Spatiotemporal Visual Analysis of Human Actions

Antonios Oikonomopoulos

Presented for the degree of Doctor of Philosophy and the Diploma of Imperial College London

May 2010
Abstract

In this dissertation we propose four methods for the recognition of human activities. In all four of them, the representation of the activities is based on spatiotemporal features that are automatically detected at areas where there is a significant amount of independent motion, that is, motion that is due to ongoing activities in the scene. We propose the use of spatiotemporal salient points as features throughout this dissertation. The algorithms presented, however, can be used with any kind of features, as long as the latter are well localized and have a well-defined area of support in space and time. We introduce the utilized spatiotemporal salient points in the first method presented in this dissertation.

By extending previous work on spatial saliency, we measure the variations in the information content of pixel neighborhoods both in space and time, and detect the points at the locations and scales for which this information content is locally maximized. In this way, an activity is represented as a collection of spatiotemporal salient points. We propose an iterative linear space-time warping technique in order to align the representations in space and time and propose to use Relevance Vector Machines (RVM) in order to classify each example into an action category. In the second method proposed in this dissertation we propose to enhance the acquired representations of the first method. More specifically, we propose to track each detected point in time, and create representations based on sets of trajectories, where each trajectory expresses how the information engulfed by each salient point evolves over time. In order to deal with imperfect localization of the detected points, we augment the observation model of the tracker with background information, acquired using a fully automatic background estimation algorithm. In this way, the tracker favors solutions that contain a large number of foreground pixels. In addition, we perform experiments where the tracked templates are localized on specific parts of the body, like the hands and the head, and we further augment the tracker’s observation model using a human skin color model. Finally, we use a variant of the Longest Common Subsequence algorithm (LCSS) in order to acquire a similarity measure between the resulting trajectory representations, and RVMs for classification. In the third method that we propose, we assume that neighboring salient points follow a similar motion. This is in contrast to the previous method, where each salient point was tracked independently of its neighbors. More specifically, we propose to extract a novel set of visual descriptors that are based on geometrical properties of three-dimensional piece-wise polynomials. The latter are fitted on the spatiotemporal locations of salient points that fall within local spatiotemporal neighborhoods, and are assumed to follow a similar motion. The extracted descriptors are invariant in translation and scaling in space-time. Coupling the neighborhood dimensions to the scale at which the corresponding spatiotemporal salient points are detected ensures the latter. The descriptors that are extracted across the whole dataset are subsequently clustered in order to create a codebook, which is used in order to represent the overall motion of the subjects within small temporal windows. Finally,
we use boosting in order to select the most discriminative of these windows for each class, and RVMs for classification. The fourth and last method addresses the joint problem of localization and recognition of human activities depicted in unsegmented image sequences. Its main contribution is the use of an implicit representation of the spatiotemporal shape of the activity, which relies on the spatiotemporal localization of characteristic ensembles of spatiotemporal features. The latter are localized around automatically detected salient points. Evidence for the spatiotemporal localization of the activity is accumulated in a probabilistic spatiotemporal voting scheme. During training, we use boosting in order to create codebooks of characteristic feature ensembles for each class. Subsequently, we construct class-specific spatiotemporal models, which encode where in space and time each codeword ensemble appears in the training set. During testing, each activated codeword ensemble casts probabilistic votes concerning the spatiotemporal localization of the activity, according to the information stored during training. We use a Mean Shift Mode estimation algorithm in order to extract the most probable hypotheses from each resulting voting space. Each hypothesis corresponds to a spatiotemporal volume which potentially engulfs the activity, and is verified by performing action category classification with an RVM classifier.
Acknowledgements

I would like to thank, first and foremost, my advisors, Dr. M. Pantic and Dr. I. Patras for their constant inspiration, expert guidance and constructive criticism throughout the conduction of this thesis. Their constant monitoring and passion for perfection have pushed me in seeking high standards in my work and has made it a great pleasure working with them.

I would like to express my gratitude to my parents, my sister and my brother for their constant support and encouragement in my educational pursuits. Special thanks to my friends, both in the UK and in Greece, for helping me keeping my mind together when needed.

Finally, I would like to express my gratitude to the Greek State Scholarships foundation (I.K.Y.), which funded the initial stages of this research.
To my parents
Contents

Abstract i

Acknowledgements iii

1 Introduction 1

1.1 Main Challenges ....................................... 2
1.2 Motivation and Objectives .................................. 4
1.3 Major contributions and organization of the thesis ................... 6
1.4 Statement of Originality ..................................... 9

2 Related Work 11

2.1 Datasets .................................................. 11
2.1.1 Weizmann dataset ........................................ 12
2.1.2 KTH dataset ............................................. 12
2.1.3 HoHa dataset ............................................. 12
2.1.4 YouTube action dataset ................................... 13
2.1.5 INRIA XMAS dataset .................................... 13
2.1.6 UCF sports dataset .................................... 13
2.2 Representation ............................................... 13
2.2.1 Holistic Representations ........................................ 14
2.2.2 Local Representations ........................................... 21
2.2.3 Tracking Based Representations .................................. 27
2.2.4 Conclusion ...................................................... 31

2.3 Classification ....................................................... 32
2.3.1 Template Matching ................................................ 32
2.3.2 State-Space Approaches .......................................... 37
2.3.3 Conclusion ...................................................... 40

2.4 Activity Detection ................................................... 41

2.5 Conclusions and Discussion ........................................ 42

3 Spatiotemporal Salient Points ......................................... 45

3.1 Spatiotemporal Salient Point Detection .............................. 46

3.2 Salient Regions ...................................................... 49

3.3 Recognition .......................................................... 52

3.3.1 Space-Time Warping .............................................. 53

3.3.2 Relevance Vector Machine Classifier ............................ 54

3.4 Experimental results .................................................. 57

3.5 Conclusions .......................................................... 65

4 Tracking ........................................................................ 69

4.1 Auxiliary Particle Filtering ............................................ 70

4.2 Observation Model ...................................................... 71

4.2.1 Invariant Color Distance ......................................... 71
4.2.2 Background Model ........................................ 73
4.2.3 Skin Model ........................................ 74
4.3 Propagation Model ...................................... 75
4.4 Longest Common Subsequence ............................... 75
4.5 Experimental Results ..................................... 77
  4.5.1 Body Feature Tracking ................................ 77
  4.5.2 Salient Region Tracking ............................... 81
4.6 Conclusions .......................................... 83

5  B-Spline Representations ................................ 85
  5.1 Representation ........................................ 86
    5.1.1 B-spline Surfaces .................................. 87
    5.1.2 Optical Flow ..................................... 88
    5.1.3 Spatiotemporal Descriptors ............................. 90
    5.1.4 Codebook Creation .................................. 94
  5.2 Classification ......................................... 95
    5.2.1 Feature selection ................................... 96
    5.2.2 Relevance Vector Machine classifier ................. 96
  5.3 Experiments .......................................... 97
    5.3.1 Datasets ........................................ 97
    5.3.2 Camera motion .................................... 97
    5.3.3 Classification Results ............................... 98
  5.4 Conclusions .......................................... 103
List of Tables

2.1 Summary of holistic representation methods ........................................... 21
2.2 Summary of local representation methods ............................................. 26

3.1 Recall and Precision rates for the kNN and RVM classifiers .................... 62

4.1 Recall and Precision rates for the compared trackers.......................... 81
4.2 Recall and Precision rates for the RVM classifier.................................. 83

5.1 Successive steps of the proposed approach.......................................... 87
5.2 Recall and precision rates acquired on the three datasets for the three classification experiments................................................................. 99
5.3 Comparisons of the proposed method to various methods proposed elsewhere for KTH dataset ................................................................. 104

6.1 Comparisons of the proposed method to various methods proposed elsewhere for the KTH dataset ................................................................. 122
List of Figures

2.1 (a) A still frame from the Weizmann database and (b) the extracted silhouette of
the depicted subject, acquired by background subtraction. (c),(d) Motion energy and
motion history images respectively, created using 10 frames of the sequence depicted in
(a) .......................................................... 16

3.1 (a) Single frame from a sample image sequence where the subject is raising its right
hand. (b) Entropy plot of the pixels lying within cylindrical neighborhoods applied
around the center of the white circle in (a), as a function of their spatial and temporal
scales. .......................................................... 48

3.2 Still frames of the activities depicted in the aerobics dataset. ....................... 58

3.3 Detected spatiotemporal features in four sample image sequences, corresponding to two
action classes, for five time instances, $t_i, t'_i, i = 1 \ldots 5$. For each class the detected
regions are drawn for two different subjects performing the action. Consistency in the
location and scale of the detected salient regions between the different executions of the
same activity is apparent. ........................................ 59

3.4 Effect of time warping. First column: Reference sequence, second column: space-time
warped sequence, third column: stretched sequence and 4th: original sequence . . . . 60

3.5 RVM Confusion Matrix along with corresponding recall (main diagonal) and precision
rates (last row) for the original aerobics dataset. ........................................ 63

3.6 (a) RVM Confusion Matrix along with corresponding recall and precision rates for the
aerobics2 dataset and (b) RVM Confusion Matrix along with corresponding recall and
precision rates for the Weizmann dataset ............................................. 65
3.7 Detected spatiotemporal features in two misclassified image sequences for three time instances, \( t_i, i = 1 \ldots 3 \). The sequences correspond to two different action classes that are performed by the same subject. ........................................ 66

3.8 Overall recognition rate, with respect to the sigma of the Gaussian noise that was added to the saliency values of the detected salient points prior to clustering. ................. 66

4.1 Examples of estimated backgrounds. Each pixel in the images is the result of the weighted sum of the learnt Gaussian means corresponding to that pixel. .................. 74

4.2 The notion of LCSS. ................................................. 77

4.3 Automatic localization of the hands and the head on the first frame of an image sequence. By subtracting the estimated background in (b) from the first frame in (a), a blob around the subject is extracted, depicted in (c). Subsequently, by combining the result at (c) with the pixels labeled as skin in (d), an estimate over the location of the hands and head is acquired. This is depicted in (e). ...................... 78

4.4 Performance evaluation of the proposed tracking scheme versus Condensation and simple Auxiliary Particle Filtering algorithms. .................................................. 79

4.5 Tracking result for one of the sequences in the database, where the subject is raising its right hand, reaches an apex and returns to the initial position. Green: APF incorporating the proposed observation model. Red: Tracking using only color similarity in the observation model. In the first case, the algorithm is distracted by the background, and tracking is eventually lost. Incorporation of the background and skin models alleviates this problem. .......................................... 80

4.6 Extracted trajectories for two different actions. Failure of tracking is common for regions lying on the boundary between the background and the subject, where the color is uniform, e.g. due to clothing (see (a)). In this case the tracked template moves along the boundary (as is displayed from the dashed circles), resulting in an erroneous trajectory. (Best viewed in color) ................................. 82
5.1 B-spline surfaces of different orders, fitted around a set of control points (black crosses). B-splines of 3rd degree (a) fit well around the control points, as opposed to 6-degree splines (b), which are smoother but fitted further away from them. 88

5.2 (a) A single frame from a handwaving sequence in which camera zoom is occurring and the corresponding optical flow field, before (b) and after (d) the application of the local median filter. Removal of flow vectors that are due to the camera zoom is evident. (c) Some of the salient points detected using the optical flow field in (d). 91

5.3 (a) Points in sequence belonging to an arbitrary curve and (b) the boundary selection. 92

5.4 (a) The set of points $O$ that are engulfed within a spatiotemporal neighborhood. The line connections between the points are for illustration purposes, to depict the ones belonging to the same frame. (b) The resulting B-spline approximation described in section 5.1.1 93

5.5 First derivatives with respect to (a) $u$ and (b) $v$, drawn as three dimensional vectors. 94

5.6 Motion compensation results using local median filters and a global affine model. Top row: Individual frames. 2nd row: Estimated optical flow. 3rd row: Residual flow vectors after locally subtracting the median of the optical flow. Bottom row: Residual flow vectors after the application of robust global affine motion compensation. 98

5.7 Confusion Matrix for the KTH dataset 100

5.8 Confusion Matrix for the Weizmann dataset 101

5.9 Confusion matrix for the Weizmann dataset, where the codebook used for representing the examples was created from the aerobics dataset 103

6.1 Overview of the spatiotemporal voting process. Activated codewords cast spatial and temporal votes with respect to the center and spatial lower bound of the subject and the start/end frame of the action instance. Temporal votes for candidate start/end positions are cast jointly. Local maximums in the spatial and temporal voting spaces are extracted using mean shift and provide estimates for the position of a reference point in each frame of the test sequence and the temporal boundaries of an action instance respectively. 106
6.2 Voting example using a simplified case of an ensemble consisting of just two features. (a) During training, the position $\theta_d$ of the activated ensemble is stored with respect to one or more reference points (e.g. the subject center), along with its average spatiotemporal scale $S_d$. (b) During testing, votes are cast using the stored $\theta_d$ values, normalized by $S_qS_d^{-1}$ in order to account for scale changes. (Best viewed in color.)............. 109

6.3 Ensemble weighting for the temporal case. (a) Ensembles that are activated in a specific phase of the action (as shown in the histogram at the top row) receive a large weight. In the depicted example, the ensemble is localized on the hand, which moves to the right and stops after a few frames. (b) Conversely, ensembles that are commonly activated throughout the action receive a smaller weight. In the depicted example, the activated ensemble corresponds to the constant motion of the hand towards the left. (Best viewed in color.)................................. 113

6.4 Visualization of a feature selection matrix. Selection is performed from 50 features, and using 40 positive and 60 negative examples. Features that have a high likelihood of match (light areas) to the positive examples and a low likelihood of match (dark areas) to the negative examples are eventually selected. .................. 114

6.5 Illustration of the spatiotemporal voting scheme. First row: temporal voting space. Votes towards the start/end frames are cast jointly. Second, third row: Start/end frame projections along lines passing from maximum hypothesis in the temporal voting space. Evidence is accumulated as time progresses, resulting in more votes at the most probable positions. Fifth, sixth row: Spatial voting spaces. Based on its location, each ensemble votes for the most probable position of the center and lower bound of the subject. Fourth row: Fitted bounding boxes resulting from the maximum responses in the spatial voting spaces. (Best viewed in color.)...................... 116

6.6 Confusion Matrix for the KTH dataset. ......................................................... 121

6.7 Confusion Matrix for the HoHa dataset. ......................................................... 121
6.8 Spatial localization results achieved for the subject center and lower bound of the subject, for each class of the KTH dataset. Applying a Kalman filter to the raw outcomes of the mean shift mode estimator lead to a considerable increase in the localization performance. The increase is more prominent for the case of the boxing, handclapping and handwaving classes. x-axis: distance from ground truth annotation in pixels. y-axis: percentage of frames in the database at which the localization estimate’s distance from the ground truth was less or equal to the values in the x-axis.  

6.9 Comparative temporal localization results for the 6 classes of the KTH dataset, between the proposed algorithm (ST-Voting) and the Self-Similarity with Progressive Elimination (SS-PE) algorithm of [1]. x-axis: distance from ground truth annotation in frames. y-axis: percentage of recovered instances.  

6.10 Joint Localization and recognition: ROC curves corresponding to each class of the KTH dataset.  

6.11 Occlusion settings for the boxing, handclapping, handwaving and walking classes. The setting for the jogging and running classes is similar to that of the walking class.  

6.12 Average spatial localization results for the selected occluded sequences. (a) Center of the subject (b) Lower bound of the subject.  

6.13 Localization accuracy achieved for the center and lower bound of the subject when tips of the limbs are occluded.  

6.14 (a) Instance of synthetic sequence depicting two activities. (b),(d) Voting spaces for center and lower bound derived using the model for boxing. (c),(e) Voting spaces for center and lower bound derived using the model for handwaving. Notice that each model favors votes belonging to the activity it was trained for.  

6.15 Average spatial localization accuracy results achieved for the sequences depicting multiple activities. For comparison, the accuracy achieved on the clean sequences is also depicted.  

6.16 Temporal voting spaces corresponding to the image sequence of Fig. 6.14, for (a) boxing and (b) handwaving. Using Mean Shift, 6 instances of boxing are extracted from (a) and 2 instances of handwaving in (b). (Best viewed in color.)
A.1 Regions $X, X_t$ and $X_c$. $X_t$ and $X$ are created by the same, free-form sampling window and and

$X$ is the union of $X_t$ and $X_c$. ........................................ 140
Chapter 1

Introduction

The impressive rise in the performance of modern computer systems and the almost exponential growth of internet in terms of speed and capacity, has given rise to a variety of applications in the last few years, most notably applications that involve a great deal of visual information. Typical examples include high quality digital video, internet television and video sharing websites, like YouTube. Processing of the available visual information is perhaps one of the biggest challenges faced by researchers in the field of computer vision. Due to the huge amount of visual information available, manual annotation and retrieval of videos is practically impossible. This problem becomes even more prominent in applications like surveillance and image/video retrieval. In the former, a human operator is usually needed in order to detect and evaluate the recorded events, a process that can be extremely time consuming, considering the amount of recordings in such systems. In the latter, retrieval is usually performed based on the image/video title, which is manually provided by a human user. A typical example is the retrieval of videos in websites like YouTube, which, at the time this thesis was written, was performed without considering the content of the videos at all. It is apparent, therefore, that an automatic system that would be able to automatically analyze videos based on their content would be highly desirable.

One of the most important aspects in visual information processing is the recognition of human activities. Unarguably, the ultimate goal in this direction is to achieve full analysis and recognition of the human activities that are depicted in input images and/or videos. That is, to reproduce the amazing performance of human visual perception. Ideally, a human activity should be detected regardless of the subject that conducts it, or the environment within which the activity takes place.
Moreover, a robust recognition system should be able to analyze an activity based on its previous experience, that is, based on previously seen examples of the same category. That is, it should be able to perceive its environment, interact with it and increase its knowledge about it through this interaction, much in the same way that humans do. In order to achieve such a performance, the combination of knowledge and reasoning on data coming from sensors like cameras is essential, and is one of the most recent trends in the pursuit for human-like computer vision capabilities. However, the incorporation of reasoning into an artificial vision system is still far from realization.

\section*{1.1 Main Challenges}

Human activity analysis from videos is an area that has received a great amount of interest in the last few years, mainly due to its increasing importance to areas like surveillance, entertainment, human-computer interaction and content-based video retrieval. However, several challenges in the processing of the available information that is contained in image sequences depicting human activities, make this problem very hard to solve.

As an inherently inverse problem, the goal of human activity analysis and computer vision in general, is to estimate the state of a system based on the given visual data. This problem, however, is ill-posed, in the sense that more than one solution may exist, and these solutions may be hugely affected by noise due to e.g. imperfect imaging techniques, variations in illumination, compression artifacts, low resolution of the recordings, etc. Such conditions may cause errors in the extraction of the available information, which can propagate to all subsequent levels of the analysis. Their minimization, therefore, plays a very important role for the development of a robust human activity analysis system. Furthermore, the presence of occlusions, dynamic background, motion of the camera and large viewpoint variations are factors that inhibit the application of activity analysis algorithms to real-world conditions. Modern human activity analysis algorithms, therefore, need to represent and model the information that is engulfed in a scene such that errors coming from such conditions are suppressed.

A major challenge in the analysis of human activities is the large variability that can be observed within the same activity class. For one, differences in the execution speed of the actions by the same or different subjects is an issue that needs to be taken into careful consideration. In cases where constant execution speed can be assumed, modeling such differences can be easily performed.
by assuming a linear warping model. However, such an assumption is not always true, and in many cases it may lead to mappings that are not optimal. In such cases, non-linear methods, like Dynamic Time Warping (DTW) are more suitable [2], since they can model any non-linear warping function, and are therefore more generic. Inter-person and intra-person variations in the execution style of the same action is also an issue that human activity analysis algorithms need to resolve. In order to do so, sufficiently large training sets are required, that are able to cover as many of these variations as possible. By using sufficiently large training sets, features that are characteristic of the specific activity class can be selected, e.g. via feature selection procedures, and features characteristic of a particular execution style can be suppressed. Covering the whole space of the possible variations, however, is very hard, due to the inherent difficulty in acquiring large amounts of data in general. Finally, variations that are due to differences in subject size and appearance also need to be resolved in order for a recognition system to be robust. In the absence of prior information, modeling such variations is usually performed using ad hoc methods, which normalize the extracted features.

Activity detection is another major challenge in the area of human activity analysis. Apart from classifying an activity into an action class, the goal of activity detection is to additionally localize an activity in space and time. Methods that perform both localization and recognition are deemed more suitable for real-world datasets, since the latter are usually minimally processed. By contrast, systems that solely perform recognition use datasets in which a single activity is performed by a single person at each example, that is, they are segmented in time. The main challenges faced by this family of approaches lie in their ability to suppress features that are due to noise, dynamic background, or belonging to activities that do not match the search criteria each time.

The high dimensionality of the input space and the automatic selection of the scale at which activities appear in an input image sequence are additional challenges that human activity analysis algorithms need to address. In the former case, it is well known that not all of the information included in a scene is useful. In fact, a large amount of information that can be extracted can be misleading, since it may belong to areas of the scene that are irrelevant to the purpose of the analysis, like for instance, information belonging to the background. Furthermore, redundancy in the extracted information needs to be reduced in order for the algorithms to be efficient in terms of memory and computer resources usage. The extraction of sparse, local features instead of global ones partly addresses both problems. Various feature selection and dimensionality reduction methods, like boosting and Principal Component Analysis (PCA) respectively, further reduce the amount of information that is essential
for the task at hand, and lead to more discriminative representations for action categories. Automatic
selection of scale, on the other hand, is one of the most difficult problems in the area of computer
vision. Since prior knowledge of the scale is not available, scale estimates are a function of noise and
modeling errors, which are often very hard to discriminate.

Activity recognition algorithms often rely on training, during which, characteristic parts of an activity
are learned with the help of a training set. Learning is usually performed by estimating the parameters
of a statistical model, while in some cases no particular model is used, and the entire training set is
considered to be representative of the action. While the latter approach is simpler, using statistical
models is particularly advantageous, especially in suppressing outliers in the training set, occurring e.g.
due to noise. An important issue in statistical modeling is overfitting, which occurs when the utilized
model fits too well to the training data, but is unable to model previously unseen data. An equally
important issue is the model selection. Gaussian models, for instance, are a very popular choice,
due to their convenient form and relative ease in estimating their parameters (e.g. via Expectation-
Maximization). However, when a chosen model does not correspond to the actual distribution, false
conclusions or predictions can be made. Although a large amount of training data is beneficial to avoid
overfitting, the inherent difficulty is the acquisition of sufficient amounts of data. As a consequence,
modern vision algorithms attempt to train their models, or propose ways, in which good generalization
performance can be achieved with only a few training examples. The benefits of such approaches are
two fold. Apart from using as little data as possible, significantly less time is needed for training,
making a training algorithm very efficient both in terms of memory and computation time. A limited
amount of training data does not pose a significant problem for application specific algorithms, like
algorithms that are designed to work in controlled environments. For more generic applications,
however, care has to be taken in order for the system to be robust and to generalize well.

1.2 Motivation and Objectives

The recognition of human activities from videos, an area in which this thesis belongs, is, among others,
a special branch of computer vision. The goal of human activity recognition is to represent and classify
human activities depicted in input image sequences, into action categories. Typically, a human activity
can be any action performed by a human, ranging from facial expressions to activities involving the
whole body. However, since the former is a separate research area by itself, the term 'human activity'
usually refers to actions like hand gestures or whole body activities, e.g. walking. Arguably, full body activity analysis is a much more challenging problem than facial expression analysis. Indeed, while it is common for face analysis algorithms to assume the relative positions and movement of facial features, similar assumptions cannot be easily adopted when it comes to the whole body. For instance, in the case of the face, features located on the eyes are always placed above features located on the lips. By contrast, parts of the body, like the hands, can move anywhere in relation to e.g., the head.

Current approaches to human activity recognition can be divided into two main categories. In the first category fall approaches that assume that a single subject performing a single activity is present in the scene. The sole goal of this family of approaches, therefore, is to represent and classify the performed activity into an action category. While not very efficient in real-world applications, approaches of this kind are preferable for evaluating novel representation or classification methods, due to the controlled settings of the datasets they use, like static background, static camera settings and frontal subject views. By contrast, approaches that fall within the second category make no such assumptions. That is, they assume that an input image sequence may include more than one subjects performing multiple activities. Such approaches usually include further processing steps, in which an activity is initially localized in space and time and subsequently classified. As such, these approaches are more generic and therefore more suitable for real-world applications.

The first three approaches presented in this thesis fall within the first category. That is, they assume that there is a single subject performing a single activity at an input image sequence. A local approach to activity representation is followed throughout, realized by the automatic detection of a set of patches localized on the subjects. The patches are detected by taking into account variations in the information content of spatiotemporal pixel neighborhoods, and are therefore localized at areas with a significant amount of motion. In the event of noise in the background, like shadows and reflections due to illumination variations, the use of local representations is particularly advantageous in relation to global representations, e.g. ones that are based on the extraction of human silhouettes. In the latter, a global threshold is usually required in order to discard outliers that are due to noise. By contrast, in the utilized local representations, outliers due to e.g, shadows, are automatically assigned a low weight, since the variations in the pixel values within their area of support are minimal. Furthermore, the particular choice of representation allows for the detection of the patches in domains other than the pixels themselves, like for instance using the estimated optical flow of the videos. This approach is followed in the third method of this thesis, where the patches are detected using motion compensated
optical flow vectors, for handling general camera motion. Finally, the use of local patches ensures that the final representation is sparse and compact, with apparent benefits in terms of memory usage.

Due to their generic nature, approaches that fall within the second category mentioned earlier in this section, are more suitable when there is no prior knowledge over the number of subjects in the scene or the number of different activities they perform. Such approaches need to address several additional issues, like general camera motion, varying or cluttered background and the presence of occlusion, either spatial or temporal. These challenges make the use of global representations forbidding, since the latter require the background to be effectively modeled, and the subjects performing the activity accurately localized. Consequently, local approaches like the ones based on sparse spatiotemporal patches are more suitable, and is the kind of representation that is also followed in the fourth approach presented in this thesis. Using such a representation, a probabilistic framework providing estimates over the spatiotemporal localization of an activity is formulated. More specifically, the estimates are acquired via a spatiotemporal voting process, where each vote depends on when and where an observed patch appeared in the training set. This approach has apparent benefits when there is a considerable amount of occlusion, since in this way good estimates can be acquired as long as a good portion of the activity is visible, similar to the ability of humans in detecting a known, partly observed event.

1.3 Major contributions and organization of the thesis

In this thesis we focus on the representation and recognition of human activities that are depicted in input image sequences. As mentioned earlier, in order to address issues like representation compactness and robustness to varying camera conditions, we follow a local approach throughout this thesis. The remainder of this thesis is as follows. In chapter 2 we give an overview of the related work in the field of human activity representation and recognition. In chapter 3 we propose a novel set of sparse features, that are based on automatically detected spatiotemporal salient points [3]. The main contributions of this work can be summarized as follows:

- we propose a new set of spatiotemporal features, that are an extension in time of the spatial salient point detector developed by Kadir and Brady [4]. The proposed features are detected by measuring the changes in the information content of pixels that fall within local spatiotemporal neighborhoods. This leads to a sparse representation of a human activity, where each point corresponds to areas where there is a significant amount of motion.
• we develop a novel method for aligning pairs of image sequences, using their spatiotemporal salient point representations. The proposed method is based on a gradient-descent process, which minimizes a distance measure between the representations, by compensating for scale changes in space and time.

• we propose new kernels for use by a Relevance Vector Machine (RVM) [5] classification scheme, which are specifically tailored to the proposed spatiotemporal salient point representations. The basis of these kernels is the optimized distance measure of the proposed sequence alignment step.

In chapter 4 we enhance the spatiotemporal salient point representations of the previous chapter by using tracking [6][7]. More specifically, we propose to use trajectory based human activity representations, where each trajectory is acquired by tracking feature points located on a subject performing an activity. The latter are either localized on automatically detected spatiotemporal salient points or at characteristic parts of the human body, like the hands and the head. The main contributions of this work can be summarized as follows:

• we propose a new representation for human activities, based on sets of short trajectories. The latter express the evolution in time of automatically detected salient points, and are acquired by tracking in time each salient point for a number of frames.

• we propose to enhance the observation model of the utilized tracker in order to increase its accuracy. More specifically, we propose to augment the utilized observation model with information about the background. By this, we increase the robustness of the tracker to imperfect localization of the salient points, by forcing the tracker to favor solutions that contain a large number of foreground pixels. In the case where the tracked templates are localized on skin regions of the body (e.g. the hand and the head), we further augment the observation model with skin color information. By doing so, the tracker also favors solutions which contain a large number of skin pixels.

In chapter 5 we present the third of the proposed methods, where we assume that neighboring salient points follow a similar motion [8]. This is in contrast to the method of chapter 4, where each salient point was tracked independently of its neighbors. More specifically, we propose to extract a novel set of visual descriptors that are based on geometrical properties of three-dimensional piece-wise polynomials, where each polynomial is fitted around spatiotemporal salient points that fall within local
spatiotemporal neighborhoods, and are assumed to follow a similar motion. The main contributions of this work can be summarized as follows:

- through the use of piecewise polynomials, we establish temporal correspondences between salient points in neighboring frames. We do so by assuming that neighboring salient points follow a similar motion.

- we propose to extract a new set of visual descriptors on the fitted polynomials, that characterize the local space-time shape of the conducted activity. These descriptors are robust to motion induced by a moving camera, due to the use of filtered optical flow for the detection of the salient points on which the polynomials are fitted.

- we propose a feature selection step, where the most informative temporal slices are selected for each class. These slices express the short term motion of the subject performing an activity, and consist of the set of polynomials that are localized within these slices.

In chapter 6 we address the joint problem of spatiotemporal localization and recognition of human activities, by proposing a framework that is based on the spatiotemporal localization of ensembles of spatiotemporal features. The latter are localized around automatically detected spatiotemporal salient points. The main contributions of this work can be summarized as follows:

- we propose an extension in time of the implicit shape model of Leibe et al. [9]. This leads to the creation of a spatiotemporal shape model, which allows us to perform localization both in space and in time.

- we propose to use feature ensembles in the proposed model, instead of single features.

- we propose a novel weighting scheme, in which votes from ensembles that are informative (i.e. they are characteristic of the phase of the action) are favored, while votes from ensembles that are commonly activated (i.e. they are activated in many phases of the action) are suppressed.

- since spatiotemporal votes are accumulated from each observed ensemble in the test set, the proposed method effectively deals with occlusion, as long as a portion of the action is visible. Moreover, the use of class-specific codebooks and spatiotemporal models in a voting framework enables us to deal with the presence of dynamic background and with activities that occur simultaneously.
Finally, in chapter 7 final conclusions are drawn. In the same chapter, we discuss limitations of the proposed methods, and give several directions for future work in the area of human activity analysis.

1.4 Statement of Originality

I hereby declare that this thesis is my own work and contains no material previously published or written by another person, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at Imperial College London or elsewhere, is explicitly acknowledged.
Chapter 2

Related Work

In this work, we consider the problem of assigning labels of action classes to videos of human motion. This area of research has received a huge amount of interest in the last few years, motivated by the particularly impressive increase in the use of multi-camera and multimedia applications. Typical examples include automatic annotation of videos, allowing efficient search and retrieval, online surveillance and monitoring applications, and interactive applications, like human computer interaction and games.

In this chapter we provide an overview of the research findings in the field of human activity analysis that have been reported in the literature so far. We treat the problem as a combination of two steps, namely, representation and subsequent classification. After we review, in section 2.1, the most common datasets that are used in this particular research area, we discuss these two steps in sections 2.2 and 2.3 respectively. In addition, we also discuss in section 2.4 methods that deal with action detection. The goal of these methods is the spatiotemporal localization of human activity instances in an image sequence and their subsequent or simultaneous classification into an action category. Finally, we conclude the chapter in section 2.5.

2.1 Datasets

The apparent need for comparison between different human activity recognition methods has led to the creation of several publicly available datasets of human activities. In this section we review the most commonly used ones.
2.1.1 Weizmann dataset

The Weizmann dataset of human actions was proposed by Blank et al. [10]. It contains 10 different activities, namely walk, run, jump, gallop sideways, bend, one-hand wave, two-hands wave, jump in place, jumping jack and skip, each of which is performed by 9 or 10 subjects. The main characteristics of this dataset is the static and relatively uniform background, and the static position of the camera. In addition to this dataset, the same authors provide a set of walking sequences for robustness evaluation against different viewpoints and against several variations, like occlusions, walking with objects (e.g. a briefcase) and different styles of walking.

2.1.2 KTH dataset

The KTH dataset of human actions was proposed by Schüldt et al. [11], and contains 6 different actions: boxing, hand-clapping, hand-waving, jogging, running, and walking. Each action is performed by 25 subjects several times under different conditions. These include scale changes, indoors/outdoors recordings, and varying clothes. The main challenges in this dataset include small camera motion (mainly camera zoom and translation), noise in the otherwise uniform background, shadows, and large variability in the conduction of the activities by the subjects.

2.1.3 HoHa dataset

Originally proposed by Laptev et al. [12], the Hollywood Human Actions dataset (HoHA) contains video samples of human actions from 32 movies, and is one of the most challenging ones in the area of human activity recognition. Each sample is labeled according to one or more of 8 action classes: AnswerPhone, GetOutOfCar, HandShake, HugPerson, Kiss, SitDown, SitUp, StandUp. The main challenge of this dataset is the huge variability of the actions depicted, due to different lighting conditions, different view-points, cluttered and dynamic background and significant camera motion. A second version of this dataset is proposed in [13], and contains 4 additional classes, namely DriveCar, Eat, Fight and Run. Furthermore, the number of the examples in the original 8 classes has been increased.
2.1.4 YouTube action dataset

Introduced by Liu et al. [14], the YouTube action dataset contains 11 different activities, namely, *basketball shooting, biking/cycling, diving, golf swinging, horse back riding, soccer juggling, swinging, tennis swinging, trampoline jumping, volleyball spiking,* and *walking with a dog.* The videos in the dataset were collected from YouTube or from personal videos. Each class in the dataset is divided into 25 groups, depending on the actor performing the activities, background, viewpoint and so on. The main challenges of this dataset include object appearance and pose, object scale, camera viewpoint, background, illumination conditions, etc.

2.1.5 INRIA XMAS dataset

The INRIA XMAS multi-view dataset of human actions was proposed by Weinland et al. [15], and contains 14 different actions (*check watch, cross arms, scratch head, sit down, get up, turn around, walk, wave, punch, kick, point, pick up, throw over head and throw from bottom up*), performed by 11 different subjects under 5 different camera view-points. Its main characteristics include static background, fixed camera position and constant illumination conditions.

2.1.6 UCF sports dataset

Introduced by Rodriguez et al. [16], the UCF sports dataset consists of 150 sports sequences like *diving, golf swinging, kicking, weightlifting, horseback riding, running, skating, swinging a baseball bat* and *walking.* The main challenges of this dataset include camera motion, different viewpoints, cluttered background, and large variations in the conduction of the depicted activities.

2.2 Representation

The efficient representation of the information within an image sequence is perhaps the most important step in order to perform recognition. Ideally, a good representation has to be able to deal with small variations in the conduct of an activity, differences in subject appearance and size, arbitrary backgrounds and differences in viewpoint. Furthermore, a good representation needs to combine richness and efficiency. That is, it needs to be rich enough in order to allow for robust classification,
and to efficiently encode the extracted information in order to minimize computer memory and resource requirements. The latter is particularly important for applications where (near) real-time processing is required.

In order to provide a comprehensive summary of the related work on representation issues, we consider three main categories: holistic representations, local representations and representations that are based on tracking. Holistic representations take into account the information depicted in the scene as a whole. As such, they encode most of the information that is present in it. However, their main drawback is that they require more preprocessing, like background subtraction or tracking, and they are sensitive to noise (e.g. due to imperfect imaging techniques, variations in illumination, compression artifacts, etc), occlusions and viewpoint changes. These limitations make holistic representations unsuitable for many problems. On the other hand, local representations encode the information as a collection of small local parts. These parts are usually small patches, centered around areas of interest, i.e. around detected interesting points. The main advantage of local representations is that they are less sensitive to noise, and preprocessing steps like background subtraction, or tracking are usually not necessary. Finally, representations that use tracking are based on temporal transitions of a set of observations. These observations can either be small patches (e.g. interest points or points manually selected) or complete configurations of the human body (e.g. through the use of kinematic models). Tracking methods usually do not require preprocessing steps like background subtraction, and are relatively robust to noise that is due to dynamic background. However, they are sensitive to partial occlusions, fast motions, and changes in appearance of the tracked subject, like deformations and viewpoint variations.

The remainder of this section is organized as follows: in section 2.2.1 we present an overview of the holistic methods that are used for representation. Section 2.2.2 deals with local representations, and finally, in section 2.2.3 we present an overview of representation methods that use tracking.

### 2.2.1 Holistic Representations

Holistic representations encode the information as a whole. Usually this involves the information that is contained within a specific region of interest (ROI), like for instance, the area around a person performing an activity. Common practices for acquiring this ROI are background subtraction, edge detection or motion information, e.g. by computing the optical flow over the whole scene. We discuss
these methods in section 2.2.1.1. Due to their global nature, these representations are sensitive to noise, occlusion and viewpoint changes. In order to partially overcome these inherent problems of global methods, a number of works propose the placement of a grid over the whole scene, and extract information within each cell of the grid. The final representation, therefore, is a collection of local observations, each localized within a grid cell. We discuss these methods in section 2.2.1.2. Finally, spatiotemporal Volumes (STVs) have also become very popular for representing human activities, where each volume is created by simply stacking image frames on top of each other, and interpolating between them. We discuss this family of methods in section 2.2.1.3.

2.2.1.1 Global Methods

Global representations usually rely on the use of silhouettes, which are extracted by background subtraction. An example of an extracted silhouette is depicted in Fig. 2.1(b). This practice implies that the background is known a priori. For realistic applications, however, the latter is usually not the case. Online estimation of the background [17] is usually a good alternative, albeit under the assumption that the background is either static or changing very slowly. Silhouettes in general are insensitive to changes in appearance, and encode a great deal of information, which is acquired using their area or their contour. However, they are sensitive to different viewpoints and may contain a significant amount of noise, due to imperfect extraction techniques.

A typical example where silhouettes form the basis of the representation are the temporal templates of Bobick and Davis [18]. In this work, the activity taking place in a scene is summarized in a motion energy (MEI) and a motion history image (MHI). The MEI indicates where the motion in the scene is located, while the MHI is an image in which the pixel intensities are a recency function of the silhouette motion (i.e. higher intensity indicating more recent activity). Examples of MEIs and MHIs are depicted in Fig. 2.1(c) and (d) respectively. While the method is innovative in the sense that three dimensional information is summarized using just a couple of two dimensional images, it suffers from a number of problems. Firstly, noise and shadow effects create small, non-zero regions in areas of the MEIs and MHIs where no motion exists. Secondly, uniform clothing of the subjects create empty areas (holes) in the silhouettes of the subjects. Both of these issues adversely affect the features that are subsequently used for recognition. Finally, the use of MHIs requires the definition of a temporal window in which the activity will be summarized, an issue that is not addressed in the original paper by Bobick and Davis. Despite these problems, temporal templates have been used for a variety of
Chapter 2. Related Work

Figure 2.1: (a) A still frame from the Weizmann database and (b) the extracted silhouette of the depicted subject, acquired by background subtraction. (c),(d) Motion energy and motion history images respectively, created using 10 frames of the sequence depicted in (a)

applications, ranging from recognition of facial expressions [19] to multiple view systems [20][21][15]. In the latter, silhouettes from multiple cameras are combined in order to form 3D voxels, called Motion History Volumes (MHV). In contrast to motion history images, the silhouette exemplar sets proposed by Weinland and Boyer [22] do not utilize any dynamic information. Instead, sets of discriminative silhouettes are selected for each class, and matching is performed directly using a Euclidean distance measure. In the same work, the Chamfer distance is also proposed as an alternative to background subtraction, where the edges of the silhouettes are matched instead. Despite the improvement in the performance, the main problem of imperfect silhouette extraction remains nonetheless.

The presence of noise in the background, like shadows, have negative effects on the extracted silhouettes. For this reason, Ahmad and Lee [23] propose a shadow elimination algorithm as a preprocessing step. By learning a distribution over the intensity and chromacity of background pixels from a set of training frames, pixels in a test frame are labeled as shadow if they differ little from the learned chromacity model and significantly from the learned intensity model. Pixels that do not fit any of the
learned models are labeled as foreground and are used in order to form the silhouettes. Subsequently, morphological operations are applied in order to smooth the resulting silhouettes and eliminate any remaining artifacts due to the shadows in the background. In order to further reduce the dependency on the shape of the silhouettes, optical flow features are calculated and used for recognition, augmented by image moments on the smoothed silhouettes. For averaging noise, Wang and Suter [24] propose to calculate the mean intensity of pixels over a sequence of centered frames. In this way, an average silhouette is obtained. An alternative representation based on the mean shape, instead of the mean silhouette is also proposed in the same work. The latter is calculated from the extracted silhouettes using a boundary following algorithm. However, the recognition performance achieved using this alternative representation was lower. Goldenberg et al. [25] perform singular value decomposition on silhouette contours in order to extract eigenshapes for analysis of periodic motions. However, as noted by the authors, the method relies heavily on the accuracy of the segmentation and the tracking steps that are used in order to extract the contours.

In cases where the background of the scene is not known, silhouette extraction cannot be performed. In order to deal with this issue, a number of methods propose to use motion information in order to represent an activity. This is usually realized by the extraction of a set of optical flow vectors around the subject, which usually need to be compensated for camera motion. Efros et al. [26] follow an approach of this kind in order to recognize activities in sports videos. In order to eliminate noise, the optical flow is half-wave rectified and smoothed using a Gaussian filter. However, their work is based on figure-centered sequences acquired by tracking, and therefore the overall performance of the method relies on the tracker’s reliability. Fathi and Mori [27] also acquire figure-centric representations by tracking before extracting their mid-level optical flow and spatial gradient features. Their work, therefore suffers from similar problems. Finally, Jiang and Martin [28] acquire shape flow features using tracking. Matching is directly performed using the optical flow lines. However, the matching problem is NP-hard, and while relaxation methods can reduce the computational complexity, it still remains high. Furthermore, similar to the works discussed previously, the use of tracking for the extraction of the flow lines makes the method dependent on the tracker’s performance.

2.2.1.2 Grid-based Methods

The presence of noise in the background, like shadows, reflections due to illumination conditions etc., as well as partial occlusions, adversely affect global representations. In order to partly overcome these
issues, grid-based representations have been proposed. By using a grid, descriptors are calculated within each of its cells, and the final representation is derived by concatenating these descriptors. The innovative idea behind this approach is that noise or partial occlusion will only affect the descriptors in the cells that are located on the affected areas, like the ones where occlusion occurs, while the majority of cells will remain unaffected, leading to more robust representations.

The kind of descriptors used in grid-based methods vary. Histograms of oriented gradients (HOG) \[29\] and flow (HOF) \[30\] are a popular choice, due to their invariance against geometric and photometric transformations. The latter usually appear in large spatial regions, larger than the localized cells in which the descriptors are extracted. Furthermore, the descriptors at each cell can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and using this value to normalize the descriptors within all cells in the block. Due to their ability of detecting humans in cluttered backgrounds and their robustness to partial occlusions, HOG and HOF descriptors have been extensively used for human activity recognition. For instance, Thurau \[31\] uses this kind of descriptor in order to retrieve human poses from still images. Human activities are subsequently encoded as a sequence of poses, called n-grams. An extension of this approach is used in \[32\], for increased robustness against background clutter. More specifically, HOG representations are combined with non-negative matrix factorization (NMF) \[33\] in order to learn a set of basis representations from human pose images, and suppress edges within the grid cells that belong to the background. For the same purpose, Ikizler-Cinbis et al. \[34\] initially apply a boundary operator in order to find the edges that most probably belong to the subject. Subsequently, they extract HOG descriptors using a grid on the selected edges. Their method is applied for activity recognition in videos acquired from the web and is shown to be robust against clutter and partial occlusion. Contrary to these methods, which use a sliding window in order to detect the subjects in the scene, Lu and Little \[35\] use tracking in order to localize a subject in time. Tracking is performed using the shape of the subjects, which is encoded using HOG descriptors. Their method is successfully tested on sports sequences, and is robust to changes in illumination, dynamic background and partial occlusions.

Alternatives to HOG/HOF features have also been proposed in grid-based human activity representations. However, the majority of these methods require background subtraction in order to work, and are therefore sensitive to clutter. For instance, Ikizler and Duygulu \[36\] use gradients in order to fit oriented rectangular patches on the silhouette of the subjects in the scene. Subsequently, a grid is used in order to bin the orientations of the patches into histograms and perform recognition of poses
by histogram comparison. Tran et al. [37] use a frame descriptor called motion context, in analogy to the shape context of Belongie et al. [38]. The descriptor is derived from the concatenation of silhouette shape and optical flow radial histograms, where each histogram is created from the values that lie within grid cells centered on the subject. Shape contexts are also used in [39], combined with motion contexts in order to represent the shape and motion of human silhouettes, where each context is extracted within local cells positioned on the body. Finally, Lin et al. [40] use optical flow and foreground pixel counts in order to describe motion and shape in spatiotemporal grids defined on blocks centralized on the subject. The blocks are derived using either background subtraction or by using human detectors. Subsequently, a tree of prototype poses is created and used to recognize human activities.

2.2.1.3 Space-Time Volumes

Similar to silhouettes of human subjects in images, Space-Time Volumes (STV) can be viewed as three dimensional silhouettes of human activities. Assuming an efficient segmentation process that can isolate a subject in an image sequence, and a process that can account for scale changes (e.g. due to camera zoom), a space-time volume can be easily created by stacking together the segmented silhouettes from each frame, and by interpolating between frames. Subsequently, spatiotemporal features can be extracted at the boundaries of the volume in order to describe its shape, like for instance, local spatiotemporal gradients [41]. Such features lead to descriptions of the spatiotemporal shape of an activity, and therefore to more descriptive representations compared to the ones created using just sequences of silhouettes.

One of the most characteristic approaches in this direction is proposed by Blank et al. [10] [42]. In their work, human activities are represented as space-time shapes, created by stacking extracted silhouettes over a given sequence. The method, however, relies on accurate background subtraction and is therefore sensitive to noise, like for instance, due to motion clutter. Oikonomopoulos et al. [8] propose an alternative approach, where B-spline polynomials are locally fitted on automatically detected interest points lying on the motion boundary of the activity. As such, they locally approximate its spatiotemporal shape. Due to its local nature, and the use of visual descriptor codebooks, the proposed method is more robust to dynamic background and occlusion, although this is not exclusively addressed.
A popular way to deal with clutter, dynamic background and occlusion is to initially create template STVs from clean sequences and use correlation, during testing, in order to detect depicted activities. Such a method is presented in [16], where spatiotemporal gradients are used in order to create STVs from each example in the training set. Subsequently, filters designed to minimize intra-class variances are created by combining the Fourier Transforms of the created STVs. During testing, activities are recognized by convoluting each class-specific filter to the test sequence. The same principle is followed by Ke et al [43], who create STVs during training, using shape and flow features. During testing, mean shift clustering is used in order to segment an input video to spatiotemporal regions, according to their location and pixel intensity values. Similar features are then extracted from each region, which are then correlated to the template STVs that were created during training. This method is extended by the same authors in [44], by splitting the volumes in the training set into parts. This allows the detection of individual spatiotemporal events in the activity (e.g. the upward motion of the hand).

Yilmaz and Shah [45] use contour tracking in order to track a silhouette through a sequence of frames, and create a volume by stacking the recovered contours at each frame. Subsequently, local differential geometrical properties are used in order to represent the volume. Yan et al. [46] use multiple view sequences in order to construct a 4D Action Feature Model (AFM). The method uses similar features as the ones used in [45], in order to describe the shape and motion of the AFM model. Subsequently, these features are used in order to match activities from arbitrary viewpoints. However, although contour tracking is a good option in order to create volumes in the presence of noise, like motion clutter, the robustness of these methods rely on the reliability of the tracking process.

### Conclusion

In order to conclude this section, we present, in Table 2.1, a summary of some of the methods that were discussed. From the table, we conclude that in terms of overall recognition performance and handling of occlusions and dynamic background, methods that were based on spatiotemporal volumes performed the best. This was mainly due to the use of correlation during testing, and the fact that training was performed on clean sequences. Grid-based methods also exhibited some robustness against occlusion or dynamic background. However, this robustness was mainly achieved using sliding windows and an exhaustive search (like, e.g. in [31]), making their use more demanding in terms of computational complexity. On the other hand, global methods were unable to handle such conditions. This is due to the fact that these methods require the definition of the region of interest of the action a priori.
### Table 2.1: Summary of holistic representation methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Representation</th>
<th>Features</th>
<th>Datasets used</th>
<th>Occ./D.Back.</th>
<th>Classifier</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tran et al. [37]</td>
<td>Global grid</td>
<td>Motion context</td>
<td>Weizmann</td>
<td>No</td>
<td>NN</td>
<td>100</td>
</tr>
<tr>
<td>Ikizler et al. [36]</td>
<td>Global grid</td>
<td>Hist. of rectangles</td>
<td>Weizmann</td>
<td>No</td>
<td>SVM</td>
<td>100</td>
</tr>
<tr>
<td>Gorelick et al. [42]</td>
<td>STV</td>
<td>local, various</td>
<td>Weizmann</td>
<td>Yes/No</td>
<td>NN</td>
<td>96.7/92.3</td>
</tr>
<tr>
<td>Wang et al. [24]</td>
<td>Global</td>
<td>AMF/MMS</td>
<td>Weizmann</td>
<td>No</td>
<td>NN</td>
<td>94.4</td>
</tr>
<tr>
<td>Thurau et al. [32]</td>
<td>Global grid</td>
<td>HOG-NMF</td>
<td>Weizmann</td>
<td>Yes</td>
<td>NN</td>
<td>94.4</td>
</tr>
<tr>
<td>Weinland et al. [22]</td>
<td>Global</td>
<td>silhouettes</td>
<td>Weizmann</td>
<td>No</td>
<td>Custom, discr.</td>
<td>93</td>
</tr>
<tr>
<td>Jiang et al. [28]</td>
<td>Global</td>
<td>shape flows</td>
<td>Weizmann</td>
<td>Yes</td>
<td>NN</td>
<td>90</td>
</tr>
<tr>
<td>Thurau [31]</td>
<td>Global grid</td>
<td>HOG</td>
<td>Weizmann</td>
<td>Yes</td>
<td>NN</td>
<td>86.6</td>
</tr>
<tr>
<td>Fathi et al. [27]</td>
<td>Global</td>
<td>flow-based</td>
<td>Weizmann/KTH</td>
<td>No</td>
<td>Adaboost</td>
<td>100/90.5</td>
</tr>
<tr>
<td>Lin et al. [40]</td>
<td>Global grid</td>
<td>optical flow</td>
<td>Weizmann/KTH</td>
<td>No</td>
<td>NN</td>
<td>100/95.77</td>
</tr>
<tr>
<td>Rodriguez et al. [16]</td>
<td>STV</td>
<td>ST-gradients</td>
<td>KTH</td>
<td>Yes</td>
<td>Correlation</td>
<td>88.3</td>
</tr>
<tr>
<td>Ahmad et al. [23]</td>
<td>Global</td>
<td>motion/shape flows</td>
<td>KTH</td>
<td>No</td>
<td>MDHMM</td>
<td>88</td>
</tr>
<tr>
<td>Ke et al. [43]</td>
<td>STV</td>
<td>shape/flow</td>
<td>KTH</td>
<td>Yes</td>
<td>SVM</td>
<td>80.9</td>
</tr>
<tr>
<td>Oikonomopoulou et al. [8]</td>
<td>STV</td>
<td>B-splines</td>
<td>KTH</td>
<td>Yes</td>
<td>RVM</td>
<td>80.8</td>
</tr>
<tr>
<td>Weinland et al. [21]</td>
<td>Global</td>
<td>silhouettes</td>
<td>IXMAS</td>
<td>No</td>
<td>HMM</td>
<td>81.2</td>
</tr>
<tr>
<td>Ke et al. [44]</td>
<td>STV</td>
<td>shape/flow</td>
<td>custom</td>
<td>Yes</td>
<td>NN</td>
<td>N/A</td>
</tr>
<tr>
<td>Yilmaz et al. [45]</td>
<td>STV</td>
<td>n-jets</td>
<td>custom</td>
<td>Yes</td>
<td>NN</td>
<td>N/A</td>
</tr>
<tr>
<td>Yan et al. [46]</td>
<td>STV</td>
<td>n-jets</td>
<td>custom</td>
<td>Yes</td>
<td>NN</td>
<td>N/A</td>
</tr>
<tr>
<td>Lu et al. [35]</td>
<td>Global grid</td>
<td>PCA-HOG</td>
<td>custom</td>
<td>Yes</td>
<td>HMM</td>
<td>N/A</td>
</tr>
<tr>
<td>Mendoza et al. [39]</td>
<td>Global grid</td>
<td>shape-motion</td>
<td>KTH+Weizmann</td>
<td>No</td>
<td>CRF/HMM</td>
<td>92.5/90.2</td>
</tr>
</tbody>
</table>

When the latter can be easily acquired, these methods perform very well, as depicted in the table.

### 2.2.2 Local Representations

The lack, or the difficulty in obtaining an accurate background model, and the presence of noise and partial occlusions, pose significant difficulties in accurately defining the region of interest of an action. This poses a significant problem for global methods, and usually leads to the enforcement of several constraints in order for these methods to work. In order to tackle these issues, local representations have been proposed. In local representations, the observation is sampled into patches, and the whole set of samples is used in order to form the final representation. Sampling can be performed either densely, throughout the observation, or sparsely, that is, at points that have a high probability of being important or characteristic of the activity. Subsequently, a set of spatiotemporal descriptors are extracted in order to describe the local space-time shape within the area of support of each patch, i.e. its scale. Several types of local feature descriptors are covered in section 2.2.2.1.

A large number of works treat patches independently. That is, they assume that there is no implicit correlation between the spatiotemporal locations at which they are extracted, or between the information that they engulf. In such works, recognition can either be performed directly, using a distance measure between the patches themselves, or by using a codebook. The latter is usually created by clustering a set of patches in the training set. Recognition is performed by comparing the codeword distributions in the test set, for instance by histogram comparison. This is called the ‘bag of words’
approach, and is covered in section 2.2.2.2. By contrast, a great deal of works exploit the relationships between the patches. In order to account for small variations in the locations of the patches, or in the descriptors that characterize them, these relationships are encoded using a probabilistic framework. Section 2.2.2.3 deals with this family of works.

2.2.2.1 Local spatiotemporal feature representations

For a lot of applications, a good description of the scene can be obtained by considering the information around certain points of interest such as corners and edges, that is, in areas that are rich in information. However, determining which part of the visual information is relevant is an open problem, because it naturally depends on the semantic description that we wish to obtain.

The success of interesting points in object detection and localization, their sparsity, and robustness against illumination, clutter, and viewpoint changes \cite{47} have inspired a number of methods in the area of motion analysis and activity recognition. A typical example is the space-time interest points \cite{48,11,49}. An extension of the Harris corner detector in time, the space-time interesting points are extracted by detecting significant local variations of gradients in space and time, and correspond, therefore, to areas where motion changes direction abruptly. As such, the resulting representations might be insufficient for activities that are characterized by unidirectional motion, like e.g. gait activities. Oikonomopoulos et. al. \cite{3} propose to use spatiotemporal salient points for activity representation, by extending in time the spatial salient point detector of Kadir and Brady \cite{4}. In contrast to \cite{48}, the points that are detected correspond to areas where there is a significant amount of motion. The proposed spatiotemporal salient points are detected by measuring the variations in the information content of pixels that lie within local spatiotemporal neighborhoods. Subsequently, local extremes of changes in the entropy across scales are detected and the saliency of each point at a certain scale is defined in terms of both the entropy and its rate of change at the scale in question. In an alternative approach, Rapantzikos et al. \cite{50}, calculate saliency by minimizing an energy function consisting of neighboring voxel interactions, like proximity, scale and similarity. In the same direction is the work of Dollar et al. \cite{51}, who use 1D Gabor filters in order to capture intensity variations in the temporal domain, acquiring relatively dense representations. Jhuang et al. \cite{52} use a hierarchy of Gabor filters in order to construct their C-features. Their method is inspired by the human visual system, and the features that are derived are invariant to scale changes in space and time. Schindler and Van Gool \cite{53} extend the work of \cite{52} by combining both shape and optical flow responses. Their
2.2. Representation

system is subsequently used for action recognition, and more specifically, in determining the minimum number of frames required in order for an action to be successfully recognized. One of the most common type of descriptors stems from the Scale Invariant Feature Transform (SIFT). Introduced in [54], it has been widely used in a variety of applications, including object (e.g. [55] [56]) and scene classification (e.g. [57] [58]). The underlying concept in SIFT is the use of a cascade of Gaussian filters of variable width. Keypoints are subsequently detected as the extrema of the Difference of Gaussian filters (DoG) across different scales. Finally, the Speeded Up Robust Features (SURF) [59] utilize second order Gaussian filters and the Hessian matrix to detect interesting points, while the use of integral images [60] makes the calculation of the filter response extremely fast. Instead of extracting gradient descriptors around the detected points (as in SIFT), the authors of [59] use Haar wavelets.

The main characteristic of the aforementioned local representations is their sparseness and their robustness against scale variations in space and time. However, they do not address issues like general camera motion, dynamic background, occlusions, multiple activities in the same scene and so on. In this direction, Laptev et al. [61] propose a local velocity adaptation mechanism in order to compensate for the motion of the camera. Somewhat similar is the work of Oikonomopoulos et al. [8], who use filtered optical flow in order to detect spatiotemporal salient points. Filtering is performed using local median filters, where the terms corresponding to global motion are subtracted from the original optical flow field in order to single out the vectors that correspond to independent motion, e.g. due to occurring activities. Gilbert et al. [62] use 2D Harris corner detectors in three channels in order to extract corners in space and time. Hierarchical clustering is subsequently performed in order to group the features. During testing, data mining techniques are utilized in order to recover similar feature clusters from the training database, and localize in this way activities in the presence of camera motion, occlusion and background clutter. Han et al. [63] extract HOG and HOF features around space-time interesting points for recognition of actions in movies. In order to enhance their representation, they also take into account the context of the scene. They achieve this by training classifiers that fire in the presence of several objects, like cars, telephones etc. Bregonzio et al. [64] extend the method in [51] by using Gabor filters in both spatial and temporal dimensions. Activities are then represented by modeling the patterns of the detected keypoints, i.e. their spatial distribution and the time at which they appear. As such, they can effectively deal with the presence of occlusion. Similar features are used by Reddy et al. [65], who construct, instead, a tree of feature class membership. The tree is subsequently used in order to assign labels on individual features in the test set and detect activities in the presence of motion clutter and occlusions.
Chapter 2. Related Work

2.2.2.2 Bag of word representations

The ability of local feature representations to successfully encode shape and motion has played a major role in the development of codebooks of visual words. The latter are created by initially clustering the extracted feature descriptors in the training set [66]. Each of the centers of the resulting clusters is considered to be a codeword and the set of codewords forms the ‘codebook’. By using a codebook, the information depicted in images and videos can be summarized as a histogram of visual words, instead of a simple collection of descriptor vectors. This is the classical ‘bag of words’ approach, in which recognition is usually performed by histogram comparison.

Apart from representation compactness, visual codebooks increase robustness against dynamic background, since, ideally, features that are due to the latter will not match well the entries of the utilized codebook and will be therefore suppressed. Visual codebooks have been extensively used for detecting objects, activities and humans. Aiming at object recognition, Agarwal and Triggs [55] extract SIFT-like descriptors in a hierarchical way, where each level of the hierarchy is a spatially coarser version of the previous level. The resulting features are called hyperfeatures, and are matched against visual codebooks created for each level. Then, the histogram of the descriptors at each level of the hierarchy is classified using Support Vector Machines (SVM). SIFT descriptors in a bag-of-words framework are also used by Li and Fei-Fei [57] for the combined problem of event, scene, and object classification, with application to sports images. Scovanner et al. [67] extend the SIFT descriptor in time. Their features are binned using a spatiotemporal grid of histograms, and polar representations are used in order to quantize the descriptors. Similar is the work of Kläser et al. [68], who propose regular polyhedrons in order to quantize their 3D gradient descriptors and create their codebook. Laptev et al. [12] extract HoG and HoF features around detected space-time interest points, and use k-means in order to construct their codebooks. Their method is subsequently used for retrieving actions from movies. Similar is the work presented in [13], where SIFT features are also used, while scripts from the utilized movies are used in order to automatically annotate scenes during training. Using the space-time interest points of [48], Niebles et al. [69] represent each class as a distribution of visual words from the codebook and learn a probabilistic Latent Semantic Analysis (pLSA) model [70] on each of the representations. Finally, similar to the work of Jhuang et al. [52], Ning et al. [71] use the responses of 3D Gabor filter banks in order to build their descriptors. A bag of words model is subsequently used in order to localize instances of human activities in videos, using sliding temporal windows of varying duration.
2.2.2.3 Representations based on spatiotemporal consistency

Despite their success in object [55], [36] and scene [57] classification, the use of ‘bag of words’ models poses a significant drawback. That is, by binning a set of neighboring descriptors into a histogram, the information concerning the spatiotemporal relationships between these descriptors is lost. Voting methods have been very popular in preserving such relationships. In such methods, the spatiotemporal location of activated codewords is implicitly encoded via a set of reference points. Such methods are very efficient in object/event detection in the presence of motion clutter, dynamic background or partial occlusions. For instance, Leibe et al. [9] propose an implicit shape model for object detection. Their model consists of a codebook of visual words, in which the relative position of each word with respect to the object center is maintained. Subsequently, the stored locations are used during testing in order to cast probabilistic votes towards the object center and localize an object. A similar method is proposed by Opelt et al. [72], where fragments extracted via edge detection are used. In [73], a similar voting scheme is implemented for activity recognition and the per-frame spatial localization of the subjects performing them. The commonality of these methods is that the positions of the activated codewords are stored, during training, with respect to a reference point, e.g. the object center. During testing, each descriptor that is matched against the codebook casts probabilistic votes to where the object center lies. In this way an estimate of the position of the object center is obtained. Furthermore, by storing the positions of the activated codewords with respect to a reference point, spatial consistency of the codewords is implicitly preserved. An interesting variation of this concept is presented in the work of Marszalek and Schmid [74], where codewords appearing in the foreground (i.e. around the object of interest) are positively weighted compared to the ones belonging to the background. In this way a map that indicates the most probable position of the query object is created.

Instead of specifying a reference point, a number of methods directly encode existing spatiotemporal relationships between the codewords. In the majority of these works, the codebook entries consist of groups of codewords rather than single codewords. Such approaches avoid the specification of reference points in order to localize an activity, and rely instead on correlation methods for detection, offering faster recognition, and the use of minimal training sets. For instance, Sivic et al. [75] propose the use of doublet codewords, in which each entry in the codebook consists of a pair of codewords. Boiman and Irani [76] generalize this approach, by proposing feature ensembles. In their model, the features in the ensemble are represented as a star graph, and their number can be arbitrary.
Table 2.2: Summary of local representation methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Representation</th>
<th>Features</th>
<th>Datasets used</th>
<th>Occ./D.Back.</th>
<th>Classifier</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Junejo et al. [78]</td>
<td>Local-detection</td>
<td>HOG</td>
<td>Weizmann/IXMAS</td>
<td>Yes</td>
<td>SVM</td>
<td>94.6/72.5</td>
</tr>
<tr>
<td>Scovanner et al. [67]</td>
<td>Local-BoW</td>
<td>3D-SIFT</td>
<td>Weizmann</td>
<td>No</td>
<td>SVM</td>
<td>82.6</td>
</tr>
<tr>
<td>Klauser et al. [68]</td>
<td>Local-BoW</td>
<td>3D gradients</td>
<td>Weizmann/KTH</td>
<td>No</td>
<td>SVM</td>
<td>84.9/91.4</td>
</tr>
<tr>
<td>Laptev et al. [61]</td>
<td>Local-BoW</td>
<td>HOG-HOF</td>
<td>KTH/HoHA</td>
<td>Yes</td>
<td>SVM</td>
<td>91.8/38.4</td>
</tr>
<tr>
<td>Gilbert et al. [62]</td>
<td>Local</td>
<td>ST-interest points</td>
<td>KTH</td>
<td>No</td>
<td>SVM</td>
<td>96.5</td>
</tr>
<tr>
<td>Han et al. [63]</td>
<td>Local</td>
<td>HOG/HOF</td>
<td>KTH</td>
<td>Yes</td>
<td>Gaussian Process</td>
<td>94.1</td>
</tr>
<tr>
<td>Uemura et al. [73]</td>
<td>Local-detection</td>
<td>combination</td>
<td>KTH</td>
<td>Yes</td>
<td>tree-based NN</td>
<td>93.1</td>
</tr>
<tr>
<td>Bregonzio et al. [64]</td>
<td>Local</td>
<td>ST-interest points</td>
<td>KTH</td>
<td>Yes/No</td>
<td>SVM</td>
<td>93.1</td>
</tr>
<tr>
<td>Jhuang et al. [52]</td>
<td>Local</td>
<td>C-features</td>
<td>KTH</td>
<td>No</td>
<td>SVM</td>
<td>91.7</td>
</tr>
<tr>
<td>Reddy et al. [65]</td>
<td>Local</td>
<td>ST-cuboids</td>
<td>KTH</td>
<td>Yes</td>
<td>NN</td>
<td>90.3</td>
</tr>
<tr>
<td>Rapantzikos et al. [50]</td>
<td>Local</td>
<td>Salient points</td>
<td>KTH</td>
<td>No</td>
<td>SVM</td>
<td>88.8</td>
</tr>
<tr>
<td>Schindler et al. [53]</td>
<td>Local</td>
<td>C-features</td>
<td>KTH</td>
<td>No</td>
<td>SVM</td>
<td>88</td>
</tr>
<tr>
<td>Nowozin et al. [80]</td>
<td>Local</td>
<td>ST-cuboids</td>
<td>KTH</td>
<td>No</td>
<td>SVM</td>
<td>87</td>
</tr>
<tr>
<td>Niebles et al. [69]</td>
<td>Local-BoW</td>
<td>Gabor filters</td>
<td>KTH</td>
<td>Yes</td>
<td>SVM</td>
<td>81.5</td>
</tr>
<tr>
<td>Dollar et al. [51]</td>
<td>Local</td>
<td>ST-cuboids</td>
<td>KTH</td>
<td>No</td>
<td>NN</td>
<td>81.2</td>
</tr>
<tr>
<td>Schuldt et al.[11]</td>
<td>Local</td>
<td>ST-interest points</td>
<td>KTH</td>
<td>No</td>
<td>SVM</td>
<td>71.7</td>
</tr>
<tr>
<td>Marszalek et al. [13]</td>
<td>Local-BoW</td>
<td>SIFT-HOG-HOF</td>
<td>HoHA2</td>
<td>Yes</td>
<td>SVM</td>
<td>32.6</td>
</tr>
</tbody>
</table>

Matching is subsequently performed using belief propagation based on the descriptor similarity and the spatiotemporal arrangement of the features in the ensemble. A similar method, using constellations of static and dynamic feature collections is presented in [77], [69]. The method employs a hierarchy of two levels. In the upper level there is a constellation of parts, each connected to a local bag of features model in the lower level. In this way, each part is associated with a collection of similar features in terms of location and appearance. Areas in images/videos that share similar geometric properties and similar spatio(temporal) layouts are matched in [1], using a self similarity descriptor. The latter encodes the local spatio(temporal) shape within the same image(video). Matching is subsequently performed using the ensemble matching algorithm of [76]. A similar method is presented in [78], where a Self Similarity Matrix (SSM) is created for view-independent human activity recognition. Finally, Seo and Milanfar [79] extend the method in [1] by proposing local steering kernels as features instead.

### Conclusion

We present, in Table 2.2, a summary of some of the methods that were discussed in this section. By comparing tables 2.2 and 2.1, we conclude that local methods perform better than global ones in the presence of dynamic background or occlusions, as it is evident from the achieved recognition rates. This conclusion, along with the fact that local features were the choice of researchers for the analysis of more challenging datasets, like HoHA, justifies the choice of local representations in order to deal with such issues.
2.2.3 Tracking Based Representations

Although a separate area of research by itself, tracking has been extensively used for human activity analysis and recognition. This section by no means intends to give a complete overview of tracking. Instead, we initially give a short introduction of tracking and the main methods that are used in section 2.2.3.1, while we focus on the use of tracking for the problem of human activity representation in section 2.2.3.2.

2.2.3.1 Tracking basics

The main objective of tracking is to estimate the state $x_k$ (e.g. position, pose) given all the measurements $z_{1:k}$ up to the current time instant $k$. In a probabilistic framework, this translates in the construction of the a posteriori probability $p(x_k|z_{1:k})$. Theoretically, the optimal solution in case of Gaussian noise in the measurements is given by the Kalman filter [81], which yields the posterior being also Gaussian. Kalman filters and their variants, like the Extended (EKF) and the Unscented Kalman Filters (UKF) [82], [83], [84] have been extensively used for a variety of tracking applications [85], [86]. However, in nonlinear and non-Gaussian state estimation problems Kalman filters can be significantly off.

To overcome the limitations of Kalman filtering, the classical particle filtering algorithm, or so-called Condensation, was proposed [87], [88]. The main idea behind particle filtering is to maintain a set of possible solutions called particles. Each particle is associated with a weight, the latter expressing the likelihood of the particle being the actual solution. By maintaining a set of solutions instead of a single estimate as is done by Kalman filtering, particle filters are more robust to missing and inaccurate data. The major drawback of the classic Condensation algorithm, however, is that a large amount of particles might be wasted because they are propagated into areas with small likelihood. In order to overcome this problem, a number of variants to the original algorithm have been proposed, having as a common characteristic the goal of achieving a more optimal allocation of new particles. Since particle weights determine how the particles are being resampled, the likelihood function has an essential influence on the tracking performance [89]. Several attempts have been made in order to adjust the way new particles are assigned, through the use of kernels [90], [91], [92], orientation histograms [93] or special transformations like Mean Shift [94].

Despite the improvement in the tracking performance of the previous methods, the inherent problem
of the classic condensation algorithm, that is, the propagation of particles in areas of small likelihood is not sufficiently addressed. In order to effectively deal with this issue, the Auxiliary Particle Filtering (APF) algorithm was proposed by Pitt and Shephard [95]. The APF algorithm operates in two steps. At first, particles are propagated and their likelihood is evaluated. Subsequently, the algorithm chooses again and propagates the particles according to the likelihood of the previous step. Since the introduction of the APF algorithm, a number of variants have been proposed in order to address different issues. In [96] a modified APF tracking scheme is proposed for the tracking of deformable facial features, like mouth and eye corners. The method uses an invariant color distance that incorporates a shape deformation term as an observation model to deal with the deformations of the face. In order to take into account spatial constraints between tracked points, the particle filter with factorized likelihoods is proposed in [97], where the spatial constraints between different facial features are pre-learned and the proposed scheme tracks constellations of points instead of a single point, by taking into account these constraints.

Particle filters are often used within a template tracking framework. The object’s appearance is captured in the first frame of an image sequence and subsequently tracked throughout the end of the sequence. The underlying assumption behind template tracking is that the object will not significantly change its appearance throughout the duration of the video. This assumption, however, is not realistic, since an object can undergo several rotations, deformations or partial occlusions, making the template no longer an accurate model of the appearance of the object. A simple but rather naive solution to this problem is to update the template at every frame with a new template corresponding to the tracked position of the object. This approach, however, leads to error accumulation, as small errors are constantly introduced in the appearance of the template. As a result, the template eventually drifts away from the object and in the most cases gets stuck on the static background of the scene. A compromising solution between these two extremes is to partially update the template, as the weighted average (e.g. 90-10 %) of the current and the initial template, a process often called exponential forgetting. Although this solution offers a somewhat more robust tracking, by allowing the template to adapt, it does not avoid error accumulation, and there is still a high probability that the template will eventually drift away from the object.

Matthews et al specifically address the drift problem in [98]. The tracked template is updated at every frame, while maintaining the initial template specified in the first frame. To eliminate drift, the new template is aligned every time to the initial one using a gradient descent rule. This strategy, however,
is most suitable for tracking rigid objects (e.g., cars). For objects whose appearance changes over time, the authors adopt an approach of template tracking with Active Appearance Models (AAM). The appearance model and the template are updated at every time instance, leading to a more robust tracking algorithm. A similar framework is presented in [99], where a set of adaptive appearance models are used for motion-based tracking. The appearance model used consists of three components. The stable component ($S$) is used to capture the behavior of temporally stable and slowly varying image observations, the data outlier or ‘lost’ component ($L$) is used to capture data outliers due to failures in tracking, occlusion or noise and finally the ‘wandering’ component ($W$) is used to model sudden changes in the appearance of the object. The parameters of the model are adjusted online via EM and the system is tested in tracking scenarios where a high degree of partial object occlusion occurs. Finally, in [100] a Support Vector Machine (SVM) is used in order to provide an initial guess for an object position in the first frame. The position of the initial guess in subsequently refined so that a local maximum of the SVM score is achieved. The whole framework is called Support Vector Tracking (SVT) and is implemented in moving vehicle tracking scenarios.

2.2.3.2 Representation

A major component in human computing research is localization and tracking of the human body, either as a whole or as a part (e.g., head, limbs). Especially for the purposes of scene analysis and activity recognition, body tracking has received a lot of attention in the last few years. Due to its high degree of freedom (usually 28-60), body tracking is inherently a very difficult problem. Because of that, it calls upon sophisticated tracking algorithms, that can address the problem of high dimensionality. Furthermore, large appearance changes, occlusion between body parts, and the absence of typical appearance due to clothing, pose additional problems that need to be dealt with.

In contrast to rigid objects, tracking of articulated objects is inherently a much more difficult problem, mainly due to the high number of degrees of freedom that are involved. Accurate human body tracking, in particular, is an extremely important aspect for human computing applications. A possible strategy for estimating the configuration of articulated objects is sequential search, in which a number of parameters are initially estimated and, assuming that this estimation is correct, the values of several other parameters are determined. For instance, Gavrila and Davis in [101] first locate the torso of the human body and then use this information in order to initialize a search for the limbs. This approach, however, only works for specific views and is very sensitive to self-occlusion that is, occlusion between
different body parts. A similar approach is presented in [102], where a particle filtering framework is used for the purposes of hand tracking. For the same purpose, Cipolla et al [103] propose a view-based hierarchical probabilistic tracking framework that can deal with changes in view and self occlusions. The system uses edge and color cues in order to estimate the likelihood function of the hand position and configuration and subsequently a Bayesian filtering framework that performs the tracking. In [104] a particle filtering approach is adopted for articulated hand tracking. The tracker is guided by attractors, pre-collected training samples of possible hand configurations whose observations are known, while the whole process is modeled by a Dynamic Bayesian Network. A Bayesian Network is also adopted in [105] in order to model the existing constraints between the different parts of the human body. These constraints are learned using Gaussian Mixture Models (GMM) and training is done using motion-capture frames of walking data as the ground truth. Observations are based on multi-scale edge and ridge filters while the whole process is assisted with a pooled background model derived by the set of training images. In [106] a Dynamic Markov Network is utilized instead to model the relations between body parts and tracking is done using an sequential Monte Carlo algorithm. A similar approach is presented in [107], where an elastic model is used to represent relations and constraints between the limbs and a Nonparametric Belief Propagation (NBP) algorithm for the purpose of tracking. In [108] a combination of particle filters and Hidden Markov Models (HMM) is used for tracking and recognition respectively, of articulated hand gestures. Appearance-based models are learned for the non-rigid motion of the hand and a filtering method is used for the underlying rigid motion. Both treatments are unified into a single Bayesian framework. A similar approach is implemented in [109], where arm gestures are recognized as a sequence of body poses. The latter are recognized via edge matching and HMMs are used in order to extract the gestures from the pose sequences. HMMs are also used in [110] for recognizing pointing gestures. Skin information is used to localize the hands and the head of the subject in a scene and a multiple hypothesis scheme is used for the tracking. Subsequently, an HMM-based approach is adopted for recognizing the gestures.

Articulated object tracking, and particularly human body tracking suffer from dimensionality issues, an inherent problem whenever there is a large number of degrees of freedom. This fact makes the use of tracking algorithms like particle filters rather impractical. The reason for this is that a very large number of particles is required in order to represent the posterior function in a sufficient way, making this kind of tracking algorithms slow and computationally expensive. The problem becomes even more prominent whenever real-time performance is required, such as in monitoring applications, virtual trainers or augmented reality applications. In order to deal with this issue, a number of
different techniques have been proposed, either by constraining the configuration space [101] or by restricting the range of the movements of the subject [111]. These approaches, however, greatly reduce the generality of the implemented trackers, making them impractical in real applications. Eigenspace decomposition [112] and principal component analysis [113] offer an interesting alternative for dimensionality reduction. In [114], a modified particle filtering approach is used in order to reduce the complexity of human body tracking. The main characteristic of the utilized tracker is its ability to avoid local maxima in the tracking by incorporating a search based on simulated annealing, and thus called annealed particle filter. Apart from dimensionality reduction techniques, several researchers have attempted to modify the way classical tracking algorithms work in order to achieve computational efficiency and real-time tracking performance. A simple example are the earlier mentioned kernel-based particle filters [90], [91], [92], [115] or particle filters that use special transformations, as in [93], [94]. These methods attempt to limit the number of required particles for efficient tracking, effectively reducing the computational complexity of their algorithms. Finally, an interesting approach for real-time tracking and recognition of hand actions is presented in [116],[117]. The motion of the hand is extracted using skin cues and is subsequently tracked using the Mean-Shift Tracking scheme of [115]. The spatiotemporal curvatures of the extracted trajectories are used in order to represent the actions performed. The local maxima of these curvatures are view-invariant and are used for image sequence alignment and matching of the actions.

### 2.2.4 Conclusion

In this section we presented a review of the most common representation methods for human activities existing in the literature. We have divided these methods into three parts, namely holistic representations, local representations and representations that are based on tracking.

Holistic approaches that use spatiotemporal volumes for representation can handle more effectively the presence of dynamic background or partial occlusions, due to the use of correlation during testing. However, when silhouettes or tracking are used for the creation of the volumes, these approaches heavily rely on the prior knowledge of the background or the reliability of the tracking process respectively. Global methods require the region of interest to be defined a priori, and therefore they are inherently sensitive to occlusions, segmentation errors, and varying backgrounds. However, when the subject can be accurately localized within the image sequence, these representations can be very descriptive. Grid-based representations can partly handle issues like dynamic background and occlusions. This is
usually performed by exhaustively searching for the subject in the scene, by using sliding overlapping windows (like, e.g. in [31]). Local representations, on the other hand, can be very efficient in dealing with dynamic background, occlusions, multiple activities and so on. Especially when a codebook is used, features belonging to the background are to some degree suppressed, since the probability of them matching the codewords in the codebook can be small. This probability becomes even smaller when constellations of features are used instead of single codewords.

Finally, representations based on tracking have been shown to be very robust against conditions like viewpoint changes, occlusions, noise in the background due to motion clutter etc. However, most trackers fail when there is a significant deformation in the tracked templates, a problem which is more prominent when articulated objects, like the human body are tracked.

### 2.3 Classification

In this section we give an overview of action classification methods. That is, methods that assign an action label to an unseen image sequence, given its representation. Similar to the surveys of Aggarwal and Cai [118], and Wang et al. [119], we divide action classification techniques in two groups: template matching and state-space approaches. Template matching approaches compare the representations directly with a set of action exemplars or action prototypes. We deal with these approaches in section 2.3.1. On the other hand, state-space approaches use graphical models in order to represent action classes. We discuss this family of methods in section 2.3.2.

#### 2.3.1 Template Matching

Template matching approaches work by comparing representations of image sequences directly. As such, they are easier to implement and have low computational complexity. However, they are more sensitive to temporal differences in the conduction of the activities and are view-dependent. Matching can be performed using either exemplars or action prototypes. The former case is simpler, and involves direct comparison of the test sequences to the action exemplars. The number of comparisons needed, however, is linear to the number of exemplars in the training set, and therefore, this approach is not suitable for very large databases. Action prototypes tend to alleviate this problem, since they can be created by the exemplars by means of e.g. clustering, reducing the number of comparisons needed to
classify a test example. Section 2.3.1.1 deals with this family of works. In contrast to exemplar-based matching, discriminative classifiers learn to distinguish between two or more classes. That is, they do not model each class separately, but rather learn a function that optimally separates the classes. We discuss this family of classifiers in section 2.3.1.2.

### 2.3.1.1 Exemplar-based matching

As mentioned in the introduction of this section, matching using exemplars is based on direct matching between representations. The simplest classifier that can be used in this category is the k-Nearest Neighbor classifier (kNN). In kNN, an action instance is classified by a majority vote of its neighbors, with the instance being assigned to the class most common amongst its k nearest neighbors. k is a positive integer, typically small. If k = 1, then the object is simply assigned to the class of its nearest neighbor.

The distance measures that can be used within the context of kNN classification depend on the application. For patch-based methods, a histogram of codewords is often used in order to obtain a fixed-length descriptor. Comparison can then be performed using measures tailored for probability distributions, like the $\chi^2$ distance or the Kullback-Leibler divergence. These measures are more suitable than e.g. Euclidean distance, since they take the frequency of occurrence of each codeword into account. Oikonomopoulos et al. [8], e.g., use the $\chi^2$ distance in order to obtain a similarity measure between codeword histogram counts. Ning et al. [71], on the other hand, use the KL-divergence in order to compare their codeword histograms. Furthermore, to make their similarity measure symmetric, they calculate the divergence both ways and average the result. In the case of global methods, representations can be compared directly. Bobick and Davis [18] compare their temporal templates using the Mahalanobis distance. The latter is used in order to account for the different orders of the image moments that are used for the description of the utilized templates. Similarly, Blank et al. [10] use Euclidean distance on normalized global features and a leave-one-out 1NN strategy in order to classify their examples. Tran et al. [37] define discriminative distance measures for use in their NN classification scheme, like the Large Margin Nearest Neighbors (LNNN). The latter attempt to learn a covariance matrix that maximizes the Mahalanobis distance between examples of different labels and minimizes the same distance between examples of the same labels. Chamfer distance measures [120] are also popular, mainly for their robustness against outliers. For example, a Chamfer distance measure is used in [3] in order to compare salient point representations. This distance is further minimized
using a gradient optimization scheme in order to align the representations in space and time, and is subsequently used to define kernels for Relevance Vector Machine (RVM) classifiers. Finally, contour matching using chamfer distance measures is proposed by Weinland and Boyer [22], for matching silhouette edges and avoid the need for background subtraction.

One of the main drawbacks of kNN classifiers is that they are linear to the number of available exemplars, since a distance/similarity has to be calculated to each one of them in order to classify an unseen example. One way to alleviate this is the use of action prototypes. A simple way to generate action prototypes is by averaging over all exemplars of the same class. This approach is followed by Wang and Suter [24], who calculate the mean intensity of pixels over a sequence of centered frames. Weinland et al. [15] create action class prototypes by performing PCA on the training set. Every class is then represented by the mean value of the descriptors over the available population of the action, while an unseen action instance is classified according to a Mahalanobis distance associated to a PCA based dimensional reduction of the data vectors. An obvious disadvantage of action class prototypes, however, is their inability to capture the full variance of an activity class. In fact, this is not only a problem of activity prototypes, but a problem of exemplar-based methods in general, and is closely related to the number of training examples that are available. This problem is partly addressed by Rodriguez et al. [16], who attempt to model the intra class variance from spatiotemporal volumes in the training set.

Temporal differences in the conduction of human activities is one of the most important issues that need to be resolved prior to classification. Linearly normalizing activities of different durations (e.g. by stretching in time) [10][3] is in some cases acceptable, however not optimal. Dynamic Time Warping (DTW) is a very popular method for aligning sequences, and signals in general, of different durations and sizes. In DTW, the sequences are ”warped” non-linearly in the time dimension to determine a measure of their similarity independent of certain non-linear variations in the time dimension. Warping is performed by computing the best nonlinear time normalization of the test sequence in order to match the template sequence by performing a search over the space of all allowed time normalizations. This search is efficiently performed by dynamic programming. Veeraraghavan et al. [121] use DTW in order to match sequences, by taking into account the non-Euclidean nature of the distance between their normalized shape features. In later work [122], the same authors address the alignment of sequences by considering the space of warping functions for a given activity. Finally, Lin et al. [40] match frames in the test set with stored action prototypes and form an action similarity matrix. DTW is
subsequently applied on that matrix and a distance is calculated based on the optimal alignment path. Classification is then performed using kNN on a set of labeled prototype sequences.

In many cases, reducing the dimension of the feature space can be of benefit for the purpose of classification. Dimensionality reduction can be realized in two ways: by feature selection and by subspace analysis. Feature selection is the process where only a subset of the initial feature set is used for representation and classification. In general, this subset is selected in such a way so that it contains the most informative features for a specific class. Boosting is probably the most popular way in order to perform feature selection, although statistical methods have also appeared in the literature [123]. In boosting, the final classifier is a linear combination of a set of weak classifiers. In turn, each weak classifier operates on a different dimension/feature of the feature vector. At each stage the algorithm picks the weak classifier that, given a set of weights, separates the examples of different classes best. A typical application of boosting is by Fathi and Mori [27], who use Adaboost [124] and two rounds of boosting on their low-level motion features. In the first round, they select mid-level motion features, that are characteristic of local neighborhoods within the action sequence. The selected features are combined by the second round in order to form characteristic sets that correspond to the whole action class. Laptev and Pérez [49] use Adaboost in order to select characteristic histogram features for recognition of actions in movies. In order to increase speed, Ke et al. [125] use a greedy approach, called forward feature selection, in order to extract motion features for human activity detection. Similar is the work of Nowozin et al.[80], who use LPBoost instead. Smith et al. [126] use a variation of boosting that takes into account past responses of the weak classifiers before updating the weights. Finally, Torralba at al. [127] apply a modification of the gentleboost algorithm [128] in order to construct decision boundaries shared by more than two classes. The final classifier for each class is then a combination of the learned decision boundaries.

Similar to feature selection, the goal of subspace analysis is to reduce the dimensionality of the data before classification. The main idea is to map the data into a lower dimensional space, while retaining most of its variance. This lower dimensional space is in many cases termed as the manifold, and can be determined in many ways. Masoud et al. [129] use Principal Component Analysis (PCA) on motion features in order to define a manifold, for the purpose of human activity recognition. Similarly, Weinland et al. [15] classify their examples using a Mahalanobis distance associated to a PCA based dimensional reduction of their data vectors. Instead of PCA, Wang and Suter [130] use Locality Preserving Projections (LPP) in order to reduce the dimensionality. Subsequently, Gaussian mixture
Chapter 2. Related Work

models are used in order to model the data in the low dimensional space. Finally, Zhang and Sim [131] follow an approach based on Linear Discriminant Analysis (LDA) in order to maximally separate their classes and address the problem of high dimensionality.

2.3.1.2 Discriminative classifiers

In contrast to template-based classification, discriminative classifiers do not model each class separately. Instead, they learn a function, or equivalently, a decision boundary, that best separates two or more classes. One of the most popular and effective classifiers of this kind is the Support Vector Machine (SVM) [132]. SVMs work by constructing a hyperplane on a higher dimensional space which optimally separates the classes. The optimal hyperplane is the one that has the largest distance from the nearest training points of any class. These training points define the margin, and are called support vectors. SVMs have been extensively used in conjunction with bag of word approaches, and for a variety of features. SIFT features and SVMs are used in [55] and [68] for the recognition of images and human activities from videos respectively. Similarly, Seovanner et al. [67] use three dimensional SIFT and SVMs for the same purpose. SVMs with Gabor features inspired by the human visual system are used by Jhuang et al [52]. Schüldt et al. [11] use SVMs with a gaussian kernel defined using the $\chi^2$ distance between feature histograms in order to classify human activities. The descriptors that are extracted are based on the computation of spatiotemporal derivatives at the center of each feature, and represent shape and motion within the area of support of the detected features. Similar is the work in [12], for the recognition of actions in movies. A similar method is followed in [61], albeit with an extended descriptor set containing optical flow features as well. Similar to SVMs, Relevance Vector Machines (RVM) [5] have also been used for human activity recognition [3]. RVMs are similar in function to SVMs, however, they provide a probabilistic classification instead of the hard decisions provided by SVMs. Unlike the SVM classifiers, the non-zero weights of RVM are not associated with examples close to the decision boundary, but rather appear to represent prototypical examples of classes. However, they are more prone to get stuck in local minima, due to the EM algorithm used for the estimation of their parameters.
2.3. Classification

2.3.2 State-Space Approaches

As their name suggests, state-space models consist of a series of states. These states are connected to each other with edges, and hence have the form of a graph. For this reason, they are also called graphical models. The edges in these models reveal probabilistic interactions between the states, and between the states and the observations. Within the context of human activity recognition, the observation can be a feature vector describing a frame or a series of frames. Consequently, states in graphical models correspond to different phases of the activity. The most general form of a graphical model is the Bayes network. Each state is a random variable, and connections between the states model dependencies between the variables. A Bayes network modeling a time series instead, is called a Dynamic Bayes Network (DBN). The simplest form of DBN is the Hidden Markov Model (HMM) [133], and has been extensively used for activity recognition. HMMs make two independence assumptions. Firstly, they assume that state transitions only depend on the previous state (Markov assumption). Secondly, they assume that observations are conditioned only on the current state, that is, subsequent observations are statistically independent. As such, HMMs model specific classes. These assumptions drastically reduce the number of parameters that need to be learned and make inference tractable. Furthermore, HMMs have the ability to generate observations for a given activity. For this reason, they belong to the family of generative models. The latter, and in particular HMMs are discussed in section 2.3.2.1.

A second family of graphical models are the discriminative models. Discriminative models learn probabilities of the action classes conditioned on the observations. In contrast to HMMs, discriminative models can take into account multiple observations, from different time instances. As their name suggests, discriminative models do not model each class separately. Rather, they learn differences between classes. We discuss this family of models in section 2.3.2.2.

2.3.2.1 Generative Models

As has been already mentioned, HMMs are probably the most popular generative models used for activity recognition. Training is performed using the Baum-Welch algorithm, while Viterbi decoding is used in order to determine the probability of observing a given sequence. The sequence is subsequently classified to the class of the HMM that can generate the sequence with the highest probability. HMMs can be applied to model the transitions of the whole body or specific parts of the body. In the first
Chapter 2. Related Work

case, observations regarding the whole body configuration are taken into account. Weinland et al. [21], for instance, use HMMs in order to model the sequence of body poses during the conduction of an activity. In their implementation, they explicitly model the evolution of the body pose depending on the observation and viewpoint. However, the method they propose is based on silhouettes that are acquired by background subtraction, and is therefore sensitive to occlusions, motion clutter and dynamic background. The work of Ahmad and Lee [23] suffers from similar limitations. However, it is more robust to noise that is due to shadows, due to several preprocessing step in the silhouette extraction process. Similarly, Feng and Perona [134] assign a single codeword to each state of the utilized HMM. Each codeword is an action movelet, that is, a collection of features extracted on silhouettes acquired by background subtraction. Lv and Nevatia [135] on the other hand, do not use silhouettes or background subtraction, but rather rely on motion capturing systems during learning in order to explicitly encode key poses and viewpoints. They consider, therefore the vision problem solved and focus on the recognition part, which is based on a variation of HMMs, but with additional links between non-successive nodes. Lu and Little [35] combine HMMs with tracking in an interchangeable way in order to track and recognize activities. HMMs for each action are initially learned offline. Subsequently, during tracking, the response of each HMM up to the current time instant is used to infer the next state, reducing the searching space for the utilized tracking algorithm. The proposed algorithm is shown to be robust to dynamic background and partial occlusion. However, this is subject to the efficient performance of the utilized tracker. Finally, Ramanan and Forsyth [136][137] learn appearances of body parts and their possible configurations in order to detect and track subjects in videos. Subsequently, HMMs are used for recognition of the activities. The utilized learning step makes the proposed method able to function in a variety of conditions, including background clutter.

A second family of methods use HMMs in order to model the motion of each body part separately. Apart from the reduction in complexity, these approaches allow for the recognition of activities that do not belong to the training set. That is, learnt body part motions that occur in different activities can be combined in order to recognize unseen classes. Furthermore, since each part of the body is modeled separately, these approaches are robust to partial occlusions. This approach is followed by Ikizler and Forsyth [138], who train their HMMs using the individual motion of the hands and limbs of the subjects in the training set. Given a query video, they determine the individual motions of the limbs in the query, resulting in more detailed description of the action taking place. Lv and Nevatia [139] use a similar approach, however they use the HMMs for each joint as individual weak classifiers for an AdaBoost classification scheme. Furthermore, they compare their HMM-based recognition
framework with a classifier based on template matching, and report significantly better results on the former. Finally, Peursum et al. [140] use a similar model as the one used by Weinland et al. [21]. However, they replace the state corresponding to body orientation with a set of states that capture the configuration of the body joints. However, in order to reduce complexity they ignore interactions and dependencies between the individual joints.

2.3.2.2 Discriminative Models

Discriminative models differ from generative models in that they do not allow sample generation from the joint distribution between the observed and unobserved variables (e.g. the action labels). That is, while generative models specify a joint distribution between the observed variables and the labels, discriminative models specify a conditional distribution on the labels given the observations. As a consequence, discriminative models can be trained so that they learn to discriminate between different classes. In general, discriminative models are more suitable for distinguishing classes that are very similar. The latter could be easily confused using generative models like HMMs. However, since these models do not make independence assumptions like HMMs, they require a lot more training data for parameter learning.

Conditional Random Fields (CRF) are a category of discriminative models that have been extensively used in activity recognition applications. In contrast to Markov Random Fields (MRF) and HMMs, which, for reason of inference tractability, only take into account interactions between neighboring nodes, CRFs have the ability of encoding dependencies between distant variables, e.g. distant objects in images. Furthermore, due to their discriminative nature, they require far less data in order to be trained. CRFs were originally proposed by Lafferty et al. [141], who showed that CRFs significantly outperform HMMs and maximum entropy markov models (MEMM). The latter are discriminative models related to CRFs, however they assume the Markov property. These results are also supported by Sminchisescu et al. [142], who use CRFs with first order dependencies in order to recognize human activities. Mendoza and de la Blanca [39], on the other hand, show that HMMs outperform CRFs when motion features are used. However, when shape features are used, CRFs achieve higher recognition rates.

Variants of CRFs have also been proposed in the literature. For instance, He et al. [143] propose the multiscale CRF (mCRF), which combines conditional distributions that capture statistical structure
Chapter 2. Related Work

In images at different scales. In a similar fashion, Quattoni et al. [144] propose CRFs with hidden variables for part-based object recognition in images. Torralba et al. [145] propose boosted random fields, CRFs in which boosting is used in order to learn the local evidence potentials of the CRFs. Finally, Kumar and Hebert [146] propose an extension of the CRF algorithm, called Discriminative Random Fields (DRF). While CRFs do not explicitly model dependencies between the observations, DRFs model local interactions between labels and observations. The method is subsequently used for the detection of man-made objects.

2.3.3 Conclusion

In this section we presented a review of the most common classification algorithms that are used in the area of human activity recognition. We have divided these algorithms into two categories, template matching approaches and state-space approaches. In the former, a test example is usually compared with examples from a training set. Nearest neighbor classifiers are very common for that purpose. Their simplicity, and the fact that they do not require any parameters to be specified has made them very popular. However, as has been already mentioned, NN classification can be very time consuming, due to the number of comparisons that need to be performed in order for a test example to be classified. More specifically, the number of comparisons that need to be performed is linear to the number of the examples in the training set. This fact makes the use of NN classification unsuitable for very large datasets. In addition, NN classification is sensitive to the presence of outlier examples. To deal with such issues, discriminative classifiers have been proposed, like SVMs and RVMs. The main characteristic of these classifiers is their sparsity, due to the relatively small number of examples that are required in order to define the boundary that divides the classes. Since the definition of this boundary is performed via an optimization process, discriminative classifiers are less sensitive to the presence of outliers.

State-space approaches consist of purely probabilistic models. Each of these models has the form of a graph, the edges of which model dependencies between problem variables. State-space approaches are divided into generative and discriminative models. A generative model models a single class, and can generate new observations after it is trained. For inference tractability, several independence assumptions are usually made, like e.g. the Markov assumption in the case of HMMs. On the other hand, discriminative models learn to discriminate between pairs of classes. Contrary to generative models, they can model dependencies between distant states, and it has been shown that they can achieve
improved recognition rates compared to, e.g. HMMs. However, due to the absence of independence assumptions between their states, they require significantly more data for training.

2.4 Activity Detection

In this section we focus on methods that perform activity detection. The methods belonging to this family attempt to localize and simultaneously classify individual instances of human activities in unseen videos. The method presented in the last chapter of this dissertation belongs to this family of methods. Activity detection methods do not explicitly model a class, and they do not attempt to discriminate between different classes. Instead, they resort to methods like correlation or voting in order to complete the detection task. As such, they are very robust to the presence of dynamic background and occlusions. This property makes this family of methods extremely suitable for use in real-world applications, in which such challenges are prevalent.

Detection methods were primarily applied in the area of object localization and recognition. A typical example is the work of Leibe et al. [9], who propose an implicit shape model for object detection. Their model consists of a codebook of visual words, in which the relative position of each word with respect to the object center is maintained. Subsequently, the stored locations are used during testing in order to localize an object. A similar method is proposed by Opelt et al. [72], with the boundary fragment model (BFM). The success of these approaches, has inspired a number of methods in the area of human activity detection. For instance, Mikolajczyk and Uemura [73] utilize a similar method as the one in [9] in order to localize subjects in image sequences that depict human activities. However, the detection remains limited in the spatial domain, without accounting for detection in time, that is, without determining the action’s temporal boundaries. A similar case is presented in video google, by Sivic and Zisserman [58], who search for objects or object selections in movie frames. Searching is very efficient due to the application of several geometrical constraints between the utilized features. However, matching is restricted to spatial regions only. This limitation is addressed by Boiman and Irani [76], who develop a very efficient algorithm for behaviour detection. In their work, activities are represented as ensembles of patches. Each ensemble is encoded as a star graph, where each node contains a patch and each edge of the graph connects a patch with the spatiotemporal center of the ensemble. For activity detection, a progressive elimination algorithm is implemented, in which the search space is progressively shrunk, during search, depending on the
ensemble members that are detected. Areas in images or videos that share similar geometric properties and similar spatio(temporal) layouts are matched in [1], using a self similarity descriptor. The latter encodes the local spatio(temporal) shape within the same image(video). Matching is subsequently performed using the ensemble matching algorithm of [76]. A similar method is presented in [78], where a Self Similarity Matrix (SSM) is created for view-independent human activity recognition. Seo and Milanfar [79] extend the method in [1] by proposing local steering kernels as features instead. Their method exhibits greater robustness to noise and lower computational complexity. Ning et al. [71] correlate image sequences using histograms of codewords consisting of Gabor filter responses. However, searching is performed using sliding temporal windows. Correlation is also used by Rodriguez et al. [16] in order to localize activities in space and time. Their method works by creating filters, during training, that minimize intra-class variances. This is performed by combining the Fourier transforms of features that are extracted on the spatiotemporal volumes of the activities.

2.5 Conclusions and Discussion

In this chapter we have presented a short overview of the most recent methods in the area of human activity representation and recognition. In this section, we summarize these works. Furthermore, we identify and discuss the most significant trends of future research stemming from these works.

As has been already mentioned, representation is perhaps the most important issue in activity recognition. This is because a poor representation will almost certainly lead to poor recognition results even in the presence of the most sophisticated classifier. Efficiency is another important issue. The desirability to cope with conditions found in realistic scenes, like, for instance, motion and background clutter, occlusions, and changes in viewpoint has shifted most research performed on representation issues away from global methods. The latter are deemed unsuitable when this kind of conditions are prevalent, since they heavily depend on the accurate definition of the ROI of the action. However, global methods are very useful and already used in commercial applications in which the background is static or easy to determine. A typical example is that of gaming and of virtual trainers. Furthermore, applications like surveillance in indoor areas, like in elderly homes are ideal for this kind of methods. On the other hand, local methods like the ones based on local spatiotemporal descriptors are expected to be used in more demanding applications. These include applications like monitoring and surveillance of public areas and content based video indexing and retrieval. Especially the latter is of key
2.5. Conclusions and Discussion

importance, due to the increasing amount of video data available online. Finally, although tracking applications are themselves sensitive to viewpoint changes and occlusions, they can offer tremendous help in both global and local representations, by defining the ROI around the object of interest. In the case of global methods, this definition is the main issue, so therefore, tracking and global methods often appear together, like in [25][35]. On the other hand, in the case of local methods, tracking can reduce noise from background clutter, and reduce the number of potential false positives.

Classification is, naturally, the next step after acquiring a representation of an activity. As discussed, there are two main families of classification techniques. Template based approaches and state-space approaches. The former usually involve a set of exemplars, which form the training set. An unknown example is then classified according to its similarity with one or more of these exemplars. On the other hand, state-space approaches are essentially models that attempt to capture the dynamics of the activity classes. In the case of generative models, like HMMs, an unknown example is classified according to the model that can generate it with the highest likelihood. On the other hand, in the case of discriminative models like CRFs, the example is classified depending on the highest probability of class membership given the observation. For both families of classification approaches, the amount of training data plays a major role. In the case of template based methods, good classification depends on the amount of class variability covered by the training set. For state-based approaches, larger amount of training data leads to better parameter estimation. Requiring large amounts of training data, however, has several drawbacks. First of all, one has to consider the difficulty in acquiring data in general, let alone data that covers most of the variability a class can contain. Secondly, large amounts of training data make training a time consuming process. For instance, as has already been discussed, the time needed to classify an unseen example using a kNN classifier is linear to the number of examples in the training set. It is evident, therefore, that recognition with as little training data as possible is desired.

Perhaps the most prevalent trend in the area of activity analysis is the use of more realistic videos for training and testing of the algorithms proposed. The increasing amounts of visual information that need to be automatically processed (e.g. for surveillance applications) is an important factor dictating this need. Therefore, algorithms need to be developed and trained that will be able to cope with challenging conditions in these videos, like low resolution, significant motion clutter, or abrupt camera motions. Furthermore, the high classification rates achieved for already available datasets of human activities (e.g. Weizmann, KTH) reveal that activity analysis algorithms are mature enough to handle
more challenging datasets. A good example is the HOHA dataset [12], which features examples of human activities retrieved from films. The huge variability of the activities depicted in datasets of this kind has forced algorithms to consider issues like context. That is, recognize the setting under which an activity is performed and subsequently detect the activity or not. For instance, the presence of a car in scene would be a strong cue and a prerequisite for activities like Drive car or Get out of car to occur.

Finally, a significant issue with which lots of recent works deal is activity detection. That is, the spatiotemporal localization and subsequent classification of human activities in continuous video streams. Up to now, a large amount of works ignored the problem of localization and focused solely on classification issues. Under this scope, the problem of localization was considered to be solved. This approach, assumes that the two problems are decoupled and can be treated independently. However, as dictated by recent works (e.g. [1],[78]), it is far more effective to treat these issues under the same framework.
Chapter 3

Spatiotemporal Salient Points

In this chapter we propose to extract a set of spatiotemporal features that are the extension, in the temporal direction, of the spatial salient feature detector developed in [4]. Our goal is to obtain a sparse representation of a human action as a set of spatiotemporal points that correspond to activity variation peaks. In contrast to the work of Laptev [48], in which a sequence is represented by the local activity endpoints (starts/stops), the proposed representation consists of points that are localized at areas where there are peaks in activity variation, such as the edges of a moving object. Like the authors of [4], we automatically detect the scales at which the entropy achieves local maxima. Subsequently, we propose a clustering algorithm that forms clusters consisting of spatiotemporal salient points with similar locations and scales. Each image sequence is then represented as a set of spatiotemporal salient regions, the locations of which are normalized in order to achieve invariance against the translation of the subjects performing the actions. We use the Chamfer distance as an appropriate distance measure between two representations. In order to deal with different speeds in the execution of the actions and to achieve invariance against differences in subject size, we propose a linear space-time warping method which linearly warps two examples by minimizing their Chamfer distance. A simple kNN classifier and one based on Relevance Vector Machines, introduced in [5], are used in order to test the efficiency of the representation. We test the proposed method using real image sequences, where we use aerobic exercises as our test domain. Our experimental results show fairly good discrimination between specific motion classes.

The remainder of this chapter is organized as follows: In section 3.1, the proposed spatiotemporal feature detector is described in detail, while in section 3.2 the algorithm for creating salient regions
from the detected salient points is given. In section 3.3 the proposed recognition method is analyzed, including the proposed space-time warping technique. In section 3.4, we present our experimental results, and in section 3.5, final conclusions are drawn and limitations of the proposed algorithm are discussed.

3.1 Spatiotemporal Salient Point Detection

Let us denote by \( N_c(s, \vec{v}) \) the set of pixels in an image \( I \) that belong to a circular neighborhood of radius \( s \) that is centered at pixel \( \vec{v} = (x, y) \). In [4], in order to detect salient points in static images, Kadir and Brady define a saliency measure \( y_D(s, \vec{v}) \) based on measuring changes in the information content of \( N_c \) for a set of different radii (i.e. scales). In order to detect spatiotemporal salient points at peaks of activity variation we extend the Kadir’s detector by considering cylindrical spatiotemporal neighborhoods at different spatial radii \( s \) and temporal extends \( d \). The latter correspond to varying numbers of frames forward and backward in time that are taken into account in the definition of the neighborhood. More specifically, let us denote by \( N_{ct}(\vec{s}, \vec{v}) \) the set of pixels in a cylindrical neighborhood of scale \( \vec{s} = (s, d) \) centered at the spatiotemporal point \( \vec{v} = (x, y, t) \) in the given image sequence. At each point \( \vec{v} \) and for each scale \( \vec{s} \) we will define the spatiotemporal saliency \( y_D(\vec{s}, \vec{v}) \) by measuring the changes in the information content within \( N_{ct}(\vec{s}, \vec{v}) \). Since we are interested in activity within an image sequence, we consider as input signal the convolution of the intensity information with a first-order Gaussian derivative filter. Gaussian derivative filters have been extensively used for detecting interesting points in static images. Here, we apply them in the temporal domain in order to arrive at a measure of activity. Formally, given an image sequence \( I_0(x, y, t) \) and a filter \( G_t \), the input signal that we use is defined as:

\[
I(x, y, t) = G_t * I_0(x, y, t). \tag{3.1}
\]

Subsequently, for each point \( \vec{v} = (x, y, t) \) in \( I \), we calculate the Shannon entropy of the signal histogram in a spatiotemporal neighborhood around it. Let us note that we considered cylindrical spatiotemporal neighborhoods of radius \( s \) and temporal extend \( d \) for simplicity reasons. However, more complicated shapes, such as elliptical neighborhoods at different orientations and with different axes ratios could
be considered. The signal entropy $H_D(\vec{s}, \vec{v})$ in the spatiotemporal neighborhood $N_d(\vec{s}, \vec{v})$ is given by:

$$H_D(\vec{s}, \vec{v}) = - \int_{q \in D} p_D(q, \vec{s}, \vec{v}) \log_2 p_D(q, \vec{s}, \vec{v}) dq,$$

where $p_D(q, \vec{s}, \vec{v})$ is the probability density of the signal histogram as a function of scale $\vec{s}$ and position $\vec{v}$. By $q$ we denote the signal value and by $D$ the set of all signal values. In this chapter we use the values that arise from eq. 3.1 as signal values. It is possible, however, to use other kinds of descriptors, such as optical flow vectors. We use the histogram method to approximate the probability density $p_D(\vec{s}, \vec{v})$. Alternatively, the probability density can be estimated using Parzen window density estimation or any other density estimation technique.

Subsequently, we proceed with the automatic selection of the scale [147] [4]. More specifically, we consider the scales at which the entropy values achieve a local maximum as candidate salient scales. Let us define as $\hat{S}_p$ the set of scales at which the entropy is peaked, that is,

$$\hat{S}_p = \left\{ \vec{s} : \frac{\partial H_D(\vec{s}, \vec{v})}{\partial s} = 0 \land \frac{\partial H_D(\vec{s}, \vec{v})}{\partial d} = 0 \land \frac{\partial^2 H_D(\vec{s}, \vec{v})}{\partial s^2} < 0 \land \frac{\partial^2 H_D(\vec{s}, \vec{v})}{\partial d^2} < 0 \right\}. \quad (3.3)$$

Then, following the approach of [4], we can define the saliency measure at the candidate scales as follows:

$$y_D(\vec{s}, \vec{v}) = H_D(\vec{s}, \vec{v}) W_D(\vec{s}, \vec{v}), \quad \forall \vec{s} \in \hat{S}_p. \quad (3.4)$$

Eq. 3.4 gives a measure of how salient a spatiotemporal point $\vec{v}$ is at certain candidate scales $\vec{s}$. The first term of eq. 3.4 is a measure of the variation in the information content of the signal. The weighting function $W_D(\vec{s}, \vec{v})$ is a measure of how prominent the local maximum is at $\vec{s}$, and is given by:

$$W_D(\vec{s}, \vec{v}) = s \int_{q \in D} \left| \frac{\partial}{\partial s} p_D(q, \vec{s}, \vec{v}) \right| dq + d \int_{q \in D} \left| \frac{\partial}{\partial d} p_D(q, \vec{s}, \vec{v}) \right| dq, \quad \forall (s, d) \in \hat{S}_p. \quad (3.5)$$

where the values in front of each summation in the right part of eq. 3.5 are normalization factors. More details on the way the latter are derived are given in Appendix A.

When a peak in the entropy for a specific scale is distinct, then the corresponding pixel probability
density functions at the neighboring scales will differ substantially, giving a large value to the integrals of eq. 3.5 and thus, to the corresponding weight value assigned. By contrast, when the peak is smoother, then the integrals in eq. 3.5, and therefore the corresponding weight, will have a smaller value.

If we take the discrete sampling of the grid into account, eq. 3.2, eq. 3.3 and eq. 3.5 become:

\[
H_D(s, d, \vec{v}) = - \sum_{q \in D} p(q,s,d, \vec{v}) \log_2 p(q,s,d, \vec{v}),
\]  

(3.6)

\[
\hat{S}_p = \{(s, d) : H_D(s - 1, d, \vec{v}) < H_D(s, d, \vec{v}) > H_D(s + 1, d, \vec{v}) \wedge \\
H_D(s, d - 1, \vec{v}) < H_D(s, d, \vec{v}) > H_D(s, d + 1, \vec{v})\},
\]  

(3.7)

\[
W_D(s, d, \vec{v}) = \frac{s^2}{2s - 1} \sum_{q \in D} |p(q, s, d, \vec{v}) - p(q, s - 1, d, \vec{v})| + d \sum_{q \in D} |p(q, s, d, \vec{v}) - p(q, s, d - 1, \vec{v})|, \forall (s, d) \in \hat{S}_p,
\]  

(3.8)

In Fig. 3.1(b), an example of an entropy plot is presented, for the corresponding action whose one instance is shown in Fig.3.1(a). The scale which corresponds to the peak of the plot is considered a candidate salient scale, and is assigned a saliency value, according to eq. 3.4 and eq. 3.5.
3.2 Salient Regions

The analysis of the previous section leads to a set of spatiotemporal salient points \( S = \{ (\vec{s}_i, \vec{v}_i, y_{D,i}) \} \), where \( \vec{v}_i = (x, y, t) \), \( \vec{s}_i = (s_i, d_i) \) and \( y_{D,i} \) are respectively, the position vector, the scale and the saliency value of the feature point with index \( i \). In order to make the feature detector more robust against noise and to reduce the dimensionality of the resulting feature space, we follow a similar approach as that in [4] and develop a clustering algorithm, which we apply to the detected salient points. We define in this way corresponding salient regions instead of salient points. The location of these regions should be more stable than the individual salient points, since noise is unlikely to affect all of the points within the region in the same way. The proposed clustering algorithm removes salient points with low saliency value and creates clusters that are a) well localized in space, time and scale, b) sufficiently salient and c) sufficiently distant from each other.

The steps of the proposed algorithm can be summarized as follows:

1. Derive a new set \( S_T \) from \( S \) by applying a global threshold \( T \) to the saliency of the points that consist \( S \). Thresholding removes salient points with low saliency, that is,

\[
S_T = \{ (\vec{s}_i, \vec{v}_i, y_{D,i}) : y_{D,i} > T \}.
\]

2. Select the point with index \( i \) in \( S_T \) that has the highest saliency value. Use the salient point \( i \) as a seed to initialize a salient region \( R_k \) (in the first iteration \( k = 1 \)). That is,

\[
R_k = \{ i \}.
\]

3. Add points \( j \) to the region \( R_k \) that are nearest, in terms of Euclidean distance, to the seed \( i \), as long as the intra-cluster variance does not exceed a threshold \( T_V \). That is, as long as

\[
\frac{1}{|R_k|} \sum_{j \in R_k} d_j^2 < T_V,
\]

where \( R_k \) is the set of the points in the current region \( k \) and \( d_j \) is the Euclidean distance of the \( j \)th point from the seed point \( i \).

4. If the overall saliency of the region \( R_k \) is lower than a saliency threshold \( T_S \), that is,
\[ \sum_{j \in R_k} y_{D,j} \leq T_S, \]  

discard the points in the region back to the initial set of points and continue from step 2 with the next highest salient point. Otherwise, calculate the Euclidean distance of the center of region \( R_k \) from the center of salient regions already defined in the previous steps of the algorithm, that is, from salient regions \( R_{k'}, k' < k \).

5. If the distance is lower than the average scale of the region, discard the points in the region, put them back to the initial set of points, and continue from step 2 with the next highest salient point. Otherwise, accept the region as a new cluster and store it as the mean scale and spatial location of the points in it.

6. Form a new set \( S_T \) consisting of the remaining salient points, increase the cluster index \( k \) and continue from step 2 with the next highest salient point.

By setting the threshold \( T_V \) in step 3, we define clusters that have local support and are well localized in space and time. In this way, we avoid clusters with large variance in their spatiotemporal position and scale. In addition, we want to take the saliency of the points into consideration such that the overall saliency of the region is sufficient. We do this in step 4, by setting a saliency threshold, \( T_S \). Finally, the purpose of step 5 is to accept and create clusters that are sufficiently distant from each other. To summarize, a new cluster is accepted only if it has sufficient local support, its overall saliency value is above the saliency threshold, and it is sufficiently distant in terms of Euclidean distance from already existing clusters.

We set the global threshold \( T \) of the first step of the proposed clustering algorithm equal to 10% of the maximum saliency value. In order to ensure the sparseness of the resulting representation, we set the variance threshold equal to half the maximum spatial scale of the utilized cylindrical sampling window. Furthermore, we set the saliency threshold equal to 0.1% of the global detected saliency of the scene. We have found empirically that these values were a reasonable compromise between the amount of noise that we wish to remove and the actual signal values that we want to keep. However, cross validation methods could be used for the selection of the thresholds.

The algorithm described above requires estimation of the spatiotemporal saliency measure (eq. 3.4) for each point in the image sequence. This involves calculations at spatiotemporal neighborhoods at
3.2. Salient Regions

different scales, which can be computationally very expensive. For \( N \) pixels in an image sequence, \( O(N(sd)(s^2d)) \) number of operations are required in order to calculate the entropy, where \( (sd) \) is proportional to the number of the cylindrical neighborhoods used and \( (s^2d) \) is proportional to the average number of pixels per cylindrical neighborhood. In order to reduce the computational complexity, we also propose a two-step approach for the detection of salient points, which is an approximation of the full search approach. In the experimental section we will present and compare results from both approaches. In the first step of the proposed approximation approach, we select only salient points in space for every frame of the sequence. Among these detected points, there may be some that are also salient in time. We detect these in a second step, by extending the salient feature detector in the temporal dimension. By applying this procedure, we discard image points that are not salient in space. In other words, we direct the salient point detector to select spatiotemporal salient points that are located on the edges of the moving objects. For the two step approach, a proportional to \( O(Ns(s^2) + R(s^2d)(sd)) \) number of operations is needed in order to calculate the entropy. The first term of the summation is proportional to the operations needed for the detection of salient points using only spatial information. More specifically, \( s \) is proportional to the number of circular neighborhoods used and \( s^2 \) is proportional to the average number of pixels per circular neighborhood. The second term is very similar to the complexity of the full search, only in this case, \( R \) (the total number of pixels in the sequence for which the spatial entropy is maximized) is used instead of \( N \). In general, \( R \) is one order of magnitude smaller than \( N \), yielding a substantial reduction in the complexity of the specific approach. More specifically, the two step approach is as follows:

1. In the first step, we detect salient regions in the spatial domain, that is, for every individual frame without taking into account neighboring frames. The computational gain is due to the use of circular neighborhoods instead of cylindrical ones. The position vector \( \vec{v} \) in eq. 3.2 - 3.5 is 2-dimensional in this case, \( \vec{v} = (x, y) \), where \( 1 \leq x \leq N_1 \) and \( 1 \leq y \leq N_2 \). The above procedure leads to the creation of feature sets of the form \( F_t = \{ (x_{t,i}, y_{t,i}, s_{t,i}, y_{Dt,i}) \mid 1 \leq t \leq K, 1 \leq i \leq L_t \} \), where \( t \) is the frame number and \( L_t \) is the total number of salient points detected in frame \( t \).

2. In the second step, we set \( \vec{v} = (x_{t,i}, y_{t,i}, t) \), \( 1 \leq t \leq K, 1 \leq i \leq L_t \), and we apply eq. 3.2 - 3.5 for cylindrical neighborhoods of scale \( \vec{s} = (s, d) \). After clustering the detected spatiotemporal salient points, we derive a feature set consisting of salient regions in the space-time domain, \( F = \{ (x_j, y_j, t_j, \vec{s}_j, y_{Dj}) \mid 1 \leq j \leq L \} \), where \( L \) is the number of salient regions detected.
3.3 Recognition

Using the feature detection scheme described in sections 3.1, 3.2, we represent a given image sequence by a set of features, where each feature corresponds to a cylindrical salient region of the image sequence in the space-time domain. In what follows, we will define an appropriate distance measure that can be subsequently used for learning and recognition of human actions in image sequences. Indeed, a wide variety of classification schemes, ranging from kNN to Support Vector Machines, depends on the definition of an appropriate distance measure. We use the Chamfer Distance \[148\], as it can provide a distance measure between feature sets with unequal number of features. Chamfer Distance Transformations have been used in [120] with edge matching in order to match images of different resolutions. Here, since the number of matching points in the corresponding representations is relatively small, we loop through the Chamfer distance measure proposed in [148] in order to find the best matching points. More specifically, for two feature sets \(F = \{(x_i, y_i, t_i), 1 \leq i \leq M\}\) and \(F' = \{(x'_j, y'_j, t'_j), 1 \leq j \leq M'\}\) consisting of an \(M\) and \(M'\) number of features, respectively, the Chamfer distance of the set \(F\) from the set \(F'\) is defined as follows:

\[
D(F, F') = \frac{1}{M} \sum_{i=1}^{M} \min_{j=1}^{M'} \sqrt{(x'_j - x_i)^2 + (y'_j - y_i)^2 + (t'_j - t_i)^2}. \tag{3.13}
\]

In other words, the proposed distance is defined as the average over the set of the minimum Euclidean distances between the \(M\) feature points of set \(F\) and the \(M'\) feature points of set \(F'\). The distance measure of eq. 3.13 is not symmetrical, since \(D(F, F') \neq D(F', F)\). For recognition purposes, it is desirable to select a distance measure that is symmetrical. A measure that satisfies this requirement is the average of \(D(F, F')\) and \(D(F', F)\), that is,

\[
D_c(F, F') = \frac{1}{2}(D(F, F') + D(F', F)). \tag{3.14}
\]

Let us note that for the calculation of the distance measure we only consider the spatiotemporal position of the detected salient points.
3.3. Recognition

3.3.1 Space-Time Warping

There is a large amount of variability between feature sets due to differences in the execution speed of the corresponding actions from subject to subject. Furthermore, we need to compensate for possible shifting of the representations forward or backward in time, caused by imprecise segmentation of the corresponding actions. To cope with both these issues, we propose a linear time warping method with which we model variations in time using a time-scaling parameter \(a\) and a time-shifting parameter \(b\). In addition, in order to achieve invariance against scaling of the subjects performing the actions, we introduce a scaling parameter \(\sigma\) in the proposed time warping technique. Prior to warping, we transform the \(x\) and \(y\) coordinates of the detected salient regions in each sequence so that they have zero mean value. We do this in order to achieve invariance against translation. The parameters \(a, b\) and \(\sigma\) are estimated with a gradient-descent iterative scheme that minimizes the Chamfer distance between the sets. More specifically, let us denote by \(F_w = \{(\sigma x_i, \sigma y_i, at_i - b), 1 \leq i \leq M\}\) the feature set \(F\) with respect to feature set \(F'\). Then, the distance between \(F'\) and \(F_w\) is given by eq. 3.13 as:

\[
D(F_w, F') = \frac{1}{M} \sum_{i=1}^{M} \min_{j=1}^{M'} \sqrt{(x'_j - \sigma x_i)^2 + (y'_j - \sigma y_i)^2 + (t'_j - at_i + b)^2}. \tag{3.15}
\]

Similarly, the feature set \(F'\) with respect to feature set \(F\) can be represented as \(F'_w = \{(\frac{1}{\sigma} x'_j, \frac{1}{\sigma} y'_j, \frac{1}{a} t'_j + b), 1 \leq j \leq M'\}\) and their distance, as given by equation 3.13, as:

\[
D(F'_w, F) = \frac{1}{M'} \sum_{j=1}^{M'} \min_{i=1}^{M} \sqrt{(x_i - \frac{1}{\sigma} x'_j)^2 + (y_i - \frac{1}{\sigma} y'_j)^2 + (t_i - \frac{1}{a} t'_j - b)^2}. \tag{3.16}
\]

The distance to be optimized follows from the substitution of eq. 3.15 and eq. 3.16 to eq. 3.14. We follow an iterative gradient descent approach for the adjustment of the \(a, b\) and \(\sigma\) parameters. The update rules are given by:

\[
a^{n+1} = a^n - \lambda_1 \frac{\partial D_c}{\partial a^n}
\]

\[
b^{n+1} = b^n - \lambda_2 \frac{\partial D_c}{\partial b^n}
\]

\[
\sigma^{n+1} = \sigma^n - \lambda_3 \frac{\partial D_c}{\partial \sigma^n}
\]
where $\lambda_1, \lambda_2,\lambda_3$ are the learning rates and $n$ is the iteration index. The algorithm iteratively adjusts the values of $a, b$ and $\sigma$ towards the minimization of the Chamfer distance between the two feature sets, given by eq. 3.14. In order to determine the values of $a, b$ and $\sigma$, we need to know the values of $\frac{\partial D_c}{\partial a}, \frac{\partial D_c}{\partial b}$ and $\frac{\partial D_c}{\partial \sigma}$ for every iteration $n$. Let us denote by $k$ the index of the point in $F'$ that is closest in terms of Euclidean distance to the point $i$ in $F$, and by $A_{ik}$ the corresponding distance. Similarly, let us denote by $m$ the index of the point in $F$ that is closest to the point $j$ in $F'$ and by $A'_{mj}$ the corresponding distance. Then, from eq. 3.14 we get:

$$\frac{\partial D_c}{\partial a} = \frac{1}{2M} \sum_{i=1}^{M} \frac{\partial}{\partial a} A_{ik} + \frac{1}{2M'} \sum_{j=1}^{M'} \frac{\partial}{\partial a} A'_{mj}$$

$$= \frac{1}{2M} \sum_{i=1}^{M} \frac{\partial}{\partial a} \left[ (x'_{k} - \sigma^n x_i)^2 + (y'_{k} - \sigma^n y_i)^2 + (t'_{k} - a^n t_i + b^n)^2 \right]^{\frac{1}{2}}$$

$$+ \frac{1}{2M'} \sum_{j=1}^{M'} \frac{\partial}{\partial a} \left[ (x_m - \frac{1}{\sigma^n} x'_j)^2 + (y_m - \frac{1}{\sigma^n} y'_j)^2 + (t_m - \frac{1}{\sigma^n} t'_j - b^n)^2 \right]^{\frac{1}{2}}$$

After some operations we get:

$$\frac{\partial D_c}{\partial a} = \frac{1}{2M} \sum_{i=1}^{M} a^n t_i - t'_k - b^n A_{ik} + \frac{1}{2M'} \sum_{j=1}^{M'} t'_j - \frac{1}{\sigma^n} t'_j + t_m - b^n (a^n)^2 A'_{mj}$$

(3.20)

Similarly, for $\frac{\partial D_c}{\partial b}$ and $\frac{\partial D_c}{\partial \sigma}$ we get:

$$\frac{\partial D_c}{\partial b} = -\frac{1}{2M} \sum_{i=1}^{M} a^n t_i - t'_k - b^n A_{ik} + \frac{1}{2M'} \sum_{j=1}^{M'} \frac{1}{\sigma^n} t'_j - t_m + b^n$$

(3.21)

$$\frac{\partial D_c}{\partial \sigma} = \frac{1}{2M} \sum_{i=1}^{M} a^n (x^2_i + y^2_i) - x_i x'_k - y_i y'_k + \frac{1}{2M'} \sum_{j=1}^{M'} -\frac{1}{\sigma^n} (x^2_j + y^2_j) + x'_j x_m + y'_j y_m (\sigma^n)^2 A'_{mj}$$

(3.22)

By using eq. 3.20, eq. 3.21 and eq. 3.22, we determine the values of $a, b$ and $\sigma$ in every iteration using the update rules given in eq. 3.17, eq. 3.18 and eq. 3.19. The iterative procedure terminates when the values of $a, b$ and $\sigma$ do not change significantly or after a fixed number of iterations.

### 3.3.2 Relevance Vector Machine Classifier

Once a distance function is defined, a large number of pattern classification methods can be used for solving the $L$-class classification problem of classifying a data sample (i.e. a feature set $F'$) in one
of the $L$ classes of human actions. In this chapter, we use a kNN and a Relevance Vector Machine classification scheme, where $k = 1$. Since the application of kNN is straightforward, we will only discuss the Relevance Vector Machine Classifier.

A Relevance Vector Machine Classifier (RVM) is a probabilistic sparse kernel model identical in functional form to the Support Vector Machine Classifier (SVM). Relevance (and Support) Vector Machines have been used successfully in a large range of classification problems. In their simplest form, they attempt to find a hyperplane defined as a weighted (linear) combination of a few Relevance (Support) Vectors that separate data samples of two different classes. In RVM, a Bayesian approach is adopted for learning, where a prior is introduced over the model weights, governed by a set of hyperparameters, one for each weight. The most probable values of these hyperparameters are iteratively estimated from the data. Sparsity is achieved because the posterior distributions of many of the weights are sharply peaked around zero. Unlike the support vector classifiers, the non-zero weights of RVM are not associated with examples close to the decision boundary, but rather appear to represent prototypical examples of classes. These examples are called relevance vectors and in our case they can be thought of as representative executions of a human action. The main advantage of RVM is that while it is capable of a generalization performance comparable to that of an equivalent SVM, it uses substantially fewer kernel functions. Furthermore, predictions in RVM are probabilistic, in contrast to the deterministic decisions provided by SVM. In their original form, Relevance Vector Machines are suitable for solving 2-class classification problems.

In order to use RVMs in an $L$-class classification problem, we train multiple ($L$) Relevance Vector Machines, each of which separates a class of human actions from all other classes of human actions. Given a data sample $F$, each of the $L$ Relevance Vector Machines gives a probability that $F$ belongs to each of the $L$ classes. A data sample is classified to the class with the highest probability. In what follows, we will first briefly outline the use of the Relevance Vector Machines for a 2-class classification problem and then we will formally define our classification scheme for the $L$-class classification problem.

Given a training dataset of $N$ input-target pairs $\{(F_n, l_n), 1 \leq n \leq N\}$, an RVM learns the weights $w = \langle w_1, \ldots, w_n \rangle$, such that the conditional probability $P(l|w, F)$ can be used for predicting the label $l$ of a data sample $F$. Learning is performed using a maximum a posteriori estimation scheme where a) the conditional $P(l|w, F)$ is appropriately modeled, and b) a prior probability $p(w|a)$ ensures that the weight vector $w$ is sparse.
More specifically, given a training dataset of $N$ input-target pairs $\{(F_n, l_n), 1 \leq n \leq N\}$, an RVM learns functional mappings of the form:

$$y(F) = \sum_{n=1}^{N} w_n K(F, F_n) + w_0,$$

(3.23)

where $\{w_n\}$ are the model weights and $K(., .)$ is a Kernel function, which in the case of RVM can be viewed as a basis function. Gaussian or Radial Basis Functions have been extensively used as kernels in RVM and can be viewed as a similarity measure between $F$ and $F_n$. In our case, we use as a kernel a Gaussian Radial Basis Function defined by the distance function of eq. 3.14. That is,

$$K(F, F_n) = e^{-\frac{D_n(F, F_n)^2}{2\eta}},$$

(3.24)

where $\eta$ is the Kernel width. For classification, we want to predict the posterior probability of class membership given the input $F$. The conditional probability $P(l_n|w, F_n)$ is given by:

$$P(l_n|w, F_n) = \sigma\{y(F_n)\}^{l_n} [1 - \sigma\{y(F_n)\}]^{1-l_n},$$

(3.25)

where $\sigma(y) = 1/(1 + e^{-y})$ is the logistic sigmoid function. Since Maximum Likelihood estimation of the weights will lead to severe overfitting, a Gaussian prior is introduced over the weights:

$$p(w|a) = \prod_{i=1}^{N} N(w_i|0, a_i^{-1}),$$

(3.26)

where $a_i$ is an individual hyperparameter for every weight, leading to an $\alpha$ vector of $N$ hyperparameters. In order to estimate the weights, an iterative procedure is utilized, and a Gaussian approximation over the posterior of the weights is calculated. From that, the hyperparameters are updated and the process is repeated until the change in the hyperparameter values is minimal or when a maximum number of iterations has been reached. A detailed description of the training process of an RVM classifier can be found in [5].

In the classification phase, for the 2-class problem, a sample $F$ is classified to the class $l \in [0, 1]$ that maximizes the conditional probability $p(l|F)$. In order to use RVM classifiers for multiclass problems,
one classifier is trained for each separate class. For $L$ different classes, $L$ different classifiers are trained and a given example $F$ is classified to the class for which the conditional distribution $p_i(l|F)$, $1 \leq i \leq L$ is maximized, that is:

$$\text{Class}(F) = \arg \max_i (p_i(l|F)) .$$

(3.27)

### 3.4 Experimental results

We use three different datasets for the experimental evaluation of the proposed method. The first is a dataset depicting 19 aerobic exercises, performed by amateurs wearing everyday clothes, that have seen a video with an instructor performing the same set of exercises. Each exercise is performed twice by four different subjects, leading to a set of 152 examples. Still frames from each class in this dataset are depicted in Fig. 3.2. The second dataset is also depicting aerobic exercises, and has been recorded at a higher temporal resolution than the first aerobics dataset, in order to reduce the effect of motion blur evident in low frame-rate videos. Furthermore, it consists of reduced set of actions with respect to the first dataset. More specifically, this dataset consists of 15 different aerobic exercises, performed twice by five different subjects, leading to a set of 150 sequences. In what follows, we will refer to this dataset as the aerobics2 dataset. Finally, we present classification results on the Weizmann dataset of human actions [10], which includes 10 different activities, namely walk, run, jump, gallop sideways, bend, one-hand wave, two-hands wave, jump in place, jumping jack and skip. Each activity is performed once by 9 or 10 different actors, leading to a set of 93 different examples.

In order to illustrate the ability of the proposed method to consistently detect spatiotemporal events, we present in Fig. 3.3 the salient regions detected in five instances of four sample image sequences of the aerobics dataset. The first two columns depict two executions of the same exercise by two different subjects while the last two columns depict the execution of another exercise by another pair of subjects. It is apparent that there is consistency in the location and scale of the detected spatiotemporal salient regions between different executions of the same exercise. The detected salient points seem to appear in areas with significant amount of activity, such as the points in space and time at which the hands move fast. Since we use as input signal the convolution of the image sequence with a first-order Gaussian derivative filter, some of the detected points are located on the edges of moving objects rather than on the objects themselves (e.g. at instance $t_1'$ of the second pair of sequences). Moreover,
there seems to be a correlation between the scale of the detected regions and the motion magnitude, that is, the scale of the detected regions is large when the motion is fast (instances $t_4, t_5, t'_2, t'_3, t'_4$), and smaller when the motion is slower ($t_1, t_2, t_3, t'_1, t'_5$). This can be explained by the fact that when the motion is fast, the activity spreads over a larger spatiotemporal region than when the motion is slow.

Finally, let us note that the algorithm does not guarantee that the detection of corresponding regions across the examples will occur at the same time instance. For example, at the time instances $t_2$ and $t_3$ of the first pair of image sequences of Fig. 3.3 (i.e. first two columns), the salient point detection on the arms does not occur at the same, but at neighboring time instances. Note, that the image sequences that are presented in Fig. 3.3 are time-warped pairwise.

In order to test the influence of the proposed space-time warping algorithm, and consequently, the
Figure 3.3: Detected spatiotemporal features in four sample image sequences, corresponding to two action classes, for five time instances, \( t_i, t'_i, i = 1 \ldots 5 \). For each class the detected regions are drawn for two different subjects performing the action. Consistency in the location and scale of the detected salient regions between the different executions of the same activity is apparent.
robustness of the proposed method with respect to scale variations, we randomly selected one example per class from the aerobics set and we resized it to 1.2 and 1.5 times its initial size. We applied in each of these sequences the spatiotemporal salient point detector of section 3.1 and we used the resulting representations in order to warp them in space and time with a reference sequence. The result for a single pair of reference-resized sequences is shown in Fig. 3.4, where in the first column is the reference sequence and in the second column is the space-time warped sequence. We also stretched the latter sequence in time, so that its duration matches that of the reference one. The result is shown in the third column of Fig. 3.4. From the figure it is clear, that the space-time warped sequence is closer to the reference one, indicating that the proposed algorithm effectively warps a sequence in space and time with another, by using just the spatiotemporal salient features detected in both of them. The $\sigma$ parameter for the resized sequence was calculated equal to 1.18, which is very close to the actual value of 1.2.

In order to test the efficiency of the proposed method towards recognition, we applied a simple kNN classifier and a Relevance Vector Machine Classifier to the available feature sets. We performed our experiments in the leave-one-subject-out manner. That is, in order to classify a test exercise performed by a specific test subject, we trained our classifiers using all available data except for those belonging to the same class and performed by the same subject as the test exercise.
For the kNN classifier, the label assigned to each test example was the label of the feature set belonging to the training set with the smallest resulting Chamfer distance. The recall and precision rates achieved for every class of the aerobics dataset using the kNN classifier are depicted in Table 3.1. As can be seen from the Table, for many classes the recognition rate is higher than 80%, while for some classes, all the examples were correctly classified. For certain classes, however, the recall and precision rates are lower (e.g. classes 7, 13). An examination of the corresponding image sequences reveals that there is very little difference between the kind of motion depicted in them, and therefore, there is little difference in their resulting spatiotemporal representations, as can be seen from Fig. 3.7. The main difference in the sequences shown in the figure, is that in one of the cases the torso of the subject remains in the upright position throughout the conduction of the activity, while in the other case the subject bends a little in the front. Since there is only one camera placed in front of the subject, this ambiguity cannot be resolved by the representation, and the algorithm cannot make a clear distinction between the two classes. The overall calculated recall rate for the kNN classifier was 74.34%.

In order to classify a test example using the Relevance Vector Machines, we follow a one-against-all strategy. That is, we construct one classifier for each class and we calculate for each test example $F$ the conditional probability $p_i(l|F), 1 \leq i \leq L$, where $L$ is the total number of classes in the dataset. Each example was subsequently assigned to the class for which the corresponding classifier provided the maximum conditional probability, as depicted in eq. 3.27. Note that for estimating each of the $p_i(l|F)$, an RVM is trained by leaving out the example $F$ as well as all other instances of the same exercise that were performed by the subject from $F$. The resulting recall and precision rates for the aerobics dataset are given in Table 3.1, where an improvement in their values is visible for some classes. In Fig. 3.5 the confusion matrix generated by the RVM classifier for this dataset is given. As can be seen from the figure, there are mutual confusions between specific classes, for instance between classes 5, 6, 7, 12 and 13 and between classes 15 and 16. This is due to the minor differences in the corresponding representations, as can be seen in the example depicted in Fig. 3.7. As mentioned earlier, the reason for some of these confusions lies to the inadequacy of a single camera to capture the valuable depth information needed in order to discriminate the activities in question (e.g. the squatting activities of classes 5, 6, 7, 12 and 13). On the other hand, the confusion between classes 15 and 16 is due to the small differences in the actions themselves. The global recall rate for the RVM classifier was 77.63%, which is a relatively good performance, given the small number of examples with respect to the number of classes, and the fact that the subjects were not trained.
We present, in Fig. 3.6(a), the confusion matrix acquired by the RVM classifier for the aerobics2 dataset, along with the corresponding recall (main diagonal) and precision rates (last row) for each class. As has been mentioned in the beginning of this section, the main difference between this dataset and the aerobics dataset is the higher temporal resolution at which the former was recorded. Moreover, certain classes from the initial aerobics dataset were not included, and more specifically classes 5, 12, 13 and 15. As can be seen from Fig. 3.6, the most prominent confusion is between classes 5 and 6, which correspond, respectively, to classes 6 and 7 of the initial aerobics dataset. The main reason for this confusion is the depth ambiguity between these classes, as described for similar cases in the initial aerobics dataset. The average recall rate achieved for this dataset is 93.3%, while the average precision rate is 94.3%.

The confusion matrix acquired by the RVM classifier for the Weizmann dataset is given in Fig. 3.6(b). As can be seen from the figure, most of the confusions occur between the run, side and skip classes, while confusions for the rest of the classes are minimal. Let us note that, since each class consists of 9 or 10 examples, a 10% rate of confusion corresponds to a single example being misclassified. The average recall rate achieved for this dataset is 80.6%, with an average precision of 81.7%, the worst class being the skip class in terms of performance.

The confusion matrices of Fig. 3.5 and Fig. 3.6 conceal the fact that for some of the misclassified examples the correct matching move might be very close in terms of distance to the closest move selected. We used the average ranking percentile in order to extract this kind of information and to measure the overall matching quality of the proposed algorithm. Let us denote with $r_{F_n}$ the position of the correct match for the test example $F_n$, $n = 1 \ldots N_2$, in the ordered list of $N_1$ match values.

The average ranking percentile is defined as follows:

$$
\rho = \left( \frac{1}{N_2} \sum_{n=1}^{N_2} \frac{N_1 - r_{F_n}}{N_1 - 1} \right) \times 100%.
$$

(3.28)
3.4. Experimental results

Each of the $N_1$ values in the above equation is provided by the $N_1$ trained RVM classifiers for each dataset. The average ranking percentile for the aerobics, aerobics2 and Weizmann datasets was, respectively, 97.25%, 99.2% and 96%. The high values that are achieved reveal that for the majority of the misclassified examples, the correct matches are located in the first positions in the ordered list of match values.

In order to verify the robustness of the proposed method to scaling, we performed the same classification experiments as before, using resized versions of the original image sequences of the aerobics dataset. More specifically, we randomly selected one example per class and we resized it to 1.2 and 1.5 times its initial size. Classification was performed by considering each resized example as the test set and the entire initial set of examples as the training set, except for those belonging to the same class and performed by the same subject as the test example. We used the initial aerobics dataset for this experiment. For the resized by 1.2 set, 14 out of 19 examples were correctly classified with the kNN classifier, and 13 out of 19 with the RVM classifier, while for the resized by 1.5 set, 13 out of 19 examples were correctly classified by both classifiers.

Figure 3.5: RVM Confusion Matrix along with corresponding recall (main diagonal) and precision rates (last row) for the original aerobics dataset.
The clustering process of section 3.2, clusters the detected salient points into salient regions by selecting the point with the highest saliency value in the set as the starting point. In order to examine the sensitivity of the proposed method with respect to the estimates of the saliency values, we performed the same classification experiments as before, but with noisy versions of the original unclustered representations. More specifically, we added Gaussian noise of zero mean and variance $\sigma$ to the saliency values of the detected salient points. The resulting representations were clustered once again using the process of section 3.2. Salient points whose saliency was less than zero after the noise addition were not taken into account during the clustering process. The overall recognition rate that was achieved for the initial aerobics dataset, for five levels of noise of increasing variance, is plotted in Fig. 3.8. From the figure, we conclude that the saliency values of the detected salient points carry important information, since the performance deteriorates as the noise increases. However, the deterioration is not very large, considering the amount of added noise.

We also compare the results achieved by the proposed method with the ones achieved using the temporal templates of Bobick and Davis [18]. In [18], each single-view test example was matched against seven views of each example in the training set, which in turn, was performed several times by an experienced aerobics instructor. A performance of 66.67% (12 out of 18 moves) was reported. Our training set, however, consists of single view examples, performed several times by non expert subjects. Furthermore, noise and shadow effects in the sequences of our dataset create small, non-zero pixel regions in areas of the corresponding MEIs and MHIs where no motion exists. The overall recognition rate that was achieved in the aerobics dataset, using the method in [18] was 46.71%. Removal of most of the spurious areas with different simple morphological operations (removal of small connected components) led in deterioration in the overall performance.

Finally, we present, in Table 3.1, the RVM classification results on the aerobics dataset for the two-step approximation of a full search, presented in section 3.2. As can be seen, the recall and precision rates are lower than the ones corresponding to the full search approach, leading to an overall recognition rate of 73.68%. However the reduction is low and therefore it remains a good alternative to the full search method when faster recognition is required.
3.5 Conclusions

In this chapter we presented a novel method for the representation of human activities depicted in given image sequences. The proposed representation is based on the detection of a sparse set of spatiotemporal features that, loosely speaking, correspond to activity variation peaks. The proposed features are an extension of the concept of saliency in the temporal domain. That is, they are detected by measuring the variations in the information content of neighboring pixels not only in space but also in time. Furthermore, we devised an appropriate distance measure between sparse representations containing different numbers of features, based on the Chamfer distance. The proposed distance measure allows us to use an advanced kernel-based classification scheme, the Relevance Vector Machine. In order to achieve invariance against scale changes, we proposed an iterative space-time warping method.

We presented results on real image sequences that illustrate the consistency in the spatiotemporal localization and scale selection of the proposed method. Classification results are presented for two different types of classifiers, displaying the efficiency of the representation in discriminating actions of different motion classes. Furthermore, the classification results clearly illustrate the superiority of the proposed kernel-based classification scheme over the simple kNN classification.

The main limitation of the proposed method lies on the requirement that the camera and the background of the scene are static. For one, the presence of a moving camera and background clutter would generate a large number of salient points in areas other than the ones covered by the human activity, rendering the proposed recognition scheme ineffective. Detecting the salient points using the
Figure 3.7: Detected spatiotemporal features in two misclassified image sequences for three time instances, $t_i, i = 1 \ldots 3$. The sequences correspond to two different action classes that are performed by the same subject.

Figure 3.8: Overall recognition rate, with respect to the sigma of the Gaussian noise that was added to the saliency values of the detected salient points prior to clustering.
optical flow of the image sequence instead of the raw pixels would alleviate this problem, as long as the optical flow vectors are compensated for the global motion of the camera. This approach is followed in the representations used in chapters 5 and 6 of this thesis. Even if the camera is static, the presence of dynamic background has a similar effect, that is, the detection of salient points at areas other than the subject. The use of codebooks of visual words partly deals with this problem. The codebooks are pre-learned using ‘clean’ sequences. During testing, salient points detected on the background will have relatively large matching cost with the ones belonging to the codebook, suppressing in this way their influence in the final classification. The effect of dynamic background is mostly dealt with in chapter 6. Finally, despite the fact that the proposed salient points (and any kind of interesting points in general) contain valuable information concerning the spatiotemporal localization of a human activity, they provide no information over the temporal evolution of the activity itself. That is, no information is provided over the direction of motion, or how the information engulfed by the detected points evolves over time. This problem is dealt with in the next chapter, with the application of tracking.
Chapter 4

Tracking

In the previous chapter we discussed an important limitation of the acquired spatiotemporal salient point representations: the absence of information about the temporal evolution of each detected salient point. Such information is important, since it contains valuable information over the dynamics of a human activity. In this chapter we propose to use tracking in order to address this issue. Our goal is to create a representation based on a set of trajectories, each corresponding to the temporal transitions of certain points located on the human body. A number of works in the literature (e.g. [149] [150]) consider these temporal transitions to be known a-priori, and focus solely on the problem of recognition. By contrast, in this chapter, we deal with both problems, that is, both tracking and recognition. More specifically, we propose to use Auxiliary Particle Filtering (APF) [95] in order to track in time points located on subjects performing human activities. We use the illumination-invariant color distance with shape deformation of [96] as the observation model of the particle filter, and we augment it with information over the background. We do this in order to deal with imperfect localization of the tracked templates. This is particularly evident when the latter are localized on the automatically detected salient points of chapter 3, since, due to the use of temporal derivative filters for their detection, they are localized on the motion boundary of the activities, rather than on the subjects. By augmenting the observation model of the tracker with background information, we enforce the tracker to favor solutions that contain a small number of background pixels, thus reducing their influence and preventing the template from getting stuck in the background. In addition, we perform tracking and recognition experiments using points localized on specific body regions, like the hands and the head of the subjects performing the actions. For these experiments, we further augment the observation model of the tracker using skin color cues. We propose a variant of the Longest Common
Subsequence algorithm (LCSS) [151], [152] in order to acquire a similarity measure between different sets of trajectories. Finally, in order to address the classification problem, we propose new kernels for a Relevance Vector Machine (RVM) classification step [5]. The basis of these kernels is the LCSS similarity between the trajectory sets.

The remainder of the chapter is organized as follows: in section 4.1 we present the utilized tracking method, that is based on auxiliary particle filtering. Section 4.2 discusses the various observation models used in the particle filter, namely the illumination-invariant color distance of [96] and the background and skin color models, while section 4.3 presents the utilized particle propagation model. In section 4.4 we present the modified LCSS algorithm used in order to compare the derived trajectory sets. We present our experimental results in section 4.5, and in section 4.6, final conclusions are drawn, and limitations of the proposed algorithm are discussed.

4.1 Auxiliary Particle Filtering

Recently, particle filtering tracking schemes [87], [95], have been successfully used [153] [154] [96] in order to track the state of a temporal event given a set of noisy observations. Its ability to maintain simultaneously multiple solutions, called particles, makes it particularly attractive when the noise in the observations is not Gaussian and makes it robust to missing or inaccurate data.

Let us denote by \( \alpha \) the unknown location of the feature that is being tracked at the current time instant and by \( Y = \{\ldots, y^-, y\} \) the observations up to the current time instant. The main idea of particle filtering is to maintain a particle based representation of the a posteriori probability \( p(\alpha | Y) \) of the state \( \alpha \) given \( Y \). The distribution \( p(\alpha | Y) \) is represented by a set of pairs \((s_k, \pi_k)\), such that if \( s_k \) is chosen with probability equal to \( \pi_k \), then it is as if \( s_k \) was drawn from \( p(\alpha | Y) \). Our knowledge about the a posteriori probability is updated in a recursive way. Suppose that we have a particle based representation of the density \( p(\alpha^- | Y^-) \), that is we have a collection of \( K \) particles and their corresponding weights (i.e. \((s_k^-, \pi_k^-))\). Then, the Auxiliary Particle Filtering can be summarized as follows:

1. Propagate all particles \( s_k^- \) via the transition probability \( p(a^- | a^-) \) in order to arrive at a collection of \( K \) particles \( \mu_k \).

2. Evaluate the likelihood associated with each particle \( \mu_k \), that is let \( \lambda_k = p(y | \mu_k) \).
3. Draw \( K \) particles \( s_k^- \) from the probability density that is represented by the collection \( (s_k^-, \lambda_k, \pi_k^-) \).

In this way, the auxiliary particle filter favors particles with high \( \lambda_k \), that is particles which, when propagated with the transition density, end up at areas with high likelihood.

4. Propagate each particle \( s_k^- \) with the transition probability \( p(\alpha|\alpha^-) \) in order to arrive at a collection of \( K \) particles \( s_k' \).

5. Assign a weight \( \pi_k' \) to each particle as follows,

\[
 w_k' = \frac{p(y|s_k')}{\lambda_k}, \quad \pi_k' = \frac{w_k'}{\sum_j w_j} \tag{4.1}
\]

This results in a collection of \( K \) particles and their corresponding weights (i.e. \( \{(s_k', \pi_k')\} \)) which is an approximation of the density \( p(\alpha|Y) \).

### 4.2 Observation Model

In this section we define the observation model of the utilized tracker, that is, \( p(y|\alpha; \{m\}) \). This likelihood expresses how well the image content \( y \) can be explained given that the template is at position \( \alpha \), and is parameterized by the set \( \{m\} \). In the case where the tracked templates are initialized at the locations of automatically detected salient points, we define the observation model as the factorization of a color template model \( c \) and a background model \( b \). For the case where the hands and the head of the subject are the features that are being tracked, the observation model is further factorized using a skin color model \( s \).

#### 4.2.1 Invariant Color Distance

In this section the goal is to define the probability \( p(y|\alpha; c) \), where \( c \) is the color template that is to be tracked over time. Let us denote with \( y(\alpha) \) the image patch centered around the spatial position \( \alpha \) in the image \( y \), and with \( y(\alpha, i) \) the color of pixel \( i \) of patch \( y(\alpha) \). Similarly, let us denote with \( c(i) \) the color at the pixel \( i \) of the color template \( c \). In order to define the likelihood \( p(y|\alpha; c) \), we need to define a distance measure \( d(y(\alpha), c) \) between the template \( c \) and the image patch \( y(\alpha) \). In order to do
so, we adopt the illumination-invariant color distance with shape deformation of [96]. According to this model, $d(y(\alpha), c)$ is defined as:

$$d(y(\alpha), c) = \min_{\Phi} (d_c(c, y(\alpha, \Phi))(1 + \lambda d_f(\Phi)^p)), \quad (4.2)$$

where $\Phi : N^2 \rightarrow N^2$ is a transformation function that expresses the deformation of template $c$ and gives the correct correspondence between the pixel coordinates of the color template $c$ and the image patch $y(\alpha)$. The first term of the product is used to penalize large color-based distance, and the second term is used to penalize large shape deformations $\Phi$. The parameters $\lambda$ and the exponent $p$ control the relative importance of the shape deformation term. We should note that there are also alternative ways of combining the color-based distance and the shape-based deformation cost, e.g. by considering the minimum of their sum, i.e. $d(y(\alpha), c) = \min_{\Phi} (d_c(c, y(\alpha, \Phi)) + \lambda d_f(\Phi))$. However, as stated by the authors of [96], it is easier to tune the parameter $\lambda$ using the expression of eq. 4.2. In this work, we used the same values as in [96] for the $\lambda$ and $p$ parameters, that is, $\lambda = 0.1$ and $p = 0.3$.

The color distance is defined to be invariant to local changes in the intensity by normalizing each color template with the average intensity. Formally, the color-based difference is defined as follows:

$$d_c(c, y(\alpha, \Phi)) = \frac{1}{\sigma_c} E_i \left\{ \| c(i) - \frac{y(\alpha, \Phi(i))}{E\{y(\alpha)\}} \|_1 \right\}, \quad (4.3)$$

where $E\{c\}$ is the average intensity of the color template $c$, $\| \cdot \|_1$ is the $L_1$ norm and $\sigma_c$ is a scaling factor. The second term of eq. 4.2 is used in order to penalize large deformations. Formally, $d_f$ is defined as the average Euclidean distance over the pixel based displacements:

$$d_f(\Phi) = E_i \left\{ \sqrt{\| i - \Phi(i) \|_2^2} \right\}, \quad (4.4)$$

where $i$ denotes pixel coordinates. Since both color difference and the deformations of neighboring pixels are considered independently, the minimization of eq. 4.2 can be performed independently for each pixel. The transformation $\Phi^*$ that minimizes eq. 4.2 is given by:

$$\Phi^*(i) = \arg \min_j \frac{1}{\sigma_c} \left\{ \| c(i) - \frac{y(\alpha, j)}{E\{y(\alpha)\}} \|_1 \right\} (1 + \lambda \sqrt{\| i - j \|_2^2}). \quad (4.5)$$
Finally, the observation likelihood \( p(y|\alpha; c) \) can be defined as:

\[
p(y|\alpha; c) = \max \left\{ \epsilon, \frac{1}{Z} e^{-d(y(\alpha); c)} \right\},
\]

where \( Z \) is a normalization term and \( \epsilon \) is a constant that is used in order to deal with large occlusions, and effectively assumes a uniform likelihood at occlusions or when the tracking is lost.

### 4.2.2 Background Model

In this section the goal is to define the probability \( p(y|\alpha; b) \), where \( b \) expresses a background model that is learned over the input image sequence. Our motivation for incorporating background information into the observation model of the tracker stems from the imperfect localization of the initial templates to be tracked. This problem is more prominent in the case of salient region tracking. Indeed, since the input signal that is used for the salient point detection is the convolution of the input image sequence with a Gaussian derivative filter along the temporal dimension (see eq. 3.1), a large number of detected salient points are localized at the edges of the moving objects existing in the scene, rather than on the objects themselves. This fact may deteriorate the output of the tracker, since the patches of the sequence that are being tracked may include a considerable portion of the scene’s background.

We adopt the adaptive background estimation algorithm of [17] in order to determine which pixels belong to the foreground and which ones to the background. According to this algorithm, the values of a particular pixel over time are considered as a temporal process. The recent history of each pixel is modeled by a mixture of \( K \) Gaussian distributions, whose parameters are estimated using the Expectation-Maximization (EM) algorithm and by using a small portion of the available data (i.e. the first few frames of the image sequence). Examples of backgrounds estimated this way are shown in Fig. 4.1, where the value of each pixel in the depicted images was calculated as the weighted sum of the learnt Gaussian means corresponding to that pixel. As can be seen from the figure, parts of the body that do not present significant motion are also considered part of the background. On the other hand, fast moving parts (e.g. hands) are considered to belong to the foreground and are not included in the estimation.

We can define \( p(y|\alpha; b) \) as:
Figure 4.1: Examples of estimated backgrounds. Each pixel in the images is the result of the weighted sum of the learnt Gaussian means corresponding to that pixel.

\[
p(y|\alpha; b) = \frac{1}{1+e^{-r(d_b(y(\alpha), b) - q)}}, \tag{4.7}
\]

where \(r, q\) are the sigmoid function parameters and \(d_b(y(\alpha), b)\) is the average Mahalanobis distance between the pixels in the patch \(y(\alpha)\) and the model \(b\), that is:

\[
d_b(y(\alpha), b) = \frac{1}{\sigma_b} E_i \left\{ \sum_{j=1}^{K} w_{ij} \sqrt{(X_i - \mu_{ij})^T S_{ij}^{-1}(X_i - \mu_{ij})} \right\}, \tag{4.8}
\]

where \(X_i\) is the pixel value at position \(i\), \(w_{ij}, \mu_{ij}, S_{ij}\) are, respectively, the weight, mean and covariance of Gaussian \(j\) at position \(i\) and \(\sigma_b\) is a scaling factor. This procedure tends to favor patches that contain a small amount of background pixels, since for these patches, the distance calculated at eq. 4.8, and consequently, the probability assigned to that patch by eq. 4.7, will be large.

### 4.2.3 Skin Model

Similar to the case of the background, we use a mixture of \(K\) Gaussian distributions in order to model the color of human skin. We train the model on approximately 700 labeled frontal facial images from the FERET database [155], and we use EM in order to estimate the parameters of the Gaussians. Our goal is to estimate the probability \(p(y|\alpha; s)\), where \(s\) denotes the skin model. We should note, however, that we apply this type of observation only in the case where the hands and head are being tracked, and not in the case of salient region tracking, since the latter can appear anywhere on the subject, and not only at regions whose color is that of the human skin. Similar to the background, we
can define $p(y|\alpha; s)$ as:

$$p(y|\alpha; s) = 1 - \frac{1}{1 + e^{-r(d_s(y(\alpha), s) - q)}},$$  \hspace{1cm} (4.9)

where $r, q$ are the inverse sigmoid function parameters and $d_s(y(\alpha), s)$ is the average Mahalanobis distance between the pixels in the patch $y(\alpha)$ and the model $s$, computed similarly to eq. 4.8. Since we use an inverse sigmoid function, this procedure tends to favor patches that contain a large amount of skin pixels, assigning a large probability to patches whose distance from the skin distribution is small.

### 4.3 Propagation Model

The role of the propagation model is to express the probability $p(\alpha|\alpha^-)$. This probability models our knowledge about the dynamics of the features, that is, our prior knowledge over their current position $\alpha$ given their position $\alpha^-$ in the previous frame. Usually, constraints over the underlying motion are imposed in the form of first or second order models. In this work, we use a zero order model, modeled as an inverse sigmoid function around the position $\alpha^-$ in the previous frame. That is,

$$p(\alpha|\alpha^-) = 1 - \frac{1}{1 + e^{-r(d_e(\alpha,\alpha^-) - q)}},$$  \hspace{1cm} (4.10)

where $r, q$ are the parameters of the inverse sigmoid function and $d_e(\alpha|\alpha^-)$ is the Euclidean distance between position $\alpha$ and $\alpha^-$. 

### 4.4 Longest Common Subsequence

By tracking a set of points located on a subject, a human activity is represented by a set of trajectories $\{A_i\}, i = 1 \ldots K$, where $K$ is the number of trajectories that constitute the set. Each trajectory is defined as $A_i = \{(x_{i,n}, y_{i,n})\}, n = 1 \ldots N$, where $x_{i,n}, y_{i,n}$ are spatiotemporal coordinates and $N$ is the number of samples that constitute $A_i$. Let us define another trajectory set $\{B_j\}, j = 1 \ldots K'$ representing a different image sequence. Similar to $\{A_i\}$, the trajectories in $\{B_j\}$ are defined as $B_j = \{(x_{j,m}, y_{j,m})\}, m = 1 \ldots M$, where $M$ is the number of samples that constitute $B_j$. We use a variant of the LCSS algorithm presented in [151], in order to compare the two sets. Before we proceed
with the comparison, we transform the $x$ and $y$ coordinates of the trajectories so that they have zero mean. Furthermore, we assume for now an aligning procedure that aligns the two sets in time. Let us define the function $Head(A_i) = \{(x_{i,n}, y_{i,n})\}, n = 1 \ldots N - 1$, that is, the individual trajectory $A_i$ reduced by one sample. We use a modified LCSS similarity measure in order to compare trajectories $A_i$ and $B_j$, defined as:

$$
LCSS_{\delta,\varepsilon}(A_i, B_j) = \begin{cases} 
0, & \text{if } A_i \text{ or } B_j \text{ is empty} \\
\exp(-\alpha d_e((x_{i,n}, y_{i,n}), (x_{j,m}, y_{j,m}))) \\
+ LCSS_{\delta,\varepsilon}(Head(A_i), Head(B_j)), & \text{if } |x_{i,n} - x_{j,m}| < \varepsilon \text{ and } |y_{i,n} - y_{j,m}| < \varepsilon \text{ and } |n - m| < \delta \\
\max(LCSS_{\delta,\varepsilon}(Head(A_i), B_j), LCSS_{\delta,\varepsilon}(A_i, Head(B_j))), & \text{otherwise}
\end{cases}, \quad (4.11)
$$

where $d_e$ is the Euclidean distance, $\varepsilon$ is the matching threshold and $\delta$ is a constant which defines how far in time we can go in order to match a point from one trajectory to a point in another trajectory. The main modification of the utilized measure with respect to the classic LCSS similarity is the use of the exponent of the Euclidean distance in the case where there is a match between the points of the compared trajectories. This was performed in order to take the actual distance of the points into account, and make in this way comparison softer. We used a dynamic programming algorithm in order to efficiently compute the measure of eq. 4.11 and determine which parts of the compared trajectories are common to each other. The notion of the LCSS similarity of eq. 4.11 for the one-dimensional case is depicted in Fig. 4.2.

Subsequently, the similarity between sets $\{A_i\}$ and $\{B_j\}$ is defined as follows:

$$
S_L(\{A_i\}, \{B_j\}) = \frac{1}{K} \sum_i \max_j LCSS_{\delta,\varepsilon}(A_i, B_j) + \frac{1}{K'} \sum_j \max_i LCSS_{\delta,\varepsilon}(B_j, A_i), \quad (4.12)
$$

that is, the average over the set of the maximum similarities, as they have been defined in eq. 4.11, between the $K$ trajectories of set $\{A_i\}$ and the $K'$ trajectories of set $\{B_j\}$. 

4.5 Experimental Results

We use the aerobics2 dataset described in chapter 3 in order to evaluate the proposed method, due to its high recording resolution in both space and time. The dataset consists of 15 different aerobic exercises, performed twice by five different subjects, leading to a set of 150 sequences. The results presented in this section consist of two parts. The first part involves tracking of body features, like the hands and head of the subjects performing several activities. The hands and head are localized at the first frame and are subsequently tracked throughout the image sequence. In the second experiment, we track the salient regions that are automatically detected using the process of chapter 3. Tracking is performed for a small number of frames, leading to a representation of short trajectories, each of which corresponds to the temporal evolution of each salient region. In what follows, we describe each of these experiments separately.

4.5.1 Body Feature Tracking

In this section, we perform tracking and recognition experiments using a set of pre-defined points positioned on the human body, e.g. the hands and the head of a subject performing an activity. We use a combination of the created background and skin models in order to automatically localize these points at the first frame. More specifically, using background subtraction and morphological operations, we initially localize a blob around the subject in the first frame. Subsequently, by taking into account the pixels that are labeled as skin within the blob, we localize the points to be tracked,
Figure 4.3: Automatic localization of the hands and the head on the first frame of an image sequence. By subtracting the estimated background in (b) from the first frame in (a), a blob around the subject is extracted, depicted in (c). Subsequently, by combining the result at (c) with the pixels labeled as skin in (d), an estimate over the location of the hands and head is acquired. This is depicted in (e).

as the three largest connected regions consisting of skin pixels. This process is tailored for the utilized dataset, since all of the actions start at an almost neutral state, that is, the subject approximately standing upright, facing the camera. An example of this procedure is depicted in Fig. 4.3.

We subsequently proceed with the evaluation of the proposed tracker, concerning both localization and classification accuracy. To provide ground truth for our experiments, every 5th frame is manually labeled by a human operator. We use the following measure as a criterion for success:

\[
m_c = \sum_{i=1}^{n} h_i, \tag{4.13}\]

where \( h_i = 1 \) if the Euclidean distance of the computed point exceeds a predefined threshold and 0 otherwise. We set the threshold equal to the size of the template used for the tracking of the point. In other words, we consider that an error has occurred if the tracked point is sufficiently far from the ground truth.

For this experiment, we use as our observation model the combination of the invariant color distance between the templates along with background and skin color information. That is, we define as our observation model the following product:

\[
p(y|s_k) = \alpha p(y|s_k; c)p(y|s_k; b)p(y|s_k; s), \tag{4.14}\]

where \( p(y|s_k; c) \), \( p(y|s_k; b) \) and \( p(y|s_k; s) \) are given by eq. 4.6, eq. 4.7 and eq. 4.9 respectively. Using this observation model, the particle filter favors solutions that are similar to the tracked template \( c \), and contain large amounts of foreground and skin pixels.
4.5. Experimental Results

To illustrate the effectiveness of the proposed observation model, we present, in Fig. 4.4, comparative results between the proposed tracking scheme and the Condensation and simple APF algorithms. In the latter two methods, only the invariant color distance of eq. 4.6 was incorporated into their observation model. As can be seen from the figure, the proposed tracker managed to track without error almost 52% of the videos in the database, while in the case of Condensation and simple APF trackers, the percentage was about 28%. Moreover, for almost 80% of the videos in the database, the cumulative error in terms of percentage of frames in which tracking failed in each video was 20%, while for the case of Condensation and simple APF, this error rose to about 35%. This result clearly illustrates the performance improvement provided by the adapted observation model. Another interesting observation from Fig. 4.4 is that the Condensation algorithm outperforms the simple APF, even though the latter was designed in order to overcome one of the major drawbacks of Condensation, that is, a large number of particles being propagated in areas with small likelihood. This is due to the fact that, contrary to Condensation, the simple APF does not utilize any motion model, and the particles are thrown with equal probability around the point at the current time instant.

An example depicting the tracking result using the proposed observation model is depicted in Fig. 4.5. As can be seen from the figure, using just the color similarity in the observation model is not sufficient, since in this case, tracking of the raising right hand is eventually distracted by the background, which has a similar color to the tracked template (marked with a white rectangle). The latter corresponds to the palm of the subject. Incorporating a background model seems to alleviate this problem, since particles lying on it are assigned a low probability. The result is the tracker being able to follow the tracked template until the end of the activity.
Figure 4.5: Tracking result for one of the sequences in the database, where the subject is raising its right hand, reaches an apex and returns to the initial position. Green: APF incorporating the proposed observation model. Red: Tracking using only color similarity in the observation model. In the first case, the algorithm is distracted by the background, and tracking is eventually lost. Incorporation of the background and skin models alleviates this problem.

We use Relevance Vector Machines (RVM) in order to classify the examples in the utilized dataset, with a kernel defined by the LCSS similarity of eq. 4.12. Before computing the LCSS similarity between the individual trajectories, we transform their \( x \) and \( y \) coordinates so that they have zero mean. Furthermore, we assume that the temporal differences between the image sequences in the database can be approximated by a linear process, and therefore we stretch the trajectories in time so that they have the same duration. In effect, this is performed by resampling the acquired trajectories so that they consist of a pre-determined number of points. We constructed 15 different RVM classifiers, one for each class, and we calculated for each test example the conditional probability \( p_i(l|F), 1 \leq i \leq 15 \), where \( l \) denotes the class and \( F \) the example. Each example was assigned to the class for which the corresponding classifier provided the maximum conditional probability. We followed a leave-one-out subject cross validation scheme, that is, for estimating each of the \( p_i(l|F) \), an RVM is trained by leaving out the example \( F \) as well as all other instances of the same exercise that were performed by the subject from \( F \). We performed this experiment for all of the trackers under comparison. Our classification results are depicted in Table. 4.1. As can be seen from the table, incorporating the proposed observation model led to an increase of almost 5% in classification performance compared to Condensation and classic APF.
Table 4.1: Recall and Precision rates for the compared trackers.

<table>
<thead>
<tr>
<th>Class Labels</th>
<th>COND. R/P</th>
<th>APF R/P</th>
<th>APF+Obs. R/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9 / 1</td>
<td>0.9 / 0.75</td>
<td>1 / 1</td>
</tr>
<tr>
<td>2</td>
<td>0.2 / 0.14</td>
<td>0 / 0</td>
<td>0.6 / 0.5</td>
</tr>
<tr>
<td>3</td>
<td>0.9 / 0.81</td>
<td>1 / 1</td>
<td>1 / 0.76</td>
</tr>
<tr>
<td>4</td>
<td>0.5 / 0.71</td>
<td>0.6 / 0.75</td>
<td>0.7 / 0.7</td>
</tr>
<tr>
<td>5</td>
<td>0.6 / 0.85</td>
<td>0.9 / 0.81</td>
<td>0.7 / 1</td>
</tr>
<tr>
<td>6</td>
<td>0.9 / 0.81</td>
<td>0.8 / 0.72</td>
<td>0.7 / 0.58</td>
</tr>
<tr>
<td>7</td>
<td>0.4 / 0.33</td>
<td>0.4 / 0.23</td>
<td>0.6 / 0.5</td>
</tr>
<tr>
<td>8</td>
<td>1 / 0.9</td>
<td>1 / 0.9</td>
<td>1 / 0.9</td>
</tr>
<tr>
<td>9</td>
<td>1 / 0.9</td>
<td>1 / 0.77</td>
<td>1 / 0.9</td>
</tr>
<tr>
<td>10</td>
<td>0.6 / 0.35</td>
<td>0.2 / 0.2</td>
<td>0.4 / 0.4</td>
</tr>
<tr>
<td>11</td>
<td>0.5 / 1</td>
<td>0.3 / 0.6</td>
<td>0.4 / 0.8</td>
</tr>
<tr>
<td>12</td>
<td>0.8 / 0.72</td>
<td>0.7 / 0.78</td>
<td>0.7 / 0.78</td>
</tr>
<tr>
<td>13</td>
<td>0.5 / 0.83</td>
<td>0.8 / 1</td>
<td>0.5 / 1</td>
</tr>
<tr>
<td>14</td>
<td>1 / 0.83</td>
<td>1 / 1</td>
<td>1 / 0.83</td>
</tr>
<tr>
<td>15</td>
<td>0.4 / 0.67</td>
<td>0.6 / 0.5</td>
<td>0.7 / 0.64</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.68 / 0.72</td>
<td>0.68 / 0.66</td>
<td>0.73 / 0.75</td>
</tr>
</tbody>
</table>

Finally, we notice that the classification performance achieved using the proposed scheme is inferior to the 93% rate achieved for this dataset using the process of chapter 3. The main reason is the inadequacy of the utilized three-point representation to describe activities in which other parts, or the whole of the body are involved. Furthermore, errors in the automatic initialization of the templates around the hands and head of the subjects, according to the procedure depicted in Fig. 4.3, lead to erroneous trajectories, and the eventual misclassification of the corresponding sequences.

### 4.5.2 Salient Region Tracking

In this section we proceed with the tracking of the automatically detected salient regions of chapter 3. Since the majority of these regions are localized at non-skin areas of the body, we do not include the human skin model in the observation model of the utilized tracker. Formally, the observation model that we use is formulated as:

\[ p(y|s^k) = \alpha p(y|s^k; c)p(y|s^k; b), \]  

(4.15)  

where \( p(y|s^k; c) \) and \( p(y|s^k; b) \) are given by eq. 4.6 and eq. 4.7 respectively. Using this observation
model, the particle filter favors solutions that are similar to the tracked template \( c \), and contain large amounts of foreground pixels.

In Fig. 4.6 we present the trajectories that were extracted from two different activities, along with a snapshot of the corresponding actions. As can be seen from the figure, the extracted trajectory set seems to correctly capture the pattern of the motion performed. This can easily be observed from the arch-like trajectories of the lower part of the figure, which correspond to the motion of the subject’s hands. However, there are several regions for which tracking fails, like for instance, regions lying on the boundary between the background and the subject, where the color is uniform, e.g. due to clothing (see Fig. 4.6(a)). In this case the tracked template moves along the boundary instead of following the actual motion, resulting in an erroneous trajectory.

Similar to section 4.5.1, we use Relevance Vector Machines (RVM) in order to classify the examples
4.6 Conclusions

Table 4.2: Recall and Precision rates for the RVM classifier.

<table>
<thead>
<tr>
<th>Class Labels</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVM R/P</td>
<td>1/1</td>
<td>0.4/0.6</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>0.5/0.45</td>
<td>0.5/0.4</td>
<td>1/1</td>
</tr>
<tr>
<td>Class Labels</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>Total</td>
</tr>
<tr>
<td>RVM R/P</td>
<td>0.88/1</td>
<td>0.63/0.63</td>
<td>0.63/0.83</td>
<td>0.88/0.88</td>
<td>0.3/0.7</td>
<td>1/1</td>
<td>0.9/0.8</td>
<td>0.7747/0.82</td>
</tr>
</tbody>
</table>

in the utilized dataset, with a kernel defined by the LCSS similarity of eq. 4.12. Before computing the LCSS similarity between the individual trajectories, we transform their $x$ and $y$ coordinates so that they have zero mean. Moreover, using the original salient point representations and the space-time warping process of section 3.3.1, we align the resulting trajectories pair-wise, that is, for each pair of examples in the dataset. Similar to section 4.5.1, we train one classifier for each class, and use a leave-one-subject-out cross validation scheme in order to classify a test example. The classification results achieved for each class in the form of recall and precision rates are depicted in Table 4.2.

From the table we conclude that there is a 3.5% increase in overall recall rate using short trajectories compared to the body feature tracking experiments of Table 4.1. Considering, however, the richness of the utilized short trajectory representation, compared to the 3 point tracking of section 4.5.1, this increase is small, and is attributed to several erroneous trajectories that are additionally acquired, like the ones of Fig. 4.6.

4.6 Conclusions

In this chapter we introduced an adapted Auxiliary Particle Filter in order to track human body regions, for the purpose of human activity recognition. We incorporated background and skin color cues in order to enhance the observation model of the utilized tracker. In this way, the tracker favors particles that contain a small number of background and/or skin pixels. Each human activity is then represented as a set of trajectories. We have shown that the proposed tracking scheme outperforms the Condensation and classic APF algorithms, when the tracked regions are the hands and the head of the subject. Furthermore, we performed experiments in which the tracked templates are localized on the automatically detected spatiotemporal salient points of chapter 3, and created representations based on sets of short trajectories, where each trajectory is created by tracking a single salient point for a number of frames. We proposed to use a modified LCSS similarity measure in order to compare the trajectory sets that were derived by tracking, and we used this measure in order to define suitable kernels for an RVM classification scheme.
Despite the improvement that was achieved in tracking performance with the proposed observation model, the classification rate achieved was by far lower than the one achieved by using just the salient point representations of chapter 3. That is, instead of leading to more informative representations, application of tracking led to a deterioration in overall classification performance. We attributed this result to several errors occurring during the tracking process, leading to erroneous trajectories that do not express the actual motion that is taking place. One of the most common types of error stems from changes in appearance of the tracked template, e.g. due to rotations, deformations or partial occlusions, making the template no longer an accurate model of the appearance of the tracked object. Updating the template fully or partially (e.g. through exponential forgetting) at each frame is a potential solution to this problem. This approach, however, may lead to error accumulation as time evolves, leading to a high probability that the template will eventually drift away from the object and tracking to be lost. Furthermore, a number of errors stem from tracking of regions that are not informative enough, like, for instance, regions lying between the background and parts of the body that have a uniform color, e.g. due to clothing, as depicted in Fig. 4.6(a). In this case, the tracked template can move anywhere along the boundary, and the final trajectory that is acquired does not correspond to the actual motion of the subject.
Chapter 5

B-Spline Representations

In the previous chapter, we described a method in which a human activity was represented by a set of short trajectories, where each trajectory was derived by tracking a salient point for a short period of time. As such, each trajectory was treated independently, that is, the motion of each salient point was assumed to be independent of the motion of its neighbors. However, given the smooth motion of the subjects performing the activities, it makes sense to assume that neighboring salient points follow a similar motion. In this chapter we propose a method for human activity representation and recognition that is based on this assumption. More specifically, we propose to establish temporal correspondences between salient points that fall within local spatiotemporal neighborhoods, by locally fitting three-dimensional piecewise polynomials, namely B-Splines, on these points. Each local neighborhood is centered at a salient point, and its dimensions are proportional to the detected space-time scale of the point. Contrary to previous chapters, we detect the salient points on filtered versions of the optical flow field, in order to deal with motion induced by a moving camera. More specifically, we first estimate the optical flow using the algorithm proposed in [156], and we locally subtract the median of the optical flow vectors, estimated within a local window. The local nature of the filtering process that we apply, helps us reduce the influence of motion components that are due to global translational motion and vectors that originate from more general camera motion, like rotation and scaling. The salient points that we therefore extract, correspond to areas where independent motion occurs, like ongoing activities in the scene. After fitting each B-spline, we extract a set of novel visual descriptors, derived as the partial derivatives of the resulting polynomial. Subsequently, the set of descriptors extracted from each spline is accumulated into a number of histograms. This number depends on the maximum degree of the partial derivatives. Since our descriptors correspond to geometric properties
Chapter 5. B-Spline Representations

of the spline, they are translation invariant. Furthermore, the use of the automatically detected space-time scales of the salient points for the definition of each local neighborhood ensures invariance to space and time scaling. Similar to other approaches (e.g. [36] [77]), where a codebook of visual words is created from a set of appearance descriptors, we create a codebook of visual verbs by clustering our descriptors across the whole dataset. Here, we use the term ‘verb’ instead of a ‘word’ for our codebook entries, since each entry corresponds to a combined shape and motion descriptor rather than just a shape descriptor. Instead of representing the whole sequence as a histogram of codewords, we use the resulting codebook in order to represent temporal slices, were each slice consists of the descriptors that are centered at each frame of the image sequence. Subsequently, we use boosting in order to select a set of characteristic temporal slices for each class. Finally, we use a Relevance Vector Machine (RVM) classifier [5], in order to classify test examples into one of the classes present in the training dataset.

We evaluate the proposed method using three different databases of human actions. These include the widely used Weizmann [10] and KTH [11] datasets as well as the aerobics2 dataset of chapter 3. Finally, we present experiments aimed at evaluating the generality of our descriptors, that is, their ability to encode and discriminate between unseen actions, coming from an entirely different dataset than that on which the method is trained. A list of the successive steps of our algorithm is given in Table 5.1.

The remainder of the chapter is organized as follows. In Section 5.1 we describe our feature extraction process. This includes the optical flow computation, the detection of the salient points, the subsequent B-spline fitting and the creation of the visual codebook. In Section 5.2 we present our classification method, including the feature selection procedure that we applied for selecting the most discriminant time windows of each class. In Section 5.3 we present the experimental results and in Section 4.6 we draw some conclusions.

5.1 Representation

In this section we introduce the visual descriptors that we use in order to represent an image sequence. We will initially provide some basics on B-splines and we will subsequently describe how they are used in extracting local, spatiotemporal, image-sequence descriptors. Finally, we will briefly explain the process that we carry out in order to create a codebook from these descriptors.
Table 5.1: Successive steps of the proposed approach.

1. Compute the optical flow according to the algorithm of [156] (Fig. 5.2b), and compensate for camera motion using local median filters (Fig. 5.2d).
2. Detect spatiotemporal salient points on the resulting flow field using the algorithm of [3] (Fig. 5.2c).
3. Place each salient point at the center of a space-time cube with dimensions proportional to the space-time scale of the point (Fig. 5.4a).
4. Fit a B-spline polynomial on the salient points that fall within the space-time cube (Fig. 5.4b).
5. Compute the partial derivatives of the resulting polynomial (Fig. 5.5).
6. Bin the computed partial derivatives into a histogram and form a descriptor vector for each B-spline polynomial.
7. Create a codebook of $K$ verbs by clustering the resulting descriptor vectors across the whole dataset.
8. Perform feature selection using the Gentleboost algorithm [128] in order to select the most informative temporal slices for each class.

5.1.1 B-spline Surfaces

Let us define an $M \times N$ grid of control points $\{P_{ij}\}, i = 1 \ldots M$ and $j = 1 \ldots N$. Let us also define a knot vector of $h$ knots in the $u$ direction, $U = \{u_1, u_2, \ldots, u_h\}$ and a knot vector of $k$ knots in the $v$ direction, $V = \{v_1, v_2, \ldots, v_k\}$. Then, a B-spline surface of degrees $p$ and $q$, in the $u$ and $v$ directions respectively, is given by:

$$F(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} Q_{i,p}(u)Q_{j,q}(v)P_{ij}, \quad (5.1)$$

where $Q_{i,p}(u)$ and $Q_{j,q}(v)$ are B-spline basis functions of degree $p$ and $q$, respectively, defined as:

$$Q_{i,0}(u) = \begin{cases} 1 & \text{if } u_i < u < u_{i+1} \\ 0 & \text{otherwise} \end{cases},$$

$$Q_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i}Q_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}}Q_{i+1,p-1}(u). \quad (5.2)$$

The grid of control points is also referred to as the control net, and defines the convex hull of the B-spline surface. This is a direct extension of the Strong Convex Hull property of B-spline curves, which states that a B-spline curve is contained in the convex hull of its control polyline. More specifically, the Strong Convex Hull property states that if $u$ is in knot span $[u_i, u_{i+1})$, then $F(u)$ is in the convex hull of control points $P_{i-p}, P_{i-p+1}, \ldots, P_i$. The knot vectors define the points at which the surface is
evaluated. As a piecewise polynomial, a B-spline surface is composed of a number of surface segments, each of which is defined on a knot span. Consequently, modifying the position of the knots will change the shape of the surface. In this work we use uniform B-splines, that is, splines with equidistant knot vectors.

The number of the knots at each direction defines the degree of the B-spline at that direction. Formally, given a grid of $M \times N$ control points, the degree of the fitted B-spline surface along the $u$ and $v$ directions of the $h$ and $k$ knots respectively, will be $(M - h, N - k)$. It is well known that increasing the order of a B-spline surface increases its smoothness, and the surface tends to move further away from its control net, i.e. its control points. An example is depicted in Fig. 5.1, where B-spline surfaces of 3rd and 6th degree are fitted on the same set of control points. In this work, we would like to avoid precise fitting of the surfaces to the control points, as this would lead to overfitting and would make the surface sensitive to noise. However, we would like the fitted surface to be close enough to the control points, as these describe the motion that is taking place in the scene. As a good tradeoff between descriptiveness and robustness we use in this work 3rd degree polynomials, that is, $p = q = 3$.

### 5.1.2 Optical Flow

Our analysis relies on the motion field that is estimated using an optical flow algorithm. Our goal is to detect spatiotemporal interest points and subsequently extract spatio(temporal) descriptors at areas with significant variation in motion information, such as motion discontinuities, rather than at areas with significant spatiotemporal intensity variation, such as space-time intensity corners (see
5.1. Representation

The latter approach generates too many points in the case of camera motion in a moving, textured background. By contrast, as long as the camera motion is smooth the spatiotemporal salient point detection at motion discontinuities should be invariant to it. We estimate optical flow using the algorithm of [156], due to its robustness to motion discontinuities and to outliers to the optical flow equation.

The presence of general camera motion, like camera translation, small rotations, and scale changes (resulting from camera zoom) makes the application of a motion compensation technique an essential step prior to feature extraction. In this way, the extracted features will describe solely the independent motion taking place in the scene, like human activities. In the proposed method we use local median filtering in order to compensate for the local motion component. In a similar way, a global affine motion model can be estimated, and then the corresponding component be compensated for. For both, the goal is to provide representations that are invariant (mainly) to camera motion.

The advantages of global versus local methods for obtaining representations that are invariant to certain transforms (in our case the camera motion) are a subject of ongoing debate in the field of Computer Vision. For example, in order to compensate for changes in the illumination, both local (e.g. local filtering, color-invariant features) and global models (e.g. gamma correction) have been proposed. A clear disadvantage of global parametric models is their sensitivity to outliers (in our case, independently moving objects, including the human subject). On the other hand, the disadvantage of local methods is that they result to representations that may be less descriptive (i.e. ‘too invariant’). For example, after local intensity normalization gray and white areas cannot be distinguished. The motion compensation method that we use in this work falls within the area of local methods, and is very similar to the filtering that is applied in order to compensate for illumination changes, for example, for extracting Quotient Images [157]. The latter are shown to be robust to local intensity variation due to, for example, cast shadows.

Examples of motion compensation are depicted in Fig. 5.2 (local) and Fig. 5.6 (local and global). It can be seen that most vectors that are due to camera motion are suppressed, and the ones corresponding to independent motion in the scene (i.e. the human activities) are pronounced.
5.1.3 Spatiotemporal Descriptors

After compensating for optical flow vectors that are due to camera motion, as explained in section 5.1.2, we use the algorithm proposed in [3] in order to extract a set of spatiotemporal salient points $S = \{(\vec{c}_i, \vec{s}_i)\}$. Here, $\vec{c}_i = (x, y, t)$ is the spatiotemporal position of the point with index $i$. The vector $\vec{s}_i$ is the spatiotemporal scale at which the point was detected and has a spatial and temporal dimension. This scale is automatically detected by the algorithm in [3], as the scale at which the entropy of the signal within the local spatiotemporal neighborhood defined by it is locally maximized. A subset of the salient points detected on a frame of a handwaving sequence is shown in Fig. 5.2c. We should note that for the detection of the points shown in Figure 5.2c, contribute a number of frames before and after the shown frame (temporal scale).

5.1.3.1 Preprocessing

In this section we will describe the preprocessing steps that are followed prior to the B-spline fitting on the detected salient points. In order to fit a B-spline polynomial we first need to define its control net, that is, $P_{ij}$. Formally, for each salient point location we want to fit a polynomial having as control net the points within a small neighborhood around the point in question. For a good fit, however, ordering of the control points in terms of their spatiotemporal location is an important factor in order to avoid loops. In order to make this more clear, let us consider a set of points $L = \{L_i\}$ sampled randomly from an arbitrary curve, as shown in Fig. 5.3(a). Ideally, a polynomial having the set $L$ as its control net would approximate the curve with the one depicted as a dotted line in the same figure. However, in order for this to happen, the points in $L$ should be given in the correct order, that is, $L = \{L_1, L_2, \ldots, L_n\}$, as shown in Fig. 5.3(a). If this is not the case, then the polynomial will attempt to cross the points in a different order, creating unwanted loops. Furthermore, it is clear that any points enclosed by the curve, like the one marked as a triangle in the same figure will also degrade the approximation and should not be considered. In order to overcome these problems, we perform two preprocessing steps on the set $S$ of the detected salient points, both performed frame-wise.

In the first step, we eliminate points that are enclosed within the closed surface defined by the motion boundary. In our implementation, a point lies on the motion boundary if it lacks any neighbors within a circular slice shaped neighborhood of radius $r$, minimum angle $\alpha$, and having the point as origin. This process is demonstrated in Fig. 5.3(b), where the point in the centre of the circle is selected as
5.1. Representation

Figure 5.2: (a) A single frame from a handwaving sequence in which camera zoom is occurring and the corresponding optical flow field, before (b) and after (d) the application of the local median filter. Removal of flow vectors that are due to the camera zoom is evident. (c) Some of the salient points detected using the optical flow field in (d).

In the second step, we order the selected boundary points. We do this by randomly selecting a point on the boundary as a seed and by applying an iterative recursive procedure that matches the seed point with its nearest neighbor in terms of Euclidean distance. This process is repeated using as new seed the selected nearest neighbor, until there are no nearest neighbors left, that is, either an edge has been reached or all points have been selected.

Let us note that the described procedure above is local in nature. The primary role of $r$ is the selection of the points that are on the motion boundary. By properly setting up the radius $r$, the points in the boundary of a moving object will be selected even if there are more than one subjects performing...
activities in the same scene, as long as they are at a distance of at least $r$ pixels from each other. Due to the use of salient point representations (i.e. as the control points for the spline approximations), the presence of noise will minimally affect the boundary selection procedure. Due to the local entropy measurements for the salient point detection, noise will not greatly affect the conveyed information that leads to their detection. While noise may lead to the detection of spurious salient points, their saliency measure will be low compared to the points that belong to the actual motion boundary and therefore, will be discarded by the algorithm described in [3]. In this work we have empirically selected a radius of 10 pixels and an angle of 70 degrees.

5.1.3.2 Spline Approximation

Let us denote with $S' = \{(\vec{c}_i', \vec{s}_i')\}$ the set of spatiotemporal salient points located on the motion boundary, that are obtained by the procedure described in the previous section. For each salient point $(\vec{c}_i', \vec{s}_i')$ we define a spatiotemporal neighborhood centered at $c_i'$ with dimensions proportional to the scale vector $\vec{s}_i'$. Let us denote with $O$ the set of points engulfed by this neighborhood (see Fig. 5.4a). Fitting B-spline surfaces that approximate the points within $O$ (e.g. by using least squares) leads to a quadratic optimization problem for the estimation of the each of the surface’s control points, and could potentially add a significant overhead in the B-spline fitting process. We follow instead a different approach, and fit a B-spline surface using as control points the detected spatiotemporal salient points that fall within each $O$. The latter form a grid on which the B-spline surface is fitted. This grid is not and does not need to be uniform, that is, the pairwise distances of the control points may differ. The knot vectors $U$ and $V$ are a parameterization of the fitted B-spline, and essentially encode the
Figure 5.4: (a) The set of points \( O \) that are engulfed within a spatiotemporal neighborhood. The line connections between the points are for illustration purposes, to depict the ones belonging to the same frame. (b) The resulting B-spline approximation described in section 5.1.1 way in which the B-spline surface changes with respect to its control points. More specifically, the knot vector \( U \) encodes the way the \( x \) coordinates change with respect to \( y \), while the knot vector \( V \) encodes the way both \( x \) and \( y \) change with respect to \( t \).

Using this process, any given image sequence is represented as a collection of B-spline surfaces, denoted by \( \{ F_i(u, v) \} \). Recall that we fit one surface per salient point position and therefore, the number of surfaces per sequence is equal to the number of points in \( S' \). An example of a spline fitted to a set of points is presented in Fig. 5.4. Each member of the set \( \{ F_i(u, v) \} \) is essentially a piecewise polynomial in a three dimensional space. This means that we can fully describe its characteristics by means of its partial derivatives with respect to its parameters \( u, v \). That is, for a grid of knots of dimensions \( k \times h \) we calculate the following matrix \( R_i \) of dimensions \( (pq - 1) \times (hk) \):

\[
R_i = \begin{bmatrix}
\frac{\partial F_i(u_1, v_1)}{\partial u} & \cdots & \frac{\partial F_i(u_h, v_1)}{\partial u} \\
\vdots & \ddots & \vdots \\
\frac{\partial^{(p-1)(q-1)} F_i(u_1, v_1)}{\partial u^{p-1} \partial v^{q-1}} & \cdots & \frac{\partial^{(p-1)(q-1)} F_i(u_h, v_1)}{\partial u^{p-1} \partial v^{q-1}}
\end{bmatrix}
\] (5.3)

where \( \frac{\partial^p}{\partial u^p} \) is the partial derivative of order \( p \) with respect to \( u \). Note (see Eq. 5.1) that \( F_i(u, v) \) is the value of the spline at \( u, v \), that is, \( F_i(u, v) \) is a \( 3 \times 1 \) vector. Consequently, each element of the matrix in Eq. 5.3 is a vector of the same dimensions, and more specifically a vector that specifies
the direction of the corresponding derivative. In Fig. 5.5 an illustration of the first derivatives with respect to \( u \) and \( v \) is given. The derivatives are drawn as three dimensional vectors, superimposed on the spline from which they were extracted.

Our goal is to represent each \( F_i \) with a single descriptor vector. For this reason, we bin each row of \( R_i \) into a single histogram of partial derivatives and we concatenate the resulting \((pq - 1)\) histograms into a single descriptor vector. This vector constitutes the descriptor of \( F_i \) and consequently the descriptor of a specific region in space and time of the image sequence. By repeating this process for each \( F_i \), we end up with a set of descriptors for the whole sequence.

5.1.4 Codebook Creation

Applying a clustering algorithm to the whole set of descriptors, in order to create a codebook, is usually very time and memory consuming. As suggested in [74], the way a vocabulary is constructed has little or no impact to the final classification results. In accordance to this finding, we randomly subsample our descriptor set. Subsequently, we cluster our randomly selected features using K-means clustering. The resulting cluster centers are treated as codewords, and the set of codewords constitutes the utilized codebook. In this work we used 1000 clusters, as a compromise between representation accuracy and speed.
5.2 Classification

Having constructed a codebook, the goal is to represent and classify a test image sequence into one of the available classes in the training set. A conventional application of a ‘bag of verbs’ approach would dictate that each image sequence in the dataset is represented as a histogram of visual codewords drawn from the codebook. Using the codebook in this way for our specific set of descriptors resulted in recognition rates of about 60% or less, using a 1-NN classifier based on the $\chi^2$ distance between the histograms of the test and training sequences. The $\chi^2$ distance was selected as it is more suitable for comparing histograms than the Euclidean distance. The low recognition rate that was obtained using this approach clearly indicates that using a single histogram of codewords to describe a whole sequence is not suitable. The most plausible reason for this is that a large number of descriptors in the codebook is common to many (if not all) classes. We adopt, therefore, a different approach and use the codebook in order to represent temporal slices instead of whole sequences, where each temporal slice consists of the polynomials that are centered at each frame of the image sequence. The descriptors that belong to these slices have a specific extent in time, depending on the temporal scales at which they were extracted (see Section 5.1.3.2).

Even though KNN based classification using a nearest neighbor approach based on the $\chi^2$ distance between the temporal slices works quite well (see Table 5.2), it has an important drawback. A large number of frames in the dataset are likely to be common to many classes and therefore uninformative of the class. For example, for a database depicting aerobic exercises, frames that correspond to the neutral position of the human body, that is, standing upright and facing the camera with the hands resting along the body, are such common yet uninformative frames. It is apparent that temporal slices that are centered on these frames will be matched for all classes that contain them and they cannot be considered characteristic for a specific activity. It is evident, therefore, that a selection step preceding the classification would be highly beneficial in our case.

In this work we use the GentleBoost algorithm [128] in order to select useful features for classification, due to its performance in terms of convergence speed and classification accuracy [127]. Subsequently, we use the selected features in a Relevance Vector Machine (RVM) classification scheme.
5.2.1 Feature selection

In feature selection by GentleBoost, at each stage a weak classifier is trained on a weighted version of the dataset. Here, each weak classifier operates on a different dimension/feature of the feature vector. At each stage the algorithm picks the weak classifier that, given the current sample weights \( w \), separates the examples of different classes best. Then, the classification error is estimated and the samples are reweighted so that misclassified samples weigh more. This procedure is repeated until the classification error does not change significantly between iterations. The performance measure used by GentleBoost to learn the weak classifiers and evaluate their performance is the classification rate, that is, the percentage of correctly classified examples, regardless of their class.

Our goal in this work is to select temporal slices that are characteristic of a specific class. In order to do so, we initially sample \( L \) slices from the examples in the positive set. As mentioned before, each slice is represented as a histogram of codewords from the codebook. Using the \( \chi^2 \) distance, we match each of the positive slices to the remaining ones in the positive set and the ones in the negative set. Our expectation is that slices characteristic of the positive set will have a small distance to the slices belonging to that set, and a larger distance to the slices in the examples belonging to all other classes (i.e. the negative set). In order to make the selection tractable, we only keep the \( N' \) best matches for each sequence. This procedure, results in \( N'M_p \) positive training vectors of dimension \( 1 \times L \) and \( N'M_n \) negative training vectors of the same dimension, where \( M_p, M_n \) are the total number of positive and negative training sequences in the training set respectively. By using this process, Gentleboost will select a set of characteristic temporal slices for the positive class. Therefore, by performing the procedure for each class, we end up with a set of characteristic slices for each class.

5.2.2 Relevance Vector Machine classifier

We use a Relevance Vector Machine classifier (RVM) in order to classify each example in our datasets into an action category. We use the distance of each test example to the selected features of each class in order to define a Gaussian kernel for the RVM. That is, given a dataset of \( N \) input-target pairs \( \{(F_n, l_n), 1 \leq n \leq N\} \), the kernel that we use is defined as:

\[
K(F, F_n) = e^{-\frac{D(F, F_n)^2}{2\sigma}},
\]

(5.4)
where $\eta$ is the width of the kernel and $D$ is the average of the minimum distances between the temporal slices of the test sequence and the informative temporal slices of each class as selected by Gentleboost. Since RVM classifiers are specially suited for binary problems, we follow in this work an one-against-all strategy in order to classify a given example $F$ into one of $L$ action categories. That is, we train $L$ different classifiers, and we classify $F$ to the class for which the conditional distribution $p_i(l|F), 1 \leq i \leq L$ is maximized:

$$\text{Class}(F) = \arg \max_i (p_i(l|F)).$$  \tag{5.5}

### 5.3 Experiments

#### 5.3.1 Datasets

For our experiments we use three different datasets of human activities. The first one is the KTH dataset [11], containing 6 different actions: boxing, hand-clapping, hand-waving, jogging, running, and walking. Each action is performed by 25 subjects several times under different conditions, including scale changes, indoors/outdoors recordings, and varying clothes. The second is the Weizmann dataset, used in [10], and contains 10 different activities, namely walk, run, jump, gallop sideways, bend, one-hand wave, two-hands wave, jump in place, jumping jack and skip, each of which is performed by 9 or 10 subjects. Finally, we used our own created aerobics2 dataset that was presented in chapter 3, and contains 15 different aerobic exercises performed twice by 5 different subjects.

#### 5.3.2 Camera motion

In order to demonstrate the effectiveness of the local median filter in compensating for general camera motion, we simulate the latter in videos from the aerobics dataset. In contrast to the KTH dataset, the aerobics dataset contains sequences with textured, non-planar background. In order to simulate camera motion, we apply a rectangular cropping window around the subjects in the dataset. Subsequently, we apply rapid, random displacements of the cropping window. For comparison, we also apply a global affine model for motion compensation. We use an iterative weighted least squares algorithm for estimating the parameters of the affine model, where the weights are updated at each
iteration using the robust m-estimator of Geman-McClure [158]. In Fig. 5.6 the results of both motion compensation techniques are depicted.

As can be seen from the figure, both methods efficiently filter out the majority of the flow vectors that are due to the camera motion. For the case of the global model, there exist a number of residual flow vectors that do not belong to the occurring activity (frames (a),(d),(e)). While the median filter does not seem to suffer from this problem, it occasionally tends to filter out vectors that belong to the activity. This is evident in frames (b) and (c), and is directly related to the size of the utilized filtering window. In this work, we used a window of $25 \times 25$ pixels.

### 5.3.3 Classification Results

We performed our classification experiments using cross validation, carried out in the leave-one-subject-out manner. That is, in order to classify a test example performed by a specific test subject,
Table 5.2: Recall and precision rates acquired on the three datasets for the three classification experiments.

<table>
<thead>
<tr>
<th>Database</th>
<th>1-NN</th>
<th>Gentle-NN</th>
<th>Gentle-RVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerobics2</td>
<td>0.84/0.85</td>
<td>0.91/0.94</td>
<td>0.95/0.96</td>
</tr>
<tr>
<td>Weizmann</td>
<td>0.88/0.88</td>
<td>0.9/0.93</td>
<td>0.92/0.92</td>
</tr>
<tr>
<td>Weizmann*</td>
<td>0.78/0.82</td>
<td>0.83/0.84</td>
<td>0.84/0.84</td>
</tr>
<tr>
<td>KTH</td>
<td>0.67/0.68</td>
<td>0.78/0.78</td>
<td>0.81/0.82</td>
</tr>
</tbody>
</table>

we created a codebook and trained the respective classifiers using all available data instances except of those belonging to the same class and performed by the same subject as in the test example. In order to assess the impact of each step of our methodology (the feature selection and the RVM classification), we present classification results from three different experiments. In the first experiment, each temporal slice of a test sequence is matched with the closest slice of a training sequence in terms of their $\chi^2$ distance. The overall distance measure between the image sequences is then calculated as the average of the minimum calculated slice distances. The test example is then classified to the class of the training example for which the smallest overall distance has been calculated. In the second experiment the slices of each test example are matched against the selected slices of each class (selected by the Gentleboost in the feature selection step). Once again, this is done in terms of the $\chi^2$ distance. The test sequence is then assigned to the class for which the smallest resulting distance has been calculated. Finally, in the third experiment we present the classification results obtained using RVM as a classifier. More specifically, we use the distance of each test example to the selected slices of each class in order to define the kernel of the RVM, according to Eq. 3.27. For comparison, we present two different results for the Weizmann dataset. In the first, the skip class is included in the database. In the second one, this class is not included, since several researchers present results that do not include this class, which is arguably the most difficult to recognize [10] [52]. The collective classification results for all three datasets and all three experiments, in terms of recall and precision rates are given in Table 5.2, where the reduced Weizmann dataset is denoted by Weizmann*.

As we can see from Table 5.2, there is a considerable increase in classification performance on all three datasets when the feature selection step is introduced, that is, when the most discriminative temporal slices/windows per class are selected for training. This result clearly suggests that slices which are common in a large number of classes have a negative impact on the classification performance. This justifies our choice to conduct feature selection. On the other hand, there is only a slight increase in classification performance on the Weizmann and KTH datasets by additionally using RVM for classification, while the increase for the aerobics dataset is about 4%. We attribute this to the fact
that the most informative elements are already selected by our feature selection scheme. We should note, however, that the contribution of the RVM classification step is always positive, but not very significant except for the aerobics dataset.

The average recall rate for Gentle-RVM approach applied to the KTH dataset is about 81%. From the confusion matrix in Figure 5.7, we can see that confusions are commonly made between similar classes running and jogging. However, as noticed by Schuldt et al [11], these confusions are in fact reasonable, since what appears to some people as running may appear to others as jogging and vice versa. Concerning the Weizmann* dataset (where the skip class is excluded), the average recall rate of Gentle-RVM approach is 92%. From the confusion matrix in Figure 5.8, we can see that there are some confusions between similar classes like jump, run, walk and side, as well as wave1 and wave2. However, as we can see from Figure 5.8, these confusions are rather rare. Finally, we performed similar classification experiments using a global affine model for camera motion compensation. The parameters of the model were estimated as described in section 5.3.2. We achieved a 75% average recall rate for the KTH dataset, using Gentleboost for feature selection and RVM for classification.

As shown in Table 5.3, the results that we obtained for the Gentle-RVM approach on the KTH dataset outperform the ones presented in, e.g., [125] [11]. Furthermore, we achieve similar results as the ones reported in [51] [69]. Contrary to our method, however, these works do not specifically address camera
5.3. Experiments

<table>
<thead>
<tr>
<th></th>
<th>bend</th>
<th>jack</th>
<th>jump</th>
<th>pjump</th>
<th>run</th>
<th>side</th>
<th>walk</th>
<th>wave1</th>
<th>wave2</th>
</tr>
</thead>
<tbody>
<tr>
<td>bend</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.11</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>jack</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>jump</td>
<td>0.0</td>
<td>0.0</td>
<td>0.78</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.11</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>pjump</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>run</td>
<td>0.0</td>
<td>0.0</td>
<td>0.11</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>side</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.78</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>walk</td>
<td>0.0</td>
<td>0.0</td>
<td>0.11</td>
<td>0.0</td>
<td>0.11</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>wave1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.78</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>wave2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.11</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Figure 5.8: Confusion Matrix for the Weizmann* dataset

motion, since they assume a stationary camera. Furthermore, we do not apply any preprocessing step to the raw image sequences prior to feature detection, contrary to Fathi and Mori [27], who use stabilized sequences of cropped frames, centered on the human figure as their input. Similarly, Wong and Cipolla [159], temporally normalize their sequences to have similar length. Instead, we handle temporal variations by automatically detecting temporal scale in the spatiotemporal salient point detection step and by using this scale in order to define the neighborhoods for the B-spline approximation. Finally, we do not perform any background subtraction before detecting our features, as opposed to Jhuang et al. [52] and Ahmad and Lee [23]. The latter, use a Gaussian Mixture Model (GMM) in order to identify foreground pixels as the ones which vary over time.

Concerning the Weizmann dataset, our results are almost 4% lower than those reported in [10] and [52]. However, besides handling camera motion, the main advantage of our method compared to these works is the feature selection step. By contrast in [10] [52] the whole set of the extracted features is used for classification purposes. In addition, our system uses a sparse representation as opposed to [10], where a whole image sequence is represented as a space-time shape. Sparse, local representations, are shown to be significantly better in dealing with clutter and occlusions for object detection and recognition in comparison to global representations (e.g. see [160]). Similar observations can be therefore expected in the case of action recognition problems. As can be seen from the results in
Table 5.2 and Figures 5.7,5.8, this assumption proved to be true. The only other work presented so far in the body of the related literature that uses a sparse and structured representation is that proposed in [77]. However, a recognition rate of 72.8% is reported on the Weizmann dataset which is by far inferior to the 92% achieved by our method.

As previously mentioned, we used cross validation in a leave-one-subject-out manner in order to evaluate our method. This means that for any test example, the codebook contains information about the class of this example. We would like to determine here, if our features are general enough to handle completely unknown classes. That is, given a codebook of visual verbs we want to examine how well can this codebook discriminate classes that did not contribute to its creation. Our motivation for this experiment lies in the fact that our system is able to consistently recover short-term motion in small spatiotemporal regions. Therefore, given that an unknown class can share a number of similar spatiotemporal regions with several known classes, there should be some ability for good discrimination. We performed two different experiments. In the first experiment we created a codebook from 14 classes of the aerobics dataset. The remaining class was used for testing. In other words, the remaining class was represented by using visual verbs defined for other classes. The result was that 8 out of 10 instances of the test class were correctly classified. In the second experiment, we created a codebook from the whole aerobics dataset and tested it for discrimination of classes from the Weizmann dataset. The classes between these two datasets are almost completely different. Exceptions are the classes jack, wave1, and wave2 of the Weizmann dataset which are also present in the aerobics dataset. The average recall rate for this experiment was 67.7%, with the worst performing classes being jump, run, walk, and skip, as we can see from the confusion matrix of Figure 5.9. However, poor results for these classes could be expected, as these classes do not seem to share common frames with classes of the aerobics dataset. Overall, these results indicate that it might be possible to use the proposed descriptors for representing new classes of actions. We intend to investigate this issue in further detail using all action databases and performing the same experiments with features that are currently the state of the art in the field, like those proposed in [10], [52] and [77] (i.e. Poisson, C2, and Gradient features).
5.4 Conclusions

In this chapter we presented a feature-based method for human activity recognition. The features that we extract stem from automatically detected salient points and contain static information concerning the (moving) body parts of the subjects as well as dynamic information concerning the movements/activities. Furthermore, our features are robust to camera motion, through the use of filtered optical flow for their extraction. We used the extracted features to recover similar temporal windows that essentially encode the short-term motion typical for a given activity in a ‘bag of verbs’ approach. Our results showed that our representation is able to recover a wide variety of different motion/activity classes. Furthermore, our experiments showed that our system is able to generalize well and handle unknown classes (i.e. those that did not contribute to the creation of the utilized codebook).

Despite the apparent benefits of the proposed method in representing and recognizing human activities in the presence of camera motion, there are certain limitations that need to be discussed. As mentioned earlier in this chapter, each descriptor vector that is extracted from a fitted polynomial is compared against a codebook that is created during training. The use of a codebook allows the proposed method to partly deal with the presence of dynamic background. That is, since the codebook is created using

![Confusion matrix for the Weizmann dataset](image-url)

Figure 5.9: Confusion matrix for the Weizmann dataset, where the codebook used for representing the examples was created from the aerobics dataset.

![Confusion matrix for the Weizmann dataset](image-url)
Table 5.3: Comparisons of the proposed method to various methods proposed elsewhere for KTH dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Features</th>
<th>Classifier</th>
<th>Weaknesses / benefits</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td>B-splines</td>
<td>Gentleboost+RVM</td>
<td>Camera motion handling, sparse representation (+)</td>
<td>80.8</td>
</tr>
<tr>
<td>Ke et al. [125]</td>
<td>Optical Flow</td>
<td>Boosting</td>
<td>Robust to camera motion (+), but no specific handling(-)</td>
<td>62.97</td>
</tr>
<tr>
<td>Ahmad et al. [23]</td>
<td>Flow + Moments</td>
<td>MDHMM</td>
<td>Background subtraction (-)</td>
<td>88.3</td>
</tr>
<tr>
<td>Dollar et al. [51]</td>
<td>Gabor filters</td>
<td>NN</td>
<td>Stationary camera (-)</td>
<td>81.17</td>
</tr>
<tr>
<td>Wong et al. [159]</td>
<td>DoG + NMF</td>
<td>SVM</td>
<td>Samples preprocessed into similar temporal length (-)</td>
<td>86.7</td>
</tr>
<tr>
<td>Niebles et al. [69]</td>
<td>Gabor filters</td>
<td>pLSA+SVM</td>
<td>Stationary camera (-)</td>
<td>81.5</td>
</tr>
<tr>
<td>Fathi et al. [27]</td>
<td>Optical flow</td>
<td>Adaboost</td>
<td>Stabilization (-)</td>
<td>90.5</td>
</tr>
<tr>
<td>Jhuang et al. [52]</td>
<td>C features</td>
<td>SVM</td>
<td>Background subtraction (-)</td>
<td>91.7</td>
</tr>
</tbody>
</table>

‘clean’ sequences, descriptor vectors that are due to motion in the background will not match well to the codewords in the codebook. This fact, however, does not guarantee that such spurious descriptors are completely suppressed, and therefore there is a good chance that the final representations will be corrupted due to the background motion. Similar is the effect in the case of occlusion, since the absence of certain codewords from the final temporal slice representations might affect the matching that is performed between these slices and the ones selected during training via gentleboost. Both of these issues, that is, the presence of dynamic background and occlusion are more effectively and exclusively addressed in the method presented in chapter 6.
Chapter 6

Spatiotemporal Localization and Recognition

The methods that have been presented so far assume that the examples in the utilized datasets are segmented a priori. That is, they assume that each example depicts a human activity of a single class, performed by a single subject. As such, they focus solely on recognition, that is, they do not provide the means for localizing the activity in space and time. The method presented in this chapter deals with these issues. More specifically, we propose a method that builds on the work of Leibe et al. [9] on object categorization and segmentation. We extend their framework by proposing a voting scheme in the space-time domain that allows both the temporal and spatial localization of activities. Our method uses an implicit representation of the spatiotemporal shape of an activity that relies on the spatiotemporal localization of ensembles of spatiotemporal features. The latter are localized around spatiotemporal salient points that are detected using the method described in [3]. We model the feature ensembles using a modified star graph model that is similar to the one proposed in [76], but compensates for scale changes using the scales of the features within each ensemble. During training, we use boosting in order to create codebooks of characteristic ensembles for each class. Subsequently, we match the selected codewords with the training sequences of the respective class, and store the spatiotemporal positions at which each codeword is activated. This is performed with respect to a set of reference points, (e.g. the center of the torso and the lower bound of the subject) and with respect to the start/end of the action instance. In this way, we create class-specific spatiotemporal models, that encode the spatiotemporal positions at which each codeword is activated in the training set. During testing, each activated codeword ensemble casts probabilistic votes to the location in
Figure 6.1: Overview of the spatiotemporal voting process. Activated codewords cast spatial and temporal votes with respect to the center and spatial lower bound of the subject and the start/end frame of the action instance. Temporal votes for candidate start/end positions are cast jointly. Local maximums in the spatial and temporal voting spaces are extracted using mean shift and provide estimates for the position of a reference point in each frame of the test sequence and the temporal boundaries of an action instance respectively.

The remainder of this chapter is organized as follows. In section 6.1 we present our approach. That is, the creation of our spatiotemporal models for each class and the way they are used in order to perform localization and recognition. Section 6.2 presents our experimental results, and finally, section 6.3 concludes the chapter.
6.1 Spatiotemporal Voting

6.1.1 Probabilistic framework

We propose to use a probabilistic voting framework in order to spatiotemporally localize human activities. This framework is based on class-specific codebooks of feature ensembles, where each feature is a vector of optical flow and spatial gradient descriptors, extracted around automatically detected spatiotemporal salient points. Given a training set, each codebook is created using a feature selection process which is based on boosting, and selects a set of discriminative ensembles for each class. Each class-specific codebook is associated with a class-specific spatiotemporal localization model, which encodes the spatiotemporal locations and scales at which each codeword is activated in the training set. This process is described in section 6.1.4.

Given a codebook-spatiotemporal localization model pair for each class, our goal is to estimate a set of parameters \( \{\theta_s\}, \{\theta_t\}\) that define, respectively, the location in space-time of a human activity depicted in an unknown image sequence. More specifically, we denote with \(\theta_s\), the location of a set of reference points positioned on the subject (e.g. the center of the torso and the lower bound of the subject), that define its location at each frame of the image sequence. Furthermore, we denote with \(\theta_t\), the temporal extend of the activity, that is, the frame at which it starts and the frame at which it ends.

In order to acquire a probability distribution over \(\{\theta_s\}, \{\theta_t\}\), we propose the use of a spatiotemporal voting scheme, which is an extension in time of the implicit shape model proposed by Leibe et al. [9]. In the proposed model, an activated codeword in the test set casts probabilistic votes for possible values of \(\theta_s, \theta_t\), according to information stored during training. We use ensembles of spatiotemporal features as codewords, modeled using the star-graph model of [76]. In the following, and without loss of generality, we drop the subscripts on the \(\theta_s, \theta_t\) parameters, and describe the utilized probabilistic framework for the generalized parameter \(\theta\). The probability of \(\theta\) can be formulated as:

\[
P(\theta) = \sum_{q=1}^{Q} P(\theta|e_q)P(e_q),
\]

where \(\{e_q\}\) is the set of observed ensembles and \(P(e_q)\) is the prior probability of observing \(e_q\). In the absence of prior knowledge, we model this probability as a uniform distribution, i.e. \(P(e_q) = 1/Q\).
where $Q$ is the number of observed ensembles. Let us assume that we have already created a codebook $E = \{e_d\}$ of feature ensembles $e_d$ from a training set. Each observed ensemble $e_q$ is matched against each codeword $e_d$ from the codebook $E$. By marginalizing $P(\theta|e_q)$ on $e_d \in E$ we get:

$$P(\theta|e_q) = \sum_{e_d \in E} P(\theta|e_d, e_q)P(e_d|e_q). \quad (6.2)$$

For now, let us assume that $P(e_d|e_q)$, that is, the likelihood of match between codeword $e_d$ and the observed ensemble $e_q$ is known. The way this likelihood is calculated is described in section 6.1.2. After matching $e_q$ to $e_d$, we consider $P(\theta|e_d, e_q)$ as being independent of $e_q$. $P(\theta|e_d)$ expresses the probabilistic vote on location $\theta$ given that the activated codebook entry is $e_d$. Let us denote with $\{\theta_d\}$ the set of the votes associated with the activated codebook entry $e_d$. These votes express the spatiotemporal positions at which $e_d$ was observed in the training set, relatively to the subject/action reference system. The way the $\theta_d$ values are learned is explained in more detail in section 6.1.4. $P(\theta|e_d)$ can be modeled as:

$$P(\theta|e_d) = w_d \sum_{\theta_d} P(\theta|\theta_d, e_d)P(\theta_d|e_d), \quad (6.3)$$

where $w_d$ is a weight learned during training, which expresses how important the ensemble $e_d$ is, in accurately localizing the action in space and time. The way $w_d$ is calculated is described in section 6.1.3. The first factor of the summation in Eq. 6.3 is independent of $e_d$, since votes are cast using the $\theta_d$ values. Votes are cast according to the following equation:

$$\theta = \theta_q + S_q S_d^{-1} \theta_d, \quad (6.4)$$

where $S_q, S_d$ are diagonal matrices containing the scale of the $e_q, e_d$ ensembles respectively and $\theta_q$ denotes the location of the observed ensemble $e_q$ in absolute coordinates. The concept of eq. 6.4 for the spatial case is depicted in Fig. 6.2(b). That is, the position of the center is given by the vector addition of $\theta_q$, the position of the observed ensemble $e_q$, and the quantity $S_q S_d^{-1} \theta_d$. The latter is the scale-normalized position at which the codeword ensemble $e_d$ (with which $e_q$ is matched) was observed in the training set with respect to the center of the subject. By normalizing with $S_q S_d^{-1}$ we achieve invariance to scale differences between the observed ensemble and the activated ensemble.
6.1. Spatiotemporal Voting

Figure 6.2: Voting example using a simplified case of an ensemble consisting of just two features. (a) During training, the position $\theta_d$ of the activated ensemble is stored with respect to one or more reference points (e.g., the subject center), along with its average spatiotemporal scale $S_d$. (b) During testing, votes are cast using the stored $\theta_d$ values, normalized by $S_qS_d^{-1}$ in order to account for scale changes. (Best viewed in color.)

codeword. $S_d, S_q$ are calculated as the average spatiotemporal scales of the features that constitute the ensembles. Since we only use the stored $\theta_d$ and $S_d$ values for casting our votes, we can model $P(\theta|\theta_d)$ as:

$$P(\theta|\theta_d) = \delta(\theta - \theta_q - S_qS_d^{-1}\theta_d), \quad (6.5)$$

where $\delta(.)$ is the Dirac delta function. Finally, we model $P(\theta_d|e_d)$ using a uniform distribution, that is, $P(\theta_d|e_d) = 1/V$, where $V$ is the number of $\theta_d$ values associated with $e_d$. Alternatively, this probability can be modeled using a density estimation method. That is, a larger probability can be assigned to the $\theta_d$ values that were more commonly observed during training.

The use of class-specific codebook-model pairs enables us to deal with the presence of dynamic background and multiple activities in the test set. The purpose of such models is to search for activities of a specific class in an unknown image sequence. Ideally, observed ensembles localized around activities of different class, or around any other kind of motion in the background will not match well with the codewords in the codebook, and therefore their votes according to the corresponding model will be assigned a very small probability. This is evident from eq. 6.2. Finally, the use of a voting framework for localization increases the robustness of proposed method to partial occlusion. Since votes are cast from each observed ensemble in the test set, a good estimate can be acquired, as long as a good portion
of the activity is still visible.

6.1.2 Feature ensembles

As mentioned in section 6.1.1, in order to increase the spatiotemporal specificity of our method, we use ensembles of spatiotemporal features instead of single features. By taking feature ensembles into account, sets of features that have similar spatiotemporal configurations between the training and test sets are matched. We form ensembles by sampling individual features as seeds and subsequently taking into account their $N - 1$ nearest neighbors. We discard points that have a significant degree of overlap with the seed. In our implementation, two points have a significant degree of overlap if their normalized Euclidean distance with respect to their spatiotemporal scale is smaller than a specific threshold.

Let $e_d = (c_d, \{v_{id}^i, l_{id}^i\}_{i=1...M})$ be an ensemble in the database consisting of $M$ features, where $c_d$ is the spatiotemporal center of the ensemble, and $v_{id}^i, l_{id}^i$ are, respectively, the descriptor vector and the spatiotemporal location of the $i^{th}$ feature.

We model our feature ensembles using a modification of the star graph model of [76]. More specifically, we model the joint probability $P(e_d, e_q)$ between the database ensemble $e_d$ and the query ensemble $e_q$ proportional to:

$$P(e_d, e_q) \propto P(c_d, v_{d1}^1, ..., l_{d1}^1, ..., c_q, v_{q1}^1, ..., l_{q1}^1, ...).$$  \hspace{1cm} (6.6)

The likelihood in Eq. 6.6 can be factored as:

$$P(c_d, v_{d1}^1, ..., l_{d1}^1, ..., c_q, v_{q1}^1, ..., l_{q1}^1, ...) = \alpha \prod_i \max_j (P(l_{jq}^j | l_{id}^i, c_d, c_q) P(v_{jq}^j | v_{id}^i)) P(v_{id}^i | l_{id}^i).$$ \hspace{1cm} (6.7)

The first factor in the maximum in eq. 6.7, that is, $P(l_{jq}^j | l_{id}^i, c_d, c_q)$, expresses the similarity in the topology of the ensembles, and the second factor expresses the similarity in their descriptor values. Consequently, each feature $i$ of the ensemble $e_d$ is matched to the feature $j$ of the ensemble $e_q$ with the maximum similarity in relative location within the ensemble and descriptor value. We model the first factor as follows:
6.1. Spatiotemporal Voting

\[ P(l_q^i|c_q^i, c_d) = z_1^{-1} \exp(-((l_q^i - c_q)S_q^j - (l_d^i - c_d)S_d^j)^T S^{-1}((l_q^i - c_q)S_q^j - (l_d^i - c_d)S_d^j)) \]  

where \( z_1 \) is a normalization factor, and \( S \) is a fixed covariance matrix controlling the allowable deviations in the relative feature locations. Finally, \( S_d^i, S_q^j \) are diagonal matrices containing the inverse spatiotemporal scales of the points located at locations \( l_d^i, l_q^i \) respectively. That is,

\[ S^i = diag((\sigma_i, \sigma_i, \tau_i)^{-1}) \]

where \( \sigma_i, \tau_i \) are the spatial and temporal scales of the \( i^{th} \) feature. By normalizing the distance between the individual features and the ensemble center with the spatiotemporal scales of the features, we achieve invariance to scaling variations.

In order to model the second factor in the maximum in eq. 6.7, that is, \( P(v_q^j|v_d^i) \), we use an exponential distribution:

\[ P(v_q^j|v_d^i) \propto z_2^{-1} \exp\left(-z_3^{-1}D(v_q^j, v_d^i)\right), \]

where \( z_2, z_3 \) are normalization factors, and \( D(., .) \) is the \( \chi^2 \) distance.

The last factor in Eq. 6.7 expresses the relations within the ensemble \( c_d \), i.e. the relation between the feature descriptor and its location. Similar to [76], we model this factor using examples from the database:

\[ P(v_d|l_d) = \begin{cases} 1 & (v_d, l_d) \in DB \\ 0 & otherwise \end{cases} \]

where \( v_d, l_d \) are, respectively, an arbitrary descriptor and location. That is, \( P(v_d|l_d) \) is equal to one if and only if the feature descriptor \( v_d \) appears in location \( l_d \) in the database.
6.1.3 Localization accuracy

In this section we will describe a methodology to learn \( w_d \), that is, the weight that is used in eq. 6.3 and expresses the importance of ensemble \( e_d \) in accurately localizing an activity in space and time. More specifically, we would like to favor votes from ensembles that are informative (i.e. they are characteristic of the location at which they appear within the action instance) and suppress votes from ensembles that are commonly activated (i.e. they are activated at many locations in the action instance in question). Let us denote by \( P_d(l) \) the probability that the ensemble \( e_d \) was activated at location \( l \). This distribution is learned during training. Then, the votes of each ensemble \( e_d \) are weighted as follows:

\[
    w_d = e^{-\int P_d(l) \log P_d(l) dl},
\]

(6.12)

where the exponent is the entropy of \( l \). The exponent in Eq. 6.12 is the Shannon entropy of the distribution of the votes that the ensemble \( e_d \) casts. Ensembles that are informative will have a distribution with low entropy, since their votes will be concentrated in a few values, resulting in a large weight. An example is given in Fig. 6.3(a) for the temporal case, where the depicted ensemble is activated mostly at the middle of the action, and corresponds to the withdrawing and the stopping of the subject’s hand. By contrast, we depict in Fig. 6.3(b) an ensemble that is activated throughout the conduction of the activity, as shown in the histogram of votes at the top row of the figure. The depicted ensemble corresponds to the constant motion of the subject’s hands towards the left. Since it is commonly activated, this ensemble receives a low weight.

6.1.4 Feature Selection and Codebook Creation

We use Gentleboost [128] in order to select characteristic ensembles that will form the codewords for each class-specific codebook \( E \). Our goal is to select feature ensembles that appear with high likelihood in the positive examples and with low likelihood in the negative examples. In order to do so, we randomly sample \( L \) (e.g. 5000) ensembles from the examples in the positive set. Using Eq. 6.6, we match these ensembles to the remaining ones in the positive set and the ones in the negative set. Our expectation is that ensembles characteristic of the positive set will have a low matching cost (i.e. high likelihood of match) to ensembles in the examples belonging to that set, and a higher matching
6.1. Spatiotemporal Voting

Figure 6.3: Ensemble weighting for the temporal case. (a) Ensembles that are activated in a specific phase of the action (as shown in the histogram at the top row) receive a large weight. In the depicted example, the ensemble is localized on the hand, which moves to the right and stops after a few frames. (b) Conversely, ensembles that are commonly activated throughout the action receive a smaller weight. In the depicted example, the activated ensemble corresponds to the constant motion of the hand towards the left. (Best viewed in color.)

cost (i.e. low likelihood of match) to ensembles in the examples belonging to all other classes (i.e. the negative set). Since each image sequence in the training set comprises a few thousands of features, we only keep the $N'$ best matches from each sequence, in order to make the selection tractable. This procedure results in $N'M_p$ positive training vectors of dimension $1 \times \mathcal{L}$ and $N'M_n$ negative training vectors of the same dimension, where $M_p$ and $M_n$ are the total number of the positive and negative image sequences in the training set respectively. Using these training vectors, Gentleboost selects a set of characteristic ensembles for the positive class. This set is a subset of the initially extracted set of $\mathcal{L}$ ensembles. By performing this process for each class we end up with a set of characteristic ensembles for each class. An example of the training vectors that are created using this procedure is depicted in Fig. 6.4. As it can be seen from the figure, several features, namely the first 15, are not characteristic
Figure 6.4: Visualization of a feature selection matrix. Selection is performed from 50 features, and using 40 positive and 60 negative examples. Features that have a high likelihood of match (light areas) to the positive examples and a low likelihood of match (dark areas) to the negative examples are eventually selected.

of the class, since their likelihood of match in both positive and negative examples is low (dark areas in the figure). On the other hand, several features have a high likelihood of occurrence in the positive examples (light areas), while their likelihood in the negative examples remains low. These are features that are eventually selected by Gentleboost. We should note that we do not pose any constraints to the number of training rounds, that is, we do not place any bound to the number of ensembles that can be selected. Since, intuitively, the more similar the classes are to each other, the higher the number of features required to discriminate between them, a higher number of ensembles will be selected for these classes. Indeed, this procedure led to the selection of more than 1000 characteristic ensembles for the jogging, running and walking classes of the KTH dataset, while the number of selected ensembles for the rest of the classes in the same dataset was considerably lower.

We use each class-specific codebook in order to create a spatiotemporal model for each class. Each model is created by accumulating information over the spatiotemporal positions at which each codeword is activated in the training set. That is, for each class-specific codebook, we iterate through the training sequences that belong to the same class as the codebook and activate each ensemble $e_d$ whose likelihood of match to the ensembles belonging to these sequences is above a threshold. Subsequently, we store all the positions $\theta_d$ at which each $e_d$ was activated relative to a set of reference points in
6.1. Spatiotemporal Voting

space and time. In addition, we also store a diagonal matrix $S_d$ containing the spatiotemporal scale at which codeword ensemble $e_d$ was activated. The scale is taken as the average of the scales of the features that constitute $e_d$. During testing, the values $\{\theta_d\}, S_d$ are used in order to cast spatiotemporal votes concerning the spatiotemporal extent of an activity in the test set, given that the codeword $e_d$ is activated, as explained in section 6.1.1.

An example of the spatiotemporal model creation process is depicted in Fig. 6.2(a), for the case of single reference point that corresponds to the center of the subject’s torso. Since we are interested in space-time localization, it makes sense to select the start and end frames as reference points in time. Concerning space, the choice depends on the kind of activity that is to be recognized. For actions involving single subjects, a good choice could be, for instance, points lying on the center and lower bound of the subject (see Fig. 6.5).

6.1.5 Features

The proposed framework can be utilized with any kind of local descriptors. In our implementation, we use a combination of optical flow and spatial gradient descriptors, extracted around automatically detected spatiotemporal salient points. We use the algorithm proposed in [3] in order to extract the set of spatiotemporal salient points, which we denote with $S = \{(c_i, s_i)\}$. Here, $c_i$ is the spatiotemporal position of the point with index $i$. The vector $s_i$ is the spatiotemporal scale at which the point was detected and has a spatial and temporal dimension. This scale is automatically detected in [3], as the scale at which the entropy of the signal within the local spatiotemporal neighborhood defined by it is locally maximized. In order to achieve robustness against camera motion, like translation, small rotations zoom, we detect the salient points on the filtered version of the optical flow field. More specifically, we locally subtract the median of the optical flow within a small spatial window. Alternatively, a global method, like an affine model can be applied in order to compensate for the motion of the camera. We use the algorithm in [156] for computing the optical flow, due to its robustness to motion discontinuities and to outliers to the optical flow equation. In order to form our descriptors, we take into account the optical flow and spatial gradient vectors that fall within the area of support of each salient point. This area is defined by the spatiotemporal scale ($s_i$) at which each salient point is detected. Using their horizontal and vertical components, we convert these vectors into angles and bin them into histograms using a bin size of 10 degrees.
Figure 6.5: Illustration of the spatiotemporal voting scheme. First row: temporal voting space. Votes towards the start/end frames are cast jointly. Second, third row: Start/end frame projections along lines passing from maximum hypothesis in the temporal voting space. Evidence is accumulated as time progresses, resulting in more votes at the most probable positions. Fifth, sixth row: Spatial voting spaces. Based on its location, each ensemble votes for the most probable position of the center and lower bound of the subject. Fourth row: Fitted bounding boxes resulting from the maximum responses in the spatial voting spaces. (Best viewed in color.)
6.1.6 Activity detection

The goal of activity detection is to spatiotemporally localize and classify an activity depicted in an unsegmented image sequence. Using the probabilistic framework of section 6.1.1, the proposed algorithm initially casts spatial votes according to the information stored in the training stage. Examples of spatial voting spaces for the center and lower bound of the subject are depicted in the fifth and sixth rows of Fig. 6.5 respectively. Since the class of the human activity is unknown, this procedure is performed for each class-specific codebook-model pair. We use a Mean Shift Mode estimation algorithm [161] in order to localize the most probable centers and lower bounds of the subjects at each frame of the image sequence. In addition, we apply a Kalman [81] filter using as observations the raw estimates of these points as they are given by the mean shift mode estimation process. Kalman filtering has the effect of smoothing the estimates of the points from frame to frame, and increases robustness against outliers in the mean shift mode estimation. Using the estimates of these two points, we are able to fit a bounding box around the subject, as depicted in the fourth row of Fig. 6.5. In order to reduce the influence of clutter, we cast temporal votes by only taking into account the ensembles that contributed to the most probable center in the spatial voting space. Finally, using Mean Shift Mode estimation on the resulting temporal voting spaces, the most probable hypotheses concerning the temporal extent of the activity are extracted. An example of a temporal voting space is depicted in the top row of Fig. 6.5, where the y-axis indicates the frame at which the instance starts and the x-axis the frame at which it ends. From the figure, it is evident that 7 hypotheses can be extracted, one for each local maximum. Since the votes for the start/end frames are cast jointly, most votes are concentrated above the main diagonal, reflecting the fact that the start frame position must temporally precede the end frame position. In order to illustrate the evolution of the temporal votes as time progresses, we also depict, in the second and third row of the same figure, one dimensional projections of the temporal voting space along horizontal and vertical lines that pass through one of the maximums, as this is given by Mean Shift Mode estimation. As shown in the figure, as time progresses, more evidence is accumulated concerning the most probable position in time where the action instance starts and ends.

Depending on the voting space from which each hypothesis is extracted, a class label can be assigned directly to it. We perform instead a hypothesis verification stage. More specifically, let us denote with $e_{tm}$ the maximum response of the $m$ spatial voting space at frame $t$, as this is given by mean shift mode, where $m$ denotes the class. That is, each $e_{tm}$ expresses the belief of the voting algorithm that the center of the subject is at a specific location at frame $t$ for model $m$. Other points (i.e. the lower
bound of the subject), or a combination of them can also be used for this purpose. Furthermore, let us denote an extracted hypothesis with $F_{ij}$, where $i, j$ are the indexes of the frames at which, according to the hypothesis, the activity starts and ends respectively. Our hypothesis verification step relies on the calculation of the following measure:

$$R_{ijm} = \frac{1}{(t_j - t_i)} \sum_{i = t_i}^{t_j} e_{tm}.$$  \hspace{1cm} (6.13)

That is, each $R_{ijm}$ is the average sum of the mean shift output of the $m$ spatial voting space, between frames $t_i, t_j$. Using $R_{ijm}$, we define a thin plate spline kernel for an RVM classification scheme. That is,

$$K_{ijm} = R_{ijm} \log R_{ijm}.$$  \hspace{1cm} (6.14)

We train $L$ different classifiers, in an one against all fashion. Each classifier outputs a conditional probability of class membership given the hypothesis, $P_m(l|F_{ij}), 1 \leq m \leq L$. Subsequently, each hypothesis $F_{ij}$ is assigned to the class for which this conditional probability is maximized. That is,

$$\text{Class}(F_{ij}) = \arg \max_m (P_m(l|F_{ij})).$$  \hspace{1cm} (6.15)

**6.2 Experimental Results**

We use four different datasets for our experimental evaluation, namely, the KTH dataset [11], the Hollywood Human Actions dataset (HoHA) [12], the robustness dataset of [10] and a set of synthetic sequences that we created. After we give a short description of each dataset in section 6.2.1, we describe, in section 6.2.2 how we create the training set. Subsequently, we present a series of different experiments that were conducted in order to demonstrate the effectiveness of the proposed algorithm. In section 6.2.3, we present classification results on temporally segmented image sequences. These experiments were performed in order to provide comparable results with methods presented elsewhere in the literature, and in which pre-segmented sequences are used. In section 6.2.4 we present spatiotemporal localization results on sequences whose class is known a priori. The objective of this experiment is to determine how accurate the proposed algorithm is in spatiotemporally localizing the instances in
the given sequences. Joint localization and subsequent classification results are presented in section 6.2.5. In section 6.2.6, we present results on synthetic sequences with a significant degree of occlusion, and finally, in section 6.2.7 we present results on synthetic sequences with dynamic background.

### 6.2.1 Datasets

The KTH dataset contains 6 different actions: *boxing*, *hand-clapping*, *hand-waving*, *jogging*, *running*, and *walking*. Each action is performed by 25 subjects several times under different conditions. These include scale changes, indoors/outdoors recordings, and varying clothes. The main challenges in this dataset include small camera motion (mainly camera zoom and translation), noise in the otherwise uniform background, shadows, and large variability in the conduction of the activities by the subjects.

Containing video samples of human actions from 32 movies, the HoHA dataset is one of the most challenging ones in the area of activity recognition. Each sample is labeled according to one or more of 8 action classes: *AnswerPhone*, *GetOutOfCar*, *HandShake*, *HugPerson*, *Kiss*, *SitDown*, *SitUp*, *StandUp*. The main challenge of this dataset is the huge variability of the actions depicted, due to different lighting conditions, different view-points, cluttered and dynamic background and significant camera motion.

We also perform experiments using the robustness dataset of [10]. The sequences in this dataset have different, non-uniform but static backgrounds, and include walking activities under varying conditions. These include different viewpoints and 11 ‘deformation’ sequences, like walking with a dog, walking with the knees up, walking while a significant portion of the feet is occluded, etc. We use this dataset only for testing, while training is performed using the walking actions of the KTH dataset.

Finally, we created various synthetic sequences in order to test the performance of the proposed algorithm in conditions like partial occlusions, dynamic background and presence of multiple activities in the same scene. In order to test the performance of the proposed algorithm in the presence of occlusion, we selected one sequence per class and placed an artificial occluding bar of varying width in areas of the action that are important for the recognition of that action, like, for instance, on the moving legs of subjects, in classes like *walking*. We address the issues of dynamic background and multiple activities in the scene under the same setting, that is, by collating frames of sequences depicting activities of different classes, as shown in Fig. 6.14(a).
6.2.2 Training set

In this work, we consider a single repetition of an activity as an action instance, like a single hand-clap, a single-hand wave, etc. Our goal is to localize and classify these instances in a continuous video stream. To create a training set, we manually select a subset of action instances for each class and we register them in space and time. More specifically, we spatially resize the frames in the selected instances so that the subjects in them have the same size. Moreover, we linearly stretch the selected instances so that the depicted actions in each class have the same duration. Finally, we manually localize and store the subject centers and lower bounds in the registered training set, where each center is defined as the middle of the torso.

6.2.3 Classification

We use activity instances pre-segmented in time in order to evaluate the classification accuracy of our algorithm and compare it to the state of the art. We use the process of section 6.1.6 in order to perform classification, where each hypothesis corresponds to a pre-segmented example. That is, we calculate, for each example, its similarity to each of the trained models according to eq. 6.13 and use this similarity in order to define a kernel for the RVM, according to eq. 6.14. Classification is subsequently performed using eq. 6.15. In Fig. 6.6, the confusion matrix for the KTH dataset is depicted. As it can be seen from the figure, the largest degree of confusion is between the classes \textit{jogging} and \textit{running}, while the confusion between the other classes is low. As noticed by Schuldt et al [11], these confusions are in fact reasonable, since what appears to some people as running may appear to others as jogging and vice versa. The average recall rate achieved by the RVM classifier for the KTH dataset is 88%. By contrast, using just the measure of eq. 6.13 and a 1-NN classifier, the average recall rate was about 75.2%. The largest improvement when applying the RVM classifier was noted on the \textit{running} class, with an increase from 53% to 85% in the recall rate.

In Fig. 6.7, we present the confusion matrix for the HoHa dataset. Due to the small number of representative examples, we discard the classes \textit{GetOutOfCar}, \textit{HandShake}, \textit{SitUp} and present results using the five remaining classes of the dataset. It can be observed that there are several confusions between classes that are not very similar. It is interesting to note, however, that the largest confusion is between the classes \textit{HugPerson} and \textit{Kiss}, since both involve two persons coming progressively closer to each other.
6.2. Experimental Results

Figure 6.6: Confusion Matrix for the KTH dataset.

Figure 6.7: Confusion Matrix for the HoHa dataset.

We use a cross-dataset approach in order to acquire classification results on the robustness dataset of [10]. More specifically, we consider the latter only for testing, using the models that we created on the KTH dataset. Using this approach, our algorithm was able to correctly classify 9 out of the 11 sequences of the deformed set and 6 out of the 10 sequences of the multi-view set, with all confusions being between the *walking* and *jogging* classes. While Blank et al. [10] report 100% recognition rate on this dataset, their training is based on the Weizmann dataset of human actions [10], which does not include the *jogging* class. By removing the *jogging* class from our classification process, our classification rate on this dataset also reaches 100%.

Finally, we present, in Table 6.1, comparative classification results between the proposed method and
Table 6.1: Comparisons of the proposed method to various methods proposed elsewhere for the KTH dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td>88.0</td>
</tr>
<tr>
<td>Ke et al. [125]</td>
<td>62.97</td>
</tr>
<tr>
<td>Schuldt et al. [11]</td>
<td>71.83</td>
</tr>
<tr>
<td>Ahmad and Lee [23]</td>
<td>88.3</td>
</tr>
<tr>
<td>Dollar et al. [51]</td>
<td>81.17</td>
</tr>
<tr>
<td>Wong and Cipolla [159]</td>
<td>86.7</td>
</tr>
<tr>
<td>Niebles et al. [69]</td>
<td>81.5</td>
</tr>
<tr>
<td>Fathi and Mori [27]</td>
<td>90.5</td>
</tr>
<tr>
<td>Jhuang et al. [52]</td>
<td>91.7</td>
</tr>
<tr>
<td>Rapantzikos et al. [50]</td>
<td>88.3</td>
</tr>
<tr>
<td>Ali and Shah [162]</td>
<td>87.7</td>
</tr>
</tbody>
</table>

several methods proposed in the literature. Since these methods use pre-segmented sequences, we compare their results with the ones achieved by the algorithm, presented in section 6.1.6. As can be seen from Table 6.1, the classification results that we obtained for the KTH dataset outperform the ones presented in, e.g., [125] [11]. Furthermore, we achieve similar results as the ones reported in [50] [23]. However, the main advantage of the proposed method compared to these works is that we also provide the means for localization of the actions in space and time. Furthermore, we do not assume a stationary camera as these works do. Instead, by using filtered optical flow we minimize the effect of camera motion in the extracted features. We also do not apply any preprocessing step to input image sequences prior to feature detection, contrary to Fathi and Mori [27], who use stabilized sequences of cropped frames centered on the human figure. Similarly, Wong and Cipolla [159] temporally normalize their sequences to have similar length. Instead, we handle temporal variations by automatically detecting temporal scale in the spatiotemporal salient point detection step and by using this scale throughout our proposed algorithm. Finally, we do not perform any background subtraction before detecting our features, as opposed to Jhuang et al. [52] and Ahmad and Lee [23]. The latter, use a Gaussian Mixture Model (GMM) in order to identify foreground pixels as the ones which vary over time. In the proposed method, however, we achieve a similar effect by detecting the spatiotemporal salient points at areas in which there is significant amount of motion, as described in [3].
6.2.4 Localization

6.2.4.1 Spatial Localization

In this section we evaluate the accuracy of the proposed algorithm in localizing a subject at each frame of an image sequence. Here, we assume, that the activity class that the subject is performing is given. Following the process of section 6.1.6, the proposed algorithm is able to provide an estimate of the subject center and lower bound for each frame of a sequence. In order to account for the smooth motion of the subjects, and to deal with possible outliers, we apply a Kalman filter to the estimates of the subject location. The results achieved for each class of the KTH dataset are depicted in Fig. 6.8. As can be seen from the figure, using just the raw estimates, our algorithm is able to localize the center of the subject in 70% of all frames in the dataset on average, with the estimate’s distance from the ground truth annotation being smaller or equal to 15 pixels. Given that the width of the subjects in the KTH dataset is on average about 20 pixels, our estimate, in most cases, falls within its range. The worst performing class in these experiments is "running", which, for the same distance from the ground truth yields around 55% accuracy in the localization of the subject center. By applying a Kalman filter on the raw estimates, we achieve an increase in performance of about 10% for the "boxing", "handclapping" and "handwaving" classes, while there was a smaller increase in the performance of the "jogging", "running" and "walking" classes.

6.2.4.2 Temporal localization

In this section we evaluate the accuracy of the proposed algorithm in localizing in time several instances of a known activity that occur in an image sequence. That is, the localization of all instances of a specific class that occur in an image sequence. For this experiment, we apply the process of section 6.1.6, and compare each extracted hypothesis with the ground truth annotation. The latter was performed in such a way so that each annotated instance includes a single repetition of the activity, i.e. a single punch in "boxing", a single hand clap, etc. Each extracted hypothesis specifies the frames in the image sequence at which the action instance starts and ends. The error of each hypothesis was calculated as the difference in frames between the ground truth annotation and the start/end frames specified by the hypothesis. In this way, we were able to construct Fig. 6.9, which plots the percentage of the recovered hypotheses as a function of this frame difference.
We compare these results with the ones acquired by the algorithm of Shechtman and Irani [1]. By implementing their algorithm, we compute self-similarity descriptors for all sequences in the KTH dataset and apply their progressive elimination algorithm in order to match a query to each sequence. Matching was performed using 5 query sequences per class from our training set and averaging the acquired results. This gives us an estimate of the spatiotemporal extend of each recovered instance. This is very similar to the hypothesis extraction process of our method, and is the main reason why we chose to perform comparison with the method of [1]. The localization accuracy achieved is depicted in Fig. 6.9. As can be seen from the figure, the temporal localization accuracy of the proposed method is similar to the one achieved by the algorithm of [1] for the boxing class and slightly better for the jogging and running classes. For the handwaving and handclapping classes, 70% of the hypotheses extracted by the proposed algorithm are localized within 3 frames from the ground truth on average, in comparison to 15% achieved by [1].
6.2. Experimental Results

6.2.5 Joint Localization and Recognition

In this section, we present experimental evaluation for simultaneously localizing and classifying human activities that may occur in an unsegmented image sequence. In contrast to previous experiments, both the localization and the class of the activities that occur in the sequence are unknown. Given an unknown image sequence, each class-specific model that was created during training, results in a different voting space for this sequence. Using a mean shift mode estimation algorithm, as described in section 6.1.6, a set of hypotheses is extracted from each voting space, and classified to a specific action category. Each hypothesis corresponds to an interval in time in which the activity takes place, and is assigned a weight, equal to the response in the voting space at the point at which the hypothesis was extracted. A low weight on a hypothesis means that the proposed algorithm does not have a strong belief on its validity. Therefore, by setting up a threshold on the weights, we can control which of the hypotheses are considered as being valid by the algorithm. By varying this threshold, we construct the ROC curves depicted in Fig. 6.10, for each class of the KTH dataset. Note that all curves are well above the main diagonal, meaning that regardless of the threshold value, the number of true positives is always larger than the number of false positives.
6.2.6 Occlusions

We use synthetic image sequences in order to demonstrate the robustness of our method against occlusion. More specifically, we used vertical or horizontal bars in order to occlude parts of human activities. Examples are depicted in Fig. 6.11, for the classes boxing, handclapping, handwaving and walking. We performed our experiments using 10 sequences from each class, i.e. 10% of the data, with a variable bar width. In order to be able to determine the effect of the occlusion in classification accuracy, we selected sequences that were correctly classified in the classification stage of section 6.2.3. Despite the occlusion, our algorithm was able to correctly classify all of the selected sequences. In addition, we present, in Fig. 6.12, average spatial localization results for all of the selected examples as a function of the degree of occlusion. The latter is defined as the ratio between the activity extend in space and the width of the occluding bar. Note that for actions like handclapping, the spatial activity extend only covers the moving hands of the subject. As can be seen from Fig. 6.12, our method is robust to relatively small amounts of occlusion. For 60% of occlusion, that is, the largest degree tested, there was a 20% drop in the localization accuracy of the subject center compared to no occlusion at all, with the estimate of the center being within a radius of 10 pixels from the ground truth annotation. However, the proposed method behaves very well for smaller amounts of occlusion, with an average drop of about 10% in performance for the subject center for a 35% degree of occlusion.
Finally, we performed experiments, using one sequence per class, in which the synthetic bar completely occludes the limbs of the subjects during the apex (e.g. in handwaving) or throughout the conduction of the activity (e.g. in walking). An example is shown in Fig. 6.13, along with the localization accuracy achieved, compared with no occlusion at all. As can be seen from the figure, there is only a small drop in localization performance. We conclude, therefore, that the proposed method is able to sufficiently localize a subject, as long as a good portion of the activity is not affected by the occlusion.

6.2.7 Dynamic background

We use synthetic sequences in order to demonstrate the robustness of the proposed algorithm against dynamic background. Our goal is to demonstrate that the proposed algorithm is not distracted by movement that is due to a varying background or irrelevant activities in the scene. In order to simulate such conditions, we create synthetic sequences in which more than one activities are depicted in the same frame. An example is shown is Fig. 6.14(a), where a boxing and a handwaving activity have been merged in the same scene. Our goal is to be able to localize each of these activities regardless of
the presence of the other. A depiction of the spatial voting spaces derived by the application of the *boxing* and *handwaving* models for one instance of the activity is given in Fig. 6.14. As can be seen from the figure, each class-specific model manages to suppress the information coming from activities other than its class. For instance, the votes attained by the *boxing* model are concentrated around the subject that performs this activity. The reason for this is that ensembles that are localized around the *handwaving* subject do not match well or at all the codewords in the *boxing* codebook. In Fig. 6.15 we present the effect of this experiment to the achieved spatial localization, after applying a Kalman filter on the outcomes of the mean shift mode estimator. For comparison, we also plot the same estimates for the clean sequences. As can be seen from the figure, due to false codeword matches, the localization accuracy of the center of the subject drops about 10%, while for the subject’s lower bound the effect is more severe.

Finally, in Fig. 6.16 we depict the temporal voting spaces that were created using the *boxing* and *handwaving* models. As can be seen, there are 6 prominent peaks in the *boxing* temporal voting space, and 2 peaks in the *handwaving* temporal voting space. These correspond to the actual number of instances of these activities depicted in the image sequence under consideration. Using the Mean Shift mode estimation algorithm, we are able to extract the corresponding hypotheses. Following the process described in section 6.1.6, the spatiotemporal volumes that correspond to those hypotheses are classified in an RVM based classification scheme.
6.3 Conclusions

In this work we have presented a framework for the localization and classification of human actions. The voting nature of the proposed method allows us to perform spatiotemporal localization and classification in image sequences that have not been pre-segmented. The proposed method utilizes class-specific codebooks of characteristic ensembles and class-specific spatiotemporal models that encode the spatiotemporal positions at which the codewords in the codebook are activated during training. The codebook-model pairs are utilized during testing, in order to accumulate evidence for the spatiotemporal localization of the activity in a probabilistic spatiotemporal voting scheme. We have presented results on publicly available datasets and have demonstrated the robustness of the proposed method in the presence of occlusion and dynamic background. Furthermore, we have shown the ability of the proposed method in localizing and classifying multiple activities that take place in the same scene. Finally, we have demonstrated the effectiveness of the proposed method by presenting comparative classification and localization results with the state of the art.
Figure 6.14: (a) Instance of synthetic sequence depicting two activities. (b),(d) Voting spaces for center and lower bound derived using the model for boxing. (c),(e) Voting spaces for center and lower bound derived using the model for handwaving. Notice that each model favors votes belonging to the activity it was trained for.
6.3. Conclusions

Figure 6.15: Average spatial localization accuracy results achieved for the sequences depicting multiple activities. For comparison, the accuracy achieved on the clean sequences is also depicted.

Figure 6.16: Temporal voting spaces corresponding to the image sequence of Fig. 6.14, for (a) boxing and (b) handwaving. Using Mean Shift, 6 instances of boxing are extracted from (a) and 2 instances of handwaving in (b). (Best viewed in color.)
Chapter 7

Discussion and future research

This chapter concludes this thesis. In section 7.1 we summarize our contributions in the field of vision-based human activity analysis, and discuss the limitations of the proposed methods. This evaluation serves as a starting point for further discussion concerning our future research in the field, which is presented in section 7.2.

7.1 Summary of our contributions

The methodology underlying all our contributions concerns the spatiotemporal salient points that were proposed in chapter 3. In this work, we have proposed to measure the variations in the information content of pixel neighborhoods both in space and time, in order to detect our salient points. Each point was then detected at the locations and scales for which this information content was locally maximized. We have shown that the proposed method is consistent concerning the spatiotemporal localization and scale of the detected salient points (see Fig. 3.3), a property that is highly desired in most keypoint-based representations. The scale of each salient point was automatically determined by the proposed algorithm. For its definition, cylindrical neighborhoods were used. The use of rotated cylindrical neighborhoods with respect to the spatial axes would also provide information concerning the orientation of the detected features. However, such an approach would significantly increase the scale search space, and would add to the complexity of the proposed detection algorithm. Overall, the proposed salient point detection process led to representations where the detected points were localized at areas of significant amount of motion. This property made the proposed method suitable for
representing activities where motion is unidirectional, like, e.g., gait activities. By contrast, methods like, e.g., the one presented in [61], detect keypoints at areas where motion changes direction, and therefore are not entirely suitable for recognition of such unidirectional-motion-based actions, since the detected keypoints are localized on local activity endpoints and can therefore be very sparse. Furthermore, one of the advantages of the proposed algorithm is its flexibility with respect to the domain in which the salient points are detected. For instance, the spatiotemporal salient point representations of chapter 5 were acquired using motion compensated optical flow fields instead of the raw pixels used in the salient point representations of chapter 3. We proposed further an iterative space-time warping method. It utilizes a gradient descent algorithm that minimizes a distance between two representations by adjusting a set of parameters. These correspond to two parameters for scaling in space and time and a temporal translation parameter that models delays in the onset of an activity. The proposed model assumes a linear mapping, that is, it is assumed that the conduction speed of an action and the size of the subject is constant. Although this assumption holds for the datasets used for evaluation, it does not hold in all cases and variations in subject size and speed of the actions should be expected in real-world scenarios. Furthermore, an important limitation emanates from the use of gradient descent for optimization, which is prone to getting stuck to local minima. A more suitable warping method would be one which would be able to account for such non-linear variations, like for instance, Dynamic Time Warping (DTW).

In order to enhance the salient point representations of chapter 3, we have proposed, in chapter 4 the use of tracking. More specifically, we have proposed the use of an auxiliary particle filter in order to track the detected salient points for a short number of frames, arriving to sets of short trajectory representations. The main contribution of this chapter was the use of an augmented observation model, which, in the case of salient point tracking, favored solutions that contained a considerable amount of foreground pixels. This was achieved by using a fully automatic background estimation algorithm, based on Gaussian Mixture Models (GMM). According to this algorithm, pixels that did not change significantly over time were labeled as background. The main motivation for enhancing the tracker’s observation model in this way was the imperfect localization of the detected salient points. The latter was due to the use of temporal derivative filters for the salient point detection, which caused the latter to be localized on the motion boundary rather than on the moving parts of the subject. However, since we only used the first few frames in order to acquire a background estimate for each sequence, this approach could potentially lead to errors in the estimation, especially in cases where the amount of motion in the beginning of the action is minimal. We have also conducted experiments where the
7.1. Summary of our contributions

tracked templates were localized on skin areas of the subject, like the hands and the head. For the purpose of these experiments, we further augmented the observation model of the tracker using a pre-learned skin color model. Similar to the background case, the use of such a model caused the tracker to additionally favor solutions that contained a large number of skin pixels. Despite these improvements, however, there was a large number of erroneous trajectories in the final representations, which led to a deterioration in the overall recognition performance. The reason for this stems from the sensitivity of template-based trackers in changes in the appearance or deformations of the tracked templates. As it has been mentioned earlier, periodically updating the tracked template could be a good solution in order to account for such deformations. However, this solution is risky due to error accumulation, and may lead to even worse results. The presence of self occlusion, that is, when a part of the body occludes another, may also lead to tracking errors. In order to handle such cases, the use of a first or second order motion model, that is, a model that assumes constant speed or constant acceleration respectively, could be beneficial. However, such a model could lead to further errors, especially in cases where motion changes direction. Since the latter frequently occurs during the conduction of a human activity, the use of a zero-order propagation model in the proposed algorithm is justified.

Contrary to chapter 4, where each salient point was tracked independently of its neighbors, the main assumption that was made in chapter 5 was that salient points that fall within local spatiotemporal neighborhoods follow a similar motion. This is a natural assumption, emanating from the anatomy of the human body. Indeed, since the detected salient points are approximately localized on moving body parts, they are bound to approximately follow a similar motion. In cases where the direction of motion changes, this assumption might not hold. However, as long as the local neighborhood that is being considered is not very large, and the motion of the subject is smooth, the impact of this violation can be kept to a minimum. A major contribution was the detection of the salient points using motion compensated versions of the optical flow field of the image sequences. Motion compensation was performed using local median filters, however global methods could also be used. By doing so, the detected salient points corresponded to independent motion in the scene, that is, motion that was due to ongoing activities. We proposed to use polynomial surfaces in order to describe the motion and the spatial arrangement of the detected salient points falling within locally defined spatiotemporal neighborhoods. We proposed further to extract a novel set of descriptors that were based on geometrical properties of the fitted polynomials, in order to describe the spatiotemporal shape of each fitted surface. As such, the extracted descriptors were translation invariant. Moreover, we proposed to couple the dimensions of the local neighborhoods with the scales of the detected salient
points, making the extracted descriptors also invariant to scale variations in space-time. As has been mentioned in section 5.1.3.1, in order to fit the proposed surfaces, we used ordered salient points, localized on the motion boundary. By doing so, we avoided the need for background subtraction for defining the latter. However, the ordering and boundary estimation algorithms that were proposed are potentially error prone, and more efficient methods could be used instead. Finally, the proposed method is potentially sensitive to occlusions and the presence of dynamic background. The use of a codebook partly addresses this problem, in the sense that surfaces fitted on background features will not match well the codewords in the codebook. However, occlusion and dynamic background issues were not exclusively addressed in the method.

In order to address the joint problem of localization and recognition, as well as to deal with issues like occlusion and dynamic background, we have proposed, in chapter 6, a framework that implicitly modeled the spatiotemporal shape of an activity. Each class-specific model was learned during a training step, where the spatiotemporal locations of activated codewords of feature ensembles were stored with respect to a set of spatial reference points and with respect to the temporal bounds of the activity. Subsequently, activated codewords in the test phase provided estimates concerning the location of the subject per frame as well as the frames at which the activity started and ended, depending on the information that was stored during training. Furthermore, we proposed a novel weighting scheme, where votes from ensembles that are informative (i.e. they are characteristic of the phase of the action) were favored, while votes from ensembles that are commonly activated (i.e. they are activated in many phases of the action) were suppressed. We have proposed the use of class-specific codebooks and class specific spatiotemporal models throughout the proposed framework. Due to this choice, an unknown image sequence needs to be evaluated against all learned classes during testing, which can be a time consuming process. Finally, we have proposed the use of feature ensembles within the proposed voting framework. The use of ensembles enabled us to perform matching using constellations of features, increasing the spatiotemporal specificity of the proposed method. The number of samples within each ensemble was a fixed variable in the proposed algorithm. However, an evaluation concerning the optimal value of this parameter would be beneficial both in terms of overall detection rate, and in terms of research value.
7.2 Future Work

In order to define suitable avenues for future research, one needs to consider what are the main challenges and current trends in the field of human activity recognition. As it has been mentioned in the introduction section, the impressive increase in the amount of visual information that has become available, makes the development of approaches that deal with real videos imperative. Furthermore, the almost perfect recognition rates that have been achieved on traditional human activity datasets, like the Weizmann and KTH datasets, indicate that human activity analysis algorithms have reached a sufficient level of maturity, and indicate the need for the introduction of more challenging datasets. The recent introduction of datasets like the HoHA and the YouTube action datasets verifies this trend. It is apparent, therefore, that algorithms that are able to analyze and provide satisfactory results on such challenging datasets are required. This translates to algorithms that will be able to deal with significant amount of clutter, dramatic changes in viewpoint and illumination, and significant amounts of occlusion.

While some of the aforementioned issues were addressed in chapter 6 of this thesis, there is still plenty of room for improvement in that direction. For instance, the use of context proved to be very helpful towards recognition of complex classes, like e.g. the action of driving a car. In the latter case, the identification of a car in the scene offers a very good indication for the occurrence of such an action. There are several ways in order to obtain this information, ranging from audio and text scripts (in the case of movies) to pure vision methods, like for instance using specialized classifiers. The latter are able to detect objects or particular conditions in the scene (e.g. outdoors/indoors). The extracted information can then serve as prior knowledge, assisting the task of recognition.

As has been already mentioned, the use of class-specific codebooks of chapter 6 adds a significant overhead in the recognition process, since each example needs to be tested against all classes that have been learned during training. In addition, the use of a separate codebook for each class means that the proposed algorithm does not account for features that may be common between classes. The latter is commonly termed as feature sharing, and can lead to more compact representations and potentially faster recognition, and is therefore one of the directions that are going to be pursued in our future work. A good way in order to achieve this is feature selection, e.g. via boosting. Apart from finding common characteristics between activities, feature sharing may allow for the detection of activity parts, with apparent benefits concerning occlusions or recognition of classes that are unknown,
but share common features with the ones that are learned during training.

Finding correspondences between detected features is an important issue, since through these correspondences a lot of information over the dynamics of an action can be acquired. This issue has been partly addressed in this thesis, and more specifically in chapter 5, where polynomial surfaces were used in order to establish such correspondences in space and time. The concept of chapter 4 was similar, however the goal was to monitor the evolution of each salient point in time, rather than to establish correspondences between individual salient points. In the former case, the established correspondences were based on the assumption that neighboring salient points follow a similar motion, which might not hold in certain cases, like when motion changes direction. In the latter case, the tracker’s performance was not sufficient. An interesting direction for future research, therefore, is to find more efficient ways in order to establish such correspondences. For instance, by further improving tracking or by finding similarities between the detected points, e.g. by taking into account the motion and shape information that they engulf.

Finally, concerning the issue of representation, we have already highlighted the consistency of the utilized salient points in terms of localization and scale. As mentioned earlier, a potential improvement for the proposed representation would be the adoption of more sophisticated spatiotemporal neighborhoods, which would also provide information over a feature’s orientation in space-time, and therefore more effectively capturing the dynamics of the action. The apparent increase in the search space, however, is an important issue that would need to be addressed in that direction. The extraction of suitable descriptors around the detected features is also an important issue that needs to be investigated further. Apart from the optical flow and gradient vectors that were used in chapter 6, a variety of additional descriptors are available, like for instance, wavelet responses, edge-based shape contexts, etc. Finally, the combination of static and dynamic features has been proven very useful for activity recognition not only from videos, but from still images as well. The inclusion of such descriptors in the proposed methods, and particularly that of chapter 6 is straightforward.
Appendix A

Weighting Function Normalization

Factors

Let us denote by $X$ a neighborhood containing $t+c$ number of pixels, by $X_t$ a neighborhood containing $t$ number of pixels and by $X_c$ a neighborhood containing $c$ number of pixels. By definition, and as shown in Fig. A.1, $X$ and $X_t$ correspond to different scales of the same, free form sampling window and $X$ is the union of $X_t$ and $X_c$.

According to [163], the weighting function $W_D$ is defined by:

$$ W_D = \sum_{q \in D} |p_{q,X_c} - p_{q,X_t}|, \quad (A.1) $$

that is, the sum of absolute differences of the pixel probability density functions that correspond to the regions $X_t$ and $X_c$. $p_{q,X_c}$, $p_{q,X_t}$ are the pixel probability density functions of the pixels with value $q$ in the regions $X_c$, $X_t$ respectively. For practical applications, it is more convenient to express $W_D$ in terms of $X$ and $X_t$ instead of $X_c$ and $X_t$, since $X$ and $X_t$ correspond to two different scales of the same sampling window and it is them, which are directly derived during the sampling. In what follows, we will express the weighting function $W_D$ in terms of $X$ and $X_t$ by using a cylindrical sampling window.

Let us denote by $X_{s,d}$ a cylindrical region of radius $s$ and depth $d$, containing $n$ number of pixels, by $X_{s-\Delta s,d}$ a cylindrical region of radius $s - \Delta s$ and depth $d$ containing $t_1$ number of pixels and by $X_{s,d-\Delta d}$ a cylindrical region of radius $s$ and depth $d - \Delta d$ containing $t_2$ number of pixels. Furthermore, let us denote by $X_{\Delta s,d}$ the region between $X_{s,d}$ and $X_{s-\Delta s,d}$ containing $c_1$ number of pixels and by
Appendix A. Weighting Function Normalization Factors

Figure A.1: Regions $X, X_t$ and $X_c$. $X_t$ and $X$ are created by the same, free-form sampling window and and $X$ is the union of $X_t$ and $X_c$.

$X_{s,\Delta d}$ the region between $X_{s,d}$ and $X_{s,d-\Delta d}$ containing $c_2$ number of pixels. Finally, let us denote by $p_{q,s,d}, p_{q,s,\Delta s,d}, p_{q,s,d-\Delta d}, p_{q,\Delta s,d}, p_{q,s,\Delta d}$ the corresponding probabilities of a pixel in these regions taking value $q \in D$.

The weighting function of eq. A.1 can be rewritten as follows for the case of a cylindrical neighborhood:

$$W_D = \sum_{q \in D} |p_{q,\Delta s,d} - p_{q,s-\Delta s,d}| \pm \sum_{q \in D} |p_{q,s,\Delta d} - p_{q,s,d-\Delta d}|, \quad (A.2)$$

Our goal is to express the above equation in terms of $p_{q,s,d}, p_{q,s-\Delta s,d}, p_{q,s,d-\Delta d}$. We have:

$$\sum_{q \in D} |p_{q,s,d} - p_{q,s-\Delta s,d}| = |p_{q_1,s,d} - p_{q_1,s-\Delta s,d}| + \ldots + |p_{q_r,s,d} - p_{q_r,s-\Delta s,d}|$$

$$= \left| \frac{N_{q_1,s,d}}{n} - \frac{N_{q_1,s-\Delta s,d}}{t_1} \right| + \ldots + \left| \frac{N_{q_r,s,d}}{n} - \frac{N_{q_r,s-\Delta s,d}}{t_1} \right|$$

$$= \left| \frac{N_{q_1,s-\Delta s,d} + N_{q_1,\Delta s,d}}{t_1 + c_1} - \frac{N_{q_1,s-\Delta s,d}}{t_1} \right| + \ldots + \left| \frac{N_{q_r,s-\Delta s,d} + N_{q_r,\Delta s,d}}{t_1 + c_1} - \frac{N_{q_r,s-\Delta s,d}}{t_1} \right|$$

$$= \left| \frac{t_1 N_{q_1,s-\Delta s,d} - c_1 N_{q_1,s-\Delta s,d}}{t_1(t_1 + c_1)} \right| + \ldots + \left| \frac{t_1 N_{q_r,s-\Delta s,d} - c_1 N_{q_r,s-\Delta s,d}}{t_1(t_1 + c_1)} \right| \quad (A.3)$$
where \( N_{q,s,d} \) is the number of pixels in region \( X_{s,d} \) with value \( q_r \). Let us define \( p_{q,s-\Delta s,d} = N_{q,s-\Delta s,d}/t_1 \) and \( p_{q,s,d} = N_{q,s,d}/c_1 \). Then, we have:

\[
\sum_{q \in D} |p_{q,s,d} - p_{q,s-\Delta s,d}| = \sum_{q \in D} \left| \frac{c_1 t_1 p_{q,\Delta s,d} - c_1 t_1 p_{q,s-\Delta s,d}}{t_1 (t_1 + c_1)} \right| + \sum_{q \in D} \left| \frac{c_1 t_1 p_{q,s,d} - c_1 t_1 p_{q,s-\Delta s,d}}{t_1 (t_1 + c_1)} \right| \]

Therefore:

\[
\sum_{q \in D} |p_{q,s,d} - p_{q,s-\Delta s,d}| = \frac{t_1 + c_1}{c_1} \sum_{q \in D} |p_{q,s,d} - p_{q,s-\Delta s,d}| \quad \text{(A.5)}
\]

The area contained within a cylindrical neighborhood of radius \( s \) and depth \( d \) is given by:

\[
B(s, d) = \pi s^2 d \quad \text{(A.6)}
\]

Therefore, we may write:

\[
t_1 + c_1 = B(s, d) = \pi s^2 d \quad \text{(A.7)}
\]

\[
c_1 = B(s, d) - B(s - \Delta s, d) = \pi s^2 d - \pi (s - \Delta s)^2 d = \pi d (2s \Delta s - \Delta s^2) \quad \text{(A.8)}
\]

In the same way, for the second summation of eq. A.2 we get:

\[
\sum_{q \in D} |p_{q,s,\Delta d} - p_{q,s,d-\Delta d}| = \frac{t_2 + c_2}{c_2} \sum_{q \in D} |p_{q,s,d} - p_{q,s,d-\Delta d}| \quad \text{(A.9)}
\]

where \( t_2, c_2 \) are the number of pixels in regions \( X_{s,d-\Delta d} \) and \( X_{s,\Delta d} \) respectively.
For this case, we may write:

\[ t_2 + c_2 = B(s,d) \]
\[ = \pi s^2 d \]  
(A.10)

\[ c_2 = B(s,d) - B(s,d - \Delta d) \]
\[ = \pi s^2 d - \pi s^2(d - \Delta d) \]
\[ = \pi s^2 \Delta d \]  
(A.11)

Therefore, from eq. A.2 we get:

\[ W_D = \frac{t_1 + c_1}{c_1} \sum_{q \in D} |p_{q,s,d} - p_{q,s,d-\Delta s,d}| + \frac{t_2 + c_2}{c_2} \sum_{q \in D} |p_{q,s,d} - p_{q,s,d-\Delta d}| \]
\[ = \frac{\pi s^2 d}{\pi d(2s \Delta s - \Delta s^2)} \sum_{q \in D} |p_{q,s,d} - p_{q,s,d-\Delta s,d}| + \frac{\pi s^2 d}{\pi s^2 \Delta d} \sum_{q \in D} |p_{q,s,d} - p_{q,s,d-\Delta d}| \]
\[ = \frac{s^2}{2s - \Delta s} \sum_{q \in D} \frac{|p_{q,s,d} - p_{q,s,d-\Delta s,d}|}{\Delta s} + d \sum_{q \in D} \frac{|p_{q,s,d} - p_{q,s,d-\Delta d}|}{\Delta d} \]  
(A.12)

Setting \( \Delta s = 1 \) and \( \Delta d = 1 \) in eq. A.12 we get eq. 3.8:

\[ W_D = \frac{s^2}{2s - 1} \sum_{q \in D} |p_{q,s,d} - p_{q,s,d-1,d}| + d \sum_{q \in D} |p_{q,s,d} - p_{q,s,d-1}| \]  
(A.13)

Furthermore, as \( \Delta s \rightarrow 0 \) and \( \Delta d \rightarrow 0 \), eq. A.12 becomes:

\[ W_D = \frac{s}{2} \int_{q \in D} \left| \frac{\partial}{\partial s} p_{q,s,d} \right| dq + d \int_{q \in D} \left| \frac{\partial}{\partial d} p_{q,s,d} \right| dq, \]  
(A.14)

which is equivalent to eq. 3.5.
Bibliography


[35] W.L. Lu and J.L. Little. Simultaneous tracking and action recognition using the pca-hog de-


[43] Y. Ke, R. Sukthankar, and M. Hebert. Spatio-temporal shape and flow correlation for action
recognition. In Proceedings of the Workshop on Visual Surveillance (VS) at the Conference on


recognition. In Proceedings, IEEE Conference on Computer Vision and Pattern Recognition,
pages 1–7, 2008.


