Removed figs 1.1 and 1.2, pp.2-3

Recognising and localising human actions

Michael Sapienza

Department of Computing and Communications Technology
Faculty of Technology, Design and Environment
Oxford Brookes University

A dissertation submitted to the Faculty of Technology in partial fulfilment
of the requirements of the award of Doctor of Philosophy.

September 2014

Abstract

Human action recognition in challenging video data is becoming an increasingly important research area. Given the growing number of cameras and robots pointing their lenses at humans, the need for automatic recognition of human actions arises, promising Google-style video search and automatic video summarisation/description. Furthermore, for any autonomous robotic system to interact with humans, it must first be able to understand and quickly react to human actions.

Although the best action classification methods aggregate features from the entire video clip in which the action unfolds, this global representation may include irrelevant scene context and movements which are shared amongst multiple action classes. For example, a waving action may be performed whilst walking, however if the walking movement appears in distinct action classes, then it should not be included in training a waving movement classifier. For this reason, we propose an action classification framework in which more discriminative action *subvolumes* are learned in a weakly supervised setting, owing to the difficulty of manually labelling massive video datasets. **The learned models are used to simultaneously *classify* video clips and to *localise* actions to a given space-time subvolume.** Each subvolume is cast as a bag-of-features (BoF) instance in a multiple-instance-learning framework, which in turn is used to learn its class membership. We demonstrate quantitatively that even with single fixed-sized subvolumes, the classification performance of our proposed algorithm is superior to our BoF baseline on the majority of performance measures, and shows promise for space-time action localisation on the most challenging video datasets.

Exploiting spatio-temporal structure in the video should also improve results, just as deformable part models have proven highly successful in object recognition. However, whereas objects have clear boundaries which means we can easily define a ground truth for initialisation, 3D space-time actions are inherently ambiguous and expensive to annotate in large datasets. Thus, it is desirable to adapt pictorial star models to action datasets without location annotation, and to features invariant to changes in pose such as bag-of-feature and Fisher vectors, rather than low-level HoG. Thus, **we propose local deformable spatial bag-of-features (LDSBoF) in which local discriminative regions are split into a fixed grid of parts that are allowed to deform in both space and time at test-time.** In our experimental evaluation we demonstrate that

by using local, deformable space-time action parts, we are able to achieve very competitive classification performance, whilst being able to localise actions even in the most challenging video datasets.

A recent trend in action recognition is towards larger and more challenging datasets, an increasing number of action classes and larger visual vocabularies. For the global classification of human action video clips, the bag-of-visual-words pipeline is currently the best performing. However, the strategies chosen to sample features and construct a visual vocabulary are critical to performance, in fact often dominating performance. Thus, **we provide a critical evaluation of various approaches to building a vocabulary and show that good practises do have a significant impact.** By subsampling and partitioning features strategically, we are able to achieve state-of-the-art results on 5 major action recognition datasets using relatively small visual vocabularies.

Another promising approach to recognise human actions first encodes the action sequence via a generative dynamical model. However, using classical distances for their classification does not necessarily deliver good results. Therefore **we propose a general framework for learning distance functions between dynamical models, given a training set of labelled videos.** The optimal distance function is selected among a family of 'pullback' ones, induced by a parametrised mapping of the space of models. We focus here on hidden Markov models and their model space, and show how pullback distance learning greatly improves action recognition performances with respect to base distances.

Finally, the action classification systems that use a single global representation for each video clip are tailored for offline batch classification benchmarks. For human-robot interaction however, current systems fall short, either because they can only detect one human action per video frame, or because they assume the video is available ahead of time. In this work **we propose an online human action detection system that can incrementally detect multiple concurrent space-time actions.** In this way, it becomes possible to learn new action classes on-the-fly, allowing multiple people to actively teach and interact with a robot.

Acknowledgements

First and foremost I would like to thank my supervisors Dr. Fabio Cuzzolin and Prof. Philip H.S. Torr, for providing the opportunity for me to pursue a doctorate in Computer Vision - how cool is that?! I also thank my supervisors for their time, advice, for pushing me out of my comfort zone and criticising my work; under their supervision I have learned enormously. During my studies I also discovered 'learning through osmosis' - the natural flow of information across a concentration gradient!

I was fortunate enough to share the lab, and pub, with several friends who helped me along the way - Paul Sturgess, Sunando Sengupta, Vibhav Vineet, Lubor Ladicky, Shuai Zheng, Marco Cecotti, Cristian Roman, and Bassel Yousef. Sharon Howard provided tip-top administration, whilst Khaled Hayatleh and Cigdem Sengul kindly gave their time as postgraduate tutors. Nigel Crook kept a friendly and enthusiastic eye over the department.

I am grateful to my family in Malta, and my sister in London, for their generosity; and to Sophie for her unlimited patience.

Contents

Abstract **iii**

Acknowledgements **v**

List of Figures **xi**

List of Tables **xv**

Nomenclature **xvii**

1 Introduction **1**

1.1 Motivating applications of human action recognition from videos . 1

1.2 What is an action? 2

1.3 Why is human action recognition hard? 3

1.4 Dissertation outline 5

1.5 Avenues of investigation 5

1.5.1 Learning with sets of local histograms 6

1.5.2 Adding deformable action structure, general subvolume shapes and saliency maps 7

1.5.3 Evaluating global bag-of-feature pipeline variations 8

1.5.4 Encoding action dynamics & learning distances between models 10

1.5.5 Real-time and online multiple action detection 12

1.6 Contributions 13

1.7 Resulting publications and impact 14

2 Related work **17**

2.1 Interest point detection 17

2.2 Descriptors 17

2.3 Mid-level representations 18

2.4 Learning the discriminative parts of an action video 19
2.5 Discriminative video part extensions 21
2.5.1 Incorporating mid-level pictorial structures 21
2.5.2 General subvolume shapes and action saliency 23
2.6 Design choices in the bag-of-visual-words pipeline for video data . 24
2.7 Pullback distances for time-series classification 26
2.8 Online multiple action detection 29
2.8.1 Multiple action detection in space and time 29
2.8.2 Online video stream-processing 30
2.8.3 Proposed online multiple action detection system 31

3 Datasets and performance indicators 35
3.1 Action classification 35
3.1.1 KTH 35
3.1.2 YouTube 36
3.1.3 Hollywood2 37
3.1.4 HMDB51 37
3.1.5 UCF101 38
3.1.6 Performance indicators 38
3.2 Action detection 42
3.2.1 Performance indicators 43

4 Learning discriminative actions from weakly labelled videos 49
4.1 MIL-BoF action models 49
4.2 A learnt mapping from instance to bag labels 52
4.3 Experimental Evaluation 54
4.3.1 Baseline BoF algorithm 54
4.3.2 MIL-BoF experimental setup 55
4.4 Results and discussion 55

5 Towards adding local structure and general subvolume shapes 59
5.1 Local Deformable SBoF models (LDSBoF) 59
5.2 Experimental setup 62
5.2.1 Experiments 65
5.3 Quantitative results and discussion 66
5.3.1 Experiment 1 - adding structure 66
5.3.2 Experiment 2 - general subvolume shapes 66
5.4 Computational timings 70

5.5 Qualitative localisation discussion . . . 71
5.5.1 Bounding box detection with LDSBoF . . . 71
5.5.2 Class-specific saliency . . . 71

6 Feature sampling and partitioning for visual vocabulary generation on large action classification datasets 79
6.1 Constant experimental settings . . . 80
6.2 Variable components of the experiments . . . 82
6.3 Results and discussion . . . 82

7 Learning pullback distances for dynamical action models 91
7.1 Pullback metrics in Riemannian geometry . . . 91
7.2 Pullback distance learning framework . . . 92
7.3 Pullback distances for HMMs . . . 93
7.3.1 The space $\mathcal{H}$ of hidden Markov models . . . 93
7.3.2 The space of transition matrices . . . 94
7.3.3 Learning an approximate observation space . . . 94
7.3.4 An automorphism of $\mathcal{H}$ . . . 95
7.3.5 Sampling the parameter space of the automorphism . . . 97
7.4 A proof of concept . . . 98
7.4.1 Synthetic HMM sequences . . . 99
7.4.2 Experimental setup . . . 100
7.4.3 Preliminary results & discussion . . . 100
7.5 Experiments on human action recognition . . . 102
7.5.1 Implementation details . . . 102
7.5.2 Results and discussion . . . 104
7.5.3 Other nonlinear automorphisms . . . 106
7.5.4 Beyond HMMs . . . 108
7.5.5 Close-form optimisation by volume element matching . . . 108

8 Online learning of multiple concurrent space-time actions 111
8.1 Fast region proposal generation . . . 111
8.2 Selective tubes . . . 113
8.3 Feature extraction . . . 114
8.4 Action category learning . . . 115
8.4.1 Class specific feature cache . . . 115
8.4.2 Growing the learner set . . . 115

8.4.3 Solving an SVM with batch stochastic gradient descent and hard example mining . . . 116
8.4.4 Life-long learning . . . 117
8.5 Preliminary tests . . . 118
8.5.1 Selective tubes . . . 118
8.5.2 Learning of multiple categories streamed online . . . 119

9 Conclusions and future directions **121**

Appendices **127**

A Supporting ideas **129**
A.1 Pullback metric . . . 130
A.2 Rectifying a non-linear boundary . . . 132
A.3 Improving classification: automorphisms on a simplex . . . 133
A.4 Bag of visual words pipeline . . . 136
A.5 Visualising visual words for inspiration . . . 139
A.6 Pictorial structure matching . . . 142
A.7 Distance transforms . . . 144
A.8 Representation, representation, representation . . . 145
A.8.1 How to assess the quality of a representation? . . . 146

References **149**

List of Figures

1.1 Human action recognition applications: online and offline. 2
1.2 The large intra-class variation in human action recognition. 3
1.3 A sample of images from popular action classification datasets. . . 4
1.4 The disadvantages of using global information to represent action clips. 7
1.5 A sample result showing fixed-size cubic action subvolumes learned in a max-margin multiple instance learning framework. 8
1.6 A video plotted in space and time showing root and part filter detections from a handwaving action pictorial structure model. . . 9
1.7 A sample result showing a dense handwaving-action location map, where each pixel is associated with a score indicating its class-specific saliency 10

2.1 Illustration of the dense trajectory feature extraction process. . . 18

3.1 Action recognition state-of-the-art performance over time. 36
3.2 Image samples from the KTH dataset. 37
3.3 Image samples from the YouTube dataset. 38
3.4 Image samples from the Hollywood2 dataset. 39
3.5 Example precision-recall curve. 41
3.6 A plot of the classification performance against the number of classes. 43
3.7 A graph showing the classification performance against the imbalance in examples per class. 44
3.8 Image samples from the HMDB51 dataset. 46
3.9 Image samples from the UCF101 dataset. 47
3.10 A sample image from the LIRIS human activities dataset used to benchmark action detection systems. 48

4.1 An action is defined as a collection of space-time action parts contained in subvolumes shapes of cube/cuboidial shape. 50
4.2 Action localisation results on two challenging videos from the Hollywood2 dataset. 58

5.1 Action recognition with a local deformable spatial bag-of-features model (LDSBoF). 60
5.2 Local space-time subvolumes of different sizes are drawn in two videos of varying length at random locations. 64
5.3 Quantitative graphs for learning local discriminative subvolume models via multiple-instance learning. 68
5.4 Top, and side views from a test boxing video sequence in the KTH dataset. 72
5.5 Detected LDSBoF configurations in the challenging Hollywood2 dataset. 73
5.6 Action classification and localisation on the KTH dataset. 75
5.7 Action localisation results on the HMDB51 dataset. 76
5.8 Misclassifications in the HMDB51 dataset. 77

6.1 KTH dataset classification accuracy using a global video representation. 85
6.2 Comparing YouTube dataset classification results. 86
6.3 The classification accuracy versus K cluster centres and representation dimensionality on the Hollywood2 dataset. 87
6.4 The classification accuracy versus K cluster centres and representation dimensionality for the HMDB dataset. 88

7.1 Encoding videos as dynamical models. 92
7.2 The space of transition matrices. 95
7.3 Fitting a mixture of Gaussians (MoG) to the LLE embeddings of the state output matrix columns, and mapping observation vectors in the embedded space via a MoG. 96
7.4 Transition matrices $\boldsymbol{A}^\top$ based on fictional weather patterns from each country. 99
7.5 State-output probability matrices $\boldsymbol{C}^\top$ based on fictional snacking patterns for each country under different weather conditions. . . . 100
7.6 The % Accuracy plotted against the sampling density. 102
7.7 The % Accuracy plotted against the number of states in the HMM. 103

7.8 Performance plots comparing the Frobenius 'base' distance to the pullback-Frobenius distance. . . . 105
7.9 Confusion matrices for KTH dataset. . . . 106
7.10 Confusion matrices for the YouTube dataset. . . . 107
7.11 Comparing the performance measures achieved by different base and pullback distances. . . . 107
7.12 Automorphisms which preserve lines and others mapping lines to curves. . . . 108

8.1 Multiple action detection in a video stream using tubes. . . . 112
8.2 A sequence of images showing the propagation of region proposal labels in time via Hungarian assignment. . . . 118
8.3 A depiction of the SVM learning 10 categories (1-vs-rest) streamed online after 2000 iterations. . . . 120

A.1 Illustration of a pullback distance. . . . 131
A.2 Illustration of an automorphism rectifying a classification boundary. 132
A.3 Improving classification on a simplex by line-preserving automorphisms. . . . 134
A.4 Improving classification on a simplex by other nonlinear automorphisms. . . . 135
A.5 Training and testing examples for a toy example plotted in $\mathbb{R}^2$. . 136
A.6 Plotting the normalised histograms in $\mathbb{R}^3$. . . . 137
A.7 Precision recall curves per action class for three competing classifiers. . . . 138
A.8 Visual words from four action classes in the KTH dataset side-by-side. . . . 139
A.9 Visual words from a running class action video plotted in space and time. . . . 140
A.10 Visual words from a handwaving class action. . . . 140
A.11 Visual words from a walking class action. . . . 141
A.12 Visual words from a boxing class action. . . . 141
A.13 A 3-part pictorial structure model and its optimal configuration. . 142
A.14 Cost function measuring how well each part matches the image. . 143
A.15 Transformed and offset responses. . . . 143
A.16 An illustration of a distance transformed image. . . . 144
A.17 An example illustrating a description of an image, its histogram representation, and one of its nearest neighbours. . . . 146

List of Tables

2.1 Statistics on the amount of dense trajectory features extracted per video for various datasets. 26

3.1 A typical confusion matrix for an action classification task with $K = 3$ classes. 40

4.1 Quantitative results from the state-of-the-art (S-o-t-a), our BoF baseline, and our MIL-BoF method for various fixed-size subvolumes. 56

5.1 A table showing the results for using our local deformable spatial bag-of-features (LDSBoF) with Fisher vectors generated using $K = 32$ Gaussians. 67

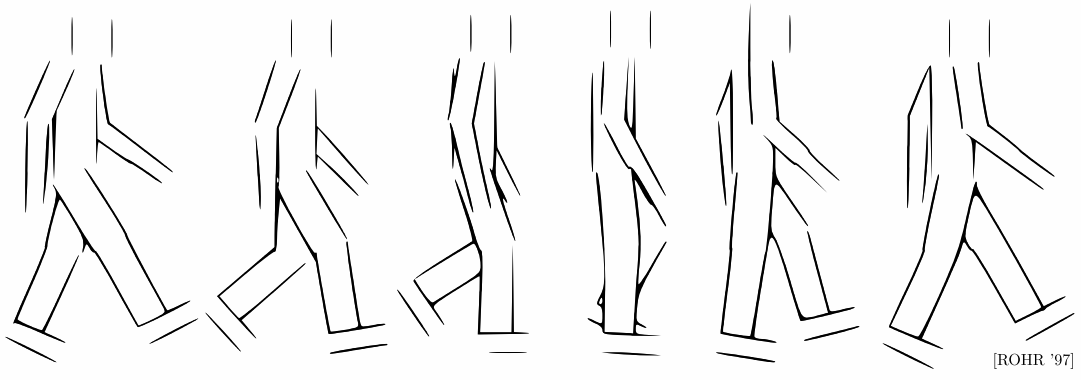
5.2 A table showing the state-of-the-art results, and our results using Fisher vectors with $K = 32$ Gaussians (F32). 70

6.1 The current state-of-the-art results compared to our best results obtained with the global bag-of-features pipeline. 89

8.1 Quantitative results showing the performance of various online linear SVM variations. 119

"and the life which is unexamined is not worth living"

[Plato's Apology, (38a)]



Chapter 1

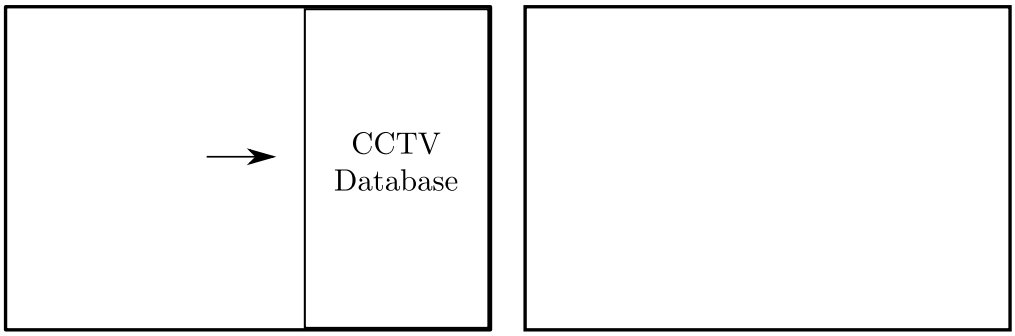
Introduction

If you've sat down on a comfortable chair and picked up this dissertation, you've already performed a couple of actions, as indicated by the verbs in the opening phrase. If a computer was to detect actions from sentences, then detecting verbs would be a good place to start. What if, however, the computer only has a sequence of images to form a video? What would it look for? Why is this even a good idea?

1.1 Motivating applications of human action recognition from videos

Human action recognition from video is becoming an increasingly prominent research area in computer vision, with far-reaching applications. On the Internet, the recognition of human actions will allow the organisation, search, description, and retrieval of information from the massive amounts of video data uploaded each day [1–3]. In every day life, human action recognition may provide a natural way to communicate with robots, and provide novel ways of interacting with computer games and virtual environments [4–6]. These applications fall naturally into two groups. On the one hand, those which afford offline processing, and on the other, those for which online and real-time processing is critical.

Consider a security application where one needs to search through or summarise hundreds of hours of recorded CCTV footage automatically in the aftermath of a theft, as illustrated in Fig.1.1(a). Here, the video has been recorded some time in the past, and the entire video, from beginning to end, is available. The speed at which the result of the search is reached, and the ordering with which possible matches are checked, are not critical to its operation. Therefore



(a) Offline processing (b) Online processing

Figure 1.1: (a) Search and retrieval applications in massive video collections afford offline batch processing, since videos have been recorded in the past. Moreover, speed is not critical to the search performance. (b) Human robot interaction applications, in contrast, require instantaneous feedback to the interacting agents. A robot only has access to present and past video frames, and computation needs to be real-time for natural interaction.

this kind of search affords offline batch processing. In contrast, for robots in human environments such as ASIMO [7] and Baxter [8], both the robot and the person interacting with it need instantaneous feedback, as seen in Fig.1.1(b). The robot must process one frame at a time and update the knowledge of its environment immediately after each frame. As opposed to offline batch processing, only the present and past frames are available, and the frame processing needs to be quick for natural interaction with humans to take place. The greater part of this dissertation is devoted to applications affording offline batch processing, whilst the last chapter changes direction towards challenging online scenarios. Before moving on, we must first address a most important question: What is an action?

1.2 What is an action?

This question may be answered from two perspectives, depending on whether you ask a computer or a human. From a computer's perspective, an action needs to have a specific definition in order for it to be explicitly programmed. However, due to the difficulty of defining an action down to small details, and the difficulty of programming a well thought out definition, researches are usually content with their own definition, tailored for their application. In this way, an action is simply what you defined it to be.

For the purpose of our *human* understanding, previous researchers have de-

Figure 1.2: One reason why human action recognition is such a hard problem is that within the same class of actions, for example 'jumping', there is a huge variability over the person's appearance, clothing, and motion, which an action model must generalise over. This can be seen in the above images, where three different events in the London 2012 Summer Olympics all involve a quite different jumping action.

fined an action to be the "intentional bodily movement of biological agents interacting with their environment for a specific purpose" [1, 9], though they quickly disregard any notion of movement, intention, and agent in what followed. In the gap between what computers and humans understand by an action, lie many open and challenging problems.

1.3 Why is human action recognition hard?

Although actions may be trivial for humans to understand, achieving this capability on machines poses considerable challenges. In addition to the classical difficulties in computer vision [10] of dealing with objects in the world with variations in illumination, viewpoint, background and part occlusions, human actions inherently possess a high degree of geometric and topological variability[1]. For instance, a jumping motion may vary in height, frequency and style, yet this is still the same action, as shown in Fig.1.2. Additional ambiguity arises when trying to define the start and end of an action. Querying unconstrained video data introduces other nuisance factors since the recording is often of low quality, and affected by camera motion, zooming and image blur from shaking. It is therefore critical for an action recognition system to generalise over a wide group of actions in the same class, and yet to discriminate between actions in different classes [11].

Despite these difficulties, significant progress has been made in learning and recognising human actions from videos [11, 12]. Whereas early action recognition

[1]The ability of humans to appear as figures which may deform into topologically non-equivalent surfaces, such as a sphere and a torus.

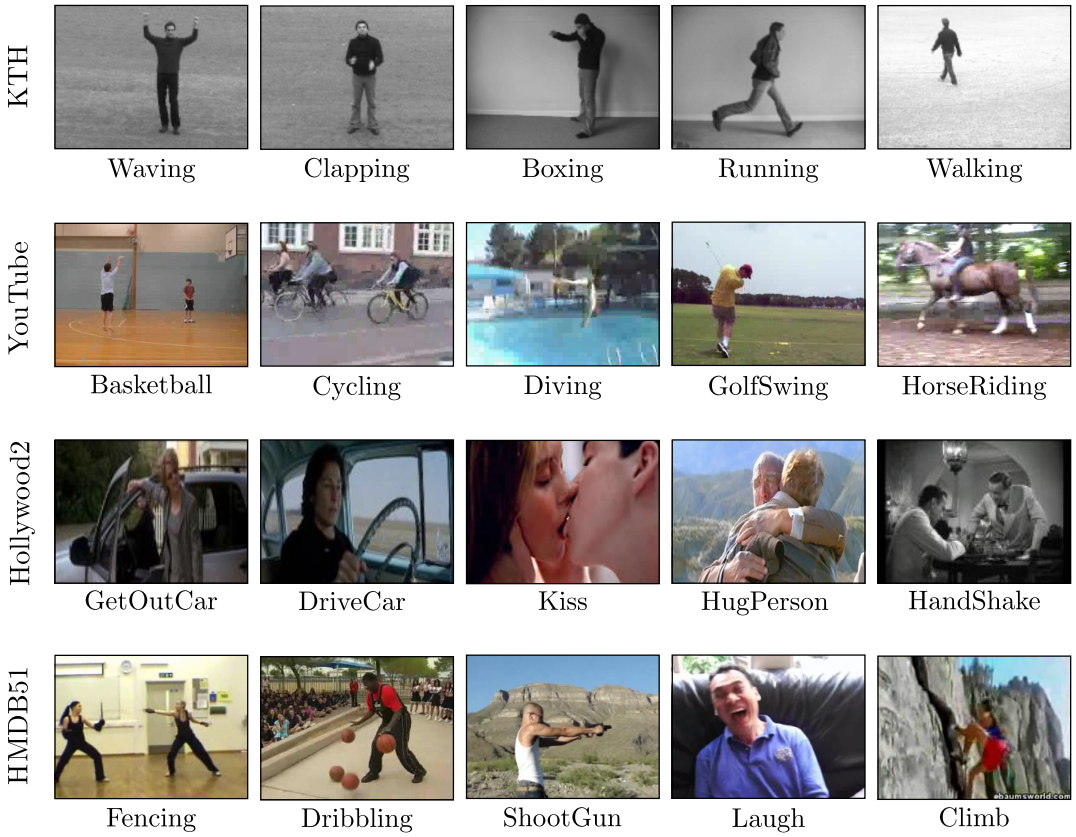


Figure 1.3: A sample of images from popular action classification datasets. Whereas early action recognition datasets like the KTH [13] included videos with single, staged human actions against homogeneous backgrounds, more recently challenging uncontrolled movie data from the Hollywood2 dataset [17] and amateur video clips available on the Internet seen in the YouTube [16] and HMDB51 [3] datasets are being used to evaluate action recognition algorithms. These challenging datasets contain human actions which exhibit significant variations in appearance, style, viewpoint, background clutter and camera motion, as seen in the real world.

datasets included videos with single, staged human actions against simple, homogeneous backgrounds [13, 14], more recently, challenging uncontrolled movie data [15] and amateur video clips available on the Internet [3, 16] have been used to evaluate action recognition algorithms. These datasets contain human actions with large variations in appearance, style, viewpoint, background clutter and camera motion, common in the real world. A sample of images from four popular action classification datasets is shown in Fig. 1.3, in order of increasing difficulty, from KTH [13] to the more challenging HMDB51 [3] dataset. In the next section, we present the outline of this dissertation, together with the action recognition tasks explored; these vary in difficulty, but can all be explicitly programmed for a computer to solve automatically.

1.4 Dissertation outline

First and foremost, we present the human action recognition datasets used throughout this work in Chapter 3. Next, we formulate the problem of local, weakly-supervised action clip classification in Chapter 4, where in addition to predicting the global action clip label, the computer must also predict the action location in space and time, without having location annotation during training. An extension of this work to include pictorial structure models and more general subvolume shapes is presented in Chapter 5.

In chapters 6 & 7 we consider the problem of global, supervised, action clip classification, where the task is to assign an action label to an unknown video clip, given a training dataset of videos with ground truth labels. Note that the video clips are of predefined length and it is assumed that one video belongs to a single action class. Whereas in chapter 6 videos are represented globally as a bag-of-visual-words, a global hidden Markov model is used in chapter 7.

Lastly, in Chapter 8 we address the problem of online multiple human action *detection*, which is the task of associating a local 3D-subvolume/tube that may be growing in time with an action category. Notice that both action clip classification with localisation, as well as multiple action detection pose a greater challenge to a computer than global action clip classification. This is because for a classification task, a video may take only one of C class labels, giving a random guess a $\frac{1}{C}$ chance of choosing the right label. In order to localise the action, one must also predict where, out of millions of possible locations, the action is happening, giving a randomly guessed location a very small chance of being correct.

1.5 Avenues of investigation

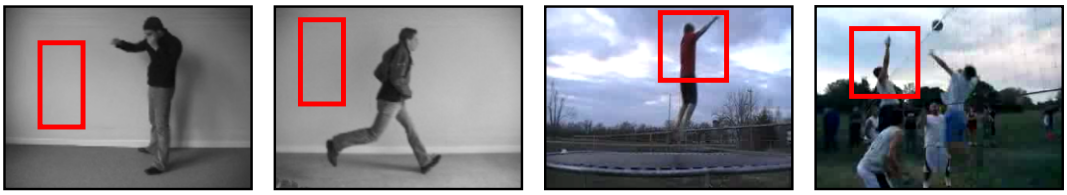
Effective data representations are key to the success of machine learning algorithms, since they make explicit the underlying factors of variation in the data which are important for discrimination [10, 18] (cf. Section A.8). Effective action representations are also key to the success of human action recognition algorithms. In the following subsections we introduce the action representations used in this work, as well as the original techniques developed to address the action recognition problems set out in Section 1.4.

1.5.1 Learning with sets of local histograms

One highly successful approach to represent an action video has been in the form of a single 'bag-of-visual-words', also known as 'bag-of-features' (BoF), histogram (cf. Section A.5), and its extensions to VLAD/Fisher vectors [19]. The surprising success of BoF may be attributed to its ability to summarise local features as a simple histogram, without regard for human detection, pose estimation or the location of body-parts, which so far cannot be extracted reliably in unconstrained action videos [20]. Moreover, the histograms provide a fixed length representation for a variable number of features per video, which is ideal for traditional learning algorithms [21].

A drawback arising from a single global representation per action video is that no local information may be derived to tell more specifically where the action of interest is taking place. To overcome this limitation, we explored the possibility of representing an action video with a set of local representations (cf. Chapters 4 & 5). Moreover, current space-time human action classification methods [22–25] derive an action's representation from an entire video clip, even though this representation may contain motion and scene patterns pertaining to multiple action classes. For instance, in the state-of-the-art BoF approach [26], dense space-time features are aggregated globally into a single histogram representation per video. This histogram is generated from features extracted from the whole video and so includes visual word counts originating from irrelevant scene background (see Fig. 1.4(a) & 1.4(b)), or from motion patterns shared amongst multiple action classes. For example the action classes 'trampoline jumping' and 'volleyball spiking' from the YouTube dataset [16] both involve jumping actions, and have a similar scene context, as shown in Fig. 1.4(c) & 1.4(d). Therefore in order to discriminate between them, it is desirable to automatically select those video parts which tell them apart, such as the presence of a moving ball, multiple actors and other action-specific characteristics.

This motivates a framework in which action models are derived from smaller portions of the video volume, subvolumes, which are used as learning primitives rather than the entire space-time video. Since large action classification datasets only have labels for the global video clip, and not the label of each individual video subvolume, we propose to cast action classification in a weakly labelled framework. In this way, action models may be derived from automatically selected video parts which are most discriminative of the action. An example illustrating the result of learning the discriminative fixed-size action cubes from which action models are derived is shown in Fig. 1.5. Even though features were



(a) boxing (b) running (c) trampoline jump (d) volleyball spiking

Figure 1.4: The disadvantages of using global information to represent action clips. Firstly, global histograms contain irrelevant background information as can be seen in the ***(a)*** *Boxing and* ***(b)*** *Running action videos of the KTH dataset [13]. Secondly, the histograms may contain frequency counts from similar motions occurring in different action classes, such as the* ***(c)*** *trampoline jumping and* ***(d)*** *volleyball spiking actions in the YouTube dataset [16]. In Chapter 4, we propose a framework in which action models can be derived from local video subvolumes which are more discriminative of the action. Thus important differences such as the moving ball, the presence of multiple people and other action-specific characteristics may be captured.*

extracted over the entire video (as marked by the black dots in Fig. 1.5), only regions where the action is taking place were selected during training, since the background is found in other action classes of the dataset, and therefore not discriminative of the action.

1.5.2 Adding deformable action structure, general subvolume shapes and saliency maps

In addition to discriminative local action models, we propose to incorporate deformable structure by learning a pictorial structure model for each action class. In the absence of ground truth location annotation, we use automatically selected video regions to learn a 'root' action model. Action part models are subsequently learnt from the root location after dividing it into a fixed grid of regions, which are allowed to deform at test-time. This extends spatial-BoF models [27] to incorporate deformable structure. The result of testing a 3-part handwaving model is shown in Fig. 1.6.

As another extension to Section 1.5.1 in which only fixed size cubes were used, we aggregate local histograms over subvolumes of varying cuboidial sizes. Moreover, in contrast to localising actions via bounding cuboids (see Fig. 1.5 & Fig. 1.6), we propose to build an action-specific saliency map by aggregating the predicted detection scores from all subvolume shapes. An example illustrating the result of using general subvolume shapes visualised as a saliency map is shown in Fig. 1.7.

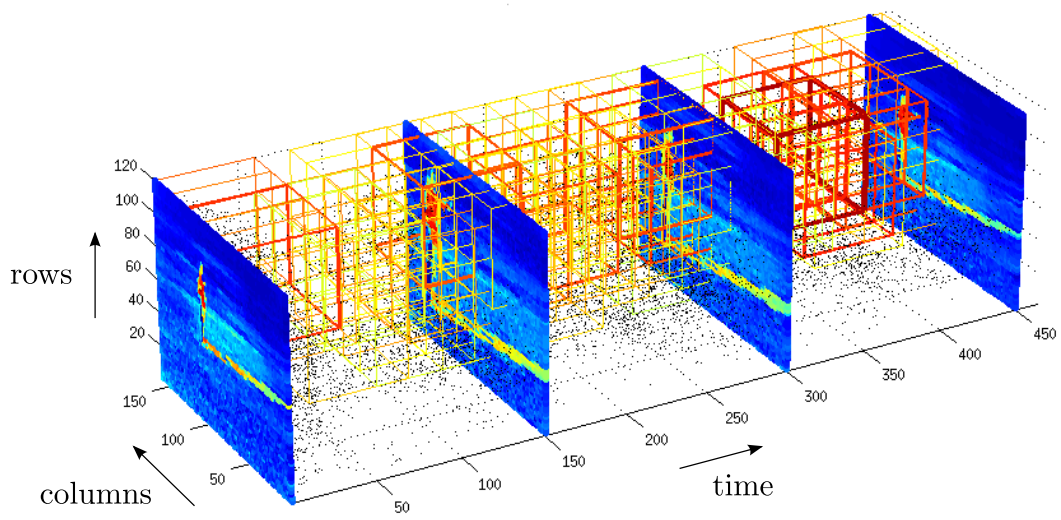


Figure 1.5: A boxing video sequence taken from the KTH dataset [13] plotted in space and time. Black dots denote the mean position from where features have been extracted. Notice that in this particular video, the camera zoom is varying with time, and features (black dots) were extracted from areas of motion caused by camera zoom. Overlaid on the video are discriminative cubic action subvolumes learned in a max-margin multiple instance learning framework (cf. Section 4.1), with colour indicating their class membership strength. Since the scene context of the KTH dataset is not discriminative of the particular action, only subvolumes around the actor were selected as positive instances (best viewed in colour).

By extending global mid-level representations (e.g. BoF, Fisher vectors) to deformable part-models and general video subvolumes, we aim to both improve classification results, as compared to the global baseline, and capture location information.

1.5.3 Evaluating global bag-of-feature pipeline variations

Since a local bag-of-visual-words representation was used in the previous two sections to describe video parts, it is important to ensure that the visual word vocabulary pipeline is tuned to give maximum performance. Despite extensive work showing various aspects in which the BoF pipeline may be improved to squeeze out additional performance [21, 28–33], there remains interesting questions yet to be evaluated empirically, particularly when considering huge *action* classification datasets with a large number of classes [3, 34]. For example, the recent UCF101 dataset [34] contains 101 action classes and $\sim$13,000 video clips. Using the state-of-the-art 'dense trajectory' features, this generates $\sim$679GB of features. Therefore one cannot easily load all the training features into memory. Furthermore, randomly subsampling the features generates a bias towards

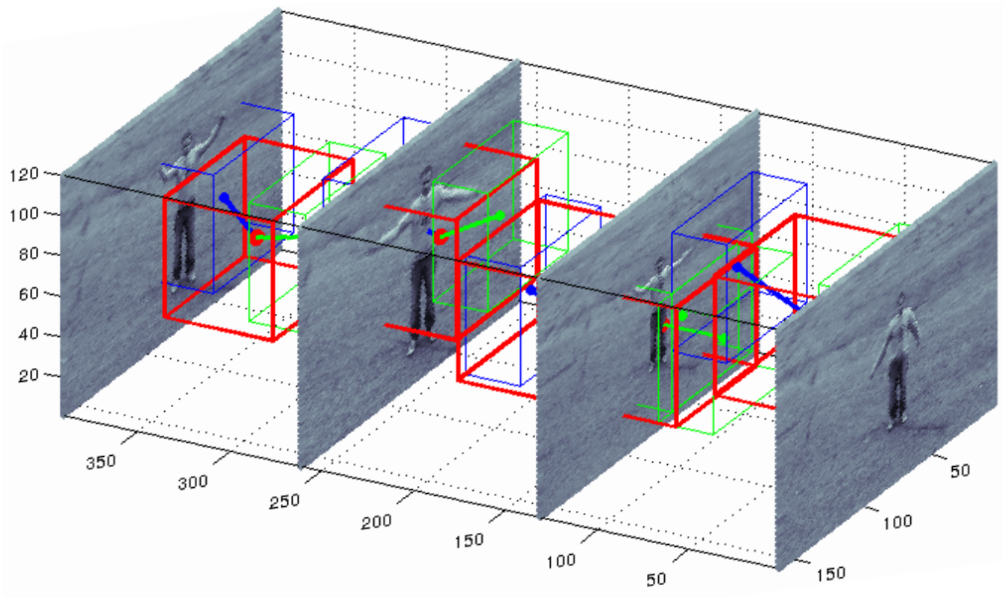


Figure 1.6: A handwaving video sequence taken from the KTH dataset [13] plotted in space-time. The action is localised in space and time by a pictorial structure model, despite the latter being trained in a weakly supervised framework (in which no action location annotation is available). Overlaid on the video are the root filter detections, drawn as red cubes, and part filters (shown in green and blue respectively), linked to the root by green and blue segments. The star model for the handwaving action (above) is detected at multiple steps in time, and thus well suited to detect actions of unknown duration (best viewed in colour).

action classes associated with a greater share of videos, or disproportionately longer video sequences[2].

Secondly, state-of-the-art space-time descriptors are typically formed by a number of components. The issue arises of whether it is best to learn a single visual vocabulary over the whole, joint feature space, or whether to learn a vocabulary for each feature component separately. Finally, visual vocabularies may also be learnt separately for each action category [21]. Although it generates redundant visual words which are shared among multiple action classes, such a strategy is worth exploring for the dimensionality of the representation increases linearly with the number of classes C. Thus learning multiple dictionaries per-category, each with a small number of clusters K, may still produce large vector representations (key to the success of Fisher vectors [2]) since the final dictionary size will be $K \times C$.

An exploration into various ways of generating visual vocabularies for action classification is laid out in Chapter 6, where the following unanswered questions

[2] Note that the issue of having a large discrepancy in the number of features per video sample is not usually encountered when using image datasets.

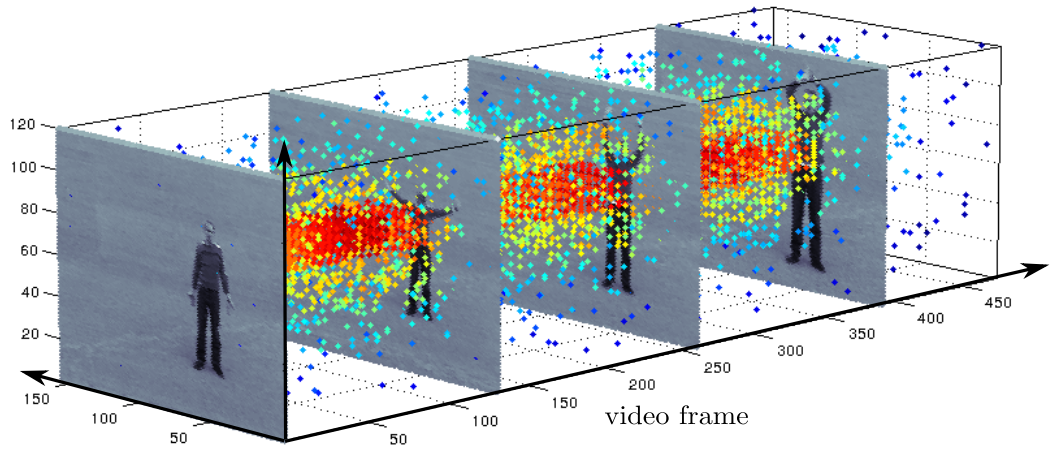


Figure 1.7: A handwaving video sequence taken from the KTH dataset [13] plotted in space and time. Notice that in this particular video, the handwaving motion is repeated continuously and the camera zoom is varying with time. Overlaid on the video is a dense handwaving-action location map, where each pixel is associated with a score indicating its class-specific saliency. This dense map was generated by aggregating detection scores from general subvolumes sizes, and is displayed sparsely for clarity, where the colour, from blue to red, and sparsity of the plotted points indicate the action class membership strength. Since the scene context of the KTH dataset is not discriminative for this particular action, only the movement in the upper body of the actor is detected as salient (best viewed in colour).

are addressed: i) What is the best way to randomly subsample features to build a vocabulary for human action recognition? ii) What are the effects of learning separate rather than joint visual vocabularies when considering multiple feature components or multiple action classes?

1.5.4 Encoding action dynamics & learning distances between models

Due to the sequential nature in which human actions unfold, another popular way of initially representing action clips has been in the form of a time-series of features extracted from video frames, or adjacent blocks of frames [35–37]. For encoding these time-varying observations into a single, global representation, generative dynamical models have been employed since they possess a number of desirable properties for action recognition. First, they are effective in coping with observation sequences of the same category which vary in time or speed, also known as time warping [37]. Secondly, they are able to describe the causal relationships in activity patterns [38]. Indeed, many researchers have explored the idea of encoding time-varying features via linear, nonlinear [39], and chaotic

[36] dynamical models. Hidden Markov models [40], in particular, have been widely employed [37]. They are typically classified by learning a new model for each test sequence, measuring its distance from the old models, and attributing to it the label of the closest model(s) [35].

A number of distance functions for comparing dynamical models (e.g. Cauchy kernels [39]) and HMMs (e.g. KL-divergence [41]) have been introduced. Nevertheless, no single distance function can possibly outperform all the others in every classification problem, as the same models can be endowed with different class labellings. Therefore, a sensible approach when training data are available consists in learning in a supervised fashion the 'best' distance function from the data, for example, by maximising the classification performance on a validation set. This approach is widely supported by the literature, particularly in linear spaces[3] [42]. However, generative dynamical models live in non-linear spaces[4], and thus the need for a principled way of learning distances in general metric spaces arises.

An interesting tool is provided by *pullback metrics* (cf. Section A.1). If the models belong to a Riemannian manifold[5] $\mathcal{M}$, any automorphism[6] of $\mathcal{M}$ onto itself induces such a metric on $\mathcal{M}$. By designing a suitable family of automorphisms we obtain a family of pullback metrics we can optimise upon (cf. Section A.3). A strong rationale for pullback metric learning comes from the fact that, as we argue here for $\mathcal{M} = \mathbb{R}^n$ (Euclidean space is a special case of a manifold with only one coordinate chart), any differentiable, invertible (with no self-intersections) non-closed hypersurface can be made linear by stretching the data domain via an appropriate automorphism; ideal for a linear classifier. Moreover, for a non-linear classifier such as nearest neighbour, the automorphism may move points of the same class closer together, and further away from points of distinct classes (cf. Section A.3). Therefore, in general, we only need to explore the functional space of all automorphisms to find those able to produce the desired separation between classes.

[3] A linear or vector space $\mathcal{V}$ is a set that is closed under vector addition and scalar multiplication.

[4] A linear combination of models does not necessarily result in another valid model.

[5] A topological space $\mathcal{M}$ locally isomorphic to a Euclidean space, equipped with a *Riemannian metric* $g : T\mathcal{M} \times T\mathcal{M} \to \mathbb{R}$ which takes as input a pair of tangent vectors $\mathbf{v}, \mathbf{w} \in T\mathcal{M}$ and returns a scalar $g(\mathbf{v}, \mathbf{w})$, in a way which generalises the properties of the dot product of vectors in a Euclidean space.

[6] An isomorphism is a transformation which preserves a structure, in our case, the structure of smooth manifold.

1.5.5 Real-time and online multiple action detection

The aforementioned action recognition propositions (cf. Sections 1.5.1 - 1.5.4) are aimed at massive databases of videos processed offline. A potentially much larger source of visual data however will arise from future robotic platforms, with cameras operating continuously. Whereas pre-recorded video search and retrieval is aimed ultimately for human viewing, robotic systems need not record the raw visual data for later playback. Rather, robots will process the visual information as it arrives, and store only discriminative data in a representation useful for it to immediately understand its surroundings. Here we focus on the latter scenario in which a robot asks the questions:

i) Are there any relevant human actions happening now?
ii) If so, what kind of actions are they?
iii) How many are there?
iv) Where are they?

These questions may be answered by space-time human action detection, which we define as the task of associating a 3D subvolume that may be growing in time with an action category. The immediacy of the information required for human-robot interaction may be addressed by processing the data in an online fashion. We use the term 'online' to denote the incremental fashion by which the entire action recognition pipeline operates; that is, the space-time region-proposal generation, feature extraction, and learner are updated as soon as a video frame becomes available. Note that only current and past data is available to the robot [43].

Consider the following description of an 'open door' action: "A human stands in front of a door[1], moves his arm towards the keyhole[2], inserts a key[3], turns the key[4], pulls down on a handle[5], extends his arm to open the door[6], and enters the building[7]". It is tempting to approach the detection of the 'open-door' action [44] in a bottom-up, fully supervised fashion, requiring the detection of humans, the surrounding environment, interacting objects, and their movement in time. At some point during this process, it may be inferred that a person intended to open a lock, for the purpose of entering a building. Note that phrases $^{(2-4)}$ would remain identical if a person was starting a car [1]. This approach, however, is plight with difficulty, as each individual sub-action $^{(1-7)}$ requires a suite of detectors and a deluge of training data in order to generalise to a wide variety of scenarios.

In order to circumvent this problem, previous works [1] have treated human actions as 'space-time objects', and used video descriptors which capture the

object's appearance, motion, and environment context into an unordered collection of visual words [25]. However, state-of-the-art bag-of-visual-word action recognition systems are currently offline, since features are extracted and encoded assuming the entire video volume is available [20, 26]. Thus, a class label is assigned to a video sequence only after a sufficient number of frames have been processed [45]. Therefore, we propose to bridge a gap in the action recognition literature by including an online action recognition system capable of learning and detecting multiple space-time actions in a video stream.

1.6 Contributions

Firstly, I cast the conventionally supervised BoF action clip classification approach into a weakly supervised setting, where clips are represented as bags of histogram instances with latent class variables. In order to learn the subvolume class labels, I applied a multiple instance learning framework to 3D space-time videos (cf. Section 4.1), as actions may be better defined within a subvolume of a video clip rather than the whole video clip itself. Further, I proposed a mapping from *instance* decisions learned in the 'mi-SVM' approach to *bag* decisions (cf. Section 4.2), as a more robust alternative to the current bag margin MIL approach of taking the sign of the maximum margin in each bag. This allows our MIL-BoF approach to learn the labels of each individual subvolume in an action clip, as well as the label of the action clip as a whole. The resulting action recognition system is suitable for both clip classification and localisation in challenging video datasets, without requiring the labelling of action locations [46].

Secondly, I proposed to add deformable structure to local mid-level action representations (e.g BoF, Fisher vectors). This extended Spatial-BoF to allow the deformation of the rigid template at test-time (cf. Section 5.1). I also demonstrated quantitatively that our SVM-map strategy for mapping instance scores to global clip classification scores outperforms taking the argument of the maximum instance score in each video. Moreover, in Section 5.5 I show qualitative localisation results using a combination of classification and detection to output action-specific saliency maps; I was the first to show qualitative localisation results on challenging movie data such as the HMDB51 dataset.

Next, as my third contribution, when videos are represented as global bags of visual words, I demonstrated various design choices that lead to an improved classification performance across 5 action recognition datasets and 3 performance measures (cf. Chapter 6). I proposed a simple and effective feature sampling

strategy used to scale the vocabulary generation to a large number of classes and thousands of videos (cf. Algorithm 6.1). The outcomes of this evaluation suggested that i) sampling a balanced set of features per class gives minor improvements compared to uniform sampling, ii) generating separate visual vocabularies per feature component gives major improvements in performance, and iii) BoF per-category gives very competitive performance for a small number of visual words, achieving state-of-the-art on KTH with only 32 clusters per category, although Fisher vectors won out on the rest.

In fourth, when videos are represented as a time-series of observations, we proposed a general framework for learning optimal pullback distances given a training set of generative dynamical models, identified from a collection of labelled observation sequences (cf. Chapter 7). The parameters of the pullback distance which optimises classification performance were found by cross-validation [47]. We applied this framework to hidden Markov models in Section 7.3. We studied their product space structure and design there appropriate automorphisms. In Section 7.5 I conducted proof-of-concept tests, as well as experiments on the KTH and YouTube datasets which demonstrated the significant improvement in action classification rates (with respect to the chosen base distance) delivered by pullback learning under challenging conditions.

Finally, we outline an action detection framework which may learn new action categories online. My main contribution is a proposal based on local space-time tubes, formed by incrementally connecting region proposals in time. Along the way, I made algorithmic adaptations which will be crucial for incremental online learning and detection of multiple actions, including i) selective search ranking of region proposals, ii) selective search region correspondence in time, and iii) a variant of batch stochastic gradient descent with hard example mining and long-term class memory to support the inclusion of new categories online. An end-to-end system implementation and experimental evaluation is however left as future work.

1.7 Resulting publications and impact

Our work on 'learning discriminative space-time actions from weakly labelled videos' [48] won a poster prize at the INRIA Visual Recognition & Machine Learning (VRML) Summer School 2012. The corresponding paper [48] was soon after accepted for publication at the British Machine Vision Conference (BMVC

2012), for which it was accepted as an oral[7]. Later, we were invited to submit an extended version of the paper to the International Journal of Computer Vision (IJCV), an indication that it was among the top papers at BMVC. One of our original Figures (cf. Fig. 1.5) also appeared on the cover of the conference proceedings. An extended version of our BMVC paper including pictorial structures and more general subvolume shapes was later accepted at IJCV [20] (IF 3.62).

Our evaluation on feature subsampling and partitioning strategies for global bag-of-feature generation was archived as a technical report [49] with code available online[8]. The software used to generate the experiments was also used extensively in the following publications [20, 48, 50–52].

The results of our work on 'learning pullback-HMM distances' with application in human action recognition have been published in the IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI, IF 4.8) [51]. We recently submitted a journal paper applying pullback distance learning to the recognition of disease conditions from inertial measurement unit readings [52].

[7] `http://videolectures.net/bmvc2012_sapienza_labelled_videos/`

[8] `https://sites.google.com/site/mikesapi/downloads/global-video-representation`

Chapter 2

Related work

2.1 Interest point detection

In recent state-of-the-art methods, space-time feature extraction is initially performed in order to convert a video to a vectorial representation. Features are extracted around local points in each video, which are either determined by a dense fixed grid, or by a variety of Interest Point Detectors (IPDs) [26]. Whereas IPDs such as Harris3D [53], Cuboid [54] and Hessian [55] allow features to be extracted sparsely, saving computational time and memory storage, IPDs are not designed to capture smooth motions associated with human actions, and tend to fire on highlights, shadows, and video frame boundaries [56, 57]. Furthermore, [26] demonstrated that dense sampling outperformed IPDs in real video settings such as the Hollywood2 dataset [17], implying that interest point detection for action recognition is still an open problem.

2.2 Descriptors

A plethora of video features have been proposed to describe space-time patches, mainly derived from their 2D counterparts: Cuboid [54], 3D-SIFT [58], HoG-HoF [15], Local Trinary Patterns [59], HOG3D [60], extended SURF [55], and C2-shape features [61]. More recently [25] proposed 'dense trajectory' features which, when combined with the standard BoF pipeline [26], outperformed the recent 'learned hierarchical invariant' features [23, 62]. Therefore, even though this framework is independent from the choice of features, we used the dense trajectory features [25] to describe space-time video blocks.

Dense trajectory features are formed by the sequence of displacement vectors in an optical flow field, together with the HoG-HoF descriptor [15] and the

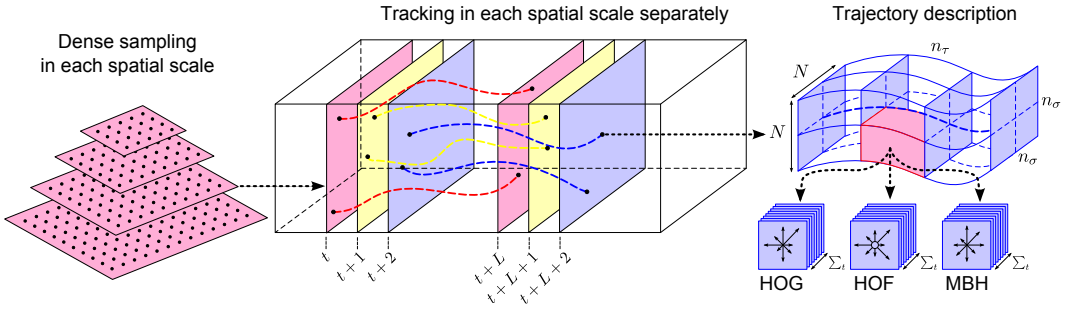


Figure 2.1: Dense trajectory feature extraction starts with a dense sampling of points at a discrete set of scales (left). Each point is subsequently tracked by mean filtering a dense optical flow [64] field, for a maximum of L frames (typically $L = 15$). Descriptors are calculated around the trajectory (right). Some structure is captured by splitting the trajectory volume into a fixed grid (Figure reproduced with permission from [25]).

motion boundary histogram (MBH) descriptor [63] computed over a local neighbourhood along the trajectory. The MBH descriptor represents the gradient of the optical flow, and captures changes in the optical flow field, suppressing constant motions such as camera panning. Thus, dense trajectories capture a trajectory's shape, appearance, and motion information. These features are extracted densely from each video at multiple spatial scales, and a pruning stage eliminates static trajectories such as those found on homogeneous backgrounds, or spurious trajectories which may have drifted [25]. The dense trajectory feature extraction process is illustrated in Fig. 2.1.

2.3 Mid-level representations

Many highly successful action recognition systems transform the aforementioned low-level descriptors [15, 25, 58, 60] into more invariant representations of 'intermediate', or 'mid-level' complexity [65]. One such mid-level representation is the bag-of-visual-words/bag-of-features (BoF) representation (cf. Section A.4), which represents a collection of low-level descriptors as an unordered collection of 'visual words', i.e., a histogram. A dictionary of visual words is usually generated by partitioning the descriptor space into a number K of clusters by k-means. Each video descriptor is then assigned to the nearest cluster in the visual vocabulary using the Euclidean distance (cf. Section A.5). A bag-of-feature histogram is defined by a vector $\mathbf{h} = (h_1, \ldots, h_K)^\top$, where h_k is the frequency of occurrence of visual word 'k' in $\mathbf{h}$.

Another successful mid-level representation is the Fisher vector [2]. Instead

of creating a visual vocabulary by clustering the feature space into K centroids by k-means, as done in the BoF approach, for Fisher vectors it is assumed that the features are distributed according to a Gaussian Mixture Model (GMM) with K components. Whereas in BoF, the feature quantisation step is a lossy process [66], a Fisher vector is formed through the soft assignment of each feature point to each Gaussian in the visual vocabulary. Therefore, instead of being limited by a hard assignment, it encodes additional information about the distribution of each feature. Let $\boldsymbol{D} = \{\mathbf{d}_\mathrm{t},\ \mathrm{t} = 1, \ldots, T\}$ be the set of dense trajectories extracted from part or the entirety of a video, and let the GMM have parameters $\Gamma = \{w_\mathrm{i}, \boldsymbol{\mu}_\mathrm{i}, \boldsymbol{\Sigma}_\mathrm{i}\},\ \mathrm{i} = 1, \ldots, K$, where the w_i's are mixture weights that satisfy $\sum_{\mathrm{i}=1}^{K} w_\mathrm{i} = 1$; $\boldsymbol{\mu}_\mathrm{i}$ is a mean vector, and $\boldsymbol{\Sigma}_\mathrm{i}$ is a covariance matrix with diagonal covariance; the variance vector along the diagonal of $\boldsymbol{\Sigma}_\mathrm{i}$ will be denoted as $\boldsymbol{\sigma}_\mathrm{i}$. Let $\gamma_\mathrm{t}(\mathrm{i}) = p(\mathrm{i}|\mathbf{d}_\mathrm{t})$ be the soft assignment of descriptor $\mathbf{d}_\mathrm{t}$ to the i^{th} Gaussian:

$$\gamma_\mathrm{t}(\mathrm{i}) = \frac{w_\mathrm{i} p_\mathrm{i}(\mathbf{d}_\mathrm{t})}{\sum_{\mathrm{j}=1}^{K} w_\mathrm{j} p_\mathrm{j}(\mathbf{d}_\mathrm{t})}. \tag{2.1}$$

Let D denote the dimensionality of descriptors $\mathbf{d}_\mathrm{t}$, and $\mathbf{g}_\mathrm{i}^{\boldsymbol{D}}$ be the D-dimensional gradient with respect to the mean $\boldsymbol{\mu}_\mathrm{i}$ of Gaussian 'i'. Mathematical derivations lead to:

$$\mathbf{g}_{\boldsymbol{\mu},\mathrm{i}}^{\boldsymbol{D}} = \frac{1}{T\sqrt{w_\mathrm{i}}} \sum_{\mathrm{t}=1}^{T} \gamma_\mathrm{t}(\mathrm{i}) \left(\frac{\mathbf{d}_\mathrm{t} - \boldsymbol{\mu}_\mathrm{i}}{\boldsymbol{\sigma}_\mathrm{i}} \right). \tag{2.2}$$

The final gradient vector $\mathbf{g}_\Gamma^{\boldsymbol{D}}$ is the concatenation of the $\mathbf{g}_\mathrm{i}^{\boldsymbol{D}}$ vectors, and therefore each Fisher vector is of dimensions $K \times D$, where K is the number of probabilistic visual words, and D is the dimensionality of each feature type [67, 68].

2.4 Learning the discriminative parts of an action video

As mentioned in the previous chapter, the current state-of-the-art algorithms for the classification of challenging human action data are based on the bag-of-features (BoF) on spatio-temporal volumes approach [15, 26]. However, its representational power diminishes with dataset difficulty (e.g. Hollywood2 dataset [17]) and an increased number of action classes (e.g. HMDB dataset [3]). This may be partly due to the fact that current BoF approaches represent entire video clips [25] or subsequences defined in a fixed grid [15]. Thus, many similar action parts, and background noise are also included in the histogram representation.

By splitting up the video clip into overlapping subvolumes, a video clip is instead represented as a bag of histograms, some of which are discriminative of the action at hand ('positive' subvolumes) while others ('negative' ones) may hinder correct classification. A more robust action model can therefore be learned based on these 'positive' subvolumes in the space-time video. Moreover, the classification of local subvolumes has the additional advantage of indicating *where* the action is happening within the video.

The BoF approach has been coupled with single frame person/action detection to gain more robust performance, and to estimate the action location [1,69]. In contrast, by learning discriminative action subvolumes from weakly-labelled videos, the method we propose allows action localisation *without* using any training ground truth information, in a similar spirit to [16,56]. Unlike previous work, however, we select discriminative feature histograms and not the explicit features themselves. Moreover, instead of using a generative approach such as pLSA [70], we use a max-margin multiple instance learning (mi-SVM) framework to handle the latent class variables associated with each space-time action part.

In many machine learning applications such as molecular activity, document categorisation and image/video detection, gathering detailed labels for the purpose of supervised training is prohibitively expensive. This makes weakly supervised algorithms such as multiple instance learning (MIL) very attractive, as only groups of examples need to be labelled. In MIL, examples or 'instances' are grouped together in positive and negatively labelled 'bags', without knowledge of the instance labels. However, it is assumed that a negatively labelled bag only contains negative instances, and that a positive bag must contain at least one positive instance [71, 72]. The aim is to learn a model which is able to categorise bags, its instances, or both [20, 48], after training on such weakly labelled data. Gaidon *et al.*. [73] also leveraged weakly labelled video datasets, this time for learning a higher-level global activity representation. In [73], a video is decomposed into an unordered binary tree of mid-level BoFs encoding related space-time motion patterns. Instead of using human annotated labels to split each video into motion-related parts, a clustering algorithm is employed.

Some insight to MIL comes from its use in the context of face detection [74]. Despite the availability of ground truth bounding box annotation, the improvement in detection results when compared to those of a fully supervised framework suggested that there existed a more discriminative set of ground truth bounding boxes than those labelled by human observers. The difficulty in manual labelling arises from the inherent ambiguity in labelling objects or actions (bounding box

scale, position) and the judgement, for each image/video, of whether the context is important for that particular example or not. A similar MIL approach was employed by Felzenszwalb and Huttenlocher [75] for object detection in which possible object part bounding box locations were cast as latent variables. This allowed the self-adjustment of the positive ground truth data, better aligning the learned object filters during training. In action detection, Hu *et al.* used an MIL learning framework called SMILE-SVM [76]; however, this focused on the detection of 2D action boxes, and required the approximate labelling of the frames and human heads in which the actions occur. In contrast, we propose casting the space-time subvolumes of cubic/cuboidial structure as latent variables, with the aim to capture *salient action patches* relevant to the human action.

In action clip classification only the label of each action clip is known, and not the labels of individual parts of the action clip. Thus, this problem is inherently *weakly-labelled*, since no approximate locations of the actions or ground truth action bounding boxes are available. That is why we propose to learn action subvolumes in a weakly-labelled, multiple instance learning (MIL) framework. At test time, human action classification is then achieved by the recognition of action instances in the query video, after devising a sensible mapping from instance scores to the final clip classification decision. To this end, we use multiple instance learning (MIL), in which the recovery of *both* the instance labels and bag labels is desired, without using two separate iterative algorithms [71]. Our proposed *SVM-map* strategy provides a mapping from instance scores to bag-scores which quantitatively outperforms taking the argument of the maximum score in each bag.

2.5 Discriminative video part extensions

Following on from the previous section (§ 2.4), we propose to use the discriminatively learned action video parts to initialise a pictorial structure model for each action class (§ 2.5.1), and to expand the subvolumes to more general shapes rather than just fixed sized cubes, with application in space-time action saliency (§ 2.5.2).

2.5.1 Incorporating mid-level pictorial structures

Attempts to incorporate structure into the BoF representation for video classification have been based on the spatial pyramid approach [15]; here a spatio-

temporal grid was tuned for each dataset and action class. Although spatial pyramids have been successful in scene classification [77], in which parts of the scene consistently appear in the same relative locations across images, it is unclear whether they are useful for difficult clips such as those captured from mobile devices, in which the same action can appear in any location of the video.

In order to model human actions at a finer scale, it is desirable to localise the spatial and temporal extent of an action. Initial work by [1] learnt a boosted cascade of classifiers from spatio-temporal features. To improve space-time interest point detectors for actions such as 'drinking', the authors incorporated single frame detection from state-of-the-art methods in object detection. It is however desirable to remove the laborious and ambiguous task of annotating keyframes, especially when considering huge online video datasets.

In another approach, [69] split the task into two: firstly by detecting and tracking humans to determine the action location in space, and secondly by using a space-time descriptor and sliding window classifier to temporally locate two actions (phoning, standing up). In a similar spirit, our goal is to localise actions in space and time, rather than time alone [78–80]. However, instead of resorting to detecting humans in every frame using a sliding window approach [69], we localise actions directly in *space-time*, either with 3D bounding box detection windows (Fig. 1.6), or by aggregating detection scores to form a saliency map (Fig. 1.7).

To incorporate temporal structure into the BoF framework, Gaidon *et al.*. [79, 80] proposed to model actions as a sequence of key atomic action units or 'actoms'. Their actom sequence models enforce a soft ordering between temporal parts of an action in a particular category, and need to be annotated manually during training. In video event detection, [57] used a search strategy in which oversegmented space-time video regions were matched to manually constructed volumetric action templates. Inspired by the pictorial structures framework [81], which has been successful at modelling object part deformations [75], [57] split their action templates into deformable parts making them more robust to spatial and temporal action variability. Despite these efforts, the action localisation techniques described [1, 57, 69] require manual labelling of the spatial and/or temporal [79, 80] extent of the actions/parts in a training set. In contrast, we propose to localise human actions via a pictorial structure model automatically from weakly labelled observations, *without* human location annotation. Moreover, unlike previous work [16, 56], we select local discriminative subvolumes represented by mid-level features [65] such as BoF and Fisher vectors, and not

the low-level space-time features themselves (e.g. HoG, HoF).

In order to incorporate object structure, [75] used a star-structured part-based model, defined by a 'root' filter, a set of 'parts', and a structure model. However, the objects considered had a clear boundary which was annotated by ground truth bounding boxes. During training, the ground truth boxes were critical for finding a good initialisation of the object model, and also constrained the plausible position of object parts. Furthermore, the aspect ratio of the bounding box was indicative of the viewpoint it was imaged from, and was used to split each object class into a mixture of models [75]. In contrast, for the action classification task, the spatial location, the temporal duration and the number of action instances are not known beforehand. Therefore we propose an alternative approach in which part models are learnt by splitting the 'root' filter into a grid of fixed regions, as done in spatial pyramids [27, 77]. Unlike the rigid global grid regions of spatial pyramids [77] or spatial-BoF [27], our rigid template used during training is *local*, not global, and is allowed to deform at test-time to better capture the action warping in space and time.

2.5.2 General subvolume shapes and action saliency

Recent work by [23] used saliency models to prune features and build discriminative histograms for the Hollywood2 action *classification* dataset [17]. Amongst various automatic approaches to estimate action saliency, such as tensor decomposition, the best approach selected features according to their spatial distance to the video centre; a central mask. This approach is ideal for Hollywood movies in which actors are often centred by the cameraman, but less well suited for general videos captured 'in the wild'. Furthermore, the saliency masks in [23] were precomputed for each video individually, without considering the dataset's context. For example, in a dataset of general sports actions, the presence of a swimming pool is highly discriminative of the action 'diving'. Less so in a dataset which contains only different types of diving action categories. Thus, in our view, action saliency should also depend on the *differences* between actions in distinct classes. Learning the discriminative parts of a video (cf. Section 2.4) may be used to address this issue and highlight salient parts of the video specific to an action class.

Whereas in the previous section, only fixed-sized subvolumes were used, we propose to extend the set to include a variety of subvolume shapes, since different actions may be better captured by a different-sized subvolumes. Moreover, since actions may not be well bounded inside cubic/cuboidial shapes, we experiment

with aggregating the scores from a set of subvolume shapes to generate a saliency map (cf. Section 5.5), with promising qualitative results.

2.6 Design choices in the bag-of-visual-words pipeline for video data

Current state-of-the-art human action classification systems rely on the global aggregation of local space-time features [19], to form a bag-of-features representation (cf. Section 2.3). The areas in which BoF may be improved were broadly outlined by Nowak *et al.* [30] and Jiang *et al.* [31]. These include the *patch sampling strategy*, for which uniform random sampling of the image space was shown to outperform interest point-based samplers such as Harris-Laplace and Laplacian of Gaussian [30]. Previous action recognition BoF evaluation [26] focused on comparing a variety of local spatio-temporal features, and also found that dense sampling consistently outperformed interest point detection in realistic video settings. Later, the authors of [26] proposed a new *visual descriptor* called 'dense trajectories' which achieved the state-of-the-art in action classification [25]. Further areas of improvement were the *feature-space partitioning algorithm*, for which Nister *et al.* [32] extended k-means to produce vocabulary trees, and Farquhar *et al.* [21] used a Gaussian mixture model instead of k-means; the *visual vocabulary size* also improved performance, though it saturated at some data-dependent size [30]. Another area of improvement was the *weighting of frequency components* in a histogram (tf-idf) [33], and the use of sets of bipartite histograms representing universal and class-specific vocabularies [29]. More recently, Fisher and VLAD vectors achieved excellent results by using small vocabularies and soft quantisation, instead of a hard assignment to form high dimensional vectors [67, 68]. A comprehensive evaluation of various feature encoding techniques on image classification datasets was provided by Chatfield *et al.* [82]. Finally the choice of *classification algorithm* has largely been dominated by support vector machines [15, 25, 28]. In contrast to previous work [26,30,82], here we focus on feature subsampling and partitioning *after* feature extraction has taken place but *prior* to encoding, with the aim of improving the subsequent dictionaries learned for encoding features.

For earlier systems which used BoF for object retrieval in videos [33], building a visual vocabulary from all the training descriptors was known to be a "gargantuan task" [33]. Thus researchers tended to select subsets of the features,

dedicating only a sentence to this process. For example, quoting [33], "a subset of 48 shots is selected covering about 10k frames which represent about 10% of all frames in the movie", amounting to 200k 128-D vectors for learning the visual vocabulary. Csurka *et al.* trained a vocabulary with "$600k$ descriptors" [28], whilst Nister *et al.* used a "large set of representative descriptor vectors" [32]. In order to construct visual vocabularies, Jiang *et al.* used all the training keypoint features in the PASCAL dataset, and subsampled $80k$ features in the TRECVID dataset [31]. In the evaluation proposed in Chapter 6, we clarify this process by comparing two specific methods to sample features from the smallest to largest action recognition datasets available. The sheer volume of features extracted from each dataset used here is listed in Table 2.1.

It is clear that with thousands of videos in the training set, in practice one cannot use all of the features to build a vocabulary. Also note that in video classification datasets, longer sequences have disproportionately more features than shorter ones, in contrast to image datasets. Moreover, some action classes also have many more videos associated with them than others[1]. To see the extent by which the number of features per video varies in action recognition, check the difference between the 'Maximum' and 'Minimum' rows of Table 2.1. Thus, in general, sampling uniformly at random may result in a pool of features which is biased towards a particular class. For example in the Hollywood2 dataset [17], 'driving' action clips are typically much longer than 'standing up' clips. We hypothesise that the subsampling strategy may have a significant impact on the classification performance especially when dealing with a large number of action classes and videos.

The selection of good partitioning clusters to form a visual vocabulary is also important, as they form the basic units of the histogram representation on which a classifier will base its decision [21]. Thus, for a categorisation task, having clusters which represent feature patches that distinguish the classes is most likely to make classification easier. This motivates per-category clustering, to preserve discriminative information that may be lost by a universal vocabulary [21], especially when distinct categories are very similar. The downside is that learning a separate visual vocabulary per class may also generate many redundant clusters when features are shared amongst multiple categories. On the other hand, since the complexity of building visual vocabularies depends on the number of cluster centres K, clustering features independently allows to reduce the number of K

[1]Bias in the number of examples per class is common in many areas of computer vision, for example, the semantic segmentation of road scenes where 70% of the image pixels are labelled as road.

Table 2.1: Various statistics on the amount of dense trajectory features extracted per video from the KTH, Hollywood2 (HOHA2), HMDB and UCF101 datasets. The number of action categories are denoted within parenthesis. The 'Memory' and 'Sum' are calculated over the whole dataset assuming 32-bit floating point. Following these are the mean, standard deviation, median, maximum, and minimum number of features per video in each dataset.

	KTH (6)	**HOHA2 (12)**	**HMDB (51)**	**UCF-101**
Memory (GB)	6	45	160	679
Sum	$4_{,000,000}$	$28_{,000,000}$	$99_{,000,000}$	$421_{,000,000}$
Mean	$9_{,000}$	$34_{,000}$	$16_{,000}$	$32_{,000}$
Std Dev	$5_{,000}$	$44_{,000}$	$14_{,000}$	$30_{,000}$
Median	$8_{,000}$	$20_{,000}$	$13_{,000}$	$22_{,000}$
Maximum	$36_{,000}$	$405_{,000}$	$138_{,000}$	$410_{,000}$
Minimum	967	438	287	358

whilst keeping the representation dimensionality high ($K \times C$), and makes the vocabulary learning easily parallelisable. So far, per-category training has seen promise on a single dataset with a small number of classes [21]; it is therefore to be seen how it performs on challenging action classification data with a large number of action classes.

2.7 Pullback distances for time-series classification

When representing an action video as a sequence of 'mid-level' observations, its recognition becomes a problem of time series classification [83] which is applied, among others, to stock market predictions, electric motor fault diagnosis [84] and EEG analysis [85]. A number of techniques have been proposed for time series classification, including SVMs and boosting of interval-based predicates [86]. Neural networks have been widely employed, often to first and second order statistics [87], or in combination with MAP estimation [88].

While most of the literature analyses times series without recurring to an intermediate representation, the use of dynamical models has also been explored. State-space analysis has indeed been widely proposed for modelling and regression with time series. A first example is [89], where a metric for ARIMA models based on their autoregressive representation was introduced. Lyapunov exponents have been used as features for classifying speech signals [90]. In [91] the

authors perform maximum likelihood classification based on a Gaussian Mixture Model (GMM) representation in the Reconstructed Phase Space (RPS), a topological embedding of the original time series.

Dynamical models provide a compact representation for time series which can mitigate the related dimensionality issue[2], but cannot be naively classified. Metric learning is a natural answer. The various forms of Mahalanobis distance learning such as Relevant Component Analysis (RCA) [42] assume an L_2 distance on the original data space $\mathcal{M} = \mathbb{R}^m$ and a linear mapping between $\mathcal{M}$ and a transformed space $\mathcal{N} = \mathbb{R}^n$. This is a special case of pullback learning, in which both the original and the transformed space are linear and the map linking them is also linear: $\mathbf{y} = \boldsymbol{A}\mathbf{x}$. Indeed, in the pullback paradigm the automorphism $F : \mathcal{M} \to \mathcal{N}$ can map the model space to a different metric space [92] (cf. Section A.1).

Pullback metrics' adoption has been recently proposed by Lebanon for document retrieval. In [92], however, rather than classification rates, the inverse volume of the pullback manifold is maximised in an unsupervised setting. Moreover, for hidden or variable length [93] Markov models, a proper Riemannian metric is difficult to identify. For this reason, we need to relax the constraint of having a proper manifold structure, by considering mere distance functions in a metric space[3].

Thus, in Chapter 7, we propose to maximise classification performance considering models for which a Riemannian structure is not known. In contrast, Jaakkola and Haussler's Fisher kernels [94] are purely geometrical (they make use of the Fisher geometry of families of distributions). Lafferty and Lebanon's 'heat' kernels [95] are also purely geometric, and require a Riemannian structure on the data. Unlike geodesic distances, they are Mercer kernels that can be used in an SVM directly.

A strong rationale for pullback metric learning comes from its consequences on class separation. In kernel approaches [96], classes that cannot be linearly separated in the original domain are more likely to be so in a higher-dimensional 'feature space'. In the pullback setup, instead of looking for better separation in a higher-dimensional feature space, we seek to stretch the original space to maximise classification performance. For instance, if a linear separator is used,

[2] The curse of dimensionality, which broadly refers to the volume of space increasing exponentially as the dimensionality of your data representation increases. In machine learning this usually necessitates an exponential increase in the number of samples used for training to avoid overfitting the data.

[3] A set for which distances between all members of the set are defined and satisfy all properties of a metric (c.f Section A.1).

it can be proven that for any differentiable, invertible non-closed hypersurface (separating data points of two different classes) in $\mathcal{M} = \mathbb{R}^n$ there exists an automorphism of $\mathbb{R}^n$ that maps it to a hyperplane (cf. Section A.2). Note that this does not hold for *closed* boundaries (e.g. a circle in $\mathbb{R}^2$). Consequently, we only need to thoroughly explore the functional space of all automorphisms to find the 'right' one, able to produce the desired class separation.

The design of a specific parametrised family of automorphisms provides a more limited search space which makes this search feasible in practice, recalling parametrised families of kernels [97], limiting at the same time overfitting issues that would arise when searching for arbitrary automorphisms to perfectly fit the available training data. When the dataset of models is labelled, we can determine the optimal distance function in a supervised setting. From the above argument on rectifying nonlinear boundaries (also cf. Section A.2) it follows that, in our pullback framework, we should seek a max-margin separation approach in a SVM-inspired fashion. However, this is not trivial in the nonlinear case treated here[4].

A variety of objective functions were introduced in the metric learning literature, such as the ratio between inter class and intra class covariance [98], and mutual information [42]. To be able to use them in our framework we need to express them as a function of the metric. When the training set is *unlabelled*, distance function learning has to be based on purely geometrical considerations. Lebanon [92] has suggested to maximise in closed form the inverse of the volume element or Gramian '$\det g$' associated with a metric g around the given training set of points. When the dataset of models is *labelled*, we can determine the optimal metric/distance function in a supervised setting. In the linear case analytical solutions can be achieved by convex optimisation [99]. In fact, just as in the case of [42] and [92], we can imagine an objective function which allows optimisation in closed form, so that no numerical optimisation is necessary. Some proposals in this direction are formulated in Section 7.5.5.

For the nonlinear case considered here, we use *cross-validation* [47] to optimise the classification performance of the pullback distance. We extract samples from the automorphism's parameter space and pick those with maximal performance on the validation folds. Although the samples' density may be a limiting factor, it is related to the dimensionality of the automorphism's parameter space. Thus in general, with careful design of a family of automorphisms, its parameter

[4] The space of HMM models is nonlinear since addition and scalar multiplication of models does not in general produce another valid HMM.

space need not be a function of the dimensionality of the model space or the features. It is up to us to design a family of automorphisms powerful enough to generate performance gains, while being computationally feasible and able to limit overfitting issues. As the original distance function is obtained under the identity automorphism, the optimal pullback distance is guaranteed to improve performance on the validation folds. The results obtained in Section 7.5 indicate that this typically generalises to the test data as well.

2.8 Online multiple action detection

Motivated by the potential of humans to teach robots and interact with them, the need for online multiple human action detection arises. However, to the best of our knowledge, action recognition systems have not yet crossed this milestone. One reason being that current action detection systems do not incorporate all of the following necessary components:

i) video-stream processing,
ii) space-time region proposals and features,
iii) online learning, and
iv) multiple action detection[5].

In what follows, we will describe the current state-of-the-art in order to make clear our contributions.

2.8.1 Multiple action detection in space and time

Several works attempt to localise actions in space alone using a single-frame detector [100,101], or in time alone [78,79]. Here we focus on space-time detection, defined by either a 3D cuboid, or a set of 2D windows associated in time to form a 'tube', 'subvolume', or 'track', with one window in each frame and forming a continuous segment without holes [44].

Laptev *et al.* pioneered one of the first works on action detection in realistic scenarios [1], with a boosted 3D space-time window classifier combined with a single frame action detector. Multiple action detection may incur an expensive exhaustive search in space and time; this was alleviated in [1] by restricting the search space around detected 2D 'keyframes'. Since the action detection system was developed for video indexing applications, all video processing was carried

[5]We use the term 'localisation' for cases in which it is assumed that an action is always present in a video clip, and the task is to locate it. For 'detection', there may not be any action in the video, and therefore another class is needed to denote the absence of an action.

out offline. Another offline method by Willems *et al.* [102] extended an exemplar based object detection to action recognition. In order to alleviate an expensive search over space-time and scales, discriminative visual words extracted during training were stored with ground truth annotation, and used as a prior for possible detection locations. A greedy grouping procedure was used to form 3D detection windows from overlapping space-time hypothesis.

In order to reduce the action search space, Klaser *et al.* [69] first generated action hypothesis by detecting and tracking humans, and subsequently classified the space-time 'tracks'. Frame-by-frame detections were generated by training an upper human body histogram-of-oriented-gradient (HoG) detector [103], and associating the detected windows in time using KLT-tracked features. Interpolation and smoothing was needed to reduce noise and fill in gaps, whilst an additional classifier was necessary to prune erroneous tracks. Note that using trained detectors for a particular human body position like the upper body is restrictive to a class of actions in which the upper body is discriminative of the action. An efficient decision tree approach was adopted by Mikolajczyk *et al.* [104], in which multiple action detection was achieved by accumulating scores for local features on each frame along the decision tree path. However, unlike [69], localisation of actions was only performed over scale-space, and thus without association in time. In this case, a robot would be unable to recognise that a walking action over multiple frames is in fact a single action[6]. In order to associate the 2D detection windows in time, Xie *et al.* [105] used the L1 distance between colour histograms over detections in neighbouring frames. In [105], actions were represented by a fixed-length sub-sequence of deformable part model (DPM [75]) scores over a segment of video frames. It is not clear how well this approach performs however, since it was not tested on publicly available detection benchmarks.

2.8.2 Online video stream-processing

Despite recent advances in online action recognition [45, 59], current systems are only tailored for the recognition of a single action class per video frame. For example, Yeffet *et al.* [59] proposed an online action recognition system inspired by local binary patterns, in which grid based video regions were represented as a histogram of Trinary strings, encoded by comparing pixel regions in neighbouring

[6] Note that it is not easy for us humans to determine whether an action detection system is connecting actions in time from a video with bounding box annotations, since we perform the data association effortlessly whilst watching.

frames. A video segment descriptor is formed from the concatenation of these histograms over the grid. Whilst this approach is fast and online, it only encodes motion information, and crucially is limited to recognise a single action category per frame.

Yu *et al.* [45] designed a very fast action recognition system based on ensembles of random decision trees that quickly translate interest points into visual codewords for classification. The downside is that it also only allows single action classification per group of frames. Moreover, the use of fast interest point detectors makes it sensitive to image pixel noise, and may fire on salient video regions which are not relevant to the human action [20, 57].

2.8.3 Proposed online multiple action detection system

A person or object may potentially be seen in any location of an image and may be small or large depending on its distance form the camera. Thus previous researchers [75] explored the entire image space exhaustively using a sliding window at multiple scales. Due to the computational load of an exhaustive search, however, various methods were proposed to select image regions which have a high chance of containing an object of interest [106–109]. Whereas 'objectness' [107] style classifiers depend on the data they were trained on, the 'selective search' [109] algorithm uses a graph-based unsupervised segmentation algorithm [110] to generate object proposals from the image structure. Since selective search is category independent, it is ideal for the online learning of new categories. However, the full selective search method [109] combined with, for instance, CNN features [111] may still take tens of seconds per image [112]. Therefore, we propose to augment the selective search algorithm [109] by learning incrementally the region proposal action relevance, and only pass a small subset for expensive feature extraction.

Actions may be detected by bounding them in cubes/cuboids [1, 20, 48], however in this case a more flexible enclosing structure is provided by a connected set of windows in time, commonly called tracks [69] or tubes [113]. Building upon recent work on temporally consistent superpixels [114, 115] may provide a mechanism to extract region proposals in 3D without resorting to an exhaustive search [20]. Grundmann *et al.* [115] achieved temporally coherent superpixels by casting a video as a 3D graph for segmentation, however this assumes that the entire video is known apriori, making online recognition impractical. In contrast, Chang *et al.* [114] find temporal correspondence between superpixels in time. The upside is that superpixel correspondences may be found even when

the pixel regions are not connected in time, as is often the case for quick movements [114]. However, with the generative probabilistic model used in [114], inference takes tens of seconds per video frame, too slow for practical online applications. Instead we propose to augment the selective search strategy once more, this time by associating region proposals in time via combinatorial optimisation. Connecting region proposals in time to form tubes allows the extraction of motion vectors, which may subsequently be used as discriminative features.

For representation learning and classification, the field of Neural Networks is back to the forefront of machine-learning research, with an increased depth of network layers and more effective learning strategies [18, 116]. The increased depth of the network layers enables it to learn multiple levels of representation corresponding to different levels of feature abstraction, rather than resorting to hand-crafted features. These deep networks have recently achieved state-of-the-art performance on tasks such as speech recognition, character recognition [117], and image classification where Krizhevsky *et al.* [111] reduced the error rate on ImageNet from 26.1% to 15.3%. Recently, Girshick *et al.* also pushed the best performance on the PASCAL object detection challenge [118] from a mean average precision of 34% to 48%; a massive leap forward. For this reason, we propose to use the CNN network architecture as [111] trained on the ImageNet, and observe whether the network output transfers to the task of action detection. Thus we use the output of the network as highly non-linear mapping from the raw image pixels, to features suitable for classification.

The CNN features, after being aggregated and normalised to action tube proposals, may then be classified using an online linear SVM with hard negative mining [75]. In contrast to the stochastic gradient descent algorithm described by Felzenszwalb and Huttenlocher [75], which sampled training examples from a large offline dataset of images, here we design a batch variant which samples examples from the current set of action tubes. Note that there will only be a subset of the classes appearing in the video at any one time, and we found that without keeping a 'previously seen' action example cache, the incremental learner begins to 'forget' the action categories which appeared in the past.

Previous works [20,75,112] used online learning to tackle large static datasets. In contrast, in this work:

i) examples are received incrementally in time;
ii) the number of action categories grows incrementally;
ii) action categories that appear for a short duration in time need to be remembered, and respective models updated when new classes appear later

on in time;

iii) convergence of the learner over time is not desirable, since if a new class appears, all other models need to be updated.

With these differences in mind, we designed a multi-class incremental online learning algorithm (Algorithm 8.2), and tailor a classical stochastic gradient descent algorithm for learning SVM models (Algorithm 8.3).

In the following proposed methodology we do not assume any training has been done beforehand on the dataset in use, for instance, to detect possible action locations [1, 9], to detect humans [69], to refine features [112], or bootstrap the learning [75]. We assume that at the first frame of training the recognition system has never seen the dataset before. We believe that an online incremental learner should learn as it views the world from one frame to the next, using only information it has seen in the past and the present [43]. This is a paradigm shift which separates our work from [1, 45, 59, 69, 105], and is key to evaluate how well new action classes may be learned 'on the fly'. A detailed methodology of the proposed online multiple action detection framework is laid out in Chapter 8. In the next Chapter, we review the datasets and performance measures we used to evaluate action classification and detection methods.

Chapter 3

Datasets and performance indicators

The datasets in this chapter fall into two groups: those designed to evaluate action clip classification algorithms (§ 3.1), and those designed for action detection (§ 3.2).

3.1 Action classification

Amongst many action classification datasets, we chose a subset from the relatively easy to the most challenging, suitable to benchmark and compare various action recognition algorithms. The task here is to automatically assign the correct action label to each video, on a so called 'test' set; a set of videos previously unknown to the computer/learner. A variety of performance indicators/metrics are commonly used to quantify how well a computer was able to perform classification (cf. Section 3.1.6). A graph showing the state-of-the-art accuracy for various action classification datasets[1] in chronological order can be seen in Fig. 3.1.

3.1.1 KTH

The *KTH* dataset[2] [13] contains 6 action classes, 25 actors, and four scenarios. The actors perform repetitive actions at different speeds and orientations. Sequences are longer when compared to those in the YouTube [16] or the HMDB51 [3] datasets, and contain clips in which the actors move in and out of the scene

[1] Results reported from March 2012.

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

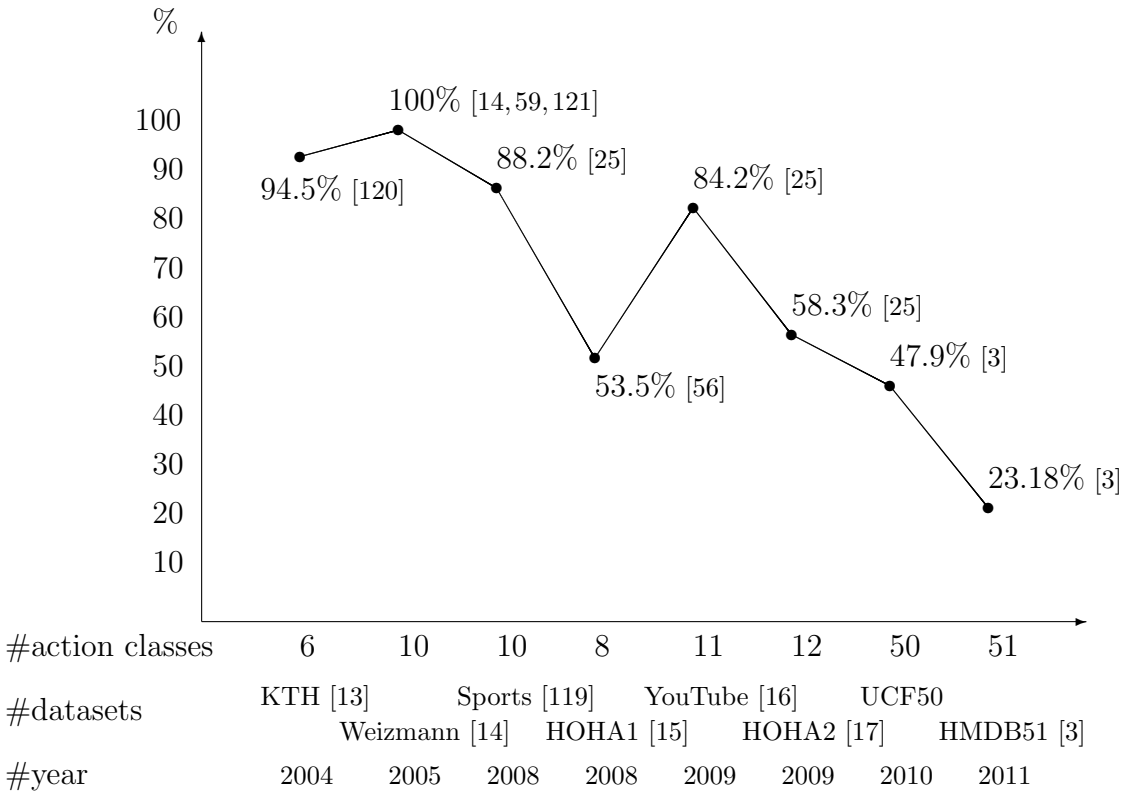


Figure 3.1: State-of-the-art results plotted for a number of datasets in date-of-release order (older - newer). One may see a general decrease in performance as the number of action classes increases. Note that the difficulty of each dataset does not only depend on the number of classes, it also much depends on the differences between videos of distinct classes (similar classes makes classification harder), and the differences between videos used in training and testing (similar videos makes classification easier).

during the same sequence. In our experiments, we split the video samples into training and test sets as in [13], and considered each video clip in the dataset to be a single action sequence. Sample images from the KTH dataset are shown in Fig. 3.2.

3.1.2 YouTube

The *YouTube* dataset[3] [16] contains videos collected online from 11 action classes. The data is challenging due to camera motion, cluttered backgrounds, and a high variation in action appearance, scale and viewpoint. In our tests, 1600 video sequences were split into 25 groups, following the author's evaluation procedure of 25-fold, leave-one-out cross validation. An image sample from each action class of the YouTube dataset is shown in Fig. 3.3.

[3] `http://www.cs.ucf.edu/~liujg/YouTube_Action_dataset.html`

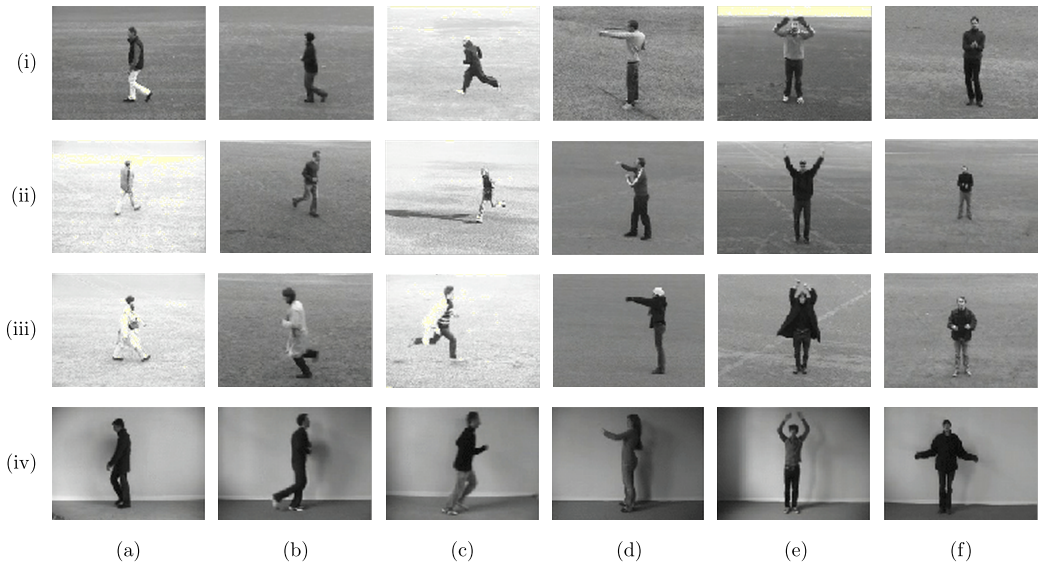


Figure 3.2: Image samples from the KTH dataset. Rows (i-iv) show four different scenarios: (i) outdoors, (ii) outdoors with scale variation, (iii) outdoors with different clothes, and (iv) indoors. Columns (a-f) show the six action classes: (a) walking, (b) jogging, (c) running, (d) boxing, (e) handwaving, and (f) handclapping.

3.1.3 Hollywood2

The *Hollywood2* dataset[4] [17] contains instances of 12 action classes collected from 69 different Hollywood movies. There are a total of 1707 action samples each 5-25 seconds long, depicting realistic, unconstrained human motion, although the cameras often have their view centred on the actor. The dataset was divided into 823 training and 884 testing sequences, as in [17]. Note that all videos in this dataset were initially downsampled to half their size; a modification to be taken into account when comparing results. This downsampling procedure was also done in [62], due to high video resolutions, and the huge amount of features that would be subsequently extracted. Samples of images from each action class are presented in Fig. 3.4.

3.1.4 HMDB51

The *HMDB* dataset[5] [3] contains 51 action classes, with a total of 6849 video clips collected from movies, the Prelinger archive, YouTube and Google videos. Each action category has a minimum of 101 clips. We used the non-stabilised videos with the same three train-test splits proposed by the authors [3]. A collection of image samples is shown in Fig. 3.8, where each action class is denoted directly

[4] `http://www.di.ens.fr/~laptev/actions/hollywood2/`
[5] `http://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/`

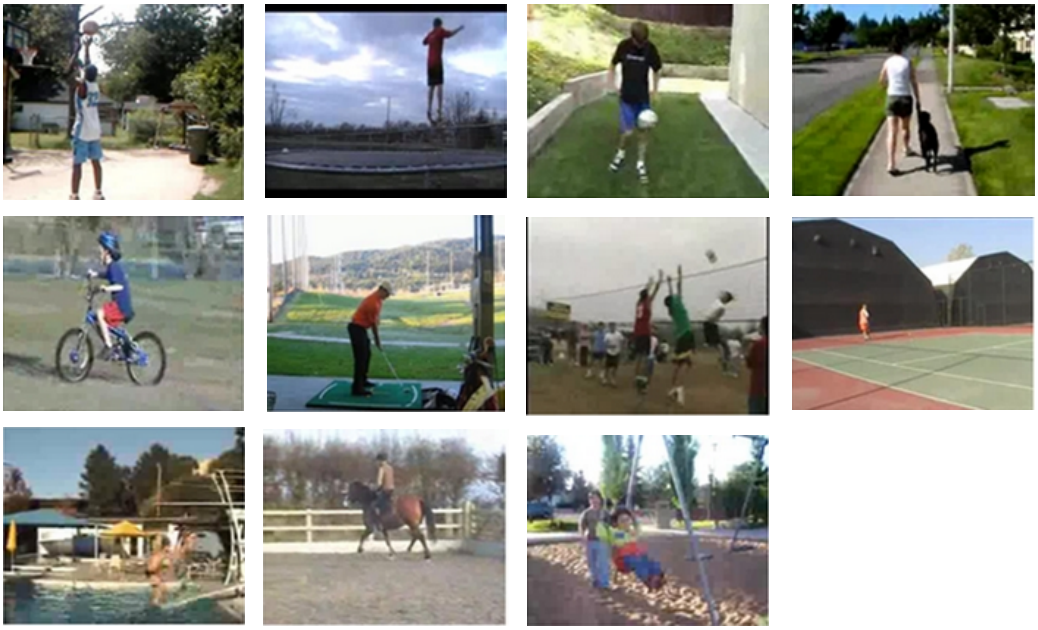


Figure 3.3: Image samples from the YouTube dataset. From top left down to bottom right the action classes are: basketball shooting, trampoline jumping, soccer juggling, walking with a dog, biking/cycling, golf swinging, volleyball spiking, tennis swinging, diving, horse back riding and swinging. Note that many of the YouTube actions were captured from low quality cameras, and contain significant camera movement.

below each image.

3.1.5 UCF101

The *UCF101* dataset [34] contains 101 action classes, approximately 13,000 clips and 27 hours of video data. This is currently the largest video classification dataset to date. The sheer volume of data, high number of action classes and unconstrained videos make this a very hard dataset. We used the recommended three train-test splits proposed by the authors [34]. Image samples from each action class in the dataset are shown in Fig. 3.9. A complete list of action categories can be found online[6].

3.1.6 Performance indicators

Previous authors (e.g. [1, 26]) have evaluated action classification performance through a *single* measure, such as the *accuracy* or *average precision*. In our experimental evaluation, we used *three* performance measures for each dataset, in order to present a more complete picture of each algorithm's performance, as

[6] `http://crcv.ucf.edu/data/UCF101.php`

Figure 3.4: Image samples from the Hollywood2 dataset. Can you guess what the action classes are from the sample images? Answers below[7]*. This exercise is intended to show the reader that it is not always easy for us to recognise an action class from a single still image.*

presented in the following subsections.

Accuracy

The classification Accuracy (Acc) is calculated as the #correctly classified testing clips /#total testing clips. Typically, for a multi-class classification task with k $\in \{1, \ldots, K\}$ classes, and i $\in \{1, \ldots, N\}$ examples, a $K \times K$ confusion matrix $\boldsymbol{C}$ is first constructed by incrementing the element of the matrix at indices $\boldsymbol{C}[\text{gt(i)}, \text{pred(i)}]$, for each pair of ground truth and predicted labels. A typical confusion matrix is presented in Table. 3.1. The accuracy is then the trace of $\boldsymbol{C}$, divided by the total sum of its elements:

$$Acc = \frac{a + e + i}{N}, \tag{3.1}$$

where a, e and i are the diagonal elements of the 3×3 confusion matrix shown in Table. 3.1.

[7] From top left to bottom right: GetOutCar, Run, SitUp, DriveCar, StandUp, AnswerPhone, SitDown, FightPerson, Eat, HandShake, Kiss, HugPerson.

Table 3.1: A typical confusion matrix for an action classification task with $K = 3$ classes.

		Predicted label			
		Run	Box	Hop	Total
Ground truth label	Run	a	b	c	$a+b+c$
	Box	d	e	f	$d+e+f$
	Hop	g	h	i	$g+h+i$
	Total	$a+d+g$	$b+e+h$	$c+f+i$	N

F1-score

The F1-score is found by weighting the 'Recall' and 'Precision' of a classifier equally and is calculated as the ratio:

$$F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}. \tag{3.2}$$

The Recall measures the fraction of correctly classified examples from one class, to the total number of examples of that class in the ground truth. The Recall was calculated separately for each class as:

$$\text{Recall(k)} = \frac{\boldsymbol{C}[k,k]}{\sum_{\text{b}=1}^{\text{b}=K} \boldsymbol{C}[\text{k, b}]}, \quad \text{or in MATLAB: } \frac{\boldsymbol{C}(\text{k, k})}{\texttt{sum}(\boldsymbol{C}(\text{k}, :))}. \tag{3.3}$$

The Precision is the fraction of correctly classified examples from a particular class, in the total number of examples predicted as being of that particular class:

$$\text{Precision(k)} = \frac{\boldsymbol{C}[k,k]}{\sum_{\text{a}=1}^{\text{a}=K} \boldsymbol{C}[\text{a, k}]}, \quad \text{or in MATLAB: } \frac{\boldsymbol{C}(\text{k, k})}{\texttt{sum}(\boldsymbol{C}(:, \text{k}))}. \tag{3.4}$$

To report a single number quantifying the classification performance, the F1 score for each class may be averaged as follows:

$$mF1 = \frac{\sum_k F1_{\text{k}}}{K}. \tag{3.5}$$

Average Precision

The Average precision (AP), is a score which considers the ordering in which the results are presented, and is best used for retrieval/ranking tasks. Nevertheless, it has been widely used to benchmark classification problems in the action recognition literature, and is presented here to allow comparison to other works.

To calculate the average precision, it is first necessary to sort the results by their classification score, and consider what the values of precision and recall

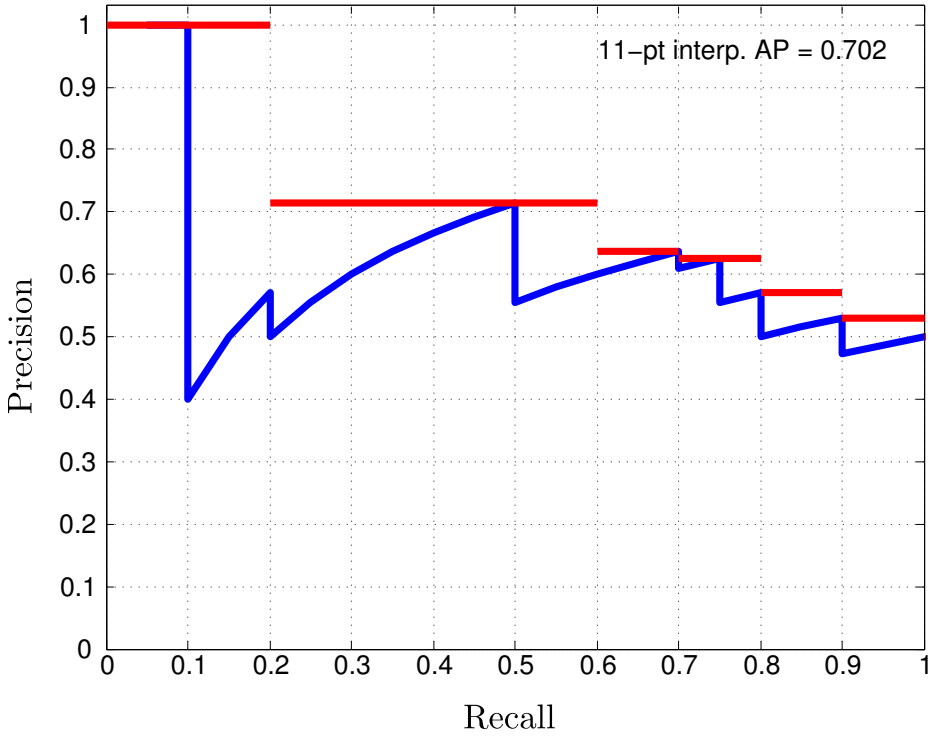


Figure 3.5: An example of a precision-recall curve generated from random example scores to illustrate the difference between the strict average precision (area under blue curve), and the 11-point interpolated average precision (area under red curve).

would be had the retrieved examples been in the top 'i', where 'i' varies from $1, \ldots, N$. It is now possible to plot a graph with the N values of precision and recall as points, joined by straight lines, as shown by the blue precision-recall curve in Fig. 3.5.

The average precision value, which is strictly equal to the precision averaged over all values of recall, equates to taking the area under the precision-recall curve. However, to follow convention, in this work we use the 11-point interpolated average precision, in which the precision p_{r} at a certain recall level 'r' is defined as the maximum precision recorded for any recall level $\mathrm{r}' \geq \mathrm{r}$:

$$p_{\mathrm{r}} = \max_{\mathrm{r}' \geq \mathrm{r}} p_{\mathrm{r}'}. \tag{3.6}$$

A thorough analysis and justification for this measure can be found in [122]. In practice the 11-point interpolated average precision in most cases overestimates the area under the precision recall curve, as illustrated by the red lines in Fig. 3.5. In a similar fashion to the F1 score, the mean average precision (mAP) is used to give a single performance indicator for a multi-class classification problem.

Comparison and discussion

In order to compare each performance measure, we conducted two experiments from randomly generated classifier scores. In the first experiment, we compared the performance obtained when classifying scores generated uniformly at random over a varying number of classes. We expect the outcome of a random classifier to be on average $1/K$ for a K class problem. The averaged results over 1000 runs is displayed in Fig. 3.6. Note that each class has the same number of examples.

In the second experiment, we benchmarked the performance of a random classifier against an increasing imbalance in the number of examples per class. The number of classes was fixed at 2, with one class having 20 examples and the other having $20^n - 20$ more examples, where n was varied from 1 to 4, as shown in Fig. 3.7. Again, results were averaged over 1000 runs.

Notice in Fig. 3.6 that the Acc and mF1 are exactly the same given the same number of examples per class, and diverge with a class imbalance (see Fig. 3.7). This demonstrates the importance of using the F1 score when working with datasets in which there can be dominating and rare classes; for example if a dataset had 100 examples of a walking action and only 10 examples of a cartwheel action, then if all 110 examples were predicted as walking, the accuracy would still be 91%, whilst the F1 score drops to 48%.

For a multi-class classification problem with an imbalance in the number of samples per class, the accuracy may be normalised by dividing each row of the confusion matrix by the sum of the elements in each row. The normalisation ensures that each class receives the same weight in the accuracy calculation. Since the 11-point average precision overestimates the area under the precision recall curve, the exact area under the curve is also used as a performance measure (see the PASCAL VOC challenge [118]). Stating the performance measure being used is thus essential when comparing different algorithms, and still not commonplace.

3.2 Action detection

In order to evaluate our online multiple action detection algorithm, we selected the challenging LIRIS human activities dataset [44]. The LIRIS dataset is attractive because it contains image sequences containing multiple actions annotated in space and time, some of which occur simultaneously, as shown in Fig.3.10. Moreover, it contains scenes with human actions amidst other irrelevant human motion (other people performing irrelevant actions). The LIRIS dataset contains

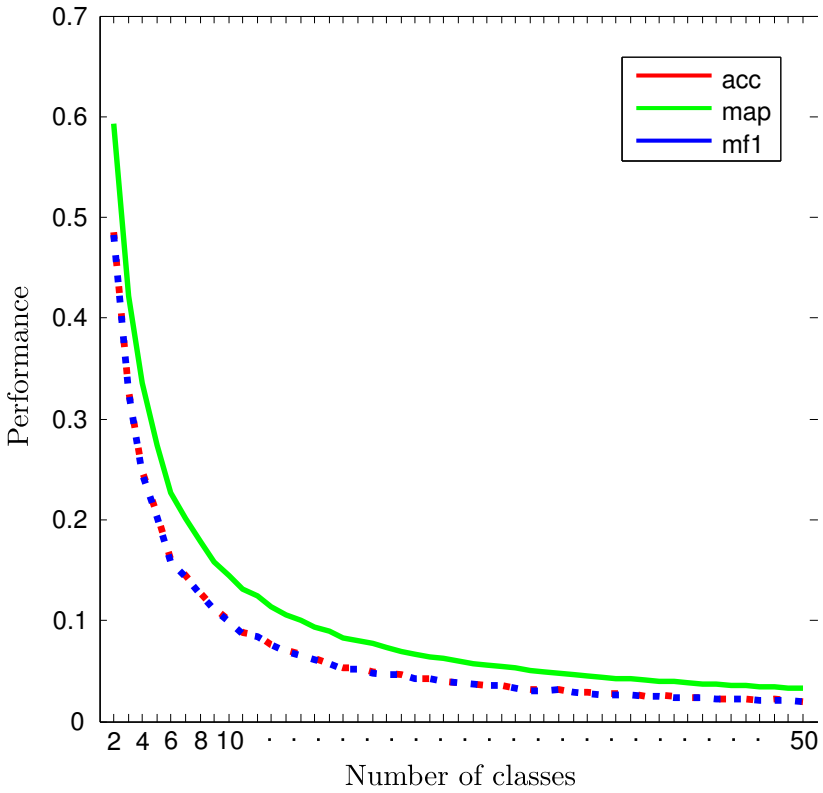


Figure 3.6: A plot of the classification performance (accuracy, 11-point mean average precision, and mean F1 score) against the number of classes. Since the classifier scores are generated randomly we expect to see the curve $1/K$, followed accurately by the accuracy and mF1 score, and slightly overestimated by the mAP.

10 action categories, which include human-human interactions and human-object interactions, for example, 'discussion of two or several people', and 'a person types on a keyboard'. A full list of categories may be found on the dataset's website[8]. In particular, we used the D2 sequences shot with a Sony camcorder with a resolution of 720×576, and captured at 25 frames per second.

3.2.1 Performance indicators

The quantitative performance was calculated with the evaluation tool provided for the LIRIS-HARL competition [44]. First, a detected action tube is assigned to the closest ground truth tube based on the normalised overlap over all frames. Second, a detected action tube is accepted as positive if the tubes have the same class, and

i) if there is sufficient overlap with respect to thresholds on *spatial pixel-wise recall* t_{sr}, and *temporal frame-wise recall* t_{tr},

[8] http://liris.cnrs.fr/voir/activities-dataset

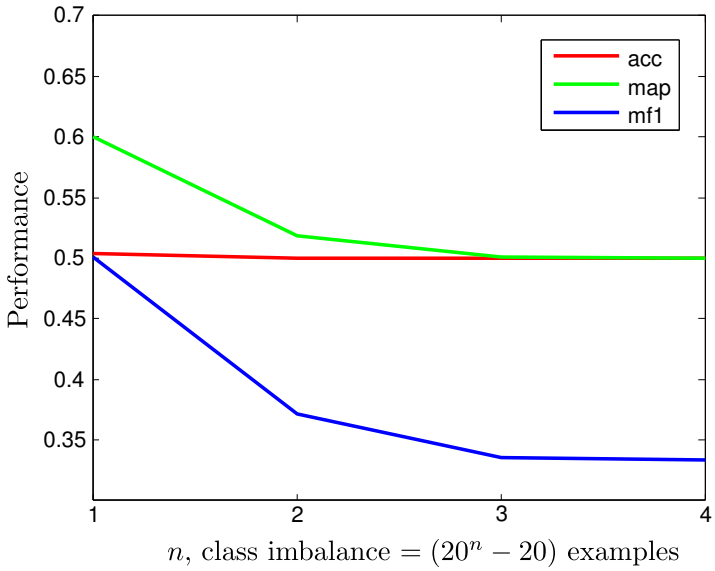


Figure 3.7: A graph showing the classification performance against the imbalance in examples per class, for a two class problem. The accuracy hugs the 0.5 line irrespective of the number of examples per class, whilst the mAP initially overestimates and then converges to 0.5 *when the imbalance is over* 8000 *examples. The mF1 score decreases gradually as the imbalance increases, demonstrating that it is a more reliable metric to use when there is a disproportionate number of action/object classes in a dataset.*

ii) if the excess duration is sufficiently small with respect to thresholds for *spatial pixel-wise precision* t_{sp}, and *temporal frame-wise precision* t_{tp}.

By fixing the four thresholds that need to be satisfied, the recall and precision may be calculated as:

$$\text{Recall} = \frac{\#\text{correctly found actions}}{\#\text{actions in ground truth}} \tag{3.7}$$

$$\text{Precision} = \frac{\#\text{correctly found actions}}{\#\text{number of found actions}} \tag{3.8}$$

The F1-score combines these measures as defined in Equation 3.2. A final performance measure may be obtained by integrating the F1-score measure over the range of possible threshold values. First four integrated F1-score values $(I_{sr}, I_{sp}, I_{tr}, I_{tp})$ are calculated by varying one threshold and keeping the others fixed at a small value ($\eta = 0.1$). Finally an averaged score is obtained by averaging the four values:

$$\text{Integrated Performance} = \frac{I_{sr} + I_{sp} + I_{tr} + I_{tp}}{4} \tag{3.9}$$

The final score is advantageous to use because it is independent of arbitrary thresholds on the spatial or temporal overlap [123].

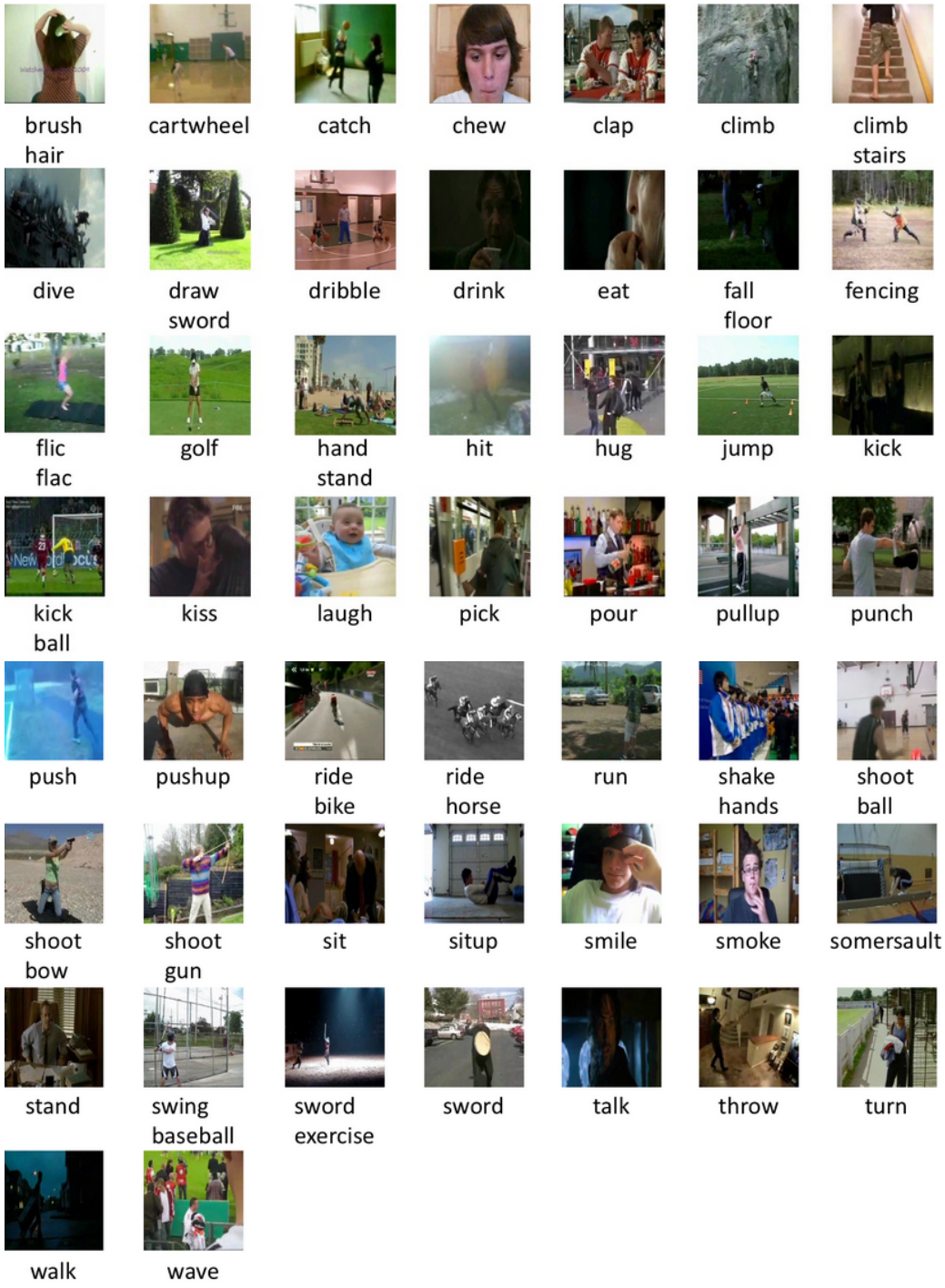


Figure 3.8: Image samples from the HMDB51 dataset. The 51 action categories may be grouped into the following categories: general facial actions (e.g. chew), facial actions with object manipulations (e.g. drink), general body movements (e.g. somersault), body movements with object interaction (e.g. brush hair), and bodily movements with human interaction (e.g. shake hands).

Figure 3.9: Image samples from the UCF101 dataset. The authors [34] claim that the UCF101 dataset contains the largest collection of diverse actions, with large variations in camera motion, object appearance, pose, object scale, viewpoint, cluttered background and illumination conditions. The action categories can be divided into five types: human-object interaction, body-motion only, human-human interaction, playing musical instruments and sports.

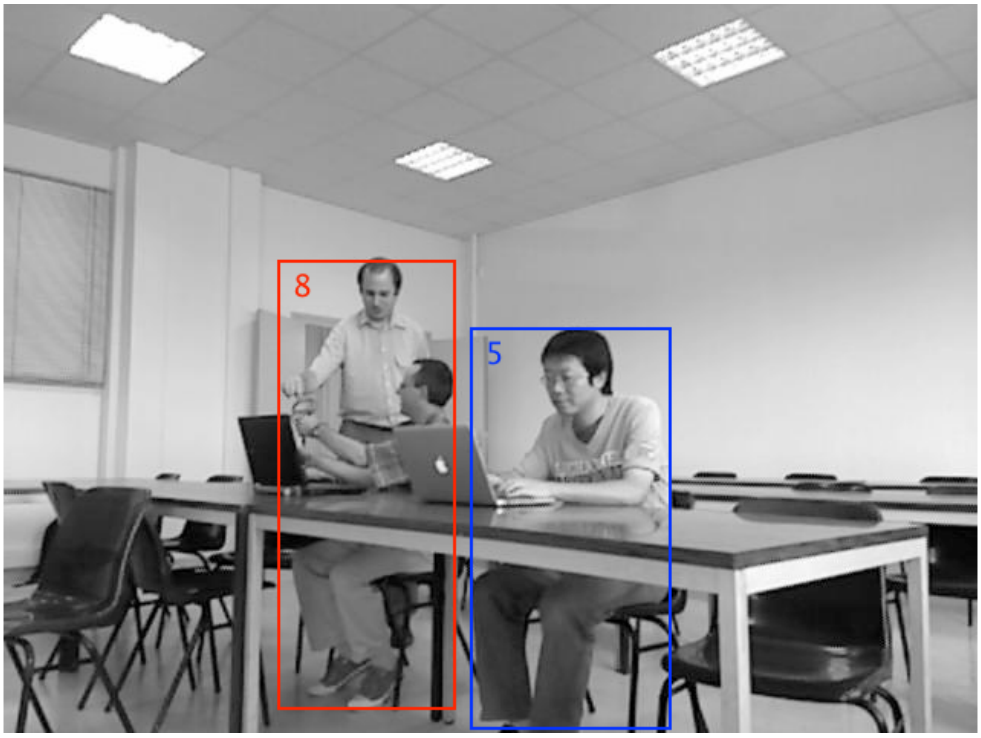


Figure 3.10: A sample image from the LIRIS human activities dataset used to benchmark action detection systems. The image show two different actions occurring simultaneously; the red tube numbered 8 has a label 'a person gives an item to a second person', whilst the blue action tube numbered 5 has a label 'a person typing on a keyboard'.

Chapter 4

Learning discriminative actions from weakly labelled videos

This chapter is motivated by two main ideas:

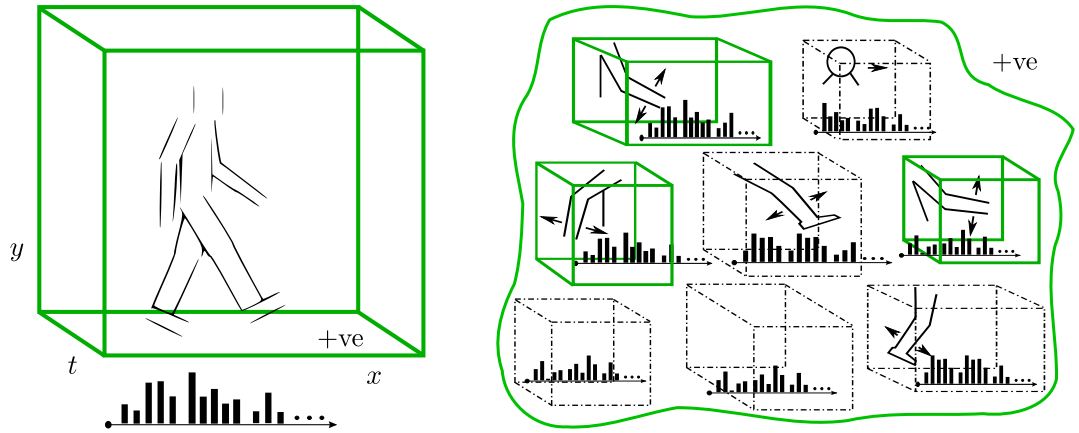
- an action model should be learned from those parts of the video which are relevant to the action class, and
- local action models may be used to locate discriminative video patches of human actions in large video datasets.

Since in many large datasets, collecting the locations of actions in videos by human annotators for training is prohibitively expensive, we propose an action recognition system that leverages weakly labelled videos. In the following sections, we present the three main building blocks making up our approach, namely:

i) the description of space-time videos via histograms of dense trajectory features [25] (cf. Section 2.2);
ii) the representation of a video clip as a 'bag of subvolumes', and the learning of positive subvolumes from weakly labelled training sequences within a max-margin multiple instance learning framework (cf. Section 4.1), and
iii) the mapping of instance/subvolume scores to bag/clip scores by learning a hyperplane on instance score features (cf. Section 4.2).

4.1 MIL-BoF action models

In this work, when using BoF, we define an *instance* to be an individual histogram obtained by aggregating the dense trajectory features within a local subvolume, and a *bag* is defined as a set of instances originating from a single space-time video. Since we perform multi-class classification with a *one-vs-all*



(a) Global space-time volume (b) Bag of general local subvolumes

Figure 4.1: Instead of defining an action as a space-time pattern in an entire video clip (a), an action is defined as a collection of space-time action parts contained in subvolumes shapes of cube/cuboidial shape (b). One ground-truth action label is assigned to the entire space-time video or 'bag', while the labels of each action subvolume or 'instance' are initially unknown. Multiple instance learning is used to learn which instances are particularly discriminative of the action (solid-line cubes), and which are not (dotted-line cubes).

approach, we present the following methodology as a binary classification problem.

In action classification datasets, each video clip is assigned to a single action class label. By decomposing each video into multiple instances, now only the class of the originating video is known and not those of the individual instances. This makes the classification task *weakly labelled*, where it is known that positive examples of the action exist within the video clip, but their exact location is unknown. If the label of the bag is positive, then it is assumed that one or more instances in the bag will also be positive. If the bag has a negative label, then all the instances in the bag must retain a negative label [71]. The task here is to learn the class membership of each instance, and an action model to represent each class, as illustrated in Fig. 4.1.

The learning task may be cast in a max-margin multiple instance learning framework (MIL), of which two methods were proposed in [71]; a 'bag-margin' and an 'instance-margin' formulation, the former designed to estimate 'bag-labels' and the latter to estimate 'instance-labels'. Since we are interested in both the bag and instance labels, we first adopted the instance-margin formulation and subsequently proposed a mapping from instance scores to bag-labels, presented in the following section (§ 4.2).

Let the training set $\mathcal{D} = (\langle \boldsymbol{X}_1, Y_1 \rangle, \ldots, \langle \boldsymbol{X}_N, Y_N \rangle)$ consist of a set of bags,

where $\boldsymbol{X}_{\mathrm{i}} = \{\mathbf{x}_{\mathrm{i}1}, ..., \mathbf{x}_{\mathrm{i}M_{\mathrm{i}}}\}$ of different length M_{i}, with corresponding ground truth labels $Y_{\mathrm{i}} \in \{-1, +1\}$. The range of the indices are $\mathrm{i} \in \mathcal{I} = \{1, \ldots, N\}$, and $\mathrm{j} \in \mathcal{J} = \{1, \ldots, M_{\mathrm{i}}\}$. Each instance $\mathbf{x}_{\mathrm{ij}} \in \mathbb{R}^n$ represents the j^{th} BoF histogram in the i^{th} bag, and has an associated latent class label $y_{\mathrm{ij}} \in \{-1, +1\}$ which is initially unknown for the positive bags ($Y_{\mathrm{i}} = +1$). The class label for each bag Y_{i} is positive if there exists at least one positive instance in the bag, that is,

$$Y_{\mathrm{i}} = \max_{\mathrm{j}} \{y_{\mathrm{ij}}\}. \tag{4.1}$$

Therefore the task of the mi-MIL is to recover the latent class variable y_{ij} of every instance in the positive bags, and to simultaneously learn an SVM instance model $\langle \mathbf{w}, b \rangle$ to represent each action class.

The max-margin mi-SVM learning problem results in a semi-convex optimisation problem, for which [71] proposed a heuristic approach. In mi-SVM, each instance label is unobserved, and we maximise the usual soft-margin jointly over hidden variables and discriminant function:

$$\min_{y_{\mathrm{ij}}} \min_{\mathbf{w}, b, \xi} \ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{\mathrm{ij}} \xi_{\mathrm{ij}}, \tag{4.2}$$

$$\begin{aligned} \text{subject to :} \ \ & y_{\mathrm{ij}}(\mathbf{w}^\top \mathbf{x}_{\mathrm{ij}} + b) \geq 1 - \xi_{\mathrm{ij}}, \quad \forall \mathrm{i}, \mathrm{j} \\ & y_{\mathrm{ij}} \in \{-1, +1\}, \quad \xi_{\mathrm{ij}} \geq 0, \\ \text{and} \ & \sum_{\mathrm{j} \in \mathcal{J}} (1 + y_{\mathrm{ij}})/2 \geq 1 \ \ \text{s.t.} \ Y_{\mathrm{i}} = +1, \\ & y_{\mathrm{ij}} = -1 \ \ \forall_{\mathrm{j} \in \mathcal{J}} \ \text{s.t.} \ Y_{\mathrm{i}} = -1, \end{aligned}$$

where $\mathbf{w}$ is the normal to the separating hyperplane, b is the offset, and ξ_{ij} are slack variables for each instance $\mathbf{x}_{\mathrm{ij}}$. The heuristic algorithm proposed by [71] to solve the resulting mixed integer problem is laid out in Algorithm 4.1.

Consider training a classifier for a *walking* class action from the bags of training instances in a video dataset. Initially all the instances are assumed to have the class label of their parent bag/video (STEP 1). Next, a walking action model estimate $\langle \mathbf{w}, b \rangle$ is found using the imputed labels y_{ij} (STEP 2), and scores

$$f_{\mathrm{ij}} = \mathbf{w}^\top \mathbf{x}_{\mathrm{ij}} + b, \tag{4.3}$$

for each instance in the bag are estimated with the current model (STEP 3).

Algorithm 4.1 Heuristic algorithm proposed by [71] for solving mi-SVM.

STEP 1. Assign positive labels to instances in positive bags: $y_{ij} := Y_i$ for $j \in \mathcal{J}$
repeat
  STEP 2. Compute SVM solution $\langle \mathbf{w}, b \rangle$ for instances with estimated labels y_{ij}.
  STEP 3. Compute scores $f_{ij} := \mathbf{w}^\top \mathbf{x}_{ij} + b$ for all $\mathbf{x}_{ij}$ in positive bags.
  STEP 4. Set $y_{ij} := \text{sgn}(f_{ij})$ for all $j \in \mathcal{J}$, where $Y_i = +1$.
  for all positive bags $\boldsymbol{X}_i$ **do**
    if $\sum_j (1 + y_{ij})/2 = 0$ **then**
      STEP 5. Find $j^* := \underset{j \in \mathcal{J}}{\text{argmax}}\, f_{ij}$, set $y_{ij}^* = +1$
    end if
  end for
until class labels do not change
Output $\mathbf{w}$, b

Whilst the negative labels remain strictly negative, the positive labels may retain their current label, or switch to a negative label (STEP 4). If, however, all instances in a positive bag become negative, then the least negative instance in the bag is set to have a positive label (STEP 5), thus ensuring that there exists at least one positive example in each positive bag.

Now consider walking video instances whose feature distribution is similar to those originating from bags in distinct action classes. The video instances originating from the walking videos will have a positive label, whilst those from the other action classes will have a negative label (assuming a 1-vs-all classification approach). This corresponds to a situation where points in the high dimensional instance space are near to each other. Thus, when these positive walking instances are reclassified in a future iteration, it is likely that their class label will switch to negative. As the class labels are updated in an iterative process, eventually only the discriminative instances in each positive bag are retained as positive.

4.2 A learnt mapping from instance to bag labels

So far the focus has been on learning instance-level models for detection. However in order to make a classification decision, the video class label as a whole also needs to be estimated.

The instance margin MIL formulation detailed in section 4.1 aims at recovering the latent variables of all instances in each positive bag. When recovering the optimal labelling y_{ij} and the optimal hyperplane $\langle \mathbf{w}, b \rangle$ (4.2), *all* the positive and negative instances in a positive bag are considered. Thus, only the query instance labels may be predicted:

$$\hat{y}_{ij} = \text{sgn}(\mathbf{w}^\top \mathbf{x}_{ij} + b). \tag{4.4}$$

An alternative MIL approach called the 'bag margin' formulation is typically adopted to predict the *bag* labels. The 'bag-margin' approach adopts a similar iterative procedure as the 'instance margin' formulation, but only considers the 'most positive' and 'most negative' instance in each bag. Therefore predictions take the form:

$$\hat{Y}_i = \text{sgn} \max_{j \in \mathcal{J}} (\bar{\mathbf{w}}^T \mathbf{x}_{ij} + \bar{b}), \tag{4.5}$$

where $\langle \bar{\mathbf{w}}, \bar{b} \rangle$ are the model parameters learnt from the 'bag-margin' formulation [71].

In order to avoid this iterative procedure for retrieving the bag labels, we propose a simple and robust alternative method in which bag scores are directly estimated from the instance scores f_{ij} (4.3). One solution is to use the same max decision rule in (4.5) with the instance scores: $\hat{Y}_i = \text{sgn} \max_j(f_{ij})$. However, the scores from the max may be incomparable, requiring calibration on a validation set to increase performance [124,125]. Moreover, better cues may exist to predict the bag label. Cross-validation can be used to select a threshold on the number of positive instances in each bag, or a threshold on the mean instance score in each bag. The downside is that the number of instances in each bag may vary significantly between videos, making values such as the mean instance score between bags incomparable. For example, in a long video clip in which a neatly performed action only occupies a small part, there would be large scores for instances containing the action, and low scores elsewhere. Clearly, the mean instance score would be very low, even though there was a valid action in the clip.

As a more robust solution we propose to construct a feature vector by combining multiple cues from the instance scores f_{ij} in each bag, including the number of positive instances, the mean instance scores, and the maximum instance score in each bag. The feature vector $\mathbf{v}_i$ is constructed as follows:

$$\mathbf{v}_i = \Big(\#p,\ \#n,\ \frac{\#p}{\#n},\ \frac{1}{M_i} \textstyle\sum_j (f_{ij}),\ \max_{j \in \mathcal{J}}(f_{ij}),\ \min_{j \in \mathcal{J}}(f_{ij}) \Big), \tag{4.6}$$

where $\#p$ and $\#n$ are the number of positive and negative instances in each bag respectively. In this way, the variable number of instance scores in each bag are represented by a six-dimensional feature vector $\mathbf{f}_{\mathrm{i}} \mapsto \mathbf{v}_{\mathrm{i}}$, and a linear SVM decision boundary, $\langle \mathbf{w}', b' \rangle$, is learnt from the supervised training set $\mathcal{D}' = (\langle \mathbf{v}_1, Y_1 \rangle, \ldots, \langle \mathbf{v}_N, Y_N \rangle)$, in this constant dimensional space. Now predictions take the form:

$$\hat{Y}_{\mathrm{i}} = \mathrm{sgn}(\mathbf{w}'^{\top} \mathbf{v}_{\mathrm{i}} + b'). \tag{4.7}$$

Apart from mapping multiple instance scores to single bag scores, this SVM-map strategy generates comparable bag scores for various action classes, thus avoiding any instance score calibration [124].

4.3 Experimental Evaluation

In order to validate our action recognition system, we evaluated its performance on four challenging action datasets, namely the KTH, YouTube, Hollywood2 and HMDB datasets, as detailed in Chapter 3. Next we present the baseline pipeline (§ 4.3.1), followed by the details of our MIL-BoF experimental setup (§ 4.3.2), and an ensuing discussion (§ 4.4).

In order to make the comparison across different datasets fairer, all clips were down-sampled to a common 160×120 resolution. For each dataset, we present both the state-of-the-art result as reported in the literature, and the baseline BoF results in our own implementation, to which we compare our MIL-BoF on sub-volumes framework. As performance measures, we report the accuracy (mAcc), the average precision (mAP), and the mF1-score, defined in Chapter 3. Unlike previous work, we are the first to report *all three* performance measures for each dataset, to give a more complete picture of the overall algorithm performance.

4.3.1 Baseline BoF algorithm

We implemented the baseline BoF approach described in [26] to ensure a fair comparison between BoF and MIL-BoF. A codebook was generated by randomly sampling 100,000 features and clustering them into $K = 4000$ visual words by k-means. Descriptors were assigned to their closest vocabulary word using the Euclidean distance, and the resulting histograms of visual words were used to represent each video clip. We reported the performance achieved using a χ^2-kernel SVM [26], and performed multi-class classification using the *one-vs-all* approach. We fixed the histogram normalisation to the $L1$-norm, and kept

the SVM regularisation constant $C = 100$ throughout, the same value used by [26, 62].

4.3.2 MIL-BoF experimental setup

The same BoF setup as the baseline has been used for the MIL-BoF approach. Subvolumes were extracted from a regular grid with a grid spacing of 20 pixels in space and time. Results were reported for a number of different MIL-BoF models, each characterised by different cube-[60-60-60], [80-80-80], [100-100-100] or cuboid-[80-80-160], [80-160-80], [160-80-80] shaped subvolumes, where [x-y-t] denotes the dimensions of the subvolume. In addition, we also allowed for a certain type of cuboid to stretch along the total time duration of the clip [80-80-end], in a similar spirit to the spatial grid (with no division in time) used by [15] for adding weak geometrical information to the video's representation.

The decomposition of a video into multiple subvolumes, each with the same histogram dimensionality as used in the baseline, makes the learning problem at hand *large-scale*. Typical values for the number of instances generated from the KTH dataset range between 100,000-200,000. In practice calculating the full χ^2-kernel takes a prohibitively long time to compute. Recent work by Vedaldi and Zisserman on the homogeneous kernel map [126] demonstrates the feasibility of large scale learning with non-linear SVMs based on additive kernels, such as the χ^2-kernel. The map provides an approximate, finite dimensional feature representation in closed form, which gives a good approximation of the desired kernel in a compact linear representation. The map parameters were set to $N = 2$, and $\gamma = 0.5$, which gives a $K \times (2^N + 1)$ dimensional approximated kernel map for the χ^2-kernel. Similarly to the baseline, we keep the SVM parameters constant across all datasets at C=0.1, which has proven to give good results in practice. The quantitative results are shown in Table 4.1.

4.4 Results and discussion

On the **KTH** dataset the MIL-BoF approach surpassed the baseline BoF in all three performance measures, demonstrating a clear advantage of representing videos with subvolumes on this dataset. Common scene and motion elements were pruned by the multiple-instance learning as shown in Fig. 1.5, resulting in a stronger action classifier per class. Contrary to our expectations, both the BoF and MIL-BoF surpassed the state-of-the-art accuracy, which may be attributed

Table 4.1: Quantitative results from the state-of-the-art (S-o-t-a), our BoF baseline, and our MIL-BoF method for various fixed-size subvolumes.

Dataset:	**KTH**			**YouTube**			**HOHA2**			**HMDB**		
Perf:	mAcc	mAP	mF1	mAcc	mAP	mF1	mAcc	mAP	mF1	mAcc	mAP	mF1
S-o-t-a:	94.5 [56, 120]		–	**93.77** [127]		–	–	**63.3** [19]		**66.79** [127]		–
BoF:	95.4	96.5	94.0	76.0	79.3	57.5	39.0	**48.7**	32.0	**31.5**	**31.4**	21.4
MIL-BoF:												
60-60-60	94.9	96.5	94.2	73.4	81.0	70.0	38.5	43.5	**39.4**	27.6	26.3	23.1
80-80-80	95.4	**97.0**	94.8	77.5	83.9	73.9	37.3	44.2	37.5	28.7	29.0	25.3
100-100-100	93.5	96.5	93.7	78.6	85.3	76.3	37.4	40.7	32.3	27.5	28.6	23.9
80-80-160	**96.8**	96.7	95.8	**80.4**	86.1	**77.4**	37.5	42.0	33.7	28.2	29.6	**25.4**
160-80-80	96.3	96.6	94.4	79.1	85.0	76.1	36.9	42.1	32.1	29.0	30.5	24.8
80-160-80	95.8	96.6	94.4	78.3	84.9	75.7	37.8	42.6	35.3	28.7	28.8	25.3
80-80-end	**96.8**	96.9	**96.0**	79.3	**86.1**	75.9	**39.6**	43.9	36.0	29.7	30.3	25.2

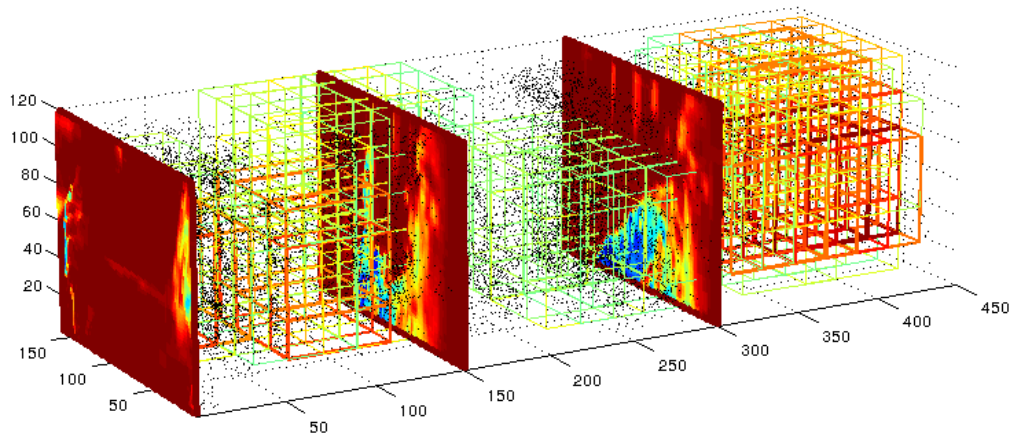
to using the whole action videos rather than clean action slices during training. The best result was achieved using a subvolume model more extended in time than in space [80-80-160], that achieved 96.76% accuracy. Similarly on the **YOUTUBE** dataset, the MIL-BoF framework outperformed the baseline BoF on *all* performance measures, achieving a 4.36%, 6.73%, and 19.81% *increase* in accuracy, average precision and F1 score respectively. This demonstrates the MIL-BoF ability to learn more robust action models on the challenging YouTube data. The MIL-BoF approach did not improve the AP compared to the baseline on the **HOHA2** dataset, however, this was made up for by a 0.59% increase in Accuracy and a 7.38% improvement on the F1 score, which weights precision and recall equally. On the **HMDB** dataset, we reported a BoF baseline performance superior to the then state-of-the-art (23.2% [3], 2012), but less than half the accuracy of the current state-of-the-art at 66.79% [127]. Similarly to the Hollywood2 dataset, our MIL-BoF approach outperforms the BoF baseline on the F1 score, in this case by 4.05%. In accord with observations in [15], we achieve good results with subvolume shapes in which there is no temporal subdivision of the sequence [80-80-end], however, we show that a temporal subdivision of the action sequence [80-80-160] can in fact result in a sizable improvement over considering no temporal subdivision at all, as may be seen in the F1-scores of the YOUTUBE [80-80-160] and HOHA2 [60-60-60] dataset.

Our MIL-BoF algorithm is not guaranteed to converge to the optimal solution, and may be one reason why it did not improve over the baseline Accuracy and AP on the HMDB dataset. However, bear in mind that the full χ^2-kernel is calculated for the BoF baseline whilst the linear approximation [126] was used in the MIL-BoF. We expect the results to improve further in the case of full resolution videos. Moreover, due to the large computational cost associated with

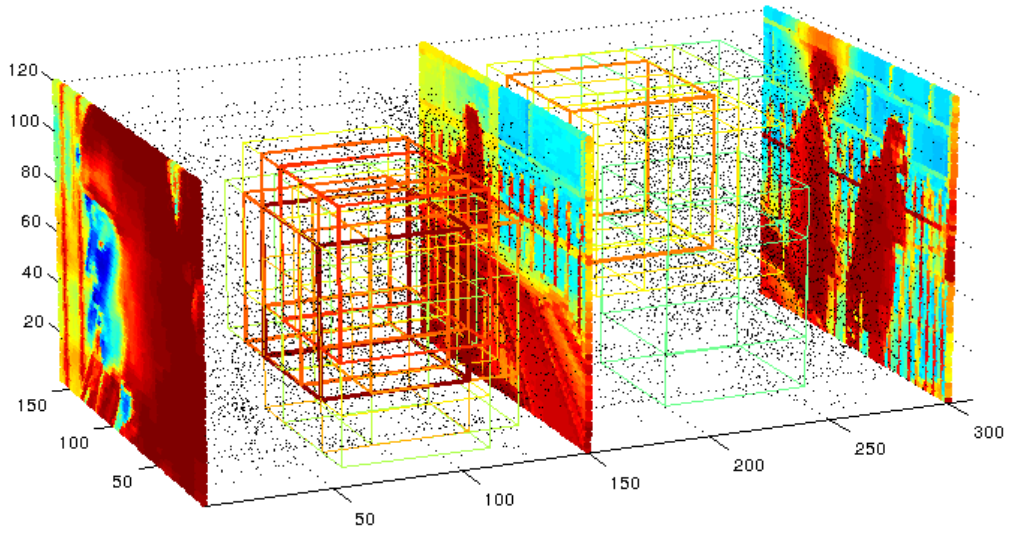
space-time subvolumes, the full potential of our algorithm has yet to be realised, when a more general mixture of subvolume shapes is tailored automatically for each action class. Despite these current setbacks, the MIL-BoF method still outperforms the baseline BoF method in all performance measures on the KTH and YOUTUBE dataset, whilst outperforming the HOHA2 and HMDB on the F1 score, even with fixed-sized subvolumes. Finally, in addition to clip classification, the MIL-BoF method is able to localise challenging actions in space-time, such as the DriveCar and GetOutOfCar actions in the HOHA2 dataset shown in Fig. 4.2(a) & 4.2(b) respectively. The same figures are played as a video in a supplementary multimedia attachment[1].

In the next Chapter, we aim to improve results further by incorporating pictorial structures to better generalise over the variability of space-time actions, and include a larger set of subvolume shapes rather than fixed sized cuboids. We also wish to compare the results we obtain using BoF to another mid-level representation, the Fisher vector.

[1] `https://www.youtube.com/watch?v=VmragBectpo`



(a) actioncliptest00032



(b) actioncliptest00058

Figure 4.2: Action localisation results on two challenging videos from the Hollywood2 dataset, which we encourage the reader to watch in addition to this figure. The colour of each box indicates the positive rank score of the subvolume belonging to a particular class (green-red). (a) Actioncliptest00032 begins with two people chatting in a car. Half-way in, the camera shot changes to a view from the roof of the car. Finally the shot returns to the two people, this time the steering wheel is visible and the driving action is evident. This is reflected by the densely detected, high scoring subvolumes towards the end of actioncliptest00032. (b) In actioncliptest00058, a woman is getting out of her car, however, this action occurs in the middle of the video and not at the beginning or end, as indicated by the detected subvolumes.

Chapter 5

Towards adding local structure and general subvolume shapes

Following from the previous chapter, here we propose to use a pictorial structure model [75, 81] to locate actions in large weakly-labelled datasets. The action recognition system is composed of three main building blocks:

i) learning discriminative local action subvolumes (§ 4.1), where the resulting SVM model $\langle \mathbf{w}_0, b_0 \rangle$ will represent the 'root' filter in our space-time BoF pictorial structure model;

ii) learning and matching of part models extracted from the learnt 'root' subvolumes (§ 5.1), and

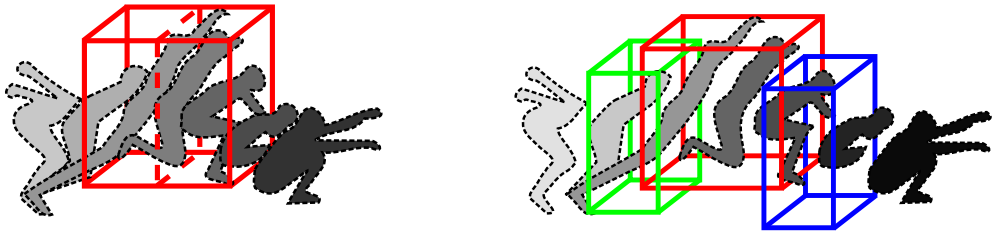
iii) mapping local instance scores appropriately to global video clip scores (§ 4.2).

The methodology in this chapter is presented as an extension to bag-of-features (BoF); however the same methodology extends to other mid-level feature representations, as shown in the experiments (§ 5.2).

5.1 Local Deformable SBoF models (LDSBoF)

In order to learn space-time *part* models, we first select the best scoring root subvolumes learnt via Algorithm 4.1 (Section 4.1). The selection is performed by first pruning overlapping detections with non-maximum suppression in space and time, and then picking the top scoring 5%. Subvolumes are considered to be overlapping if their intersection over the union is greater that 20%. This has the effect of generating a more diverse sample of high scoring root subvolumes to learn the part models from.

The part models are generated by splitting the root subvolumes using a fixed



(a) A training action sequence of class jumping. A discriminative local action subvolume selected via MIL is drawn as a red solid-line cube. The dotted red line denotes the temporal grid into which the root is split in order to learn two separate part models.

(b) A test action sequence of class 'jumping' similar to that in (a) but stretched in time. The detected 'root' subvolume is drawn as a red solid cube, and the parts are shown as green and blue cuboids respectively.

Figure 5.1: Action recognition with a local deformable spatial bag-of-features model (LDSBoF). ***(a)*** *Training 'root' and 'part' action models. The method described in section 4.1 first selects discriminative root subvolumes (red cube). To learn part filters, the 'root' subvolume is divided into a grid of parts; in this case a temporal grid with two parts as denoted by the red dotted line.* ***(b)*** *At test time, the root filter alone (solid red cube) learnt from the action in (a) is not suited to detect the action in (b). However, it is better able to detect this type of action variation with the addition of part filters (solid green and blue cuboids) loosely connected to the root.*

grid, as illustrated in Fig. 5.1(a) & 5.1(b). For our experiments we split the root into $P = 2$ equal-sized blocks along the time dimension (Fig. 5.1(a)), and recalculate BoF vectors for each part. We found that with our current low-level feature sampling density (§ 5.2), subdividing the root to generate more parts creates subvolumes which are too small to aggregate meaningful statistics. Finally, part models $\langle\, \mathbf{w}_{\mathrm{k}}, b_{\mathrm{k}} \,\rangle$, $\mathrm{k} \in \{1, \ldots, P\}$, are individually learnt using a standard linear SVM. The grid structure of SBoF removes the need to learn a structure model for each action class, which simplifies training, especially since no exact or approximate location annotation is available to constrain the part positions [75].

In the following, an action is defined in terms of a collection of space-time *action-parts* in a pictorial structure model [57, 81, 128], as illustrated in Fig. 5.1(b). Let an action be represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertices $\mathcal{V} = \{\mathbf{v}_1, \ldots \mathbf{v}_P\}$ represent the P parts of the action, and $\mathcal{E}$ is a set of edges such that an edge set $\mathcal{E}_{\mathrm{kl}} = \{\mathbf{v}_{\mathrm{k}}, \mathbf{v}_{\mathrm{l}}\}$ represents a connection between part $\mathbf{v}_{\mathrm{k}}$ and $\mathbf{v}_{\mathrm{l}}$. An instance of the action's configuration is defined by a matrix $\boldsymbol{L} = (\mathbf{l}_1, \ldots \mathbf{l}_P)$, where $\mathbf{l}_{\mathrm{k}} \in \mathbb{Z}^{3+}$ specifies the location of part $\mathbf{v}_{\mathrm{k}}$ in a 3-dimensional grid. The detection score volume $s(\mathbf{l})$ is associated with a feature map $\phi(\mathbf{l})$ of BoF histograms for each subvolume at position $\mathbf{l}$, and there exists for

each action part $\mathbf{v}_\mathrm{k}$, a linear BoF filter $\mathbf{w}_\mathrm{k}$, which when dotted with a histogram $\mathbf{h}_\mathrm{k} = \phi(\mathbf{l}_\mathrm{k})$, gives a score indicating the presence of the action part $\mathbf{v}_\mathrm{k}$. Thus, the dot product

$$\mathbf{w}_\mathrm{k} \cdot \phi(\mathbf{l}_\mathrm{k}), \tag{5.1}$$

measures the correlation between a filter $\mathbf{w}_\mathrm{k}$ and a feature map $\phi(\mathbf{l}_\mathrm{k})$ at location $\mathbf{l}_\mathrm{k}$ in the video. Let the distance between action parts $d(\mathbf{l}_\mathrm{k}, \mathbf{l}_\mathrm{l})$ be a cost function measuring the degree of deformation of connected parts from a model. The overall score for an action located at root position $\mathbf{l}_0$ is calculated as:

$$s(\mathbf{l}_0) = \max_{\mathbf{l}_1,\ldots\mathbf{l}_P} \left(\sum_{\mathrm{k}=0}^{P} \mathbf{w}_\mathrm{k} \cdot \phi(\mathbf{l}_\mathrm{k}) - \sum_{\mathrm{k}=1}^{P} d(\mathbf{l}_\mathrm{k}, \mathbf{l}_\mathrm{l}) \right), \tag{5.2}$$

which optimises the appearance and configuration of the action parts simultaneously. The scores defined at each root location may be used to detect multiple actions, or mapped to bag scores in order to estimate the global class label (cf. Section 4.2). We modified the efficient algorithm described in [75] to compute the best locations of the parts as a function of the root locations (5.2), on a sparse grid, the reasons for which are described next.

In practice, we do not calculate an optimal root filter response (Equation 5.2) densely for each pixel, but rather on a subsampled grid. When processing images for the task of 2D object detection, one may pad the empty grid locations with low scores and subsample the distance transform responses [75], since high scores are spread to nearby locations taking into consideration the deformation costs. The computational cost of including the low-score grid locations is small. However with video data, the difference in the number of grid locations for the full and subsampled video is huge. For example, between a 2D image grid of size 640×480, and one half its size (320×240), there is a difference of $\sim 23 \times 10^4$ grid locations. In corresponding videos of sizes $640 \times 480 \times 1000$ frames (approx. 30 seconds) and $320 \times 240 \times 500$, the difference in the number of grid locations is $\sim 26 \times 10^7$. Even though the efficient distance transform algorithm scales linearly with the number of possible locations, padding empty grid locations with low scores becomes computationally expensive. Therefore, we modified the distance transform algorithm to compute the lower envelope of the parabolas bounding the solution [129] at the locations defined by a sparse grid. In this way we achieve the exact same responses with a significant speedup (cf. Section A.7).

5.2 Experimental setup

Bag-of-Features

Dense trajectory features were computed in video blocks of size 32×32 pixels for 15 frames, with a dense sampling step size of 5 pixels, as set by default [25]. Each dense trajectory feature was split into its 5 components (trajectory 30-D, HOG 96-D, HOF 108-D, MBHx 96-D, MBHy 96-D), and for each, a separate K-word visual vocabulary was built by k-means. In order to generate the visual vocabulary, a random and balanced selection of videos from all action classes were sub-sampled, and 10^6 features were again sampled at random from this pool of features. The k-means algorithm was initialised 8-times and the configuration with the lowest error was selected. Lastly, each BoF histogram was L1 normalised separately for each feature component, and then jointly. To speed up the histogram generation we employed a fast kd-tree forest [130, 131] to quantise each dense trajectory feature to its closest cluster centre, delivering a four times speedup when compared to calculating the exact Euclidean distance.

χ^2 kernel approximation

The subvolume settings (§ 5.2) generated ∼3 million instances on the Hollywood2 dataset, each a high dimensional histogram, making the learning problem at hand *large-scale*. Therefore we used an approximate homogeneous kernel map [126, 131] instead of the exact χ^2 kernel, with the same settings as we used in the previous chapter (§ 4.3.2).

Fisher vectors

Excellent classification results have been achieved using Fisher vectors and linear-SVMs [67], which scale much more efficiently with an increasing number of training instances. Due to the high dimensionality of Fisher vectors, each of the 5 dense trajectory feature components were initially reduced to 24 dimensions using PCA [132]. For each feature component, a separate visual vocabulary was built with K-Gaussians each via the Expectation Maximisation (EM) algorithm [133]. The features used to learn the dictionary were sampled in the exact same manner as for BoF (§ 5.2). We follow [67] and applied power normalisation followed by L2 normalisation to each Fisher vector component separately, before normalising them jointly.

Fast linear-SVM solver

In order to quickly learn linear-SVM models we employed the PEGASOS algorithm [134]. This stochastic subgradient descent method for solving SVMs is well suited for learning linear classifiers with large data, since the run-time does not directly depend on the number of instances in the training set. We used the batch formulation with a batch size of 100, and stopped the optimisation after 500 iterations or after reaching a lower tolerance ϵ on the norm of the difference between $\mathbf{w}$-vectors in consecutive iterations ($\epsilon = 10^{-3}$). Stopping the optimisation early results in quicker training and helps generalisation by preventing over-fitting. In order to address class imbalance, we sampled a balanced set of positive and negative examples without re-weighting the objective function [135].

Multiple instance learning

Initially, all the instances in each positive bag were set to have a positive label. At each iteration, the SVM solver was initialised with the model parameters $\langle \mathbf{w}, b \rangle$ calculated in the previous iteration [71], as well as the previous learning iteration number at which $\langle \mathbf{w}, b \rangle$ were calculated. Instead of fixing the SVM regularisation parameters to values known to work well on the test set, we performed 5-fold cross validation [3] on the training set, and automatically select the best performing models based on the validation set *accuracy*. Multi-class classification is performed using the *one-vs-all* approach.

Local Deformable Spatial BoF

In Experiment 1 (cf. Section 5.3.1), the root model subvolumes were set to the same size at the smallest used in the previous chapter (cf. Section 4.3 & Fig. 4.2) ($60 \times 60 \times 60$). This will allow a direct comparison to the results in [48], and the results generated using general subvolume shapes (§ 5.2). The part model subvolumes were set to half the size of the resulting learnt root subvolumes, as shown in Fig. 5.1(a) & Fig. 5.4. We modelled the relative position of each part with respect to the root node's centre of mass as a Gaussian with diagonal covariance [57]:

$$d(\mathbf{l}_\mathrm{k}, \mathbf{l}_\mathrm{l}) = \beta \mathcal{N}(\mathbf{l}_\mathrm{k} - \mathbf{l}_\mathrm{l}, \mathbf{o}_\mathrm{k}, \textstyle\sum_\mathrm{k}) \tag{5.3}$$

where $\mathbf{l}_\mathrm{k} - \mathbf{l}_\mathrm{l}$ represents the distance between part $\mathbf{v}_\mathrm{k}$ and $\mathbf{v}_\mathrm{l}$, $\mathbf{o}_\mathrm{k}$ is the offset and represents the anchor points of each part with respect to the root, and $\sum_\mathrm{k}$ is the diagonal covariance. The parameter β which adjusts the weighting between appearance and configuration scores is set to 0.01 throughout. The offset is taken

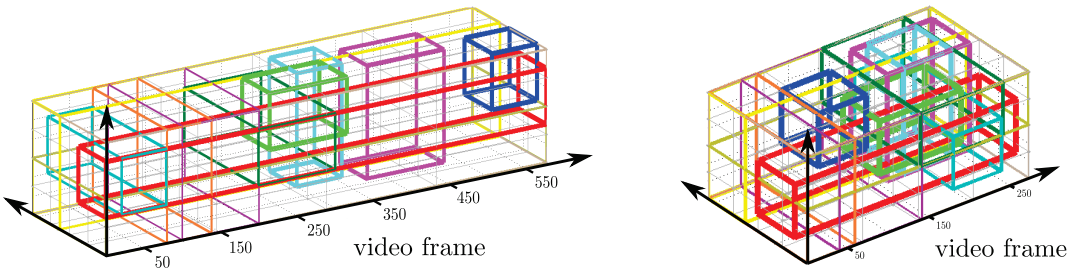


Figure 5.2: Local space-time subvolumes of different sizes are drawn in two videos of varying length at random locations. These subvolumes represent the regions in which local features are aggregated to form a vectorial representation.

automatically from the geometrical configuration resulting from the splitting of the root filter during training, and is set to the difference between the root's and the part's centres of mass. The covariance of each Gaussian is set to half the size of the root filter.

General subvolume shapes

In Experiment 2 (cf. Section 5.3.2), we extended our MIL-BoF approach (§ 4.1) by aggregating features within local subvolumes of various cuboidial sizes scanned densely over a regular grid within the video, as illustrated in Fig. 5.2. In practice there is a huge number of possible subvolume shapes in videos of varying resolution and length in time. Therefore we chose a representative set of 12 subvolume sizes and a grid spacing of 20 pixels in space and time, as a compromise between the higher localisation and classification accuracy obtainable with higher densities, and the computational and storage cost associated with thousands of high dimensional vectors. The subvolumes range from small cubes to larger cuboids, allowing for two scales in width, two scales in height, and 3 scales in time, where the largest scale stretches over the whole video (Fig. 5.2). This setup generated a total of $2 \times 2 \times 3$ subvolume sizes within each space-time volume. The smallest subvolume takes a size of $60 \times 60 \times 60$ pixels in a video of resolution 160×120, and scales accordingly for videos of higher resolutions.

Typical values for the number of subvolumes extracted per video ranged from approximately 300 to 3000, depending on the length of each video. Note that by considering only the subvolume which corresponds to the maximum size, the representation of each video reduces to that of the global pipeline in [25]. Only considering the smallest subvolumes corresponds to the setup used in the previous Chapter 4.

Baseline global algorithm

The baseline approach was set-up by using only the largest subvolume, the one corresponding to the entire video clip. This reduces to the pipeline described in [25], except that in our setup approximate methods are used for histogram building (kd-tree forest [130,131]) and SVM model learning (χ^2-kernel map [126, 131], PEGASOS linear SVM [134]).

5.2.1 Experiments

First, in Experiment 1, we employed the smallest subvolume shape with Fisher vectors, and extended it with a 3-part pictorial structure model (c.f LDSBoF, section 5.1). From this experiment we would like to observe:

- i) the difference in performance between using only the smallest subvolume size [48] and using general subvolume shapes (§ 5.2),
- ii) the merits of adding local deformable structure to mid-level action models,
- iii) the performance of our SVM-map strategy with instance scores generated by the deformable part model compared to taking the label of the maximally scored hypothesis.

The quantitative results for LDSBoF are listed in Table 5.1.

In Experiment 2, we employed our local discriminative part learning (cf. MIL-BoF, section 4.1) with general subvolume shapes but without adding structure, in order to:

- i) determine the local MIL-BoF performance with multiple subvolume shapes with respect to our global baseline (§ 5.2),
- ii) assess how the dimensionality of the instance representation affected performance,
- iii) compare BoF and Fisher representations.

Furthermore, we compared three ways of mapping instance scores to a final bag classification score:

- a) by taking the argument of the maximum value in each bag (max),
- b) by calibrating the instance scores by fitting a sigmoid function to the SVM outputs [124, 125] before taking the max (max-platt),

- c) by using our proposed SVM-map mapping strategy (cf. section 4.2).

We also address questions such as: i) What is the relative difficulty of each dataset? and ii) How important is feature dimensionality for discriminating between more classes? The results are presented in Fig. 5.3 and Table 5.2, where one standard deviation from the mean was reported for those datasets which have more than one train/test split.

5.3 Quantitative results and discussion

5.3.1 Experiment 1 - adding structure

In the first experiment we picked the smallest subvolume size and added the pictorial structures model described in section 5.1 to the local Fisher representations with $K = 32$ Gaussians. The smallest subvolumes were chosen to locate actions at a finer scale (see Fig.5.4). The results obtained using the root node corresponds to 'LDSBoF-1', whilst the results obtained by incorporating parts and structure are denoted as LDSBoF - 3(max) and LDSBoF - 3(SVM-map).

From Table 5.1, it was observed that including parts and a structure model consistently improved accuracy and mF1 measures across all datasets. For example, on the Hollywood2 dataset, between the Fisher *root* model (LDSBoF - 1) and the Fisher *3 - star* model (LDSBoF - 3(max)), we observed a significant 5.7%, 2.44%, and 7.36% improvement in accuracy, mAP and mF1 measures respectively, demonstrating the value of adding deformable parts to the model. The mAP however, which orders the predicted detection scores, dropped considerably, and mAP performance gains were only observed on the Hollywood2 dataset. Finally mapping the instance-scores to bag-scores with our SVM-map strategy produced poor results overall compared to taking the max, indicating that it is not well suited for instance scores produced by deformable part models (Equation 5.2).

5.3.2 Experiment 2 - general subvolume shapes

In the second experiment, we employed discriminative part learning with general subvolume shapes. The results are presented as graphs in Figs 5.3(a)-5.3(d), where for each dataset, the classification accuracies of various approaches (see Fig. 5.3(a)) were plotted for comparison. For each set of experiments, the mid-level feature dimensionality was varied by controlling the K-centroids used to

Table 5.1: A table showing the results for using our local deformable spatial bag-of-features (LDSBoF) with Fisher vectors generated using $K = 32$ Gaussians.

KTH	Acc	mAP	mF1
LDSBoF-1	94.44	97.65	94.45
LDSBoF-3(max)	95.83	96.97	95.84
LDSBoF-3(SVM-map)	96.76	95.27	96.76
YOUTUBE	Acc	mAP	mF1
LDSBoF-1	73.02±8.50	83.15±6.83	70.35±9.35
LDSBoF-3(max)	80.06±7.28	75.97±7.84	77.06±8.82
LDSBoF-3(SVM-map)	68.04±9.26	77.06±6.12	67.18±9.26
Hollywood2	Acc	mAP	mF1
LDSBoF-1	51.36	43.75	36.73
LDSBoF-3(max)	57.01	46.19	44.09
LDSBoF-3(SVM-map)	50.22	43.49	48.19
HMDB51	Acc	mAP	mF1
LDSBoF-1	25.49±0.28	28.54±0.56	23.62±0.61
LDSBoF-3(max)	31.09±0.53	12.67±0.50	29.46±0.63
LDSBoF-3(SVM-map)	14.60±0.34	14.82±0.18	14.43±0.71

build each visual vocabulary. The dimensions of KBoF (approximate kernel-mapped BoF) vectors were calculated as: K (centroids) $\times 5$ (feature components) $\times 3$ (kernel-map). The dimensions of Fisher vectors were calculated as: 2 (Fisher vectors [67]) $\times K$ (centroids) $\times 5$ (feature components) $\times 24$ (dimensions per feature component after PCA).

From the result plots of Fig. 5.3, a number of interesting observations emerged. First, even with a 16 fold reduction in K and without a kernel feature mapping, Fisher vectors outperformed KBoF, both in global and local representations. It is even quite surprising that with only 2 clusters per dense trajectory feature component, Fisher vectors achieved over 90% accuracy on the KTH dataset, a sign of the dataset's ease. The relative difficulty of each dataset is related to the number of categories in each dataset; the classification accuracy chance levels are: $\frac{1}{6}, \frac{1}{11}, \frac{1}{12}, \frac{1}{51}$ for Figs 5.3(a)-(d) respectively. However, this does not take into consideration the noise in the dataset labelling, or the choice of train-test splittings. For example, the YouTube and Hollywood2 datasets both have a similar chance level, however the lower accuracies of Fig. 5.3(c) as compared to Fig. 5.3(b) demonstrate the increased difficulty posed by Hollywood2.

Notice that the Hollywood2 and HMDB datasets showed a steady increase in accuracy with increasing K, which shows the benefit of learning with more model

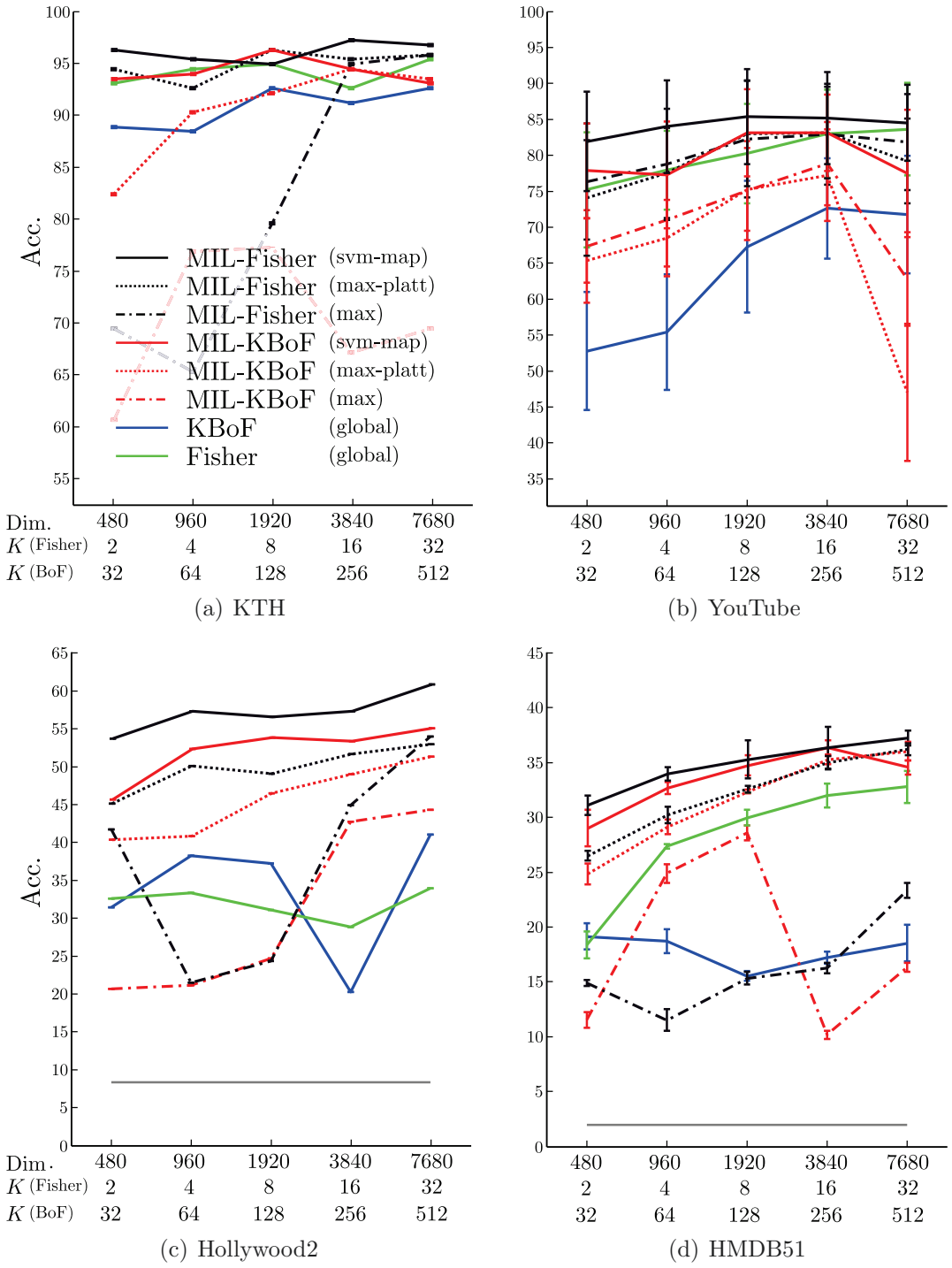


Figure 5.3: Quantitative graphs for learning local discriminative subvolume models via multiple-instance learning. Here we plotted the accuracy against the mid-level feature dimensionality, and compare i) our local MIL approach (red & black) vs. the global baseline (blue & green), ii) the performance of kernel-BoF and Fisher vectors, and iii) three instance to bag mapping strategies, namely: taking the argument of the max instance, max after Platt calibration, and our SVM-map instance to bag mapping technique. The chance level is plotted as a grey horizontal line.

parameters and higher dimensional vectors on challenging datasets. However this trend was not always observed with the KTH and YouTube datasets, a sign that cross-validation over K may improve results.

The high *variation* in accuracy obtained when taking the max (red/black dash-dotted lines), indicated that the SVM models learnt in a *one-vs-all* manner often produced incomparable scores. This was demonstrated by the boost in accuracy often observed after the scores were Platt-calibrated [124] (red/black dotted lines). Finally, further improvement was offered by mapping the instance scores directly to bag scores using our SVM-map approach (red/black solid lines). For example on the Hollywood2 dataset (Fig. 5.3(c)), MIL-Fisher (SVM-map) with $K = 32$ achieved an 8% boost compared to the respective Platt-calibrated max.

The aforementioned observations also held on HMDB, the most challenging dataset considered here with a chance level of just under 2%. It may be seen that the accuracy of global Fisher vectors outperformed that of KBoF, despite having a smaller number of K centroids. Again, Platt-calibration greatly improved the results for taking the argument of the maximum instance in each bag, however our SVM-map strategy gained further, with the local Fisher method coming out on top.

The quantitative results obtained for the Fisher vectors with $K = 32$ centroids per feature component are listed in Table 5.2. Comparing the results of MIL-F32(max) to LDSBoF - 1 demonstrates the improvements in accuracy obtained by using a general mixture of subvolumes as opposed to a single size [48].

Even though the *KTH* dataset is arguably the easiest dataset considered in this work, with already near-saturated results, one can still observe minor improvements when comparing local models to the global baseline. On the *YouTube* dataset however, our global baseline outperformed the current state-of-the-art on mAP, whilst our local MIL-F32 setup outperformed the state-of-the-art on accuracy and mF1 measures.

Performance gains over the global baseline were observed across *all* performance measures on the Hollywood2 dataset. Note that the lower mAP as compared to the state-of-the-art may be a result of using half-resolution videos and approximate methods for histogram building and learning action models, in order to cope with the large instance training sets (§ 5.2). Finally the *HMDB* is the most challenging action classification dataset, and we report a 4.4%, 8.7% and 7.5% increase in accuracy, mAP and mF1 measures when compared to our global baseline.

Table 5.2: A table showing the state-of-the-art results, and our results using Fisher vectors with $K = 32$ Gaussians (F32).

KTH	Acc	mAP	mF1
State-of-the-art	**96.76** [48]	97.02 [48]	96.04 [48]
Global-F32	95.37	96.81	95.37
MIL-F32(max)	95.83	97.43	95.84
MIL-F32(max-platt)	95.83	97.43	95.82
MIL-F32(SVM-map)	**96.76**	**97.88**	**96.73**
YOUTUBE	Acc	mAP	mF1
State-of-the-art	**93.77** [127]	86.10 [48]	77.35 [48]
Global-F32	83.64±6.43	**87.18±3.58**	80.41±7.90
MIL-F32(max)	81.84±6.68	86.53±4.65	78.59±8.31
MIL-F32(max-platt)	79.22±5.88	86.53±4.65	74.35±7.56
MIL-F32(SVM-map)	84.52±5.27	86.73±5.43	**82.43±6.33**
Hollywood2	Acc	mAP	mF1
State-of-the-art	39.63 [48]	**63.3** [19]	39.42 [48]
Global-F32	33.94	40.42	12.18
MIL-F32(max)	53.96	49.25	39.11
MIL-F32(max-platt)	52.94	49.25	36.34
MIL-F32(SVM-map)	**60.85**	**51.72**	**52.03**
HMDB51	Acc	mAP	mF1
State-of-the-art	**66.79** [127]	**40.7** [22]	25.41 [48]
Global-F32	32.79±1.46	30.98±0.69	30.62±1.19
MIL-F32(max)	23.33±0.66	35.87±0.56	16.68±0.40
MIL-F32(max-platt)	36.19±0.56	35.88±0.56	32.86±0.34
MIL-F32(SVM-map)	**37.21±0.69**	**39.69±0.47**	**38.14±0.76**

5.4 Computational timings

The experiments were carried out on a machine with 8, 2GHz CPUs and 32GB of RAM. The timings we report here are for running the method 'MIL-F32' on the HMDB dataset. Building the visual vocabulary took 4 CPU hours, whilst the local aggregation of dense trajectory features in to Fisher vectors took 13 CPU days. Finally, 5 CPU days were needed for learning local discriminative subvolumes via MIL. Since the implementation is non optimised in MATLAB, we are convinced that the computational timings can be cut considerably. Note that the low-level feature extraction, the local aggregation, and *one-vs-all* classification are easily parallelised.

5.5 Qualitative localisation discussion

In addition to action clip classification, the learnt local instance models can also be used for action localisation, the results of which are discussed in the next sections.

5.5.1 Bounding box detection with LDSBoF

The LDSBoF approach is able to predict the location of discriminative action parts by selecting the top scoring 'root' and 'part' configurations in each video clip. This is clearly seen in Fig. 5.4, where the front, top and side views of a boxing video sequence from the KTH dataset are plotted in space-time. Notice that the parts correspond to a spatial subdivision of the root subvolume, and have connections to the root that are deformable in both space and time.

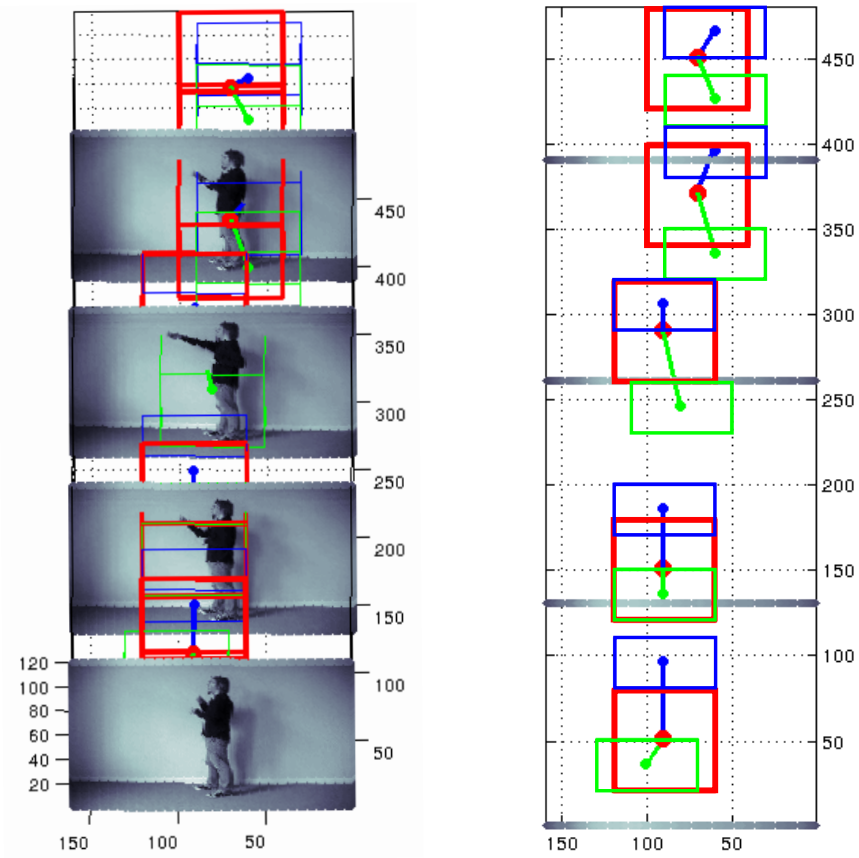
The space-time plots of Figs. 5.5(a)-(c) show qualitative results obtained when detecting actions with LDSBoF models on real-world movie sequences from the Hollywood2 dataset. Due to the large variation in video clip length, we picked the top 5% detections, which do indeed correspond to discriminative parts of the action. A supplementary video illustrating additional LDSBoF results is available on the Internet[1].

5.5.2 Class-specific saliency

In contrast to object detection, in which the boundaries of an object are well defined in space, the ambiguity and inherent variability of actions means that not all actions are well captured by bounding boxes. As an alternative we propose to use the detection scores as a measure of action location and saliency, as shown in Fig. 5.6.

Recall that at test-time each subvolume is associated with a vector of scores denoting the response for each action category in that region. Since in our framework, we also map the individual instance scores to a global video clip classification score (§ 4.2), the vector of scores associated with each subvolume can be reduced to that of the predicted video action class. It is therefore possible to build an action-specific saliency map by aggregating the predicted detection scores from all instances within the space-time video. In both Fig. 5.6(a) & (b), the saliency map is specific to the predicted global action class, and is displayed as a sparse set of points for clarity. The sparsity and colour (from blue to red) of

[1] `https://www.youtube.com/watch?v=jtLfUfBVi5k`



(a) Front view

(b) Top view

(c) Side view

Figure 5.4: (a) Top, and (b) side views from a test boxing video sequence in the KTH dataset. Plotted are the top 5 best part configurations found in the video volume. The simple tree-structured star models are drawn with blue and green links to the root node.

the plotted points indicate the action class membership strength. Moreover, the saliency map indicates the action location in both space *and* time. In both videos of Fig. 5.6, irrelevant scene background, common in most of the KTH classes, was pruned by the MIL during training, and therefore the learnt action models were able to better detect relevant action instances. Notice that the saliency map was able to capture both the consistent motion of the boxing action in

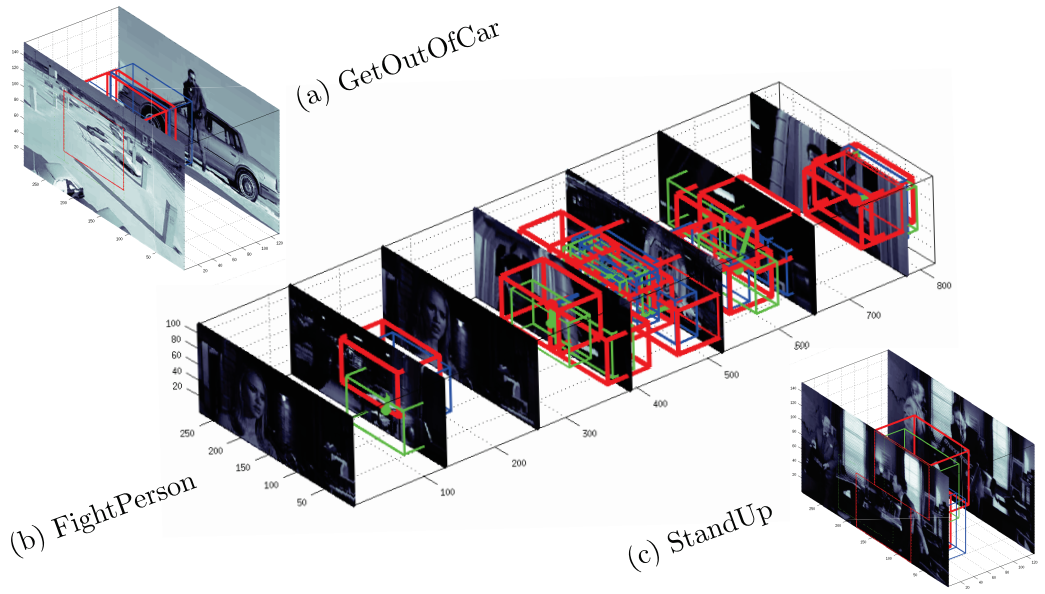


Figure 5.5: Detected LDSBoF configurations in the challenging Hollywood2 dataset. The three sequences from the Hollywood2 dataset show the detections for the videos classified as, (a) GetOutOfCar, (b) FightPerson, and (c) StandUp. It is seen that in addition to class labels, each action in the video is localised via a 3-part deformable star model.

Fig. 5.6(a), as well as the intermittent walking action of Fig. 5.6(b).

The qualitative localisation results for the *HMDB* dataset are shown in Fig. 5.7 and Fig. 5.8, where the saliency maps drawn over the videos are those corresponding to the predicted global action class. Note that such a saliency map does not represent a general measure of saliency [23], rather it is *specific* to the particular action class being considered. This has the effect of highlighting discriminative parts of the action, for example, in the pushing action of Fig. 5.7(e), the contact between the hands and vehicle is highlighted, less so the leg's motion. Likewise in Fig. 5.7(d) the pouring action is highlighted, less so the arm's motion.

The videos of Fig. 5.8 show the results obtained when the clip is incorrectly classified. In this case, higher scores are located at the borders of the video frame, since the instance scores for the wrong predicted class are low over the video parts where the action occurs. In the HMDB dataset, apart from dealing with a wide range of action classes, our algorithm has to deal with significant nuisance factors in the data, such as frames of 'Britney Spears' singing in between a 'kick_ball' action (Fig. 5.8a), and the possibility of multiple actions per video clip such as the 'walking' and 'waving' actions in Fig. 5.8(f).

In this chapter, as well as the previous one, we demonstrated the value of

leveraging large weakly labelled datasets to locate the discriminative parts of a video. This entailed breaking up each video into small overlapping blocks, making the learning problem *large-scale*. Due to the large amount of data, approximate algorithms were used to encode mid-level features and learn action models. In the next chapter, we focus on the *global* representation of action videos, which allows more compact datasets for training. This means we can afford to use exact euclidean distances for quantising features, and a kernel-SVM solver. We will also be able to run many more experiments and evaluate the merits of various strategies to building a bag-of-visual-words vocabulary.

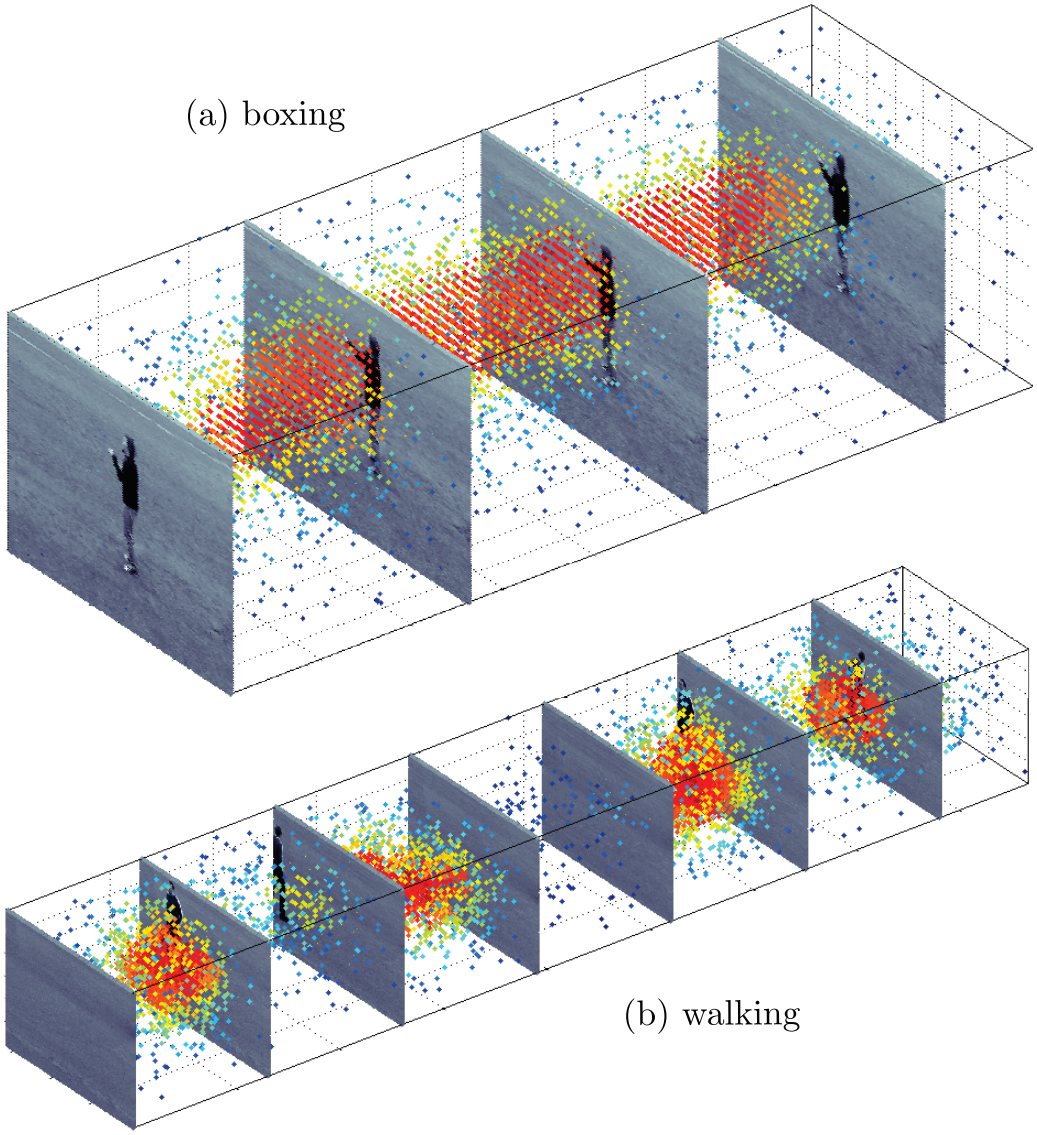


Figure 5.6: Action classification and localisation on the KTH dataset. ***(a)*** *This boxing video sequence has been correctly classified, and has overlaid the boxing action saliency map to indicate the location of discriminative action parts.* ***(b)*** *Walking action classification in an 850-frame video sequence. The actor walks in and out of the camera shot, as shown by the dense red points over the locations in which the actor was present in the shot.*

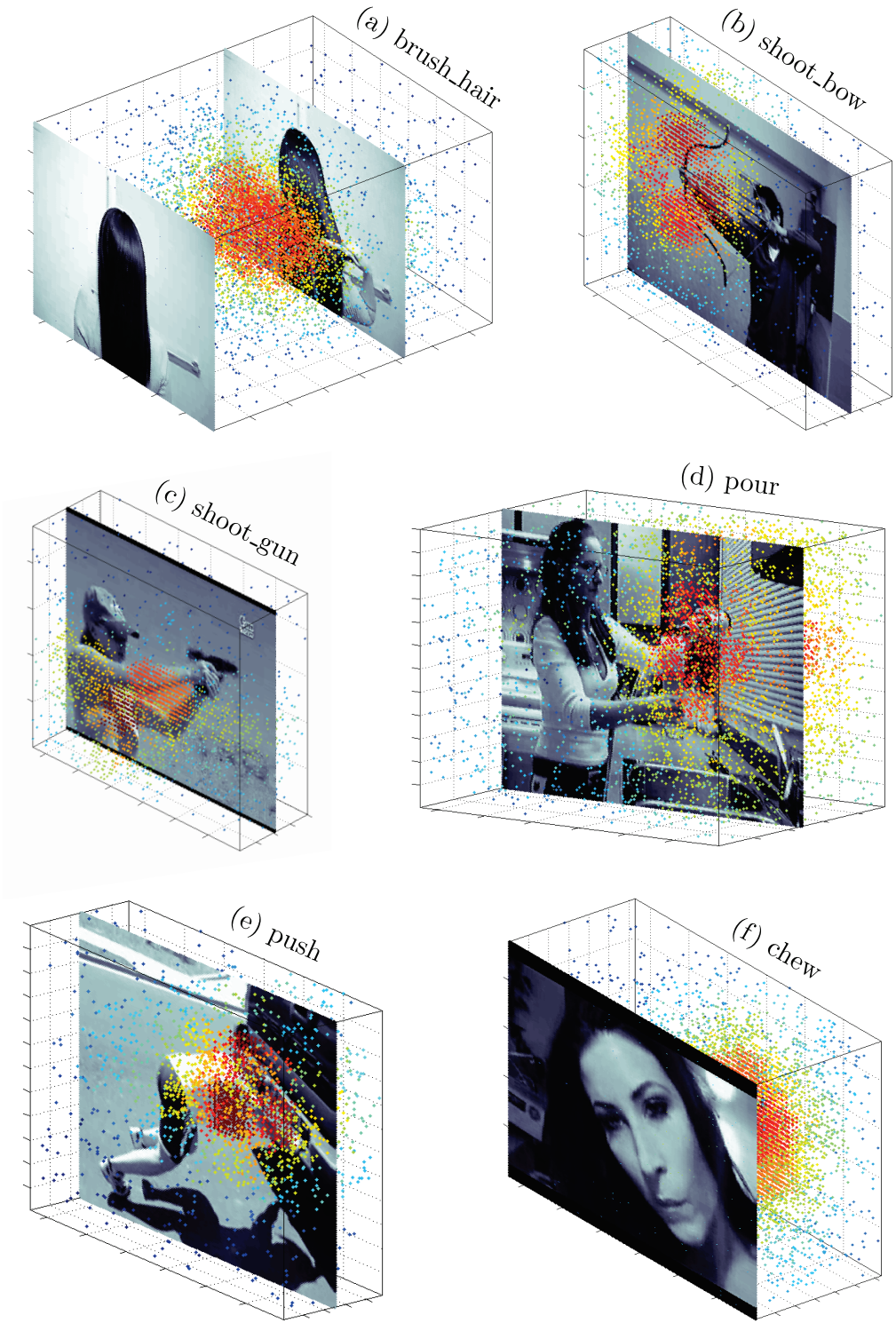


Figure 5.7: Action localisation results on the HMDB51 dataset, the most challenging action classification dataset to date. ***(a)*** *Brush hair video clip. Notice that the saliency map in* ***(b)*** *is focused on the bow and not on the person's elbow movement.* ***(c)*** *Shoot gun action. In* ***(d)****, a girl is pouring liquid into a glass, and in* ***(e)*** *a person is pushing a car. The push action does not fire over the person's moving legs but rather on the contact zone between the person and the vehicle.* ***(f)*** *Chew action.*

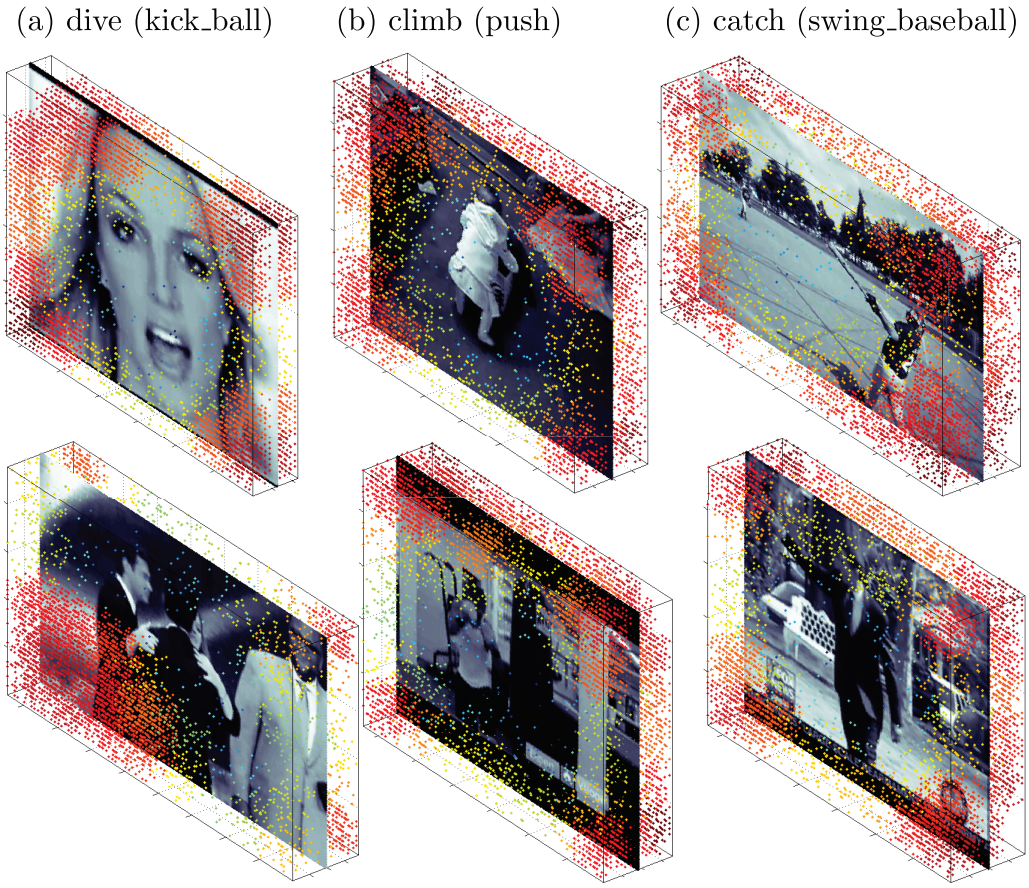


Figure 5.8: Misclassifications in the HMDB51 dataset. The predicted class is shown first, followed by the ground truth class in brackets. Note that the localisation scores shown are of the predicted class. ***(a)*** *A 'kick ball' action that has frames of Britney Spears singing in the middle.* ***(b)*** *A 'push' action wrongly classified as 'climb'.* ***(c)*** *The fast swinging movement of the 'baseball' action was classified as 'catch'.* ***(d)*** *A hug was incorrectly classified as punch, and* ***(e)*** *a 'punch' was misclassified as 'clap'.* ***(f)*** *In this case the wave action was misclassified as walk, even though President Obama was also walking. The algorithm was unable to cope with the situation in which two actions occur simultaneously (best viewed in colour).*

Chapter 6

Feature sampling and partitioning for visual vocabulary generation on large action classification datasets

In this chapter we consider global action classification in which there is:

- a huge number of features per dataset;
- an imbalance in the number of samples per class;
- an imbalance in the number of features per video;
- and with several feature components per feature descriptor.

Therefore we detail experiments to answer the following questions set out in section 1.5.3, namely:

i) What is the best way to randomly subsample features to build a vocabulary for human action recognition?
ii) What are the effects of learning separate rather than joint visual vocabularies when considering multiple feature components or multiple action classes?

We keep some parameter settings constant throughout the tests (§ 6.1) and let others vary (§ 6.2). The specifics of our vocabulary sampling and generation strategy, designed to be robust to an imbalance in the number of videos per category, and to an imbalance in the number of features per video, are laid out in Algorithm 6.1.

6.1 Constant experimental settings

Space-time features

Amongst the wide variety of space-time interest point detectors [53, 55] and descriptors [15, 60–62], the dense trajectory features of Wang *et al.* [25] gave the state-of-the-art results in experiments with challenging action video data [25, 48] (cf. Section 2.2). These dense trajectory features were pre-computed from video blocks of size 32×32 pixels for 15 frames, and a dense sampling step-size of 5 pixels, using the code made available by the authors [25]. The features were stored as groups of 4-byte floating point numbers, each associated with a video 'i' in dataset d $\in \mathcal{D}$, where i $\in \{1, ..., N\}$. With our settings the size of each feature S_{GB} was $\big((30_{traj} + 96_{HoG} + 108_{HoF} + 96_{MBHx} + 96_{MBHy}) \times 4\big)/1024^3\ GB$, where the summed integers denote the dimensionality of each feature component. The approximate number of dense trajectory features extracted per dataset and per video is presented in Table 2.1, together with further dataset specific statistics.

Visual vocabulary

The k-means algorithm was used to generate the visual vocabulary for BoF and Vectors of Locally Aggregated Descriptors (VLAD) [68, 136]. The greedy nature of k-means means that it may only converge to a local optimum. Therefore the algorithm was repeated 8-times to retain the clustering with the lowest error. The number of feature samples used to learn the vocabulary was set to a maximum of $10,000$ per cluster. We found that setting the pool of feature size to depend on the number of clusters allows faster vocabulary computation for small K without loss of performance. The maximum number of features was set to a value of 1×10^6 in order to limit memory usage and computational time for larger K.

Hardware constraints

With dense trajectory features [25], 1×10^6 features translate to $\sim$ 1.6GB of memory when using 32-bit floating point numbers, and may thus easily be worked with on a modern PC or laptop. All our experiments were carried out on an 8-core, 2GHz, 32GB workstation.

Bag-of-features

In the BoF approach, histograms are computed by quantising features to the nearest cluster (minimum euclidean distance). Each BoF histogram was $L1$ normalised separately for each feature component and then jointly [20]. The same setup was used for BoF per-category, except that separate vocabularies were generated by clustering the features from each class and then joining them into one universal vocabulary. For both BoF approaches the exponentiated χ^2-kernel SVM [15, 26, 137] was used:

$$\phi_{\chi^2}(\mathbf{h}_\mathrm{i}, \mathbf{h}_\mathrm{j}) = \exp\left(-\frac{1}{2A}\sum_{\mathrm{k}=1}^{\mathrm{K}}\frac{(h_\mathrm{ik} - h_\mathrm{jk})^2}{h_\mathrm{ik} + h_\mathrm{jk}}\right), \tag{6.1}$$

where $\mathbf{h}_\mathrm{i}$ and $\mathbf{h}_\mathrm{j}$ are the histograms associated by videos i and j, K is the number of cluster centres, and A is the mean value of all distances between training histograms, and h_k is the k^{th} element of the histogram vector $\mathbf{h}$.

VLAD and Fisher vectors

Initially, due to the high dimensionality of both VLAD and Fisher vectors (cf. Section 2.3), the dense trajectory feature components were independently reduced to 24 dimensions using PCA. We used the randomised PCA algorithm by Rokhlin *et al.* [132] to solve the USV' decomposition on the same 1×10^6 subsampled features used for dictionary learning.

Both VLAD and Fisher vectors were computed using the implementations available in the VLFeat library [131]. We followed Perronnin *et al.* and apply power normalisation followed by L2 normalisation to each Fisher vector component separately [67], before normalising them jointly. Contrarily to expectations, it has been shown that Fisher vectors achieve top results with *linear* SVM classifiers [67, 138].

Support vector machines

For both the χ^2-kernel and linear SVMs, multi-class classification was performed using a 1-vs-all approach [139], and we keep the regularisation cost C=100 constant throughout [26, 62].

6.2 Variable components of the experiments

Test 1 - comparison of uniform random sampling strategies

We compared two methods of sampling descriptors in order to learn a visual vocabulary. Note that the sample forms the data on which the k-means algorithm will operate, and therefore if important but rare features are missed out due to an imbalance in the data, it may lead to a suboptimal visual vocabulary. Method (**1a**) sampled a balanced random set of features from the dataset, accounting for an unbalanced number of videos per action category, and an unbalanced number of features per video (see Table 2.1). A uniform random selection of features *without* balancing was performed for method (**1b**). The exact sampling strategy is detailed in Algorithm 6.1, whilst the comparative performance will be discussed in section 6.3.

Test 2 - comparison of joint and component-wise visual vocabulary generation

Here we learnt a visual vocabulary separately for each feature component (**2a**) such as the Trajectories, HoG, HoF, MBHx and MBHy components, and compared this to grouping these features together and learning a joint vocabulary (**2b**).

Test 3 - comparison of global representations

For each experimental setting, we assessed the performance of the following four global representations: standard BoF (**3a**), BoF per-category (**3b**), VLAD (**3c**) and Fisher vectors (**3d**). Each variation was run for 7 values of K cluster centres, ranging from 2^2 to 2^8 in integer powers of 2. The code to reproduce our experiments and results is available online[1]

6.3 Results and discussion

Our experiments consisted of 2 binary tests (**1a-b, 2a-b**) for 7 values of K cluster centres, 4 representation types (**3a-d**) and 4 datasets, for a total of $2^2 \times 7 \times 4 \times 4 = 112 \times 4 = 448$ experimental runs, the results of which are shown in Figs. 6.1-6.4. The best quantitative results were further listed in Table 6.1

[1] `https://sites.google.com/site/mikesapi/downloads/global-video-representation`

Algorithm 6.1 The sequence of steps used for generating a visual vocabulary for action classification datasets. Note that our proposed method **(1a)** is designed to be robust to an imbalance in the number of videos per category, and to an imbalance in the number of features per video.

1. Extract dense trajectory features [25] from videos and store them on disk;
2. For each dataset $\mathrm{d} \in \mathcal{D}$, calculate the mean number of features $\mu_\mathrm{d} = \frac{1}{N}\sum_{\mathrm{i}=1}^{N} n_\mathrm{i}$, where n_i is the total number of features extracted from clip i, and $\mathrm{i} \in \mathcal{I} = \{1, \ldots, N\}$;
3. Set the maximum number of videos to sample from V_{max} based on the mean number of features per video: $V_{max} = \lfloor \frac{M_{GB}}{\mu_\mathrm{d} \times S_{GB}} \rfloor$, where M_{GB} is the maximum memory available, and S_{GB} is the memory needed to store a single feature;
4. Subsample $\lfloor \frac{V_{max}}{C_\mathrm{d}} \rfloor$ videos uniformly at random from each class (**1a**, class balancing), or (**1b**) V_{max} videos uniformly at random from the entire set of videos, where C_d denotes the number of action categories in dataset d;
5. Load features into memory by uniformly subsampling $\min(n_\mathrm{l}, \mu_\mathrm{d})$ features per video (**1a**), or by loading all the features n in each video (**1b**), where $\mathrm{l} \in \mathcal{L} \subseteq \mathcal{I}$ denotes the set of subsampled video indices;
6. Subsample $\min(1 \times 10^6, K \times 10^4)$ features (**1a**) at random, with a balanced number from each class, or (**1b**) at random from the entire set;
7. Perform k-means with K cluster centres separately per feature component (**2a**) or jointly by concatenating the feature components (**2b**);
8. Represent videos globally via BoF (**3a**), BoF per-category (**3b**), VLAD vectors (**3c**), or Fisher vectors (**3d**).

in order to compare them to the current state-of-the-art. For each dataset, we presented:

i) the classification accuracy plotted against the number of clusters K,

ii) the accuracy plotted against the final representation dimensionality D.

Note that the horizontal axis marks for Figs. 6.1-6.4 are evenly spaced, thus encoding only the ordering of K and D. The computational time needed to generate the results was approximately 130 CPU hours for KTH, 1290 CPU hours for YouTube, 1164.8 CPU hours for Hollywood2, and 974 CPU hours on the HMDB dataset.

The results for KTH are shown in Fig. 6.1. The accuracy quickly shot up with an increasing K and associated dimensionality D and saturates at around

95% accuracy. By looking at the curves of Fig. 6.1, one may identify the best performing method under a value of K and representation size D. For example the dotted line in Fig. 6.1 highlights the methods at $D = 192$, where several test configurations span an accuracy gap of approximately 12%. In this case, the BoF per-category (**3b**) at $D = 192$ (32 clusters $\times 6$ classes) surpassed the current state-of-the-art with 97.69% accuracy, 98.08% mAP, and 97.68% mF1 score. Note that the histogram was formed by sampling the features uniformly at random (**1b**) and clustering the features jointly (**1b**).

The results for the YouTube dataset did not follow the same pattern as of KTH, as seen in Fig. 6.2. The accuracy peaked and then fell for the BoF and BoF per-category, which indicated that the higher dimensional representations combined with the χ^2-kernel SVM lead to overfitting. This was not so for Fisher vectors and linear SVMs, which outperformed all the BoF results with only 4 cluster centres per feature component (see dotted line in Fig. 6.2-right).

Here one can clearly see the benefit of using separate vocabularies per feature type (see Fisher joint (**2b**) vs. Fisher separate (**2a**)), and the advantage of random balanced sampling as the number of classes and videos has increased compared to KTH. Comparing the results obtained as a function of K, on the YouTube dataset, Fisher vectors (**3d**) outperformed the other representation types 96.43% of the time, generating a visual vocabulary separately per feature was best for 85 .71% of the trials, whilst using a random and balanced feature selection was best 66% of the time.

The Hollywood2 results of Fig. 6.3 (bottom) highlight the initially increased performance obtained using BoF at low values of K, which were eventually overtaken by the high dimensional Fisher and VLAD vectors with a linear SVM at the top of the curve. In fact, at dotted line (1), BoF per-category with joint features and balanced sampling (3b-2b-1a at $D = 48$) achieves 5% higher accuracy than the equivalent Fisher vector at $D = 192$. Even though at the top of the curve Fisher vectors achieve the highest accuracy, note that BoF per-category is very close behind with a separate (**2a**) and balanced (**1a**) vocabulary (purple circle).

The Hollywood2 dataset has the largest variance in the number of features per video clip (see Table 2.1), making it more susceptible to randomly selecting an imbalanced feature set for k-means. For example, the outlier for VLAD at $D = 192$ in Fig. 6.3 (right, dotted line 2) may be caused by the biased feature sampling of method **(1b)**, which has the undesirable effect of excluding discriminative features from the learned vocabulary, and over-representing features which

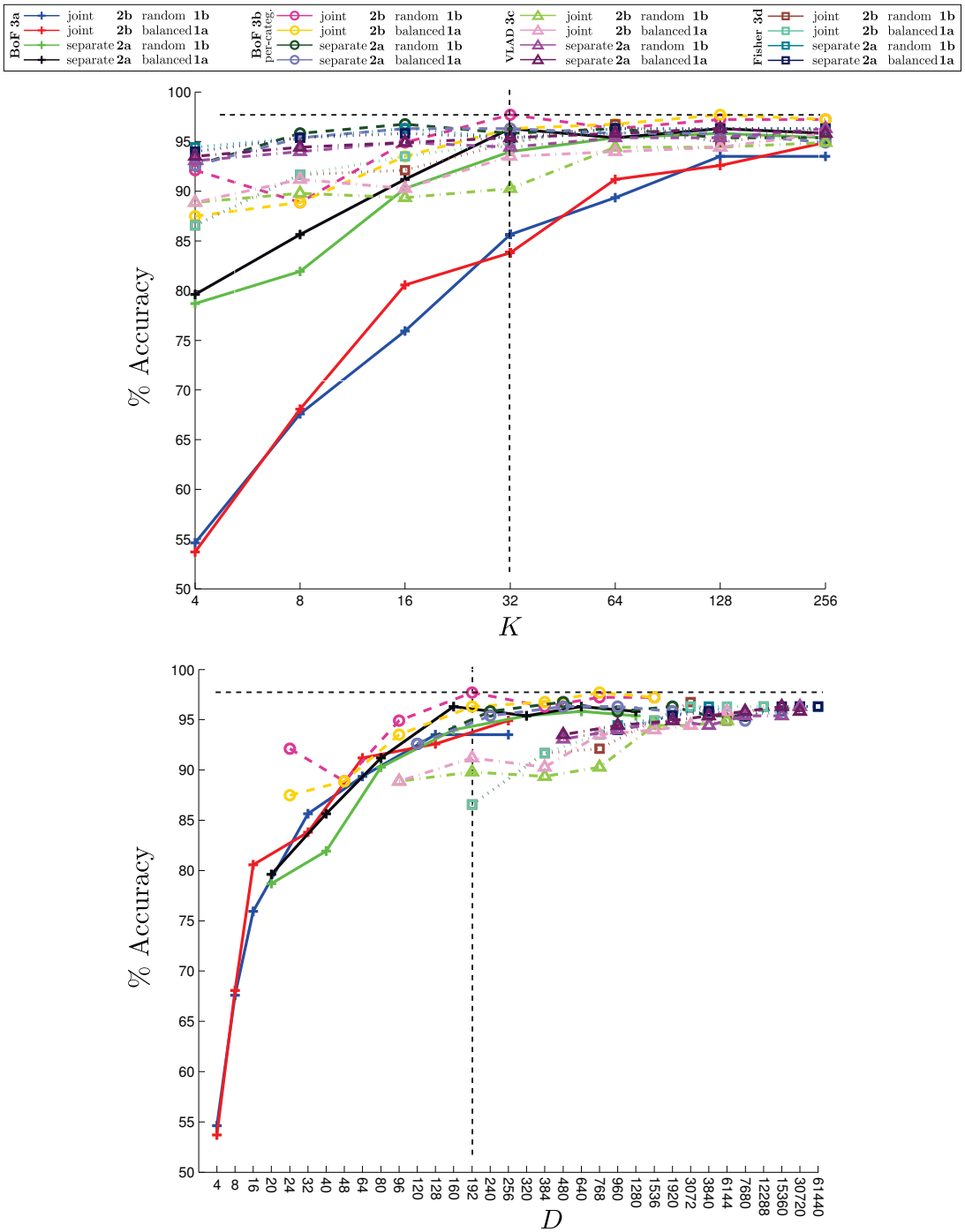


*Figure 6.1: KTH dataset classification accuracy using a global video representation. (top) The Accuracy is plotted against the number of cluster centres K, and (bottom) against the representation dimensionality. Note that with a small number of cluster centres (top), high dimensional representations may be achieved (bottom), for example when generating vocabularies separately per feature component or per action class. In the rightmost plot, at a relatively low representation dimensionality $D = 192$, there are several competing representations within a ∼12% range in accuracy. The best representation method, however, on this dataset is BoF per-category **(3b)** with a single joint visual vocabulary **(2b)** and features sampled at random **(1b)**.*

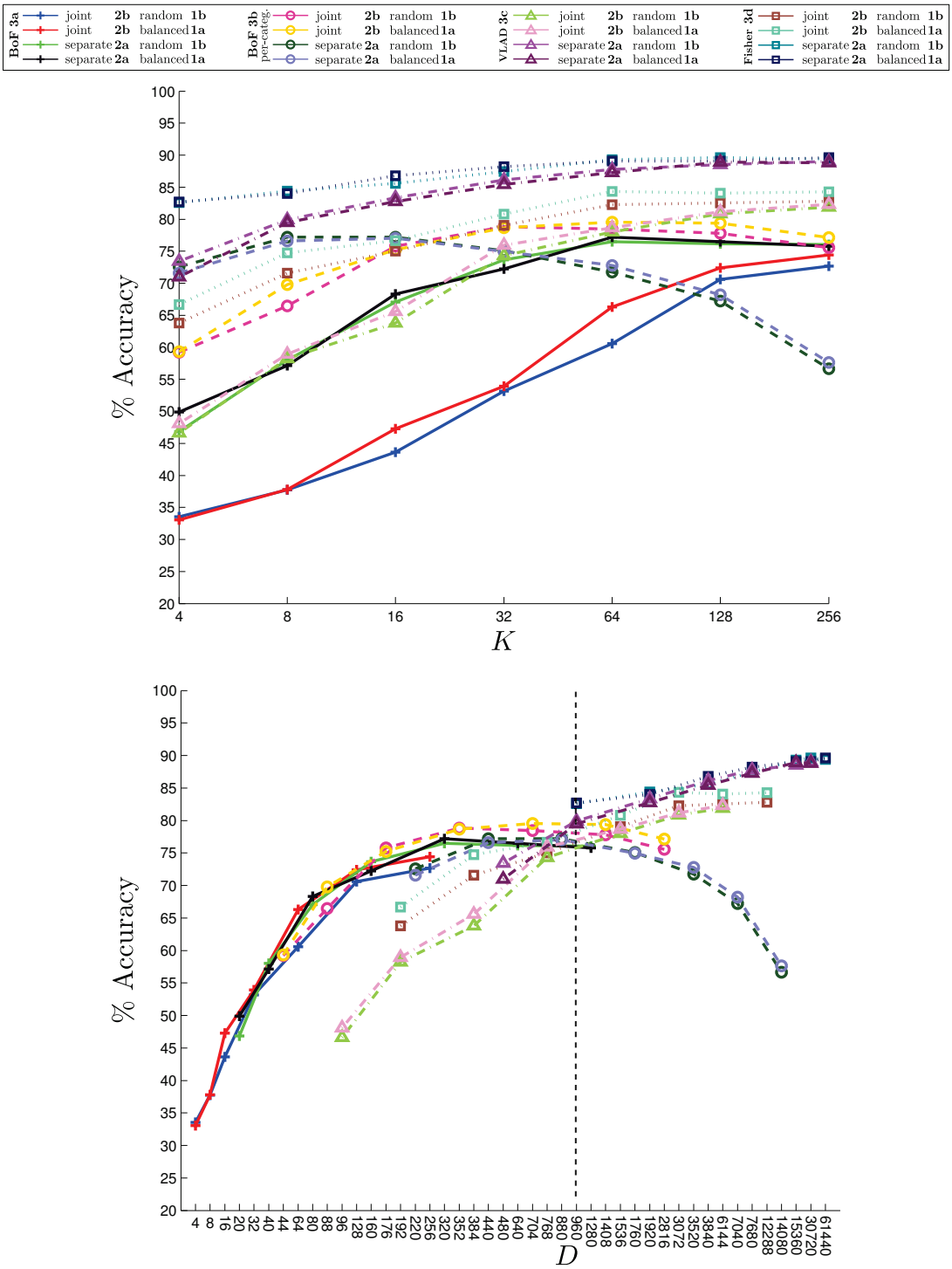


Figure 6.2: YouTube dataset classification results. (top) Accuracy vs. K cluster centres, and (bottom) accuracy vs. representation dimensionality. Notice (bottom) the inverted 'U' paths traced by the χ^2-kernel SVM (BoF and BoF per category) as opposed to those generated by the linear SVM (VLAD and Fisher vectors). The dotted line indicates the group of data points at $D = 960$ and highlights the top performing method, which is Fisher vectors with only 4 clusters per feature component. The error bars over 25 cross-validation folds have been suppressed for clarity.

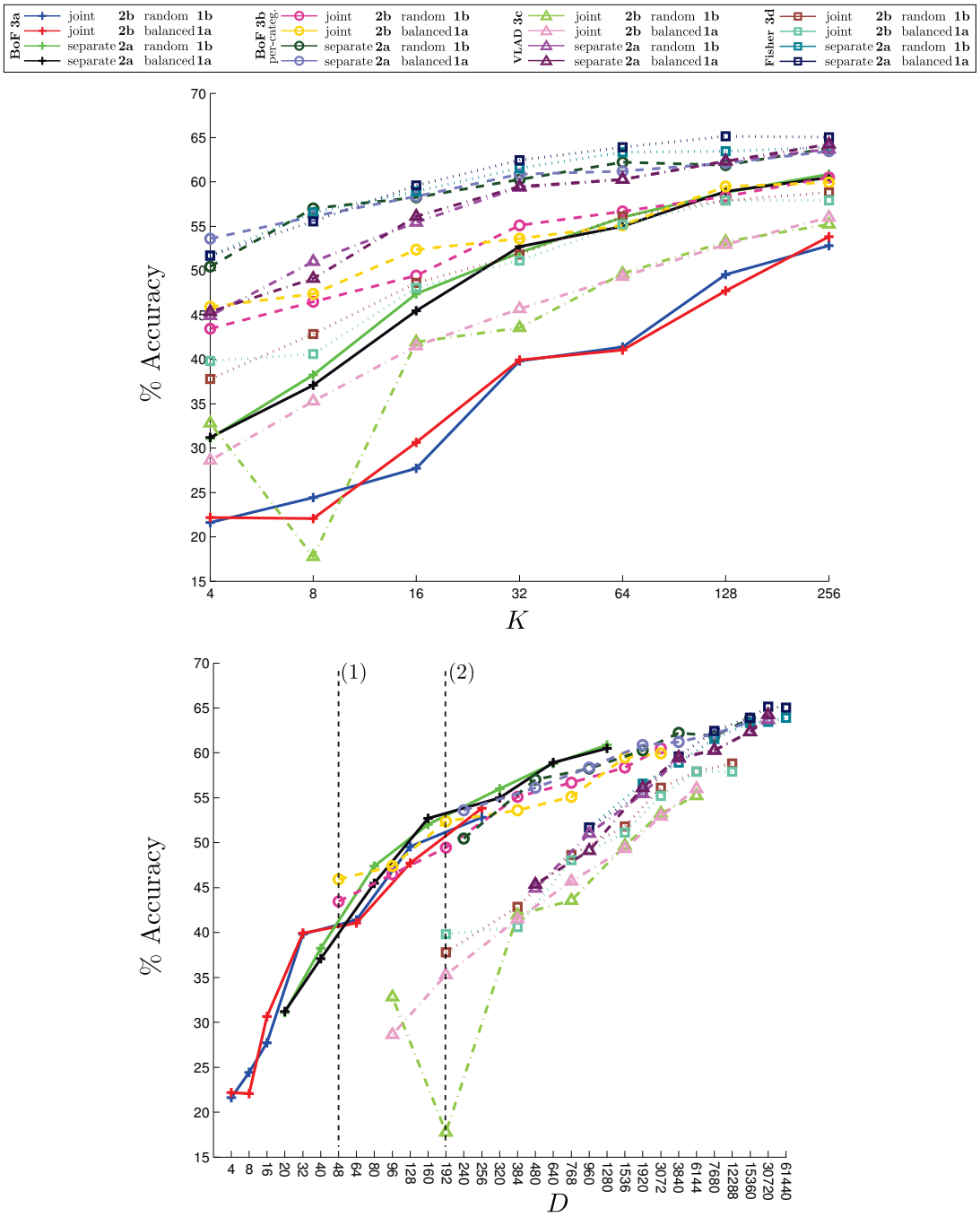


Figure 6.3: The classification accuracy versus K cluster centres (top) and representation dimensionality (bottom) on the Hollywood2 dataset. For the BoF method, notice the significant jump in performance (top) between joint (red/blue crosses) and separate (green/black crosses) feature component vocabulary learning. It is also noteworthy (bottom) that at low dimensionalities, the Chi-square kernel SVM far outperforms the linear SVM, but for higher dimensional vectors, linear SVM surpasses the former. Dotted lines (1) and (2) highlight situations in which (1) BoF-per category achieves competitive performance for small K, and (2) random balanced sampling helps preventing situations in which a bias in the sampled feature pool may give poor results.

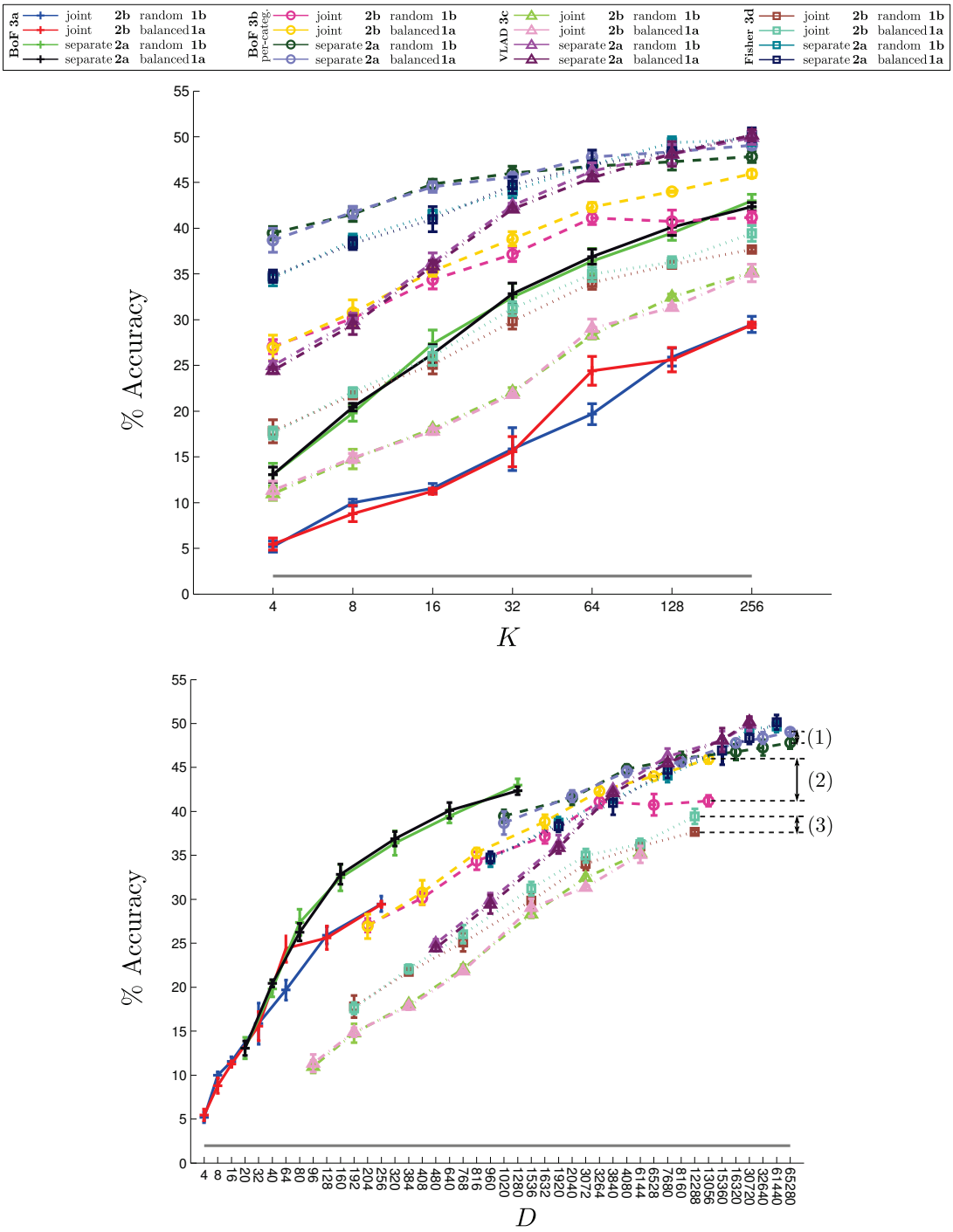


Figure 6.4: The classification accuracy versus K cluster centres (top) and representation dimensionality (bottom) for the HMDB dataset. The differences between the random and balanced features becomes more significant at higher values of K. For example, this is clearly seen between (1) the purple vs. green curves of separate BoF per-category, (2) the yellow vs. pink of joint BoF per-category, and (3) the cyan vs. brown curves of Fisher vectors. The error bars in the figure denote one standard deviation over the three train-test splits defined by the HMDB authors [3]. The grey horizontal bar just under 2% marks the random chance level.

*Table 6.1: The state-of-the-art 'S-O-T-A' results compared to our best results obtained with the global bag-of-features pipeline ('Global best'). The best results are marked in bold. The 'Method' column indicates which of the varying techniques in Algorithm 6.1 gave the best result. The variations '**a**' are marked in bold to distinguish them from variations 'b'. $K - D$ indicates the number of clusters K and the representation dimensionality D. For datasets with more than one train-test split (YouTube, HMDB, USF51, cf. Section 3), we report one standard deviation around the mean.*

KTH [13]	**S-O-T-A**	**Global best**	**Method**	K	D
Acc	96.76 [20]	**97.69**	3b-2b-1b	32	192
mAP	97.88 [20]	**98.08**	3b-**2a**-1b	64	384
mF1	96.73 [20]	**97.68**	3b-2b-1b	32	192
YouTube [16]	**S-O-T-A**	**Global best**	**Method**	K	D
Acc	**93.77 [127]**	89.62 ±5.41	3d-**2a**-1b	128	30,720
mAP	89 [19]	**94.25 ±3.50**	3d-**2a-1a**	128	30,720
mF1	82.43 [20]	**87.45 ±6.92**	3d-**2a**-1b	128	30,720
Hollyw2 [17]	**S-O-T-A**	**Global best**	**Method**	K	D
Acc	60.85 [20]	**65.16**	3d-**2a-1a**	128	30,720
mAP	**63.3** [19]	59.66	3d-**2a-1a**	256	61,440
mF1	52.03 [20]	**54.55**	3d-**2a-1a**	256	61,440
HMDB [3]	**S-O-T-A**	**Global best**	**Method**	K	D
Acc	**66.79 [127]**	50.17 ±0.614	3c-**2a-1a**	256	61,440
mAP	**54.8** [19]	50.07 ±0.33	3d-**2a-1a**	256	61,440
mF1	**52.03** [20]	48.88 ±0.94	3d-**2a-1a**	256	61,440
USF101 [34]	**S-O-T-A**	**Global best**	**Method**	K	D
Acc	**87.9 [140]**	81.24 ±1.11	3d-**2a-1a**	256	61,440
mAP	–	**82.35 ±0.97**	3d-**2a-1a**	256	61,440
mF1	–	**80.57 ±1.11**	3d-**2a-1a**	256	61,440

appear more frequently in the dataset. Again on this dataset, learning vocabularies separately per feature component greatly outperformed learning vocabularies jointly, for every representation dimensionality value D. Here, the best performing approach made use of Fisher vectors (**3d**), with random-balanced sampling (**1a**) and features clustered per component (**2a**), which outperformed the current state-of-the-art by 4.32% (see Table 6.1).

Large performance differences were also observed in Fig. 6.4 for the HMDB dataset. With an increased number of action classes, one can more easily see the performance gained by 'balanced' sampling rather than uniform random sampling. For example, see the differences between the yellow and pink curves of BoF joint per-category (Fig. 6.4 right, dotted lines 2), or the cyan and brown curves of Fisher vector joint random vs. random-balanced (Fig. 6.4 right, dotted lines 3). This was only observed at higher dimensionalities though, and overall 'balanced' sampling outperformed random sampling just 53% of the time. Note

however that the best accuracy, mean average precision and mean F1 scores on the Hollywood2 and HMDB datasets were obtained using random-balanced sampling, as shown in Table 6.1. In Fig. 6.4, one can also observe the benefit of BoF per-category, which outperforms BoF by learning separate vocabularies per action class.

We estimated that to run an additional 112 experiments on UCF101 dataset would take approximately 1 CPU year, and thus report only a single run with the best performing variable components observed in the experiments so far, namely Fisher vectors, balanced sampling and separate vocabulary learning. UCF101 is the largest current action recognition dataset (see Table 2.1), however it may not be the most challenging. We achieved, 81.24%, 82.35%, 80.57% accuracy, mAP and F1 scores respectively averaged over the three train test splits provided by the authors [34]. Our reported accuracy is 37.34% higher than the original results reported in [34], and 4.6% short of the result reported in [141], which additionally used spatial pyramids and a variation of the dense trajectory features which estimates camera motion information [142].

The computational time needed to generate this result on the UCF101 results was 163.52 CPU hours. Interestingly, a large part of the computational time is spent loading features from disk. Thus even though the chance level of UCF101 is just under 1%, our results indicate that the HMDB dataset remains the most challenging action classification dataset currently available. The overall best results are listed in Table 6.1; by using our approach (Algorithm 6.1) with an increased number of model parameters (using a larger K, or by using spatial pyramids [77]) may yield further improvements to the current state-of-the-art results.

Chapter 7

Learning pullback distances for dynamical action models

Suppose that each video in an a training set of S action sequences is represented as a sequence of observations, or a sequence of features. Suppose as well that an algorithm able to identify the parameters of the dynamical model (of a chosen class) which best fits a given sequences of feature vectors is available. Then, the videos can be mapped to a training set $\mathcal{D} = \{\mathbf{m}_1, ..., \mathbf{m}_S\}$ of dynamical models.

7.1 Pullback metrics in Riemannian geometry

Assume initially that the learned dynamical models live on a Riemannian manifold $\mathcal{M}$ (Fig. 7.1). An automorphism[1] (invertible differentiable map $F : \mathbf{m} \in \mathcal{M} \mapsto F(\mathbf{m}) \in \mathcal{M}$ from a domain to itself) on the model manifold $\mathcal{M}$ is associated with a 'push-forward' application of tangent vectors [143]:

$$F_* : \mathbf{v} \in T_{\mathbf{m}}\mathcal{M} \mapsto F_*\mathbf{v} \in T_{F(\mathbf{m})}\mathcal{M}, \tag{7.1}$$

which maps a vector tangent to a curve to the vector tangent to the image of this curve via F (Fig. 7.1-right). Given a Riemannian metric g on $\mathcal{M}$, F induces a *pullback* metric

$$g_*(\mathbf{u}, \mathbf{v}) \doteq g(F_*\mathbf{u}, F_*\mathbf{v}), \tag{7.2}$$

such that the scalar product of two vectors $\mathbf{u}, \mathbf{v}$ according to g_* is equal to the scalar product with respect to the *original* metric g of the push-forward vectors $F_*\mathbf{u}, F_*\mathbf{v}$. The corresponding *pullback distance* between two points $\mathbf{m}$, $\mathbf{m}' \in \mathcal{M}$

[1] Note that the automorphisms presented in this chapter are intended to preserve the smooth structure of the manifold and not the metric structure.

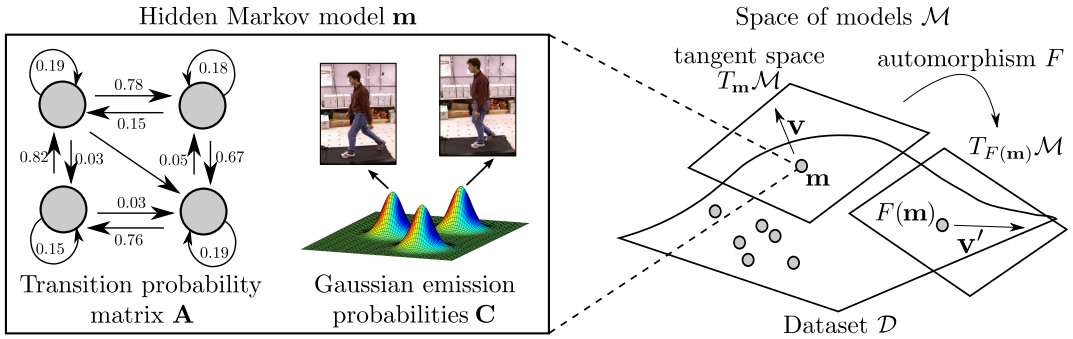


Figure 7.1: Encoding each training video as a dynamical model (e.g. a HMM, left) yields a training set of such models in the relevant space $\mathcal{M}$ (right). In the Riemannian case, any automorphism of $\mathcal{M}$ induces a push-forward map which, given a base metric, induces a pullback distance there. By parameterising the automorphism F we obtain a family of distances we can optimise upon.

(e.g. two dynamical models) is the geodesic distance (length of the shortest path) between the two points according to the obtained pullback metric.

If we define a class of such automorphisms $\{F_{\boldsymbol{\lambda}}, \boldsymbol{\lambda} \in \Lambda\}$ depending on a vector $\boldsymbol{\lambda}$ of parameters we get a family of pullback metrics $\{g_*^{\boldsymbol{\lambda}}, \boldsymbol{\lambda} \in \Lambda\}$ on $\mathcal{M}$, also depending on $\boldsymbol{\lambda}$. We can then define an optimisation problem over such a family in order to select an optimal metric. The nature of the resulting Riemannian manifold depends on the chosen objective function.

7.2 Pullback distance learning framework

Although the differential geometry of some classes of linear dynamical models has indeed been analysed [144], for many other classes used in action recognition (and hidden Markov models in particular) a proper Riemannian metric [145] has not been yet identified. Nevertheless, pullback distances can also be introduced on metric spaces[2], by defining

$$d_*(\mathbf{m}, \mathbf{m}') = d\left(F(\mathbf{m}), F(\mathbf{m}')\right). \tag{7.3}$$

We therefore propose the following more general framework for learning an optimal pullback *distance* from a training set of dynamical models, detailed in Algorithm 7.1.

[2]Sets endowed with a distance function d meeting the triangle inequality (cf. Section A.1)

Algorithm 7.1 A general framework for learning an optimal pullback *distance* from a training set of dynamical models

1. Assume that a dataset of S observation sequences $\{[\mathbf{y}_{\mathrm{t}}^{\mathrm{s}}, \mathrm{t} = 1, \ldots, T_{\mathrm{s}}], \mathrm{s} = 1, \ldots, S\}$ is available;
2. From each sequence, a dynamical model $\mathbf{m}_{\mathrm{s}}$ of a certain set $\mathcal{C}$ (e.g., a HMM) is identified, yielding a dataset of models $\mathcal{D} = \{\mathbf{m}_1, ..., \mathbf{m}_S\}$. Such models belong to a metric space $\mathcal{M}_{\mathcal{C}}$ endowed with a distance function $d_{\mathcal{M}_{\mathcal{C}}}$;
3. A family $\{F_{\boldsymbol{\lambda}}, \boldsymbol{\lambda} \in \Lambda\}$ of automorphisms from $\mathcal{M}_{\mathcal{C}}$ onto itself, parametrised by a vector $\boldsymbol{\lambda}$, is designed. Such family of automorphisms induces a family of distance functions $\{d_*^{\boldsymbol{\lambda}}, \boldsymbol{\lambda}\}$ on $\mathcal{M}_{\mathcal{C}}$;
4. Optimising over this search space of pullback distances yields an optimal distance function $\hat{d}_*$, which can eventually be used to classify previously unseen models.

7.3 Pullback distances for HMMs

A *hidden Markov model* (HMM) is a finite-state stochastic dynamical model whose states $\{X_t\} \in \mathcal{X}$ form a *Markov chain*, which linearly generates an observation process $\mathbf{y}_{\mathrm{t}} \in \mathbb{R}^O$. By identifying its N states with unit column vectors (simplex vertices) $\mathbf{e}_{\mathrm{i}}$ of $\mathbb{R}^N$ [133] we can write it as:

$$X_{t+1} = \boldsymbol{A} X_t + V_{t+1}, \tag{7.4}$$

$$\mathbf{y}_{\mathrm{t+1}} = \boldsymbol{C} X_t + \mathrm{diag}(W_{t+1}) \boldsymbol{\Sigma} X_t, \tag{7.5}$$

where $\{V_{\mathrm{k+1}}\}$ is a sequence of martingale increments and $\{W_{\mathrm{k+1}}\}$ is a sequence of i.i.d. Gaussian noises $\mathcal{N}(0, 1)$. Given a state $X_t = \mathbf{e}_{\mathrm{j}}$, the observations have a Gaussian distribution $p(\mathbf{y}_{\mathrm{t+1}}|X_t = \mathbf{e}_{\mathrm{j}})$ with mean vector $\mathbf{c}_{\mathrm{j}} = E[p(\mathbf{y}_{\mathrm{t+1}}|X_t = \mathbf{e}_{\mathrm{j}})]$, the j-th column of the matrix $\boldsymbol{C}$. The parameters of an HMM are therefore its transition matrix $\boldsymbol{A} = [a_{\mathrm{ij}}] = P(X_{t+1} = \mathbf{e}_{\mathrm{i}}|X_t = \mathbf{e}_{\mathrm{j}})$, the state output matrix $\boldsymbol{C}$ and the covariance matrix $\boldsymbol{\Sigma}$. Given a sequence of observations $\{\mathbf{y}_1, ..., \mathbf{y}_{\mathrm{T}}\}$, the HMM most likely to generate it can be identified via the Expectation Maximisation (EM) algorithm [133].

7.3.1 The space $\mathcal{H}$ of hidden Markov models

In most cases the covariance matrix $\boldsymbol{\Sigma}$ is assumed to be fixed. This helps the convergence of EM without jeopardising the generative power of the resulting

model. The space $\mathcal{H} = \{(\boldsymbol{A}, \boldsymbol{C})\}$ of HMMs with a fixed covariance matrix is then the Cartesian product space $\mathcal{H} = \mathcal{M}_{\boldsymbol{A}} \times \mathcal{M}_{\boldsymbol{C}}$, where $\mathcal{M}_{\boldsymbol{A}}$ denotes the space of all $N \times N$ transition matrices $\boldsymbol{A}$, while $\mathcal{M}_{\boldsymbol{C}}$ is the space of all state-output matrices $\boldsymbol{C}$.

7.3.2 The space of transition matrices

Transition or stochastic matrices have $N(N-1)$ free parameters, as their columns, representing conditional probability distributions $P(X_{t+1}|X_t = \mathbf{e}_{\mathrm{j}})$, sum to 1:

$$\sum_{i=1}^{N} P(X_{t+1} = \mathbf{e}_{\mathrm{i}}|X_t = \mathbf{e}_{\mathrm{j}}) = 1. \tag{7.6}$$

Probability distributions on a finite set of N elements (the states of the Markov model) live in an $(N-1)$-dimensional simplex

$$Cl(\mathbf{p}_1, ..., \mathbf{p}_N) = \big\{\alpha_1 \mathbf{p}_1 + ... + \alpha_n \mathbf{p}_N,\ \sum_{\mathrm{i}} \alpha_{\mathrm{i}} = 1\big\}, \tag{7.7}$$

having as vertices $\mathbf{p}_1, ..., \mathbf{p}_N$ the probability distributions $p_{\mathrm{i}} : \mathcal{X} \to [0, 1]$ such that $p_{\mathrm{i}}(\mathbf{e}_{\mathrm{i}}) = 1$, $p_{\mathrm{i}}(\mathbf{e}_{\mathrm{j}}) = 0\ \forall \mathrm{j} \neq \mathrm{i}$. Transition matrices live therefore in the Cartesian product of N such simplices, one for each column $\mathbf{a}_{\mathrm{j}}$ of $\boldsymbol{A}$ (see Fig. 7.2):

$$\mathcal{M}_{\boldsymbol{A}} = \prod_{\mathrm{j}=1}^{N} Cl(\mathbf{p}_1^{\mathrm{j}}, ..., \mathbf{p}_N^{\mathrm{j}}). \tag{7.8}$$

Any $\boldsymbol{A} \in \mathcal{M}_{\boldsymbol{A}}$ is identified by the collection of simplicial coordinates $(a_{\mathrm{ij}}, \mathrm{i} = 1, ..., N)$ of all its columns $\mathbf{a}_{\mathrm{j}}$, $\mathrm{j} = 1, ..., N$ in the respective simplices, i.e., the latter's stacked vector.

7.3.3 Learning an approximate observation space

The state-output $\boldsymbol{C}$ matrices are instead N-plets of vectors of the observation space $\mathcal{Y}$ of the Markov model. When a training set of models is available, we can use the information it provides to learn an approximation $\tilde{\mathcal{Y}}$ of $\mathcal{Y}$. It has indeed been observed that for specific classes of motions, observations live in a lower-dimensional manifold [146] of $\mathbb{R}^O$. By applying unsupervised nonlinear embedding to the collection of columns of all the $\boldsymbol{C}$ matrices of the training set of HMMs, we get a cloud $\tilde{\mathcal{Y}}$ of D-dimensional embedded columns approximating the observation space, as illustrated in Fig. 7.3. Here we use Locally Linear

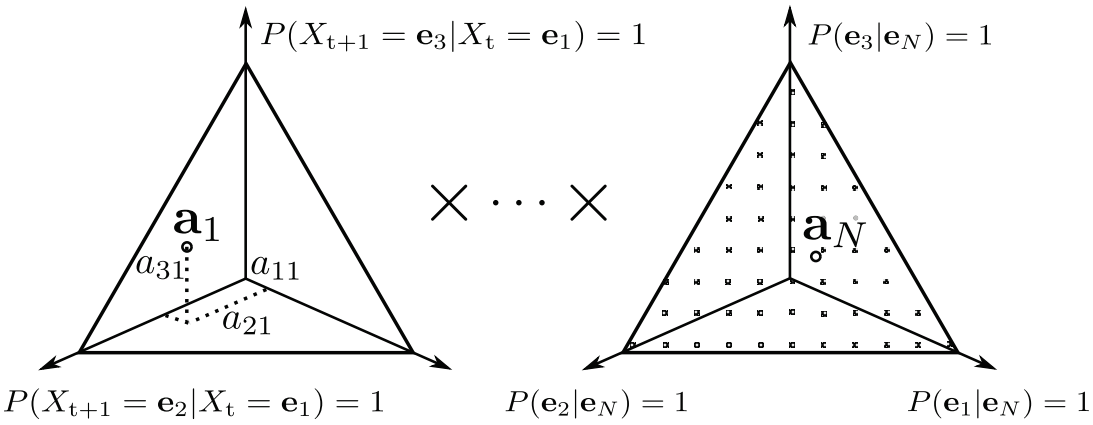


Figure 7.2: The space $\mathcal{M}_{\boldsymbol{A}}$ of transition matrices with N states is the product of the N simplices of conditional probabilities $P(X_{k+1} = e_i | X_k = e_j)$, $j = 1, ..., N$. $\mathcal{M}_{\boldsymbol{A}}$ is isomorphic to the proposed automorphism space $\Lambda_{\boldsymbol{A}}$, since the weights $\boldsymbol{\lambda}$ also form points on 2-simplices. For our tests, each simplex of $\Lambda_{\boldsymbol{A}}$ was sampled along a regular grid (right).

Embedding (LLE) [147], though other dimensionality reduction algorithms [148] such as Kernel-PCA [149], or Neural networks [150] may be employed.

Each element of the obtained lower-dimensional observation space needs to be associated with a vector of real numbers, its coordinates, to which an automorphism can be applied. This can be done by estimating via EM the Mixture of Gaussians (MoG) $\{\Gamma^k, k\}$ which best fits the approximate observation manifold $\tilde{\mathcal{Y}}$ (cf. Fig. 7.3-left). As coordinates of each (embedded) observation vector $\tilde{\mathbf{y}}$ we may then use the associated unique set of probability density values: $\mathbf{y} \mapsto \{\Gamma^k(\mathbf{y}), k\}$ (cf. Fig. 7.3-right). As each $\boldsymbol{C}$ matrix is an N-tuple of observation vectors, a point $\boldsymbol{C}$ of the (approximate) space $\tilde{\mathcal{M}}_{\mathcal{C}}$ has as coordinates the stacked coordinates of its embedded columns under the MoG.

7.3.4 An automorphism of $\mathcal{H}$

We can now design a product automorphism:

$$F(h) = F((\boldsymbol{A}, \boldsymbol{C})) = (F_{\boldsymbol{A}}(\boldsymbol{A}), F_{\boldsymbol{C}}(\boldsymbol{C})), \tag{7.9}$$

on the space $\mathcal{H}$ of HMMs, where $h \in \mathcal{H}$ is a hidden Markov model, $F_{\boldsymbol{A}} : \mathcal{M}_{\boldsymbol{A}} \to \mathcal{M}_{\boldsymbol{A}}$ and $F_{\boldsymbol{C}} : \tilde{\mathcal{M}}_{\boldsymbol{C}} \to \tilde{\mathcal{M}}_{\boldsymbol{C}}$ are automorphisms of the space $\mathcal{M}_{\boldsymbol{A}}$ of transition matrices and of the approximate space $\tilde{\mathcal{M}}_{\boldsymbol{C}}$ of $\boldsymbol{C}$ matrices, respectively.

An automorphism of transition matrices can be derived by exploiting the form of $\mathcal{M}_{\boldsymbol{A}}$ as the product of N simplices. Each probability distribution p in a probability simplex $\mathcal{P} = Cl(\mathbf{p}_1, ..., \mathbf{p}_N)$ with N vertices is a point whose

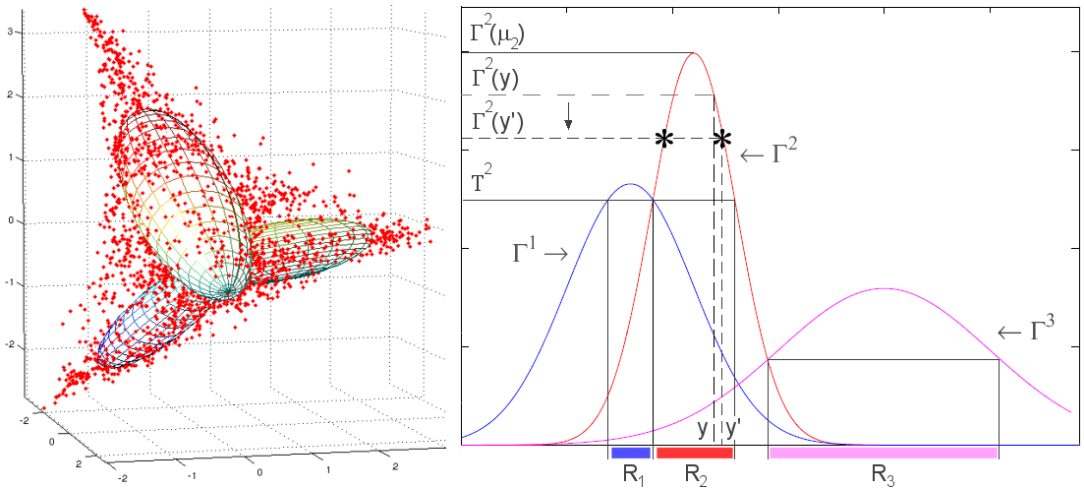


Figure 7.3: The LLE embeddings (here of dimension $d = 3$) of the columns of the $\boldsymbol{C}$ matrices of all the HMMs identified from a training set of videos form a reasonable approximation $\tilde{\mathcal{Y}}$ of the unknown observation space $\mathcal{Y}$. Fitting a Mixture of Gaussians to the embedded cloud provides an atlas of coordinates charts in the approximate observation space (left). Mapping observation vectors in the embedded space via a MoG: a 1-D example (right).

simplicial coordinates are its probability values:

$$p = \sum_{\mathrm{i}} p(\mathbf{e}_{\mathrm{i}}) \mathbf{p}_{\mathrm{i}}. \tag{7.10}$$

A simple automorphism there can be obtained by stretching the simplicial coordinates $\mathbf{p} = \big(p(\mathbf{e}_1), ..., p(\mathbf{e}_N)\big)^\top$ of its points $\mathbf{p}$ by a set of normalised weights $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]$ with $\sum_{\mathrm{i}} \lambda_{\mathrm{i}} = 1$, $\lambda_{\mathrm{i}} \geq 0$:

$$F_{\boldsymbol{\lambda}}(\mathbf{p}) = \frac{\big(\lambda_1 p(\mathbf{e}_1), \lambda_2 p(\mathbf{e}_2), ..., \lambda_n p(\mathbf{e}_N)\big)^\top}{\boldsymbol{\lambda} \cdot \mathbf{p}}, \tag{7.11}$$

where $\boldsymbol{\lambda} \cdot \mathbf{p}$ denotes the scalar product of the two vectors. A *product* automorphism $F_{\boldsymbol{A}}$ for the transition space $\mathcal{M}_{\boldsymbol{A}} = \prod_{\mathrm{j}=1}^{N} Cl(\mathbf{p}_1^{\mathrm{j}}, ..., \mathbf{p}_N^{\mathrm{j}})$ can be derived by applying a simplicial stretch with parameter $\boldsymbol{\lambda}^{\mathrm{j}}$ to each column $\mathbf{a}_{\mathrm{j}}$ of $\boldsymbol{A}$:

$$F_{\boldsymbol{A}}(\boldsymbol{A}) = \{F_{\boldsymbol{\lambda}^{\mathrm{j}}}(\mathbf{a}_{\mathrm{j}}), \mathrm{j} = 1, ..., N\}. \tag{7.12}$$

More general deformations are discussed in Sections 7.5.3 & A.3.

The Mixture of Gaussians approximation of $\tilde{\mathcal{Y}}$ can be exploited to work out an automorphism of the approximate observation space itself, and as a consequence of the space $\mathcal{M}_{\boldsymbol{C}}$ of $\boldsymbol{C}$ matrices. The rationale is the following: the space where the observation vectors live is not analytically known (unlike $\mathcal{M}_{\boldsymbol{A}}$), but needs to

be inferred from the training set. As a topological manifold is a space covered by an atlas of coordinate charts, each of them providing vector coordinates for a local neighbourhood of the manifold, the MoG $\{\Gamma^{\mathrm{k}}, \mathrm{k}\}$ provides an atlas of local charts for the (unknown) HMM observation manifold. Any global differentiable map (automorphism) there needs then to be expressed in terms of these local coordinates. Ample freedom, however, exists on how to design a set of local automorphisms in each coordinate chart.

In our case, the Euclidean coordinate(s) in each chart reduce to the probability densities $\{\Gamma^{\mathrm{k}}(\mathbf{y}), \mathrm{k}\}$ of each (embedded) observation vector $\mathbf{y}$: any local automorphism will then have to act on these density values. We propose here to apply a stretching to the density value of the local dominant Gaussian component, in analogy to what done for transition matrices. Mappings acting on the vector of all Gaussian densities in the mixture are of course conceivable. Further design choices are discussed in Sections 7.5.3 & A.3.

Each Gaussian Γ^{k} defines a region of the (embedded) observation space in which Γ^{k} is greater than all other components' densities,

$$\mathcal{O}^{\mathrm{k}} = \{\mathbf{w} : \Gamma^{\mathrm{k}}(\mathbf{w}) \geq \Gamma^{l}(\mathbf{w}) \; \forall l \neq \mathrm{k}\}. \tag{7.13}$$

Consider the maximum value attained here by any other Gaussian component $T^{\mathrm{k}} = \max_{l \neq \mathrm{k}, w \in O^{\mathrm{k}}} \Gamma^{l}(w)$, and define the 'proprietary' region $\mathcal{R}^{\mathrm{k}}$ of Γ^{k} as:

$$\mathcal{R}^{\mathrm{k}} = \{w : \Gamma^{\mathrm{k}}(\mathbf{w}) \geq T^{\mathrm{k}}\} \tag{7.14}$$

(see Fig. 7.3 for the case of a 1-D embedded observation space and 3 Gaussian components). We can define an automorphism $F_{\tilde{\mathcal{Y}}} : \tilde{\mathcal{Y}} \to \tilde{\mathcal{Y}}$ of the approximate observation space $\tilde{\mathcal{Y}}$ which acts non-trivially only on embedded observation vectors *living in the proprietary region* of a Gaussian component. The sequence of steps necessary to compute the automorphism of $\boldsymbol{C}$ is presented in Algorithm 7.2.

7.3.5 Sampling the parameter space of the automorphism

In order to pick the automorphism which generates the optimal pullback distance we need to sample the parameter space Λ of the mapping. The pullback distance associated with each sample is applied to a validation fold of the dataset, and the parameter which yields the best classification performance there is picked. The parameter space $\Lambda_{\boldsymbol{A}}$ for the automorphism $F_{\boldsymbol{A}}$ of transition matrices is the product of N simplices with N vertices (as each column of $\boldsymbol{A}$ is

Algorithm 7.2 Computing an automorphism of the approximate observation space.

Given $\tilde{\mathbf{y}} \in \tilde{\mathcal{Y}}$:

1. Select the Gaussian component with highest density value: $K = \underset{\mathrm{k}}{\operatorname{argmax}}\, \Gamma^{\mathrm{k}}(\tilde{\mathbf{y}})$;
2. If $\tilde{\mathbf{y}} \notin \mathcal{R}^K$ leave it unchanged: $\tilde{\mathbf{y}}' \doteq F_{\tilde{\mathcal{Y}}}(\tilde{\mathbf{y}}) = \tilde{\mathbf{y}}$;
3. If $\tilde{\mathbf{y}} \in \mathcal{R}^K$, its density value $\Gamma^K(\tilde{\mathbf{y}})$ belongs to the interval $[T^K, \Gamma^K(\boldsymbol{\mu}_K)]$, with convex coordinates $\mu, (1-\mu)$, $\mu \in [0,1]$: $\Gamma^K(\tilde{\mathbf{y}}) = \mu T^K + (1-\mu)\Gamma^K(\boldsymbol{\mu}_K)$, where $\boldsymbol{\mu}_K \in \mathcal{Y}$ is the mean of Γ^K;
4. If $\lambda \in [0,1]$ is the parameter of the desired automorphism, map $\Gamma^K(\tilde{\mathbf{y}})$ to the new density value in the same interval $[T^K, \Gamma^K(\boldsymbol{\mu}_K)]$ by stretching its convex coordinates via:
$$\Gamma^K(\tilde{\mathbf{y}}') = \frac{\lambda\mu}{\lambda\mu + (1-\lambda)(1-\mu)} T^K + \frac{(1-\lambda)(1-\mu)}{\lambda\mu + (1-\lambda)(1-\mu)} \Gamma^K(\boldsymbol{\mu}_K); \quad (7.15)$$
5. Within $\mathcal{R}^K$ there exists an entire level set of observation vectors with such a density value $\Gamma^K(\tilde{\mathbf{y}}')$ (the two starred points in the 1-D example of Fig. 7.3);
6. We define as $\tilde{\mathbf{y}}' = F_{\tilde{\mathcal{Y}}}(\tilde{\mathbf{y}})$ the unique vector $\tilde{\mathbf{y}}'$ which has density value $\Gamma^K(\tilde{\mathbf{y}}')$ and lies on the half-line joining $\boldsymbol{\mu}_{\mathrm{k}}$ and the original vector $\tilde{\mathbf{y}}$.

'stretched' using a normalised vector $\boldsymbol{\lambda} = [\lambda_1, ..., \lambda_N]$ of N components), i.e., it is isomorphic to $\mathcal{H}_{\boldsymbol{A}}$ itself. $\Lambda_{\boldsymbol{A}}$ is regularly sampled along a grid (see Fig. 7.2). For example, using $n_{bin} = 10$ bins per dimension, each simplex is sampled in $n_{sam} = \sum_{\mathrm{i}=1}^{n_{bin}} \binom{n_{bin}-\mathrm{i}+(N-1)}{N-1} = 55$ locations (where $N = 3$ is the number of HMM states). The total number of samples in $\Lambda_{\boldsymbol{A}}$ is then $(n_{sam})^{n_{col}}$, where n_{col} is the number of transition columns actually stretched. The map $F_{\boldsymbol{C}}$ depends instead on a single scalar parameter $\lambda \in [0,1]$: in our tests it was sampled uniformly with much greater density.

7.4 A proof of concept

As a first proof of concept, we defined an experiment on a toy problem with synthetic data in order to show the merits of our pullback-HMM learning in a controlled environment. Since our method first learns an HMM from observation sequences, and measures distances between HMMs to make a classification deci-

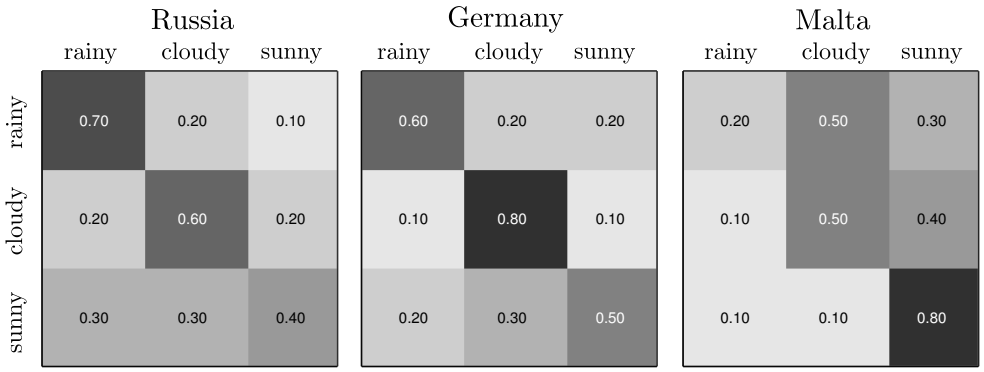


Figure 7.4: Transition matrices $\boldsymbol{A}^\top$ based on fictional weather patterns from each country. Darker shades indicate higher probabilities.

sion, we must first outline a method (see Algorithm 7.3) to generate a synthetic dataset of HMM observation sequences which can then be fed to the pullback HMM learning pipeline.

Consider a scenario in which families or groups of friends go travelling to L countries drawn uniformly at random, in this case: $\mathrm{l} \in \{$*Malta, Germany, Russia*$\}$. Each group (20 people) records the snack they ate at 4pm on each of the T days on holiday (the length of each observation sequence is $T = 14$), and the possible observations are drawn from a list of M snacks: $m \in \{$*hot chocolate, pretzels, ice cream, doughnut*$\}$. The overall group snacking distribution per day is captured by a histogram, counting how many snacks of each type have been consumed. Back home each group is instructed to learn an N-state HMM (e.g. $N = 3$) from the sequence of observations they collected. The objective is to automatically determine which country they have visited by classifying the HMM learnt on their return home. In this case, the choice of snack is dependent on the current state of the weather in a particular country, whilst the dynamics of the weather depend in turn on the country.

7.4.1 Synthetic HMM sequences

In order to generate the observation sequences, L transition and state-output matrices were predefined (see Fig. 7.4 & Fig. 7.5). The hidden states of each HMM represent possible weather states: Rainy, Cloudy, and Sunny (the possible states remain unknown to the travellers). The synthetic sequences' generation procedure is laid out in Algorithms 7.3 and 7.4.

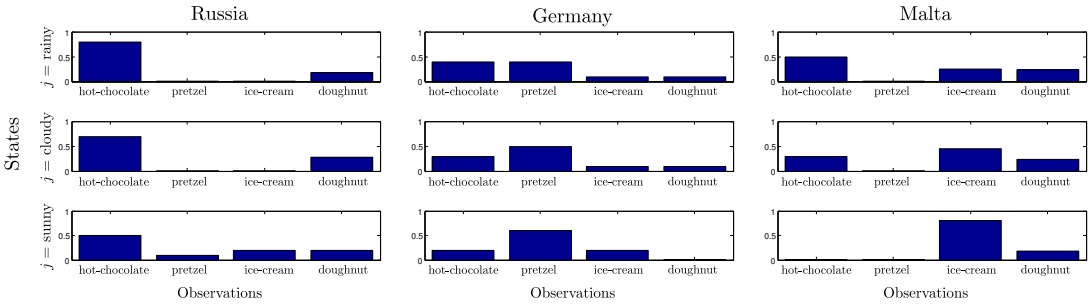


Figure 7.5: State-output probability matrices $\boldsymbol{C}^\top$ based on fictional snacking patterns for each country under different weather conditions. The possible classes are the countries the travellers went to on holiday (Germany, Malta, Russia), the hidden states correspond to the weather conditions (sunny, cloudy and rainy), and the discrete observations are drawn from a set of snacks, in this case: hot-chocolate, pretzel, ice-cream and doughnut.

7.4.2 Experimental setup

To create the artificial toy dataset we generated 300 examples per class, using Algorithm 7.3. The obtained sequences were then averaged and plotted to manually verify that the histograms indeed matched the original discrete distributions they were sampled from. The HMM parameters were identified via the EM algorithm, applied 10 times for each observation sequence in order to select the parameters yielding the highest likelihood. The toy dataset was split into three randomly selected sets, and for each set, 2/3 of the data was placed in the training set, and the remainder in the test set. For each set, the parameters of the pullback learning algorithm were optimised using 5-fold cross validation on the training set. The results over each of the three sets were plotted using the mean accuracy and one standard deviation from the mean, as shown in Fig. 7.6 & Fig. 7.7.

7.4.3 Preliminary results & discussion

In the first experiment, the results of which are plotted in Fig. 7.6, the sampling densities of the transition automorphism $F_{\boldsymbol{A}}$ and output automorphism $F_{\boldsymbol{C}}$ were varied, keeping the number of HMM states fixed at 3. To calculate the HMM approximate observation space we applied LLE with an embedding space dimensionality of $d = 3$ and number of neighbours equal to 40. In Fig. 7.6 the classification accuracy is plotted against the sampling densities in both $\Lambda_{\boldsymbol{A}}$ and $\Lambda_{\boldsymbol{C}}$. It can be clearly seen that the pullback-Frobenius performance improves steadily as the sampling density in the observation automorphism's parameter space increases.

Algorithm 7.3 Generate synthetic HMM observations.

STEP 1: Initialise dataset parameters:
-classes ($\mathrm{l} \in \{$*Malta, Germany, Russia*$\}$),
-states ($\mathrm{j} \in \{$*Rainy, Cloudy, Sunny*$\}$), $N := 3$,
-observations ($m \in \{$*hot-chocolate, pretzels, ice cream, doughnut*$\}$),
-length of trip ($T := 14$),
-group size ($G := 20$),
-examples per class ($E := 300$),
-transition matrices $\boldsymbol{A}$ (Fig. 7.4),
-state output observation matrices $\boldsymbol{C}$ (Fig. 7.5),
-initial state probabilities $\pi := (\frac{1}{N}, \frac{1}{N}, \frac{1}{N})$.
STEP 2: Monte-Carlo simulation
for $\mathrm{l} := 1$ **to** L **do**
  for $e := 1$ **to** E **do**
    $\mathrm{j} :=$ `generate-random-choice`(π),
    while `length`(*observ*) $< T$ **do**
      observ $:=$ `generate-random-histogram`$(\mathbf{c}_{\mathrm{j}}^{\mathrm{l}}, G)$,
      $\mathrm{j} :=$ `generate-random-choice`$(\mathbf{a}_{\mathrm{j}}^{\mathrm{l}})$,
    end while
    Output: *observ*, a $(T \times M)$ matrix of histograms.
  end for
end for

In the second experiment (Fig. 7.7), the sampling density was kept constant at 8 samples in $\Lambda_{\boldsymbol{C}}$ and 216 samples in $\Lambda_{\boldsymbol{A}}$, and the learnt number of states was varied from 2 to 4. All other parameters were kept constant. Interestingly, even though the predefined HMMs have 3 states, the HMM-pullback method was still able to discriminate between HMM models of 2 and 4 states, and even benefited from an increased number of states (see the minor improvement in accuracy and a reduced standard deviation in Fig. 7.7).

These preliminary results demonstrated the higher performance achievable using pullback distance learning for hidden Markov models, especially when

Algorithm 7.4 `generate-random-choice`.

Input: $\mathbf{q}$, $q_{\mathrm{i}} \in [0,1], \mathrm{i} \in \{1, \ldots, Q\}$,
$\mathrm{i} := 1$,
$q' :=$ `rand`$(0,1) - q_{\mathrm{i}}$,
while $(\mathrm{i} < Q)$ && $(q' > 0)$ **do**
  $\mathrm{i} := \mathrm{i} + 1$,
  $q' := q' - q_{\mathrm{i}}$,
end while
Output: i.

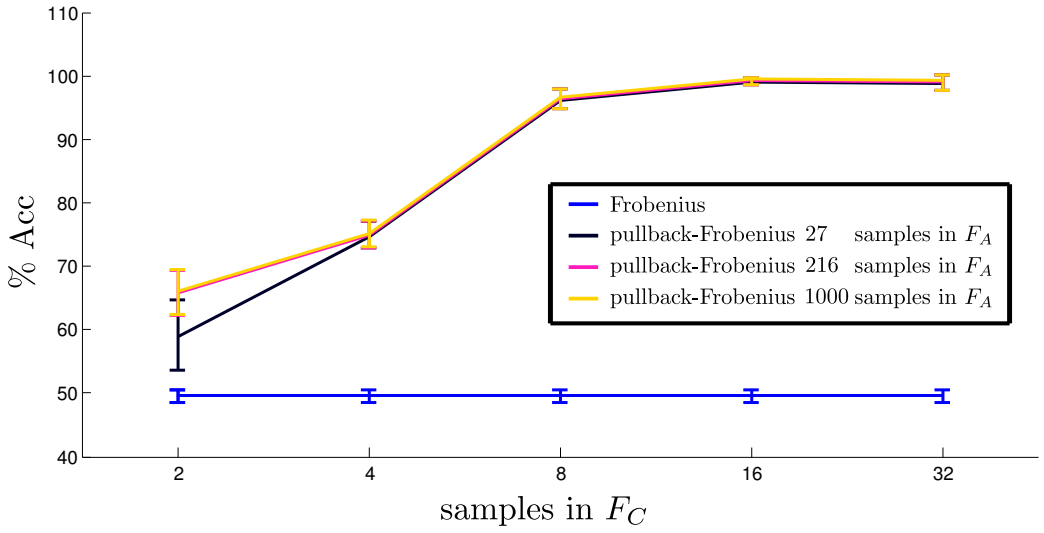


Figure 7.6: The % Accuracy plotted against the sampling density in the parameter spaces of F_A and F_C. Whereas the Accuracy of the base Frobenius distance remains constant, the Accuracy of the pullback-learning method increases steadily as the number of samples in F_C increases. The sampling density in Λ_A has a lesser effect in this case, with similar performances recorded for 216 or 1000 samples.

compared to that of the base distance, the positive effect of higher sampling densities of the parameter space, and the robustness of the method to the number of states of the models.

7.5 Experiments on human action recognition

We validated our proposal for learning optimal (classification-wise) pullback distances between HMMs on the KTH [13] and YouTube [16] datasets, the latter providing more challenging, real-world conditions (cf. Section 3.1.2). Instead of having histogram sequences extracted from the snacking desires of a group of travellers, we now consider histogram sequences extracted from features of a video in which an action was performed.

7.5.1 Implementation details

In order to encode each video as an HMM we need to associate a feature vector with each video frame. A 'sliding window' approach can be applied in which for each time instant t features are extracted from the spatio-temporal subvolume collecting the images from t to $t+\delta$ and attributed to the state X_t of the Markov chain. For these tests we picked the Dense Trajectory features (c.f Section 2.2), which demonstrated excellent performance in [25]. We kept the default parame-

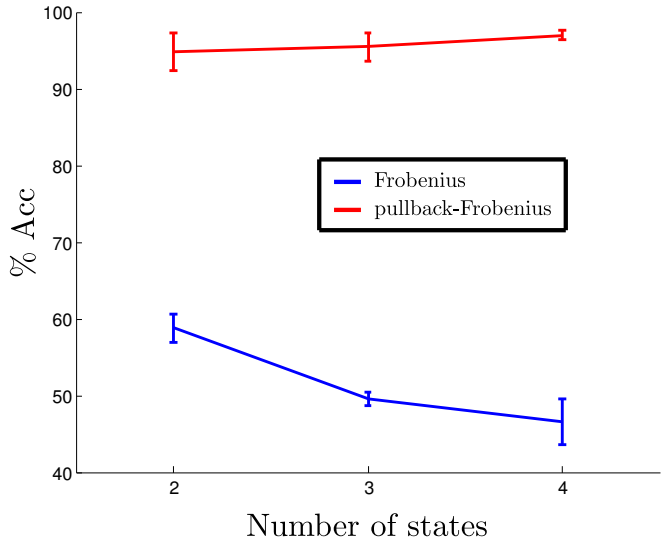


Figure 7.7: The % Accuracy plotted against the learnt number of states in the HMM. With the baseline Frobenius distance, the accuracy decreases as the number of states increases from 2 to 4. On the contrary, when using our pullback learning method the accuracy increases slightly (and the standard deviation is reduced) as the number of states increases. Also note the significant difference in accuracy in all cases.

ters: features were computed in video blocks of size 32×32 pixels for 15 frames, with a dense sampling step size of 5 pixels [25]. We used the 30-dimensional motion vectors from the Dense Trajectory features, and built a 200-word visual vocabulary by sampling features from the training set and clustering them by k-means. The k-means algorithm was initialised 8 times and the configuration with the lowest error selected. We used a sliding 3D-window of width $\delta = 30$ frames and a stride of 5 pixels, the same step size used for the Dense Trajectory features. For each time-slice, a 200-dimensional histogram was computed by quantising each feature to the learned visual vocabulary. For each feature sequence so obtained, the HMM parameters were identified via the EM algorithm [133].

HMMs characterised by a different number of states N that live in principle on different model spaces, due to the different spaces $\mathcal{H}_{\boldsymbol{A}}$ of transition matrices. Here we set the number of states to $N = 3$ to make the models comparable. Three state automata have been demonstrated to represent most simple actions effectively. Even then, a Markov model is uniquely defined only up to a permutation of the states. Therefore, when measuring distances between two models we looked for the state permutation minimising the Frobenius distance between the respective $\boldsymbol{C}$ matrices, so identifying states associated with the same 'clusters' in the observation space. To calculate the HMM approximate observation space we applied LLE with an embedding space dimensionality of $d = 6$ and size of

the neighbours equal to 201. A mixture of 3 Gaussian components was fitted to the resulting embedded columns to provide coordinates.

As a base distance on $\mathcal{H} = \mathcal{M}_{\boldsymbol{A}} \times \mathcal{M}_{\boldsymbol{C}}$ we picked the product metric obtained by applying the Frobenius norm to the $\boldsymbol{A}$ and $\boldsymbol{C}$ matrices respectively:

$$\|\boldsymbol{H}_1 - \boldsymbol{H}_2\| = \|\boldsymbol{A}_1 - \boldsymbol{A}_2\|_F + \|\boldsymbol{C}_1 - \boldsymbol{C}_2\|_F, \tag{7.16}$$

where $\|\boldsymbol{M} - \boldsymbol{M}'\|_F \doteq \sqrt{Tr((\boldsymbol{M} - \boldsymbol{M}')(\boldsymbol{M} - \boldsymbol{M}')^\top)}$, and $Tr(\boldsymbol{M}) = \sum_{\mathrm{ii}} \boldsymbol{M}[\mathrm{i}, \mathrm{i}]$ is the trace of a matrix $\boldsymbol{M}$. An alternative derives from the Bhattacharyya distance between two Gaussian pdfs $\mathcal{G}_1, \mathcal{G}_2$ with means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and covariances $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ [41]:

$$d_{bhatta}(\mathcal{G}_1, \mathcal{G}_2) = \frac{1}{8}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top (\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2})^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \frac{1}{2}\log \frac{|(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)/2|}{|\boldsymbol{\Sigma}_1|^{1/2}|\boldsymbol{\Sigma}_2|^{1/2}}. \tag{7.17}$$

Given two HMMs the average Bhattacharyya distance between pairs of emission Gaussian pdfs Γ_k in the two models quantifies their difference, and can be modified to include the K-L divergence

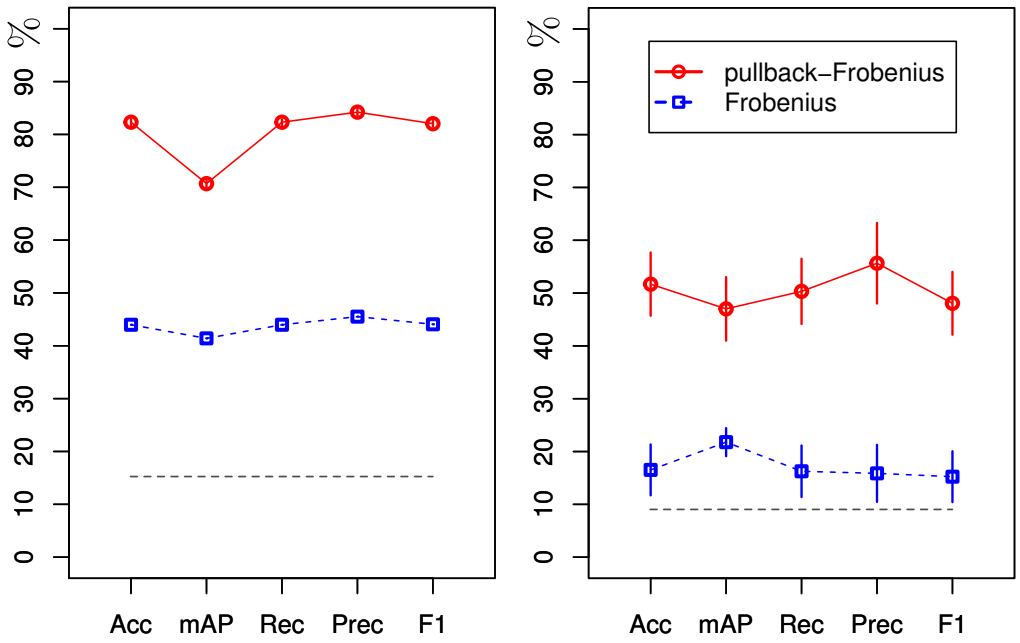
$$\sum_{\mathrm{i,j}=1,\ldots,N} \pi(\mathbf{e}_\mathrm{i}) \boldsymbol{A}_2[\mathrm{i}, \mathrm{j}] \log \left(\frac{\boldsymbol{A}_1[\mathrm{i}, \mathrm{j}]}{\boldsymbol{A}_2[\mathrm{i}, \mathrm{j}]} \right) \tag{7.18}$$

of the two transition matrices $\boldsymbol{A}_1, \boldsymbol{A}_2$ [41] (where $\pi(\mathbf{e}_\mathrm{i})$ is the probability of state $\mathbf{e}_\mathrm{i}$).

For both benchmarks, we learned an optimal pullback distance by maximising the classification performance on the training set via 5-fold cross validation [3]. We then measured the accuracy (Acc), average precision (AP), and F1-score achieved by the learnt distance on the test set (cf. Section 3.1.6). For classification we used Nearest Neighbour (1-NN): each test sequence was attributed the class of the nearest model in the training set according to the considered distance. Having fixed a classification strategy, it is possible to fairly compare pullback and base distances.

7.5.2 Results and discussion

Fig. 7.8(a) illustrates the effect of pullback distance function learning on the KTH dataset. The parameter space of the HMM automorphism was sampled in 55 points for the transition matrix and 51 points for the output space (see the above description of sampling) to optimise the performance of the pullback HMM



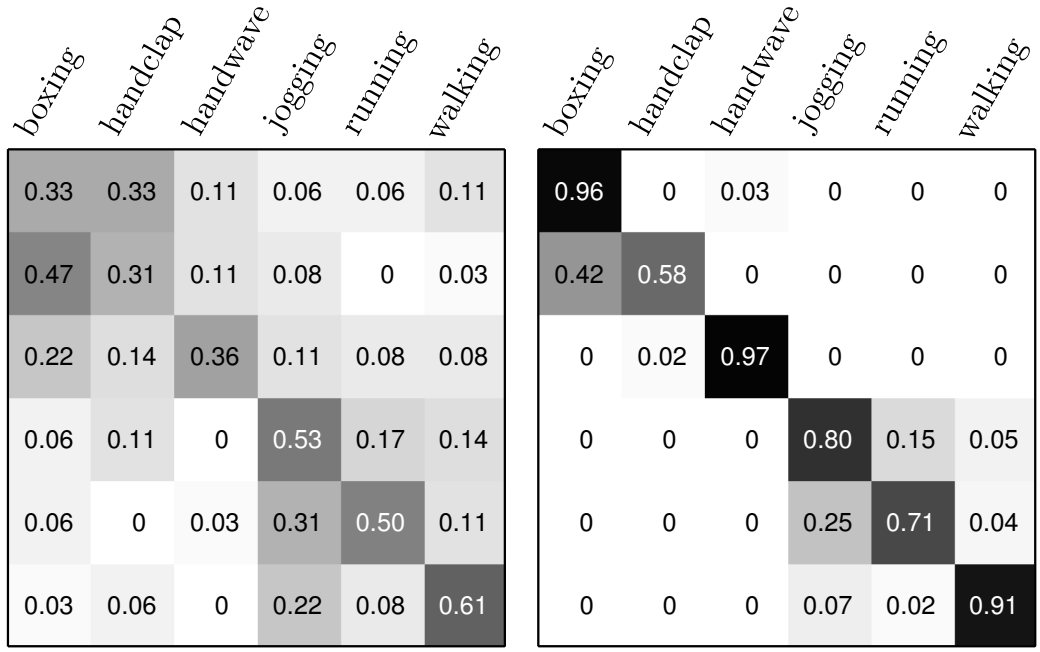
(a) KTH (b) YouTube

Figure 7.8: Performance plots comparing the Frobenius 'base' distance (blue square markers) to the pullback-Frobenius distance (red circular markers) on the ***(a)*** *KTH dataset [13], and* ***(b)*** *YouTube [16] dataset. The chance level is plotted as a dotted straight line in grey.*

distance by cross validation. When using the pullback-Frobenius distance, we achieved a very significant 38.3%, 29.3% and 37.9% improvement in classification accuracy, mAP and F1 score respectively, with respect to the base one.

Fig. 7.8(b) shows instead the improvements in action recognition rates on the YouTube dataset. Results are averaged over the 25 splits of the dataset: the vertical bars indicate one standard deviation from the mean. The pullback distance was optimised over 55 samples in the parameter space $\Lambda_{\boldsymbol{A}}$ of the transition automorphism $F_{\boldsymbol{A}}$ and 31 samples in that of the output automorphism $F_{\boldsymbol{C}}$: a total of $55 \times 31 = 1705$ samples. The accuracy for the pullback-Frobenius distance was 51.7% with a standard deviation of 5.9%, which is again a significant jump from the base distance's poor $16.5\% \pm 4.8\%$.

Figs. 7.9 and 7.10 contrast the performances of base and pullback-Frobenius distances over the two datasets via the associated confusion matrices. Classification accuracy for individual classes can be appreciated. Notice that pullback distances are able to disambiguate many classes which are confused by base ones, such as the 'boxing' and 'handwaving' actions in the KTH dataset, or YouTube's 'biking' and 'riding' action classes. Overall, our method proves able to cope with the large variety of nuisance factors present in the YouTube dataset, such as high within-class variation, unconstrained camera motion of handheld cameras, and



(a) Frobenius (b) pullback-Frobenius

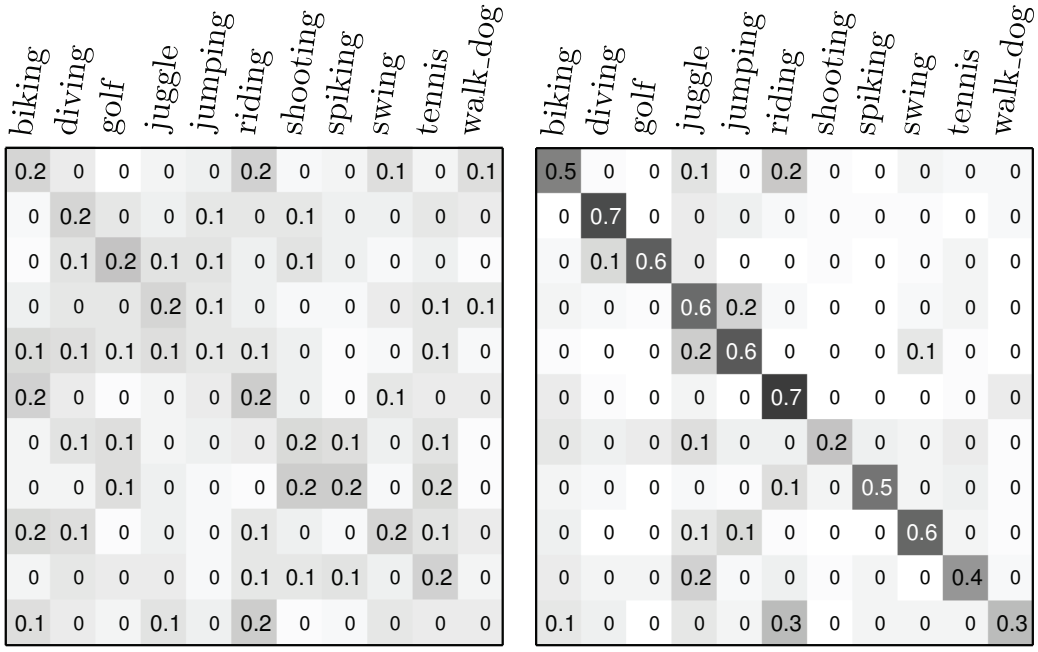
*Figure 7.9: Confusion matrices, KTH tests. Notice the significant overall improvement achieved by the pullback-Frobenius distance **(b)**, as compared to the base distance **(a)**.*

cluttered scenes.

Since the YouTube dataset contains harder action sequences and a greater number of action classes than the KTH dataset, we present here additional experiments to explore the pullback learning performance using Fisher vectors and two additional distance metrics. From the plots of Fig. 7.11 it can be seen that the pullback-Frobenius distance performs best overall with a mean Accuracy of 44.4%, compared to pullback-Bhattacharyya's 37.8% and modified pullback-Bhattacharyya's 35.5%. Interestingly, the performance results were slightly worse when using Fisher vectors rather than BoF histograms. However, there is still an appreciable improvement between base and pullback distances, demonstrating the effectiveness of our framework across multiple features and metrics. The adoption of more complex classes of automorphisms and models is discussed in the following Section.

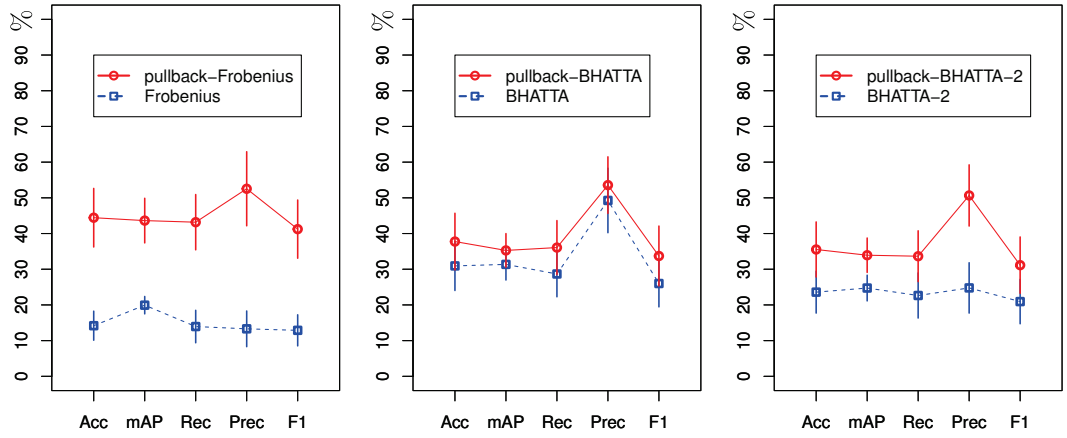
7.5.3 Other nonlinear automorphisms

The type of automorphisms employed here, as shown in Fig. 7.12(a), map linear boundaries to different linear boundaries. Other nonlinear automorphisms, for example, of the form $\boldsymbol{\lambda}\mathbf{p}^{\boldsymbol{\lambda}}$ shown in Fig. 7.12(b), have the potential to extend



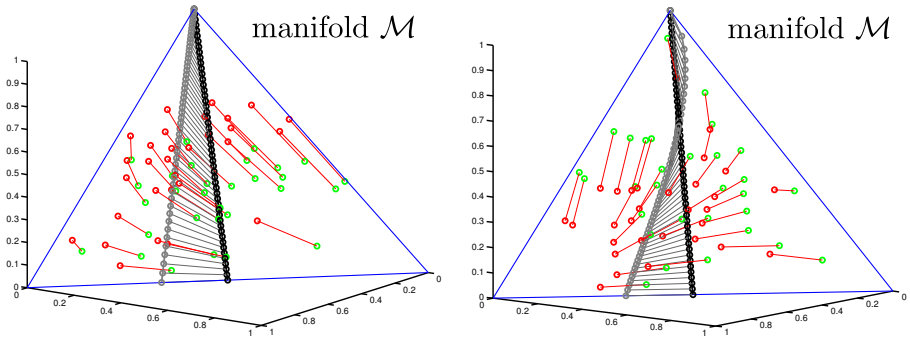
(a) Frobenius (b) pullback-Frobenius

Figure 7.10: Confusion matrices for the YouTube dataset, made up of videos captured 'in the wild'. ***(a)*** *Using the base Frobenius distance, the results are almost close to random (cf. Fig. 7.8(b)).* ***(b)*** *The significant improvement in classification accuracy achieved by the pullback-Frobenius distance is demonstrated by the strong diagonal. Improvements are shown in all classes except 'shooting'.*



(a) Frobenius (b) Bhattacharyya (c) Modified Bhattacharyya

Figure 7.11: Comparing the performance measures achieved by different base distances: Frobenius ***(a)****, Bhattacharyya* ***(b)*** *and modified Bhattacharyya* ***(c)*** *on the YouTube dataset, when using the Fisher representation. Although the Frobenius distance has the worst baseline performance, its pullback version has the best one. In any case, the pullback trick delivers significant improvements across all base distances.*



(a) Line preserving automorphism (b) Another arbitrary automorphism

Figure 7.12: ***(a)*** *An automorphism of the form* $[x, y, z] \mapsto \frac{1}{Z}[\lambda_1 x, \lambda_2 y, \lambda_3 z]$ *with random values for* $\boldsymbol{\lambda}$ *maps the green points on a 2D simplex by a non-linear stretch to the points in red. The linear decision boundary shown in black is mapped to another linear boundary (in grey).* ***(b)*** *Arbitrary automorphisms (here of the form* $\boldsymbol{\lambda}\mathbf{p}^{\boldsymbol{\lambda}}$*) can generate other kinds of curved boundaries.*

the search space towards unknown linearising automorphisms (Theorem A.1), delivering superior performances when coupled with hyperplane classifiers. Extensive testing on even more challenging datasets such as the Hollywood2 [17] and the HMDB [3] ones, employing higher-order HMMs and SVM classification, is in order.

7.5.4 Beyond HMMs

As more sophisticated classes of graphical models able to describe complex activities live on inherently higher-dimensional manifolds, we are potentially met with computational limitations. However, the dimensionality of the parameter space Λ of the automorphism is distinct from that of the model space $\mathcal{M}$ (compare our treatment of the HMM output space). Variable length HMMs [93] are in fact equivalent to HMMs with an exponential state space $2^{\mathcal{X}}$: the approach described here can be straightforwardly applied to them. For hierarchical HMMs [151], described by a set of parameters $\left\{\{\boldsymbol{A}^{q^l}\}_{l=1,\dots,L}, \{\Gamma^{q^l}\}_{l=1,\dots,L}\right\}$, where q^l denotes an arbitrary state at level l, a careful design of the family of automorphisms can much limit computational requirements.

7.5.5 Close-form optimisation by volume element matching

A desirable alternative is to handle classification performance or a related quantity directly *in closed form*. We briefly outline a proposal inspired by Lebanon's

treatment of the unsupervised case in [92]. There he suggested (in the original Riemannian manifold setting) to maximise the inverse volume element associated with a Riemannian metric around the given training set of points:

$$\mathcal{O}(D) = \prod_{\mathrm{i}=1}^{S} \frac{(\det g(\mathbf{m}_\mathrm{i}))^{-\frac{1}{2}}}{\int_M (\det g(\mathbf{m}))^{-\frac{1}{2}}\, dm}, \tag{7.19}$$

where $g(\mathbf{m}_\mathrm{i})$ denotes the metric at the point $\mathbf{m}_\mathrm{i} \in \mathcal{D}$ of the dataset, and $\det g$ the associated Gramian. The latter can be computed as $\det g_*(\mathbf{m}) = \det(\boldsymbol{J})^2 \cdot \det g(\mathbf{m})$, where $\boldsymbol{J}$ is the matrix collecting a basis of push-forward vectors [92]. As it is shown in [152] for the case of stable autoregressive models of order 2, this can be done in closed form whenever the original metric is analytically available.

Using the local Gramians of the training models we can design, in a semi-supervised setting, an objective function which aims at forcing similar points $\mathcal{S}$ to live in the same region of the pullback manifold and dissimilar points $\mathcal{S}'$ to live in different regions, by enforcing the corresponding volume elements $\det g$ to be close for the former and different for the latter. As the volume element is an expression of the local curvature of the manifold, a necessary (albeit not sufficient) condition for two points to be close is that their Gramian be close. Distinct regions of the manifold can still be locally isometric, but averaging over all the points of all the classes mitigates the risk of the optimisation process leading to a minimum generated by accidental isometries. In the semi-supervised case this amounts to solving the optimisation problem

$$\min_{g_*} \mathcal{O}_{semi}(D) = \sum_{(\mathbf{m}_\mathrm{i},\mathbf{m}_\mathrm{j})\in\mathcal{S}} d(G(\mathbf{m}_\mathrm{i}), G(\mathbf{m}_\mathrm{j})) - \sum_{(\mathbf{m}_\mathrm{i},\mathbf{m}_\mathrm{j})\in\mathcal{S}'} d(G(\mathbf{m}_\mathrm{i}), G(\mathbf{m}_\mathrm{j})) \tag{7.20}$$

where $G(\mathbf{m}_\mathrm{i}) = \det g_*(\mathbf{m}_\mathrm{i})$, g_* is a pullback metric, and d is a metric in $\mathbb{R}$. In the supervised case, the following variant in which volume elements are close for training points of the same class c, $c = 1, ..., C$ and different for pairs belonging to different classes can be defined:

$$\min_{g_*} \mathcal{O}_{super}(D) = \sum_{c=1}^{C} \sum_{\mathbf{m}_\mathrm{i},\mathbf{m}_\mathrm{j}\in c} d(G(\mathbf{m}_\mathrm{i}), G(\mathbf{m}_\mathrm{j})) - \sum_{\substack{c,c'=1 \\ c\neq c'}}^{C} \sum_{\substack{\mathbf{m}_\mathrm{i}\in c \\ \mathbf{m}_\mathrm{j}\in c'}} d(G(\mathbf{m}_\mathrm{i}), G(\mathbf{m}_\mathrm{j})). \tag{7.21}$$

When the Gramian of g_* can be computed analytically, the gradient of these objective functions with respect to the parameters $\boldsymbol{\lambda}$ of the pullback metric can also be analytically computed. Both $\mathcal{O}_{super}(D)$ and $\mathcal{O}_{semi}(D)$ can then

be optimised in closed form, at a negligible computational cost. We intend to develop this promising line of research in the future.

In the next chapter, however, we switch gears and move from the task of global action classification to the detection of multiple action classes in space and time. Rather than offline classification on large datasets, we now tackle an altogether more challenging task which has application in human-robot interaction.

Chapter 8

Online learning of multiple concurrent space-time actions

The action detection system described here is composed of several steps carried out incrementally after each frame is captured, including

i) region proposal generation,
ii) ranking of proposals,
iii) building connected space-time tubes,
iv) feature extraction,
v) multi-class online learning, and
vi) detection.

This pipeline takes inspiration from previous works (cf. Section 2.8.1), whose ideas are extended in the following sections towards the application of action detection, as illustrated in Fig. 8.1. Our proposed pipeline starts by generating a set of regions which may contain an object/action of interest.

8.1 Fast region proposal generation

To generate region proposals per frame, we implemented the single strategy selective-search algorithm [109], using HSV-transformed images. The selective-search region-merging similarity score was based on a combination of colour (histogram intersection), and size properties, encouraging smaller regions to merge early, and avoid holes in the hierarchical grouping (HSV, C+S+F). The over-segmentation strategy initialising the hierarchical grouping was performed by Felzenszwalb and Huttenlocher's fast graph based segmentation algorithm [110].

The original selective-search algorithm orders its region hypothesis set according to the order they were generated. To prioritise the regions most likely to

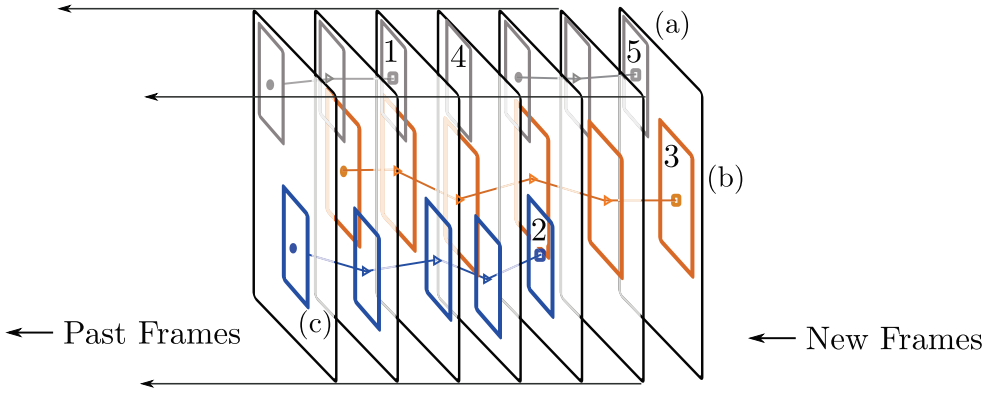


Figure 8.1: Multiple action detection in a video stream. First region proposals are associated in time to generate space-time tubes, with associated tube labels (1-5). During training, tube CNN features are fed to an online learner, whilst during testing, the tube features and learned models are used to generate action label predictions (Grey, Orange, Blue). (a) In this illustration, tubes 1,4 and 5 are labelled as 'grey' (no-action). (b) Tube 3 is labelled as 'orange', and (c) tube 2 as 'blue'. Note that only the current and past video frames are available to the algorithm, and the current state of the algorithm is updated after each new frame is processed (best viewed in colour).

contain an action when working on a computational budget, we attach a score to each hypothesis by calculating the dot product between a learned set of SVM weights and the selective-search colour features '$\mathbf{c}$'. Moreover, we attach a tube label to each region to facilitate the connection of regions into tubes, detailed in the following section. Each region proposal $\mathbf{r}_\mathrm{i}$ in a region set $\mathcal{R}$ represents a vector of connected pixel coordinates which may correspond to an object in the image. In our case, each region $\mathbf{r}_\mathrm{i}$ at time 't' is associated with a score s_i and a tube label l_i:

$$\mathcal{R}^\mathrm{t} = \{(\mathbf{r}_1, s_1, l_1)^\mathrm{t}, \ldots, (\mathbf{r}_M, s_M, l_M)^\mathrm{t}, \mathcal{L}^\mathrm{t}\}, \tag{8.1}$$

where $\mathbf{r}_\mathrm{i}$ defines a connected image region, and $s_\mathrm{i} = \mathbf{w}_\mathrm{s}^T \mathbf{c}_\mathrm{i} + b_\mathrm{s}$, where $\mathbf{c}_i$ is a vector holding the normalised colour histogram extracted from the image region indicated by $\mathbf{r}_\mathrm{i}$. The label set $\mathcal{L}^\mathrm{t}$ holds the unique tube labels in the region set, $\mathcal{L}^\mathrm{t} = \bigcup_{\mathrm{i}=1}^{M} l_\mathrm{i}$. More compactly, by arranging the region indicator vectors $\mathbf{r}_\mathrm{i}$ into a matrix, and the associated scores and labels into vectors,

$$\mathcal{R}^\mathrm{t} = \{\boldsymbol{R}^\mathrm{t}, \mathbf{s}^\mathrm{t}, \mathbf{l}^\mathrm{t}, \mathcal{L}^\mathrm{t}\}. \tag{8.2}$$

When a new region set is created (see Fig. 8.1), only the members $\boldsymbol{R}^\mathrm{t}$, and $\mathbf{s}^\mathrm{t}$ are known, the labels $\mathbf{l}^\mathrm{t}$ need to be estimated from the pair of neighbouring regions sets in time $\{\mathcal{R}^\mathrm{t}, \mathcal{R}^{\mathrm{t}-1}\}$, as detailed in the next section. The online learning of parameters $\{\mathbf{w}_\mathrm{s}, b_\mathrm{s}\}$ is detailed in Section 8.4.

8.2 Selective tubes

The region proposals $\mathcal{R}^{\mathrm{t}}$ are generated independently for each video frame. In order to incrementally connect the proposals into tubes:

$$\mathcal{T}^{l} = \{(\mathbf{r}_1, s_1)^{\mathrm{t}}, \ldots, (\mathbf{r}_N, s_N)^{\mathrm{t}+N}, l, y\}, \tag{8.3}$$

a correspondence problem must be solved to determine which regions in frame 't' correspond to those in frame $\mathrm{t}' \equiv (\mathrm{t}-1)$. From (8.3), a tube is defined by a set of regions in time having the same tube label 'l', and forms the basic bounding structure to which a multi-class action label, or a 'no-action' label 'y' may be assigned.

Let the index sets of regions in frame t' and t be denoted by $\mathcal{U} = \{1, \ldots, M_{\mathrm{t}'}\}$, and $\mathcal{V} = \{1, \ldots, M_{\mathrm{t}}\}$ respectively. Let us also assume initially that there are the same number of regions in each image, $M_{\mathrm{t}'} = M_{\mathrm{t}}$. Now, consider a function f which takes a pair of regions and returns a cost of assigning region $\mathrm{i} \in \mathcal{U}$ to region $\mathrm{j} \in \mathcal{V}$, written as: $f : \mathcal{U} \times \mathcal{V} \mapsto \mathbb{R}$. We now wish to find a bijection $\sigma : \mathcal{U} \to \mathcal{V}$ that minimises the following cost function:

$$\gamma^{*} = \min_{\sigma} \sum_{\mathrm{i} \in \mathcal{U}} f\left(\mathrm{i}, \sigma(\mathrm{i})\right). \tag{8.4}$$

In other words, we would like to find a permutation of the indices in $\mathcal{U}$ via function $\sigma()$, which minimises the total cost γ^{*}, from the $M_{\mathrm{t}}!$ possible permutations. The cost for assigning a region $\mathrm{i} \in \mathcal{U}$ to $\mathrm{j} \in \mathcal{V}$ was chosen to be a combination of the histogram-intersection and intersection-union scores:

$$f(\mathrm{i}, \mathrm{j}) = 1 - \frac{1}{2}\left(\sum_{\mathrm{k}} \min(\mathbf{c}_{\mathrm{i}}^{\mathrm{k}}, \mathbf{c}_{\mathrm{j}}^{\mathrm{k}}) + \frac{(\mathbf{r}_{\mathrm{i}} \cap \mathbf{r}_{\mathrm{j}})}{(\mathbf{r}_{\mathrm{i}} \cup \mathbf{r}_{\mathrm{j}})}\right). \tag{8.5}$$

The optimal assignment problem described by Equations (8.4) & (8.5) may be solved by a variety of combinatorial optimisation algorithms [153], of which we selected the $O(n^3)$ Hungarian method by Kuhn and Yaw [154], where $n = \max(M_{\mathrm{t}'}, M_{\mathrm{t}})$.

In practice the Hungarian assignment algorithm is able to handle rectangular cost matrices by padding the empty locations (see Algorithm 8.1). In the case where $M_{\mathrm{t}} > M_{\mathrm{t}'}$, new labels are assigned to regions without a pairing. When $M_{\mathrm{t}} < M_{\mathrm{t}'}$, a subset of labels in $\mathcal{L}^{\mathrm{t}'}$ will be discarded, marking the end of those tubes. The selective tubes correspondence algorithm is presented in

Algorithm 8.1 Selective tubes algorithm.

Input:
Regions $\mathcal{R}^{\mathrm{t}'}, \mathcal{R}^{\mathrm{t}}$
Features $\boldsymbol{C}_{\mathrm{t}'} = (\mathbf{c}_1, \ldots, \mathbf{c}_{M_{\mathrm{t}'}})$, $\boldsymbol{C}_{\mathrm{t}} = (\mathbf{c}_1, \ldots, \mathbf{c}_{M_{\mathrm{t}}})$, Note $M_{\mathrm{t}'} \neq M_{\mathrm{t}}$ in general.
Output: Tube label vector $\mathbf{l}_{\mathrm{t}} \in \mathcal{R}^{\mathrm{t}}$
Define:
$\mathcal{U} = \{1, \ldots, M_{\mathrm{t}'}\}, \mathcal{V} = \{1, \ldots, M_{\mathrm{t}}\}$
$\hat{M} := \max(M_{\mathrm{t}'}, M_{\mathrm{t}})$
for i := 1 **to** $\hat{M}$, j := 1 **to** $\hat{M}$ **do**
    $\boldsymbol{A}[\mathrm{i}, \mathrm{j}] := \begin{cases} f(\mathrm{i}, \mathrm{j}) & : (\mathrm{i} \in \mathcal{U}) \wedge (\mathrm{j} \in \mathcal{V}), \quad (Eq.\ 8.5) \\ 0 & : \text{otherwise.} \end{cases}$
end for
Permutation matrix $\boldsymbol{P} :=$ `HungarianAssignment`$(\boldsymbol{A})$
if $(M_{\mathrm{t}'} < M_{\mathrm{t}})$ **then**
    for q := 1 **to** $(M_{\mathrm{t}} - M_{\mathrm{t}'} + 1)$ **do** $\mathbf{l}_{\mathrm{t}'}$.`Append(` `NewLabel()` `)` **end for**
end if
$\mathbf{l}_{\mathrm{t}} := \boldsymbol{P}\mathbf{l}_{\mathrm{t}'}$
if $(M_{\mathrm{t}'} > M_{\mathrm{t}})$ **then**
    for q := 1 **to** $(M_{\mathrm{t}'} - M_{\mathrm{t}} + 1)$ **do** $\mathbf{l}_{\mathrm{t}}$.`EraseLast()` **end for**
end if

Algorithm 8.1.

8.3 Feature extraction

To extract features, we used the convolutional neural network (CNN) architecture by [111], which gave state-of-the-art results on the ImageNet database [155]. The output of the CNN may be seen as a highly nonlinear transformation from local image patches to a high-dimensional vector space in which discrimination may be performed by a linear classifier. To obtain an image patch descriptor from the region denoted by '$\mathbf{r}$', we first resize the minimum bounding box enclosing the region to a image of size of 224×224 pixels, and provide the $50,176$ vector of pixel intensities to the input of the CNN network. The network output is 1000-dimensional vector $\mathbf{x}_{\mathrm{cnn}}$, where each element denotes the score for a particular class in the ImageNet dataset; this vector forms our image patch descriptor. Recently Fisher vectors also provided a high dimensional feature space in which *linear* classifiers were used to achieve state-of-the-art results [2]. Whereas the CNN features encode the appearance and texture of image patches, the selective tubes provide a motion vector characterising its movement. The CNN and motion features are aggregated in time over the length of the tube

and provide a fixed-length feature '$\mathbf{x}$' for training an online learner, as detailed in the following section.

8.4 Action category learning

We now consider the task of incrementally learning a model $\mathcal{L}$ to assign a label $y \in \{1, 2, \ldots, Y_{max}\}$ to a space-time tube $\mathcal{T}^l$ (8.3), given a vector of tube features $\mathbf{x} \in \mathbb{R}^n$. Due to the success of linear SVMs when combined with CNN features [112], and their ability to be learned online [75], we train a set of linear SVMs (1-vs-rest) to classify action tube features, where a prediction takes the form:

$$\hat{y}_l = \underset{y}{\operatorname{argmax}} \left(\mathbf{w}_y^T \mathbf{x}_l + b_y \right). \tag{8.6}$$

8.4.1 Class specific feature cache

A class specific feature cache will ensure that the learner may 'remember' previously seen classes $y \in \mathcal{C}$, and update their respective models by sampling from the feature cache. Consider an example set from a video frame,

$$\mathcal{E}^{\mathrm{t}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_E, y_E), \mathcal{Y}^{\mathrm{t}}\}, \tag{8.7}$$

where E is the number of examples in the set, and $\mathcal{Y}^{\mathrm{t}}$ is the unique set of class labels present in the example set, $\mathcal{Y}^{\mathrm{t}} = \bigcup_{\mathrm{i}=1}^{E} y_{\mathrm{i}}$. A class-specific feature set of size F_y:

$$\mathcal{F}_y = \{\mathbf{x}_1, \ldots, \mathbf{x}_{F_y}\}, \tag{8.8}$$

is populated by sampling from the currently observed example set $\mathcal{E}^{\mathrm{t}}$.

8.4.2 Growing the learner set

Consider a set of linear SVMs denoted by $\Theta^{\mathrm{t}} = \{\theta_y^{\mathrm{t}_y}\}$ in which

$$\theta_y^{\mathrm{t}_y} = \{\mathbf{w}, b, \mathcal{H}\}, \tag{8.9}$$

where $\langle \mathbf{w}, b \rangle$ denotes an SVM hyperplane and bias, and $\mathcal{H}$ is a cache of hard examples. Note that $\mathrm{t} \neq \mathrm{t}_y$ in general, since 't' denotes the time relative to the first video frame in a dataset, and 't_y' the time relative to the first example in which action category y was observed. When a new class y^* is seen, a new SVM learner θ_{y^*} is simply added to the set Θ^{t}.

Algorithm 8.2 Multi-class incremental online learning.

Linear SVM set: $\Theta = \emptyset$
Feature class cache: $\mathcal{F}_y = \emptyset$
Previously seen class label set: $\mathcal{C} = \emptyset$
for t := 1 **to** T **do**
  New example set: $\mathcal{E}^{\mathrm{t}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{E^{\mathrm{t}}}, y_{E^{\mathrm{t}}}), \mathcal{Y}^{\mathrm{t}}\}$.
  for all $y \in \mathcal{Y}^{\mathrm{t}}$ **do**
    $\mathcal{F}_y := \mathcal{F}_y \cup \texttt{Sample}(\mathcal{E}^{\mathrm{t}}, y, \textit{sample-size})$
  end for
  $\mathcal{C} := \mathcal{C} \cup \mathcal{Y}^{\mathrm{t}}$.
  if $|\mathcal{C}| > 1$ **then**
    $F_1^t := \texttt{Evaluate}(\theta^t, \mathcal{E}^t)$
    for all $y \in \mathcal{C}$ **do**
      $\mathcal{E}^{\mathrm{t}} := \mathcal{E}^{\mathrm{t}} \cup \texttt{Sample}(\mathcal{F}_y, \textit{sample-size})$
    end for
    for all $y \in \mathcal{C}$ **do**
      if $\theta_y \notin \Theta^{\mathrm{t}}$ **then** $\Theta^{\mathrm{t}} := \Theta^{\mathrm{t}} \cup \theta_y$ **end if**
      $\theta_y^{\mathrm{t}+1} := \texttt{Bsgdnm}(\theta_y^{\mathrm{t}}, \mathcal{E}^t, \textit{batch-size})$
    end for
  end if
end for

The online multi-class incremental learning algorithm is laid out in Algorithm 8.2. Note that we also detect the absence of an action, and therefore the total number of classes is $Y_{max} + 1$. Next, we detail the specifics of the batch stochastic gradient descent used to learn parameters $\langle \mathbf{w}_y, b_y \rangle$ for each action category.

8.4.3 Solving an SVM with batch stochastic gradient descent and hard example mining

In a classical SVM setting, given an dataset set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_D, y_D)\}$, $y \in \{-1, +1\}$, and a vector of parameters $\hat{\mathbf{w}} = (\mathbf{w}, b)$, the following objective function is minimised:

$$o_{\mathcal{D}}(\hat{\mathbf{w}}) = \frac{1}{2}\|\hat{\mathbf{w}}\|^2 + C\sum_{\mathrm{i}=1}^{D} \max(0, 1 - y_{\mathrm{i}}\hat{\mathbf{w}}^T\hat{\mathbf{x}}_{\mathrm{i}}), \tag{8.10}$$

where D is the total number of examples in the dataset, and $\hat{\mathbf{x}} = (\mathbf{x}, 1)$ is augmented to include a bias-multiplier. Equation (8.10) is a convex function that may be solved by a quadratic programming solver [71]. Since the data is streamed in time, we use batch variant of stochastic gradient descent (Algorithm 8.3),

Algorithm 8.3 `Bsgdnm`: batch stochastic gradient descent with hard example mining. This algorithm details a single step towards the minimum of the objective function defined in Equation 8.10.

Input:
SVM hyperplane $\hat{\mathbf{w}}^{\mathrm{t}}$
Hard cache: $\mathcal{H}^{\mathrm{t}}$
Example set: $\mathcal{E}^{\mathrm{t}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_E, y_E)\}, y \in \{-1, +1\}$
if $|\mathcal{H}^{\mathrm{t}}| > 0$ **then**
  $\mathcal{H}^{\mathrm{t}} := \texttt{UpdateCache}(\hat{\mathbf{w}}^{\mathrm{t}}, \mathcal{H}^{\mathrm{t}})$
end if
$\mathcal{H}^{\mathrm{t}} := \mathcal{H}^{\mathrm{t}} \cup \texttt{Sample}(\mathcal{E}^{\mathrm{t}}, \textit{batch-size})$
$\alpha^{\mathrm{t}} = \max\left(\frac{1}{\mathrm{t}}, \textit{min-step}\right)$
$\hat{\mathbf{w}}^{\mathrm{t+1}} := \hat{\mathbf{w}}^{\mathrm{t}} - \alpha^{\mathrm{t}}\big(\hat{\mathbf{w}}^{\mathrm{t}} + C \sum_{(\hat{\mathbf{x}}_{\mathrm{i}}, y_{\mathrm{i}}) \in \mathcal{H}^{\mathrm{t}}} h(\hat{\mathbf{w}}^{\mathrm{t}}, \hat{\mathbf{x}}_{\mathrm{i}}, y_{\mathrm{i}})\big)$

which iteratively updates the parameter vector $\hat{\mathbf{w}}$ by taking a step in the negative direction of the gradient, with respect to a randomised subset $\mathcal{D}' \subseteq \mathcal{D}$ of dataset examples. The subgradient of the SVM objective in (8.10) becomes:

$$\nabla o_{\mathcal{D}}(\hat{\mathbf{w}}) = \hat{\mathbf{w}} + C \sum_{(\hat{\mathbf{x}}_{\mathrm{j}}, y_{\mathrm{j}}) \in \mathcal{D}'} h(\hat{\mathbf{w}}, \hat{\mathbf{x}}_{\mathrm{j}}, y_{\mathrm{j}}), \tag{8.11}$$

where

$$h(\hat{\mathbf{w}}, \hat{\mathbf{x}}, y) = \begin{cases} 0 & : y(\hat{\mathbf{w}}^T \hat{\mathbf{x}}) \geq 1 \\ -y\hat{\mathbf{x}} & : \text{otherwise.} \end{cases} \tag{8.12}$$

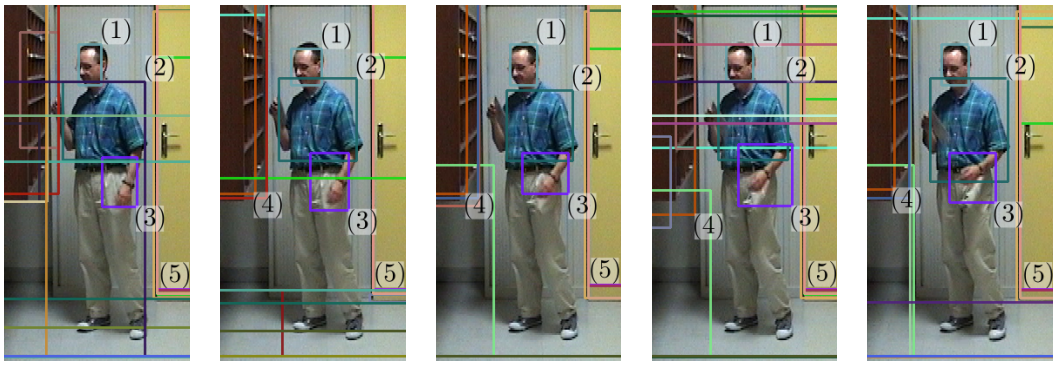
In our application, we replace the static dataset $\mathcal{D}$ with the set of examples per frame $\mathcal{E}^{\mathrm{t}}$. In order to speed up the learning, we keep a cache of hard examples [75], which are defined with respect to the SVM margin by an indicator function:

$$I_{\text{hard}} = \begin{cases} 1 & : y(\hat{\mathbf{w}}^T \hat{\mathbf{x}}) < 1 \\ 0 & : \text{otherwise,} \end{cases} \tag{8.13}$$

where $I_{\text{hard}} = 1$ indicates that the feature $\hat{\mathbf{x}}$ is misclassified or within the margin and therefore 'hard'. The sequence of steps used to learn $\langle \mathbf{w}_y, b_y \rangle$ is laid out in Algorithm 8.3.

8.4.4 Life-long learning

To the best of our knowledge, designing a step size α_{t} for a life-long online learner is still and open problem. In practice we found that restricting α_{t} to a minimum value prevents the learner from converging, allowing the SVM hyperplane to



(a) 202 (b) 203 (c) 204 (d) 205 (e) 206

Figure 8.2: A sequence of images showing the propagation of region proposal labels in time via Hungarian assignment. In this sequence of frames, the head (1), torso (2), and hand (3) of the person are correctly associated in time, although in our tests, we observed that the tracks are not stable over long periods of time. The full sequence may be viewed in an attached supplementary video. Note that there is no region which encloses the whole person in this case, and indication that there is scope to improve the superpixel generation and hierarchical merging.

update itself when a previously unseen class appears. In order to smooth the batch stochastic gradient steps, we use a moving average filter on the gradient updates [156]. Preliminary results of using an online learner are presented in the following section.

8.5 Preliminary tests

In these preliminary tests we aim to analyse key parts of our proposal *before* putting them together in an end-to-end system. A more extensive comparison to competing strategies is left for future work, as outlined in the next chapter.

8.5.1 Selective tubes

The selective tubes algorithm described in Algorithm 8.1 was implemented in `C++`, and is being tested on sequences from the LIRIS-HARL dataset [44]. In Fig. 8.2, we show the effects of associating windows in time to form tubes via Hungarian optimisation. By analogy with Fig. 8.1, we draw the tube label number in brackets next to a set of windows. An attached supplementary video shows additional preliminary results. Note that by using selective tubes, our approach is not limited to action detection, since *all* tubes in the scene are tracked. Thus, this methodology extends itself easily to space-time object detection, and the detection of scene elements without clear boundaries like 'floor' and 'wall'.

Table 8.1: Quantitative results showing the performance of various online linear SVM variations. The mean cumulative accuracy and standard deviation were calculated over 100 experimental runs for each test. Note the significant improvement achieved by adding a class-specific feature cache '$+\mathcal{F}$' and by setting a minimum step size $+\alpha'$.

	`Bsgdnm`	`Bsgdnm` $+ \mathcal{F}$	`Bsgdnm` $+ \alpha'$	`Bsgdnm` $+ \mathcal{F} + \alpha'$
Cum Acc	62.89±9.8	77.94±13.1	74.49±7.0	92.01±8.1

Note that using the graph-based oversegmentation from [128] results in superpixels whose shape and size are very sensitive to the image pixel intensities. Thus, even the same scene imaged a millisecond later may generate a very different set of superpixels. We observed that using SLICO [157] superpixels greatly improved the data association via the Hungarian assignment, although it remains to be seen whether they perform well for generating region proposals via selective-search [109].

8.5.2 Learning of multiple categories streamed online

The online learner detailed in Algorithm 8.2 was tested on a toy dataset, in order compare it to the batch stochastic gradient descent algorithm with hard negative mining (`Bsgdnm`) described in [75]. The online toy dataset was randomly generated with the following parameters:

- 10 maximum number of classes,
- 20 examples per class,
- 2000 maximum number of iterations.

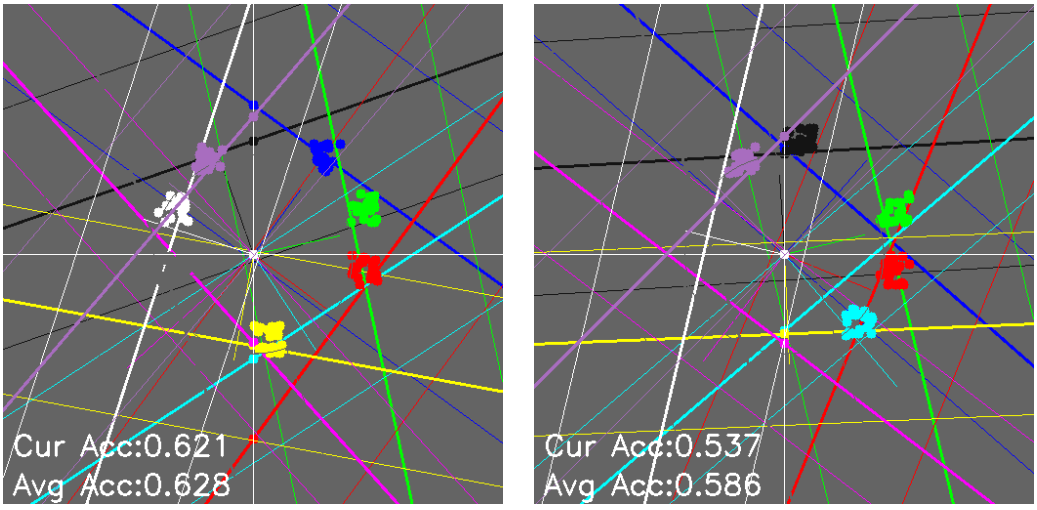
At each iteration, a new example set was randomly generated as follows:

- Initially add a pair of labels to the currently visible class set $\mathcal{C}$;
- every 20 iterations add a new randomly generated label to the set;
- every 40 iterations remove a randomly generated label from the set.

In order to generate random examples, we first select $Y_{max} = 10$ prototype points even spread around the unit ball in $\mathbb{R}^2$. An example point is subsequently generated by adding random noise to the prototype.

The results obtained in terms of the average cumulative accuracy calculated over the 2000 rounds of learning are shown in Table 8.1, where the standard online SVM is denoted as `Bsgdnm`. The addition of a class-specific feature set $\mathcal{F}_y$ of size $F_y = 50$ is denoted by '$+\mathcal{F}$', and the bound on the learning rate $\alpha^{\mathrm{t}} = \max\left(\frac{1}{\mathrm{t}}, \textit{min-step}\right)$, where we set $\textit{min-step} = 8 \times 10^{-4}$ (α remains fixed after $\mathrm{t} = 1250$ iterations), is denoted by $+\alpha'$. An illustration of the 2000^{th} iteration

of learning for each of the method variations is shown in Fig.8.3. Moreover, the output of the SVM learner at each iteration may be viewed in an attached supplementary video.



(a) `Bsgdnm` (b) `Bsgdnm` $+ \mathcal{F}$

(c) `Bsgdnm` $+ \alpha'$ (d) `Bsgdnm` $+ \mathcal{F} + \alpha'$

Figure 8.3: A depiction of the SVM learning 10 categories (1-vs-rest) streamed online after 2000 iterations. (a) `Bsgdnm`*, with a* $\frac{1}{t}$ *learning rate converges over time, and when new classes are added, older hyperplanes are not moved to reflect the new classes. (b) The same scenario is present when including a long term feature cache of previously seen classes* $\mathcal{F}$*. In (c), the learning rate is capped at a minimum value, and therefore, it is able to keep updating the hyperplanes as new classes arrive. However, without a long term class memory, it forgets about previously seen classes that are not currently visible. Lastly in (d), a large improvement in training accuracy is observed by combining the feature cache* $\mathcal{F}$*, with a cap on the learning rate* α'*.*

Chapter 9

Conclusions and future directions

In this dissertation, we proposed a variety of recognition methods for determining what action classes appear in videos. Using local video mid-level representations, we proposed a novel MIL-BoF approach to joint action clip classification *and* localisation based on the recognition of local space-time subvolumes (cf. Chapter 4). Our experiments qualitatively demonstrated that it is possible to localise challenging actions captured 'in the wild', with weak annotation, whilst achieving state-of-the-art classification results. Since in our approach the detection primitives were space-time subvolumes, there was no need to perform spatial and temporal detection separately [16]. Rather, each subvolume was associated with a location in the video, and a decision score for each action class.

We expect that by increasing the number of possible subvolumes and the density at which they are extracted, we will observe further improvements in classification and localisation accuracy. Further MIL performance gains may be obtained by multiple random initialisations, instead of assigning each instance to the label of their parent bag, although at higher computational cost. In addition to MIL, competing approaches by Siva *et al.* [158] showed excellent performance on weakly labelled data using strongly labelled negative examples for explaining the data's inter-category information. Siva *et al.* show improved performance to the MI-SVM formulation of [71] by combining 'negative mining' with saliency measures on object/action detection tasks. More recently, the CRANE algorithm by Tang *et al.* [159] demonstrated further improvements over [71, 158], whilst being robust to noisy labels in the training data. Investigating the importance of each cue in the SVM-map feature vector may also reveal improved mapping strategies.

Our LDSBoF models coupled with Fisher vectors demonstrated the merits of incorporating deformable structure into mid-level feature representations, both

quantitatively (Table 5.1) and qualitatively (Fig. 5.5). Even though our method is independent of the choice of mid-level feature representation, we found that Fisher vectors performed the best when compared to kernel-mapped BoF histograms. By using LDSBoF, we were able to better model the variability of human actions in space-time, which is reflected in higher performance achieved by a 3-part model compared to that of the root filter alone.

In the future, we will focus our attention on describing further action detail, such as whether a person is walking fast or slowly, or whether a person jumps high or low, via attribute learning. Furthermore, we plan to move towards denser feature extraction to capture meaningful mid-level representations with smaller subvolumes, extending LDSBoF to a possibly variable and higher number of parts. Moreover we envision initialising part anchor points from the feature distributions to improve over a fixed grid. Our encouraging localisation results demonstrate the potential for extending this method to larger and more challenging localisation datasets. The ability to assign multiple action labels to a single video clip is also lacking in our framework, and needs to be addressed in the future.

In Chapter 6, we proposed to evaluate aspects of the bag-of-features pipeline previously unexplored, on the largest and most challenging action recognition benchmarks, and achieved state-of-the-art results across the board. In particular, we focused on the feature subsampling and partitioning step after features have been extracted from videos, but prior to encoding. Since uniform random sampling of features to create a vocabulary may be biased towards longer videos and action classes with more data, we compared uniform random sampling to sampling a random and balanced subset from each video and action class. The best results obtained showed that the proposed sampling strategies did yield minor performance improvements, and helped preventing poor results that may follow from sampling an unbalanced set of features (see Fig. 6.3 dotted line 2); other balancing strategies may provide further improvements. We also showed that learning separate vocabularies per feature component caused very large improvements in performance. Furthermore, learning BoF per-category outperformed BoF, but was surpassed by Fisher vectors on the larger and more challenging datasets. Finally, our results demonstrated that competitive results may be achieved by using k-means with a relatively small number of cluster centres K.

When representing videos as dynamical models (cf. Chapter 7), we proposed a distance learning framework for a data-set of hidden Markov models, based on

optimising classification performance over a family of pullback metrics induced by automorphisms. The method is fully general, and extensible to other classes of dynamical models and classifiers. Our results showed a large improvement from base to pullback distances, demonstrating the effectiveness of our framework across multiple features and metrics.

In absolute terms, the approach has room to deliver much better results, closer to the state-of-the art. Firstly, the automorphisms' parameter space may be sampled at finer densities, although at a higher computational cost. Secondly, nearest neighbour classification may be replaced by a more sophisticated methodology (Gaussian kernels, SVMs, etc): note that SVM classification would couple well with our linear separation argument than 1-NN (cf. Sections 2.7 & A.2). Better base distances for HMMs can be explored; for example the Frobenius distance is quite a general-purpose matrix norm with no special relation to Markov models. The potentially significant advantages of using HMMs with more states and higher-dimensional embedding spaces remain to be explored. Furthermore, the method may be extended to other nonlinear automorphisms, and closed-form optimisation, as discussed in section 7.5.5.

For the detection of multiple actions in space and time (c.f. Chapter 8), we proposed an online framework based on the generation of space-time region proposals. Although promising preliminary results were observed using the Hungarian assignment for associating region labels in time, and a long term class memory for adding new classes incrementally, it remains to be rigorously evaluated as an end-to-end system. The method though is general and not only applicable to action detection. Since the space-time tubes effectively track the whole scene in a hierarchy, at varying levels of scale/granularity, it may also be applied to space-time object detection.

Improvements to the current proposal can be made on several fronts. Firstly, it is still unclear whether the selective tubes algorithm we proposed will provide bounding structures which are good enough to enclose an action. Pruning and post-processing may be needed to remove unwanted detections, and connect disjoint tubes [69,160]. Variations on the cost function (Equation 8.5) may provide longer-term association depending on the type of superpixels used, and the type of hierarchical grouping. The selective-tubes may also be cast as Multiple Object Tracking (MOT) [161], where the problem of associating superpixels in time is posed as a structured learning problem, which learns a template for each superpixel that minimises a cost function over a network flow in time. However it is to be seen whether it can be adapted to handle a varying number of superpixels

per frame, and the inclusion/removal of paths at various points throughout the network.

Another promising approach includes an offline method for space-time object detection proposals recently proposed by Oneata *et al.* [162], which works by merging supervoxels. We also would like to investigate the possibility of using a patch-match based algorithm [163] for establishing image region correspondences in time. Van den Bergh *et al.* propose a real-time, online temporal window objectness method for constructing tubes [164], which seems ideal for our application. Furthermore, Xu *et al.* propose a streaming hierarchical video segmentation algorithm [165], which may be adapted to generate online selective tubes. We are currently investigating these routes for action tube generation, as we believe that improvements in the tracking of scene regions will be key to the future of action detection.

The performance of our action recognition approaches has evolved significantly over the course of 3 chapters (cf. Chapters 4- 6). For example on the HMDB dataset we report the following accuracies over Chapters 4- 6 respectively: 29.7%, 37.2% and 50.2%. The first hike in performance may be attributed to the improved encoding of local subvolumes (Fisher-vectors, separate vocabularies per descriptor component), and the use of more general subvolume shapes within which to aggregate features. In Chapter 6, we used global aggregation instead of local aggregation, which meant that there we many more features per histogram (at the cost of losing descriptor locality). The compact global representation per video in Chapter 6 also meant that we could use exact quantisation and SVM learning instead of faster approximate algorithms more suitable for large-scale learning (cf. Chapter 5). Further improvements in accuracy will be obtained by using the latest version of dense trajectory features [142], by further subdividing descriptors and learning separate vocabularies per sub-descriptor (akin to product quantisation [166]), and by stacking Fisher vectors [127, 167] to obtain very high dimensional descriptors. Note that the hugely successful deep neural networks [168] have not yet surpassed the best bag-of-visual-word methods [127].

The visualisation of action saliency (cf. Figure 5.6) is also open to modification; for example, the significance of each visual word may be plotted instead of aggregating the detection scores per subvolume. The highly performing SVM-map technique (cf. Section 4.2) may be adapted to pictorial structures by including the space-time distribution in the mapping feature vector (cf. Equation 4.6). In Chapter 6, the results may only be improved by cross-validation over the SVM regularisation parameter C, and would also get rid of the unsatisfactory overfit-

ting of the learned action models on the YouTube dataset (cf. Figure 6.2). In Chapter 8, the cost-function scoring the association of two regions in time will benefit from an additional term calculating the optical-flow of pixels between regions.

Appendices

Appendix A

Supporting ideas

In this section, we describe some of the supporting ideas at the core of this dissertation. We refer the reader to topics which already have excellent resources available, and elaborate on those for which we feel further clarity is needed in the context of each Chapter.

First off, the Hidden Markov model (HMM) is a generative probabilistic dynamical model which is assumed to be a Markov process with unobserved (hidden/latent) states [169]. Distances between HMMs may be calculated using standard metrics, or redefined to form pullback metrics (cf. Section A.1). The advantage of using a parametrised family of pullback metrics is that, in the case of a linear classifier, it may rectify nonlinear class boundaries (cf. Section A.2). In the case of a nearest neighbour classifier, it suffices that examples of the same class are brought closer together, and further away form examples of distinct classes. A practical demonstration of this idea is shown in Section A.3 for points on a simplex, representing one column of an HMM's transition matrix.

Inspired by vector space-models [170], the bag-of-visual-words/bag-of-features pipeline [33] in computer vision represents images/videos as an un-structured collection or visual words (cf. Section A.4), which can be visualised in videos by assigning a unique colour to each visual word (c.f. Section A.5). Some structure may be incorporated into BoF by using spatial pyramids [77], or by means of pictorial structure models [81] (c.f. Section A.6), which got a significant boost in computer vision thanks to Felzenszwalb and Huttenlocher's linear time algorithm for computing distance transforms (c.f. Section A.7). Last but not least, the success of classification algorithms such as SVMs depend on the underlying data representation (e.g. HoG [103], SIFT [171], BoF [33], CNN features [112]), and therefore it seems apt to end with a reminder of what a representation *is* (c.f. Section A.8).

A.1 Pullback metric

A metric is a function which defines a distance between elements of a set,

$$d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}. \tag{A.1}$$

For the mapping between pairs of elements to a real number to be a valid metric, it must satisfy the following conditions for all $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3 \in \mathcal{X}$:

$$\begin{aligned} d(\mathbf{x}_1, \mathbf{x}_2) &\geq 0 \qquad \text{(non-negativity)} \\ d(\mathbf{x}_1, \mathbf{x}_2) = 0 \iff \mathbf{x}_1 &= \mathbf{x}_2 \qquad \text{(identity of indiscernibles)} \\ d(\mathbf{x}_1, \mathbf{x}_2) &= d(\mathbf{x}_2, \mathbf{x}_1) \qquad \text{(symmetry)} \\ d(\mathbf{x}_1, \mathbf{x}_3) &\leq d(\mathbf{x}_1, \mathbf{x}_2) + d(\mathbf{x}_2, \mathbf{x}_3) \qquad \text{(triangle inequality)} \end{aligned} \tag{A.2}$$

For example in one dimension, the euclidean distance between two points $x_1, x_2 \in \mathbb{R}$ on the real line is given by:

$$d(x_1, x_2) = ||x_1 - x_2||_2 = |x_1 - x_2| \tag{A.3}$$

The space $\mathcal{X}$ equipped with a metric d is called a metric space. By using a bijective function, which defines a one to one mapping between elements of set $\mathcal{X}$ and another set $\mathcal{Y}$, a new metric on the real line may be defined. For example, given a function $f : \mathbb{R} \mapsto \mathbb{R}^2$, we may define a new metric:

$$d^*(x_1, x_2) = ||f(x_1) - f(x_2)||_2 \tag{A.4}$$

where the new distance d^* is defined as the euclidean distance between the mapped points in $\mathbb{R}^2$. The metric defined in A.4 is said to *pull-back* the Euclidean metric on the plane in $\mathbb{R}^2$.

Consider the set of real numbers $x \in [-30 : 0.1 : 30]$, as shown in Fig.A.1(a). If we define f to be:

$$f(x) = \left(\frac{x + x^2}{1 + x^2}, \frac{x - x^2}{1 + x^2} \right), \tag{A.5}$$

then this results in mapping the real line to a (non-closed) circle in $\mathbb{R}^2$, as shown in Fig.A.1(b). The respective pairwise distances between the points in set $\mathcal{X}$ using distance metrics d and d^* are shown in Figs A.1(c) & A.1(d).

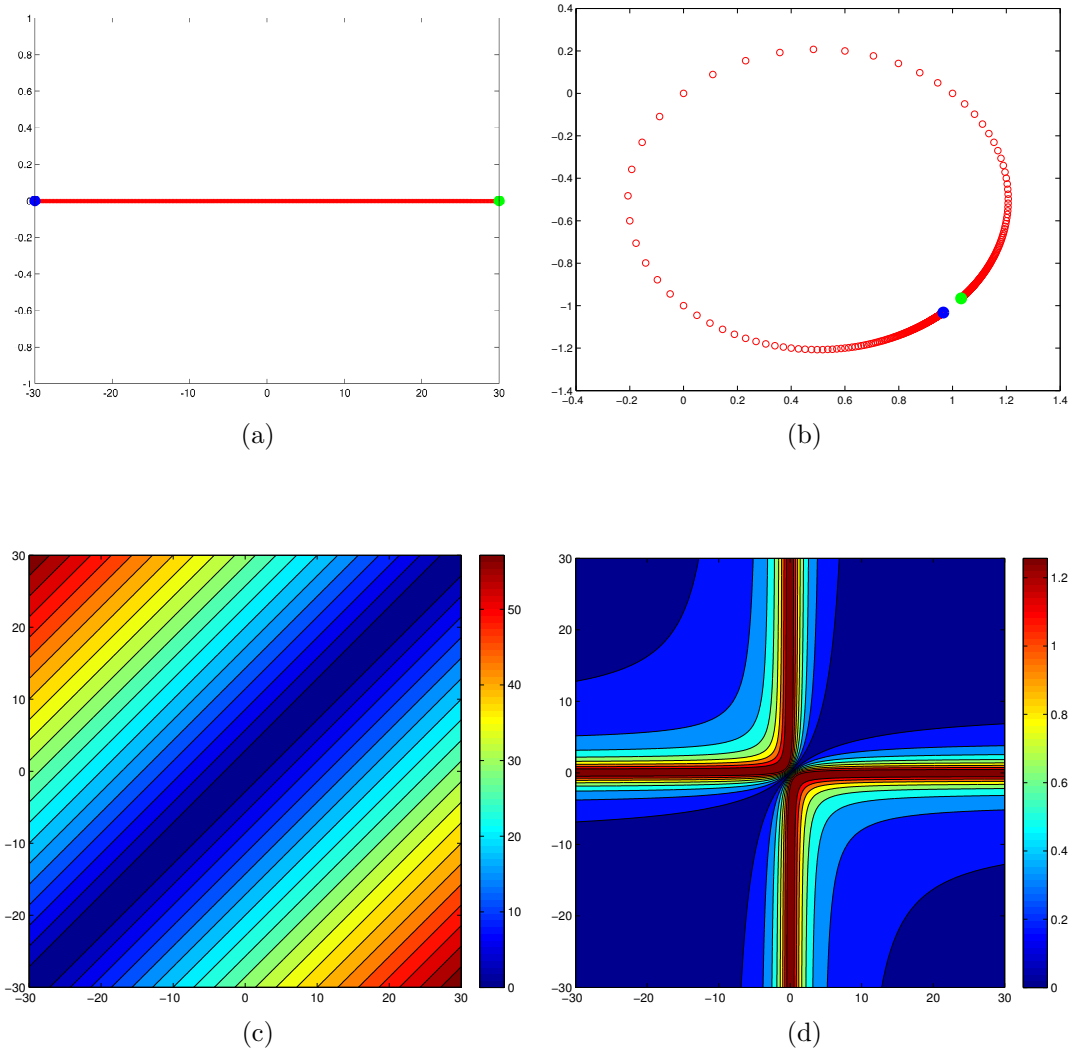
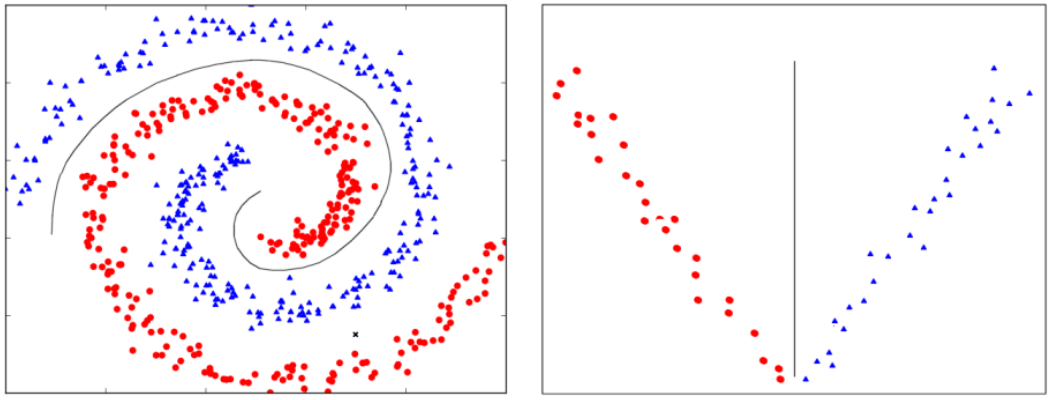


Figure A.1: ***(a)*** *A set of points on the real line.* ***(b)*** *The set of points in (a) mapped onto the Euclidean plane to form a (non-closed) circle.* ***(c)*** *Pairwise (Euclidean) distances between points x on the real line (a).* ***(d)*** *Pullback distances between points on the real line (a) using the reformulation of distance provided in Equation A.4. Notice that the extreme points on the line in (a) (blue and green) are mapped to the blue and green points in (b). In this case, via the pullback metric, the distance between large numbers (positive or negative) has become very small. Note that the criteria for a valid metric have been preserved.*



(a) Original differentiable boundary C (black) (b) Rectified boundary C' after automorphism

Figure A.2: (a) Given any differentiable invertible boundary C in $\mathcal{M} = \mathbb{R}^n$, separating data points of two different classes (e.g. two spirals in $\mathbb{R}^2$, left), there exists an automorphism of $\mathcal{M}$ that maps it to a linear boundary C' (b). Graphical illustration, no real data.

A.2 Rectifying a non-linear boundary

Theorem A.1. *(1-D case). For each differentiable bijective curve $f : \mathbb{R} \to C \subset \mathcal{M} = \mathbb{R}^n$, $\lambda \in \mathbb{R} \mapsto x = f(\lambda) \in C$ there exists an automorphism $F : \mathbb{R}^n \to \mathbb{R}^n$ of $\mathcal{M} = \mathbb{R}^n$ which maps f to a straight line in $\mathcal{M}$ (see Fig. A.2).*

Proof. since f is bijective (the curve C has no self-intersections) with differentiable inverse, there exists $f^{-1} : C \to \mathbb{R}$, $x \in C \mapsto \lambda = f^{-1}(x)$ differentiable. Now, given any versor $\hat{v}$ of $\mathbb{R}^n$ we can design a differentiable mapping $g : \mathbb{R} \to C' \subset \mathbb{R}^n$, $\lambda \mapsto x' = \lambda\hat{v}$ which defines a straight line in $\mathcal{M} = \mathbb{R}^n$, and whose inverse $g^{-1} : C' \to \mathbb{R}$, $x' = \lambda\hat{v} \mapsto \lambda$ is also differentiable. But then $g \circ f^{-1} : C \to C'$ is a differentiable bijective map (whose inverse is $f \circ g^{-1}$) that maps points of C to points of the straight line C': $x \in C \mapsto x' = \lambda(x)\hat{v}$. In conclusion, $g \circ f^{-1}$ is a automorphism from $C \subset \mathbb{R}^n$ to $C' \subset \mathbb{R}^n$: but there exist infinitely many automorphisms F of $\mathbb{R}^n$ whose restriction to C is $g \circ f^{-1}$: $F|_C = g \circ f^{-1}$. □

A.3 Improving classification: automorphisms on a simplex

An automorphism is a bijective structure-preserving map from a mathematical object to itself, and whose inverse exists due to the bijective property. For example consider the unit 2-simplex in $\mathbb{R}^3$. Our 'mathematical object' is the smallest convex set containing the vertices $(1,0,0),(0,1,0)$ and $(0,0,1)$, and forms a subset of $\mathbb{R}^3$. The following structure preserving map (A.6) in this case is designed to be smooth, mapping the differentiable manifold (2-simplex) to itself (automorphism).

Consider the automorphism which stretches the simplical coordinates by a set of normalised weights:

$$F_{\boldsymbol{\lambda}}(\mathbf{s}) = \frac{[\lambda_1 s_1, \lambda_2 s_2, ..., \lambda_N s_N]'}{\boldsymbol{\lambda} \cdot \mathbf{s}}, \tag{A.6}$$

where $\boldsymbol{\lambda} \cdot \mathbf{s}$ denotes the scalar/dot product between the two vectors. This simple transformation defines a family of pullback-metrics. When the points on the simplex are associated with class labels, we may wish to find the best distance function which improves classification performance.

Consider the following example where m points and associated labels (red/green) are generated at random in the simplex (Fig. A.3i). Let the classification algorithm be nearest neighbour, endowed with the Euclidean distance metric. We now seek a distance function between the points in Fig. A.3i which encourages neighbouring points of the same class to have a small distance, whilst neighbouring points of distinct classes to have a large distance.

In this toy example we picked random samples from the parameter space of the automorphisms, and evaluated the classification accuracy by constructing a confusion matrix and dividing its trace by the total sum of its elements (cf. Section 3.1.6). In this case we consider each point in turn as a 'test' point and assign to it the label of the closest point in the remaining set using the Euclidean distance after the transformation by Equation A.6. The Figures A.3a-A.3e illustrate this process. The huge gains in classification accuracy possible motivates the use of this technique.

Consider another nonlinear automorphism which stretches the simplical coordinates by:

$$F_{\boldsymbol{\lambda}}(\mathbf{s}) = \frac{1}{Z}[\lambda_1 s_1^{\lambda_1}, \lambda_2 s_2^{\lambda_2}, ..., \lambda_N s_N^{\lambda_N}]', \tag{A.7}$$

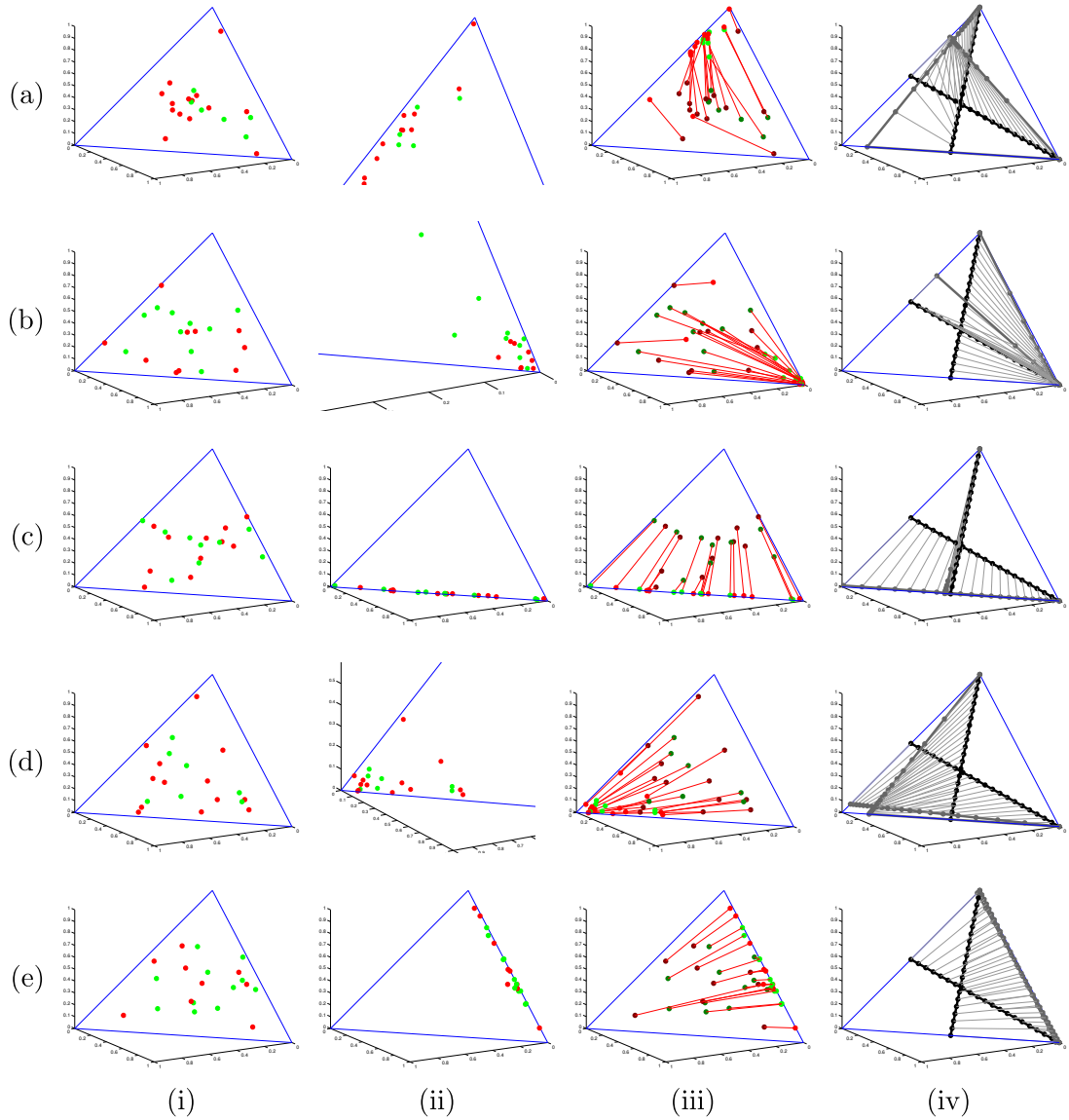


Figure A.3: ***(a-e)*** *Examples in which classification accuracy may be improved by stretching the simplical coordinates using normalised weights. The classification performance gains for (a,b,c) are +35%, (d) +40% (45% to 85%) and (e) +50% (15% to 65%). Column (i) shows the plotted points with corresponding labels in blue/red. (ii) shows the points after the transformation. (iii) plots the points before and after the transformation, with lines connecting the corresponding points. (iv) shows how two lines on the simplex would be mapped under the same transformation (best viewed in colour).*

where $Z = \sum_1^N \lambda_i s_i^{\lambda_i}$. This transformation defines another family of parametrised pullback-metrics, which when optimised upon to improve nearest neighbour classification, also gives high gains in accuracy, as shown in Figs A.4a-A.4c. Note that in this case, lines on the simplex are mapped into curves, unlike those of Fig. A.3.

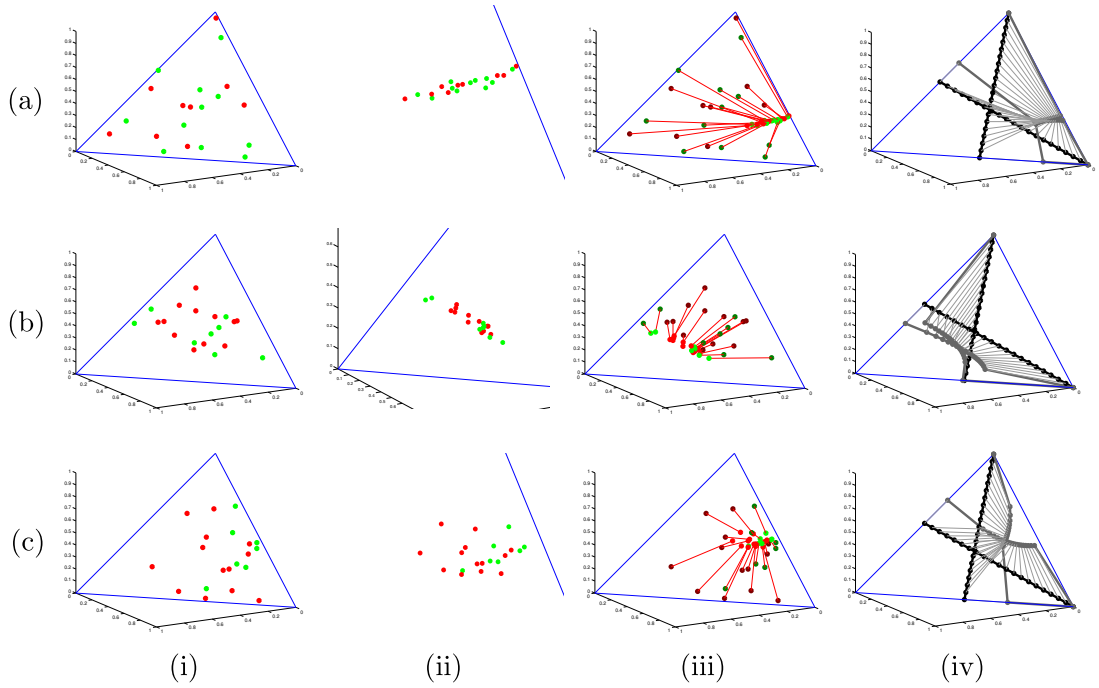
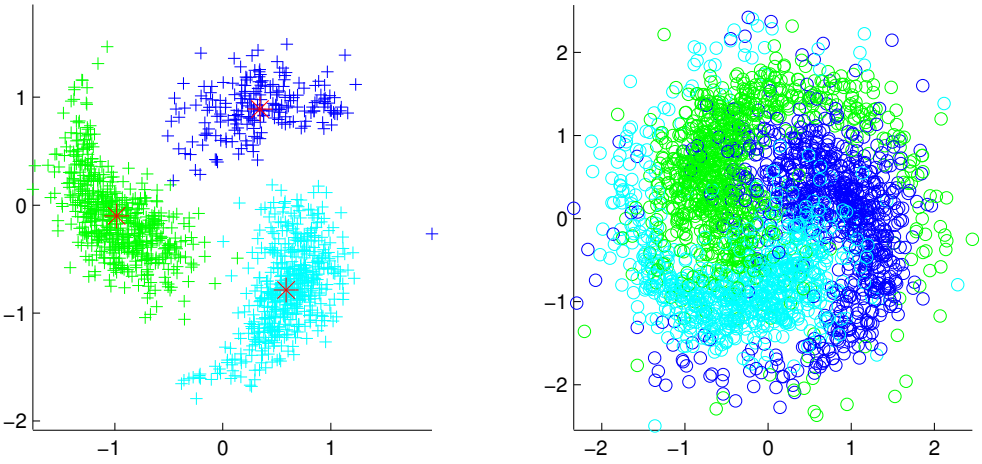


Figure A.4: ***(a-c)*** *Examples in which classification accuracy may be improved by stretching the simplical coordinates using Equation A.7. (a,c) gained +40% in classification accuracy, whilst (b) gained +35%. Note that lines on the simplex may be mapped to curves under this class of transformations (iv). Moreover, in contrast to Fig. A.3, the transformed points giving good classification performance were not found close to the boundaries of the simplex.*



(a) Training data (b) Test data

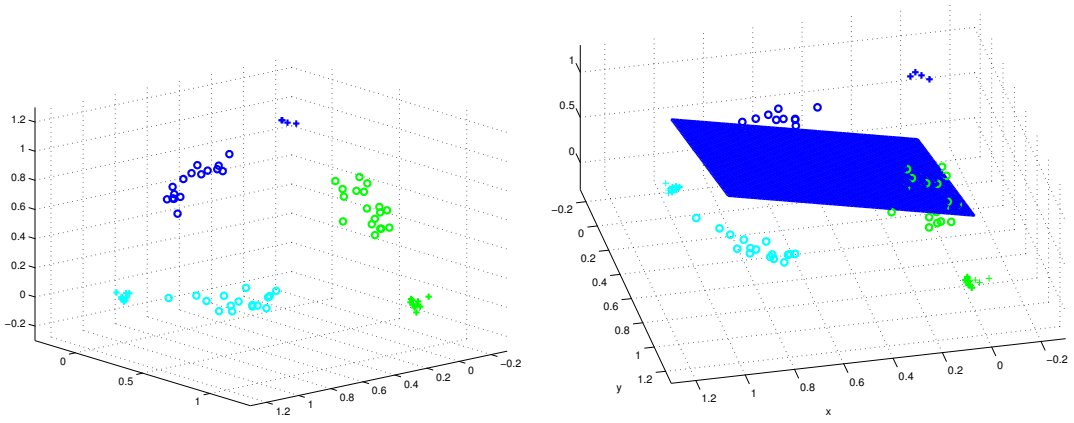
Figure A.5: (a) Training data from 3 *'action' classes foo1 (cyan), foo2 (green), and foo3 (blue) crosses) plotted in* $\mathbb{R}^2$*. There were a total of* 25 *training examples, each having* 50*,* 2*-dimensional features. Drawn as red crosses, a three visual word vocabulary was learnt via the k-means algorithm. The data was initially drawn from a Gaussian distribution and rotated/stretched to form a pinwheel. (b) Test data (cyan, green, and blue circles) drawn from the same distribution as the training data with added noise. Notice the significant overlap of the feature points in distinct classes.*

A.4 Bag of visual words pipeline

To illustrate the highly successful bag of visual words pipeline in action recognition, and get insight into its strength, consider the following toy example. Features are extracted from training examples (videos) and a vocabulary of visual words is generated by clustering the features with the k-means algorithm. These first two steps are illustrated in Fig. A.5(a).

Next, the training features from each example are quantised to the nearest cluster centre in the vocabulary, and a histogram is built based on the frequency of occurrence of these visual words. The histograms are normalised to become invariant to the number of features per example. In Fig. A.6(a) the histograms were $L2$ normalised and can be seen as points on the unit ball in $\mathbb{R}^3$. An action model is subsequently learnt using an SVM per action class, the linear SVM for the blue action is shown in Fig. A.6(b). At test-time, previously unseen examples, as plotted in Fig. A.5(b), are aggregated into a histogram and classified based on the learned model; the test histograms are shown by the hollow circles in Fig. A.6(b).

Even with a significant amount of noise in the testing data, as compared to training, the combination of the nonlinear transformation provided by the histogram, and the linear separator maximising the margin between classes, was



(a) histograms plotted in R3 (b) separating hyperplane

Figure A.6: Plotting the normalised histograms in $\mathbb{R}^3$. (a) After quantising each feature to the closest vocabulary word, and aggregating the features from each example to form a histogram, each example may be plotted as a point in $\mathbb{R}^3$ since each histogram has three bins. Training points are denoted by '+', and test points by 'o'. Notice all points lie on the L2 unit ball. (b) The learned SVM hyperplane for the blue class is displayed together with the train and test data points.

enough to achieve a high classification rate, as shown in Fig. A.7b. Other classification methods are often used, including nonlinear SVMs (Fig. A.7c), and non-parametric classifiers such as Nearest Neighbour (Fig. A.7a). An attached MATLAB script that performs this experiment is available online[1].

[1] `https://sites.google.com/site/mikesapi/downloads/bow_svm_tutorial`

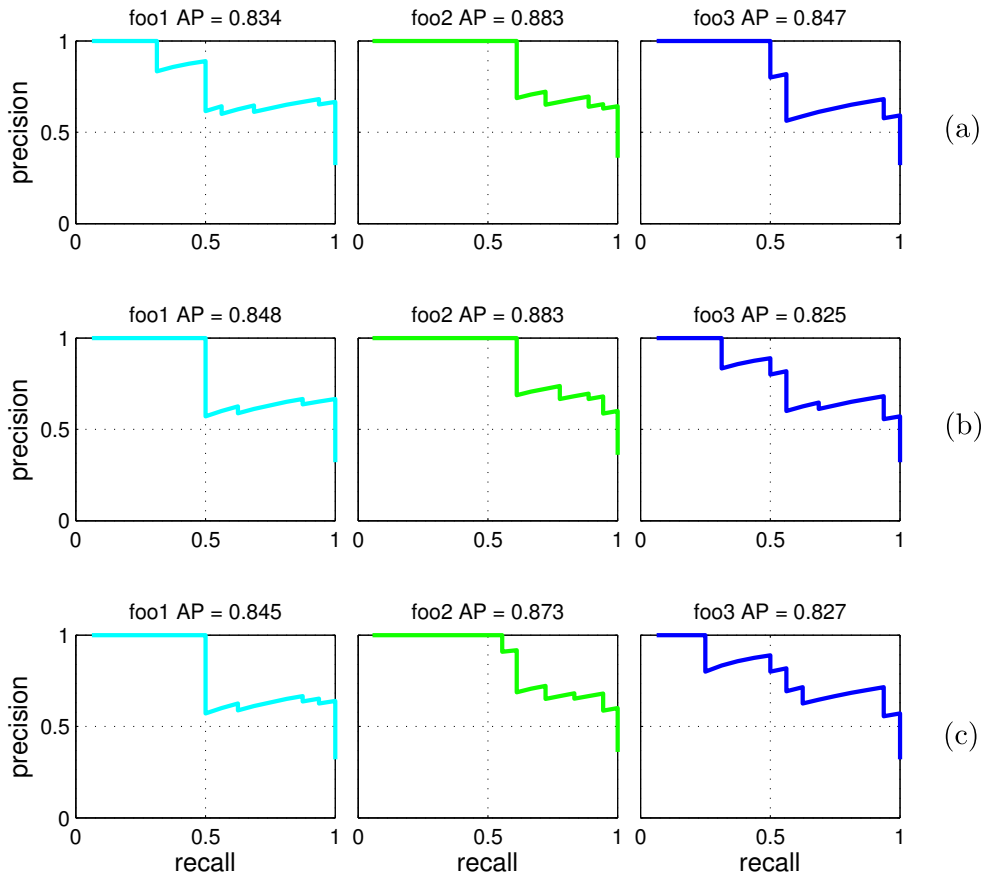


Figure A.7: Precision recall curves per action class (foo1, foo2, foo3) for three competing classifiers. (a) Nearest Neighbour, (b) Linear SVM, and (c) χ^2 kernel SVM.

A.5 Visualising visual words for inspiration

Global bag of visual word models are used to achieve state-of-the art classification performance on challenging video data. Each element in a BoF histogram represents the frequency of occurrence of a particular visual word[2]. Therefore BoF is an unstructured representation; a random permutation on the position of visual words will generate the same histogram.

The pattern of visual words in a video is not arbitrary, however finding a suitable representation to capture the structure of visual words is hard. We hypothesise that the relative positions of groups of visual words may hold important information for recognition. To see why, lets have a look at the data! A video showing four action classes side-by-side together with visual word histograms (see Fig. A.8) is available to watch on YouTube[3]. In the following Figs. A.9 - A.12 from the KTH dataset, the trajectory feature components from [25] have been clustered into a 16-visual word vocabulary. Arrows illustrate the direction of the person's motion in the video. The code used to generate these plots is made publicly available[4].

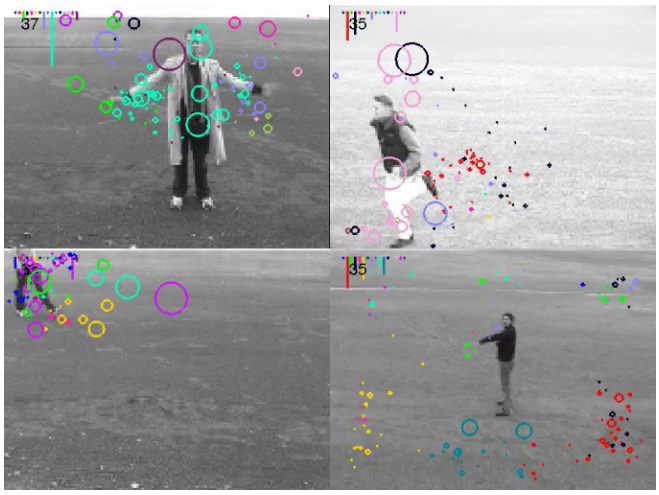


Figure A.8: Visual words from four action classes in the KTH dataset side-by-side. Each circle corresponds to mean position of a space-time feature. Each colour corresponds to a different 'visual word'. The size of the circles corresponds to the scale at which features was extracted. The histogram in top left corner of each video frame shows the frequency of each visual word on that particular frame (un-normalised).

[2] A visual word represents a group of features which are similar (by some distance measure, usually euclidean) in the feature space.

[3] `https://www.youtube.com/watch?v=N9MeDMdi4G8`

[4] `https://sites.google.com/site/mikesapi/downloads`

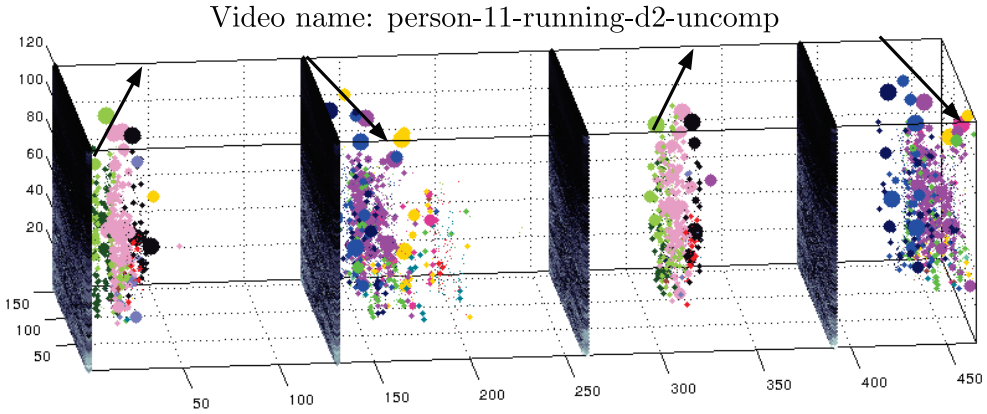


Figure A.9: Visual words from a running class action video plotted in space and time. The frequency of visual words and the spatial arrangement is particular to this action category. The black arrows drawn on top of the video indicate the direction of the person's motion in the video.

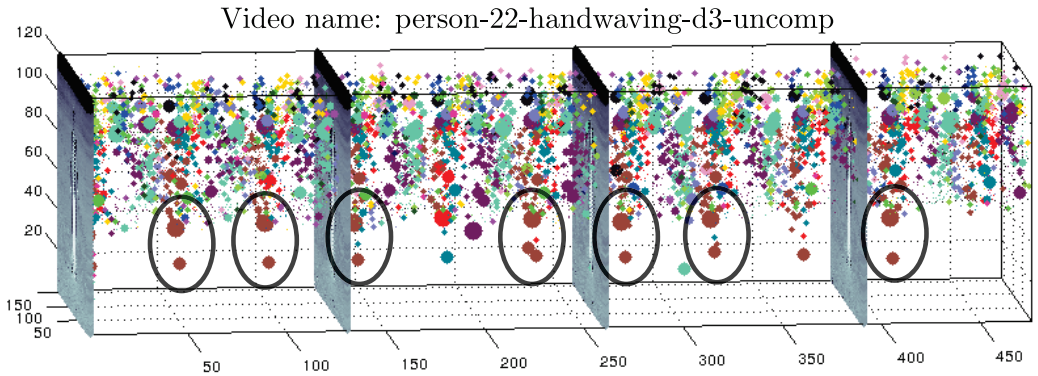


Figure A.10: Visual words from a handwaving class action video. The handwaving motion generates a particular repetitive pattern of visual words; in this case, the upward and downward motions of the hands are described by different coloured visual words. The black ovals drawn over the image are there to highlight a group of brown coloured visual words which appear periodically.

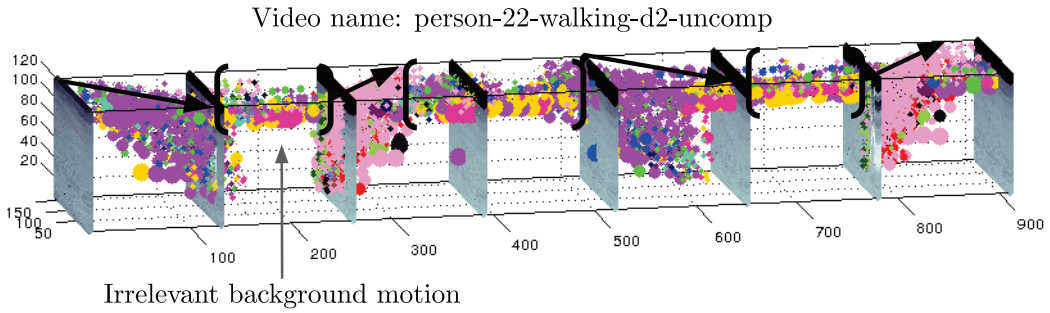


Figure A.11: Visual words from a walking class action plotted in space and time. In this case, the visual words are the same as the ones used for running! However the structure is different, for example the angle the direction of motion makes with the z axis). Note that features are also extracted over irrelevant background motion caused by the points where the grass and sky meet.

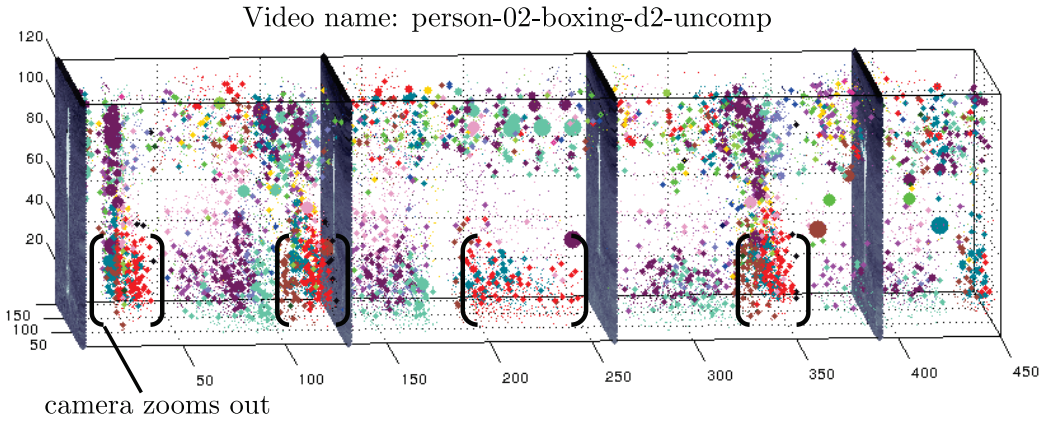
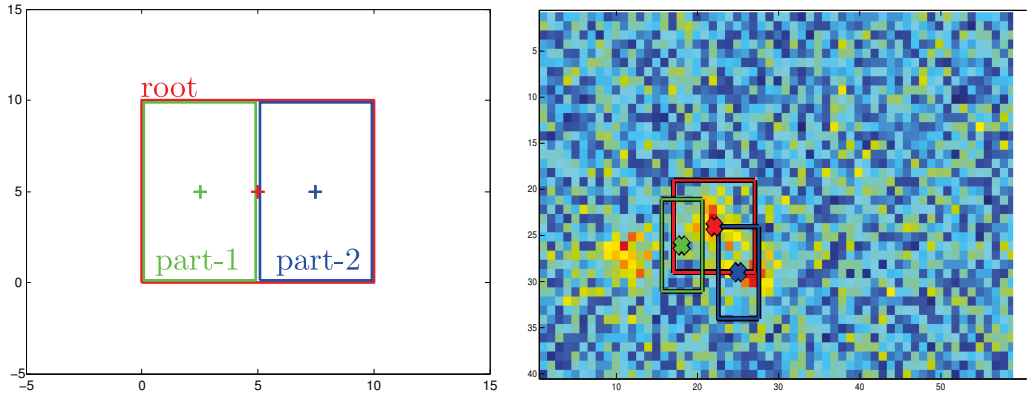


Figure A.12: Visual words from a boxing class action plotted in space and time. In this case one cannot easily see the motion pattern of the boxing action. Most of the visual words represent the apparent motion of the scene when the camera zooms. This is evidence that there still is progress to be made in extracting features for action recognition!



(a) part-1 (b) part-2

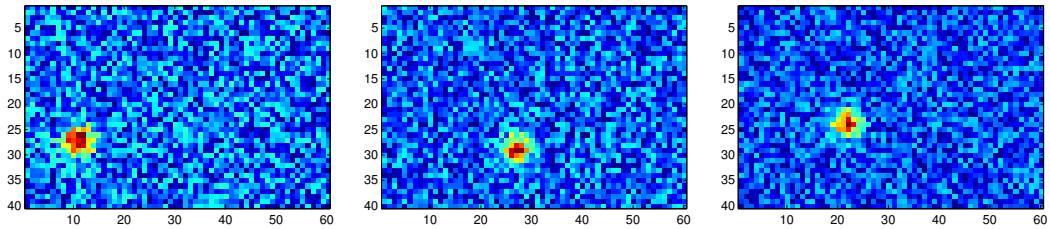
Figure A.13: A 3 part pictorial structure model composed of a root node (red), and two parts (green and blue). The green and blue '+' signs denote the anchor points, the offset position of the part with respect to the root node. The optimal configuration of the pictorial structure model overlaid on the sum of the individual cost functions in Fig. A.14. Notice that the part placements do not coincide with the peaks of the part matching responses, they are the best configuration based on the matching and geometry of the original model.

A.6 Pictorial structure matching

A pictorial structure is composed of a set of geometrically related parts (see Fig. A.13(a)) which may deform with respect to one another. Here we illustrate the matching process, which finds the best configuration of the parts and root, given a cost function measuring how well the part filter '$\mathbf{w}_\mathrm{k}$' matches the image features $\phi(\mathbf{l}_\mathrm{k})$ at location $\mathbf{l}_\mathrm{k}$, and a cost function measuring how well the location of the parts agree with the original model $d(\mathbf{l}_\mathrm{k}, \mathbf{l}_\mathrm{l})$. The overall cost of placing the root at position $\mathbf{l}_0$ is calculated as:

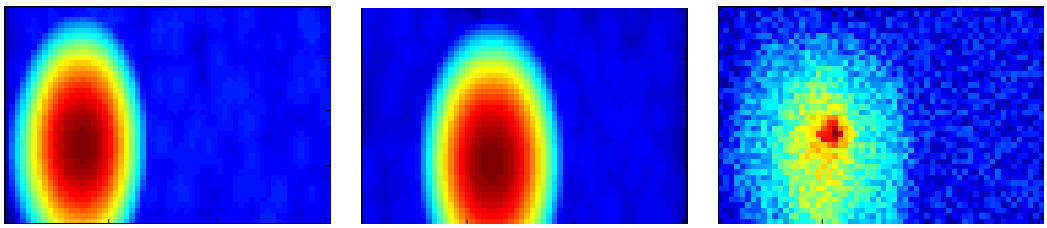
$$s(\mathbf{l}_0) = \max_{\mathbf{l}_1,\ldots\mathbf{l}_P} \left(\sum_{\mathrm{k}=0}^{P} \mathbf{w}_\mathrm{k} \cdot \phi(\mathbf{l}_\mathrm{k}) - \sum_{\mathrm{k}=1}^{P} d(\mathbf{l}_\mathrm{k}, \mathbf{l}_\mathrm{l}) \right). \tag{A.8}$$

The best solution $s(\mathbf{l}_0)^*$ is drawn in Fig. A.13(b), over the original filter responses $\sum_{\mathrm{k}=0}^{P} \mathbf{w}_\mathrm{k} \cdot \phi(\mathbf{l}_\mathrm{k})$. The individual filter responses are drawn in Fig. A.14. The 'distance-transformed' responses with quadratic cost functions are shown in Fig. A.15(a)&(b), whilst the final combined score for the root location is shown in Fig. A.15(c). The final location of the root, including the part offsets from the root is drawn in Fig. A.13(b).



(a) part-1 (b) part-2 (c) root

Figure A.14: Cost function measuring how well each part matches the image for (a) part-1, (b) part-2, and (c) the root. To generate these plots we first picked a random position for the root, and the generated random offsets for the parts. Here we used a 2-dimensional Gaussian distribution centred at a random position around the root to simulate the cost function.



(a) part-1 (b) part-2 (c) root

Figure A.15: Transformed and offset responses for (a) part-1, and (b) part-2. (c) The sum of the parts added to the root response from Fig. A.14(c). The maximum of this function denotes the position of the root. The part positions may be worked out by looking up the optimal part displacements from (a) and (b).

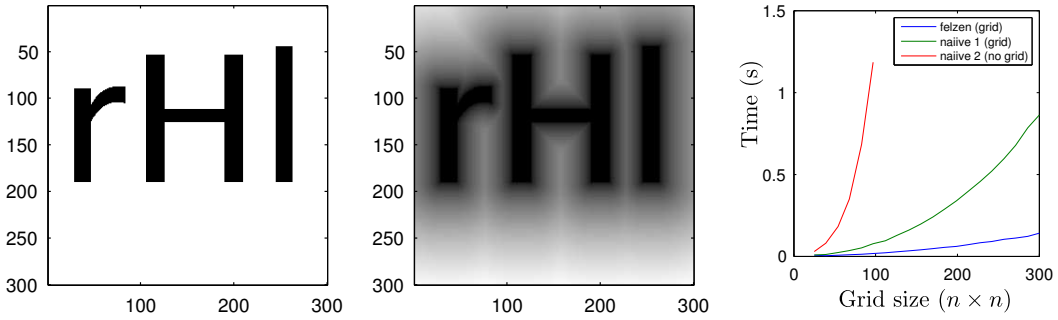


Figure A.16: (a) A binary image showing randomly permuted letters from 'Hello world', whose pixel grid positions form the set $\mathcal{B}$. (b) The distance-transformed image of (a) after solving Equation A.9 for each location z in the image grid $\mathcal{G}$. (c) A time comparison of three algorithms for computing the distance transform in (b).

A.7 Distance transforms

Given a point-set $\mathcal{B} \subseteq \mathcal{G}$, the distance transform specifies for each location z, the distance to the closest point $w \in \mathcal{G}$. For example, considering Fig. A.16(a), let the point set $\mathcal{G}$ be the the set of points defining the image grid, and $\mathcal{B}$ those points in black. Now let us define a function to calculate the distance of each point on the grid image z, to the closest point $w \in \mathcal{G}$ as follows:

$$D_{\mathcal{B}}(z) = \min_{w \in \mathcal{G}}(||z - w|| + I_{\mathcal{B}}(w)), \tag{A.9}$$

where

$$I(w) = \begin{cases} 0 & : w \in \mathcal{B} \\ \infty & : w \notin \mathcal{B}. \end{cases}$$

The distance transformation defined by A.9 finds a location w which is close to z and for which $I_{\mathcal{B}}(w)$ is small. The calculated distances are illustrated as an image in Fig. A.16(b). Now let $I'_{\mathcal{B}}(w)$ be another function, this time defined by a filter evaluated at each pixel as commonly done in object detection. This transformation, when applied to detection, spreads high scores to nearby locations, taking into account the distance from the peak [75]. The proliferation of pictorial structures [128] in computer vision got a major boost by Felzenszwalb and Huttenlocher linear-time algorithm for computing euclidean-distance transforms [129]. The difference their $O(n)$ algorithm makes in practice compared to a naive implementation is shown in Fig. A.16(c). A MATLAB implementation of the distance transform algorithm time comparison is available online[5].

[5] `https://sites.google.com/site/mikesapi/downloads/dt-time-complexity`

A.8 Representation, representation, representation

Jitendra Malik has outlined three of the most important research areas in computer vision, the 3 R's: Recognition, Reconstruction, and Reorganisation. There is also another 'R' thats arguably more fundamental: Representation! For this reason it seems apt to remind ourselves what this means, and at the same time wonder why there has not been more research to rigorously quantify the expressive power of a visual representation for use by machines.

In David Marr's seminar book 'Vision', he eloquently describes what a representation is:

> "A representation is a formal system for making explicit certain entities or types of information, together with a specification of how the system does this".

For example:

> "It is quite proper to think of an image as a representation; the items that are made explicit are the image intensity values at each point in the array, which we can conveniently denote by I(x,y) at coordinate (x,y)",

and another:

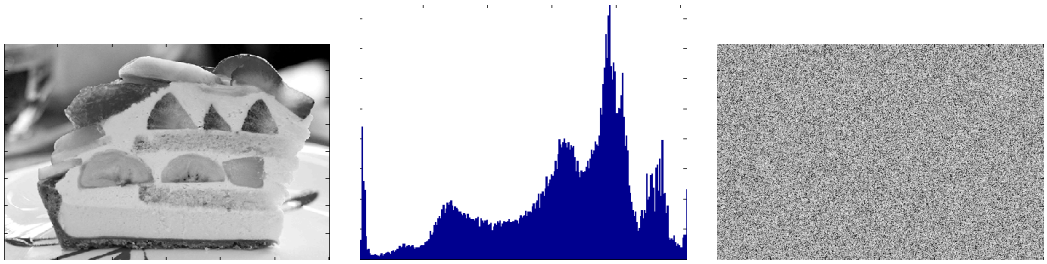
> "..a representation for shape would be a formal scheme for describing some aspects of shape, together with rules to specify how the scheme is applied to any particular shape".

Note that a description is the result of using a representation, and describes a given entity, for example, an instance of an image. This is not to be taken as trivial, and pausing in wonder D. Marr continues:

> "The notion that one can capture some aspect of reality by making a description of it using a symbol and that to do so can be useful seems to me a fascinating idea".

Finally, a word of caution:

> "The choice of which to use is important and cannot be taken lightly. It determines what information is made explicit and hence what is



(a) (b) (c)

Figure A.17: (a) A description of an image representation, (b) its corresponding description as a histogram representation, and (c) an image description that has the exact same histogram description as the image in (a).

> pushed further into the background, and it has a far-reaching effect on the ease and difficulty with which operations may subsequently be carried out on that information".

It may now be appreciated that the need for a 'good' representation for visual data will be crucial to the success of a computer vision system [172]. For instance, in many applications it is desirable to find a representational space in which descriptions of visual data can be compared by a distance metric. In the Computer Vision community, a common way to assess the performance of a representation is to first use it in an machine learning pipeline (e.g. classification) and compare it on the accuracy obtained.

How then may we construct a test criteria for a successful representation, which does not rely on a complete classification pipeline in order to asses its quality? Would this be even useful?

A.8.1 How to assess the quality of a representation?

Since this work makes extensive use of histograms, it seems fit to perform an initial investigation with histograms. A histogram is a simple non-parametric density estimate, and makes explicit the relative frequency of occurring entities. The histogram dimensionality is invariant to number of examples/features, and is the representation of choice for many state-of-the-art classification methods.

In this section, we propose one way to assess the quality of a representation whose descriptions are compared using distance metrics: look at the nearest neighbours of the description! Let the image representation be denoted by $\mathcal{R}_I$, and its description by $\mathbf{d}_I \in \mathcal{R}_I$. Similarly, let a histogram description by denoted by $\mathbf{d}_H \in \mathcal{R}_H$. We propose that in this case, the nearest neighbour to $\mathbf{d}_H$, say $\mathbf{d}'_H$

under distance metric d, should correspond to a description $\mathbf{d}'_I$ which humans see as being similar to $\mathbf{d}_I$, in this case the original image.

For example, consider the image of a cake in Fig. A.17(a), and its grey level histogram in Fig. A.17(b): What does the the nearest neighbour of the histogram look like? A histogram of an $M \times N$ image will have $(M \times N)!$ exact neighbours, since a random permutation of the pixels does not change the histogram.

It follows that the only reason why histograms work in practice is because datasets only contain natural images! Using a histogram representation, a machine cannot tell the difference between (a) and (c); they are exactly the same.

References

[1] I. Laptev and P. Pérez, "Retrieving actions in movies," in *Proc. Int. Conf. Computer Vision*, 2007.

[2] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed Fisher vectors," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2010.

[3] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *Proc. Int. Conf. Computer Vision*, 2011.

[4] M. Sapienza and K. P. Camilleri, "Fasthpe: A recipe for quick head pose estimation," tech. rep., Department of Systems & Control, University of Malta, 2011.

[5] M. Sapienza and K. P. Camilleri, "A generative traversability model for monocular robot self-guidance," in *9th Int. Conf. on Informatics in Control, Automation and Robotics*, 2012.

[6] M. Sapienza, M. Hansard, and R. Horaud, "Real-time visuomotor update of an active binocular head," *Autonomous Robots*, vol. 34, no. 1-2, pp. 35–45, 2013.

[7] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent ASIMO: system overview and integration," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

[8] R. Brooks, L. Aryananda, A. Edsinger, P. Fitzpatrick, C. Kemp, U.-M. OReilly, E. Torres-Jara, P. Varshavskaya, and J. Weber, "Sensing and manipulating built-for-human environments," *International Journal of Humanoid Robotics*, vol. 1, no. 1, pp. 1 – 28, 2004.

[9] W. Chen, C. Xiong, and J. Corso, "Actionness ranking with lattice conditional ordinal random fields," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.

[10] D. Marr, *Vision*. W. H. Freeman and Company, 1982.

[11] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 28, pp. 976–990, 2010.

[12] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer Vision and Image Understanding*, vol. 115, no. 2, pp. 224 – 241, 2011.

[13] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *IEEE Int. Conf. on Pattern Recognition*, 2004.

[14] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Proc. Int. Conf. Computer Vision*, pp. 1395–1402, 2005.

[15] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2008.

[16] J. Liu, J. Luo, and M. Shah, "Recognising realistic actions from videos "in the wild"," in *Proc. British Machine Vision Conference*, 2009.

[17] M. Marszałek, I. Laptev, and C. Schmid, "Actions in context," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2009.

[18] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1798–1828, 2013.

[19] D. Oneata, J. Verbeek, and C. Schmid, "Action and event recognition with Fisher vectors on a compact feature set," in *Proc. Int. Conf. Computer Vision*, 2013.

[20] M. Sapienza, F. Cuzzolin, and P. H. Torr, "Learning discriminative space-time action parts from weakly labelled videos," *International Journal of Computer Vision*, vol. 110, no. 1, pp. 30–47, 2014.

[21] J. Farquhar, S. Szedmak, H. Meng, and J. Shawe-Taylor, "Improving "bag-of-keypoints" image categorisation: Generative models and pdf-kernels," in *Proc. European Conf. Computer Vision*, 2005.

[22] Z. Jiang, Z. Lin, and L. S. Davis, "Recognizing human actions by learning and matching shape-motion prototype trees," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 533–547, 2012.

[23] E. Vig, M. Dorr, and D. Cox, "Space-variant descriptor sampling for action recognition based on saliency and eye movements," in *Proc. European Conf. Computer Vision*, 2012.

[24] O. Kliper-Gross, Y. Gurovich, T. Hassner, and L. Wolf, "Motion interchange patterns for action recognition in unconstrained videos," in *Proc. European Conf. Computer Vision*, 2012.

[25] H. Wang, A. Kläser, C. Schmid, and C. Liu, "Action Recognition by Dense Trajectories," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2011.

[26] H. Wang, M. Ullah, A. Kläser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," in *Proc. British Machine Vision Conference*, 2009.

[27] S. N. Parizi, J. Oberlin, and P. Felzenszwalb, "Reconfigurable models for scene recognition," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2012.

[28] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categirization with bags of keypoints," in *Proc. European Conf. Computer Vision*, 2004.

[29] F. Perronnin, C. Dance, G. Csurka, and M. Bressan, "Adapted vocabularies for generic visual categorisation," in *Proc. European Conf. Computer Vision*, 2006.

[30] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *Proc. European Conf. Computer Vision*, 2006.

[31] Y.-G. Jiang, C.-W. Ngo, and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *ACM International Conference on Image and Video Retrieval*, 2007.

[32] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2006.

[33] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proc. Int. Conf. Computer Vision*, 2003.

[34] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human action classes from videos in the wild," tech. rep., CRCV-TR-12-01, 2012.

[35] R. Chaudhry, A. Ravichandran, G. Jager, and R. Vidal, "Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2009.

[36] S. Ali, A. Basharat, and M. Shah, "Chaotic invariants for human action recognition," in *Proc. Int. Conf. Computer Vision*, 2007.

[37] Q. Shi, L. Cheng, A. Smola, and L. Wang, "Discriminative human action segmentation and recognition using semi-markov model," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2009.

[38] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis, "Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2009.

[39] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal, "Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2009.

[40] R. Elliot, L. Aggoun, and J. Moore, *Hidden Markov models: estimation and control*. Springer, 1995.

[41] M. Do, "Fast approximation of Kullback-Leibler distance for dependence trees and hidden Markov models," *IEEE Signal Processing Letters*, vol. 10, no. 4, pp. 115 – 118, 2003.

[42] N. Shental, T. Hertz, D. Weinshall, and M. Pavel, "Adjustment learning and relevant component analysis," in *Proc. European Conf. Computer Vision*, 2002.

[43] S. Paris, "Edge-preserving smoothing and mean-shift segmentation of video streams," in *Proc. European Conf. Computer Vision*, 2008.

[44] C. Wolf, J. Mille, E. Lombardi, O. Celiktutan, M. Jiu, M. Baccouche, E. Dellandra, C.-E. Bichot, C. Garcia, and B. Sankur, "The LIRIS Human activities dataset and the ICPR 2012 human activities recognition and localization competition," Tech. Rep. RR-LIRIS-2012-004, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, Mar. 2012.

[45] T.-H. Yu, T.-K. Kim, and R. Cipolla, "Real-time action recognition by spatiotemporal semantic and structural forests," in *Proc. British Machine Vision Conference*, 2010.

[46] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *Proc. Int. Conf. Computer Vision*, 2009.

[47] P. Burman, "A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods," *Biometrika*, vol. 76(3), pp. 503–514, 1989.

[48] M. Sapienza, F. Cuzzolin, and P. H. Torr, "Learning discriminative space-time actions from weakly labelled videos," in *Proc. British Machine Vision Conference*, 2012.

[49] M. Sapienza, F. Cuzzolin, and P. H. Torr, "Feature sampling and partitioning for visual vocabulary generation on large action classification datasets," tech. rep., Dept. Computing and Communications Technology, Oxford Brookes University, and Dept. Engineering Science, University of Oxford, 2014.

[50] W. Gong, M. Sapienza, and F. Cuzzolin, "Fisher tensor decomposition for unconstrained gait recognition," in *ECML/PKDD Workshop*, 2013.

[51] F. Cuzzolin and M. Sapienza, "Learning pullback HMM distances," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1483–1489, 2014.

[52] F. Cuzzolin, M. Sapienza, P. Esser, H. Dawes, J. Collett, and M. Franssen, "Identification of parkinsonian gait by means of inertial measurements units via metric learning algorithms," *Gait and Posture (in submission)*, 2014.

[53] I. Laptev and T. Lindeberg, "Space-time interest points," in *Proc. Int. Conf. Computer Vision*, 2003.

[54] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Proc. IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[55] G. Willems, T. Tuytelaars, and L. V. Gool, "An efficient dense and scale-invariant spatio-temporal interest point detector," in *Proc. European Conf. Computer Vision*, 2008.

[56] A. Gilbert, J. Illingworth, and R. Bowden, "Fast realistic multi-action recognition using mined dense spatio-temporal features," in *Proc. Int. Conf. Computer Vision*, 2009.

[57] Y. Ke, R. Sukthandar, and M. Hebert, "Volumetric features for video event detection," *Int. Journal of Computer Vision*, vol. 88, no. 3, pp. 339–362, 2010.

[58] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional SIFT descriptor and its application to action recognition," in *Proc. ACM Multimedia*, pp. 357–360, 2007.

[59] L. Yeffet and L. Wolf, "Local trinary patterns for human action recognition," in *Proc. Int. Conf. Computer Vision*, 2009.

[60] A. Kläser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3D-gradients," in *Proc. British Machine Vision Conference*, 2008.

[61] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *Proc. Int. Conf. Computer Vision*, 2007.

[62] Q. Le, W. Zou, S. Yeung, and A. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2011.

[63] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proc. European Conf. Computer Vision*, 2006.

[64] G. Farnebck, "Two-frame motion estimation based on polynomial expansion," *Image Analysis*, vol. 2749, pp. 363–370, 2003.

[65] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2010.

[66] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2008.

[67] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proc. European Conf. Computer Vision*, 2010.

[68] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, 2011.

[69] A. Kläser, M. Marszałek, C. Schmid, and A. Zisserman, "Human focused action localization in video," in *International Workshop on Sign, Gesture, Activity*, 2010.

[70] S. Wong, T. Kim, and R. Cipolla, "Learning motion categories using both semantic and structural information," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2007.

[71] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *Advances in Neural Information Processing Systems*, 2003.

[72] J. Amores, "Multiple instance classification: Review, taxonomy and comparative study," *Artificial Intelligence*, vol. 201, pp. 81 – 105, 2013.

[73] A. Gaidon, Z. Harchaoui, and C. Schmid, "Activity representation with motion hierarchies," *Int. Journal of Computer Vision*, vol. 107, no. 3, pp. 219–238, 2014.

[74] P. Viola, J. Platt, and C. Zhang, "Multiple instance boosting for object detection," in *Advances in Neural Information Processing Systems*, pp. 1417–1426, 2005.

[75] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[76] Y. Hu, L. Cao, F. Yan, Y. Gong, and T. Huang, "Action detection in complex scenes with spatial and temporal ambiguities," in *Proc. Int. Conf. Computer Vision*, 2009.

[77] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2006.

[78] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce, "Automatic annotation of human actions in video," in *Proc. Int. Conf. Computer Vision*, 2009.

[79] A. Gaidon, Z. Harchaoui, and C. Schmid, "Actom sequence models for efficient action detection," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2011.

[80] A. Gaidon, Z. Harchaoui, and C. Schmid, "Temporal localization of actions with actoms," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2782–2795, 2013.

[81] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Computers*, vol. 22, no. 1, pp. 67–92, 1973.

[82] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," in *Proc. British Machine Vision Conference*, 2011.

[83] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[84] M. Benbouzid, "Bibliography on induction motors faults detection and diagnosis," *IEEE Trans. on Energy Conversion*, vol. 14, no. 4, pp. 1065–1074, 1999.

[85] P. Sykacek and S. Roberts, "Bayesian time series classification," in *Advances in Neural Information Processing Systems*, 2002.

[86] J. J. Rodrguez, C. J. Alonso, and J. A. Maestro, "Support vector machines of interval-based features for time series classification," *Knowledge-Based Systems*, vol. 18, no. 45, pp. 171 – 178, 2005.

[87] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Information processing and technology," ch. Feature-based Classification of Time-series Data, pp. 49–61, Commack, NY, USA: Nova Science Publishers, Inc., 2001.

[88] V. Petridis and A. Kehagias, "Modular neural networks for map classification of time series and the partition algorithm," *Neural Networks, IEEE Transactions on*, vol. 7, no. 1, pp. 73–86, 1996.

[89] D. Piccolo, "A distance measure for classifying ARIMA models," *Journal of Time Series Analysis*, vol. 11, no. 2, pp. 153–164, 1990.

[90] Q. Ding, Z. Zhuang, L. Zhu, and Q. Zhang, "Application of the chaos, fractal and wavelet theories to the feature extraction of passive acoustic signal," *Acta Acustica*, vol. 24, pp. 197–203, 1999.

[91] R. Povinelli, M. Johnson, A. Lindgren, and J. Ye, "Time series classification using gaussian mixture models of reconstructed phase spaces," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, pp. 779–783, June 2004.

[92] G. Lebanon, "Metric learning for text documents," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 497–508, 2006.

[93] A. Galata, N. Johnson, and D. Hogg, "Learning variable-length Markov models of behavior," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 398–413, 2001.

[94] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances in Neural Information Processing Systems*, 1998.

[95] J. D. Lafferty and G. Lebanon, "Diffusion kernels on statistical manifolds.," *Journal of Machine Learning Research*, vol. 6, pp. 129–163, 2005.

[96] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[97] B. Scholkopf, "Statistical learning and kernel methods," Tech. Rep. MSR-TR-2000-23, Microsoft Research, 2000.

[98] G. Lebanon, "Learning riemannian metrics," in *Proc. of the 19th Conference on Uncertainty in Artificial Intelligence*, 2003.

[99] E. Xing, A. Ng, M. Jordan, and S. Russel, "Distance metric learning with applications to clustering with side information," in *Advances in Neural Information Processing Systems*, 2003.

[100] B. Yao and L. Fei-Fei, "Action recognition with exemplar based 2.5d graph matching," in *Proc. European Conf. Computer Vision*, 2012.

[101] C. Desai and D. Ramanan, "Detecting actions, poses, and objects with relational phraselets," in *Proc. European Conf. Computer Vision*, pp. –, 2012.

[102] G. Willems, J. H. Becker, T. Tuytelaars, and L. V. Gool, "Exemplar-based action recognition in video," in *Proc. British Machine Vision Conference*, 2009.

[103] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893 vol. 1, June 2005.

[104] K. Mikolajczyk and H. Uemura, "Action recognition with motion-appearance vocabulary forest," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2008.

[105] Y. Xie, H. Chang, Z. Li, L. Liang, X. Chen, and D. Zhao, "A unified framework for locating and recognising human actions," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2011.

[106] C. Gu, J. Lim, P. Arbelaez, and J. Malik, "Recognition using regions," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, June 2009.

[107] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, pp. 2189–2202, Nov 2012.

[108] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "Bing: Binarized normed gradients for objectness estimation at 300fps," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.

[109] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, 2013.

[110] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *Int. Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[111] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[112] R. Girshick, J. Donahue, T. Darrel, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.

[113] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari, "Learning object class detectors from weakly annotated video," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2012.

[114] J. Chang, D. Wei, and J. W. Fisher III, "A video representation using temporal superpixels," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2013.

[115] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph based video segmentation," 2010.

[116] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[117] D. C. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2012.

[118] 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*, vol. 88, pp. 303–338, June 2010.

[119] M. Rodriguez, J. Ahmed, and M. Shah, "Action MACH: A spatio-temporal maximum average correlation height filter for action recognition," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2008.

[120] A. Kovashka and K. Grauman, "Learning a hierarchy of discriminative space-time neighborhood features for human action recognition," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2010.

[121] K. Schindler and L. V. Gool, "Action snippets: How many frames does human actioin recognition require?," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2008.

[122] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[123] C. Wolf, E. Lombardi, J. Mille, O. Celiktutan, M. Jiu, E. Dogan, G. Eren, M. Baccouche, E. Dellandra, C.-E. Bichot, C. Garcia, and B. Sankur, "Evaluation of video activity localizations integrating quality and quantity measurements," *Computer Vision and Image Understanding*, vol. 127, pp. 14–30, 2014.

[124] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[125] H.-T. Lin, C.-J. Lin, and R. C. Weng, "A note on platts probabilistic outputs for support vector machines," *Machine Learning*, vol. 68, no. 3, pp. 267–276, 2007.

[126] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2010.

[127] X. Peng, C. Zou, Y. Qiao, and Q. Peng, "Action recognition with stacked fisher vectors," in *Proc. European Conf. Computer Vision*, 2014.

[128] P. Felzenszwalb and D. Huttenlocher, "Pictorial structures for object recognition," *Int. Journal of Computer Vision*, vol. 61, no. 1, 2005.

[129] P. Felzenszwalb and D. Huttenlocher, "Distance transforms of sampled functions," tech. rep., Cornell Computing and Information Science, 2004.

[130] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Int. Conf. on Computer Vision Theory and Applications*, 2009.

[131] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms." `http://www.vlfeat.org/`, 2008.

[132] V. Rokhlin, A. Szlam, and M. Tygert, "A randomized algorithm for principal component analysis," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1100–1124, 2009.

[133] R. Elliot, L. Aggoun, and J. Moore, *Hidden Markov models: estimation and control*. Springer Verlag, 1995.

[134] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for SVM," *Mathematical Programming, Series B*, vol. 127, no. 1, pp. 3–30, 2011.

[135] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid, "Towards good practice in large-scale learning for image classification," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2012.

[136] R. Arandjelović and A. Zisserman, "All about VLAD," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[137] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *Int. Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[138] R. Fan, K. Chang, C. Hsieh, X. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

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

[140] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *CoRR*, 2014.

[141] H. Wang, A. Kläser, C. Schmid, and C. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *Int. Journal of Computer Vision*, vol. 103, no. 1, pp. 60 – 79, 2013.

[142] H. Wang and C. Schmid, "Action recogntiion with improved trajectories," in *Proc. Int. Conf. Computer Vision*, 2013.

[143] P. Petersen, *Riemannian Geometry*. Berlin: Springer-Verlag.

[144] B. Hanzon and R. Peeters, "Aspects of Fisher geometry for stochastic linear systems," in *Open Problems in Mathematical Systems and Control Theory*, 2002.

[145] J. M. Lee, *Introduction to Topological Manifolds*. Springer, 2011.

[146] H. Li and X. Li, "LLE based gait analysis and recognition," in *Advances in Biometric Person Authentication*, vol. 3338 of *Lecture Notes in Computer Science*, pp. 671–679, Springer Berlin Heidelberg, 2005.

[147] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290(5500), pp. 2323–2326, 2000.

[148] J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*. Springer, 2007.

[149] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[150] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[151] S. Fine, Y. Singer, and N. Tishby, "The hierarchical hidden Markov model: Analysis and applications," *Machine Learning*, vol. 32, no. 1, pp. 41–62, 1998.

[152] F. Cuzzolin, "Manifold learning for multi-dimensional auto-regressive dynamical models," in *Machine Learning for Vision-based Motion Analysis*, Springer, 2010.

[153] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society of Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[154] H. W. Kuhn and B. Yaw, "The hungarian method for the assignment problem," *Naval Res. Logist. Quart*, pp. 83–97, 1955.

[155] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2009.

[156] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade - Second Edition*, pp. 421–436, 2012.

[157] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274 – 2282, 2012.

[158] P. Siva, C. Russel, and T. Xiang, "In defence of negative mining for annotating weakly labelled data," in *Proc. European Conf. Computer Vision*, 2012.

[159] K. Tang, R. Sukthankar, J. Yagnik, and L. Fei-Fei, "Discriminative segment annotation in weakly labeled video," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2013.

[160] A. Patron-Perez, M. Marszałek, I. D. Reid, and A. Zisserman, "Structured learning of human interactions in tv shows," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2441–2453, 2012.

[161] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1806–1819, Sept 2011.

[162] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid, "Spatio-temporal object detection proposals," in *Proc. European Conf. Computer Vision*, 2014.

[163] S. Gould and Y. Zhang, "Patchmatchgraph: Building a graph of dense patch correspondences for label transfer.," in *Proc. European Conf. Computer Vision*, 2012.

[164] M. Van den Bergh, G. Roig, X. Boix, S. Manen, and L. Van Gool, "Online video seeds for temporal window objectness," in *Proc. Int. Conf. Computer Vision*, 2013.

[165] C. Xu, C. Xiong, and J. J. Corso, "Streaming hierarchical video segmentation," in *Proc. European Conf. Computer Vision*, 2012.

[166] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[167] K. Simonyan, *Large-scale learning of discriminative image representations.* PhD thesis, University of Oxford, 2013.

[168] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014.

[169] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[170] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[171] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. Int. Conf. Computer Vision*, 1999.

[172] P. Meer, "Are we making real progress in computer vision today?," *Image Vision Comput.*, vol. 30, no. 8, pp. 472–473, 2012.