label when given new data point; the accuracy of its *confidence* in a prediction, i.e. how "peaky" the predicted distribution is for a given data point; and the computational efficiency [15].

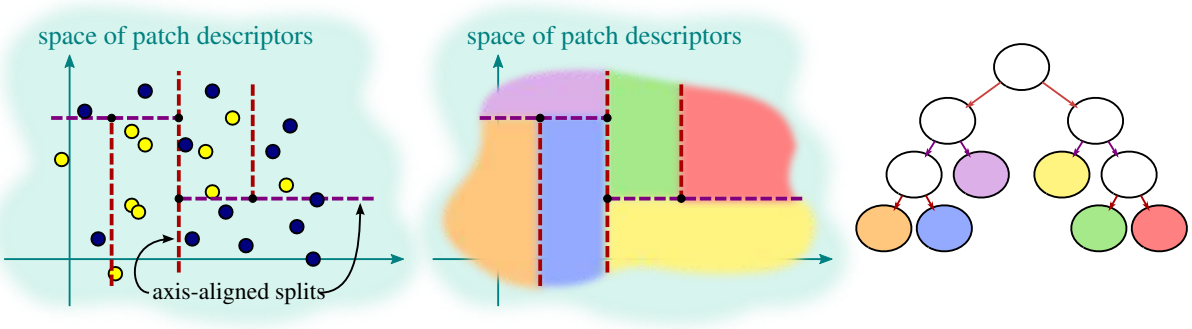


Figure 4.3: The space of patch descriptors, represented as a 2D space for visualization purposes. Each point is a patch descriptor. Its color represents its "hallucinated" label. A decision forest can be used to make a partition of this space. Each leaf is a word of a vocabulary of local descriptors.

For example, if we let a single tree grow very deep (or similarly let their minimum support to be low) we end up with leaves containing very few training points, and therefore the predictive distributions are less reliable. In the extreme case, with only one data point per leaf, the predicted distribution is concentrated on a single class, leading to an overestimated confidence in the prediction. As justified in [30], this negative effect is largely compensated by a sufficiently high number of trees T. Because the trees are diverse, they produce different output distributions that are averaged, reducing variance and making the final prediction more reliable, both in terms of accuracy and confidence. They choose for their classification experiments $n_{\min} = 2$ (fully grown trees), and they find $T = 100$ to be a sufficiently large number of trees to obtain convergence of the ensemble effect.

Criminisi et al. [15] agree on the observation that accuracy increases with the forest size T. However, they warn against fully grown trees, which may lead to overfitting, as well as against too shallow trees, which may produce low-confidence posteriors. They argue that using multiple trees alleviates indeed the problem of overfitting, but it does not cure it completely. Depth would therefore remain an important parameter to be carefully chosen.

When restricting the test functions to be axis-aligned, Geurts et al. [30] find $K = \sqrt{d}$ and $K = d$ to be good values for their classification and regression experiments, respectively.

4.3 Using random forests for local-feature quantization

As explained in the previous section, a decision tree is more than a classification function: it also constitutes a partition of the input space into non-overlapping regions. Each of these regions corresponds to a leaf of the tree and contains points with common attributes. Let us consider the input space of a decision forest to be the space of descriptors of image (or video) patches. A decision forest can then be applied to such local descriptors to obtain a visual vocabulary, as in Figure 4.3. Each leaf is a word in the vocabulary, and each local descriptor is mapped to several words (one per tree). This vocabulary can be used to obtain bag-of-words signatures of images as explained in Section 3. In practice, each tree produces one histogram, and they are concatenated to obtain the final signature, see Figure 4.4.

This is an important difference between forests for classification and forests for quantization. In a pure classification task, increasing the number of trees has remarkable positive effects, such as canceling

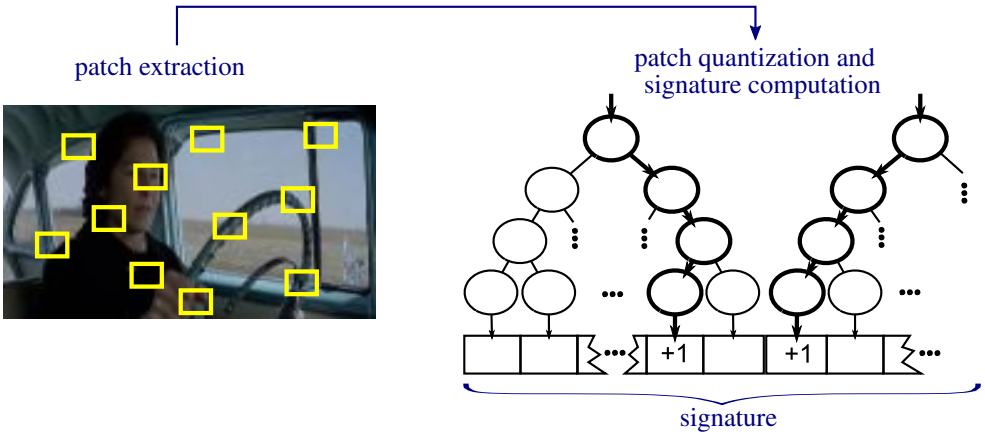


Figure 4.4: Bag-of-words representation of a video using a random decision forest.

out the variance of individual trees (due to their random nature) thanks to the averaging effect. In the bag-of-words case this cancelation is less obvious, as signatures are not averaged but concatenated.[2]

The second difference between decision forests for classification and for local-descriptor quantization is the availability of a category label per data point $\mathbf{x}$. In the classification setting, $\mathbf{x}$ is the entity to be classified, i.e. for which we want to predict a category. This category is available at training time. In the quantization setting, $\mathbf{x}$ represents a local descriptor, which we want to assign to a word. The ultimate goal is to perform supervised image (or video) classification, so we do have a category label for the image overall. However this label is not available for every local patch (and we do not try to predict it for every patch either).

Without any supervision, the individual trees would be totally random and have high variance, which cannot be canceled out through averaging (as explained previously) and could harm performance. The way to address this issue in practice is to consider that each patch "inherits" the category label of the image it belongs to –what we call a "hallucinated" label– as a means of weak supervision, see Figure 4.3. This particular use of random decision forests to obtain visual vocabularies is proposed by [80] and named ERC-Forests. Contrary to k-means clustering, these vocabularies are obtained in a supervised way, i.e. taking the target output label information into account.

Compared to k-means, the obtained clusters do not tend to be spherical or equally sized. Besides, there are more parameters that control the total number of bins in the signature: the number of trees and the number of leaves per tree –or, similarly, the maximum depth of the trees or the minimum support of a node. As in the case of k-means clustering, there is a trade-off between the representativeness of the words and their discriminative power. In the case of the forests, this is controlled by the maximal depth D. Increasing the number of trees T is likely to increase final performance thanks to the complementary information introduced by each tree's signature. This is however at the cost of a much longer signature, which may be problematic in terms of computational efficiency of later steps in the pipeline.

Benchmarks of these two methods regarding computational cost, memory needs and performance can be found in [80, 81]. The training costs are given by [81] in Big-$\mathcal{O}$ notation. Assuming that we

[2]Indeed, it would not make sense to average bin by bin, as trees correspond to different unrelated partitions of the space.

desire to obtain k words, and that we have n training descriptors of dimensionality d, k-means requires $\mathcal{O}(knd)$ operations per iteration, and typically tens of iterations to converge. In the case of hierarchical k-means with a branching factor of two, the cost is $\mathcal{O}(2nd \log_2 k)$. In the case of a tree, assuming that roughly the number of levels needed is $\log_2 k$ and that $\sqrt{d}$ splits are tested at each node as in [30], the training cost is $\mathcal{O}(n\sqrt{d} \log_2 k)$. If we consider several trees to obtain the same desired number of words, the cost is $\mathcal{O}(n\sqrt{d}T \log_2 \frac{k}{T})$. The cost to quantize a new descriptor once the vocabulary has been built is $\mathcal{O}(knd)$ for k-means, $\mathcal{O}(2nd \log_2 k)$ for hierarchical k-means and only $\mathcal{O}(\log_2 k)$ for a tree.

Table 4.1: Properties of k-means vs. ERC-Forests

k-means with Euclidean distance	**ERC-Forests**
- Resolution is tuned through parameter k (number of clusters). - All components in the descriptor have the same weight.	- Resolution is tuned through the smoothing parameter (min. support or max. depth). - Supervised. Capability to filter out noisy components.

4.4 Quantizer Forests: a dedicated objective function

The underlying hypothesis of decision forests for quantization (such as ERC-Forest [80]) is that *words* can be discriminative individually (that is the point of trying to split descriptors into subsets with homogeneous labels) and that weak supervision through "hallucinated" labels is enough to obtain the desired effect of increased performance.

The problem can be seen from a different point of view, keeping in mind that our final target is to obtain discriminative *signatures*. Let us recall the bag-of-words pipeline and the role of the forest for quantization. We first sample local patches from training videos and compute their descriptor. Then we use the decision forest training algorithm to obtain a visual vocabulary. The forest acts on local patches, i.e. the split functions induce a partition on the space of local descriptors. A signature for any video can then be obtained by passing through the forest every local patch and counting the number of patches in each leaf.

All in all, we use forests to compute histograms[3]. Every leaf in the forest is a bin in the signature. For a given video, the value of a signature bin is the number of local patches from that video that landed in the corresponding leaf. During the forest training stage, every time we split a node, we are replacing one bin by two new bins and redistributing the counts of points between the two child nodes. Video signatures are implicitly changing "on the fly" with every new split.

ERC-Forests attempt to separate local patches regarding their hallucinated label. The standard split criterion looks at the distribution of category labels over all training points. They are all taken into account independently to compute node entropies, regardless of whether they belong or not to the same

[3]Other similar kinds of signature are also envisageable.

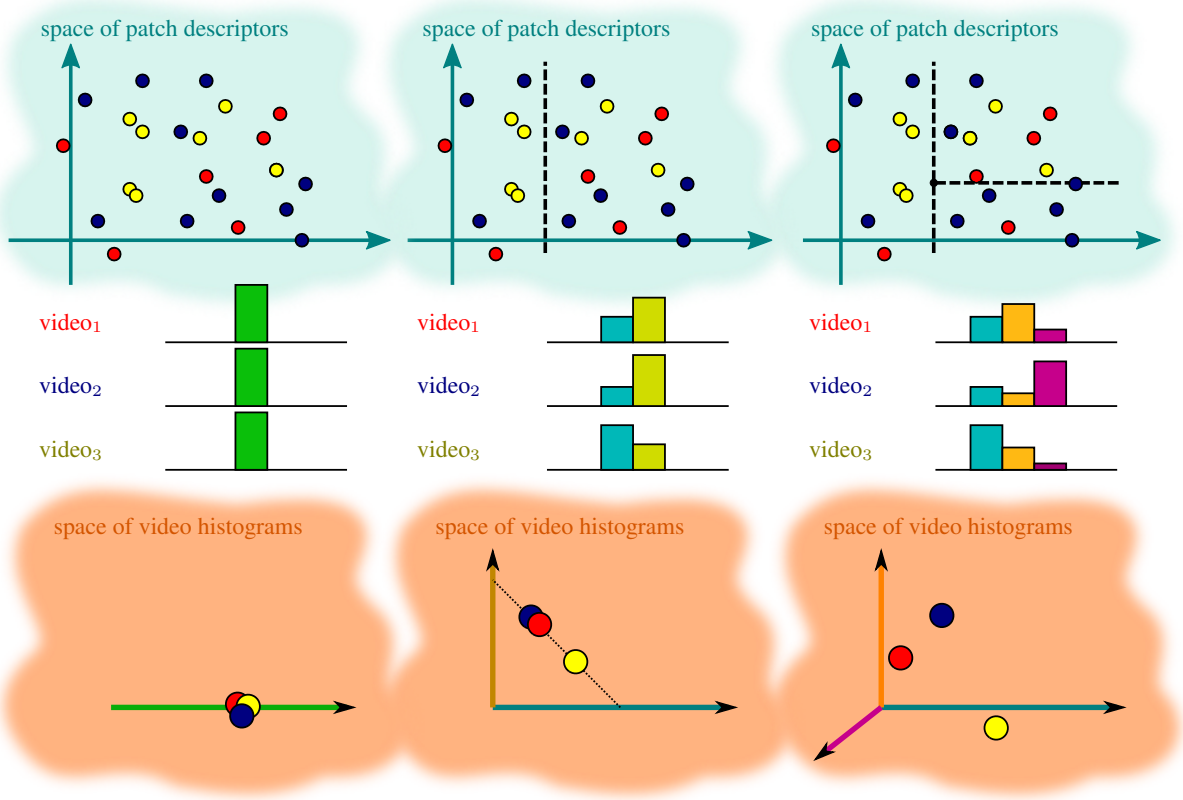


Figure 4.5: Splitting the patch-descriptor space causes an impact on the video signatures and their arrangement. Here a 2D patch patch-descriptor space is represented, with patches extracted from three videos belonging to three different categories (blue, red and yellow). Each split in this space generates an additional bin in the final video signatures. We care about the arrangement of video signatures, rather than the category entropy of local patches.

video. The training stage is trying hard to split descriptors into subsets with homogeneous labels, as if we wanted to classify the local descriptors themselves, whereas our ultimate objective is to classify videos, not local patches.

Rather than looking at the entropy distribution of local patches, we would like to look at the signatures we obtain with different candidate splits, and select those splits that facilitate signature (i.e. clip) classification, see Figure 4.5. Instead of caring about the separability of individual patches, considered independently, and trying to get discriminative individual words, we focus on the separability of signatures (which aggregate individual patches' contributions) and trying to get discriminative signatures overall.

In order to make this work, we need to enlarge each training datum with a label indicating to which video the patch belongs: $(\mathbf{x}, v(\mathbf{x}), c(v))$ with $v \in \{1, \ldots, V\}$, where V is the number of videos and $c \in \mathcal{C}$ is the category label of video v. For a given state of the tree, the video signature of video w is a histogram $\mathbf{b}_w = (b_{w1}, \ldots, b_{wj}, \ldots, b_{wn})$ in which each bin b_{wj} contains the counts of points at leaf node j:

$$b_{wj} = |\{\mathbf{x} \in \mathcal{S}_j \mid v(\mathbf{x}) = w\}| \,. \tag{4.14}$$

4.4.1 Ideal case and approximations

We propose to use forests to split the descriptor space while measuring the effects in the signature space. Roughly, we want videos with the same class label to be close in the signature space. Ideally, we would like the node objective function to be based on the performance of a classifier of video signatures.

First approximation

The objective is to maximize the performance of the classifier (the last stage of our pipeline, see Section 3) on the final signatures obtained through some quantization process. The first approximation is inherent to forest structure and training procedure. Early decisions are made near the root, leading to a hierarchical coarse-to-fine clustering and not necessarily the optimal signatures.

Optimizing the full final signature is a combinatorial problem, which could be feasible through exhaustive search only for a very low number of leaves. Imagine the root node and a target signature size of two bins. The number of operations would be only linear in the size of possible tests $\mathcal{T}$. If the target number of leaves is three, then for each candidate split of the root we should test all candidate splits of the left and right child nodes and pick the best combination of root-child splits. In general, the number of possible binary trees with $n+1$ leaves, is given by the n-th Catalan number [112],

$$C_n = \frac{1}{n+1}\binom{2n}{n} \quad \text{for} \quad n \geq 0. \tag{4.15}$$

It grows asymptotically as $C_n \sim 4^n/(n^{3/2}\sqrt{\pi})$. Each of the C_n possible trees with $n+1$ leaves has n internal nodes, and for each internal node we would evaluate $\rho|\mathcal{T}|$ candidate splits. All this without considering that a forest has several trees, so there are even more combinations to obtain a given number of leaves.

Instead, we stick to the classic greedy approach, which has already been proven to have satisfactory performance [80]. We build our trees from root to leaves, making only local decisions at the level of each node, eventually using information from the "already-split" nodes and the rest of "current leaves". The assumption is that finer signatures can be obtained from coarser signatures by subsequent splits of the forest nodes, in a way that pursues and eventually reaches a (local) optimum of their performance.

Second approximation

As explained at the beginning of Section 4.4, every new node split modifies implicitly the space of video signatures. We would like to evaluate the performance of the final classifier on those "dynamic" signatures. Our final classifier is typically a Support Vector Machine (SVM). Its performance can be estimated by cross-validation on the training set. To save time and computation, we could also separate a validation set from the training set. In both cases, any candidate split would imply the modification of all signatures in the training set, and therefore at least one (or k, in the case of k-fold cross-validation) training and evaluation process, and eventually the corresponding model-selection with regards to the SVM parameters. Note the extra computation time that this represents compared to a normal decision forest. It is impractical to repeat this process for every candidate split. We hypothesize, nevertheless, that it is possible to assess whether the candidate splits encourage an arrangement of videos in the signature space that makes the classification task easier.

To do so, we propose a new objective function and several training algorithms to put it in practice. This objective function aims explicitly at building forests that lead to discriminative signatures.

4.4.2 Objective function

We propose to solve at node j the optimization problem

$$\boldsymbol{\theta}_j^* = \arg\max_{\boldsymbol{\theta}_j \in \mathcal{T}_j} A_j, \tag{4.16}$$

where A_j represents accuracy gain.[4] This gain depends as usual on the data in the node $\mathcal{S}_j$ and on the split function parameters $\boldsymbol{\theta}_j$. Strictly speaking, it also depends on the current state of the tree (the current leaves, and the corresponding implicit video signatures).

Consider that at the time of training node j, the tree has L current leaves, so the current implicit signature is L bins long. Node j is also a current leaf. We divide the computation of A_j into two steps: **(i)** pre-split accuracy A_j^L, which is the accuracy achieved using the current L-bin-long signatures; and **(ii)** post-split accuracy A_j^{L+1}, the accuracy achieved splitting $\mathcal{S}_j$ according to $\boldsymbol{\theta}_j$ and using the resulting $(L+1)$-bin-long signatures, so

$$A_j(\mathcal{S}_{1:L}, \boldsymbol{\theta}_j) = A_j^{L+1}(\mathcal{S}_{1:L}, \boldsymbol{\theta}_j) - A_j^L(\mathcal{S}_{1:L}) \tag{4.17}$$

$$= A_j^{L+1}(\mathcal{S}_{1:L\setminus j}, \mathcal{S}_j^{\mathrm{L}}, \mathcal{S}_j^{\mathrm{R}}) - A_j^L(\mathcal{S}_{1:L}), \tag{4.18}$$

where $\mathcal{S}_{1:L}$ and $\mathcal{S}_{1:L\setminus j}$ denote all current leaf nodes, respectively including or excluding node j.

We first compute the pre-split accuracy, which is the same for all candidate splits of node j. We base our accuracy measure on pairwise distances between video signatures (Section 4.4.4). Then, for each candidate split, we compute the number of points of each video that go to the left and right child nodes, and compute the impact on the pairwise distances. We compute post-split accuracy for each candidate split and keep the one with best gain.

4.4.3 Our accuracy measure

As discussed in Section 4.4.1, we would like to assess the quality of the arrangement of video signatures using the same performance measure as the one we use to evaluate the full system. However, training and evaluating an SVM to calculate accuracy would be impractical.

A k-nearest-neighbor technique would be more feasible: we would consider one training video at a time, consider it as a "query" video, take its k nearest neighbors among the remaining training videos, and predict a category based on the most frequent class among the k videos. Accuracy would then be the fraction of good predictions compared to the ground-truth labels. Two disadvantages of this approach are that kNN accuracy depends on parameter k and it fails to capture the overall arrangement of videos. When comparing pre- and post-split accuracy on a given query video, it only takes into account the displacements of videos across the k-th neighbor barrier, and might be blind to other favorable rearrangements of videos.

[4] Due to the second approximation, this accuracy is not the actual SVM accuracy but an alternative measure, defined in Section 4.4.3.

We propose yet an alternative accuracy measure inspired by the classical Average Precision used detections tasks (see Section 3.5.1 for details). As in kNN we take one training video at a time as query. We then rank the remaining videos according to their distance to the query video, in increasing order. Consider a detection task in which the positives to be detected are all the training videos with the same label as the query video. Furthermore, consider the two-class classifier which would be the thresholded distance (less distance meaning more confidence in the detection). We would like such a classifier to have high precision and high recall, meaning that most videos with the same label as the query video are close to it and high in the ranking. Different thresholds produce different precision and recall rates, just like in a detection task. In this way, we can compute an AP per video. Our accuracy measure can then be obtained by averaging this AP over all training videos.

4.4.4 Distances between video signatures

We base our accuracy measure on pairwise distances between video signatures.

$$D(v,w) = D(\mathbf{b}_v, \mathbf{b}_w) \tag{4.19}$$

$$= \sum_{j=1}^{L} \frac{(b_{vj} - b_{wj})^2}{\frac{1}{2}(b_{vj} + b_{wj})} \qquad v,w \in \{1,\dots,V\} \tag{4.20}$$

We prefer χ^2 symmetric distances because our final SVM uses kernels based on those, but l_1 or l_2 distances would be acceptable choices too.

With every new split, all bins in the signature remain the same, except the bin corresponding to the split node. That bin disappears and is replaced by two new bins, whose values for each training video are given by the counts of points. Updating pairwise distances between videos amounts to removing the term corresponding to the split node and adding the two terms corresponding to the child nodes.

We propose two ways to train quantizer trees using our custom objective function.

Breadth-first manner

If the tree is trained in a breadth-first manner, the distance can take into account the full current L-dimensional signatures for the pre-split stage, and the $(L+1)$-dimensional signatures for the post-split stage, as conveyed by Eq. 4.20. As long as each bin contributes an additive term to the distances (e.g. no square root in l_2 distances), a global pairwise distance matrix can effectively be updated after each split by subtracting the contribution of the old node and adding the contributions of the two child nodes, without having to recompute the contributions of all other dimensions of the signature. The disadvantage of this approach is that the development of each node depends on the evolution of the tree up to present time, i.e. the winning splits of a node may depend on what happened in other branches. Therefore, this tree-training algorithm is not parallelizable.[5]

Depth-first manner

The are remarkable computational advantages to train each node independently of every other, just with the data that is present in the node. When using the previous algorithm we consider that a split might

[5]The different trees in the forest can still be computed in parallel.

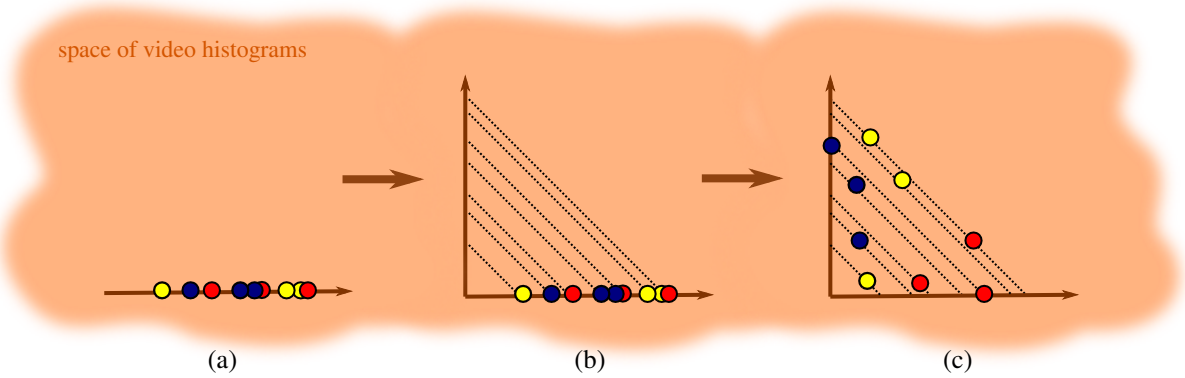


Figure 4.6: Illustration of the split of one node in a Quantizer Forest. **(a)** Distribution of videos along the one dimension corresponding to the node to split. Each point is a video, and its color represents its category label. **(b)** *Any* split of the node will make the videos move along the dotted lines. This is because each dimension in a bin in the histogram, and splitting the node in two will generate two bins whose sum is equal to the value in the original bin. **(c)** Among all possible splits, the aim is to choose a split that produces an arrangement of points that facilitates classification, keeping closer those points belonging to the same category.

be good because of its complementarity with regards to the existing full signature. The following assumption can be made that enables to keep the training of each node independent from the rest: when it comes to comparing candidate splits of the same node, the most relevant changes in accuracy can be noticed independently of the rest of the signature. Instead of looking at the L- and $(L+1)$-dimensional signatures, we take for the pre-split stage the distances in the 1-dimensional space corresponding to the current node, and take the distances in the resulting 2-dimensional space for the post-split stage.

We focus then on the given node, which represents one dimension in the signature space, see Figure 4.5. For each training video, we compute the value of that single signature bin by looking at the training points contained in the node and their corresponding video labels. A candidate split would transform this 1D space into a 2D space. We compute the 2D sub-signatures corresponding to the candidate child nodes. We prefer a split in which the resulting 2D space is such that the videos are easier to classify according to our classification accuracy measure. Roughly, we prefer splits such that videos of the same category are kept closer in the resulting 2D space.

Doing so simplifies the dependencies of pre- and post-accuracies of Eq. 4.17:

$$A_j(\mathcal{S}_j, \boldsymbol{\theta}_j) = A_j^2(\mathcal{S}_j, \boldsymbol{\theta}_j) - A_j^1(\mathcal{S}_j) \tag{4.21}$$

$$= A_j^2(\mathcal{S}_j^{\mathrm{L}}, \mathcal{S}_j^{\mathrm{R}}) - A_j^1(\mathcal{S}_j), \tag{4.22}$$

and reduces Eq. 4.20 to one and two terms.

4.4.5 Obtaining balanced trees

This objective function does not take into account the number of points that go left of right. This could lead to very unbalanced trees, which we have observed to be harming the performance of the quantization, as also warned by [15]. In order to avoid this, we restrict our thresholds to be the median of the values, randomness still remaining in the choice of other split function parameters (the choice of feature

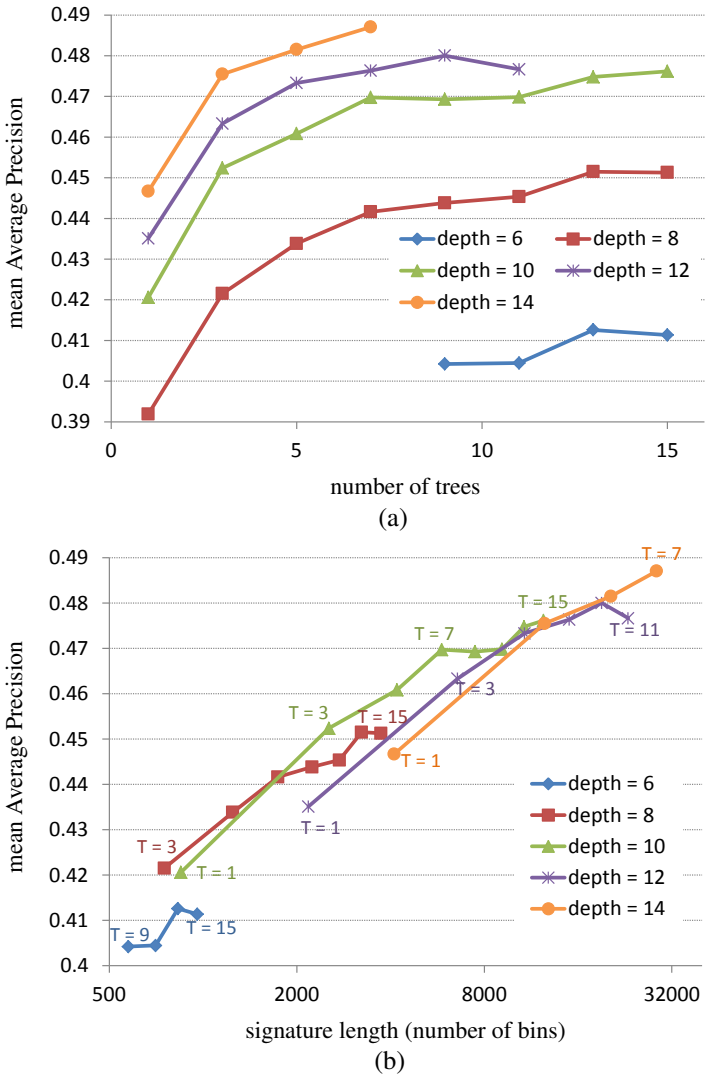


Figure 4.7: Performance obtained on Hollywood2 using random forests with increasing number of trees and depth.

in the case of axis-aligned tests). In this way, we force the algorithm to choose high-scoring splits only among those leading to balanced trees.

4.5 Experimental evaluation

Unless otherwise stated, we train forests with 5 trees, maximal depth of 14 and minimum support of 200 descriptors, with axis-aligned splits. Out of the d dimensions of the input points, we randomly select $\sqrt{d}$ at each node, and for each dimension we uniformly sample two thresholds between the minimum and maximum values of this dimension according to the training descriptors in the current node.

4.5.1 Influence of depth and forest size

When using forests for quantization, there are two parameters that affect the length of the resulting signature: maximum depth and number of trees. The same number of leaves can be obtained through different combinations of those two parameters. For example, in order to roughly double the length of the

signature, one could grow the forest one level deeper, or could double the number of trees. A reasonable question to ask is which of the two options to make signatures longer has the greatest impact on the final performance.

To gain some insight to this question, we train a forest with 15 trees and maximum depth of 14 using, then slice it at different depths and different number of trees, and evaluate the performance of the resulting signatures. For this experiment, we use the Hollywood2 dataset with its standard training/testing partition and STIP features, and we train a one-vs-all SVM with χ^2 kernels and fixed regularization parameter $C = 100$.

Figure 4.7(a) shows the impact that has adding more trees. On this dataset, going from three to one tree consistently causes a drop in performance, for all tested depths. Performance increases considerably from three to seven trees. From there on, it still increases but at a lower rate. In Figure 4.7(b) we can see how those values of depth and number of trees translate into signature length. From 8000 bins on, many points lay very close, meaning that different combinations leading to similar signature length have similar performance.

From these figures we could conclude that going from one tree to several is worthwhile, and sticking to only a few (between 5 and 10) is already enough to be on the right track. Regarding the depth, going from eight levels to ten seems to make a considerable difference.

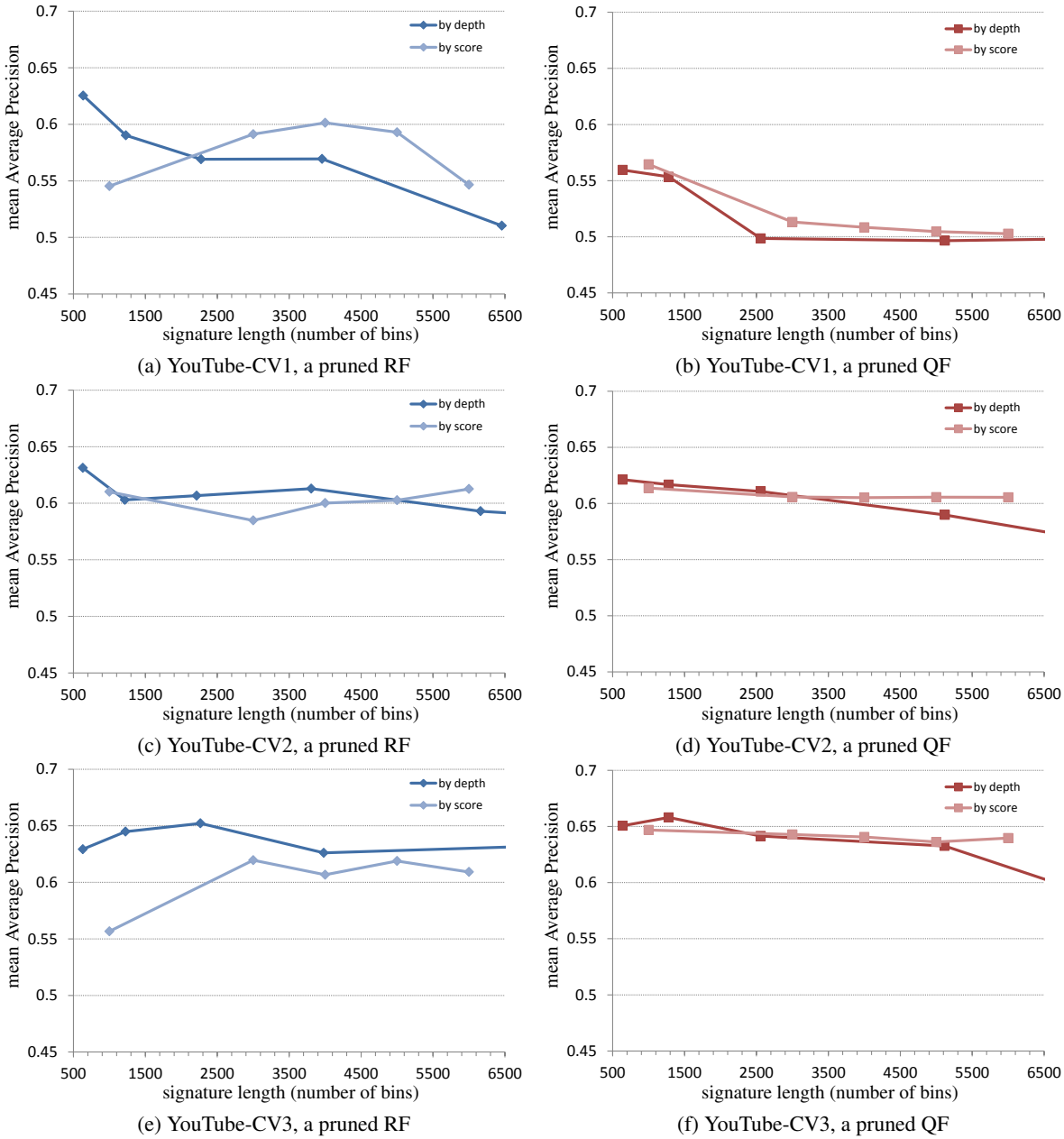
4.5.2 Pruning based on training score

We define three ways of pruning a trained tree. The first one is purely based on tree *depth*: nodes exceeding a given maximum depth are removed. The second one is based on training *score*: we go through the tree starting from the root, and we greedily select the nodes that had a higher score at training time, until a given number of leaves is reached. The third one is based on *weighted score*, and is a variant of pruning by score which would also account for the number of training points that went through the node. The idea behind it is to greedily give preference to the splits that lead to high scores on a large fraction of training points. The second and third pruning techniques enable to precisely tune the resulting number of leaves. Note that the three techniques run very fast once the tree has been trained.

In order to explore the influence of the pruning technique, we train a forest on some dataset, prune that forest by score and by depth, compute clip signatures according to the two different pruned forests, and evaluate the classification accuracy of the two sets of clip signatures in our standard supervised classification setting: one-vs-all SVM with χ^2 kernels and fixed regularization parameter $C = 100$.

Figure 4.8 shows the result of training a traditional random forest (RF) and one of our quantizer forest (QF) on STIP features of different training/testing partitions of the YouTube dataset. Each forest has been pruned by score and by depth in order to obtain different signature lengths. Following a similar experimental procedure, Figure 4.9 shows the average classification performance of five runs of RF and five runs of QF on STIP features of the Hollywood2 dataset. We see that depth works better on Hollywood2, but pruning by score may improve RF and QF performance on many partitions of YouTube.

The weighted-score pruning is not tested here, and could also be a means of improvement. An interesting object of further research would be the relationship between the performance of difference



(a) YouTube-CV1, a pruned RF

(b) YouTube-CV1, a pruned QF

(c) YouTube-CV2, a pruned RF

(d) YouTube-CV2, a pruned QF

(e) YouTube-CV3, a pruned RF

(f) YouTube-CV3, a pruned QF

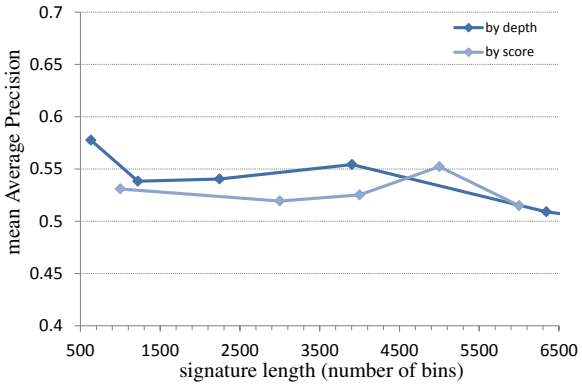
Figure 4.8: Random forests and quantizer forests pruned by depth vs. pruned by score, for different partitions of YouTube data. *(Continues...)*

pruning techniques and the balance of the resulting trees.

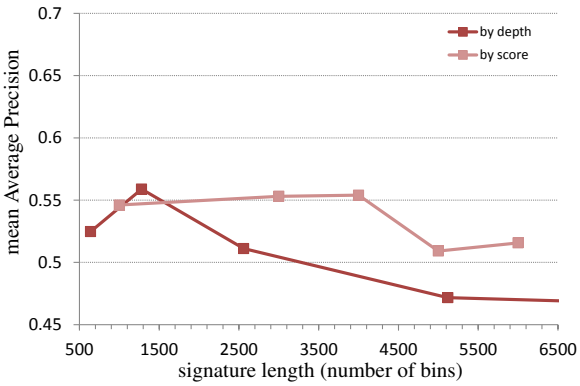
4.5.3 Traditional random forest vs. quantizer forest

In this section we compare the performance of the traditional random forest (RF) and our quantizer forest (QF) on different dataset and feature types.

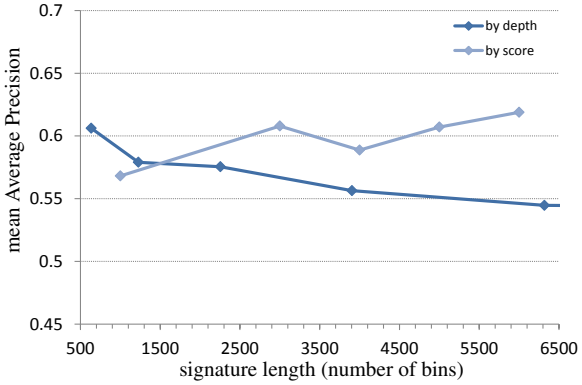
In the fist experiment, we use the KTH training and testing videos as described in Section 3.3. We use $2 \cdot 10^5$ STIP features to train five different RF and five different QF. We obtain a range of signature lengths by pruning each forest at different depths, from 2 to 14. For each pruned forest, we compute signatures of the training and testing clips, which we use to train a one-vs-all SVM with χ^2 kernels. For



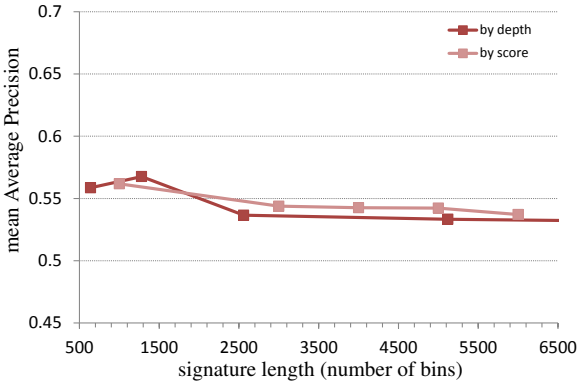
(g) YouTube-CV4, a pruned RF



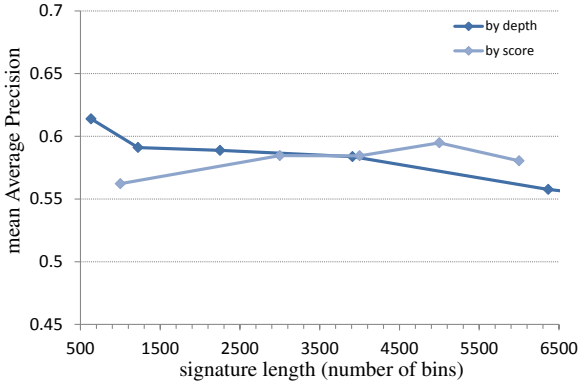
(h) YouTube-CV4, a pruned QF



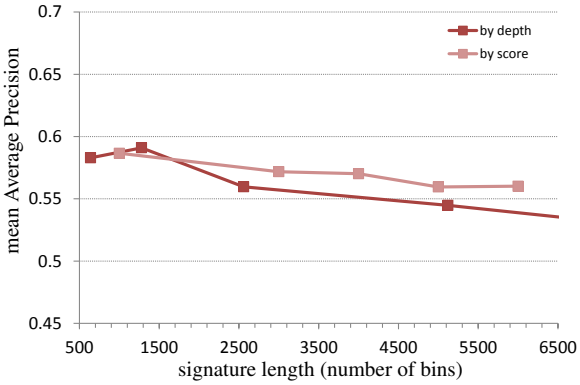
(i) YouTube-CV5, a pruned RF



(j) YouTube-CV5, a pruned QF

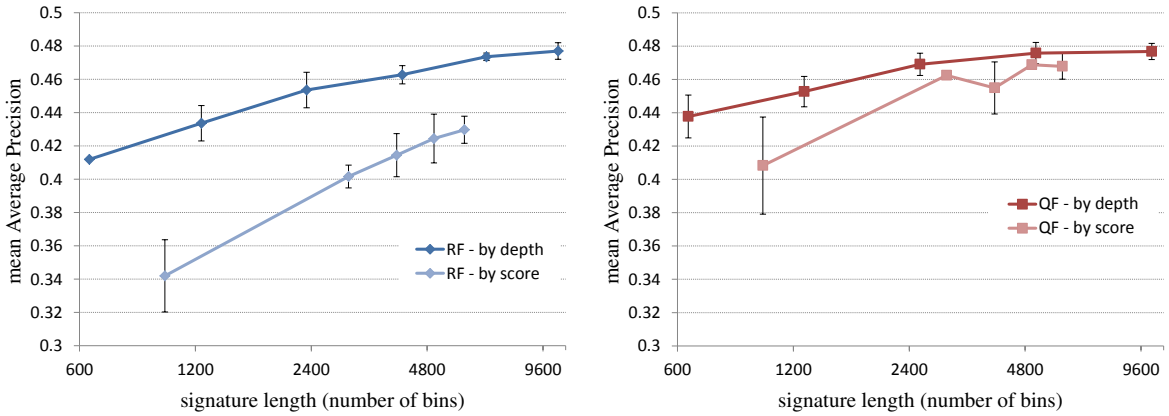


(k) YouTube, average of the pruned RF of the five partitions



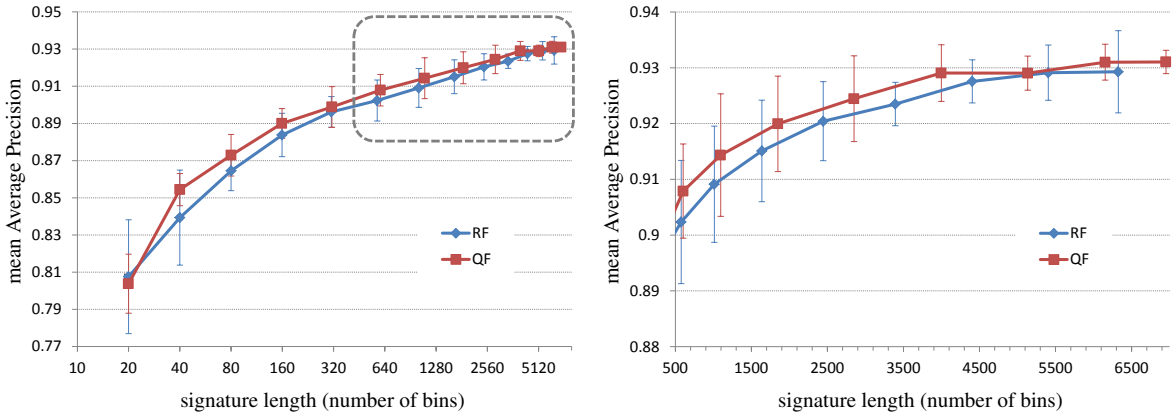
(l) YouTube, average of the pruned QF of the five partitions

Figure 4.8: *(...continued)* Random forests and quantizer forests pruned by depth vs. pruned by score, for different partitions of YouTube data.



(a) With Random Forests. (b) With Quantizer Forests.

Figure 4.9: Comparison of pruning by depth and by score on Hollywood2.



(a) (b)

Figure 4.10: Comparison of the performance of random forest and quantizer forest on KTH dataset using STIP features. Average and 95%-confidence intervals over five runs of each type of vocabulary. Note the logarithmic scale of the signature-length axis in figure (a). The detail inside the dotted rectangle is shown in figure (b). See text for more details.

each of the two-class classifiers, we choose a value of the regularization parameter $C \in \{1, 10, 100\}$ for each class through 5-fold cross-validation, and evaluate its average precision (AP) on the testing signatures. Our multiclass performance measure is the mean AP of the six action categories (see Sections 3.4 and 3.5.1 for more details). Figure 4.10 shows the performance score obtained by RF and QF for varying signature lengths. Points in the graph represent the average of the five runs, and error bars give its 95%-confidence interval. The QF curve is slightly superior, particularly at length 40, where the average performance is higher and the interval of confidence is tighter. Overall, the differences are not statistically significant.

In the next experiment, we use again the Hollywood2 dataset, with the same features and classifier as in Section 4.5.1. We compare traditional random rorests (RF) and our quantizer forests as well as k-means, for different vocabulary sizes. In the case of the forests, different sizes are obtained by slicing the trained forest at different depths. K-means is not hierarchical here, so different sizes have been obtained

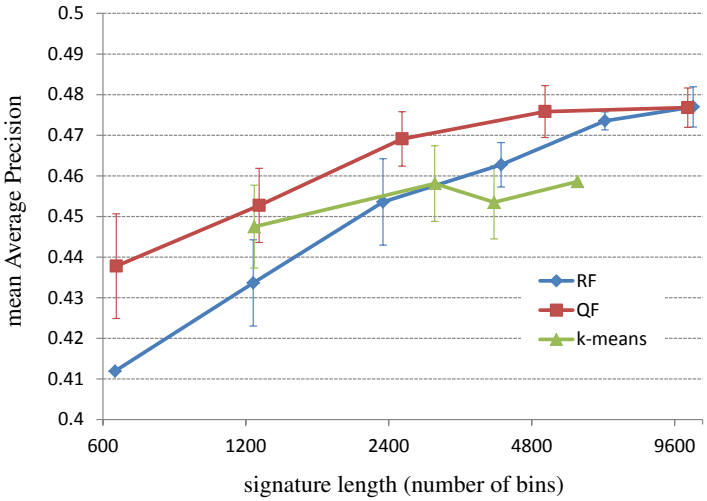


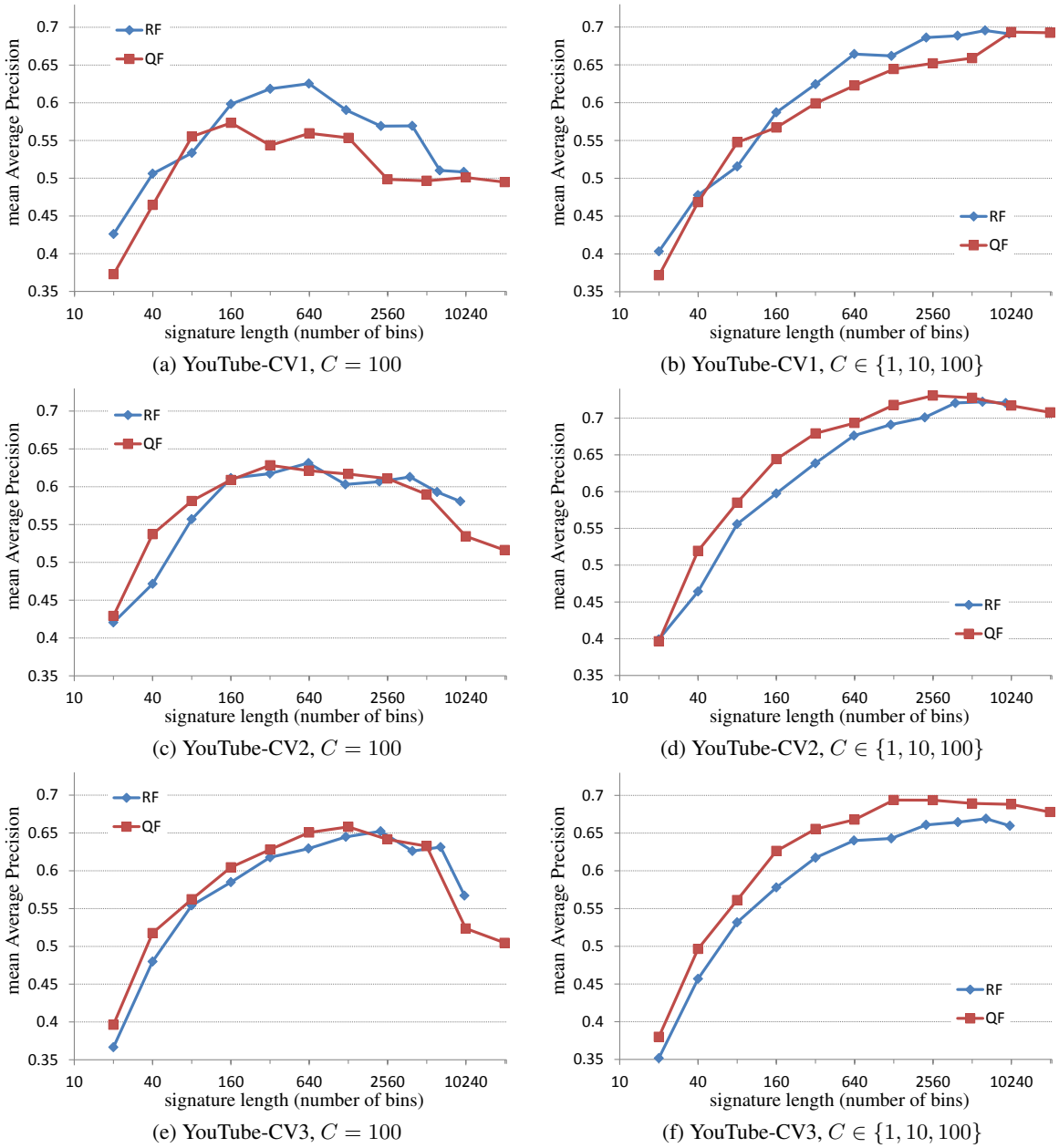
Figure 4.11: Comparison of the performance of random forest, quantizer forest and k-means on Hollywood2 dataset using STIP features. Performance is measured as the mean AP over all categories of actions. Each point in the chart shows the average performance and 95% confidence intervals over five runs of each type of vocabulary. See text for more details.

through independent trainings. Due to its high computational cost, we are able to use only 10^5 training points for k-means, whereas we can easily use 10^6 to train the forests. Each training has been repeated five times. Figure 4.11 presents the average and 95%-confidence intervals of the five runs. We can see that our QF works better that RF on this dataset and features. While the performance increase seems not very big in absolute termns, it is actually statistically significant. It can also be noted that the difference in performance is higher for shorter signatures, which has positive computational implications. This is the kind of situation in which the longer training of our QF pays off in the form of more compact and better performing video signatures. On the k-means side, the independent training for different vocabulary sizes makes it harder to see an improving tendency as the number of clusters is increased, and was very slow to train.

We also compare RF and QF on different training-testing partitions of the YouTube dataset, as described 3.3. We used STIP features and the usual classifier. On the left column of Figure 4.12, we can see the results if we use a fixed regularization parameter $C = 100$. We can see that while this value works well for the STIP signatures of Hollywood2, it does not suit the STIP signatures of YouTube. On the right column, we chose $C \in \{1, 10, 100\}$ by 5-fold cross-validation, which corresponds to a more realistic scenario in which the user tries to make the most out of the signatures. QF is again superior in most partitions and in average.

Finally, we compare again RF and QF on Hollywood2 and YouTube, but this time using dense MBH features, described in Section 3.1.2 and used more extensively in Chapters 6 and 7. These features are much denser than STIP. We show here the results of training the forest with $2 \cdot 10^5$ descriptors[6]. We choose $C \in \{1, 10, 100\}$. Results are shown in Figures 4.13 and 4.14. In this case, the superiority of QF regarding signature length is not obvious. If we instead align the scores by depth, we can see that our algorithm is still doing what it is designed to do: at each step (depth level by depth level, node by node) it tries to choose the splits that lead to higher classification accuracy. However, in this case the extra computational training effort does not pay off as in the other cases. Note that our QF use

[6]This sample is less representative of the variety of local MBH training descriptors than it could be, and thus constrained the forests in this experiment to be shallower. Further experiments should ideally use a larger training sample, e.g. 10^6.



(a) YouTube-CV1, $C = 100$

(b) YouTube-CV1, $C \in \{1, 10, 100\}$

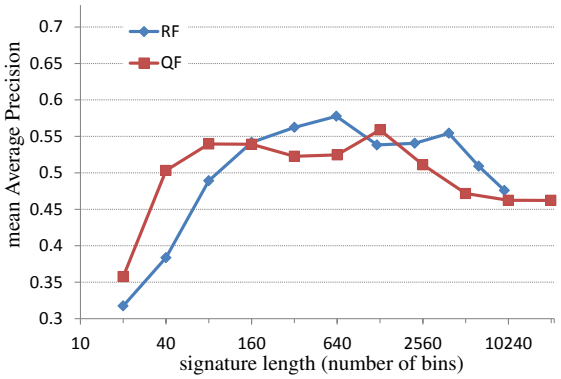
(c) YouTube-CV2, $C = 100$

(d) YouTube-CV2, $C \in \{1, 10, 100\}$

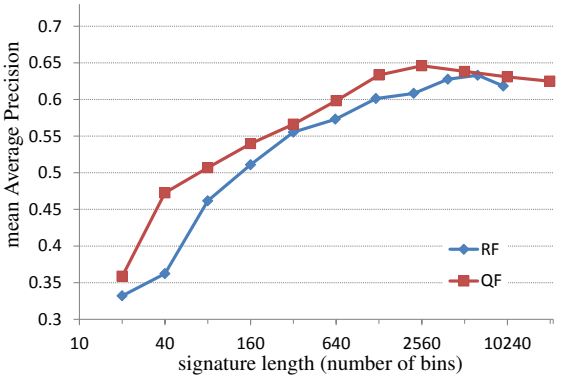
(e) YouTube-CV3, $C = 100$

(f) YouTube-CV3, $C \in \{1, 10, 100\}$

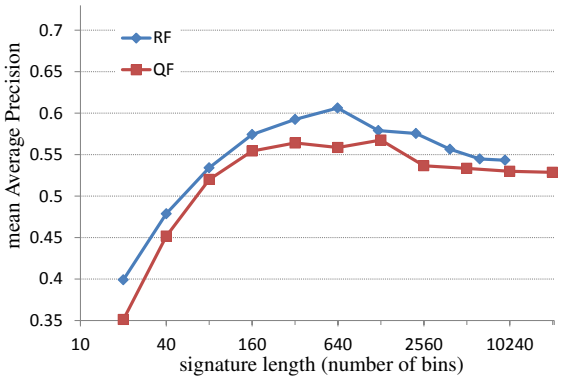
Figure 4.12: Comparison of the performance of random forest and quantizer forest on different partitions of the YouTube dataset using STIP features. *(Continues...)*



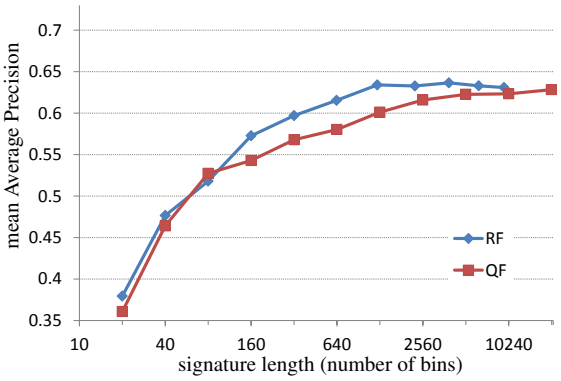
(g) YouTube-CV4, $C = 100$



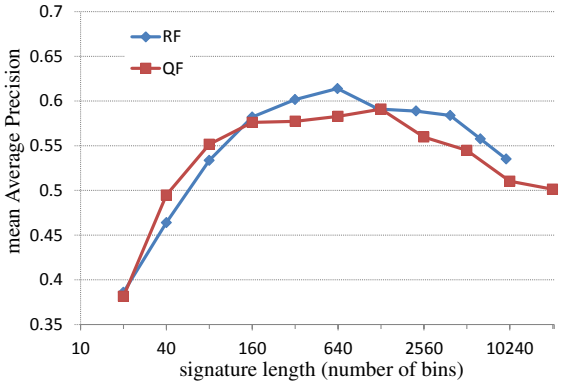
(h) YouTube-CV4, $C \in \{1, 10, 100\}$



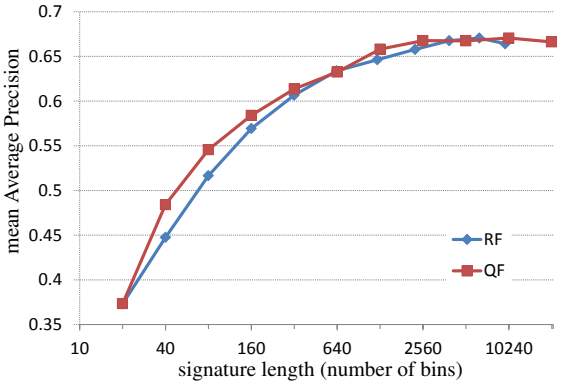
(i) YouTube-CV5, $C = 100$



(j) YouTube-CV5, $C \in \{1, 10, 100\}$

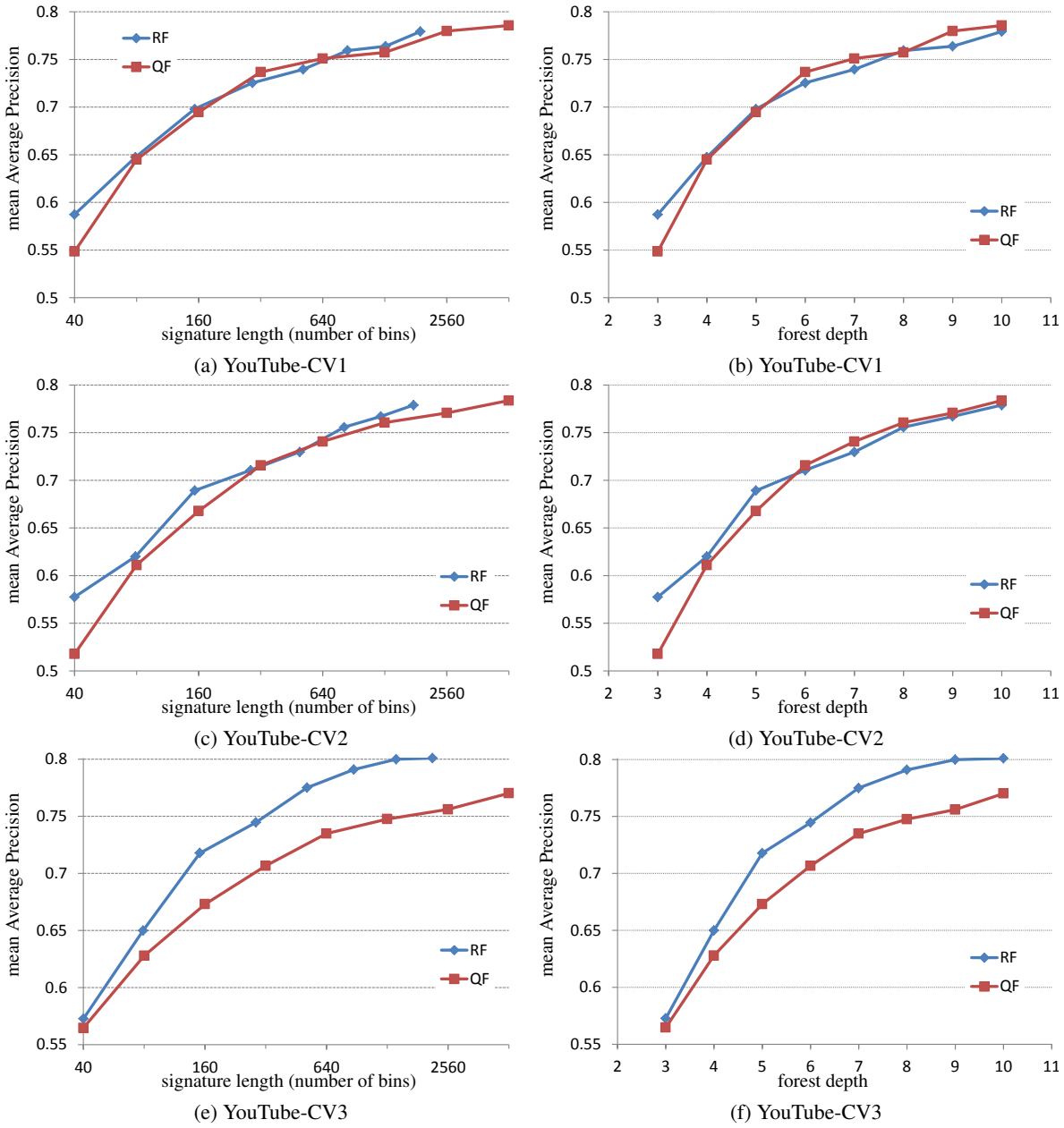


(k) YouTube, average of the five partitions, $C = 100$



(l) YouTube, average of the five partitions, $C \in \{1, 10, 100\}$

Figure 4.12: *(...continued)* Comparison of the performance of random forest and quantizer forest on different partitions of the YouTube dataset using STIP features.



(a) YouTube-CV1 (b) YouTube-CV1

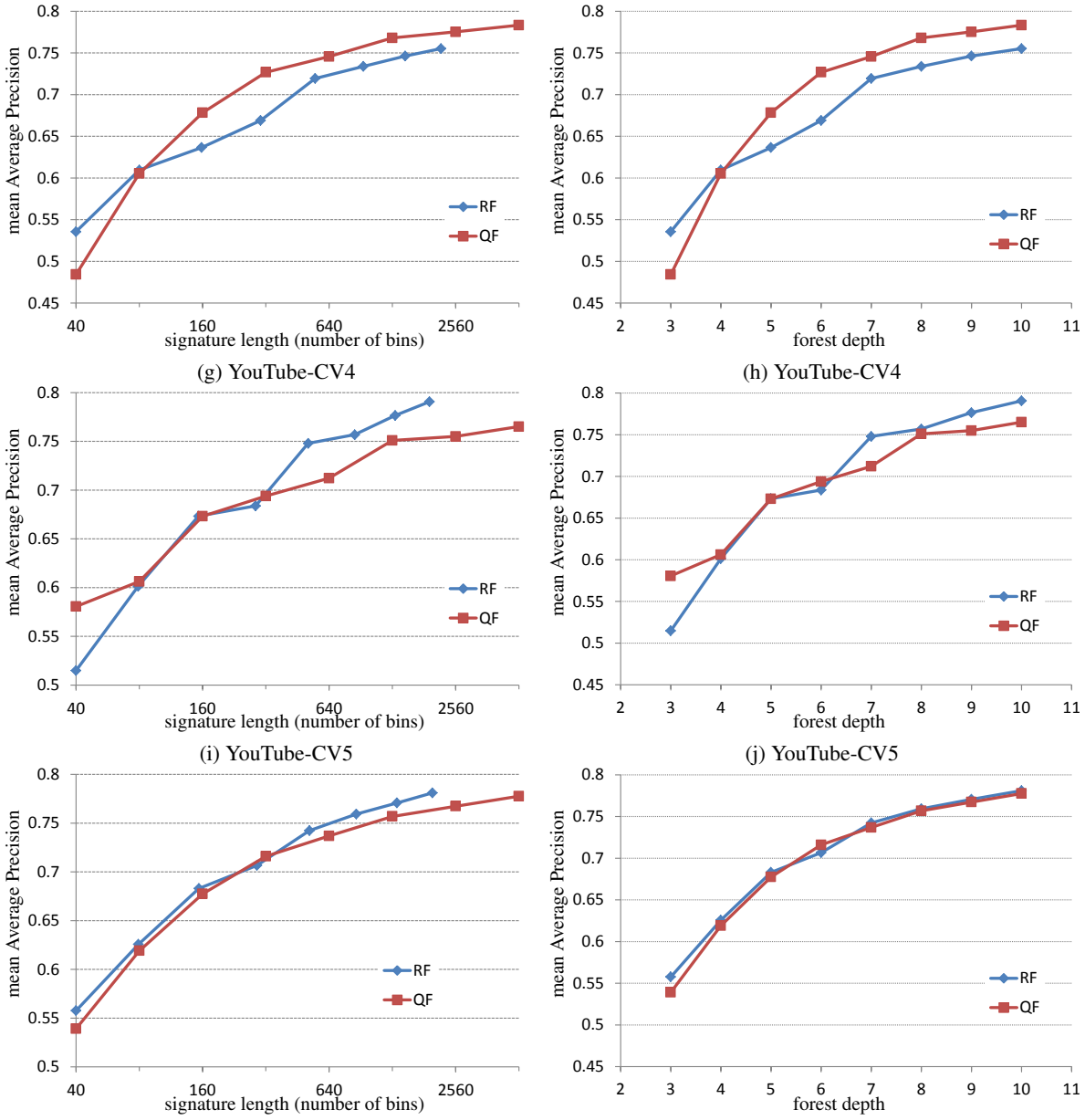
(c) YouTube-CV2 (d) YouTube-CV2

(e) YouTube-CV3 (f) YouTube-CV3

Figure 4.13: Comparison of the performance of random forest and quantizer forest on different partitions of the YouTube dataset using dense MBH features. *(Continues...)*

training signatures computed on the fly to make decisions about good splits. Due to the restricted sample of MBH descriptors during the QF training, these training signatures computed on the fly may be less representative of the final signatures, which are computed for a trained forest with all the local descriptors available for each video. Therefore, the split decisions may be less accurate and the advantage of using such training objective is reduced.

Table 4.2 compares the time needed to train one tree of RF and QF for the experiments reported in this section. While in the case of RF training time is only proportional to the number of training descriptors and the depth of the tree, in QF the training time also depends strongly on the number training videos, as it needs to compute video signatures and distances to nearest neighbors per every



(g) YouTube-CV4 (h) YouTube-CV4

(i) YouTube-CV5 (j) YouTube-CV5

(k) YouTube, average of the five partitions (l) YouTube, average of the five partitions

Figure 4.13: *(...continued)* Comparison of the performance of random forest and quantizer forest on different partitions of the YouTube dataset using MBH features.

Table 4.2: Comparison of RF and QF training time for the experiments reported in this section (approximate training time per tree).

	RF	QF
KTH STIP	$<$ 1 min	12 h
Hollywood2 STIP	2 min	60 min
YouTube STIP	1 min	20 h
Hollywood2 MBH	$<$ 1 min	50 min
YouTube MBH	$<$ 1 min	26 h

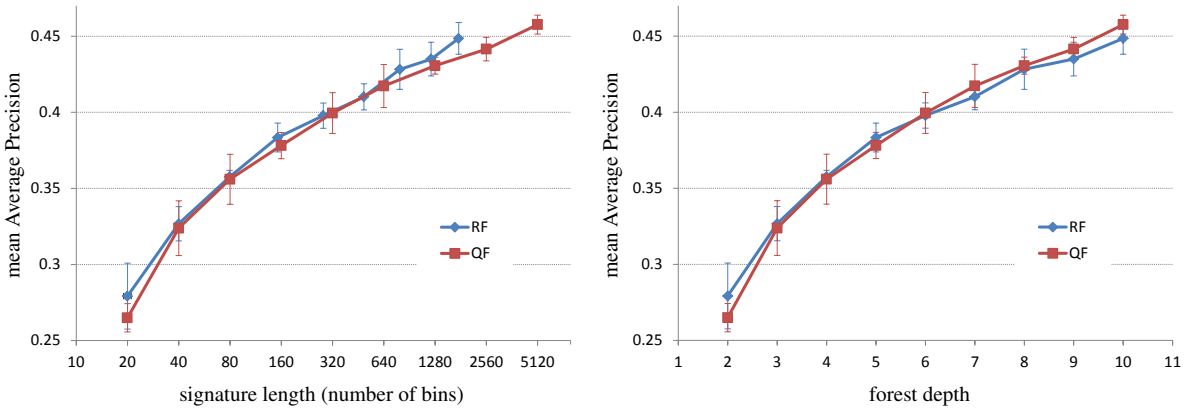


Figure 4.14: Comparison of the performance of random forest and quantizer forest on Hollywood2 dataset using MBH features. Average and 95%-confidence intervals over five runs of each type of vocabulary.

candidate split. The number of training videos is approximately 1100 in KTH, 800 in Hollywood2 and 1200 in each YouTube partition.

4.6 Discussion

In this chapter we have studied the problem of quantifying local descriptors from the point of view of training a random forest. In order to obtain visual vocabularies, random forests are preferable to other techniques such as k-means because of their comparable or better performances and higher speed.

Random forests have some parameters which are well understood when it comes to classification and regression tasks, but their behavior may differ in this quantization task. Here, we have explored some of these particularities, such as the influence that forest depth and number of trees have in the performance of the resulting bags of words. A few trees are enough to achieve the positive effect of ensemble methods. An appropriate depth is more important than in the case of forests for classification, because overfitting of individual trees is not compensated anymore by averaging of multiple trees. Instead, the signatures issued from the different trees are concatenated, and depth has a direct impact in the resolution of the quantized words. We have also compared two different pruning techniques, without conclusive results. In summary, those aspects make a difference on the final performance, and therefore they are not to be neglected when using random forests for quantization for the first time on a new dataset. They deserve consideration at least in the form of parameter selection and validation, if not further research.

The main contribution of this chapter is to draw attention to the fact that random forests are not particularly optimized for this quantization task. Indeed, their objective function is based on class entropy of local patches. This information is not really available, as there is no such category unambiguously attached to each local patch – in practice the local class label is inherited from the global video or image class label, which can be unreliable.

This is why we have proposed our new objective function, which explicitly optimizes a function of the arrangement of video signatures directly in the signature space, rather than in the space of lo-

Figure 4.15: *(Continues...)*

Figure 4.15: *(...continued)* Illustration of YouTube groups. Example frames of the basketball shooting action separated by groups. Note that what we have created 5 partitions of the YouTube dataset, using 20 groups for training and 5 groups for testing, e.g. YouTube-CV1 uses 06-25 for training and 01-05 for testing.

cal descriptors. We have shown that, at the cost of a longer training time, out algorithm can provide more compact signatures leading to higher classification accuracy. This happened in the case of the Hollywood2 dataset, which is the more challenging of the three we tested.

There was also an improvement in three out of five partitions of the YouTube dataset, partitions YouTube-CV1 and YouTube-CV5 being failures. Not all partitions of YouTube behaving the same way may be related to the peculiar nature of this dataset. As described in 3.3, YouTube has 25 groups of videos, which are supposed to separate actions performed by the same actor and/or similar background. We have partitioned this dataset into five larger groups, each containing five of the original groups. For each partition YouTube-CV1 to YouTube-CV5 we use four large groups for training and one large group for testing. Figure 4.15 shows some examples from the basketball shooting action. Note that some groups are "polluted" in the sense that very similar videos are spread through several groups. Compare for example groups 03 and 23 in Figure 4.15, which contain very similar videos. This means that when using partitions YouTube-CV1 (06-25 for training and 01-05 for testing) and YouTube-CV5 (1-20 for training, 21-25 for testing), very similar videos are being used for training *and* testing at the same time, breaking the independency between the training and testing dataset.

We have not visually inspected the hundreds of videos in this dataset, but something similar could be happening for other groups and other action categories. This undesirable correlation between training and testing sets in YouTube could be the object of further experiments. For example, a classifier could be trained on *one* large group (rather than four) and tested on each of the other large groups. The classification performance would roughly indicate how well each large group is explained by each other large group. This could reveal any asymmetries or bias in this dataset.

As discussed towards the end of the previous section, the unsatisfying results when using MBH features can be due to the smaller sample of descriptors used for training, which would make the signatures used for training our QF too dissimilar to the ones used for training the SVM classifier, making the split decisions unreliable.

Our objective function is still imperfect in several aspects. In our opinion the most compelling ingredients of random forests for quantization are the randomization of the splits and the power of ensemble methods. However, our objective works currently in a way that may be diminishing the positive effect of these ingredients. The first potential limitation is the redundancy among our trees. Each node is trained independently, and they may all be focusing on improving classification of the same restricted bunch of videos, leading to similar (redundant) splits and not paying attention to other harder videos. Instead, the desirable behaviour would be that different nodes and trees complement each other, improving the classification of different areas of the video signature space. A way to empirically visualize whether this is indeed a limitation of our QF would be to calculate statistics on the types of binary tests constituting the trees (i.e. which dimensions of the feature vector are selected and at which depth levels) and compare how diverse they are in RF and QF. A solution to reduce redundancy and encourage the desirable behavior would be to keep an Adaboost-like distribution of weights over the training videos, so that our objective function at each node focuses on better discriminating videos that have not been

correctly classified up to the current moment. We think this would work both in a breadth-first training and a depth-first training without parallelization of the branches.

The second potential limitation is the risk of overfitting the training data. A way of addressing this would be to hold out some training videos for validation of the classification accuracy.

Regarding training time, no particular effort has be put into optimizing the code, so it would be reasonable to expect improvements through a more efficient implementation of certain blocks, notably the classification accuracy evaluation of each candidate split.

We think that the proposed enhancements of QF would lead to higher accuracy than RF in most situations, at the cost of more computation. If it were not the case, a sensible conclusion would be that the main role of the supervision in forest for quantization is to reduce the variance of the individual trees and make the splits more stable. This would mean that the particular form of supervision (entropy or some other objective function) would be less critical.

The behavior observed regarding the balance of the trees is particularly interesting, as it may have connections with the original purpose and objective functions of quantization in signal processing: allocating higher resolution to zones of the input space happening more often.

Chapter 5

Early fusion of heterogeneous features

As explained in Section 2.4, there is a considerable number of available choices for interest-point detectors and local patch descriptors. We want to draw attention to alternative sources of information in the video, other than local features. Intermediate regions (or the whole video) can also be described in terms of gradients, optical flow, color, etc. as well as in terms of semantic attributes, e.g. the output of a classifier trained to detect certain concepts: materials, objects, pedestrians, etc. The optimal way to combine these heterogeneous sources of information (heterogeneous both in scale and nature) into a single global representation is still an open question.

In this chapter we enumerate different schemes to combine such heterogeneous information. We highlight the risks of naive combination of features too early in the pipeline, and illustrate them through an experimental example. Results show that our quantizer forests are more robust than information-gain-based forests at this challenging task.

5.1 Obtaining global representations from heterogeneous features

We picture below some the scenarios of our interest, from the simpler and better-known to the more complex, for which we may expect to find some difficulties. They can be seen as a generalization of the bag-of-words framework presented in Section 3.

5.1.1 One single type of local descriptor

This scenario corresponds to traditional bags of words as described in Section 3.

The *vocabulary* can be obtained using clustering algorithms, e.g. k-means. The number of clusters needs to be chosen appropriately, depending on the desired resolution. In the bag-of-words framework, the goal is to obtain a balance between too specific and too generic words.

We have frequently used Extremely Randomized Clustering Forests [80] for this purpose in our experiments, preferring this option over k-means because of the higher speed on great amounts of data, and the higher performance and stability in the bag-of-words approach. We have also proposed our own clustering algorithm in Section 4.4.

Instead of accumulating words to form an orderless global representation, the words can be used to perform other tasks, e.g. object localization through center voting after a training stage. This has been applied to actions for example by Mikolajczyk and Uemura [77].

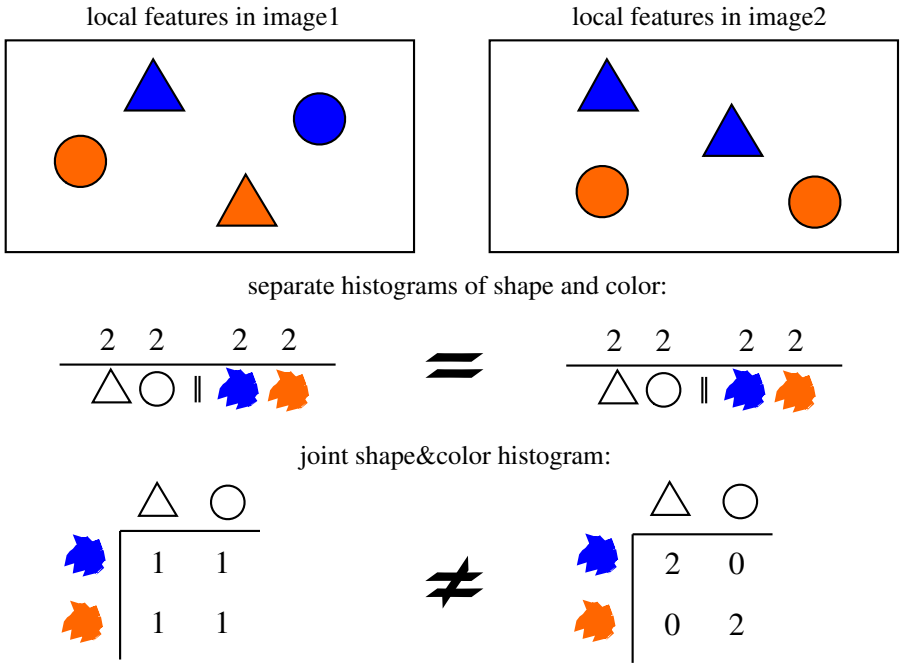


Figure 5.1: Toy example of two types of local features (shape and color) with two possible values each. At the top: two toy images showing the detected features. In the middle: each feature can be histogrammed separately, and both histograms would be combined before the classification step (an example of late fusion). In this case, this makes the representation of the two images indistinguishable. At the bottom: both types of feature can be histogrammed jointly (an example of early fusion). Which of the two strategies is preferable depends on the images labels (whether they belong or not tho the same class) and the rest of images in the dataset.

5.1.2 Several types of local descriptor

Without loss of generalization, let us consider two types of local descriptor, e.g. corresponding to two different cues, color and shape, as in [48]. This is illustrated in Figure 5.1. There are two straightforward options to combine them (adopting the terminology in [48]):

a) Late fusion

This implies building two separate vocabularies and obtaining two independent global representations. Then they can be combined by simple concatenation, or perform a more sophisticated combination and selection process. The fusion is done at the stage of the *classifier*. Some examples:

- Using different similarity measures for the different representations within a learning framework and learning relative weights, e.g. Multiple Kernel Learning [3], Heterogeneous Feature Machines [12] and, very recently, Generalized Adaptive l_p-norm Multiple Kernel Learning [140], which enable to learn a combination of features and outputs of other classifiers.
- Feature selection component by component, e.g. boosting [104], random forests [30].

b) Early fusion

Early fusion consists of computing a joint histogram of the two descriptor spaces, in which each bin represents the co-occurrence of two cues on the same local patch. In a straightforward way, the two descriptor spaces can be quantized independently into N_1 and N_2 clusters respectively. A joint histogram

of the product space would have $N_1 \times N_2$ bins.

If training examples of the joint cues are available (i.e. training patches associated with their two descriptors), then one can envisage to build a joint vocabulary, taking into account the two descriptors at once. This also leads to a single global representation, usually shorter than the $N_1 \times N_2$ histogram that would be obtained by the product of the two spaces quantized in independently.

As claimed by Khan et al. [48], this joint representation may make the learning stage harder if the class to be learnt is unrelated to one of the cues.

Nothing prevents from combining the representation obtained with the joint vocabulary with the two separate representations through late fusion. More generally, all the information may be available at any point of the pipeline to modulate the final representation, not only the very beginning or the very end. The two following examples are just particular cases of this general concept.

Khan et al. [48] show how the two cues can be used for two different tasks. In their case, color is used to build class-specific attention maps. Patches are sampled from that map and their shape descriptors are used to build a global class-specific representation of the image.

Similarly, Ullah et al. [121] and Su and Jurie [115] use one type of feature to segment a video into parts and then compute histograms of another type of feature on each part.

5.1.3 Descriptors at different scales

Let us consider two types of descriptors: one local and one global, e.g. HOG descriptor for local patches, and a color histogram for the whole video.

The local one can be turned into global in a bag-of-words manner, independently from the global feature. Then the two can be combined through late fusion.

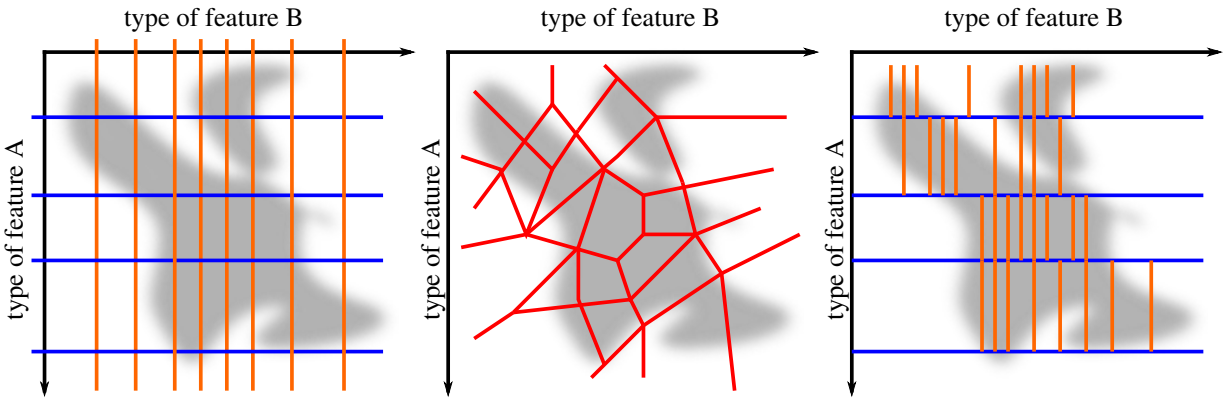
Early fusion of these two features is less straightforward, however. An attempt to build a joint vocabulary would imply including the global information at the local level. For each patch, the local descriptor (e.g. HOG) would be expanded by concatenating the global information (color histogram). The extended feature vector would have a part which is common for all the local patches belonging to the same video. A similar thing could be done for mid-level contextual information. The following sections contain further discussion and experiments exploring this subject.

5.2 Joint quantization of heterogeneous features

Whether early of late fusion is the best option depends on the features and the data to be classified. Here we explore early fusion, as we have found less work on this subject in the image and video classification literature.

We call early fusion the computation of a joint histogram of several, potentially heterogeneous features. We consider only two features without loss of generality. As evoked previously, we see several ways in which a joint histogram can be obtained (Figure 5.2).

- Independent quantization of each feature separately.
- Independent quantization of one feature, then quantization of the second feature conditioned to the quantized value of the first one.



(a) Independent quantization (b) Joint quantization (c) Conditioned B|A

Figure 5.2: Illustration of different ways to quantize two types of feature. In grey the joint distribution of both features.

- Joint quantization of the two features.

We have seen the benefits of traditional random decision forest for clustering over k-means. However, in some situations the hallucinated labels can have a harming effect. In the case of a redundant feature, strongly correlated to a given video instance, but not necessarily to the video class in general, the risk of overfitting and poor generalization is very high. Consider the scenario in which a point is described at the same time by a local patch descriptor and some contextual mid-level descriptor. The contextual features are more likely to be shared by a large number of points belonging to the same video. Those points share, in addition, the same hallucinated label (the category assigned to the video). A split based on such mid-level features would easily improve an entropy-based score. Indeed, entropy-based scores consider points independently, regardless of whether they belong to the same video.

In fact, joint quantization of local and mid-level descriptors using entropy-based measures is very challenging. Because of their correlation to hallucinated labels, mid-level features tend to appear "stronger" learners and systematically preferred over local features. This means that the tree is splitting mostly the space of the mid-level features and very few splits happen in the space of local descriptors. There is no such joint quantization actually happening.

It also has a second and more dangerous effect. This training procedure is blind to the difference between correlation to a video instance and correlation to a more general category label. It is therefore subject to a high risk of overfitting and poor generalization. On the contrary, our objective function presented in Section 4.4 does take into account the video to which the points belong, minimizing the adverse effect of entropy-based scores combined to hallucinated labels.

Performing such joint quantization with an unsupervised method such as k-means would imply similar difficulties. The contextual features are more likely to be shared by a large number of points. Therefore, positioning the prototypes regarding mostly such contextual features would more easily optimize the k-means objective, whereas the local part of the descriptor risks of being interpreted as noise. Again, we would be mainly clustering the contextual space.

A possible solution is to avoid purely joint clustering. Local information could be clustered first and

Table 5.1: Performance of a 20×20-bin global color histogram on YouTube-CV5.

action category	$C = 100$	$C \in \{1, 10, 100\}$
basketball shooting	0.859	0.576
biking	0.322	0.400
diving	0.947	0.947
golf swing	0.698	0.698
horse riding	0.578	0.579
soccer juggling	0.402	0.402
swing	0.428	0.355
tennis swing	0.528	0.528
trampoline jumping	0.505	0.505
volleyball spiking	0.709	0.709
walking dog	0.203	0.203
mean AP	0.562	0.536

then disambiguated through context. This could be achieved by the conditioned quantization mentioned earlier.

5.3 An extreme case: the global color histogram

In this section we experiment with an extreme case of mid-level feature: the global color histogram. We transform each clip to the L*a*b* color space, and regularly discretize the a* and b* dimensions to obtain a color histogram of 20×20 bins. We compute a global color histogram per clip.

We evaluate its standalone performance when used directly as video signature for classification. We also evaluate its performance when jointly quantized with local HOG|HOF features and when combined to a HOG|HOF signature through late fusion.

5.3.1 Standalone performance

Training an SVM with the global color histogram and χ^2 kernel on the YouTube-CV5 dataset leads to the classification scores shown in Table 5.1. Note that cross-validating the regularization parameter C does not produce better performance, leading us to thing that this global color histogram is not a very reliable feature, in the sense that cross-validation may not reflect accurately the generalization capabilities of this feature.

Another way to evaluate the standalone performance of this color histogram is to quantize its value and the represent each video as a histogram of quantized values[1]. We use both a random forest (RF) and our quantizer forest (QF) described in Section 4.4 to do this quantization, then perform SVM classification with a χ^2 kernel. Results are shown in Figure 5.3(a). We can see that the RF reaches its maximum performance at depth four, and then it gets worse, most likely due to overfitting. QF tends to get slowly better with depth. Interestingly, at depth ten it nearly stops growing: no more splits are found that improve our objective function. On the right side of the figure we can see the performance of the same forests when cross-validation is used to choose the SVM regularization parameter C. The RF clearly gets a higher benefit than QF from this cross-validation step. This reinforces our intuition that the RF produce too specific signatures that tend to overfit in this setting, therefore the need to regularize

correctly, whereas the QF is already trying to prevent overfitting with its objective function. However, both forests reach a similar maximum between 0.30 and 0.35 of mean AP, which is decidedly worse than the $\sim$0.55 that can be obtained when skipping the quantization step.

5.3.2 As part of a local point descriptor (joint quantization)

Now we consider the global color histogram as an extreme case of a contextual mid-level feature. We attempt to perform joint quantization of a local HOG|HOF feature the color histogram using a traditional random forest. One could think that such contextual information might help to better discriminate between two otherwise identical local patches, but in fact entropy-based quantization combined to hallucinated labels leads to the problem described in the previous section.

Both the QF and the RF build HOG|HOF signatures reaching over 0.60 of mean AP (see Figure 5.5). However, introducing the global color histogram provokes a decrease in performance, as show in Figure 5.4. QF resists better the the added "noisy" feature, and is always better than RF at this joint quantization task, meaning that it tends to select more HOG|HOF features to split the joint space. Still the performance is far below the 0.60 that can be achieved quantizing HOG|HOF features alone.

In order to force the selection of both kinds of features, we tried other approaches (such as forests alternating the choice of the two types of features, forest jointly selecting and thresholding pairs of components of the two feature types at each node, and forests looking at group entropy rather than category entropy alone), without success.

5.3.3 Late fusion

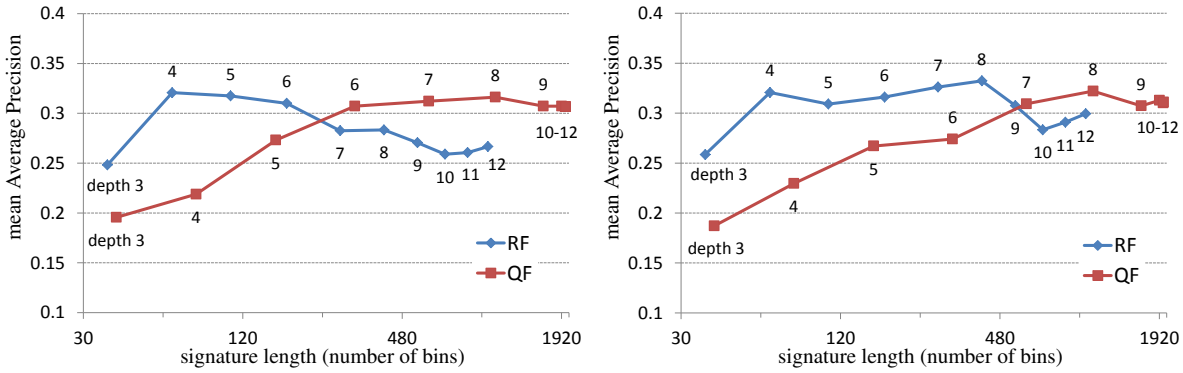
Figure 5.5 shows the mean AP that can be achieved when combining the HOG|HOF signature and the global color histogram through the sum of their respective kernels. In this setting, late fusion clearly outperforms the rest. It consistently brings an improvement to the HOG|HOF signature, for all tested lengths, although this increment tends to decrease with the length of the signature. As a by-product, this experiment shows that the global color features is indeed informative (as already shown in Table 5.1) and complementary to the HOG|HOF signature. The overall late vs. late fusion ont his dataset is shown in Figure 5.6.

This results could be even better by adding a weighting factor between the two kernels. Cross-validating such parameter for each class would potentially lead to increased performance, as Table 5.1 shows that the color histogram is more or less helpful depending on the action category.

5.4 Discussion

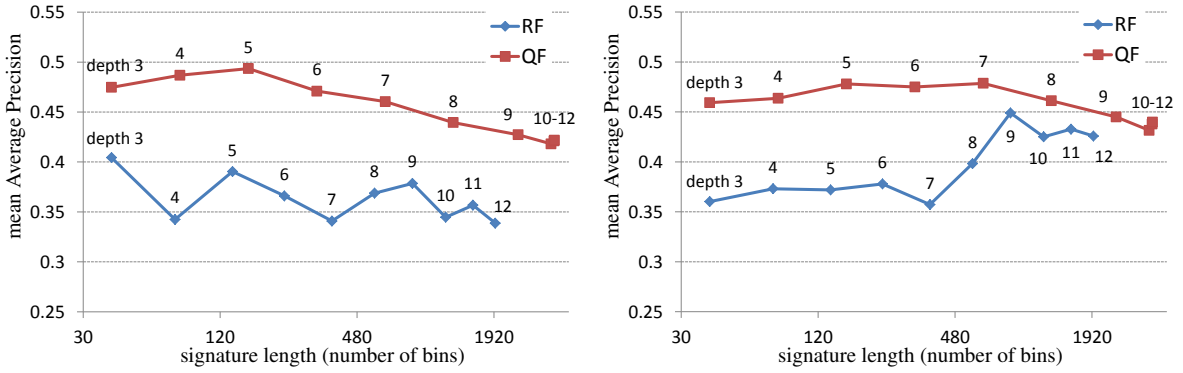
In this chapter we have presented a generalization of the bag-of-words framework, in which several heterogeneous features are to be combined, early or late in the pipeline. We have enumerated several ways of performing early fusion and drawn attention to the risks of naively mixing features with different statistical properties at the quantization step. Forest parameters such as minimum node support, maximum

[1] Actually, one of the purposes of quantization being to obtain a global signature in the bag-of-words paradigm, it does not seem necessary to apply it to which is already a global feature. Nevertheless, quantization would be meaningful in the case of mid-level features. We do this only to illustrate the behavior of the quantization of mid-level features, brought to an extreme.



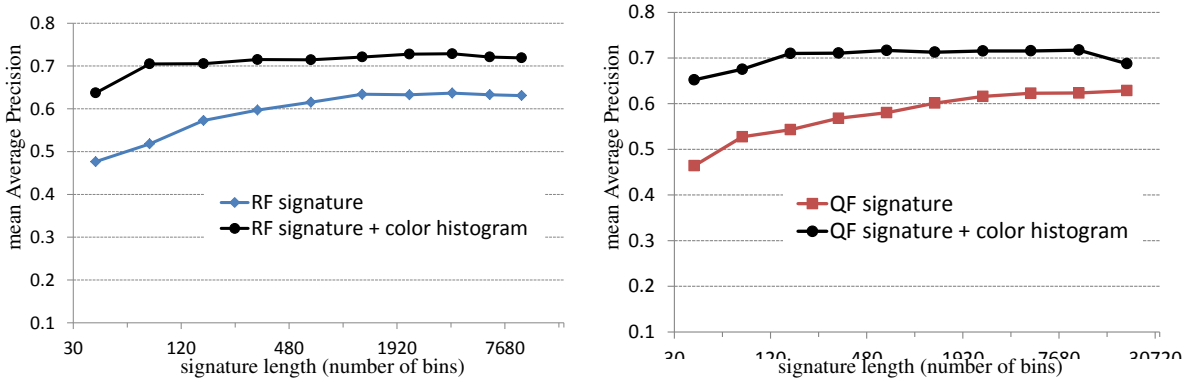
(a) With fixed $C = 100$ (b) With $C \in \{1, 10, 100\}$ and 5-fold cross-validation.

Figure 5.3: Quantization of the global color histogram using Random Forests and Quantizer Forests at different depths.



(a) With fixed $C = 100$ (b) With $C \in \{1, 10, 100\}$ and 5-fold cross-validation.

Figure 5.4: Joint quantization of the global color histogram and a concatenated HOG|HOF feature using Random Forests and Quantizer Forests at different depths.



(a) Signature computed using Random Forests. (b) Signature computed using Quantizer Forests

Figure 5.5: Late fusion of a HOG|HOF signature and the global color histogram through the sum of kernels.

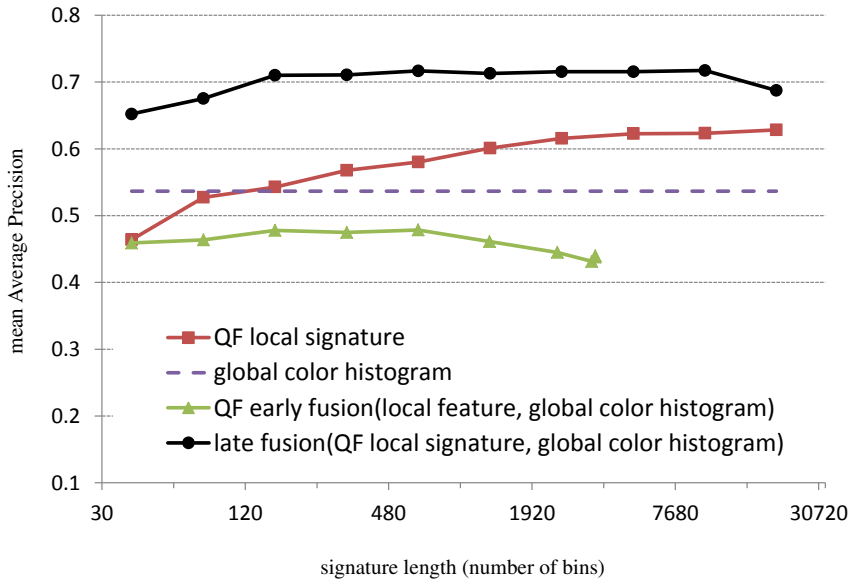


Figure 5.6: Late vs. early fusion of HOG|HOF signature and global color histogram.

depth, etc. control the forest bandwidth, which is related to the assumed level of noise in the data. If we put features with very different statistical properties into the same feature vector, one part of the feature vector can be seen as noise, and none of the intended joint quantification would actually be happening. We have also shown that our quantizer forest behaves better than the traditional random forest in this challenging quantification tasks, because it considers points grouped by the video they belong to, rather than treating all points independently. Moreover we have given an example in which the more straightforward late fusion performs better. Therefore, we would advise the designer of a vision system not to blindly commit for one or the other.

Although we have not experimented further with this idea, we have presented the possibility of *"conditioned"* quantization. This would consist of first quantizing one feature, and then quantizing the second feature conditioned to the quantized value of the first one. We hypothesize that this approach would potentially benefit from the complementary information contained in heterogeneous features, in a safer way than joint quantization, and with a more efficient use of the available bins than independent quantization. For example, conditioned quantization could be used to encode spatiotemporal relationships between local patches. Local descriptors (such as HOG|HOF) would be quantized first. Then we could continue growing each leave (or equivalently train one tree per local word) using splits based on spatiotemporal relationships, e.g. *does my i-th neighbor belong to the j-th word?* This approach would be taking into account the intermediate results of the tree, i.e. *which path in the tree did other neighboring points follow?* A similar idea has already been used in the entangled trees proposed by Montillo et al. [79] for medical applications.

Chapter 6

From local to mid-level descriptions

This chapter compiles our attempts to work with mid-level regions as the "basic unit" instead of local patches. There are several motivations to this. First, they are a step beyond bags of words, as the description of regions introduces some spatiotemporal constraints among words. Second, regions are more discriminative on their own than local patches. Therefore, they are potentially better than local words for localization, i.e. their location can be more easily identified to the location of the action. This means that in addition doing a good job at classification (as with local words), such a system could provide as a by-product a notion of at least rough location of the action –or of the elements that are relevant to the action. Closely related to this, some regions in the clip might be irrelevant or harmful to the classification task; a separate description of regions makes it possible to assign to them different importance in the final classification function, depending on their descriptor and their interactions with other regions. In addition, in a wide range of application, some extra annotation might be provided by the user of the system. This annotation can be useful to build more discriminative models, and the can quite naturally come in the form of regions with certain tags. A model that already uses a region-based representation has the potential to directly introduce such constraints in the learning process.

We review related work covering aspects such as the description of spatiotemporal neighborhoods of points, describing videos through histograms of pairwise relationships between points, dense description of regions of interest, and video description in terms of abstract concepts and semantic attributes. Then, we present our work with mid-level video representations and the questions and difficulties that arise, particularly regarding the huge amounts of data and the quantization and coding of region descriptors. We finally show that this mid-level representation can improve video classification, in agreement with contemporary work that has been recently published in the domain of 2D images.

6.1 Related work

Recent publications in the domain of action and interaction recognition –both in videos and in still images– show an increasing interest in going beyond the bag-of-words paradigm towards other representations. These approaches try to capture some of the geometric relationships among local patches, while keeping the robustness of those representations that accumulate unordered information.

Some explore the idea of building "structured" features by combination of simpler ones. They differ

in the definition of the points of interest and the extent of their neighborhood, as well as in the description of the geometric relationships among points.

Gilbert et al. [31] extract dense 2D corners in the x-y, x-t and y-t planes. For a given interest point, they consider a window of $3 \times 3 \times 3$ pixels around it and describe this window using a string of symbols plus a training label, issued from the action category of the clip from which the point was extracted. The string of symbols encodes the scale and orientation the points in the window and their position within. They use data mining to determine which strings are correlated to the labeled categories. The process is repeated hierarchically so as to describe neighborhoods at different scales.

Kovashka and Grauman [51] compute a first level of classical vocabulary of local descriptors (HOG, HOF). Each neighborhood is defined by a central interest point and its N nearest neighbors, rather than a fixed-size window. The descriptor of a neighborhood is the concatenation of cumulative histograms for increasing N, separated by word and orientation. Working with cumulative histograms lets the descriptors be more robust to the particular choice of N. Then they quantize the neighborhood descriptors using dimensionality reduction (PCA) and k-means. They repeat this process hierarchically, so that they describe neighborhoods at different scales. They have to deal with the fact that the selection of N nearest neighbors depends on the choice of distance. As the most meaningful distance measure is unknown *a priori*, they define several distance measures combining Euclidean distance and normalization of each component (x, y, t) for different normalization factors. In the end there is a vocabulary for each hierarchical level and combination of normalization factors. They represent each video by one bag of words per vocabulary. Then they combine all those bags of words using Multiple Kernel Learning [3], so that the final kernel is a weighted sum of χ^2 kernels (one per histogram) with the weights being learnt. How challenging it is to find and exploit the complementary information hidden in the spatiotemporal relationships is indicated by the big effort they put into generating so many histograms (for different levels in the hierarchy, for different distances, etc.) and then learning suitable weights.

Yao and Fei-Fei [142] work on still images of people interacting with objects. They build compound features called *grouplets*, which are OR/AND combinations of simple features. Simple features are defined by a word in a vocabulary of SIFT descriptors, their extent in the image, and their position relative to a reference point, i.e. the center of the human face. Their aim is to find the most discriminative grouplets. To efficiently explore the space of possible grouplets, they consider that the discriminative power of a grouplet can be decomposed into two necessary properties: the fact that they appear a lot in the images of the class of interest (they call this *support*), and fact they do not appear much in the other classes. They observe that for a grouplet of high order (i.e. containing several simple features) to have high support, the grouplets of lower order that it contains must be have high support too. This allows them to generate candidates for high-support grouplets from low to high order, and then only filter out from the candidates those that are not discriminative enough.

We observe some limitations in these approaches that could be addressed using more flexible definitions of the neighborhoods, or learning them with supervision. In [31] and [51], the neighborhood includes all immediate interest points. The neighborhood descriptors are hard-coded, in a way that al-

lows for no supervision. In [31] the labels are only used to pick the most useful neighborhoods at each level, while in [51] no supervision is used at all during the quantization step. Their hypothesis is that relevant information is contained in the spatiotemporal geometric relationships among close points, but they ignore the possibility that the neighborhoods might be polluted by irrelevant or harmful relationships. They do not look for distant relationships either –only describe compact neighborhoods.

Other works focus less on the local neighborhood descriptor and more on the overall arrangement of words within a short window [149] or in the clip [100], allowing for longer-range relationships. Yu et al. [149] consider a list of spatiotemporal relationships, e.g. before, after, overlap, near, far, etc., and build 3D histograms from pairs of words: two dimensions for the first and second word values, the third dimension for the relationship. They build on [100] and compare sets of words through histogram intersection. This provides a similarity measure that can be used to perform matching to training clips and classification.

We would rather let an algorithm infer the meaningful relationships among points, instead of imposing a given neighborhood descriptor or given list of relationships of interest. This is closer in spirit to the grouplets in [142], as they try to mine the discriminative combinations of features for the given labels. There are however several limitations to apply this to actions and videos. Their features are "too localized": they depend strongly on the position of the features relative to a reference point and a bounding box, which are not easily and robustly defined in the context of actions. We think that feature-centered structures are preferable. Moreover, the scores they use to select the best grouplets are "global", in the sense that they use all the training points of a given class to compute those scores. This does not account for the multiple flavors a class may have: some feature may not do a good job at describing the class as a whole, but may be very useful to describe one part of the data only. This is handled naturally in tree-like structures, in which the scores are evaluated only in a part of the data.

Following this line of thought, our quantizer forests (Section 4.4) could be used to quantize local patches based not only on their explicit pre-computed local descriptor but also other attributes, such as certain geometric relationships with their neighbors, introducing by this means some supervision in the description and range of the neighborhoods, as proposed in Section 5.4.

Other works describe regions densely rather than in terms of the words they contain. Ikizler-Cinbis and Sclaroff [40] pre-process videos to detect humans and find regions with coherent motion, and then describe those regions of interest using HOG and HOF descriptors. They assume that not all detected regions are useful for classification, and therefore adopt a multiple-instance learning (MIL) approach. MIL models for two-class classification take as input bags of instances (in this case, a video would be a bag of regions) and consider them positive if they contain at least one positive instance, and negative otherwise.

In particular, they apply MIL via Embedded Instance Selection (MILES) [14]. This approach defines a dictionary of "concepts", which are distributions generating instances. In this case, concepts are like words in a vocabulary of region descriptors. For a given bag (i.e. video), all instances (i.e. regions) are compared to all concepts, leading to a signature containing the maximum similarities to each con-

cept. The classification of the video is based on that signatureand can be done using linear SVM with l_1 norm. In other words, classification is performed after embedding videos into the concept space.

We observe the connection between the abstract concepts in [40] and the semantic attributes in other works. Liu et al. [66] manually specify a list of semantic attributes to describe actions (e.g. single leg motion, arm over shoulder motion, torso up-down motion, outdoor related, indoor related, sport related, etc.) and train classifiers to detect them taking as input the global video. Then, they use latent SVM to learn the combination of attributes that best models each action. In other words, actions are embedded in a space of semantic attributes, which are treated as latent in order to handle intra-class variability. The linear model of the latent SVM combines several elements: the output of a SVM on the raw bag of words of the video, unary attribute potentials, and pairwise attribute potentials, as well as potentials of other non-semantic data-driven attributes.

This is similar to Su et al.'s work [114] on 2D images. They train classifiers for semantic attributes (colors, shapes, materials, object parts, scene types...). They obtain global and mid-level responses from those concepts that they concatenate in a global low-dimensional image signature to perform object recognition. Of course, these semantic attributes require extra training data, but the idea is still that the image can be represented as a set of responses to the presence of concepts.

6.2 Mid-level representation of videos

In view of the related work and our own motivations presented earlier, we dedicate the remainder of the chapter to work with mid-level descriptions of videos, based on regions rather than local patches. We think that the semantic attributes in [66, 114] are not so different from the abstract concepts in [40]. They are properties than we can measure for a given video, these properties being roughly similarities to a series of (semantic or abstract) concepts.

We hypothesize that the complex pre-processing of [40] can be replaced by randomly sampling a sufficiently large number of regions in the video. We also want to avoid the specific training of separate semantic classifiers and the annotation they require, although being able to incorporate it if available would be interesting. In particular, we work with randomly sampled regions whose descriptors are bags of words, and experiment different ways of clustering them and building video signatures from the region-level.

Our approach converges with very recent work on 2D images that has been published after our own research on the subject [140, 116].

Also posteriorly to our research, and motivated as [40] by the presence of irrelevant context in the video, Sapienza et al. [101] have presented another MIL approach to action recognition. They sample regions following a regular grid and describe them through bags of words, but follow different learning method than MILES that focuses on the classification of regions themselves, skipping the quantization into concepts.

In the following sections we present our approach from the MILES point of view, and generalize this perspective to include other variants. We later discuss several questions that arise.

6.2.1 Multiple-Instance Learning

In a standard two-class supervised classification task one is given a set of training patterns $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$, supposed to be independent and identically distributed, from which to build a classifier $f : \mathbb{R}^d \to \{1, 1\}$. In multiple-instance learning, labels are only provided for sets of such patterns (bags of instances), $B_i = \{\mathbf{x}_i^1, \mathbf{x}_i^2, \ldots, \mathbf{x}_i^{m_i}\}$, where m_i is the number of instances contained in bag i. A bag is labeled as positive if *at least one* of the instances it contains corresponds indeed to the class of interest, whereas it is labeled as negative if *none* of the instances corresponds to such class, therefore introducing asymmetric constraints on the instances labels. The labels of the instances within a positive bag remain ambiguous.

Many MIL algorithms (e.g. [72, 151] cited in [14]) are build on the idea of finding *similar* instances co-occurring in *different* bags, the assumption being that those are more likely to be the positive target concepts. The earliest approaches typically consider one single target point, and label a bag as positive if at least one of its instances lies "near" the estimated target point.

MIL has been applied in both object and action localization (e.g. [39, 36]) to tackle the problem of ambiguous location and implicit alignment. In the continuum of locations containing at least partially the object of interest, some bounding boxes may be inherently ambiguous and it may be hard for a labeler to hard-classify them into the positive or negative class. Specially when the exact location of the object or action is ambiguous, forcing one particular bounding box and discarding the rest of potentially-acceptable bounding boxes may make the learning task harder. By sampling several locations around the provided or speculated box and formulating the problem as a MIL, the alignment requirements are relaxed, and the best bounding box for each training image is (at least implicitly) decided through the learning process.

We can also find in the literature the object recognition task presented as a MIL problem. An image depicting an object of interest, e.g. a car, is represented as a bag of patches. An image labeled as 'car' typically contains at least one car-like patch, indeed. However, as observed by, e.g., Gehler and Chapelle [29] and Chen et al. [14] , the opposite does not hold: the existence of a car-like patch in an image does not necessarily mean that the image is a car. It is therefore important to keep in mind the different meanings of the *bag label* and *instance label*. In applications like object recognition, if a positive instance label indicates that the instance appears to be part of the object (e.g. a car-like patch), then negative bag may contain positive instances as well. This kind of applications violate the traditional MIL constraint on the negative bags, even if presented as MIL problems.

Different versions of MIL have been defined that relax this constraint, so that negative bags can also contain some positive instances. E.g. Scott et al. [107] (cited in [14]) develop a framework in which there are several positive target concepts, and a bag is labeled positive only if it contains instances corresponding to *all* concepts.

We can distinguish two main families of approaches: those that aim at obtaining a classification function for the individual instances (the missing labels can be considered as hidden variables), e.g. [101]; and those that transform the problem into supervised classification at the bag level, without em-

phasis on the classification of the individual instances, e.g. [40].

A possible interpretation behind the second family is that the item that we want to classify –clips in our case– can be decomposed into several units –the instances, which we will imagine as being localized in space and time. It is assumed that some of these instances are more relevant than others when it comes to assigning a semantic label to the clip. Clips are represented as bags of such instances and given a semantic label globally. The aim of the feature extraction and learning process is to find a representation of such bags that, together with the training labels provided, enables to (implicitly) discriminate between relevant and irrelevant instances and therefore classify new bags. This interpretation seems well adapted to the classification task we are dealing with: training clips come with a weak classification label which says *what* action of interest happens, e.g. answering the phone; but we do not have any further information as to *where* in the clip the action happens, or *why*, i.e. what elements of interest whose absence of presence make people agree on such a semantic label for that clip.

In our work we build instances by taking rectangular spatiotemporal regions from every clip. We hypothesize that a concept-based approach may be useful to discriminate between relevant and irrelevant regions and better classify and localize the target actions.

6.2.2 Multiple-Instance Learning via Embedded Instance Selection

Multiple-Instance Learning via Embedded Instance Selection (MILES) [14, 26] corresponds to the second family of approaches mentioned in the previous setion, i.e. it reformulates the MIL problem (at the instance level) as a classic supervised classification problem (at the bag level).

Recall that the idea behind traditional MIL algorithms is to find similar instances co-occurring in different bags, assuming those are more likely to be the (positive) target point (roughly, the concept of of interest to be estimated). MILES breaks the asymmetry assumption about positive and negative bags. It allows for several target points, each of which may be related to the positive or the negative class.

Chen et al. [14] define a measure of probability that a point $\mathbf{x}$ is a target point given a bag B_i (independently of the label of the bag) using a most-likely-cause estimator, so

$$\Pr(\mathbf{x}|B_i) \propto s(\mathbf{x}, B_i) = \max_j \exp\left(\frac{\|\mathbf{x} - \mathbf{x}_i^j\|^2}{\sigma^2}\right), \tag{6.1}$$

where $\mathbf{x}_i^j$ are the instances in the bag B_i, and σ a predefined scale factor. The measure $s(\mathbf{x}, B_i)$ can be interpreted as the similarity between the point and the bag.

MILES considers all *instances* in the training set $\{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ (all bags confused) as potential concepts of interest. Then, it maps each training bag into a feature space representing the similarity of the bag to each concept.

$$m(B_i) = [s(\mathbf{x}^1, B_i), \ldots, s(\mathbf{x}^n, B_i)]^\top. \tag{6.2}$$

Once every bag has been embedded into this instance-based space, then standard supervised algorithms can be applied. They choose an SVM with l_1 norm, as they claim that the feature-selection aspect is essential in such a high-dimensional space.

6.2.3 Variants of MILES

In the MILES approach as described in [14] the dimensionality of the final feature space increases with the number of instances in the training set. There are alternative ways of transforming the problem into standard supervised classification in feature spaces with tractable dimensionality.

MIWrapper [26] performs this transformation by applying bag class labels to instances[1] and weighting the instances so that each bag has the same total weight. SimpleMI [26] obtain a bag's feature vector by averaging the feature vectors of the instances in the bag. The final feature space has the same dimensionality as the original one in both cases. In the first one there are as many training elements as instances, whereas the second only generates one training item per bag. Despite the simplicity of these two approaches, they have show comparable performance to MILES in typical MIL datasets [26].

In [40], the approach is very similar to MILES, only that the training instances are first grouped into a fixed number of clusters. The candidate concepts are then the prototypes or centroids of such clusters. Then, as in MILES, a bag's feature vector is build by measuring the similarity of the bag to each concept, i.e. the maximum similarity between all instances in the bag and the centroid.

6.2.4 Our generalization

From our point of view, several choices can be made in order to build a concept-based representation of clips, in which instances are regions sampled from the clip.

- The candidate concepts. They can be all training instances as in [14], or obtained through k-means clustering as in [40]. We can also imagine using random forests or our quantizer forests to obtain such partition. In the case of clustering, there is the question of how to chose the prototype of the cluster, to which all the similarities will be computed.
- The similarity measure between instances. Inspired from the hard-coding that we have been using to assign local features to words, we could consider a 0/1-similarity between instances looking only whether they belong to the same cluster. We can also computed similarities based on exponential functions and normalized distances, as in Eq. 6.1. The choice of distance measure and normalization (e.g. the same factor for all concepts, one factor per concept, etc.) remains unclear. Similarity could also be measured with a classifier or regressor trained in the area of the region-descriptor space corresponding to the concept.
- The measure between a bag and a concept, i.e. how to build the video signature from all the regions responses to the concepts. Using the maximum similarity of all instances of the bag to the concept is justified as a most-likely-cause estimator in some probabilistic framework (named *diverse density* [72] cited in [14]). Summing (or averaging) the responses of all instances in the bag could also be an option. In particular, the hard-coding combined to the averaging approach leads to a simple histogram of region types, i.e. what we could call a "bag of region words" or "'bag of regions" for short.

We think that these choices have consequences that deserve theoretical and experimental analysis. We conducted experiments involving k-means clustering of regions described through bags of words, using χ^2 symmetric distance (Eq. 3.8), similarities based exponential functions and different ways of

[1] Note the parallelism with the "hallucinated" labels used in ERC-Forest described in Section 4.3

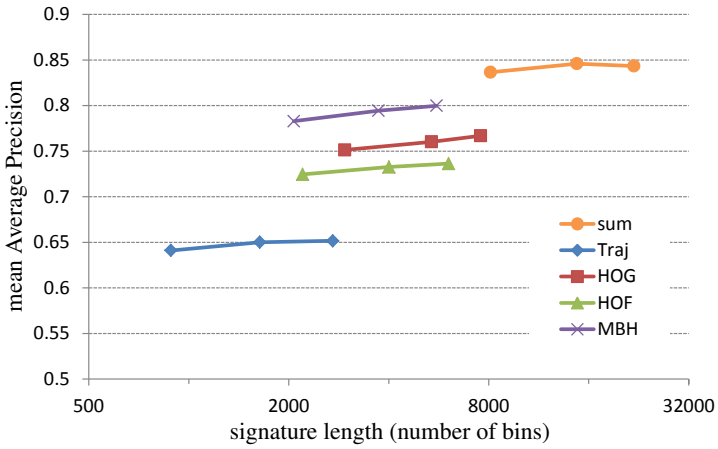


Figure 6.1: Bag-of-words performance on YouTube-CV1 using Dense Trajectory features and χ^2 kernels. We can see the performance of each descriptor individually as well as the sum of the four kernels.

obtaining normalization factors, and both maximum- and average-based pooling of region responses. We also trained classifiers dedicated to some areas in the region-descriptor space.

Further, we had the idea of decomposing each concept into its "positive part" and "negative part". This means that for each concept, a "positive prototype" can be computed from only the instances coming from positive bags, and the same for the "negative prototype". Similarly, the local classifiers could be trained not to distinguish between a concept and the rest of concepts, but to distinguish between the positive and negative part of a given concept.

Preliminary results of those experiments (not included here) indicate a strong impact of the way of measuring similarities depending on the normalizing factor. They also indicate that local classifiers for a given concept are useful provided that there is sufficient training data in the cluster, otherwise simpler measures such as the similarity are preferable. More exhaustive experiments along those lines would be needed to obtain more conclusive results.

In the following sections, we focus on reporting our experiments following the "bag of regions" approach, and compare it to the local-level bag of words.

6.3 Experiments

6.3.1 Bag-of-words baseline

In this section we compute and evaluate bag-of-words signatures of the YouTube dataset, using groups 6 to 25 for training and 1 to 5 for testing (a.k.a. YouTube-CV1). We use as local features the Dense Trajectories [124] described in Section 3.1.2, which consist of four descriptors: point trajectory (Traj), Histograms of Oriented Gradients (HOG), Histograms of Optical Flow (HOF) and Motion Boundary Histograms (MBH). We compute one vocabulary per type of descriptor. $2 \cdot 10^5$ local features were selected randomly from the training videos to train random forests, which we slice at different depths. This leads to four different signatures, according to the four descriptor types. We compute the χ^2 kernel of each signature type, then train one-vs-all SVM with fixed regularization parameter $C = 100$.

Results are shown in Figure 6.3.1. Performance seems not to vary much with the vocabulary size, the main differences being due to the the type of descriptor. We can see that these descriptors work better than sparser HOG|HOF STIP features used in previous chapters, at a higher computational cost.

Table 6.1: Bag-of-words performance on YouTube-CV1 using DenseTrack features and χ^2 kernels. Impact of the choice of regularization parameter C through 5-fold cross-validation.

forest depth	descriptor	signature length	$C = 100$	$C \in \{1, 10, 100\}$
10	Traj	883	0.641	0.638
	HOG	2 948	0.751	0.751
	HOF	2 201	0.724	0.722
	MBH	2 072	0.783	0.781
	sum	8 104	0.837	0.837
12	Traj	1 636	0.650	0.648
	HOG	5 383	0.760	0.760
	HOF	4 005	0.733	0.729
	MBH	3 724	0.794	0.792
	sum	14 748	0.846	0.846
14	Traj	2 713	0.652	0.653
	HOG	7 564	0.767	0.767
	HOF	6 053	0.736	0.737
	MBH	5 575	0.800	0.800
	sum	21 905	0.843	0.843

MBH is clearly superior in this dataset, and the sum of kernels is an appropriate way of combining such heterogeneous features is this case. We observe that for the same depth, very different lengths are obtained for each type of descriptor. This can be explained by the length of each local descriptor (and therefore the dimension of the space being quantized) and the variability of training descriptors: it seems that stopping conditions are reached earlier in the case of the Traj descriptor.

As a sanity check, we repeat the classification choosing the regularization parameter C among values $\{1, 10, 100\}$ through 5-fold cross-validation for each of the one-vs-all SVM, see Table 6.1. We do not observe significative difference in performance with regards to the default $C = 100$.

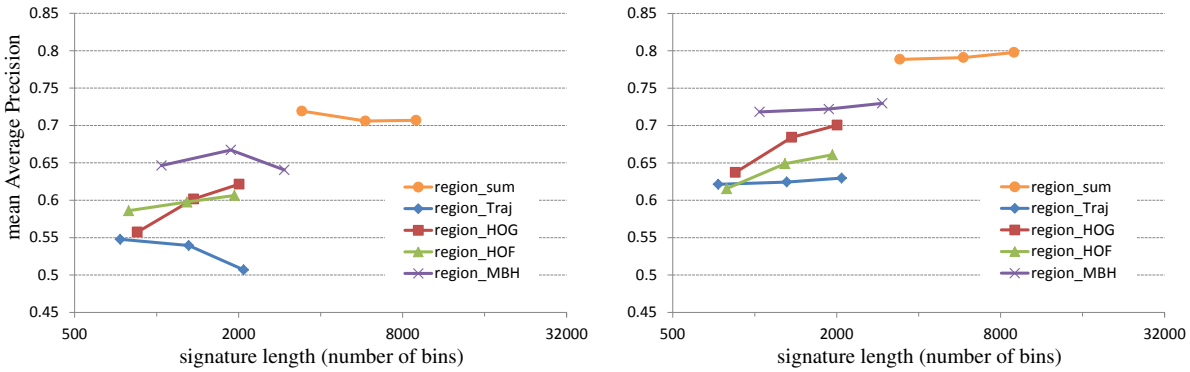
6.3.2 Bag of regions

In this experiment, we randomly sample several spatiotemporal region in each clip. We use the local-feature vocabularies from the previous section to compute bag-of-words descriptions of each region. Regions are then clustered using random forests.

We sample simple regions with the shape of spatiotemporal boxes. Spatially, we consider four different sizes: 20×40, 40×20, 20×20 and 40×40, measured in percentage of the frame width and height. For each size, we sample randomly allowing for some overlap until most of the area has been covered, up to a total of 144 rectangles per frame. Temporally, the boxes are 30 frames long, and sampled regularly every 15 frames.

We compute bag-of-words descriptions of each region using the 14-level-deep vocabularies from the previous experiment. There is one vocabulary (and therefore one region descriptor) for each local descriptor Traj, HOG, HOF and MBH.

We then use random forests to quantize the space of region descriptors. Training parameters are set to our default 5 trees and maximum depth of 14, but in order to allow the features to fit in memory we decrease the number of training regions to $4 \cdot 10^5$.



(a) With fixed $C = 100$.

(b) With $C \in \{1, 10, 100\}$ chosen through 5-fold cross-validation.

Figure 6.2: Performance of bag-of-regions descriptors and χ^2 kernels on YouTube-CV1.

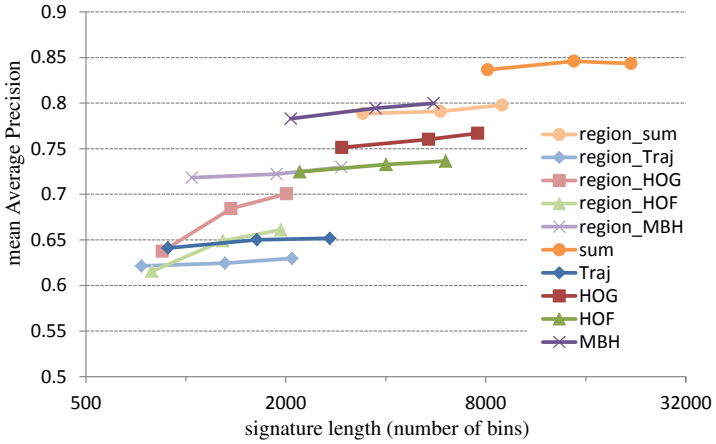


Figure 6.3: Local- vs. region-based representations on YouTube-CV1, using DenseTrack features and χ^2 kernels.

A new global signature of each video can be obtained by aggregating all the region information. In this experiment we opt for counting the number of regions in each cluster, in order to obtain a global histogram that we call a bag of regions.

Results are shown in Figures 6.2 and 6.3. The two more obvious observations are that (i) the influence of the SVM regularization parameter C is much higher here than in the case of the bag of local words, and (ii) the performance of the region-based clip representation is lower than the local-based, despite of the extra computation. Looking into the details of the cross-validation process (not included here) most classes prefer $C = 10$ rather than the default $C = 100$, which means a stronger regularization of the decision function. We speculate that the space of regions is under-represented compared to the space of local patches.

In the case of local patches, the DenseTrack features have descriptors whose lengths are in the order of tens of bins. Such features are extracted densely at every frame. Hundreds of thousands of samples are available at the time of training the random forest for quantization and all of them are used when computing the signature of a given clip.

In contrast, the spatiotemporal regions that we use in this experiment have much longer descriptors, in the order of thousands of bins. The region-descriptor space is therefore potentially much larger, which is in accordance with the fact that regions are potentially more specific than local patches. However, we sample such larger space comparatively more sparsely, with less than 200 regions every 15 frames, to be compared with a few thousands in the case of the local DT features[2]. We can also see in the figure that for the same depth of the forest (14 levels) much shorter signatures are obtained, which means that this larger space has been more coarsely quantized. All things considered, we can conclude that these regions are miss-represented and this has a negative impact on the generalization capabilities of the region-based signature.

6.3.3 Bag of regions with reduced region descriptor

A way to address the miss-representation problem described above is to reduce the dimensionality of the space of region descriptors. The straightforward option is to use the 12-level-deep local vocabularies instead of the 14-level-deep ones so as to reduce every region descriptor to approximately half the length. Another option is to use some dimensionality reduction technique. We tried Principal Components Analysis (PCA) previous to the region vocabulary training. None of the two options lead to conclusive results, and may deserve further analysis.

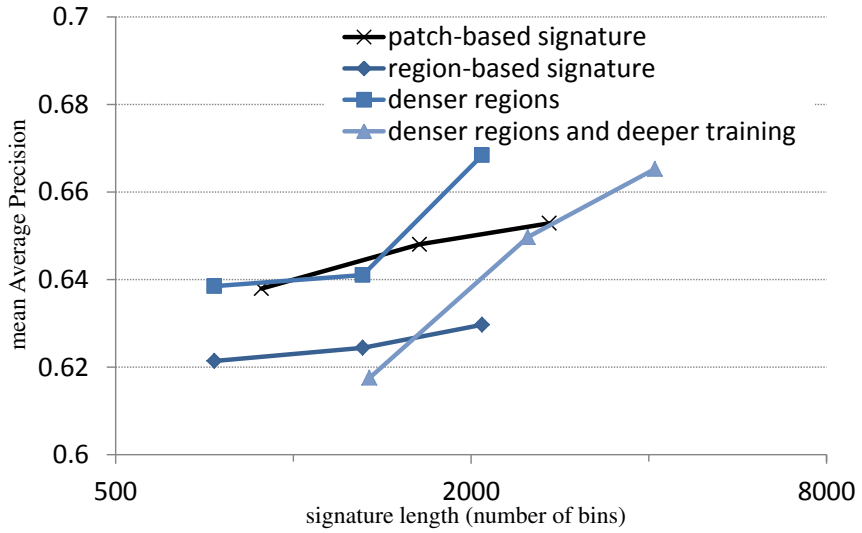
6.3.4 Bag of regions with denser regions

We conduct two small tests to see the impact of the amount of regions sampled per clip. To do so, we sample regions following the same spatial pattern described previously, but we make the boxes 15 frames long instead of 30 and sample them every 5 frames instead of every 15.

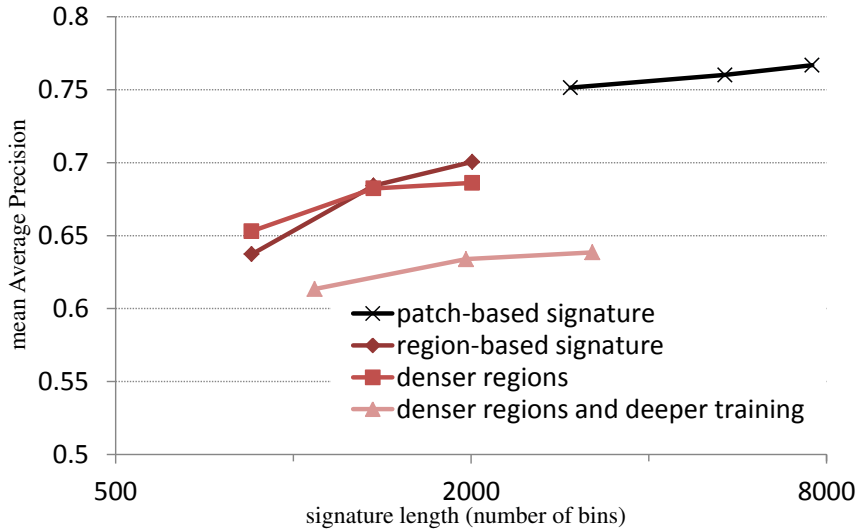
In the first experiment, we use the same region vocabulary computed in the previous section, and only use the new regions to compute each video signature. In the second experiment, we retrain the vocabularies using the new regions (but still around $4 \cdot 10^5$ regions for training) in addition to using them to compute the signatures. During training, the stopping conditions are relaxed in order to obtain deeper trees (minimum node support is decreased from 200 to 50 regions).

The results are shown in Figure 6.4, compared to the initial region-based and local-patch-based signatures. In the case of the Traj descriptor, there is a clear increase in classification performance when the same region vocabulary is used but more regions are sampled when computing each clip signature. The performance becomes then comparable with the local-based signature, without being significatively superior. HOF and MBH also seem to benefit from the denser sampling, although the performance remains worse than the local-based signature. MBH is the only case in which a deeper forest seems to improve the region-based signature.

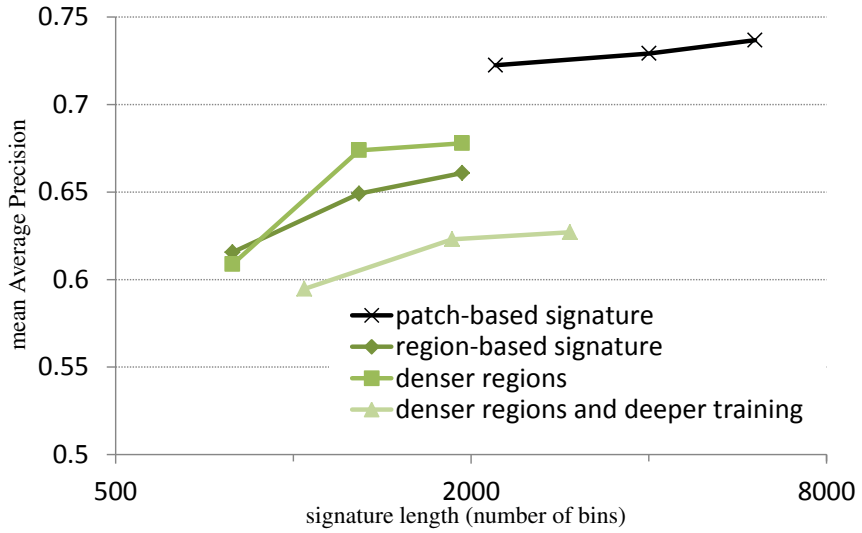
[2]A typical frame of the YouTube dataset can be 320×240 pixels. Points to be tracked are initialized following a 5×5-pixel grid, and tracked for 15 frames at most. More points are sampled in the empty areas as needed when existing tracks disappear. A rough estimate can be of $\frac{320 \cdot 240}{5 \cdot 5} \sim 3000$ features extracted every 15 frames. In practice this number can be higher. For instance, the clip 'v_biking_01_01' of this dataset has 5102 DT features between frames ♯15 and ♯29.



(a) Traj

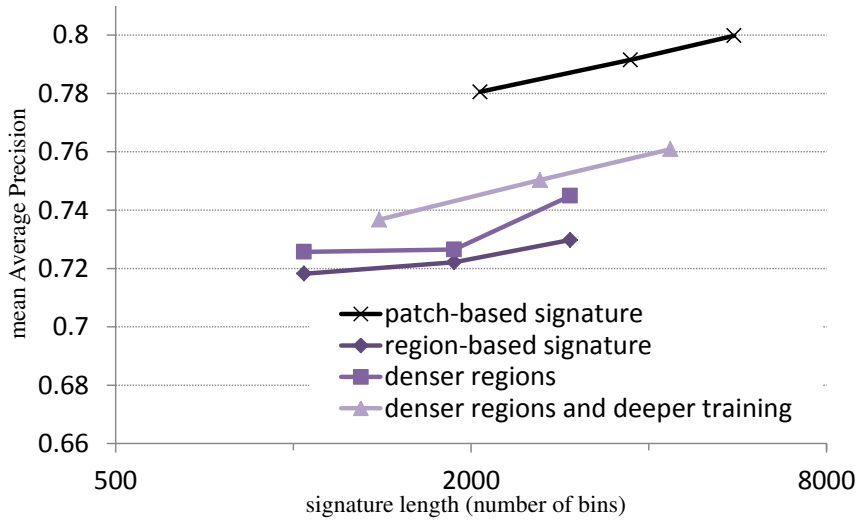


(b) HOG

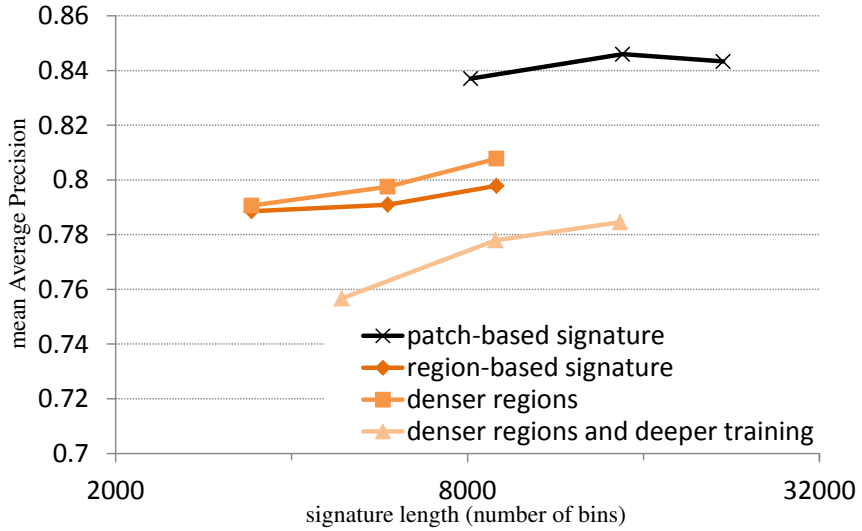


(c) HOF

Figure 6.4: Performance of bag-of-regions signatures when denser region sampling is used at signature computation time, and when denser region sampling is used combined with deeper training. *(Continues...)*



(d) MBH



(e)sum of kernels

Figure 6.4: *(...continued)* Performance of bag-of-regions signatures when denser region sampling is used at signature computation time, and when denser region sampling is used combined with deeper training.

6.3.5 Complementarity of the bag of regions and the bag of words

Table 6.2 shows the performance obtained by combining the bags of words of Section 6.3.1 and bags of regions of Section 6.3.2 through the sum of their kernels. Results are separated by class and by descriptor. We can see that in approximately half of the cases the combination is superior to any of the two kernels individually. We also observe that in the case of the Traj descriptor the bag of regions outperforms the bag of words quite often, which is never the case with the HOG and MBH descriptors. This table shows that bag of regions is not a redundant layer with regard to bag of words. Instead, and despite the limitations that we have observed in the previous sections, the bag-of-regions signature contains complementary information. This complementarily could even be exploited further through a *weighted* sum of kernels, which we have not tested here. The weights could be learnt in a Multiple

Kernel Learning manner, or selected through cross-validation with the rest of the SVM parameters. A faster approach would be to estimate the performance of each separate kernel through cross-validation and then use weights proportionally to those performances.

6.4 Discussion

The reported experiments focus on the bag-of-regions approach. It basically consists of applying a second iteration or layer to the standard bag of words so as to obtain a signature of quantized regions descriptors. Such region descriptors are here bags of words, but they could be any other descriptor of a spatiotemporal region: HOG, HOF, color, etc.. One advantage of using bags of words as region descriptors is to reuse the quantized local words, which we would compute anyway as we know that they produce useful signatures. We can see such region descriptors as introducing spatiotemporal constraints to an otherwise unordered bag of words.

We have also described the connections of this bag-of-regions representation to other MIL-inspired approaches. From this point of view, the target positive class is assumed to be related to one or several target *concepts*, and classification is performed on video signatures which contain information on the absence or presence of such concepts. Those concepts can be region prototypes, e.g. cluster centroids, and presence can be measured in terms of similarity. In the case of the bag-of-regions such concepts are possible values of the quantized region space, and presence is measured as the count of corresponding regions.

From our experience, the main drawback of this approach when applied to videos is the considerable computational needs. Compared to standard bag of words, it needs a second pass of region sampling and quantization. This extra computation may be worthwhile for image representation and classification, but it becomes quickly unfeasible when it comes to videos.

Furthermore, we have observed that this region space is quite high-dimensional, and in order to obtain reliable quantization and representations it has to be sampled quite densely. This problem becomes even worse with other similarity-based variants, which need the computation of centroids and similarities, and therefore the computation and comparison of lots of pairwise distances between regions, or between regions and centroids. Notions such as averages, similarities and nearest neighbors become tricky in such high-dimensional spaces (see for example the concentration of distribution issue [4]). Our preliminary experiments with similarity-based signatures indicate that they are indeed extremely sensitive to the way centroids are computed and distances are normalized into similarities. In order to answer to these question, a more principled approach and bibliographic research beyond the action recognition domain may prove useful. More exhaustive exploration the MIL variants that we propose in Section 6.2.4 may also help gaining further insight into region-based representations.

Nevertheless, we have observed that intermediate regions contain complementary information that may improve classification when combined to bag-of-words signatures. Region-based representations have had success in other scenarios, as can be seen in recent works [140, 116] which are contemporary to our research. We therefore encourage future research to go for such region-based representations. We find the notion of *concepts* particularly interesting, as it enables to use such abstract region prototypes

Table 6.2: Complementarity of bag-of-words (BOW) and bag-of-regions (BOR) signatures on YouTube-CV1 using DenseTrack features and χ^2 kernels. Performance of each kernel individually and their sum.

action category	Traj			HOG		
	BOW	BOR	sum	BOW	BOR	sum
basketball shooting	0.513	0.438	**0.586**	**0.596**	0.356	0.592
biking	0.737	0.745	**0.756**	0.849	0.827	**0.879**
diving	**0.811**	0.718	0.805	0.974	0.892	**0.984**
golf swing	0.766	0.701	**0.846**	0.928	0.865	**0.937**
horse riding	0.658	**0.712**	0.709	**0.895**	0.820	0.880
soccer juggling	0.712	**0.819**	0.815	0.439	0.541	**0.644**
swing	**0.759**	0.634	0.673	0.677	0.657	**0.684**
tennis swing	0.498	0.466	**0.544**	**0.582**	0.439	0.486
trampoline jumping	0.834	**0.891**	0.883	**0.833**	0.706	0.762
volleyball spiking	0.634	0.596	**0.704**	**0.996**	0.913	0.993
walking dog	**0.259**	0.208	0.246	0.666	**0.691**	0.667
mean AP	0.653	0.630	**0.688**	0.767	0.701	**0.773**

	HOF			MBH		
	BOW	BOR	sum	BOW	BOR	sum
basketball shooting	0.537	0.502	**0.551**	0.509	0.329	**0.569**
biking	0.713	0.702	**0.720**	0.935	0.918	**0.937**
diving	**0.900**	0.749	0.861	**1.000**	0.984	0.993
golf swing	**0.861**	0.725	0.846	**0.945**	0.770	0.907
horse riding	**0.853**	0.830	0.853	0.912	0.891	**0.914**
soccer juggling	**0.788**	0.745	0.786	0.900	0.883	**0.938**
swing	**0.713**	0.522	0.641	0.815	0.790	**0.819**
tennis swing	0.664	0.559	**0.679**	**0.510**	0.356	0.502
trampoline jumping	**0.840**	0.786	0.839	**0.909**	0.891	0.903
volleyball spiking	0.819	0.774	**0.833**	**0.818**	0.782	0.818
walking dog	**0.416**	0.376	0.413	**0.544**	0.434	0.495
mean AP	**0.737**	0.661	0.729	**0.800**	0.730	0.799

	4 BOW	4 BOR	all
basketball shooting	**0.737**	0.489	0.657
biking	**0.948**	0.919	0.943
diving	**0.987**	0.946	0.980
golf swing	0.976	0.944	**0.979**
horse riding	**0.919**	0.892	0.907
soccer juggling	0.837	0.836	**0.852**
swing	**0.863**	0.825	0.854
tennis swing	0.609	0.563	**0.621**
trampoline jumping	0.882	**0.885**	0.882
volleyball spiking	0.982	0.974	**0.984**
walking dog	0.536	0.504	**0.539**
mean AP	**0.843**	0.798	0.836

as if they were semantic attributes. This leaves the door open to more complex models exploiting correlations between pairs of (semantic or abstract) attributes, their location in the image, etc. It would also be interesting to exploit the particularities of videos in our favor rather than being limited by the burden of huge amounts of data. For example, smarter sampling strategies could be found that benefit from the inherent temporal redundancy in videos or from domain knowledge. One example is the concurrent work carried out by Raptis et al. [93] and also published after our work on the subject. They also try to discover the discriminative parts of an action through mid-level video representations, in which the candidates for parts of an action are clusters of trajectories. They use a graphical model that incorporates appearance and motion constraints for the individual parts and pairwise constraints for the spatiotemporal dependencies among them.

Chapter 7

Featured application: Selection of dynamic models for tracking

There are important limitations in the understanding of visual scenes. Many aspects are involved in the generation of natural image sequences: type of scene, point of view and camera motion, number and type of objects in the scene, their location, their appearance and pose, their motion and interactions, the acquisition conditions, etc. Understanding visual scenes requires being able explaining a huge variety of data. One way to address this problem is to factor the different sources of variability, and try to explain them separately or hierarchically. Motion is an example. The apparent 2D motion of objects in a video is strongly conditioned by the camera motion, as well as the type of scene and its geometry. Each object may have its own motion pattern in the context of the more global scene-dependent motion.

Our interest in this chapter is to model the latter via specialist dynamic models. The idea is that conditioning the choice of dynamic model on the "causes" of motion could improve tracking, provided that this leads to the choice of the best available model for each video. Such conditioning frees the specialist from having to cope with motions beyond its capabilities.

We propose that such specialist models are good priors that can lead to better tracking performance compared to standard general-purpose dynamic models such as Brownian and constant-velocity. Furthermore, we present a scenario in which the selection of the most appropriate dynamic model for a given video is automatized by means of a video classification system.

Our main contributions are **(i)** the design of a few specialized motion models, which only track well in scenes having similar geometry and camera motion, **(ii)** showing that a new video can be mapped to one of the available motion models automatically, **(iii)** a new dataset to evaluate 2D-point tracking, and critically **(iv)** experimental evidence of improved tracking performance when using the predicted motion model in unseen videos.

7.1 Related work

In this chapter we adopt the state-space approach to tracking [2]. We consider the evolution of the target as a dynamic system. All the relevant information about the system is contained in a state vector. Trackers can be seen as having two main components: a model describing the evolution of the state

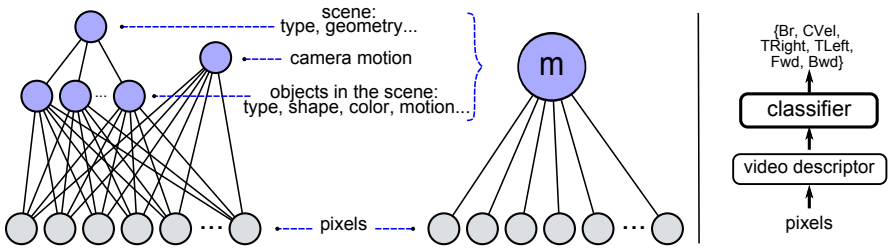


Figure 7.1: (Left) Illustration of the complex combination of causes that produce motion in a video. (Middle) Approximating the real graphical model, the many causes can be grouped into one more abstract variable. (Right) In practice, we quantize the causes into a discrete variable, and infer it using a discriminative model.

with time, i.e. the *dynamic model*, and a model relating the noisy observations to the state, i.e. the *measurement model*. We focus on choosing the dynamic model that is most appropriate for each video clip, though other related tracking research is summarized here as well.

There is a great amount of work on visual tracking proposing different measurement models. They are often related to visual appearance cues, such as contours [41] or color [89]. The measurement model can and also be related to apparent motion, i.e. optical flow [53, 97]. Indeed, an appearance descriptor can be enhanced by integrating simple motion information. For example, Kristan et al. [53] use a measurement model combining color and optical flow to resolve color-only ambiguities. They use a mixed dynamic model combining constant velocity for position and Brownian evolution for size. Čehovin et al. [122] build on that approach with a constellation of local visual parts, which get resampled spatially based on consistency with an object's higher-level appearance.

Online adaptation of the measurement model is an ongoing area of study, e.g. [42, 137], the main issue being the trade-off between adaptability versus drift. A time-weighted appearance model can be obtained through weighted online learning [144] that avoids drift and emphasizes most recent observations. When the tracked object is at least partially known, an appearance model can be learned *a priori*, or revised online, such as Holzer et al.'s online learning of linear predictors [34] or Grabner et al.'s online boosting tracker [32]. Tracking-by-detection uses temporal constraints to resolve the space of hypotheses that emerge when flexible appearance models are used. Kalal et al. [44] have an impressive example of a real-time system for solitary patches, while Prisacariu and Reid [91] demonstrate excellent tracking of even articulated shapes using level sets and nonlinear shape manifolds as shape priors for known object classes. Williams et al. [135] train a regressor which can localize the tracked target even with significant occlusions.

Most relevant to our supervised learning-based approach, Stenger et al. [113] assess the suitability of different measurement models for a particular tracking scenario, e.g. face or hand tracking. They use labeled training data of that scenario to calibrate the different measures of confidence given by each observation model, turning them into expected tracking error. By doing so, they can compare and combine the observation models reliably (e.g. in a cascade). Their experiments focus on coupling different models of appearance, so they replace the dynamic model by simple exhaustive search, and re-initialize lost tracks as needed. Yoon et al. [147] innovate in this vein. In contrast, we use a fixed

appearance model to better study the impact of choosing different motion models.

Comparatively little attention has been paid to dynamic models. Dynamic models can guide the search, tracking through brief occlusions and favoring observations with more likely motion. While it is very common to simply use Brownian or constant-velocity motion models, with parameters set *a priori*, their generality hurts them in many specific situations. Lourenço and Barreto [69] show the importance of an appropriate motion model in the case of scenes with radial distortion.

If tracking an object with a known nature, a physical model can better dictate the system's predictions [16]. If ground-truth tracks are available, Blake et al. [5] showed that it is possible to learn the parameters of a motion model for a given scenario. For simple cases, the initial tracks can come from an untrained tracker.

Rodriguez et al. [95] use topic models to learn long-term flow patterns from a sequence of a crowded scene, which improves re-tracking in the same scene. In subsequent work [97], they also cope with previously unseen sequences. For a given area, they retrieve training patches that appear similar. They use the learned flow patterns as a prior in the framework of a Kalman filter [45] with a constant-velocity motion model and a flow-based measurement model. The two components are weighted differently in light of the learned flow priors. In contrast, we handle motion models which are more complex than constant velocity, and without the assumptions imposed by the Kalman filter.

Buchanan and Fitzgibbon [9] obtain a robust motion prior from the global motion of 2D points in a time window, and apply it to re-track them. That motion model is based on rank constraints on the matrices explaining all the initial 2D tracks, without being restricted to rigid motions. While their model focused on translations of the individual points, more complex transformations are possible.

Outgrowing the closed-form learning that was possible in [5], North et al. [85] present an expanded algorithm to learn the parameters of more complex dynamics. Illustrated on the example of juggling, they model motion as a sequence of motion "classes", and concurrently learn the parameters of each motion class and the transition probabilities among classes.

The large variety of both motion models and available appearance models makes it difficult to choose among them when designing a system. Kwon and Lee [54] adaptively sample from a tracker space integrating appearance and motion models, selecting the tracker that provides the highest data likelihood. We approach a similar challenge, but choosing among very specialized motion models, and leveraging supervised learning.

Kowdle and Chen [52] also train classifiers based on both scene and camera dynamics, and use them together with shot-boundary detection to segment videos into clips of consistent motion.

From the model-selection point of view, our work is related to Mac Aodha et al.'s [70, 71]. They train a supervised classifier to predict the per-pixel success of different flow algorithms. Posteriorly to our work, Matikainen et al. [75] have presented the problem of choosing among very specialist models as a *recommentation* problem and successfully applied it to action recognition.

7.2 Dynamic models

We focus on four types of camera motion (traveling right, traveling left, moving forward, moving backward) and implement a specialized dynamic model for each. They are described below, together with two standard dynamic models used very frequently for tracking, namely Brownian motion and constant velocity. We establish six video categories that correspond to the six dynamic models, and aim to categorize each new video sequence to its most appropriate dynamic model. These coarse categories are simple enough to be automatically recognized much of the time, yet emerge as specific enough to give improved performance when tracking with the corresponding dynamic model. To validate our approach, a dataset (Section 7.3) of challenging videos was collected and augmented with manual annotations of correct tracks. Section 7.4 describes the video-clip classification process.

In order to compare the performance of the dynamic models on any given video, we choose the task of tracking 2D points. We adopt the standard Bayesian formulation for tracking and implement it as a particle filter [2]. Given the position of a 2D point in an initial frame, we want to know the position of that point in the subsequent frames. For each frame t, we estimate the posterior distribution of the state of the system $\mathbf{x}_t$ given the observed images $\mathbf{I}_{1..t}$ up to time t,

$$\underbrace{p(\mathbf{x}_t|\mathbf{I}_{1:t})}_{\text{posterior}} = \underbrace{p(\mathbf{I}_t|\mathbf{x}_t)}_{\text{measurement model}} \int \underbrace{p(\mathbf{x}_t|\mathbf{x}_{t-1})}_{\text{dynamic model}} \underbrace{p(\mathbf{x}_{t-1}|\mathbf{I}_{1..t-1})}_{\text{prior}} d\mathbf{x}_{t-1}, \tag{7.1}$$

where the posterior at time $(t-1)$ becomes the prior at time t. We now present the models in terms of equations describing the evolution of the system given the previous state. Implementation details can be found in Section 7.5.

Brownian (Br): This is the baseline dynamic model. The system's state vector is $\mathbf{x} = [x,\ y,\ w,\ h]$, where $[x, y]$ is the 2D position and $[w, h]$ are the patch width and height. Position and scale change only due to Gaussian noise. We keep the aspect ratio fixed, so

$$[x_{t+1},\ y_{t+1}] \sim \text{Norm}\left([x_t,\ y_t], \text{diag}(\sigma_x^2, \sigma_y^2)\right), \tag{7.2}$$

$$[w_{t+1},\ h_{t+1}] = s[w_t,\ h_t]; \quad s \sim \text{Norm}(0, \sigma_s^2). \tag{7.3}$$

Constant Velocity (CVel): This model is also a standard choice for tracking. We set $\mathbf{x} = [x,\ y,\ w,\ h,\ \dot{x},\ \dot{y},\ s]$, where $[\dot{x}, \dot{y}]$ are the displacements relative to the previous time-step, and s is the last increase in scale. Scale is assumed to be Brownian, so

$$[\dot{x}_{t+1},\ \dot{y}_{t+1}] \sim \text{Norm}\left([\dot{x}_t,\ \dot{y}_t], \text{diag}(\sigma_x^2, \sigma_y^2)\right), \tag{7.4}$$

$$[x_{t+1},\ y_{t+1}] = [x_t,\ y_t] + [\dot{x}_{t+1},\ \dot{y}_{t+1}], \tag{7.5}$$

$$[w_{t+1},\ h_{t+1}] = s_{t+1}[w_t,\ h_t]; \quad s_{t+1} \sim \text{Norm}(s_t, \sigma_s^2). \tag{7.6}$$

In our experiments we set $\sigma_y = \sigma_x$.

Traveling Right / Left (TRight / TLeft): These models are specialized to scenes in which the camera moves horizontally to the right (or left). The model can be written with the same equations as the

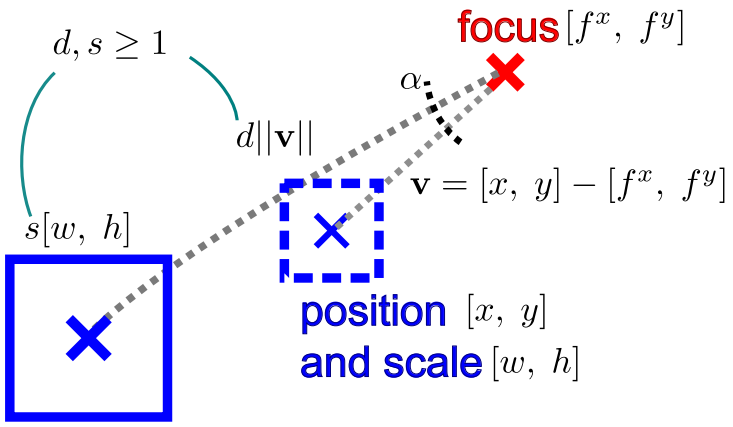


Figure 7.2: Illustration of the Forward motion model.

constant-velocity model, only $\sigma_y \ll \sigma_x$. The prior on the x displacement in TRight has the opposite sign to TLeft.

Forward / Backward (Fwd / Bwd): When the camera moves horizontally forward in a scene, some objects seem to appear at a 2D focus of expansion point $[f^x, f^y]$, and move away from it and grow as they get closer to the camera. The opposite happens when the camera moves backward. Based on validation data, we designed a dynamic model specialized for such scenes. The model assumes that the distance between a tracked point and its focus of expansion increases (or decreases) multiplied by a nearly-constant factor d. The patch size also has a nearly-constant increasing rate s. Both rates are coupled, i.e. they tend to be both greater or less than one. Fwd and Bwd models have opposite priors on these rates ($d, s \geq 1$ for Fwd and $d, s \leq 1$ for Bwd). The state vector is $\mathbf{x} = [x,\ y,\ w,\ h,\ d,\ s,\ f^x,\ f^y]$.

We model each 2D point as having its own focus of expansion (which is sampled from the previous time-step) and is moving from it in a nearly-straight line, passing through the current position. We account for slight deviations through $\alpha \sim \text{Norm}(0, \sigma_\alpha^2)$, which represents a tilt angle whose corresponding rotation matrix is $\mathbf{R}_\alpha$. See Figure 7.2. The state updates are

$$[f^x_{t+1},\ f^y_{t+1}] \sim \text{Norm}([f^x_t,\ f^y_t], \sigma_f^2 \mathbf{I}), \tag{7.7}$$

$$d_{t+1} = d_t + \sigma_d \epsilon_1; \quad \epsilon_1 \sim \text{Norm}(0, 1.0), \tag{7.8}$$

$$s_{t+1} = s_t + \sigma_s \epsilon_2; \quad \epsilon_2 \sim \text{Norm}(\epsilon_1, 0.1), \tag{7.9}$$

$$[x_{t+1},\ y_{t+1}]^T = d_{t+1}\mathbf{R}_\alpha \begin{bmatrix} x_t - f^x_{t+1} \\ y_t - f^y_{t+1} \end{bmatrix} + \begin{bmatrix} f^x_{t+1} \\ f^y_{t+1} \end{bmatrix}, \tag{7.10}$$

$$[w_{t+1},\ h_{t+1}] = s_{t+1}[w_t,\ h_t]. \tag{7.11}$$

Note that Brownian motion is a particular case of these two models, when the rates controlling scale and distance-to-focus are both one. Besides, there might be ambiguity between the Fwd and Bwd models: the same rectilinear motion can be explained as approaching a point in the direction of the motion or as moving away form a point placed in the opposite direction. This means that both models can similarly explain the observed motion of an object if the changes in scale are subtle. But when the

changes become more obvious, the wrong combination of focus position and motion direction will loose track of the object.

7.3 Dataset

While datasets of manually tracked objects exist (e.g. [123]), we needed self-consistent video clips spanning a variety of camera motions, and ground-truth 2D tracks for points rather than objects. Our dataset consists of over 100 challenging real-world videos from YouTube, encompassing various scenarios, illuminations, and recording conditions. They typically last 3-90 seconds. Through our annotation tool, users picked at least 10 interest points per video, among all FAST [98, 99] interest points. They marked each point's $[x, y]$ position in at least 10 subsequent frames, sparsely in time if possible. These annotations serve for assessing tracking performance, as described in Section 7.5.3.

We manually obtain *inspection-based* labels for each video as one way to train a supervised classifier. We ourselves inspected the videos and judged whether a clip belonged predominantly to one of the four camera motions or might be better served by the Br or CVel models. For the TRight and TLeft categories, we also include the flipped video in the dataset. In total there are 12 videos labeled as Br, 11 as CVel, 11 (+ 17 flips) as TRight, 17 (+ 11) as TLeft, 24 as Fwd, and 14 as Bwd. The flipped videos provide some symmetry with regard to the difficulty of tracking-left and tracking-right videos. Such videos look similar but their different target labels make classification harder. Some example frames of the videos in our dataset, excluding the flipped versions, are showed at the end of the chapter.

An alternative way of obtaining supervision is *performance-based* labeling. In this case, for a given set of available motion models and parameter settings, a video is labeled with the one that performed best according to the annotations of ground-truth tracks.

Note that the specialized motion models have been designed previous to obtaining this benchmark data, and in particular, previous to establishing the *inspection-based* (and potentially subjective) target categories. The default parameters in our models were not tuned for this data.

7.4 Video description and classification

We use the local DenseTrack video features presented by Wang et al. [124] for the task of action recognition (see Section 3.1.2). We quantize these local features using Extremely Randomized Trees [30] as in [80] and obtain visual-word vocabularies. For each type of descriptor, we train five trees of around 1000 leaves each. For the experiments, rather than computing the vocabulary on our dataset, we trained on completely different data, namely the action recognition UCF YouTube dataset [67]. We obtain global video signatures by computing normalized histograms of visual words, one per type of descriptor.

For video classification, we use the standard LIBSVM [13] multiclass approach, consisting of an all-pairs biclass SVM and majority voting. We pre-compute χ^2 kernels for each type of signature, and combine different types of signatures through the sum of the kernels. We have assumed a constant motion model for the whole of each video, but one could easily imagine a sliding-window extension to deal with longer, more varied videos.

We aim at classifying videos into one of six categories corresponding to the six dynamic models

Table 7.1: Parameter settings.

	Br	CVel	TRight	TLeft	Fwd	Bwd
σ'_x	3	1	1		-	
σ'_y	3	1	0.1		-	
σ'_s	0.03	0.03	0.006		0.005	
σ'_d	-	-	-		0.01	
σ'_f	-	-	-		1	
σ'_α	-	-	-		$\pi/180$	
σ_0	4					
σ_n	{0.1, 0.2, 0.5, 1, 2, 5} (default = 1)					
$\dot{x}_0$	-	0	-2	2	-	
$\dot{y}_0$	-	0	0		-	
s_0	-	1	1		1.005	0.995
d_0	-	-	-		1.01	0.99
f_0^x, f_0^y	-	-	-		N, S, E, W of frame	

in Section 7.2. Such supervised classifiers need a training label for each video, which we obtain by inspection as explained in Section 7.3.

7.5 Tracking implementation

We implement Eq. 7.1 as a particle filter. Our motion models have higher-dimensional state vectors than the baselines and therefore might enjoy a larger number of particles, but we fix all models to 50 particles so that the comparisons are fair in terms of allocated resources.

The state vector $\mathbf{x}_t$ contains at least the 2D position of the point $[x_t, y_t]$ and the dimensions $[w_t, h_t]$ of a patch around it. We evaluate trackers on the basis of only the position estimates that they produce, but width and height are also needed by the measurement model.

7.5.1 Dynamic models

In this section we detail the *state prior* for each dynamic model and the *parameter values* used in the experiments. We decompose every standard deviation in our models (σ_x, σ_y, etc.) into two factors

$$\sigma_* = \sigma_n \sigma'_* \tag{7.12}$$

where σ'_* is fixed and σ_n is a common parameter controlling the global lever of noise in the model. The standard deviations in the prior distributions are also all controlled by a factor σ_0. Parameter settings are summarized in Table 7.1.

Brownian: The initial state vector $\mathbf{x}_1 = [x_1,\ y_1,\ w_1,\ h_1]$ is fully defined by the input, so no prior distribution is used.

Constant Velocity: The four first components of the initial state vector $\mathbf{x}_1 = [x_1, y_1, w_1, h_1, \dot{x}_1, \dot{y}_1, s_1]$ are fully defined by the input, and the rest sampled from the following prior distribution

$$[\dot{x}_1,\ \dot{y}_1] \sim \text{Norm}\left([\dot{x}_0,\ \dot{y}_0], \sigma_0^2 \text{diag}(\sigma_x^2, \sigma_y^2)\right), \tag{7.13}$$

$$s_1 \sim \text{Norm}(s_0, (\sigma_0 \sigma_s)^2). \tag{7.14}$$

Traveling Right / Left: The prior distribution for horizontal displacement is

$$\dot{x}_1 \sim \text{Norm}\left(\sigma_0 \sigma_n \dot{x}_0, (\sigma_0 \sigma_x)^2\right). \tag{7.15}$$

Forward / Backward: The prior distributions are

$$[f_1^x,\ f_1^y] \sim \text{Norm}([f_0^x,\ f_0^y], (\sigma_0\sigma_f)^2\mathbf{I}), \tag{7.16}$$

$$d_1 = d_0 + (\sigma_0\sigma_d)\epsilon_1; \quad \epsilon_1 \sim \text{Norm}(0, 1.0), \tag{7.17}$$

$$s_1 = s_0 + (\sigma_0\sigma_s)\epsilon_2; \quad \epsilon_2 \sim \text{Norm}(\epsilon_1, 0.1). \tag{7.18}$$

The initial position of the focus of expansion $(f_0^x,\ f_0^y)$ is sampled from five points on the northern, southern, right, left and central areas of the frame.

7.5.2 The measurement model

The measurement model estimates the likelihood of the observed image given a state. Given the initial position of the 2D point $[x_1, y_1]$ in the initial frame $\mathbf{I}_1$, we extract a 15×15-pixel patch around it and keep as the reference template. The likelihood of any state in any subsequent frame t is measured by first extracting a patch from image $\mathbf{I}_t$ at the prescribed position and scale. Once interpolated, the patch is scored based on normalized cross-correlation (NCC) to the reference template,

$$p(\mathbf{I}_t|\mathbf{x}_t) \propto \text{cal}(\text{NCC}(\text{patch}_{\text{ref}}, \text{patch}_t)). \tag{7.19}$$

The mapping $\text{cal}(\cdot)$ aims at being a calibration, transforming raw NCC values to be proportional to likelihood values. $\text{cal}(\cdot)$ is adapted to each tracked point, and is computed in the first frame by measuring the NCC between the template patch and a number of neighboring patches.

We extract patches within a neighboring zone around the template, and compute both their area overlap with the template and their NCC. The aim is to penalize NCC values that despite being high correspond to patches with low area overlap. To do so, we clip all overlap values below 70% to a very low value of 5%. We assimilate this clipped overlap to likelihood and assume we now have pairs of likelihoods and NCC. Any regression technique can now be used to map NCC to likelihood. We choose a very simple technique[1] that is fully data-driven, parameter-free and forces the mapping to be monotone, i.e. increasing likelihood for increasing NCC values.

7.5.3 Performance evaluation

We evaluate each tracker's performance in terms of robustness on our dataset, using the ground-truth points that have been manually labeled as described in Section 7.3. Let $p_i = [x_i, y_i]$, $i \in [1, \ldots, N]$ be the position of the N ground-truth points in a given video. Let $\hat{p}_i$ be the position of the reference point p_i estimated by the tracker. We compute the percentage of successful track estimates, i.e. the fraction of point-positions that match the ground-truth locations within a certain level of precision, so

$$\text{robustness} = \frac{1}{N}\sum_{i=1}^{N} \triangle_\theta(p_i, \hat{p}_i); \quad \triangle_\theta(p_i, \hat{p}_i) = \begin{cases} 1 & \text{if } |x_i - \hat{x}_i| < \theta \text{ and } |y_i - \hat{y}_i| < \theta \\ 0 & \text{otherwise.} \end{cases} \tag{7.20}$$

This measure of performance is averaged over two runs of tracking. The trackers are not re-initialized after loss of track, so early loss of track is typically more costly.

[1] Pair-adjacent-violators (PAV) algorithm for isotonic regression. Described in (e.g.) http://sfb649.wiwi.hu-berlin.de/fedc_homepage/xplore/ebooks/html/anr/anrhtmlnode43.html

To obtain the global performance of the whole dataset, we average the robustness obtained when using the motion model that was selected for individual videos.

θ is 10 in all our experiments. This is a tolerant threshold so inaccurate versus failed tracking can be distinguished.

7.6 Experiments and results

This section experimentally validates our approach, showing that we can use machine learning techniques in order to select among dynamic models given low level features. The objective is to significantly improve tracking performance. We show the tracking performance of the six dynamic models, measuring how each performs on its own. We finally report the performance of the video classification algorithm and, critically, the results of point trackers using motion models predicted by the video classifier.

7.6.1 Tracking robustness of individual motion models

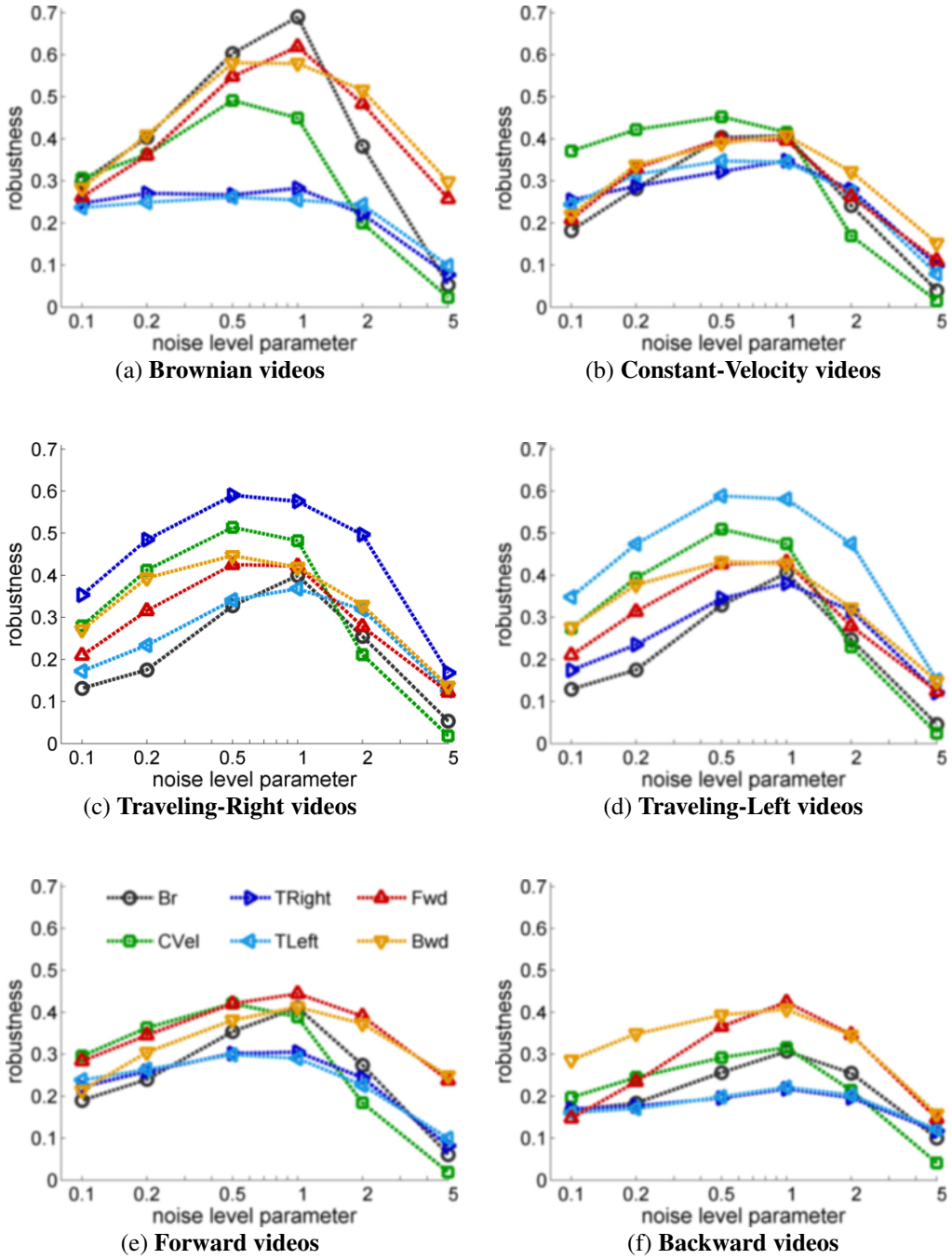
The performance of the six motion models is computed over the whole dataset. As shown in Table 7.2, no individual model performs extraordinarily overall. The hypothesis that ideally, each video should be processed using the most appropriate motion model is explored through the following tests.

We estimate the *idealized* performance that is obtained if, among the available motion models, an oracle chose the one that actually performs best on each video. The ideal performance is already better when there is a choice between just the two standard motion models (0.493). Enlarging the collection with our four specialized models raises the ceiling further (0.561). Knowing the rather subjective inspection-based labels yields a best-case score of 0.526, which is worse than the six-way performance-based oracle, but a significant improvement nonetheless.

Figure 7.3 gives insight into the behavior of the different motion models. It shows the tracking robustness obtained by each motion model for different parameter settings. The videos are grouped according to their inspection-based labels, and the performance has been averaged within groups. One can see the inclination of most groups toward their corresponding dynamic model. For example, TRight and TLeft are poor in general, but they largely outperform the rest on their corresponding families of videos. On the Constant Velocity videos, CVel outperforms other models over a broad range of parameter settings. Between the two baselines, CVel tends to outperform Br, but there is no clear winner as it depends of the parameter settings. Br performs better than CVel on videos labeled as Brownian. Fwd and Bwd perform better than Br in general, except for videos labeled as Brownian, on which they are nevertheless more robust when the level of noise in the models is high. Fwd vs. Bwd and TRight vs. TLeft only differ in their respective priors, so it is less surprising that they have similar performance profiles.

7.6.2 Tracking robustness using classification

The previous experiments showed how predicting the most appropriate motion model would lead to significant improvement of tracking robustness. We propose that such predictions can be made using a set of training videos, assigning them to categories that encapsulate shared motion properties, and using



(a) **Brownian videos** (b) **Constant-Velocity videos**

(c) **Traveling-Right videos** (d) **Traveling-Left videos**

(e) **Forward videos** (f) **Backward videos**

Figure 7.3: Average tracking robustness of each motion model (see legend) on each inspection-labeled group of videos. Different parameter settings, from low to high values of noise parameters σ_x, σ_y, σ_s, σ_d, etc. are controlled by a single noise level parameter (default = 1). Considering peak-performance and area under the curves, groups are partial to their respective motion models. Motion models differ in their robustness to parameter settings.

Table 7.2: White background: average tracking robustness of each individual motion model over all videos (default parameters). Bottom row: percentage of times *that* motion model (among six) was the best choice. Dark gray: best-case results if model is selected by either a performance-based oracle, or our inspection-based labels. Right column: tracking each video using our classifier's suggested motion model, using inspection-based training data.

	Individual motion models						Ideal predictions			Our method
	Br	CVel	TRight	TLeft	Fwd	Bwd	best{Br, CVel}	best{all}	manual labels	
tracking robustness ($\cdot 10^{-2}$)	42.3	43.2	37.9	37.2	44.7	43.7	49.3	56.1	52.6	51.9
$\pm$ std. dev. random runs	0.4	0.4	0.7	0.5	0.2	0.1	0.6	0.4	0.2	0.1
% best choice (± 2)	21	11	12	20	20	16	32	100	52	50

Table 7.3: Classifiers trained using different descriptors. First row: classification accuracy with respect to the target inspection-based labels (leave-one-out training). Below: tracking robustness obtained using the model predicted for each sequence. Note how features other than HOG lead to tracking robustness that is superior to the general-purpose motion models.

	HOG	Traj	HOF	MBH	Traj + HOF + MBH	Target labels
classification accuracy (%)	30	84	85	83	91	–
tracking robustness ($\cdot 10^{-2}$)	42.2	52.2	52.09	50.5	51.9	52.6
$\pm$ std. dev. random runs	0.6	0.1	0.01	0.4	0.1	0.2

a classifier to predict the category of a new video at test time.

As explained in Section 7.4, we train multiclass SVMs using different video descriptors and the *inspection-based* category labels, then evaluate them in a leave-one-out manner so that each video in the dataset acts as the test video once. This produces a motion category prediction for each video. Table 7.3 shows the classification performance of the different video descriptors with regard to the target inspection-based labels. The table also shows the tracking robustness obtained when tracking with the predicted motion model.

As expected, the HOG descriptor, which encodes only local appearance, is not very useful in this classification task. Ignoring HOG, the combination of three motion-related descriptors yields the best classification accuracy. However, this does not translate into the best tracking performance. Not all the classification mistakes are equally costly in terms of tracking. Classification, by design, is optimized here for mutually-exclusive classes and a binary 0-1 loss, i.e. the five possible types of error incur the same penalty. In practice, confusing TRight for TLeft could hurt tracking much more than picking CVel, for example.

Performance-based categories. We also trained a classifier on performance-based labels. Classification accuracies are much poorer (43% for six categories), indicating that this is a harder classification task than the one targeting inspection-based labels. Nevertheless, the tracking performance is comparable (0.529), which can be explained by the fact that the "best{all}" ideal predictions in Table 7.2 set a higher ceiling. Note that the human effort to obtain the inspection-based labels is much lower, as no manual ground-truth point tracks are needed, only visual inspection and some generic knowledge of the models.

7.6.3 Heuristic model selection

We experimented with a technique to select motion models based on track lengths. A way to assess the confidence in a track is to look at the output of the measurement model $p(\mathbf{I}_t|\mathbf{x}_t)$ at each time-step. One can threshold this likelihood to decide that track has been lost and establish the track length.

A straightforward way of selecting a dynamic model for a query video is to try them all and choose the one that provided more likely or longer tracks.

We put in practice this idea by taking 10 random points in the video and tracking them for as long as possible. Rather than putting a hard and somewhat arbitrary threshold on the likelihood, we compute for each tracked point the sum of likelihoods obtained at each time step. This amounts to a weighted track length.

We average this weighted track length over the 10 points, and select a motion model based on this value. We call it heuristic because there is no guaranty that this confidence is robust to the typical difficulties of tracking, e.g. appearance changes, repetitive structure, etc.. The obtained results are shown in Table 7.4.

Table 7.4: Average tracking robustness using a model selection heuristic based on track lengths, compared to our method.

	heuristic	predicted model
tracking robustness (%)	46.6	51.9
% best choice	33	50

7.6.4 Customized vocabularies

If we use customized vocabularies (i.e. trained on our dataset) rather than pre-computed ones, the classification accuracy is noticeably higher for each descriptor (up to 92 %), and this has indeed a slight impact on the tracking performance (0.525). More details in Table 7.5.

Table 7.5: Classification accuracy and tracking robustness of classifiers trained with different descriptors. The vocabulary of visual words was computed on the target dataset.

	HOG	Traj	HOF	MBH	Traj + HOF + MBH	Target labels
classification accuracy (%)	40	85	88	87	92	–
tracking robustness ($\cdot 10^{-2}$)	42.9	52.1	52.3	50.8	52.5	52.6

7.7 Application to longer videos

In the previous sections, we have presented a method that treats each query video as a whole: features are extracted throughout the whole clip, a global signature is computed, and then a single dynamic model is predicted from it. This setting makes sense if the query video is short or regular enough to reasonably assume that a single dynamic model explains it well.

Instead, we can image an online scenario, in which we want to do tracking in longer videos with potentially varying dynamics. In that case we can still apply our system in a sliding window manner. As frames are made available, we can extract features, and then compute signatures for short, potentially overlapping time intervals, by taking only the features from the corresponding frames. A dynamic model can be predicted for each time interval, and the particle filter can then be updated accordingly. Note that some latency would be introduced by the feature extractor, the signature computation and classification. The time consumed by these stages would determine its applicability in real time.

Pushing this idea further, the localized analysis could be done not only in temporal intervals but also in spatial windows, letting different regions of the video to be explained by different simple dynamic models. Each particle set corresponding to one of such regions, the appropriate dynamic model could be readily used in the particle filter.

The main condition to make these extensions feasible is to have at training time some ground-truth pairs of segmented clips and a their corresponding dynamic model, as we have in our current training dataset.

7.8 Discussion

We have presented a simple approach to choose among specialized dynamic models. Pre-classifying each video leads to improved tracking robustness, requiring prior training on separate data and only weak supervision in the form of inspection-based labels. Brownian motion and constant-velocity models have comparable tracking robustness in our experiments, while guidance from our inspection- and performance-based classifiers improves this baseline by over 20%.

The experiments using performance-based supervision highlight that considering the categories as mutually exclusive may be unnecessarily distracting to the classifier. The 0-1 penalty function used during training does not account for the fact that, depending on the video, a misclassification error may have a different impact than another in terms of tracking performance. One way to address this issue could be to employ regression or structured output spaces to predict the relative success of the motion models, but our initial experiments were disappointing.

We have used a particle filter and a simple measurement model only as a platform for comparing dynamic models. Classifying a video according to its dynamic model could be used in combination with (and should be complimentary to) any tracking approach, and may simplify the challenges associated with, for example, motion segmentation [19].

Our classification-based dynamic model selection allows us to automatize the choice of an appropriate dynamic model in the context of videos with very distinctive motion for which reliable motion models exist. It could also prove useful to apply our method to local regions or in a sliding-window fashion, so as to allow for an online application and deal with videos containing several types of motion, as described in Section 7.7. However, there is a number of limitations that would restrict its use for general tracking. One is the availability of such specialized dynamic models. Here we do not address the automatic discovery of target classes or the design of the corresponding adapted models from data. A first step in this direction would be related to Kitani et al.'s algorithm [49] to cluster videos into cate-

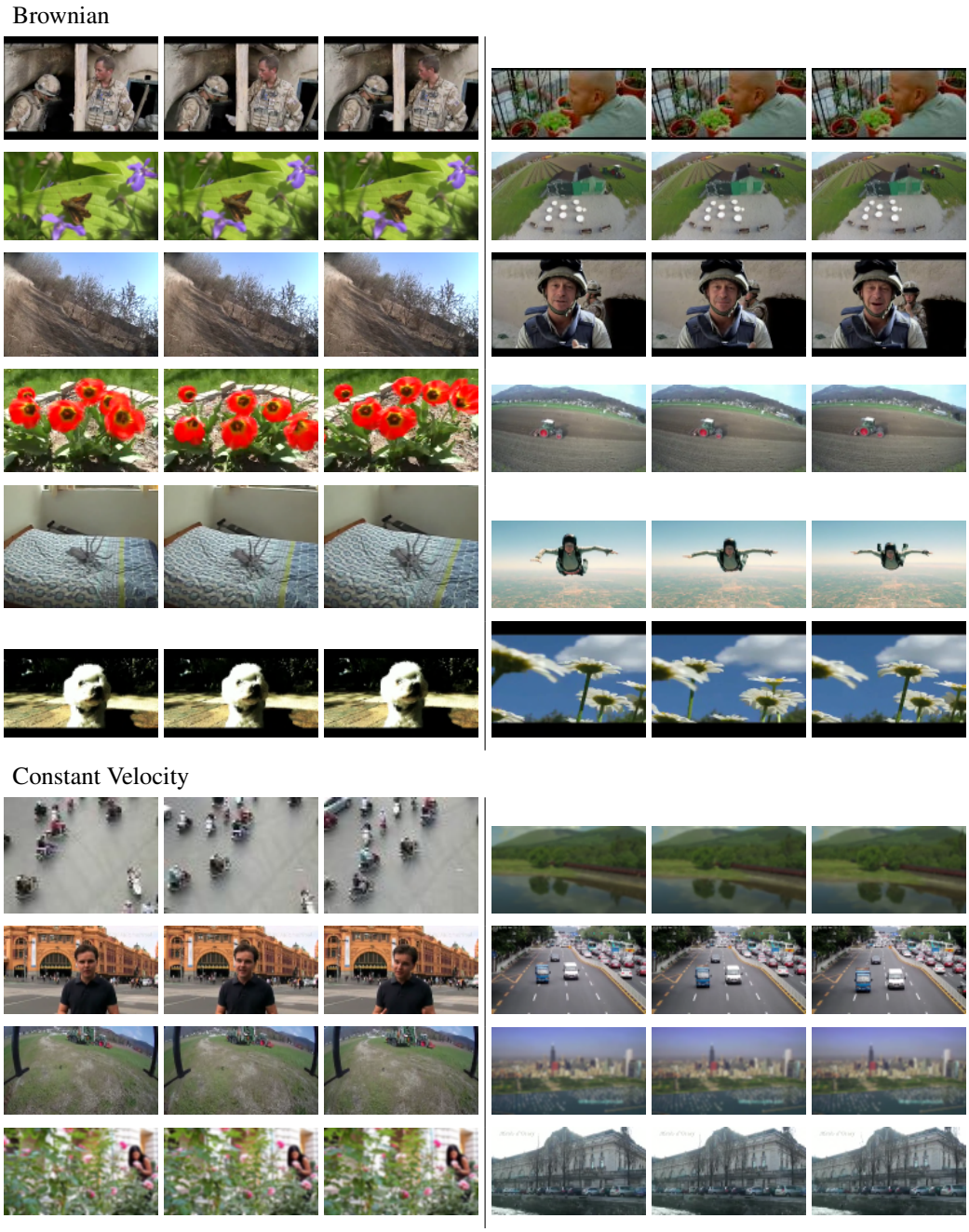
gories of first-person actions in a fast and unsupervised way. If many dynamic models could be designed systematically, then videos could be mapped to both motion models and bespoke parameters.

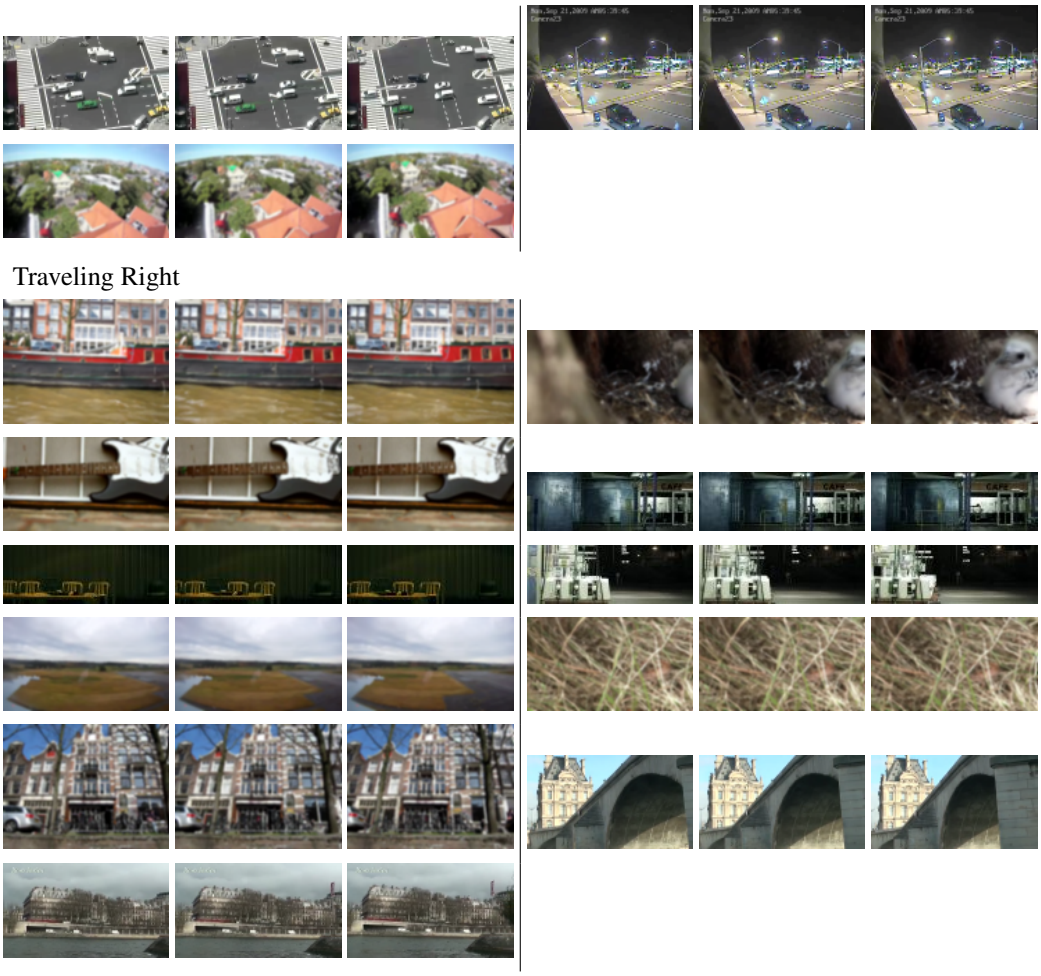
Dynamic model selection based only on "how well the data is explained" (i.e. based on the measurement model) is not reliable in general, as the measurement model is not an absolute measure of ground truth. This estimation of performance may derive and eventually lead to loose track of the object. Our approach arises from the observation that the tracker has no way of knowing if its doing fine or not. Therefore we go for the alternative reasoning "I know how to track because I have seen this situation before" and try to make link the query video to some other training video (for which we potentially have ground truth). This approach is basically transferring the key to success from the reliability of the measurement model to the reliability of the training data. Therefore, another limitation would be the availability of reliable training data. This approach would fail is there is no reliable mapping between new videos and training videos (i.e. the feature vector computation) and between training videos and target classes (i.e. the labels). This drawback is common to any image recognition system relying heavily in statistical learning.

Despite these limitations, while one can imagine other optimization methods to select among dynamic models, we have not found in our scenario one as effective as the proposed classification approach. We explored a heuristic based on track lengths, which scored half way between the general-purpose baseline models and our method. Further, classification has the potential to scale well when the number of motion models increases, making the running of each model in a try-and-see approach more costly than extracting and classifying a feature vector.

Overall, we have shown that it is possible to map local video features to dynamic models, by-passing the constraint to blindly rely on an arbitrary measurement model. Despite the limitations in practical use, there is good reason to expect that further very specialized motion models, that are not good in general but outstanding under known circumstances, are worth developing.

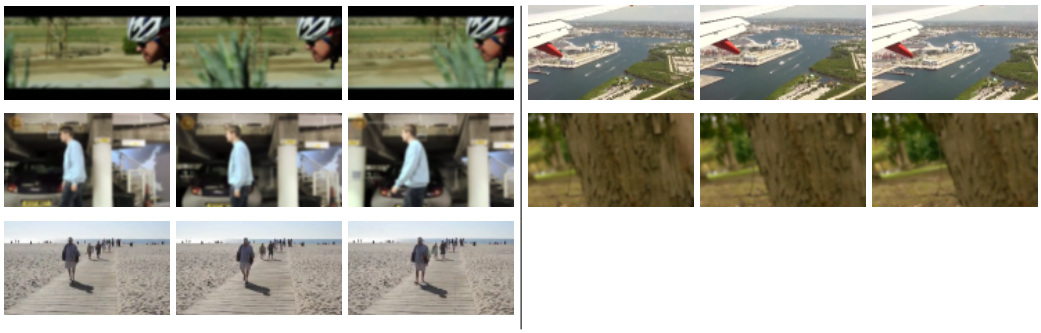
Figure 7.4: Example frames of our dataset, excluding flipped clips.



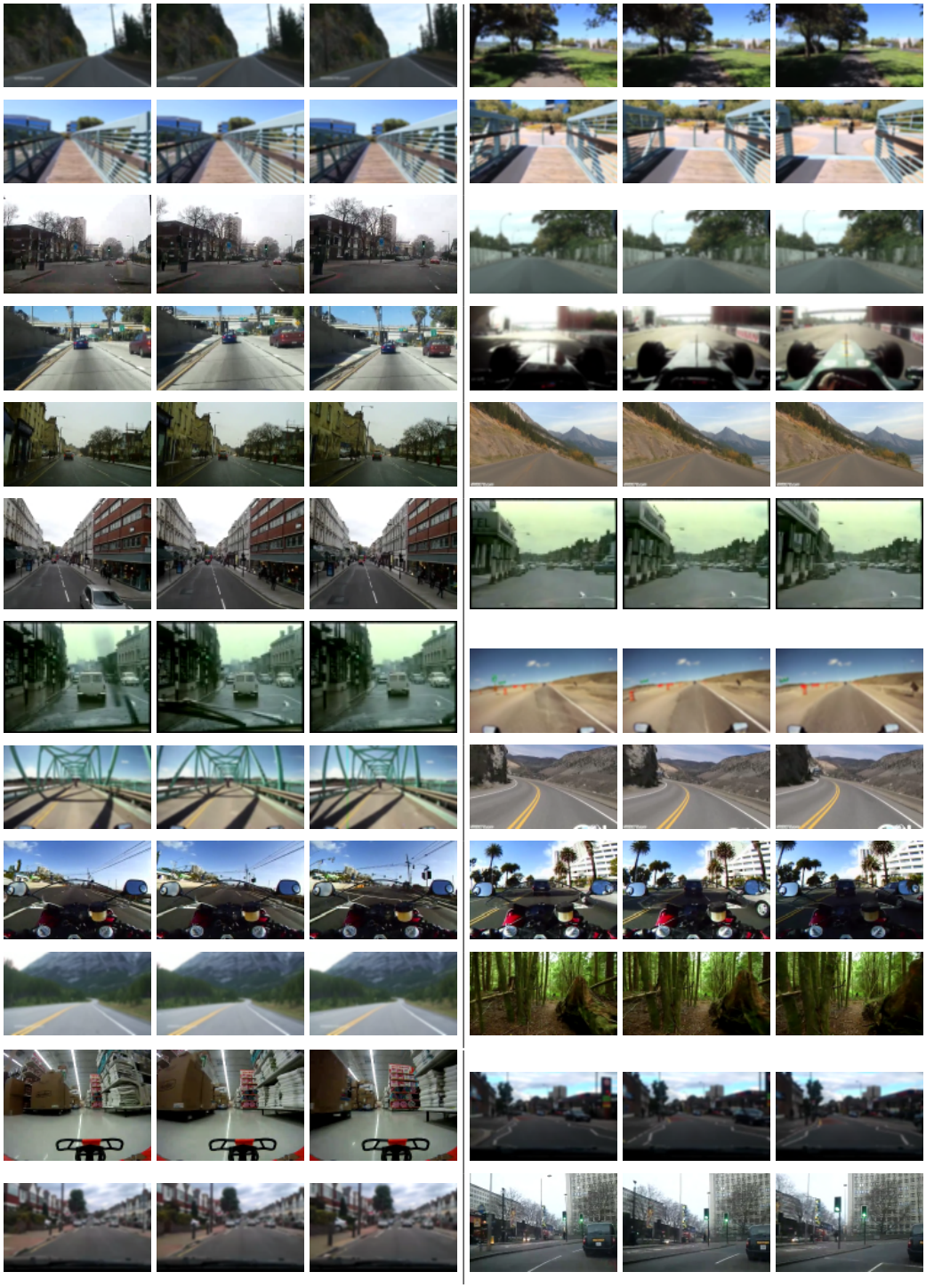


Traveling Right

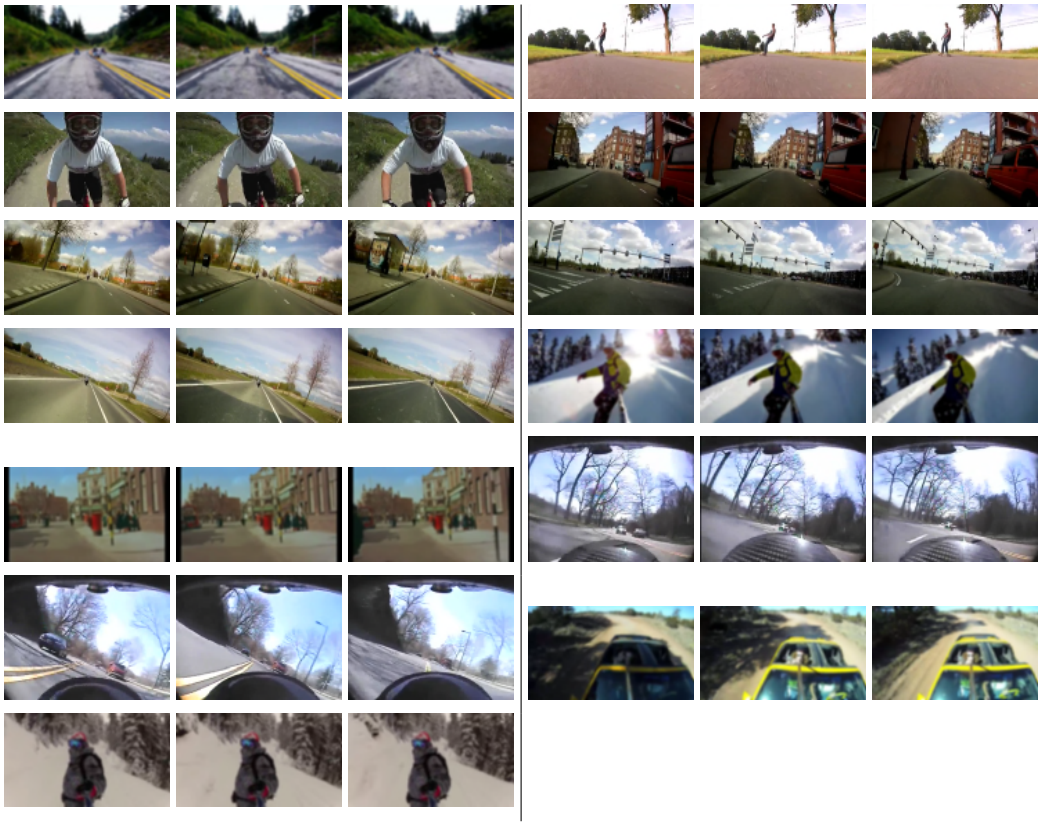
Traveling Left



Forward



Backward



Chapter 8

Conclusion

Bag-of-words representation of videos had a big success in the first action recognition datasets, which contained actions with periodic motion (such as walking, running, waving, etc.) to which histograms respond particularly well. With recent more realistic datasets of actions in the wild, bag of words is still a reference baseline, but its limitations to capture the spatiotemporal structure of local features, or to filter by location the relevant information, for example, leave room for more elaborated representations.

Our research has focused on extending the bag of words framework to better suit the problem of video classification with weak labels, and applying supervision in a more consistent way with regards the vague information provided by such labels.

We aimed at leveraging weak annotation from early stages in the pipeline. In particular, we adopted the random forests method as local feature quantizer, and analyzed the influence of its parameters. Traditional forests provide some supervision, but are not particularly optimized for this quantization task. Therefore we proposed and evaluated a new supervision scheme that explicitly aims at increasing the discriminative power of the final bag-of-words video descriptions. Our new forests proved particularly superior to traditional ones at incorporating contextual information into the local quantization process. However, we have our reservations about the real impact on performance of this enhanced supervision, specially considering the extra computation needed.

We also explored a mid-level representation of videos, mainly motivated but the fact that different regions in the video may be more or less relevant, information which is not conveyed by the weak labels. Many questions arise that deserve more principled approaches and more exhaustive evaluation, but our preliminary results show that our mid-level representation of videos do carry useful information that complements the bag of words and therefore improves classification performance. This agrees with simultaneous work that has been published after our own research on the subject.

Finally, we have proposed a novel application of video classification to improve tracking. A main issue about tracking is the reliability of different measurement models, the difficulty to assess "on the fly" how good tracking is so as to make corrections and avoid deriving. Our contribution consists of proposing video classification as a strong prior on the appropriate dynamic model. In particular, we train classifiers to predict the dynamic motion model to use in a video, among a predefined list, so as to perform what we call classification-based model selection. If appropriate specialized models and training

videos exist, analyzing local features can give an important clue to perform model selection, not relying in measurement models but in similarities to previously seen videos of the same category. We show that this is feasible for a simple scenario with dynamic models corresponding to rough camera motion. This leaves the door open to incorporating this kind of statistical learning into more sophisticated tracking schemes.

Additional axes for future research are to explore the possibilities of conditioned quantization by means of random forests in order to incorporate context or to encode spatiotemporal relationships – which could be useful also in the 2D image domain, and to continue towards making more efficient and computationally feasible the computation of mid-level representations of videos that can provide better classification and understanding through the implicit localization of relevant regions.

Bibliography

[1] S. Ali, A. Basharat and M. Shah. Chaotic invariants for human action recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[2] M. S. Arulampalam, S. Maskell and N. Gordon. A tutorial on particle filters for online nonlinear / non-Gaussian Bayesian tracking, *IEEE Transactions on Signal Processing* **50**, 174–188, 2002.

[3] F. Bach, G. R. G. Lanckriet and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *International Conference on Machine Learning (ICML)*, 2004.

[4] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft. When is "nearest neighbor" meaningful? In *International Conference on Database Theory*, 1999.

[5] A. Blake, M. Isard and D. Reynard. Learning to track the visual motion of contours, *Artificial Intelligence* **78**, 101–134, 1995.

[6] M. Blank, L. Gorelick, E. Shechtman, M. Irani and R. Basri. Actions as space-time shapes. In *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[7] M. Blank, L. Gorelick, E. Shechtman, M. Irani and R. Basri. Actions as space-time shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29**(12), 2247–2253, 2007.

[8] M. Bregonzio, S. Gong and T. Xiang. Recognising action as clouds of space-time interest points. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2009.

[9] A. Buchanan and A. W. Fitzgibbon. Combining local and global motion models for feature point tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[10] C. J. Burges. A tutorial on Support Vector Machines for pattern recognition, *Data Mining and Knowledge Discovery* **2**, 121–167, 1998.

[11] S. Calderara, R. Cucchiara and A. Prati. Action signature: a novel holistic representation for action recognition. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2008.

[12] L. Cao, J. Luo, F. Liang and T. S. Huang. Heterogeneous feature machines for visual recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[13] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines, *ACM Transactions on Intelligent Systems and Technology* **2**, 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[14] Y. Chen, J. Bi and J. Z. Wang. MILES: Multiple-instance learning via embedded instance selection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**(12), 1931–1947, 2006.

[15] A. Criminisi, J. Shotton and E. Konukoglu, 2011, Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning, Technical Report MSR-TR-2011-114, Microsoft Research.

[16] A. Cuzol and É. Mémin. A stochastic filter for fluid motion tracking. In *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[17] J. W. Davis and A. F. Bobick. The representation and recognition of action using temporal templates. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1997.

[18] P. Dollár, V. Rabaud, G. Cottrell and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VSPETS)*, 2005.

[19] R. Dragon, B. Rosenhahn and J. Ostermann. Multi-scale clustering of frame-to-frame correspondences for motion segmentation. In *European Conference on Computer Vision (ECCV)*, 2012.

[20] O. Duchenne, I. Laptev, J. Sivic and F. B. andJean Ponce. Automatic annotation of human actions in video. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[21] A. A. Efros, A. C. Berg, G. Mori and J. Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision (ICCV)*, 2003.

[22] A. Elgammal, V. Shet, Y. Yacoob and L. S. Davis. Learning dynamics for exemplar-based gesture recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. The Pascal visual object classes (VOC) challenge, *International Journal of Computer Vision* **88**(2), 303–338, 2010.

[24] A. Fathi and G. Mori. Action recognition by learning mid-level motion features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[25] P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan. Object detection with discriminatively trained part based models, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**(9), 1627–1645, 2010.

[26] J. Foulds and E. Frank, 2008, Revisiting multiple-instance learning via embedded instance selection, In *AI 2008: Advances in Artificial Intelligence*, Vol. 5360 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp.300–310.

[27] E. B. Fox, E. B. Sudderth, M. I. Jordan and A. S. Willsky. Sharing features among dynamical systems with beta processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[28] A. Gaidon, Z. Harchaoui and C. Schmid. Actom sequence models for efficient action detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[29] P. Gehler and O. Chapelle. Deterministic annealing for multiple-instance learning. In *11th International Conference on Artificial Intelligence and Statistics*, 2007.

[30] P. Geurts, D. Ernst and L. Wehenkel. Extremely randomized trees, *Machine Learning* **36**(1), 3–42, 2006.

[31] A. Gilbert, J. Illingworth and R. Bowden. Fast realistic multi-action recognition using mined dense spatio-temporal features. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[32] H. Grabner, C. Leistner and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *European Conference on Computer Vision (ECCV)*, 2008.

[33] M. Hoai, Z.-Z. Lan and F. de la Torre. Joint segmentation and classification of human actions in video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[34] S. Holzer, M. Pollefeys, S. Ilic, D. Tan and N. Navab. Online learning of linear predictors for real-time tracking. In *European Conference on Computer Vision (ECCV)*, 2012.

[35] T. Hospedales, S. Gong and T. Xiang. A markov clustering topic model for mining behaviour in video. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[36] Y. Hu, L. Cao, F. Lv, S. Yan, Y. Gong and T. S. Huang. Action detection in complex scenes with spatial and temporal ambiguities. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[37] H. Hung and S. Gong. Quantifying temporal saliency. In *British Machine Vision Conference (BMVC)*, 2004.

[38] N. Ikizler-Cinbis, R. G. Cinbis and S. Sclaroff. Learning actions from the web. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[39] N. Ikizler-Cinbis and S. Sclaroff. Object recognition and localization via spatial instance embedding. In *International Conference on Pattern Recognition (ICPR)*, 2010a.

[40] N. Ikizler-Cinbis and S. Sclaroff. Object, scene and actions: Combining multiple features for human action recognition. In *European Conference on Computer Vision (ECCV)*, 2010b.

[41] M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking, *International Journal of Computer Vision* **29**, 5–28, 1998.

[42] A. D. Jepson, D. J. Fleet and T. F. El-Maraghi. Robust online appearance models for visual tracking, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**, 1296–1311, 2003.

[43] H. Jhuang, T. Serre, L. Wolf and T. Poggio. A biologically inspired system for action recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[44] Z. Kalal, J. Matas and K. Mikolajczyk. Tracking-learning-detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 2011.

[45] R. E. Kalman. A new approach to linear filtering and prediction problems, *Transactions of the ASME–Journal of Basic Engineering* **82**(Series D), 35–45, 1960.

[46] Y. Ke, R. Sukthankar and M. Hebert. Efficient visual event detection using volumetric features. In *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[47] Y. Ke, R. Sukthankar and M. Hebert. Event detection in crowded videos. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[48] F. S. Khan, J. van de Weijer and M. Vanrell. Top-down color attention for object recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[49] K. M. Kitani, T. Okabe, Y. Sato and A. Sugimoto. Fast unsupervised ego-action learning for first-person sports videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[50] A. Kläser, M. Marszałek and C. Schmid. A spatio-temporal descriptor based on 3D-gradients. In *British Machine Vision Conference (BMVC)*, 2008.

[51] A. Kovashka and K. Grauman. Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[52] A. Kowdle and T. Chen. Learning to segment a video to clips based on scene and camera motion. In *European Conference on Computer Vision (ECCV)*, 2012.

[53] M. Kristan, J. Perš, S. Kovačič and A. Leonardis. A local-motion-based probabilistic model for visual tracking, *Pattern Recognition* **42**(9), 2160–2168, 2009.

[54] J. Kwon and K. M. Lee. Tracking by sampling trackers. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[55] I. Laptev. On space-time interest points, *International Journal of Computer Vision* **64**(2/3), 107–123, 2005.

[56] I. Laptev and T. Lindeberg. Space-time interest points. In *IEEE International Conference on Computer Vision (ICCV)*, 2003.

[57] I. Laptev and T. Lindeberg. Local descriptors for spatio-temporal recognition. In *European Conference on Computer Vision Worshop on Spatial Coherence for Visual Motion Analysis*, 2004a.

[58] I. Laptev and T. Lindeberg. Velocity adaptation of space-time interest points. In *International Conference on Pattern Recognition (ICPR)*, 2004b.

[59] I. Laptev and P. Pérez. Retrieving actions in movies. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[60] I. Laptev, M. Marszalek, C. Schmid and B. Rozenfeld. Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[61] S. Lazebnik, C. Schmid and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 10.1109/CVPR.2006.68.

[62] Q. V. Le, W. Y. Zou, S. Y. Yeung and A. Y. Ng. Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[63] Z. Lin, Z. Jiang and L. S. Davis. Recognizing actions by shape-motion prototype trees. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[64] H. Liu, R. Feris, V. Krueger and M.-T. Sun. Unsupervised action classification using space-time link analysis. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010.

[65] J. Liu and M. Shah. Learning human actions via information maximization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[66] J. Liu, B. Kuipers and S. Savarese. Recognizing human actions by attributes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011a.

[67] J. Liu, J. Luo and M. Shah. Recognizing realistic actions from videos "in the wild". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[68] L. Liu, L. Wang and X. Liu. In defense of soft-assignment coding. In *IEEE International Conference on Computer Vision (ICCV)*, 2011b.

[69] M. Lourenço and J. P. Barreto. Tracking feature points in uncalibrated images with radial distortion. In *European Conference on Computer Vision (ECCV)*, 2012.

[70] O. Mac Aodha, G. J. Brostow and M. Pollefeys. Segmenting video into classes of algorithm-suitability. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[71] O. Mac Aodha, A. Humayun, M. Pollefeys and G. J. Brostow. Learning a confidence measure for optical flow, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **to appear**, 2012.

[72] O. Maron and T. Lozano-Prez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS)*, 1998.

[73] M. Marszalek, I. Laptev and C. Schmid. Actions in context. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[74] P. Matikainen, M. Hebert and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *IEEE International Conference on Computer Vision Workshop on Video-Oriented Object and Event Classification*, 2009.

[75] P. Matikainen, R. Sukthankar and M. Hebert. Model recommendation for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[76] R. Messing, C. Pal and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[77] K. Mikolajczyk and H. Uemura. Action recognition with motion-appearance vocabulary forest. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[78] T. B. Moeslund, A. Hilton and V. Krüger. A survey of advances in vision-based human motion capture and analysis, *Computer Vision and Image Understanding* **104**(2), 90–126, 2006.

[79] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas and A. Criminisi. Entangled decision forests and their application for semantic segmentation of CT images. In *Proceedings of the 22nd International Conference on Information Processing in Medical Imaging (IPMI)*, 2011.

[80] F. Moosmann, E. Nowak and F. Jurie. Randomized clustering forests for image classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **30**(9), 1632–1646, 2008.

[81] Y. Mu, J. Sun, T. X. Han, L.-F. Cheong and S. Yan. Randomized locality sensitive vocabularies for bag-of-features model. In *European Conference on Computer Vision (ECCV)*, 2010.

[82] J. C. Niebles, C.-W. Chen and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *European Conference on Computer Vision (ECCV)*, 2010.

[83] J. C. Niebles and L. Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[84] J. C. Niebles, H. Wang and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words, *International Journal of Computer Vision* **79**(3), 299–318, 2008.

[85] B. North, A. Blake, M. Isard and J. Rittscher. Learning and classification of complex dynamics, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 2000.

[86] S. Nowozin, G. Bakir and K. Tsuda. Discriminative subsequence mining for action classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[87] A. Oikonomopoulos, I. Patras and M. Pantic. Spatiotemporal salient points for visual recognition of human actions, *IEEE Transactions on Systems, Man and CyberneticPart B: Cybernetics* **36**(3), 710–719, 2006.

[88] O. Oshin, A. Gilbert, J. Illingworth and R. Bowden. Action recognition using randomised ferns. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[89] P. Pérez, C. Hue, J. Vermaak and M. Gangnet. Color-based probabilistic tracking. In *European Conference on Computer Vision (ECCV)*, 2002.

[90] R. Poppe. A survey on vision-based human action recognition, *Image and Vision Computing* **28**(6), 976–990, 2010.

[91] V. A. Prisacariu and I. Reid. Nonlinear shape manifolds as shape priors in level set segmentation and tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[92] M. Raptis and S. Soatto. Tracklet descriptors for action modeling and video analysis. In *European Conference on Computer Vision (ECCV)*, 2010.

[93] M. Raptis, I. Kokkinos and S. Soatto. Discovering discriminative action parts from mid-level video representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[94] K. Reddy, J. Liu and M. Shah. Incremental action recognition using feature tree. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[95] M. Rodriguez, S. Ali and T. Kanade. Tracking in unstructured crowded scenes. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[96] M. D. Rodriguez, J. Ahmed and M. Shah. Action MACH. a spatio-temporal Maximum Average Correlation Height filter for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[97] M. Rodriguez, J. Sivic, I. Laptev and J.-Y. Audibert. Data-driven crowd analysis in videos. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[98] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[99] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision (ECCV)*, 2006.

[100] M. S. Ryoo and J. K. Aggarwal. Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[101] M. Sapienza, F. Cuzzolin and P. Torr. Learning discriminative space-time actions from weakly labelled videos. In *British Machine Vision Conference (BMVC)*, 2012.

[102] S. Satkin and M. Hebert. Modeling the temporal extent of actions. In *European Conference on Computer Vision (ECCV)*, 2010.

[103] S. Savarese, A. DelPozo, J. C. Niebles and L. Fei-Fei. Spatial-temporal correlatons for unsupervised action classification. In *IEEE Workshop on Motion and Video Computing (WMVC)*, 2008.

[104] R. E. Schapire, 2002, The boosting approach to machine learning: An overview.

[105] K. Schindler and L. van Gool. Action snippets: How many frames does human action recognition require? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[106] C. Schüldt, I. Laptev and B. Caputo. Recognizing human actions: A local SVM approach. In *International Conference on Pattern Recognition (ICPR)*, 2004.

[107] S. D. Scott, J. Zhang and J. Brown. On generalized multiple-instance learning, *International Journal of Computational Intelligence and Applications* **5**(1), 21–35, 2005.

[108] P. Scovanner, S. Ali and M. Shah. A 3-dimensional SIFT descriptor and its application to action recognition. In *International conference on Multimedia (MM)*, 2007.

[109] H. J. Seo and P. Milanfar. Detection of human actions from a single example. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[110] E. Shechtman and M. Irani. Space-time behavior based correlation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[111] C. Sminchisescu, A. Kanaujia and D. Metaxas. Conditional models for contextual human motion recognition, *Computer Vision and Image Understanding* **104**(2-3), 210–220, 2006. Special Issue on Modeling People: Shape, Appearance, Movement and Behavior.

[112] R. P. Stanley, 2011, *Enumerative combinatorics. Vol. 2*, Cambridge Studies in Advanced mathematics 62, Cambridge University Press.

[113] B. Stenger, T. Woodley and R. Cipolla. Learning to track with multiple observers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[114] Y. Su, M. Allan and F. Jurie. Improving object classification using semantic attributes. In *British Machine Vision Conference (BMVC)*, 2010.

[115] Y. Su and F. Jurie. Visual word disambiguation by semantic contexts. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[116] Y. Su and F. Jurie. Learning compact visual attributes for large-scale image classification. In *European Conference on Computer Vision Workshop on Parts and Attributes*, 2012.

[117] J. Sun, X. Wu, S. Yan, L. Cheong, T. Chua and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[118] G. W. Taylor, R. Fergus, Y. LeCun and C. Bregler. Convolutional learning of spatio-temporal features. In *European Conference on Computer Vision (ECCV)*, 2010.

[119] Y. W. Teh, M. I. Jordan, M. J. Beal and D. M. Blei. Hierarchical Dirichlet processes, *Journal of the American Statistical Association* **101**(476), 1566–1581, 2006.

[120] H. Uemura, S. Ishikawa and K. Mikolajczyk. Feature tracking and motion compensation for action recognition. In *British Machine Vision Conference (BMVC)*, 2008.

[121] M. M. Ullah, S. N. Parizi and I. Laptev. Improving bag-of-feaures action recognition with non-local cues. In *British Machine Vision Conference (BMVC)*, 2010.

[122] L. Čehovin, M. Kristan and A. Leonardis. An adaptive coupled-layer visual model for robust visual tracking. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[123] C. Vondrick and D. Ramanan. Video annotation and tracking with active learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[124] H. Wang, A. Kläser, C. Schmid and C.-L. Liu. Action recognition by dense trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[125] H. Wang, M. M. Ullah, A. Kläser, I. Laptev and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *British Machine Vision Conference (BMVC)*, 2009a.

[126] L. Wang and D. Suter. Recognizing human activities from silhouettes: motion subspace and factorial discriminative graphical model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[127] P. Wang, G. D. Abowd and J. M. Rehg. Quasi-periodic event analysis for social game retrieval. In *IEEE International Conference on Computer Vision (ICCV)*, 2009b.

[128] S. B. Wang, A. Quattoni, L.-P. Morency, D. Demirdjian and T. Darrell. Hidden conditional random fields for gesture recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[129] X. Wang, X. Ma and E. Grimson. Unsupervised activity perception by hierarchical Bayesian models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[130] Y. Wang and G. Mori. Max-margin hidden conditional random fields for human action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2009.

[131] D. Weinland, E. Boyer and R. Ronfard. Action recognition from arbitrary views using 3D exemplars. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[132] D. Weinland and E. Boyer. Action recognition using exemplar-based embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[133] D. Weinland, R. Ronfard and E. Boyer. Automatic discovery of action taxonomies from multiple views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[134] G. Willems, T. Tuytelaars and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *European Conference on Computer Vision (ECCV)*, 2008.

[135] O. Williams, A. Blake and R. Cipolla. Sparse Bayesian learning for efficient visual tracking, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(8), 2005.

[136] S.-F. Wong and R. Cipolla. Extracting spatiotemporal interest points using global information. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[137] T. Woodley, B. Stenger and R. Cipolla. Tracking using online feature selection and a local generative model. In *British Machine Vision Conference (BMVC)*, 2007.

[138] X. Wu, W. Liang and Y. Jia. Incremental discriminant-analysis of canonical correlations for action recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[139] J. Xiao, J. Hays, K. Ehinger, A. Oliva and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[140] S. Yan, X. Xu, D. Xu and S. Lin. Beyond spatial pyramids: A new feature extraction framework with dense spatial sampling for image classification. In *European Conference on Computer Vision (ECCV)*, 2012.

[141] L. Yang, R. Jin, R. Sukthankar and F. Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[142] B. Yao and L. Fei-Fei. Grouplet: A structured image representation for recognizing human and object interactions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[143] B. Yao and S. Zhu. Learning deformable action templates from cluttered videos. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[144] R. Yao, Q. Shi, C. Shen, Y. Zhang and A. van den Hengel. Robust tracking with weighted online structured learning. In *European Conference on Computer Vision (ECCV)*, 2012.

[145] L. Yeffet and L. Wolf. Local trinary patterns for human action recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[146] A. Yilmaz and M. Shah. Actions sketch: a novel action representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[147] J. H. Yoon, D. Y. Kim and K.-J. Yoon. Visual tracking via adaptive tracker selection with multiple features. In *European Conference on Computer Vision (ECCV)*, 2012.

[148] G. Yu, J. Yuan and Z. Liu. Unsupervised random forest indexing for fast action search. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[149] T.-H. Yu, T.-K. Kim and R. Cipolla. Real-time action recognition by spatiotemporal semantic and structural forests. In *British Machine Vision Conference (BMVC)*, 2010.

[150] L. Zelnik-Manor and M. Irani. Event-based video analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[151] Q. Zhang and S. Goldman. EM-DD: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[152] Z. Zhang, K. Huang, T. Tan and L. Wang. Trajectory series analysis based event rule induction for visual surveillance. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

RÉSUMÉ

Cette thèse porte sur la classification de vidéos --étape importante de la compréhension des vidéos-- en se focalisant sur la reconnaissance d'actions. Nous nous plaçons dans le cas où des modèles de catégories sémantiques sont à construire automatiquement à partir de données d'entrainement: des extraits vidéo associées à des catégories. Aucune information n'est fournie quant à la localisation spatio-temporelle de l'action dans la vidéo, ni aux éléments indicatifs de la catégorie.

Nous explorons trois façons d'exploiter ces annotations faibles dans le cadre des représentations de vidéos dites «sacs à mots»: (1) une supervision cohérente dès les premières étapes du pipeline, (2) la combinaison d'attributs hétérogènes en nature et en échelle, et (3) des représentations intermédiaires basées sur des régions de sorte à identifier des zones pertinentes dans les vidéos. Pour la quantification de descripteurs locaux, nous proposons une nouvelle fonction objectif d'entrainement de forêts aléatoires, qui vise explicitement à accroitre la capacité discriminatoire des sacs de mots obtenus. Nos forêts sont plus robustes dans l'incorporation d'éléments de contexte pendant la quantification, limitant les risques de la combinaison naïve d'attributs. Nous montrons que les représentations intermédiaires apportent des informations complémentaires améliorant la performance des sacs de mots.

De plus, nous proposons une nouvelle application de la classification de vidéos dans le contexte du pistage. Nous montrons que des annotations faibles peuvent être utilisées pour classer des vidéos en types de modèle dynamique. Cette sélection de modèle par classification améliore la qualité du pistage.

Leveraging weak supervision for video understanding

This research deals with the task of video classification, with a particular focus on action recognition, which is essential for a comprehensive understanding of videos. In the typical scenario, there is a list of semantic categories to be modeled, and example clips are given together with their associated category label, indicating which action of interests happens in that clip. No information is given about where or when the action happens, or why the annotator considered the clip to belong to a sometimes ambiguous category.

Within the framework of the bag-of-words representation of videos, we explore how to leverage such weak labels from three points of view: (1) the use of coherent supervision from the earliest stages of the pipeline; (2) the combination of heterogeneous features in nature and scale; and (3) mid-level representations of videos based on regions, so as to increase the ability to discriminate relevant locations in the video. For the quantization of local features, we propose a novel form of supervision to train random forests which explicitly aims at the discriminative power of the resulting bags of words. We show that our forests are better than traditional ones at incorporating contextual elements during quantization, and draw attention to the risk of naive combination of features. We also show that midlevel representations carry complementary information that can improve classification.

Moreover, we propose a novel application of video classification to tracking. We show that weak clip labels can be used to classify videos into categories of dynamic models. In this way, we improve tracking by performing classification-based dynamic model selection.

MOTS-CLES INDEXATION RAMEAU

Vision par ordinateur, Apprentissage automatique, Classification automatique, Analyse de scènes (informatique), Arbres de décision

DISCIPLINE

Informatique et applications

U.F.R. Sciences
Campus 2 - Côte de Nacre
Boulevard du Maréchal Juin
14032 Caen CEDEX