Department of Mathematics

Master's Thesis

Statistical Science

Knowing what you don't know

Novelty detection for action recognition

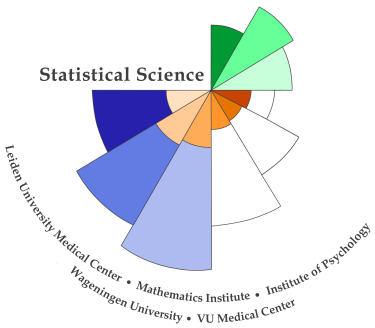
Author:
Thomas Moerland

Supervision:
Prof. Dr. Ir. Pieter Jonker
Ir. Aswin Chandarr
Vision-based Robotics, TU Delft

Dr. Tim van Erven
Mathematical Institute, Leiden University

September 2015

Universiteit Leiden
The Netherlands



Abstract

Introduction Action recognition is an important task for domestic care robots. Current action recognition literature exclusively studies closed-set recognition problems, where performance is evaluated on videos which were also available in the training set. However, the real-world environment is by definition open, and it is both theoretically and practically infeasible to supply the robot with all necessary information beforehand. This work develops novelty detection methodology applicable to HMM-based classifiers, which have shown earlier success in action recognition. By filtering unknown actions instances, our novelty detection module increases system robustness in open environments, and is a first step towards adaptively learning robots.

Methodology We first develop an ordinary action recognition system based on a new skeleton-derived feature and a HMM back-end classifier. The HMM system is estimated in three ways: clustered, Expectation-Maximization (EM) and Extended Baum-Welch (EBW). The latter is a discriminative training criterion, which could theoretically improve novelty detection accuracy. Since the EBW algorithm has only been implemented in speech recognition software, we wrote its first open-source implementation (in Matlab, publicly available from `www.github.com/thomasmoerland/Thesis`). Then, novelty detection is approached from both a posterior probability and hypothesis-testing perspective, which we unify as *background models*. Since novelty detection for action recognition has not been reported before, we investigate a diverse set of background models: sum over competing models, filler models, flat models, anti-models, reweighted anti-models, and some combinations of them.

Results Our HMM classification system has around 95% closed-set recognition accuracy on the Microsoft Action 3D dataset, which is near the state-of-the-art. Performance did not differ between the clustered, EM an EBW estimation methods, although our results do indicate the latter two might be beneficial on more challenging datasets. The optimal novelty detection module combining anti-models with flat models had 78% novelty accuracy, while maintaining 78% recognition accuracy as well. Novelty detection results were consistent over various dataset splits. Discriminative training did not alter novelty detection performance.

Conclusion We are the first to study novelty detection for action recognition. Our results could increase system robustness in an open-set real-world environment, and furthermore serve as a first step towards an adaptively learning robot.

Contents

Chapter 1

Introduction

1.1 The need for care robots

With an ageing population and decreasing economic resources, elderly care is an increasing problem in the Netherlands. According to the Dutch National Public Health Compass, the population of 65-plus aged inhabitants will roughly increase from 2.7 million in 2012 to 4.7 million in 2041 [14]. The growing elderly population puts a heavy load on our care system. Even more worrying is the changing ratio between 65-plus and 20-64 year old people, which is predicted to move from 27% to 51% over the same period. Clearly, our society will need novel approaches to elderly care in the forthcoming years.

Due to the exploding cost of the Dutch national health care system, the government is closing many professional care facilities and nursing homes (with a projected decrease of 40% in the next five years [28]). The official government policy intends to move care from the professional institutions back to the home environment, with care then depending on volunteer caregivers (i.e. usually relatives). However, due to the demographic changes in our society, the next generation will simply lack the manpower to foresee in this need.

A possible solution to this problem is provided by socially assistive (care) robots. These autonomous agents might provide domestic support in a variety of care tasks. The Technical University Delft has over the past years developed multiple versions of a domestic care robot (figure 1.1). The department is currently focused at developing a robot walker (Dutch: rollator) for home support of elderly dementia patients. The robot can be used as an ordinary walker for transportation, but can also function autonomously. It is equipped with a camera and microphones to receive visual and auditory input from the environment.

The set of potential (care) tasks for these domestic robots is almost endless. The robot could provide service-like functionality, like cleaning the house and reminding of medication. It could also monitor the patient, raise an alarm in case of emergency, or provide initial support itself. Finally, the robot could also serve as a social companion, fighting the loneliness experienced by many elderly people.

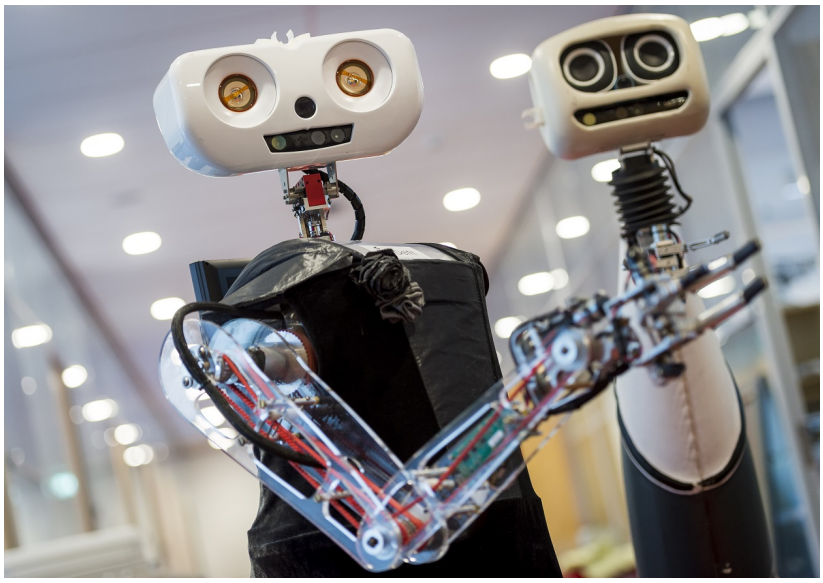


Figure 1.1 *Picture of Lea (left) and Robby, the two most recent care robots developed at the TU Delft. They competed at the Robocup@Home in 2013 and 2012, respectively.*

Most of these tasks can be decomposed in a set of more general subtasks. For example, many applications require correct localization of the patient. For some tasks we are also interested in the *action* currently performed by the patient. The canonical example is fall detection, where the robot should determine whether the patient has fallen or not. More complicated support might also depend on action recognition, like detecting that the patient is eating and reminding him of his medication.

Clearly, proper action recognition is an important component of a fully functional domestic care robot. This thesis will investigate various aspects of action recognition, which are formally listed at the end of this chapter. A small example of action recognition input and output is shown in figure 1.2. The next section will first introduce a quick general overview of robotics, the role of artificial intelligence and learning, and the positioning of action recognition in this field.

1.2 Learning

Robotics is a highly multi-disciplinary scientific field, with contributions from mechanical engineering, control theory, artificial intelligence and mathematics. A robot is usually defined as a programmable machine which can perform a variety of tasks. The scope of robotics is thereby very broad, as it ranges from industrial robot arms to social companions. However, any robot will usually need a sensory part (which monitors the environment), a task planning/reasoning functionality (which decides what action to take based on the sensory information), and actuators (to perform the desired action/response).

The above tasks will crucially depend on intelligence. The implementation of intelli-

Figure 1.2 *Example of action recognition data for three action classes {walk, pick-up, wave}. The top row shows representative stills from the RGB video. The bottom row shows the corresponding depth map. Examples are modified from [45].*

gence in machines is studied in Artificial Intelligence. We define intelligence according to Bengio et al. [19] as operational knowledge: *knowledge that is represented implicitly or explicitly in such a way that it provides an agent with the ability to perform some tasks.* In this view, more intelligent agents can cope with a broader set of tasks. Now theoretically, infinite computational power and complete understanding of the world would enable any agent to make the optimal decision in any situation.

To accurately respond to our environment we will need to process sensory information. The main communication modalities in our physical world are visual and auditory. These are both extensively studied in AI, under the names of computer vision and natural language processing. [19] However, to acquire the intelligence to process pixels into objects/actions and sounds into words, we will need to acquire knowledge. Now any particular knowledge might either be handed to the agent in advance or it might be learned. One could argue to just pass all human knowledge into machines (known as expert systems), but this turns out to be infeasible in many practical applications for two reasons. First of all, we usually don't have explicit understanding of our own operational knowledge. Furthermore, our knowledge system is so complex that it is probably impossible to rebuild it manually. Therefore, any intelligent agent will crucially rely on *learning*.

The computational study of learning is gathered in the field of machine learning, which is one of the major branches of artificial intelligence. In machine learning we study methods to predict future observations from previously observed data. Note that a video is actually a data stream (of pixels), and that recognizing things in a video might

be learned from example videos. Thereby, the aforementioned operational knowledge is formalized through examples. Learning can be applied to video and audio data, but is now also extensively studied in motor control (i.e. reinforcement learning). In this thesis we will apply learning techniques to video data, which constitutes the computer vision field. The major target will be to determine which action is performed by a subject in a given video, which can take values in a discrete set of possible actions. Thereby, action recognition is actually an instance of a classification problem, which is one of the major branches of (supervised) learning.

In short summary, robotics might be an important solution to the elderly care problem, and action recognition is an important component of robotic functionality. The most important approach to this problem is through machine learning. However, action recognition is not (yet) a resolved problem, and there remain many open questions for research.

1.3 Novelty detection: an open problem in action recognition

The main target of this work is the implementation of novelty detection for an action recognition system. Novelty detection concerns the identification of new data which was not seen in the training set (figure 1.3). A main example involves the identification of a novel class in a classification task. In literature, action recognition is usually treated as a closed set problem, i.e. performance is evaluated on action classes that were also available in the training set.

We propose novelty detection is a vital step towards real-world implementation of an action recognition system. First of all, the state-of-the-art action recognition methodology will encounter problems in unknown environment. Current robotic performance is frequently evaluated by deliberately performing known actions in front of the robot. However, especially domestic care robots will have to function in highly unpredicted environments. It is impossible to train the robot with all possible actions in advance. When the robot encounters an unknown action class, a naive system will assign the instance to the most likely class of the known set. Since this assignment is by definition wrong, we need methodology to tackle such open set problems.
The second reason we think novelty detection is important is because of its resemblance of human behaviour. One of the main components of human intelligence is our adaptivity. We can not only detect what we know, but also identify what we do not known. Moreover, we are able to use this new input to increase our knowledge, i.e. we can extend our system. This important aspect of human learning is largely ignored in ordinary machine learning, as is illustrated in figure 1.4. Novelty detection is a crucial step to make our machine learning algorithms adaptive.

We are unaware of any previous attempts in literature to perform novelty detection for action recognition. This work will investigate the possibilities to implement such a system. Novelty detection actually consists of three substeps. First, we have to identify anomalous videos in the data stream. This will be the main target of this thesis. Of

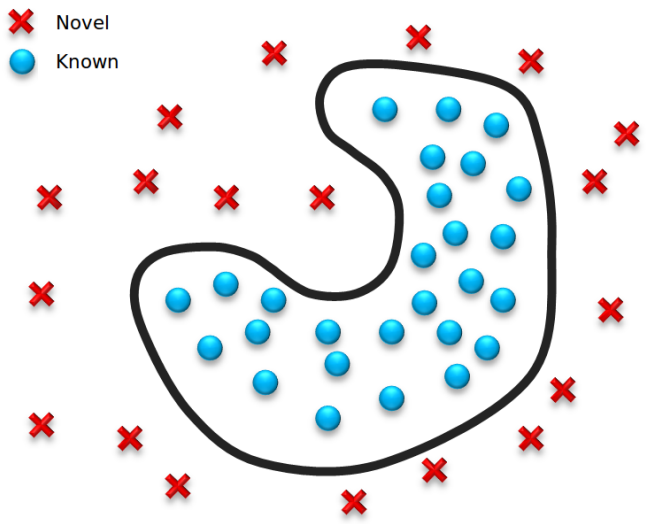


Figure 1.3 *Illustration of novelty detection. For practical purposes we assume a simplified 2D model space. Trained models are represented by blue balls in this space. The models thereby cover a certain part of the model space accurately, as indicated by the black line. However, when we encounter a new action class (red cross), our default models will not function accurately.*

course, for true retraining we also need to detect overlap (*cohesion*) among these anomalous videos and efficiently retrain our system. We will quickly investigate the second topic, while the third is largely ignored.

Before we can investigate any novelty detection we will need a working action recognition system. A major challenge in action recognition is the temporal dependency, i.e. an action is crucially defined by the temporal relation between events. A well-known solution to this problem is through Hidden Markov Models (HMM's), which we will adopt as the classification model in this thesis.

Hidden Markov Models are generative models (in a statistical sense), i.e. they model the full probability distribution of the input (X) and class labels (Y). On the contrary, discriminative models optimize the decision boundary between the classes, i.e. they model the conditional distribution $P(Y|X)$. This distinction is illustrated in figure 1.5. Discriminative training has some advantageous properties, mainly because it is directly related to our target (recognition accuracy). There exist methods to estimate the generative HMM's with a discriminative training criterion. Moreover, results in speech recognition suggest that discriminative training might improve novelty detection [10], possibly due to better class separation. However, discriminative training of HMM's has not been previously tried in action recognition literature. We will therefore implement a discriminative training procedure, to investigate both its possible benefit on a closed-set recognition task, and to assess its potential benefit in novelty detection.

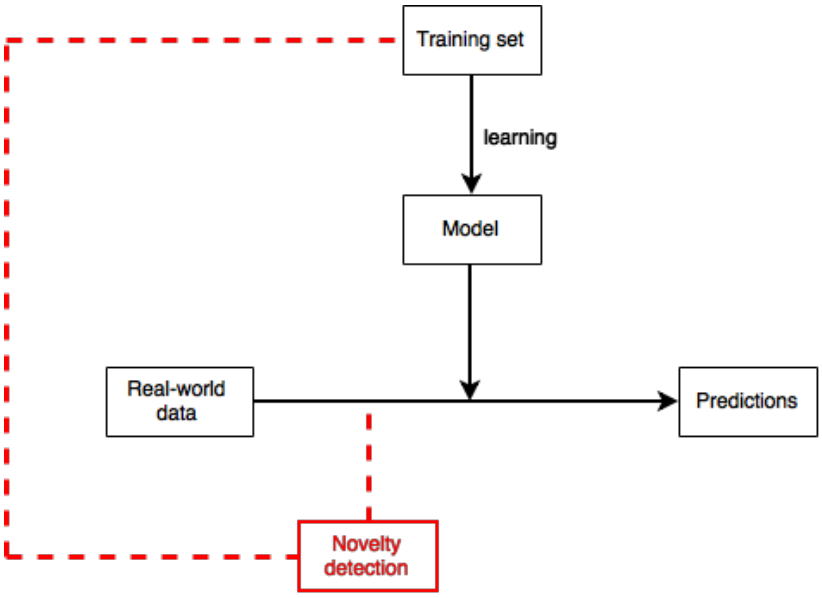


Figure 1.4 *Schematic representation of adaptively learning system. Ordinary machine learning on a closed set problem is shown in black. We manually obtain a training set with representative examples, which implicitly constitutes our knowledge. These examples are used to learn models that make predictions on real-world data. Note that this system does not contain a feedback loop.*
Novelty detection (shown in red) is actually an extension of our model, which identifies possible discrepancy between training set and real-world data. Thereby, we separate our current knowledge (as represented by the training set) from unknown aspects of the world. The detection of these novel elements allows us to extend our knowledge (by extending the training set). This makes our system adaptive to unseen environments, which mimics human behaviour.

In summary, we aim to study methods for novelty detection in action recognition, since it will enable us to built an adaptively learning system. To reach this goal we will first build an ordinary action recognition system based on a HMM classifier and then extend these models through discriminative training. Finally, we identify several novelty detection methods and compare their performance under generatively and discriminatively trained models.

1.4 Research questions

The formal declaration of our research questions is as follows:

1. **Ordinary system** Can we implement an action recognition system with state-of-the-art performance? In particular, can we develop a novel compact skeleton-derived feature, which is suitable for use in a Hidden Markov Model-based classifier?

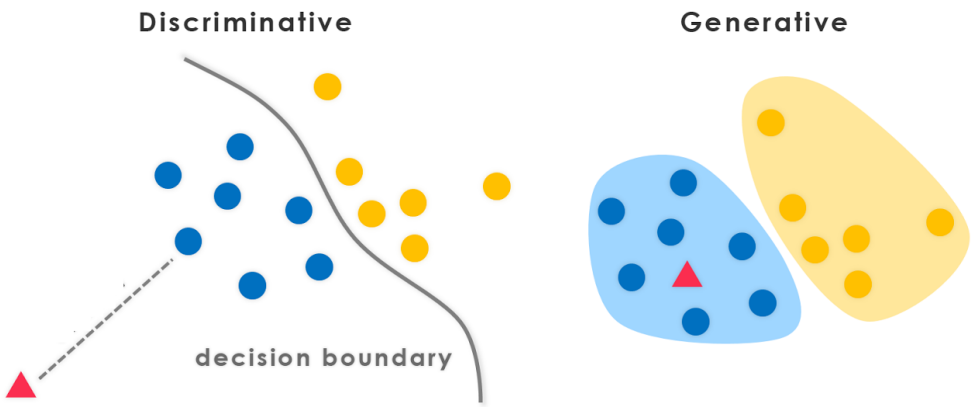


Figure 1.5 *Discriminative versus generative models. The example shows a binary classification problem (yellow versus blue labels) in a simplified model space. Left: Discriminative models optimize a decision boundary between classes. Right: Generative models estimate a probability distribution per class. Of course, the generative models will also give a decision boundary, but this is not optimized (i.e. it follows implicitly). The estimated mean of the blue class is shown as a red triangle (which in the case of discriminative models can deviate from the actual class mean).*

2. **Discriminative training** Does discriminative training (through the Extended Baum-Welch algorithm) of the HMM classifier improve recognition accuracy?

3. **Novelty detection** Are we able to detect novel/unknown videos presented to our action recognition system? How is novelty detection performance affected by different model estimation methods? Finally, is it also possible to buffer novel videos and retrain our recognition system?

1.5 Thesis structure

The current chapter introduced the societal gain of care robotics and the importance of action recognition for care tasks. We furthermore identified the importance of novelty detection to make our system learn adaptively, and thereby function autonomously in an unseen environment. The rest of this thesis is organized as follows:

(i) Chapter two will introduce the general framework of action recognition and review some state-of-the-art literature on the topic. We will furthermore present some initial modelling choices for representation and classification in this thesis.

(ii) Chapter three will cover the mathematical framework of Hidden Markov Models. The combination of the second and third chapter presents a general background, and might be skipped by the experienced reader.

(iii) Chapter four will present the results on the first research question, i.e. implementing a working Hidden Markov Model-based action recognition system. We present

a new skeleton feature borrowing from the PCA torso framework of [32]. We furthermore investigate two estimation techniques (clustered and Expectation Maximization) on a benchmark dataset (Microsoft Action 3D dataset). Both estimation methods show recognition accuracy around 95%, which is around the state-of-the art performance in literature.

(iv) Chapter five will cover discriminative training of our Hidden Markov Model framework through the Extended Baum-Welch Algorithm. We wrote the first open implementation of this algorithm, which was previously only available in speech recognition software. The Matlab code is publicly available at `https://github.com/thomasmoerland/Thesis`. We show discriminative training has equal recognition accuracy compared to the estimation techniques of the previous chapter, although our results do indicate discriminative training might be beneficial on more challenging datasets.

(v) Chapter six will cover the results on novelty detection, deriving the framework from both a posterior likelihood and hypothesis-testing perspective (which we unify as *background models*). At optimal performance we achieve 78% novelty accuracy while maintaining 78% recognition accuracy. Discriminative training (EBW) did not improve novelty detection results. We also show our results are consistent over various dataset splits.

(vi) Chapter seven will discuss the implications of our results, including the strengths and weaknesses of our methodology. We furthermore provide our ideas for future work and practical implementation.

The presented work on novelty detection will be rewritten into a separate conference paper.

Chapter 2

Action recognition

The goal of action recognition is to classify the activity (currently) undertaken by humans. Early advancements in the field date back to the early 1960's. In the meantime, a large variety of approaches have been published in literature. Some recent surveys on the topic can be found in [41], [29] and [1].

Humans are highly capable of classifying actions undertaken by other subject. In particular, any grown-up person can identify whether a person stretches his finger, waves or plays tennis. Although all of these are actions, they obviously happen at different levels. Therefore, human action is usually categorized on four hierarchical levels [1]:

(a) Gestures (atomic actions), e.g. flexing your ankle.

(b) Actions, e.g. kicking.

(c) Interactions (person-person or person-object), e.g. taking a free-kick.

(d) Group activities, e.g. playing soccer.

This ordering in actions can for example be tackled through hierarchical models. Results on atomic actions have been good, while group activities remain too challenging. Therefore, we will only focus on the action and interaction category in this thesis.

Action recognition has recently experienced an important revolution due to the advent of 3D technology. While before 5 years nearly all literature used 2D video, the release of stable and affordable 3D camera's (like for example Microsoft's Kinect) has created an tremendous increase in data quality. Techniques for 3D vision include *motion capture* (MoCap), where the videotaped subjects wear markers at their joints, *stereo-vision*, where 2D views from multiple viewpoints are matched an triangulated, and *range sensors*. The last technology processes depth information real-time, usually by either Time of Flight (ToF, by measuring the travelling distance of a light signal) or structured light (by projection of a pattern with matching), and is implemented on the Kinect camera. Finally, note that the information is not full 3D, since we can only obtain depth information from the visible side of objects (a characteristic which makes some researchers call the new technology *2.5D* rather than 3D). Nevertheless, the advent of 3D technology has created many new possibilities in action recognition.

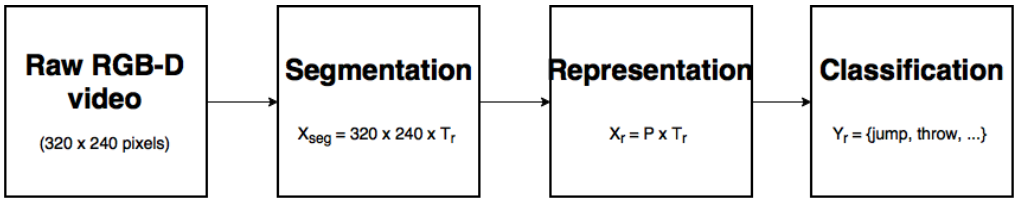


Figure 2.1 *Framework of action recognition. The input is a continuous video stream of an arbitrary resolution. The initial segmentation step splits the video in fixed-length chunks containing a single action. Subsequently, the data contained in a single segment is processed into a more compact representation (in this example of size P by T_r, i.e. one feature vector per frame) In the final step, a classifier assigns a label to the video.*

2.1 Framework

Action recognition can conceptually be divided into three parts. The acquired data involves a video sequence, with at each time-point and each pixel a regular RGB-channel and depth-channel (RGB-D) datapoint. The basic steps of action recognition are segmentation, representation and classification (see [41] and figure 2.1):

(a) *Segmentation*
Before any representation or segmentation can occur, we will first need to split the video stream into separate action occurrences. This is a fundamental problem in many stream classification tasks, like for example speech recognition. It is usually studied separately in literature, and will also be largely ignored in this thesis (i.e. we assume appropriately segmented video's as our input). Nevertheless, the problem is quickly discussed near the end of this introductory chapter (see paragraph 2.4).

(b) *Representation*
Representation is an important topic all over machine learning. It involves the accurate preprocessing over high-dimensional data into compact but informative feature vectors. Due to the very high-dimension of the input, we can not use the raw data as input to a classifier. Computer vision is probably the major example of very high-dimensional data in artificial intelligence, and the research field is dominated by finding good representations.
The representations could be learned with supervised or unsupervised methods, where the latter is actually a recent hot topic in machine learning (known as deep learning). However, in most successful machine learning applications the researchers manually design the feature vectors, usually incorporating some domain specific knowledge. Representation for action recognition will be further discussed in section 2.2.

(c) *Classification*
In the final step we want to learn a statistical model to classify future observations from the representations. A major challenge stems from the high temporal dependency (i.e. an action is crucially defined by its temporal ordering). These topics will be further introduced in section 2.3.

Figure 2.2 *Example of motion-history image representation of the action 'lift arm'. The raw video is highly reduced in dimensionality by stacking the subjects silhouette over time. The bottom representation is used as input to a classifier. Example borrowed from [2]*

Challenges Most of the main problems in computer vision reappear in action recognition as well. We will quickly enumerate these challenges:

(a) *Low-level challenges:* Primary challenges in computer vision include illumination, shadows, occlusions and a cluttered background. A wide variety of techniques have been developed to overcome these, but especially the advent of 3D technology has caused major improvement.

(b) *View-changes (perspective):* The same action may be perceived totally different from another viewing angle. Some methods might benefit from a vary large training database, but the number of possible viewpoints is almost infinite. Therefore, view-invariant representations receive a lot of interest in literature.

(c) *Temporal variation:* Subjects might perform the same actions in different duration and with different speed. Some classifiers can specifically adapt to this variation. Note the difference between segmentation (finding the start and stop moment of the action) and temporal variation (the difference in speed and duration of the action).

(d) *Subject appearance:* Subjects have very different body sizes and heights, which should usually be normalized in some way.

(e) *Scale variation:* The subject performing the action might be at different distances to the camera. However, with 3D technology we now know this specific distance (and might rescale the person).

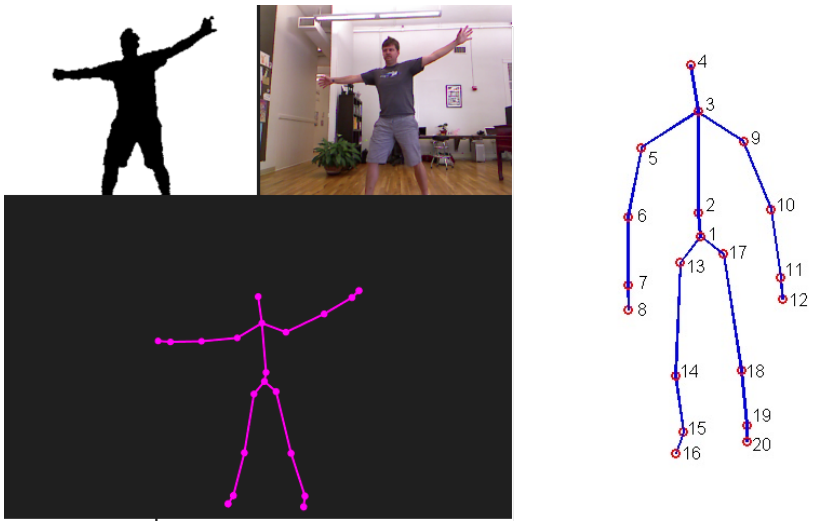


Figure 2.3 *Left: Screenshot of Microsoft's SDK for Kinect, with the original image (top-left), silhouette (top right) and skeleton (bottom). Right: 20-point skeleton with joints labelled as in the Kinect [36]. The subjects origin is located at the neck region (point 3).*

2.2 Representation

This paragraph will discuss the possible approaches to representation in action recognition, i.e. processing the raw pixel values of the segmented video into more compact feature vectors. More importantly, the choice of representation will also affect our possibilities for the classification step, as we will see in the next paragraph. The raw input modality has four dimensions, i.e. a spatial (3D) and temporal (1D) domain. Following [41], we discriminate three approaches to representation (global, parametric and local) on each domain:

Spatial domain

(a) *Global approach*
In this approach we find the detected body (i.e. a region of interest), without the need to label the specific body parts. A well-studied example are silhouettes, which mark the outer body contour. These can for example be stacked over time to form motion-history images (MHI). Due to their simple nature, these methods are computationally efficient. However, they usually suffer from loss-of-information, since they have difficulty with duration variation, and trouble with occlusions and rotations. These methods therefore have more problems with complex actions.

(b) *Skeletal approach*
The second major approach to spatial representation is related to body pose. As early work by Johansson [12] showed, humans can recognize actions by the mere movements of lights attached to the subjects major joints. This invoked the study of human body pose as a basis for action recognition. Pose estimation involves either

labelling (main) body parts or constructing *skeletons*, i.e. wire-frame connections of major joints. Although the idea is very intuitive, this approach was unreliable with conventional 2D camera's. However, a landmark article by Shotton [36] in 2012 developed a frame-wise joint localization. The method is based on a random forest classifier with a very large fully-labelled dataset, and is now implemented on Microsoft's Kinect device (figure 2.3).
Compared to for example silhouettes, skeletons contain more information and they extend neatly to person-person interactions. Most importantly, they are view- and subject invariant. Potential problems are instabilities in the extraction algorithm, and moreover the lack of information about the person's surrounding (e.g. it would be impossible to discriminate drinking from eating, since there is no object information).

(c) *Local approach*
The local approach to representation has been really successful in object recognition. Local descriptors extract many interest points from small regions in the image. Each image is represented by the frequency of occurrence of all these local interest points, i.e. represented as a histogram (known as a 'Bag of Words'). The major drawback of this representation is the complete loss of spatio-temporal relations.

Temporal domain

(a) *Global approach*
In the global temporal approach we treat the whole video as a 4D space-time vector. Thereby we 'freeze' time into the fourth dimension, i.e. into complete blocks of information. We might simply concatenate the feature vectors of multiple frames, but another example is temporal stacking, as we already encounter in figure 2.2. The action dynamics are only implicitly modelled in this approach, as they are hidden somewhere in the whole feature vector. This approach usually results in fixed-length feature vectors, which makes it suitable for traditional statistical classifiers. However, they cannot account for variation in speed. Moreover, they crucially depend on accurate segmentation, and the method does not help in segmentation itself at all.

(b) *Sequential approach*
In the sequential approach we model the action dynamics explicitly as a sequence of feature vectors. These models are directly associated to state-space models for classification (see 2.3), where the action is described as a series of transitions through a finite set of states. The feature vector might be extracted per frame or over a shorter sequence of frames. There is a clear conceptual difference between the pooling over a few frames in sequential models and the pooling/stacking in global models, which is illustrated in figure 2.4. Sequential temporal models can be combined with both silhouettes (global spatial representation) and skeletons (parametric spatial representation). The main advantage of sequential model is their capacity to deal with variation in action speed an style. Moreover, these models do not rely on appropriate segmentation. Since this representation is so clearly associated to state-space classification models, we will discuss these more extensively in paragraph 2.3.

(c) *Keyframe approach*
Humans are capable of recognizing actions from still images, when the picture is

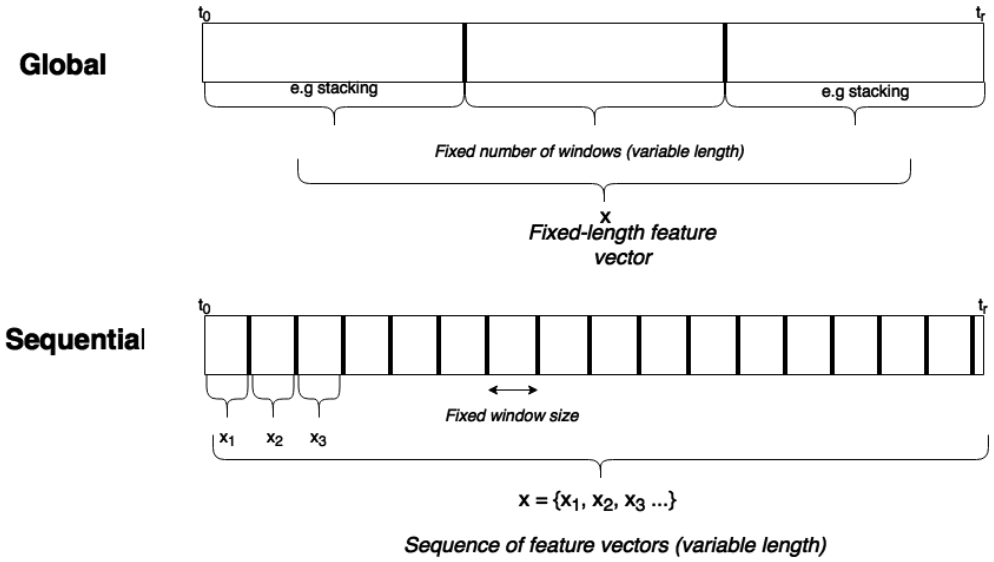


Figure 2.4 *Comparison of global and sequential temporal representation (own figure). Top: in the global approach we separate the full video in a fixed number of parts, which are then transferred (for example by stacking) into fixed-length feature vectors. The canonical example stacks the full video sequence, i.e. has only one window. Bottom: in the sequential approach we fix a (usually small) window size and extract a feature vector per window. The video is thereby represented as a sequence of feature vectors, which will be associated to states. The canonical example has window size 1, i.e. we extract a feature vector per frame.*

taken at the appropriate moment. Keyframe models originate from this notion, which identifies complete actions from unique key positions (which inevitably define a certain action). The main advantage is the complete elimination of the temporal dependency problem, but we clearly loose a lot of information. For example, we can by definition no longer discriminate two actions with similar poses but different ordering.

2.3 Classification

The classification methodology is highly associated to the type of representation. We can in general identify three possible classification strategies: traditional statistical classifiers (which work from a fixed-length feature vector), state-space models and multiple-instance learning techniques. The interplay between representation and classification is shown in table 2.1.

(a) *Traditional statistical classifiers*
All representation approaches that result in a fixed-length feature vector are suitable for most traditional statistical classifiers (support vector machines, k-nearest neighbour, logistic regression etc.). The possibilities are diverse and all methodology is highly studied in other applications as well, which are clear advantages of this approach. In the literature section we will review some recent implementations in of this group.

(b) *State-space models*
State-space models are associated to the sequential temporal representation discussed earlier. In state-space models we explicitly model the action dynamics. The sequence of feature vectors is usually modelled as a series of states. The states take values from a finite set, and the classes are distinguished by the transition probabilities between states. Classification is somewhat indirect: for a new video we usually determine the most likely state sequence (and associated probability) and assign to the class with highest posterior probability.
The main advantage of sequential models are their capacity to generalize over different action speeds and styles. For example, by learning the probability to stay in a certain state, one can model variation in state duration. Moreover, they neatly extend to hierarchical orders, for example building more complicated actions from a sequence of smaller gesture models. However, they will have difficulty with more static actions.
The major example model in this category is the *Hidden Markov Model (HMM)*, which is mainly known for its success in speech recognition applications. We will discuss the official formulation and background of these models in chapter 3, but an example is already presented in figure 2.5. A possible drawback of HMM's is their generative nature, i.e. they model the full probability distribution of video's and labels. Discriminative implementation of these models is further discusses in chapter 5.

(c) *Multiple-instance learning*
The final classification approach inevitably follows from the local spatial representation ('Bag of Words'). Since our features are now actually sparse histograms containing local characteristic, we will need different methods to train our models. A well-known example is Diverse Density estimation. Due to the main drawback (full loss of spatio-temporal relation), these methods are not frequently used in action recognition. The topic will not be further pursued here.

	Global	*Sequential*	*Keyframe*
Global	**Traditional classifier** STOP (offline) [38] Depth Motion Map [47]	**State-space model** STOP (online) [38] Bag of 3D Points [17]	-
Skeleton	**Traditional classifier** Actionlet [40] Eigenjoints [46] SMIJ [27] Orderlets [48]	**State-space model** Histogram of 3D Joint Locations (HOJ3D) [45]	**Traditional / State-space** Action Point [26]
Local	**Multiple instance learning** Random Occupancy Pattern (ROP) [39]	-	-

Table 2.1 *Relation between representation and classification approaches and state-of-the-art literature. The rows and columns represent spatial and temporal representations, respectively. For the combination of representation types the classification model is also implicitly determined (bold text in boxes). Finally, each box is also filled with some representative methods from literature (for full coverage refer to table 2.2).*

2.4 Segmentation

Segmentation is the division of a continuous video stream into subsegments containing isolated actions. The topic will be largely ignored in this thesis, since it is usually studied separately in literature. The three possible approaches are: boundary detection, sliding windows and state-space models. Boundary detection is a very generic method which searches for start/stop moments in single frames or short sequence of frames. These moments might involve extrema's or discontinuities in velocities. The second approach is through sliding windows, where the continuous stream is decoded in windows of varying sizes, and an action is assigned when some class score peaks. These methods have a high computational price, especially since we usually need multiple window sizes. Finally, state-space models (as discussed in 2.3) naturally extend to segmentation as well, since we can incorporate neutral states into our model. This would generate a hierarchical structure, as for example shown in figure 2.7b. The incoming video stream is continuously decoded to the most likely sequence, which will automatically run through neutral states

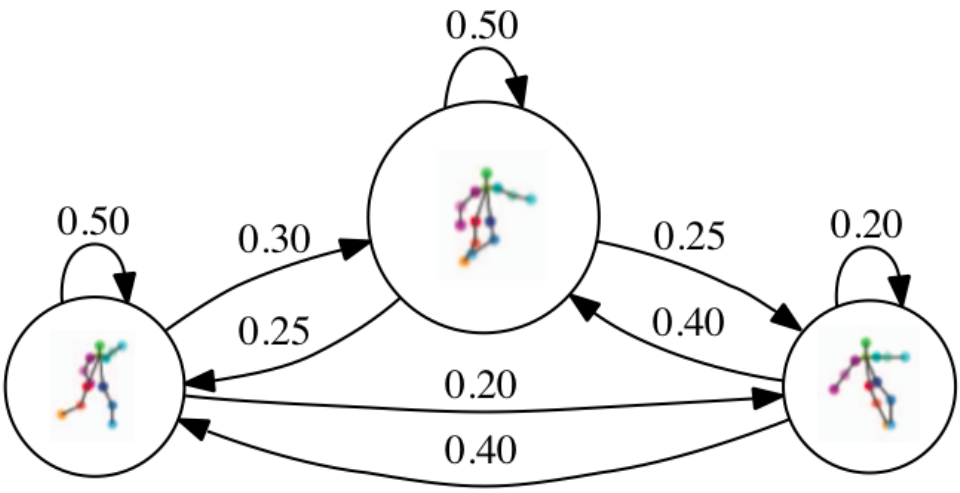


Figure 2.5 *Basic example of a three state HMM. Each state represents a certain key posture, while the whole video is then decoded as a sequence of these key postures. Note that the system can also stay in the same state, which allows for variable duration (speed) of the action.*

as well. Once again, we will work from presegmented videos throughout the rest of this thesis.

2.5 State-of-the-art methodology

The paragraph will cover some state-of-the-art methodology in action recognition. As discussed earlier, the advent of 3D technology has dramatically changed the action recognition field. Therefore, all literature in this chapter dates between 2010 and today. Most articles present results on the Microsoft Research (MSR) Action 3D dataset, which is publicly available from [18]. When available we will report recognition accuracy on the original test set-up (Test 1: 1/3 training, Test 2: 2/3 training, Test 3: Cross-subject). Further details about the MSR Action 3D dataset will be discussed at the end of this chapter.

An overview of many state-of-the-art methods is presented in table 2.2. Moreover, we extend table 2.1 by filling in the discussed literature in the appropriate cells. The text will not cover all methodology presented in the aforementioned tables, but highlight four interesting papers: *Bag of 3D points*, *STOP*, *Orderlet* and *Action Point*.

The landmark paper introducing the MSR Action 3D dataset is by Li et al. [17].

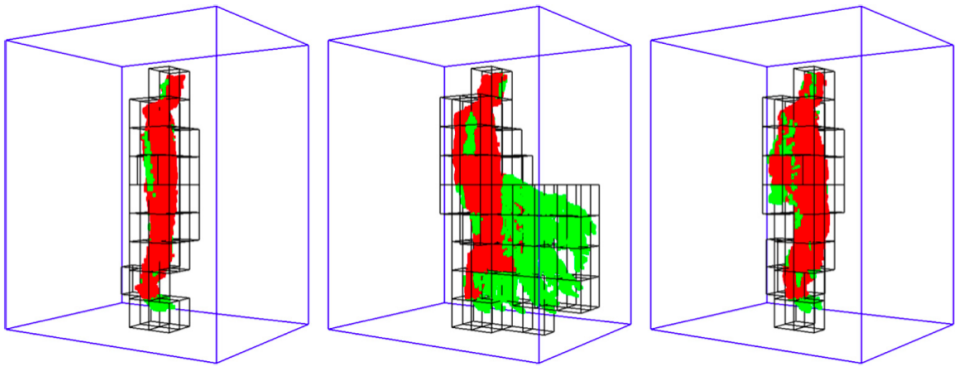
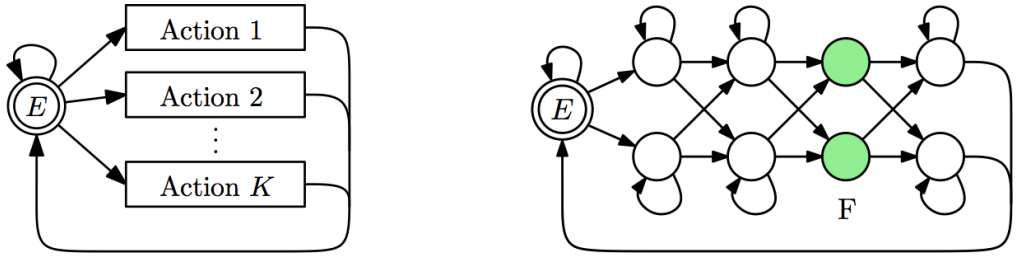


Figure 2.6 *Space-time Occupancy Pattern (STOP) for the action 'forward kick'. The full video is separated in 10x10x10 spatial and 3 temporal blocks. Red and green colour represent full and partial saturation in each cell, respectively. Note that many cells in the grid remain unoccupied during the whole sequence. Each time block holds approximately 20 frames.*

The authors extract a 3D silhouette from the depth map and subsequently sample 80 points on the contour (using the planar projections on three orthogonal planes). They claim their method combines the rich information of body contours (known from 2D) with the computational efficiency of sampling (instead of using the complete depth map). Consequently, each frame is represented as a **Bag of 3D Points**, which are used as the hidden states in an Hidden Markov model with Gaussian mixture observation model. Their recognition accuracies are fairly good, but their method is clearly view-dependent.

Another interesting paper combining global spatial features with a sequential time model is presented by Vieira et al [38]. Their representation is named **Space-Time Occupancy Pattern (STOP)**. The authors divide the 4D space-time into a regular grid of 10x10x10 spatial blocks and 3 temporal blocks. For each cell they calculate the occupancy in the depth map (with some saturation threshold to increase sensitivity). An example is provided in figure 2.6. Although in principle a depth map representation, they do also use the extracted skeleton to rotate the torso and make their representation view-invariant. The raw video representation has dimension 3000, but since most of the cells are empty throughout the whole video (especially at the borders of the grid), they PCA reduce the feature vector to length 300. Recognition is performed using a k-nearest-neighbour classifier. They moreover implement an online version of their algorithm, where they extract a feature every 5 frames (i.e. switching from global to sequential temporal model, referring back to figure 2.4). Segmentation of videos (needed for online implementation) was achieved through a SVM-based neutral pose classifier. After detection of a non-neutral pose, they start decoding the state sequence of STOP features. Their method holds the best result on an online recognition task, with the impressive recognition accuracy of 98.4%.



(a) Global model (b) Single-action model

Figure 2.7 *Action Point model [26]. a) Top level structure, in which a group of neutral state models (E) is connected to the various specific action class models (hierarchical structure). b). Firing-HMM for single Action K, where the state sequence can transition through various paths until a set of firing states (F) is reached.*

Last year Yu et al. introduced the **Orderlets** approach [48], which is especially interesting because it combines skeletal information with local occupancy patterns (LOP). This local occupancy is sampled from the depth map around specific joints (as indicated by the skeletons). This enables the method to also identify person-object interactions, as for example 'making a phone call' would create occupancy (of the phone) around the wrist joint. Furthermore, the authors propose an action can be identified by the ordering of joints, e.g. for the phone call the distance between hand and ear would have to be smaller than the distance between hand and elbow. The learn these intermediate orderings (Orderlets) through some greedy mining algorithm, and then combine them into a Adaboost classifier. Interestingly, their methods casts a class score *per frame*, and they thereby present a very efficient online implementation with smooth window search (i.e. actually just storing the maximum frame-level scores). Performance is only reported on their own (challenging) dataset, with a recognition accuracy of 71.4%.

A final interesting method is reported by Nowozin [26]. The **Action Point** approach exploits the notion of keyframes. The idea was introduced in the previous paragraph: one could recognize dynamical actions by the mere recognition of a single static posture. However, the Action Point approach extends this idea by adding some temporal information about 'how the keypose was reached'. More specifically, primary features are built from skeletons, in particular through joint velocities, joint angles and joint angle velocities. The offline implementation uses a random forest classifier, but for online implementation they built a Firing-Hidden-Markov-Model (see explanation in figure 2.7). However, optimal recognition accuracy (90% on cross-subject) is eventually achieved with ordinary HMM's. A major drawback of this method is the need to manually identify and annotate the keyframes in the dataset, which is both labour intense and somewhat arbitrary.

Method	Representation	Recognition	Recognition accuracy (%) on MSR 3D Action
Bag of 3D Points [17]	Points sampled on silhouette as extracted from depth map.	Action graph [16] with Gaussian mixture observation model	One: 91.6 Two: 94.6 Three: 74.7
Space-time occupancy pattern (STOP) [38]	Occupancy in 4D grid (10x10x10 spatial and 3 temporal boxes). Rotation w.r.t. torso for view-invariance.	Offline: k-NN Online: neutral-pose SVM with Action Graph [16] on 5-frame STOP-features.	One: 96.7 Two: 98.3 Three: 87.5
Depth Motion Maps (DMM) [47]	Stacked motion energy in depth maps	SVM	One: 95.8 Two: 97.4 Three: 91.6
Random Occupancy Pattern (ROP) [39]	Weighted sampling of local occupancy pattern in 4D space-time	Elastic net regularization (to subselect important features) combined with SVM	One: - Two: - Three: 86.6
Histogram of 3D Joint Locations (HOJ3D) [45]	Binning of 12 joint locations with respect to local coordinate system (histogram of locations per frame).	HMM	One: 96.2 Two: 97.2 Three: 79.0
Eigenjoints [46]	Framewise feature vectors from skeletons, through pairwise joint distance, pairwise joint motion, and offset joint motion.	Naive Bayes Nearest Neighbor	One: 95.8 Two: 97.8 Three: 82.3
Sequence of most informative joints [27]	Top-N joints with highest angular variation per time block. Concatenation over full video segment for feature vector.	1-NN on edit distance.	One: - Two: - Three: 34.0
Actionlet [40]	Relative pairwise position difference and local occupancy pattern (LOP) per joint. Fourier Temporal Pyramid (low-frequency coefficients) for joint-wise base features.	Mining of discriminative combinations of base features. Ensemble classifier through multiple kernel learning.	One: - Two: - Three: 88.2
Orderlet [48]	Pairwise position, temporal variation and spatial coordinates per joint combined with local occupancy around joints.	Mining of discriminative orderings (orderlets).	Not reported
Action Point [26]	Base skeleton features (per frame): joint velocities, joint angles, joint angle velocities.	Online: Firing-HMM on last 10 frames, until keyframe fires. Offline through random forest classifier.	One: - Two: - Three: 90.0 (deviating set-up)

Table 2.2 *Overview of various state-of-the-art methods in action recognition literature. For each article the approach to representation, classification and performance on the Microsoft Research Action 3D dataset are reported. The benchmark tests are 1/3 of the dataset for training (One), 2/3 for training (Two) and cross-subject test (Three).*

2.6 Implications for this thesis

This chapter introduced the general framework of action recognition, the various approaches to representation and classification, and presented a short overview of the most promising results in the field. Building from these concepts we can identify some important methodological considerations for this thesis.

Representation As mentioned multiple times now, the stable extraction of skeletons [36] has attracted a lot of interest of the research community. Skeletons seem to be a very informative higher-level data representation, and are also in accordance with our human intuition about action recognition. The skeletons are readily available from the Kinect camera, which is moreover already implemented on the institute's current robots (which would make implementation of possible results more straightforward). Finally, many good Kinect datasets with full skeleton sequences are freely available, which alleviates us from recording our own dataset. Therefore, we propose to further investigate the use of skeletons for this thesis.
It is important to realize that the raw skeleton sequences are not yet appropriate feature vectors, since they reside in an arbitrary space with respect to the camera. Constructing appropriate (frame-wise) representations will be our initial research challenge, and is further discussed in the first half of chapter 4.

Classification The most impressive results in literature on online recognition are probably provided by the STOP [38] technique, which employ a state-space model 3.2 as their classification method. The most common state-space model variant in action recognition is the 'Action Graph', which is a Hidden Markov Model with shared hidden states among actions [16].
HMM's also still hold the state-of-the-art results in speech recognition, which is another important instance of temporal modelling in machine learning. Moreover, speech recognition literature on HMM's could serve a methodological reference throughout this thesis. Finally, HMM's are very intuitive models for action dynamics. Therefore, we propose to use HMM's as the basic classification framework throughout this thesis. These models will be further introduced in the next chapter.

Dataset Our final consideration concerns the choice of dataset. As already reported in literature overview, the MSR Action 3D dataset [18] is still the most reported dataset in the field. It has been recorded a few years ago with a preliminary Kinect version, and holds some notoriously noisy videos. However, the dataset contains dynamic actions and allows for best comparison of our results to other methodology. Therefore, we will adopt the MSR Action 3D as our dataset.

Chapter 3

Hidden Markov Models

An action recognition system should be robust against variation in performance speed, handle noisy features and scale to larger number of action classes. A frequently used solution to these demands are state-space models, which are specific variants of probabilistic graphical models. Graphical models represent a set of random variables through a graph. The absence of an edge between two variables implies conditional independence, which allows for clear visualization of the model's assumed dependency structure (see for example figure 3.1).

For sequential data, like the time-series data in action recognition, we usually assume the current observation somehow depends on the previous ones. However, it is both unrealistic and computationally infeasible to consider *all* historical data. Therefore we adopt the (n-order) Markov property, in which the current observation only depends on the (n) most recent observations.

A well-known extension of these models are state-space models, which also incorporate hidden states (latent variables) to allow for more complex dependency structures. When the hidden states are discrete random variables, these models are known as *Hidden Markov Models (HMM)*. A basic example was already encountered in figure 2.5. HMM's will be used as the main action models throughout this thesis, and will therefore be formally introduced in this chapter. The derivation mainly borrows from Bishop [6] and Murphy [24].

3.1 Notation

We will first give a general overview of some notation used in the current and subsequent chapters. The reader might refer to this overview when reading further sections.

Data	
X	Video random variable
x_r	Observed video $r(\in 1:N)$, of size $P \times t_r$
Q	Set of actions, $q \in \{q_1, q_2, q_3, .., q_{15}\}$
S	Action class random variable
s_r	Class label of video r, taking values in Q
t_r	Duration of video x_r
P	Dimensionality of feature vector per frame
Models	
K	Number of keyposes
W	Set of keyposes: $w_k \in \{w_1, w_2, .., w_K\}$
μ_k	Mean vector $(P \times 1)$ associated with keyposes w_k
Σ_k	Covariance matrix $(P \times P)$ associated with keyposes w_k
A_s	Transition matrix $(K \times K)$ of action class s
z_{rt}	Hidden node in video r at time t
γ_{rtk}	$P(z_{rt} = w_k \vert x_r)$
ζ_{rtij}	$P(z_{rt} = w_j, z_{r(t-1)} = w_i \vert x_r)$
Λ	Full HMM set, as specified by $\{\mu, \Sigma, A\}$
λ_s	Single HMM corresponding to action s, i.e. $\{\mu, \Sigma, A_s\}$

3.2 Markov Models

Markov models are frequently used to model time series. Assume we observe some input modality X, in our case a video, with at each timepoint t some observation vector x_t. Now under the Markov assumption, each observation x_t depends only on the direct previous observation x_{t-1}. When the distribution $P(X_t|X_{t-1})$ is independent of time, we call the chain stationary. The parameters of these distribution might subsequently be estimated, for example through maximum likelihood (ML). When the variables are discrete we call the chain *finite-state*. For such a finite-state chain, the conditional distribution $P(x_t|x_{t-1})$ can be represented as a transition matrix. We denote the states of the finite distribution by $W = \{w_1, w_2, .., w_K\}$. In the case of action recognition, each state of the finite distribution represents a key posture, i.e. a configuration of the body in 3D space. The transition matrix A has size $K \times K$, where each entry denotes the transition probability between classes at subsequent timesteps, i.e. $A_{ij} = P(x_t = w_j|x_{t-1} = w_i)$. Note that each row of the transition matrix sums to one, since it is a probability distribution over the transitions from state i.

Hidden Markov Models extend the ordinary Markov Model through the introduction of hidden states (or latent variables). The graphical structure of an HMM is shown in figure 3.1, where we have introduced the hidden states z_t. The model still assumes the Markov property, but now for the transitions between these hidden states. Note that hereby we retain the tractable transition matrix between the discrete hidden states, while we can now also model continuous output variables x_t. We usually assume a Gaussian (mixture) distribution for $P(X|Z = w_k)$, which is called the observation or emission model. Note that although a certain observation x_t is conditionally only dependent on the associated hidden variable z_t, it is marginally dependent on all other observations.

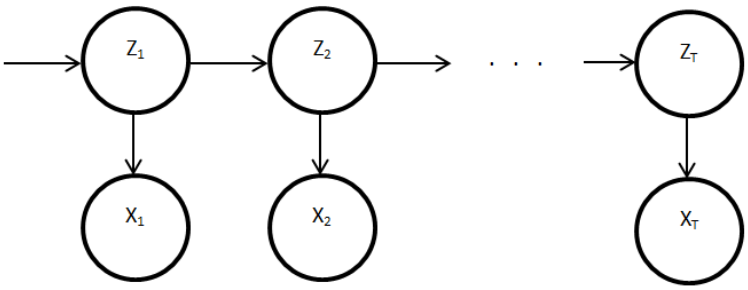


Figure 3.1 *Example of a Hidden Markov Model with T hidden nodes Z, and observation vector* $X = x_1, x_2....x_T$

HMM's can in principle be treated as black box models, but in most cases we are actually primarily interested in inferring the hidden states (i.e. they represent something meaningful). The major success of these models was in speech recognition, where the hidden states represent vowels. For action recognition the hidden states represent key postures (see figure 2.5).

3.3 Expectation Maximization

In statistics we are usually interested in estimating a set of parameters which define our assumed model for the data. In our case we want to estimate the set $\Lambda = \{\mu, \Sigma, A\}$. (Note that an ordinary HMM also estimates the initial state distribution $P(z_1|\pi)$, but we will assume this to be uniform). Probably the best-known estimation procedure in statistics is maximum likelihood (ML) estimation. The likelihood of our model is given by:

$$P(X, Z, S|\mu, \Sigma, A) = \prod_{r=1}^{N} \left(\Big(\prod_{t=1}^{t_r} P(x_{rt}|z_{rt}, s_r, \mu, \Sigma) \Big) \Big(\prod_{t=2}^{t_r} P(z_{rt}|z_{r(t-1)}, s_r, A) \Big) P(s_r) \right) \tag{3.1}$$

We have observed the variables X and S, but our main problem hides in the unobserved variables Z. To estimate our parameters we will therefore refer to the well-known **Expectation-Maximization** algorithm. In this procedure, the unobserved latent variables are iteratively replaced by their conditional expectations. In the E-step, we calculate the occupation probability of each hidden node through the forwards-backwards algorithm. This gives us $\gamma_{rtk} = P(z_{rt} = w_k|x_r)$, the probability that the hidden node in video r at time t is in state w_k. Note that taking the expectation of z_{rt} is not so straightforward. We actually need to sum over all possible paths through the graph, which grows in the order of $O(K^t)$. However, the forwards-backwards algorithm [24] gives us these occupation statistics γ_{rtk} rather efficiently. The derivation of this algorithm is omitted here.

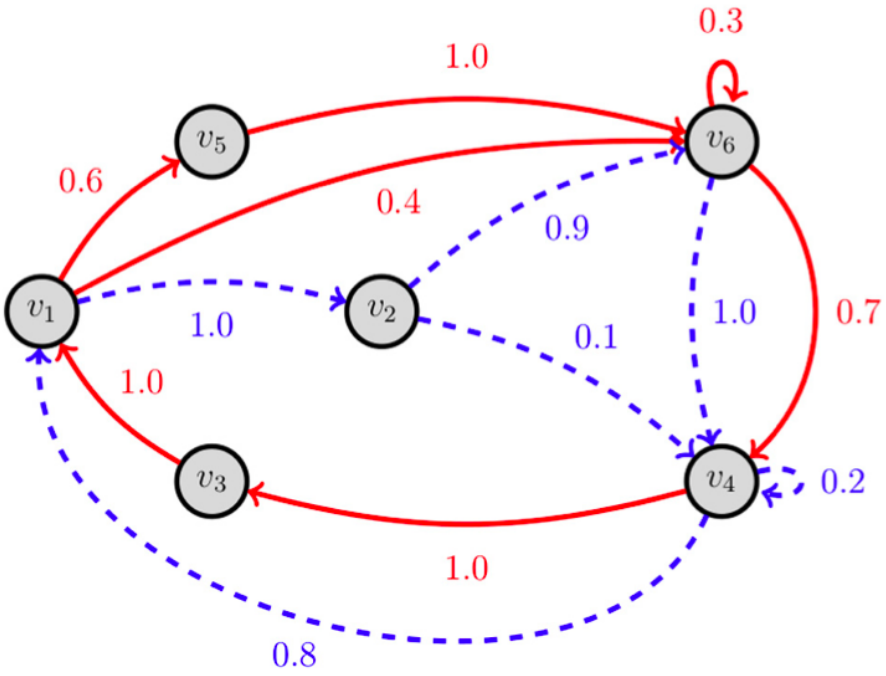


Figure 3.2 *Example of an action graph modelling two actions (classes, shown in red and blue). Both actions share the set of key postures* $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. *However, only postures* v_1, v_4, v_6 *are shared by both actions. The self-loops at some nodes indicate the probability that the duration in that posture varies for the specific action. Figure borrowed from [38]*

After the occupation statistics of the hidden variables have been calculated, we can maximize the log-likelihood function with respect to our model parameters Λ. The naive implementation would estimate separate μ, Σ and A parameters for each action class, adding up to a large number of parameters. A possible solution to this problem is parameter tying, in which case we assume that our models share some of their parameters (and we can jointly estimate these). This approach reduces the risk of overfitting, makes our estimates more stable, and also makes our models more comparable (i.e. they partially live in the same model space). An example for action recognition is given by an Action Graph [16], in which the emission model parameters (μ and Σ) are tied over all models. It is intuitive to assume that different actions use the same orientations of their body posture. The actions are rather separated by the ordering of the body postures over time, which we discriminate through class-specific transition matrices A_s (i.e. these are untied). An example of an Action Graph is shown in figure 3.2.

Rewriting equation 3.1 according to these assumptions and taking the logarithm yields:

$$F_{ML}(X, Z, S|\Lambda) = \sum_{r=1}^{N} \left(\sum_{t=1}^{t_r} log\Big(P(x_{rt}|z_{rt}, \mu, \Sigma)\Big) + \sum_{t=2}^{t_r} log\Big(P(z_{rt}|z_{r(t-1)}, s_r, A)\Big) \right) \tag{3.2}$$

We will refer to the above criterion as the ML objective function. Note that we have omitted $P(S)$ from the formula now. We will assume $P(S)$ to be uniform throughout the rest of this thesis, i.e. we don't assume any prior on the action class distribution. Furthermore, the first summation term does not depend on S any more, which is the assumption made in the Action Graph.

Maximization of equation 3.2 with respect to the model parameters is rather straightforward. It turns out that pooling of μ and Σ makes their ML estimates simply the reweighed sum over *all* videos instead of the *class-specific* videos. Therefore, the M-step results are (after introducing Lagrange multipliers to ensure the sum-to-one constraints in the transition matrix):

$$A_{s:ij} = \frac{\sum_{r=1}^{N_s} \sum_{t=2}^{t_r} \gamma_{r(t-1)i} \gamma_{rtj}}{\sum_{r=1}^{N_s} \sum_{t=2}^{t_r} \gamma_{r(t-1)i}} \tag{3.3}$$

$$\mu_k = \frac{\sum_{r=1}^{N} \sum_{t=1}^{t_r} \gamma_{rtk} x_{rt}}{\sum_{r=1}^{N} \sum_{t=1}^{t_r} \gamma_{rtk}} \tag{3.4}$$

$$\Sigma_k = diag(\frac{\sum_{r=1}^{N} \sum_{t=1}^{t_r} \gamma_{rtk} (x_{rt} - \mu_k)(x_{rt} - \mu_k)^T}{\sum_{r=1}^{N} \sum_{t=1}^{t_r} \gamma_{rtk}}) \tag{3.5}$$

Note that we use $\sum_{r=1}^{N}$ to denote the sum over all videos, while $\sum_{r=1}^{N_s}$ implies the sum over all videos belonging to action s now. It has been shown that the EM algorithm will converge to a local maximum of the log-likelihood function [44].

3.4 Initialization

Since expectation-maximization is an iterative algorithm, we will have to initialize our parameter set in some way. As mentioned before, the EM algorithm will only converge to a local maximum, so we crucially rely on proper initialization. We identified two main methods to initialize our models:

(a) *Flat initialization*
For the flat initialization, we fix all means and variances of the key postures to the global mean and covariance over all training videos. The transition matrix for each action class is initialized at random (respecting the sum-to-one constraints), to allow the algorithm to start 'pulling apart' the key postures.

(b) *Clustered initialization*
The clustered initialization is well described for HMM's, for example in [16]. We cluster all pooled training frames into K clusters by the k-means algorithm. Denote the indicator variable that frame x_{rt} is assigned to cluster w_k by I_{rtk}. Then an

intuitive initialization for our model parameters is given by:

$$A_{s:ij} = \frac{\sum_{r=1}^{N_s} \sum_{t=2}^{t_r} I_{r(t-1)k} I_{rtk}}{\sum_{r=1}^{N_s} \sum_{t=1}^{t_r} I_{r(t-1)k}} \tag{3.6}$$

$$\mu_k = \frac{\sum_{r=1}^{N} \sum_{t=1}^{t_r} I_{rtk} x_{rt}}{\sum_{r=1}^{N} \sum_{t=1}^{t_r} I_{rtk}} \tag{3.7}$$

$$\Sigma_k = \frac{\sum_{r=1}^{N} \sum_{t=1}^{t_r} I_{rtk} (x_{rt} - \mu_k)(x_{rt} - \mu_k)'}{\sum_{r=1}^{N} \sum_{t=1}^{t_r} I_{rtk}} \tag{3.8}$$

In general terms, we initialize the mean and covariance to their specific cluster means, while for the transition matrix we simply count and normalize all transitions between the clusters. Note how closely these equations resemble the EM update equation 3.3, 3.4 and 3.5. While the EM model makes a *soft* key postures assignment (i.e. spread the contributions), the clustered initialization makes a *hard* key posture assignment. Thereby, the clustered initialization is actually an appropriate model itself, and we will use it for baseline comparison.

3.5 Inference

The goal of our system is classify novel videos into the correct action class. Therefore, for a novel video we have to infer a corresponding class from our estimated set of HMM's. Our models are full probabilistic, so a natural choice is based on the posterior likelihood of the new video under each class. This approach is known as the **Maximum A Posteriori (MAP)** decision rule.

For this purpose we first identify the probability of the most likeli state sequence given the observation, i.e. $\text{argmax}_Z P(Z|X)$. In the context of HMM this is referred to as Viterbi decoding. Note that the MAP sequence is not the same as the marginally most likely state sequence. The Viterbi decoding will give $P(X|s)$ for each model class s, after which we assign the video according to:

$$\hat{s} = \underset{s}{\text{argmax}}\, P(X|s) \tag{3.9}$$

Note that the above assignment rule is computationally expensive, since we need a Viterbi path for each action class. Two modifications are proposed in [16]. One of these utilizes a general transition matrix pooled over all actions to calculate the most likeli sequence once. The probability of this sequence is then calculated for each class-specific transition matrix. For optimal performance we decide to rely on the MAP assignment rule.

3.6 Software

The current chapter formally introduced the various aspects of Hidden Markov Model classifiers. We will train a set of HMM's with shared emission models through EM. The analysis is performed in Matlab, for which multiple HMM libraries are available. None of these however directly implement a HMM with shared emission model. We therefore adopt the *HMM toolbox for Matlab* by Kevin Murphy [23]. Although there exists a more recent version of this toolbox (*pmtk3*), the toolbox by Murphy is more easily accessible for modifications.

All scripts, functions and data used throughout the following chapters is made publicly available at `https://github.com/thomasmoerland/Thesis`. The reader can reproduce all results by cloning the repository and running the *main.m* file in a Matlab installation.

The next chapter will implement an action recognition system for the MSR Action 3D dataset, based on the skeleton data (representation) and HMM framework (classification) derived in the previous chapters.

Chapter 4

Research Question 1: Implementing an action recognition system

This chapter will cover the implementation of an action recognition system (research question 1), which involves two steps. First we design a new compact representation based on the 3D joint locations of our skeleton sequences. Afterwards, we train an HMM-based classifier and investigate sensitivity to various model parameters.

4.1 Dataset

The Microsoft Research (MSR) Action 3D dataset is publicly available from [18]. In particular, we will use the real-world coordinate skeleton sequence associated with it. The original dataset contains videos of 20 actions performed by 10 subjects for ideally three repetitions (N=557). Most literature follows the test set-up of the dataset's original paper [17], where classification is evaluated in subsets of eight actions. However, we pool more actions together for better investigation of novel class detection. We selected 15 actions: horizontal arm wave, hammer, high throw, draw circle, hand clap, two hand wave, side-boxing, bend, forward kick, side kick, jogging, tennis swing, tennis serve, golf swing, pickup & throw. We only retained videos with three repetitions per subject and action, and also removed some very noisy videos.

The final dataset contained 366 videos. The test throughout this chapter are all reported over 3-fold cross-validation, where in each instance we use 2/3 of the dataset for training (i.e. two of the three videos per subject per action). This corresponds to 'Test 2' of the original datasets paper. An example of the skeleton sequence for the action 'forward kick' is provided in figure 4.1.

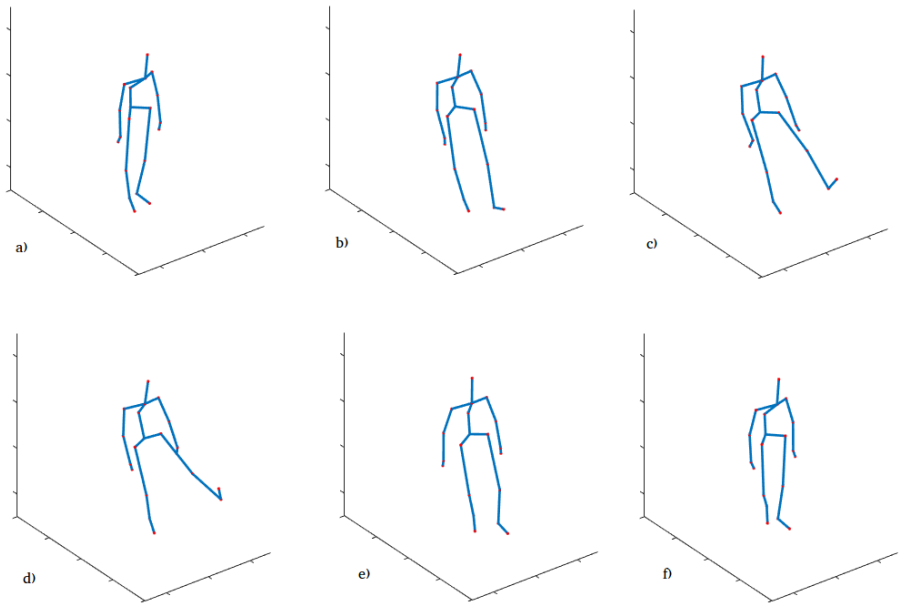


Figure 4.1 *Six still images from a video belonging to class 'forward kick' in the MSR Action 3D dataset.*

4.2 Representation

The skeleton sequences are located in an arbitrary world space, and therefore the raw joint locations are not useful as feature vectors. We would furthermore prefer view-invariant features, which will require additional processing. Proposals in literature to calculate view-invariant representations include for example all pairwise joint distances, joint velocities and joint angles [40] [48]. Our initial implementation used all pairwise joint distances (P=190), but this feature vector turned out to have quite some redundant information. Especially the seven torso joints (figure 4.2a) act almost as a rigid body (i.e. with fixed pairwise distances). Furthermore, our generative models are not optimized to separate the signal from such noise.

Therefore, we designed a more compact feature vector working form the torso PCA framework as developed by Raptis et al. [32]. The authors introduce a method to estimate a local coordinate system (with respect to the subject) based on the seven torso joints (figure 4.2). We define the mean of these seven joints as the origin of our new coordinate system (i.e. translating all skeleton joints). Furthermore, we apply a principal component analysis (PCA) on the 7×3 torso matrix. The three resulting principal components identify the ordered directions of highest variation. Due to the symmetry and elongation of the torso, these axis will align with the vertical, horizontal and forward axis of the torso, respectively. This process is illustrated in 4.2. Note that the sign of a

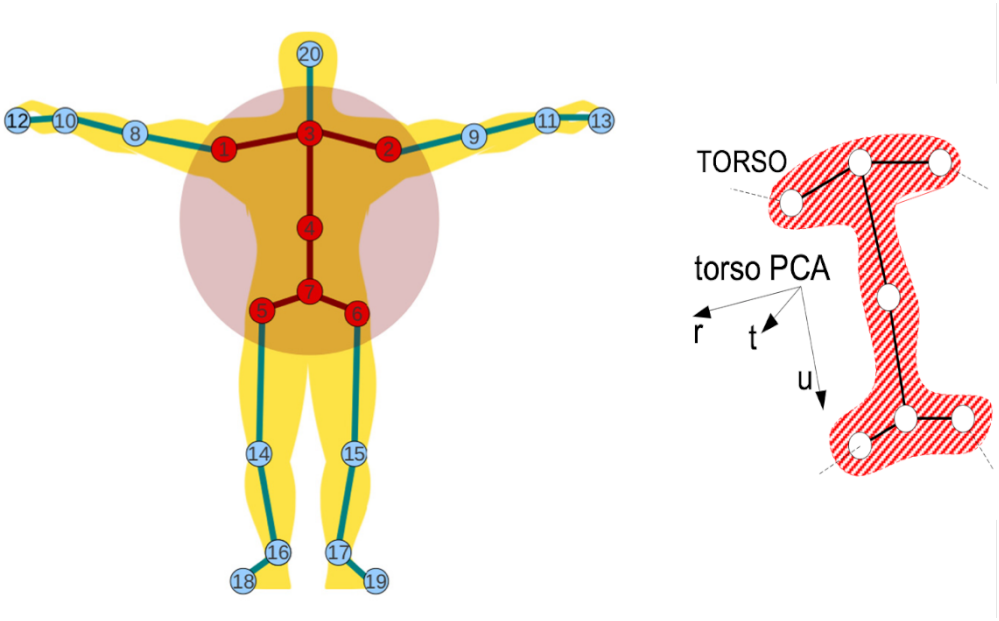


Figure 4.2 *Illustration of torso PCA framework. Left: identification of the seven torso joints which almost acts as a rigid body in 3D space. Right: Estimation of local coordinate system from the torso joints. The u, r and t axis are identified by the first, second and third principal component, respectively. Figure borrowed from [32].*

PCA axis is arbitrary, but we can easily correct this from our joint information.

We construct our final feature vector as follows. After translation we concatenate the 3D world coordinates of elbows, hands, knees, ankles and head. The hand and feet coordinates are very noisy and therefore ignored. Furthermore, we will represent the torso by its rotation in space. The three principal component axes form a rotation matrix, from which we calculate Tait-Bryan angles (i.e. yaw, pitch, roll). Note that these angles are order dependent: we decide to rotate over the vertical axis first (i.e. make the subject camera facing) and then perform the two other rotations. The three angles are concatenated as well, resulting in a P=30 dimensional feature vector.

We did consider some extensions to this feature vector, like rotating all joint locations to make the torso facing the camera (which can easily be achieved since the PCA matrix is a rotation matrix). However, it turns out to slightly decrease our recognition accuracy. We propose this is due to our dataset: most subjects perform their action while facing the camera, and therefore these rotations mostly introduce noise. However, rotation would probably become beneficial on datasets with more variable view-points. Since our main interest for this thesis is with discriminative training and novelty detection, we will adopt the unrotated joint coordinates throughout the rest of this thesis. As a second extension we tried to normalize body size, but this turned out to decrease performance as well. Our final features are illustrated in figure 4.3.

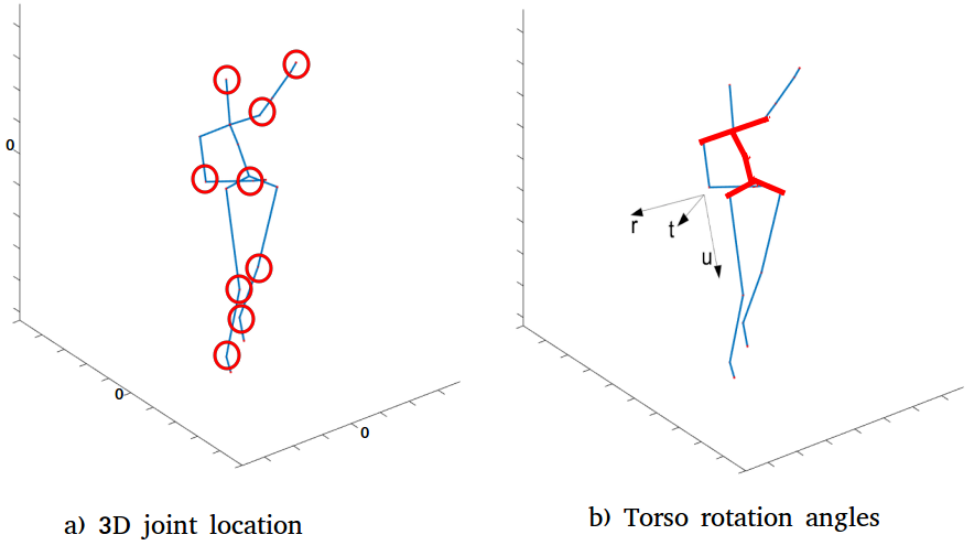


Figure 4.3 *Final features (P=30). Full skeleton is translated to torso center. The feature vector is a concatenation of the 3D locations of 9 extremity joints (circled in a) and the three rotation angles of the torso (from the torso PCA frame, shown in b).*

Principal Component analysis

Although our new frame-wise feature vector of size P=30 is already quite compact, we suspect the individual elements are still correlated (i.e. the effective dimension is smaller than 30). For example, the human body anatomy will cause a raised elbow to correspond to a raised hand as well. However, this covariance structure will interfere with the assumed diagonal covariance structure of our Gaussian emission models (an assumption identified in chapter 3). Since our key postures can only cover our feature space by *spheres* of probability mass, we might need additional key postures to accurately cover the space (as illustrated in figure 4.5). A PCA transformation of the space might remove this covariance structure.

Figure 4.4 shows the results of a principal component analysis (PCA) on the frame-wise features. Apart from removing correlation between the feature vector elements, PCA can also help further reducing the feature vector dimensionality. Figure 4.4 indeed shows the classic L-shape, indicating that the first principal components capture much of our datasets variance. The cut-off value for the number of dimensions is arbitrary, but judging from the plot we might choose 5, 10 or 15 dimensions. As just discussed, both the feature vector length (P) and the number of key postures (K) are intricately related, and we will investigate their joint sensitivity in the next section.

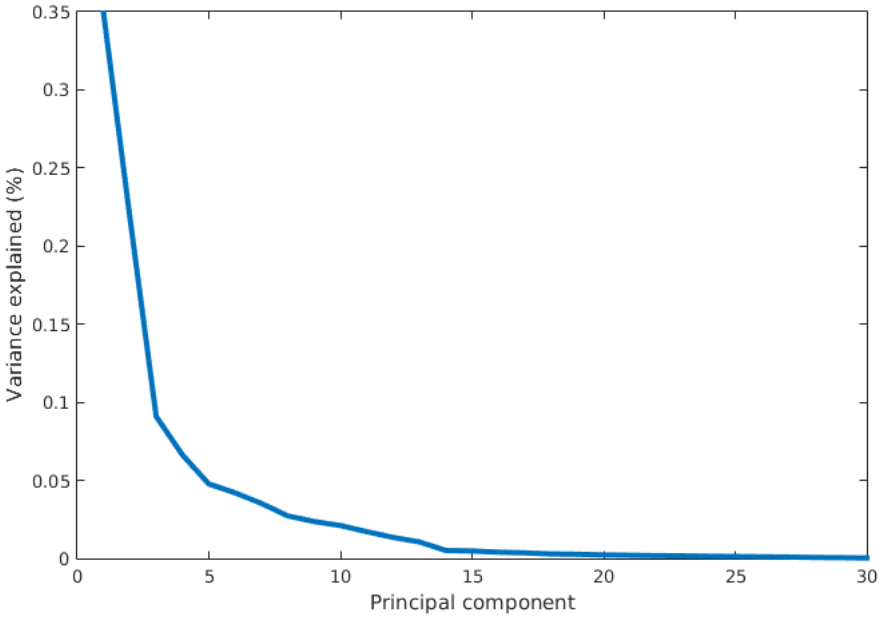


Figure 4.4 *Variance explained per principal component. The first 5, 10 and 15 principal components together capture 77%, 92% and 97% of the variance in our original dataset, respectively.*

4.3 Classification

Clustered model

We implement a HMM recognition system with shared key posture model as described in chapter 3. Initial model performance and parameter sensitivity will be evaluated for the clustered model (as defined in section 3.4). We first evaluate model sensitivity to our two main parameters: P, the feature vector length after PCA reduction, and K, the number of key postures (hidden states). Figure 4.6 shows the test set accuracy as a function of the number of key postures for different values of P. We clearly observe an effect of both K and P. The very compact feature vector (P=3) is inferior due to too much information loss. The other feature vector lengths are quite comparable, especially taking the error bars into account. Since smaller number of K and P will bring us both computational efficiency and parsimony, we will choose K=50 and P=10 as our optimal parameters.

Our approach is not optimal from a methodological perspective, since we optimized two model parameters by evaluating test set performance. However, since these are very basic model parameters, we decided to investigate their sensitivity directly. An ideal approach would independently estimated K and P on each training set, before evaluating performance of the full models on the training set. We will keep K and P fixed throughout the rest of this thesis. The performance of our clustered (K=50, P=10) model on the test dataset is already close to the state-of-the-art results, with an average test set

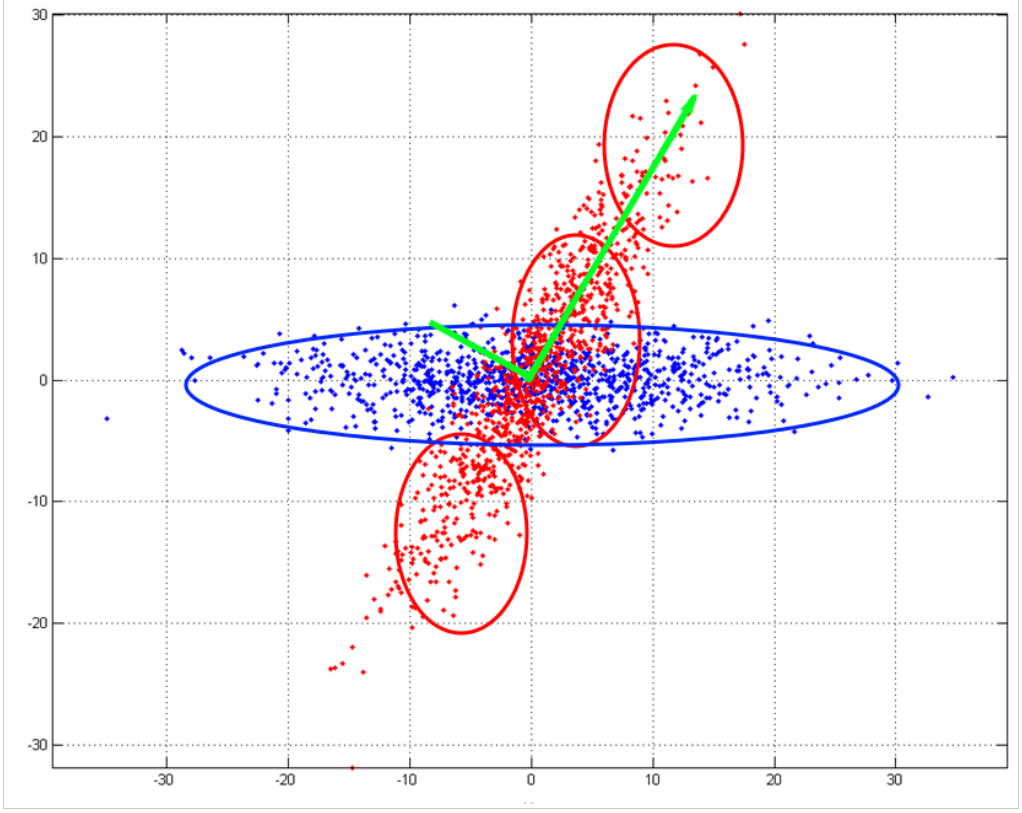


Figure 4.5 *Toy example illustrating the benefit of PCA in combination with Gaussian emission models with diagonal covariance structure. The red points denote a set of features in a two-dimensional space with clear correlation. We will need multiple diagonal Gaussians to accurately cover the probability mass in this space (red spheres). The two principal components of this dataset are shown in green. After PCA transformation of the space we obtain the blue set of points, which can now be captured by a single diagonal Gaussian (blue sphere). Note that we did not reduce the dimensionality of the data in this example.*

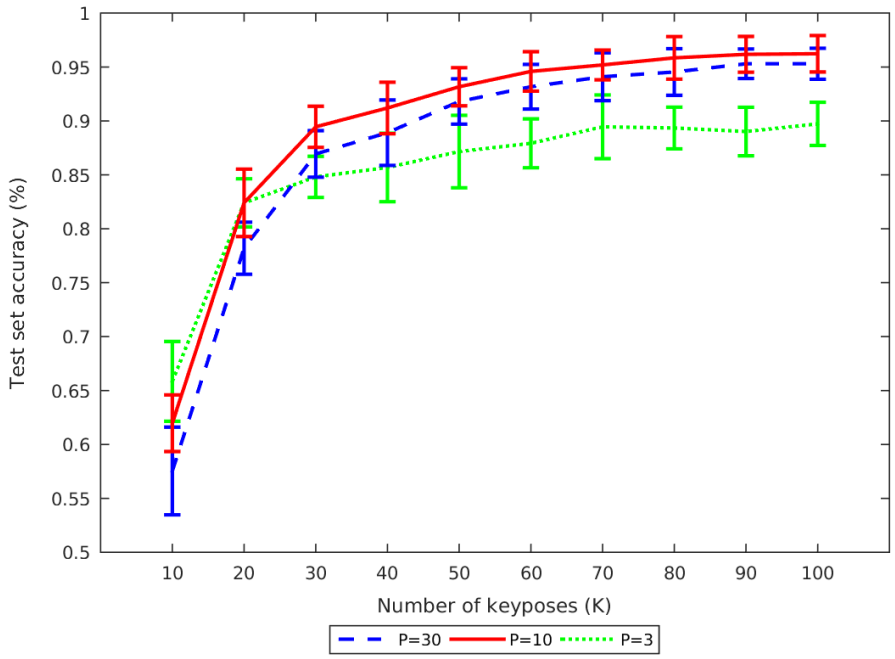


Figure 4.6 *Effect of number of keyposes (K) and feature length (P) on the test set accuracy. Optimal performance is achieved for P=10, with recognition accuracy flattening towards 95%. The very short feature vector (P=3) clear looses too much information. Interestingly, the short feature does outperform the others for smaller number of keypostures (see figure 4.5 for an explanation of this phenomenon). The curves for P=15 and P=5 are not shown, but they were highly comparable to P=10 and P=30, respectively.*

recognition accuracy around 94% (SD: 2.5%).

EM model

The results of our EM implementation are shown in figure 4.7, where the test accuracy, training accuracy and log likelihood (training criterion) are plotted as a function of the number of iterations. The EM model converges quickly, with test set accuracy above 90% after 10 iterations from flat start. The log-likelihood increases monotonically during training (expect for the first iteration, due to the random initialization). To evaluate optimal performance we will either have to specify a fixed number of iterations, or set a convergence threshold. Due to the results in the next chapter (on discriminative training) we will choose to fix the number of iterations. The number of iterations is determine through an extra split in the training set (which we will call the validation set). We adopt the number of iterations with best validation set error, and subsequently train our EM models on the full training set. Optimal performance for the EM model is around

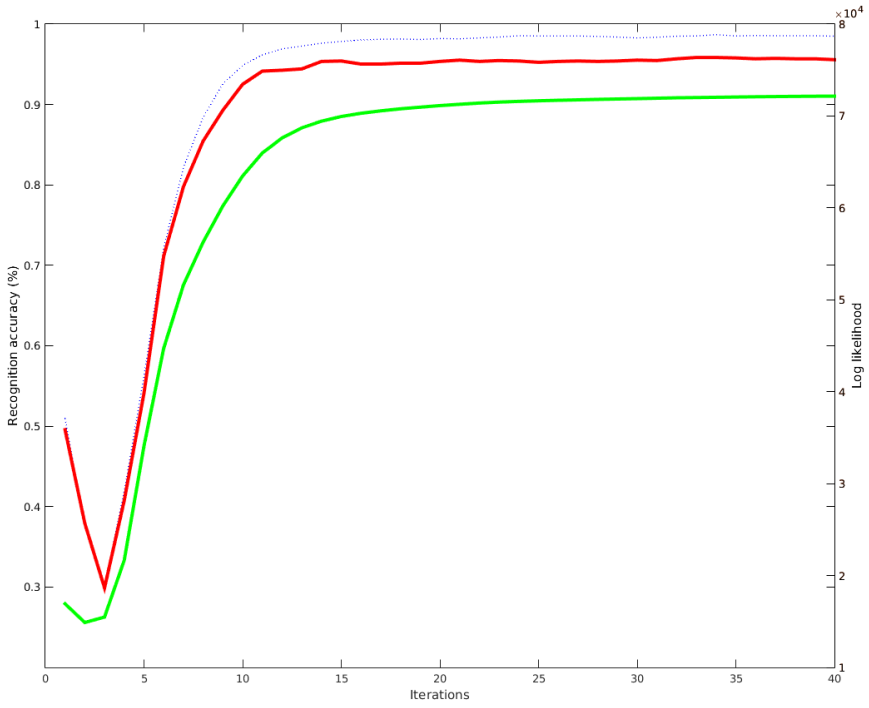


Figure 4.7 *Test accuracy (red line), training accuracy (blue dotted line) and log-likelihood (green line) as a function of the number of EM iterations from flat start. The curves are averaged over 3 runs. Optimal performance is reached around 15-20 iterations (test set accuracy around 95%). There is hardly any overfitting, i.e. test set accuracy remains almost stable after optimal performance.*

96% (SD 2.3 %). We also ran the EM model from the clustered initialization, but this did not alter performance.

Besides the overall recognition accuracy, we are also interested in the individual confusion between classes. The confusion matrix for our optimal EM model is shown in figure 4.8. We can identify some expected confusion, as for example between *Bend* and *Pick-up & Throw* (the former action is actually a sub-action of the latter).

In short summary, the current chapter described the construction of our own feature vectors from the raw skeleton data and subsequent PCA reduction. Furthermore, results of an HMM recognition system built from these feature vectors showed good recognition accuracy.

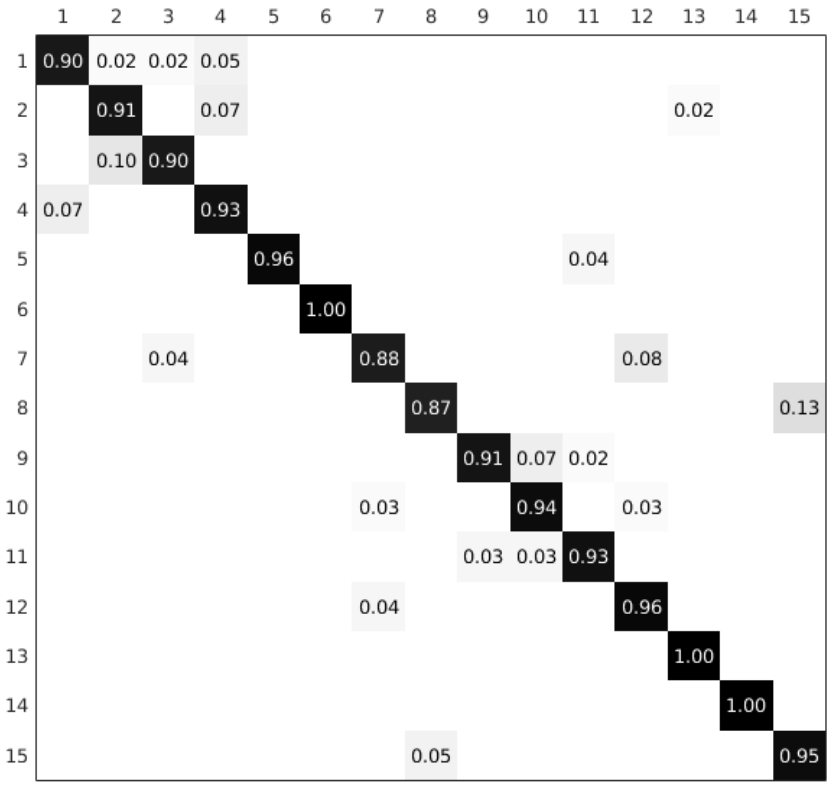


Figure 4.8 *Confusion matrix for EM recognition model. Rows and columns hold the true and assigned class, respectively. Overall accuracy was 94%. The most common confusion is between 'Bend'(8) and 'Pick-up & throw'(15).*

1. *Horizontal arm wave*
2. *Hammer*
3. *High throw*
4. *Draw circle*
5. *Hand clap*
6. *Two hand wave*
7. *Side-boxing*
8. *Bend*
9. *Forward kick*
10. *Side kick*
11. *Jogging*
12. *Tennis swing*
13. *Tennis serve*
14. *Golf swing*
15. *Pickup & throw*

Chapter 5

Research Question 2: Discriminative training

One of the major distinctions in machine learning is between generative and discriminative models. While the former model estimates the joint distribution of features and labels ($P(X, Y)$), the latter specifically optimizes the conditional distribution ($P(Y|X)$. Thereby, discriminative models are actually optimizing the decision boundary between classes (see figure 1.5). Both approaches have their strengths and weaknesses, which were already discussed in chapter 1. The most important advantage of discriminative training is its direct optimization of our true target: recognition accuracy.

In recent years discriminative training of generative models has gained much popularity in machine learning [11]. Researchers in automatic speech recognition (ASR) have investigated various methods, that have shown significant improvement in accuracy over ordinary generative training. [43]. This chapter will cover the implementation of a discriminative training procedure known as the Extended Baum Welch (EBW) algorithm. We will primarily investigate its recognition accuracy (compared to the models from the previous chapter). Hopefully, the discriminative criterion can identify more unique key postures that separate one action from the other.

This chapter will first derive the discriminative training framework, subsequently introduce the Extended Baum-Welch algorithm, and finally investigate model performance.

5.1 Objective function

The HMM system of the previous chapter was actually trained according to the maximum likelihood (ML) objective function. We used an efficient optimization algorithm known as Expectation Maximization to estimate the parameters. We rewrite the ML objective function explicitly here (modified from equation 3.3):

$$F_{ML}(\Lambda) = \sum_{r=1}^{N} log\Big(P(x_r|s_r, \Lambda)\Big) \tag{5.1}$$

The equation clearly expresses the important properties of the ML objective function: we optimize, over a parameter set Λ, the probability of each video under its *correct* action class (i.e. the criterion ignores the probability under the competing hypothesis).

The goal of our classification system is to optimize the 0/1-loss (of which the test set accuracy is an estimate). However, since the 0/1-loss is discontinuous, we usually need a continuous proxy loss-function. A well-known proxy loss for classification problems is $1 - P(S|X)$. Minimizing this loss-function is equivalent to maximizing $P(S|X)$, i.e. the posterior probability of the class given the video.

There exist several variants of the discriminative training criterion for HMM's, each with their own specific loss function. Some examples are **Maximum Mutual Information (MMI)**, Minimum Classification Error (MCE) and Large Margin Estimation (LME). The MMI criterion, as first described by Bahl [4], is probably the most studied of these. The MMI-criterion is actually directly derived from the loss function introduced just above:

$$F_{MMI}(\Lambda) = \sum_{r=1}^{N} log\Big(\frac{P(x_r|s_r, \Lambda)^U}{\sum_{s \in Q} P(x_r|s, \Lambda)^U}\Big) \tag{5.2}$$

The denominator sum runs over the whole set of models Q. The criterion directly reflects the posterior probability of the video under the correct class, but reweighed by a factor U. This smoothing factor, $U \in (0, 1)$ will increase the probability of competing models. Thereby, we smooth out local optima in our criterion and hopefully increase generalization of our model. According to [34], the scaling of U is essential for proper MMI training performance.

As an illustration of our new training criterion, we compare performance of both the ML and MMI objective function on a toy example [30]. Imagine a binary classification task (labels A and B) and an observed video $O^\star$ with probability a and b under each class, respectively. Now if the true label is A, the contribution of this new video to the training criterion will be :

$$ML : log(a)$$

$$MMI : log(\frac{a^U}{a^U + b^U})$$

(Note how b, the probability under the wrong class, only appears in the MMI criterion)

Optimization
As noted by [11], optimization of the discriminative objective function is very challenging. Due to the new denominator term EM estimation of our model will no longer work. Note that the parameter set Λ easily involves several thousands of parameters. For our ordinary model (K=50, P=10) we will estimate $15 \times 50 \times 49 = 36750$ transition matrix parameters and $2 \times 10 \times 50 = 1000$ emission model parameters. Furthermore, the numerical derivatives have become computationally expensive, which highly decreases convergence speed for gradient-descent based optimization methods.

Another approach to optimizing the MMI criterion is through growth transformation. In growth transformation, we derive an auxiliary function of our original optimization criterion. The main idea is that iterative optimization of the auxiliary function (which is feasible) will also lead to growth of the true objective function (which itself is too hard to optimize). On example of such an algorithm is **Extend Baum Welch (EBW)**, originally derived by Baum himself [5]. The name of the algorithm is due to the similarity to the ordinary EM (Baum-Welch) algorithm update equations. The EBW algorithm gives closed form update equations which are relatively efficient to compute and have good convergence properties [9].

It can actually be shown [35] that the EBW and gradient-based update equations are equivalent when choosing the appropriate smoothing constants and learning rates. We initially tried to implement a direct optimization method (BFGS), but this was computationally infeasible on a 8-core i7 laptop with 16GB RAM. We therefore moved to the EBW algorithm, especially since training time might be very important in the robotic context. For example, the robot might want to retrain its action recognition system over night, when the computational load of real-world input is low.

5.2 Extended Baum-Welch (EBW) algorithm

Full coverage of the EBW algorithm for various training criteria can be found in [9]. However, the complete derivation covers over 20 pages and will be omitted here. Instead, we will shortly follow the main rationale. Equation 5.2 actually splits up in two separate functions (for both numerator and denominator statistics), which we will denote by $G(\Lambda)$ and $H(\Lambda)$:

$$F_{MMI}(\Lambda) = \prod_{r=1}^{N} \frac{P(x_r|s_r,\Lambda)^U}{\sum_{s \in Q} P(x_r|s,\Lambda)^U} = \frac{G(\Lambda)}{H(\Lambda)} \tag{5.3}$$

We could calculate ordinary EM occupation statistics for both numerator and denominator, but the problem arises with the negation of the denominator in the log-likelihood (i.e. the function is no longer concave [30]). We therefore construct an auxiliary function of the following form [9]:

$$F_{Aux}(\Lambda|\Lambda') = G(\Lambda) - F_{MMI}(\Lambda')H(\Lambda') + D \tag{5.4}$$

For the auxiliary function we optimize the objective over a parameter set Λ around the current estimates Λ'. D is a independent parameter to ensure positive variance

estimates and determine convergence speed.

Increasing the auxiliary function will also increase the MMI objective function. This property is easily checked by plugging in the current parameter estimates Λ' into the function:

$$F_{Aux}(\Lambda'|\Lambda') = G(\Lambda') - \frac{G(\Lambda')}{H(\Lambda')}H(\Lambda') + D = D$$

And therefore,

$$\begin{aligned} F_{Aux}(\Lambda|\Lambda') - F_{Aux}(\Lambda'|\Lambda') = & F_{Aux}(\Lambda|\Lambda') - D \\ & G(\Lambda) - F_{MMI}(\Lambda')H(\Lambda) \\ & H(\Lambda)\left(\frac{G(\Lambda)}{H(\Lambda)} - F_{MMI}(\Lambda')\right) \\ & H(\Lambda)\left(F_{MMI}(\Lambda) - F_{MMI}(\Lambda')\right) \end{aligned} \tag{5.5}$$

Now the last equation guarantees our requirements, since $H(\Lambda)$ is always positive. Thereby, the left-hand difference ($F(\Lambda|\Lambda') - F(\Lambda'|\Lambda')$, our auxiliary function) is only positive when the right-hand difference ($F_{MMI}(\Lambda) - F_{MMI}(\Lambda')$, our objective function) is positive. From the newly specified objective function we can derive an EM-like algorithm, which is mainly characterized by different occupation statistics (E-step) and tuning of the D parameters (M-step).

E-step

In chapter 3 we defined the hidden state occupation γ_{rtks} as the probability of the hidden node in video r at time t to be in state k under model s. In the EBW algorithm we encounter different occupation statistics, which will be denoted by $\Delta\gamma_{rtk}$. These adjusted occupation statistics are defined as:

$$\Delta\gamma_{rtk} = \gamma_{rtks_r} - \sum_{s\in Q} P(s|x_r, \lambda_s)\gamma_{rtks} = \gamma_{rtks_r} - \sum_{s\in Q}\left(\frac{P(x_r|s)^U}{\sum_{s^\star\in Q} P(x_r|s^\star)^U}\right)\gamma_{rtks} \tag{5.6}$$

The occupation counts are quite intuitive: the first term still holds the ordinary occupation under the true model (known from EM), but we now subtract the sum over the occupations in the competing models reweighed by the posterior probability under these models. Thereby, we diminish the occupation's contribution to the key posture when it is actually outweighed by other models.

An important implementation issue is the scaling of U. In [22] U is set to $\frac{Constant}{t_r}$ for a value of 5 for the constant (but the authors note this depends on the feature type). We will investigate sensitivity of our model to the U parameter in the results section of this chapter.

M-step

With these new $\Delta\gamma_{rtk}$ occupation statistics we can maximize our parameter set through the following update equations (which again largely reflect the EM update equations 3.3 - 3.5)

$$A_{s:ij} = \frac{\sum_{r=1}^{N_s}\sum_{t=2}^{t_r}\Delta\gamma_{r(t-1)i}\Delta\gamma_{rtj} + D_k A'_{s:ij}}{\sum_{r=1}^{N_s}\sum_{t=2}^{t_r}\Delta\gamma_{r(t-1)i} + D_k} \tag{5.7}$$

$$\mu_k = \frac{\sum_{r=1}^{N}\sum_{t=1}^{t_r}\Delta\gamma_{rtk}x_{rt} + D_k\mu'_k}{\sum_{r=1}^{N}\sum_{t=1}^{t_r}\gamma_{rtk} + D_k} \tag{5.8}$$

$$\Sigma_k = diag\left(\frac{\sum_{r=1}^{N}\sum_{t=1}^{t_r}\Delta\gamma_{rtk}(x_{rt})^2 + D_k(\Sigma'_k + (\mu'_k)^2)}{\sum_{r=1}^{N}\sum_{t=1}^{t_r}\gamma_{rtk} + D_k} - \mu_k^2\right) \tag{5.9}$$

The μ'_k, Σ'_k and $A'_{s:ij}$ are the values of the previous iteration. The crucial parameter to tune is the state specific D_k parameter, which is derived from the D in the MMI auxiliary function (equation 5.4).
Tuning of the D_k parameters is crucial for algorithm performance. They should be large enough to ensure positive variances, but also small enough to allow convergence (taking the limit $D \to \infty$ keeps each update at the old value). Some practical guidelines can be found in [30], where each D_k is set to the larger of:

(i) Twice the smallest value to ensure positive variances or

(ii) Twice the denominator statistic, i.e. $2\sum_{r=1}^{N_s}\sum_{t=2}^{t_r}\sum_s P(s|x_r,\lambda_s)\gamma_{rtks}$.

Note that the ensurance of positive variances can be reduced to a quadratic equation in D_k, while the second requirement ensures the denominator of all update equations is positive.

Practical implementation

Another important issue raised by [22] is the tendency of the MMI criterion to also learn the class prior, i.e. $P(S)$, which we have assumed to be uniform. We will correct this issue by reweighing the training data of each class by the inverse of the amount of available training data, i.e. by $w_s = \left(\sum_{r=1}^{N_s}\sum_{t=1}^{t_r} 1\right)^{-1}$.

Another suggestion from the same article is to update the transition matrix through ordinary EM. The transition matrix parameters are very susceptible to small values of both numerator and denominator parts of $\Delta\gamma_{rtk}$ (see equation 5.6). Following [22], we don't expect the discriminative updating of the transition matrix to yield much advantage. Therefore, we only update the emission models in MMI iterations (equations 5.8 and 5.9). We alternate this with estimating the transition matrices through ordinary EM (equation 3.3). Note that the instability in transition matrix estimation is also well-known for gradient-descent optimization problems in the HMM context [30].

Software

To our knowledge the EBW algorithm has not been implemented in an action recognition system before. Furthermore, all available software is within a speech recognition context (e.g. Sphinx and the more generic HTK toolkit). Unfortunately, these have a speech specific design and are both unsuitable and undesirable for action recognition. There, we wrote our own implementation of the algorithm in Matlab, which is publicly available at `https://github.com/thomasmoerland/Thesis` (borrowing basic functionality again from [23]).

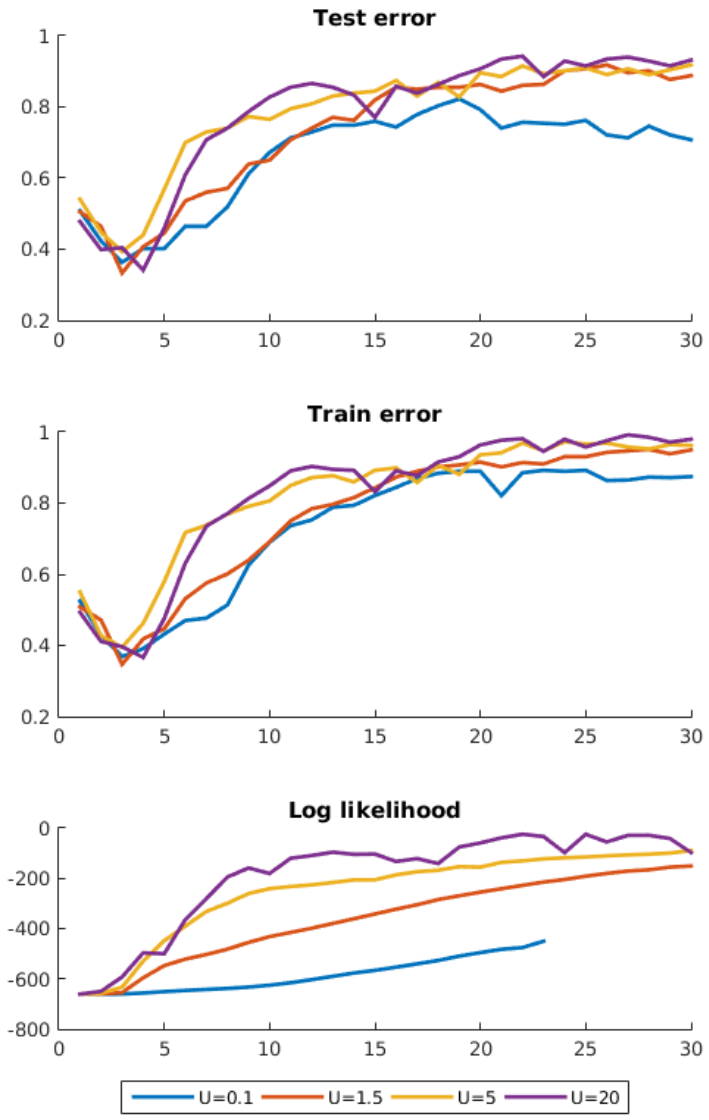


Figure 5.1 *Effect of U on EBW algorithm performance. Figures show the test error, training error and log-MMI criterion as a function of the number of iterations, for different levels of U. Small U flattens the objective function, which results in slower convergence. On the other hand, larger values of U leave a more bumpy objective function with less stable convergence.*

Method	**Recognition Accuracy**
Clustered start (no EM)	94%
EM from flat start	96%
EBW from flat start	95%

Table 5.1 *Recognition accuracy for clustered, EM (both chapter 4) and EBW (chapter 5) models. The three methods shown similar recognition accuracy.*

5.3 Results

All results are again reported for a model with K=50 and P=10. We start by investigating the sensitivity of our model to scaling of U (the smoothing parameter). Figure 5.1 shows the test set accuracy, training set accuracy and objective criterion as a function of the number of EBW iterations from flat start. The small U value of 0.1 clearly has inferior performance, while all three other values eventually reach the same test set performance. However, the larger value of U allows for quicker convergence, but is also less stable in the objective criterion. Indeed, for larger values of U we have a more bumpy objective function (less smoothing). From the plot we choose to use U=5 for further evaluating, since is grows stable in the objective function, has good test set performance, and is suggested in literature [22].

Figure 5.2 shows the performance of the EBW algorithm over more iterations. Note how closely the summed MMI criterion approximates 0 on the log-scale, which corresponds to a posterior probability approximating 1 under the correct class. We now also observe some overfitting in the long run. Optimal test set accuracy is reached around 25 iterations. We again determine the optimal number of iterations per run through a validation set (i.e. an extra split in the training set, as described in the previous chapter). Note that the EBW criterion does not grow monotonically (there are some clear bumps in figure 5.2).

Optimal performance of the EBW model is 95% (SD: 2.2%). In general, the three methods perform almost similar (table 5.1). The discriminative training did therefore not improve our models. However, with an accuracy of 95% there was already little room for improvement. It would be interesting to assess whether the three models (clustered, EM, EBW) have all identified the same global optimum, or reached totally different local optima. This is very hard to determine, since our models involve several thousands of parameters and the key postures might be ordered differently.

Another method to assess the differences between the models is through varying the size of the training set. The ability of our estimation methods to learn from smaller amounts of data is shown in figure 5.3. The plot shows the recognition accuracy for the three estimation methods as a function of the dataset size. The EBW model has some

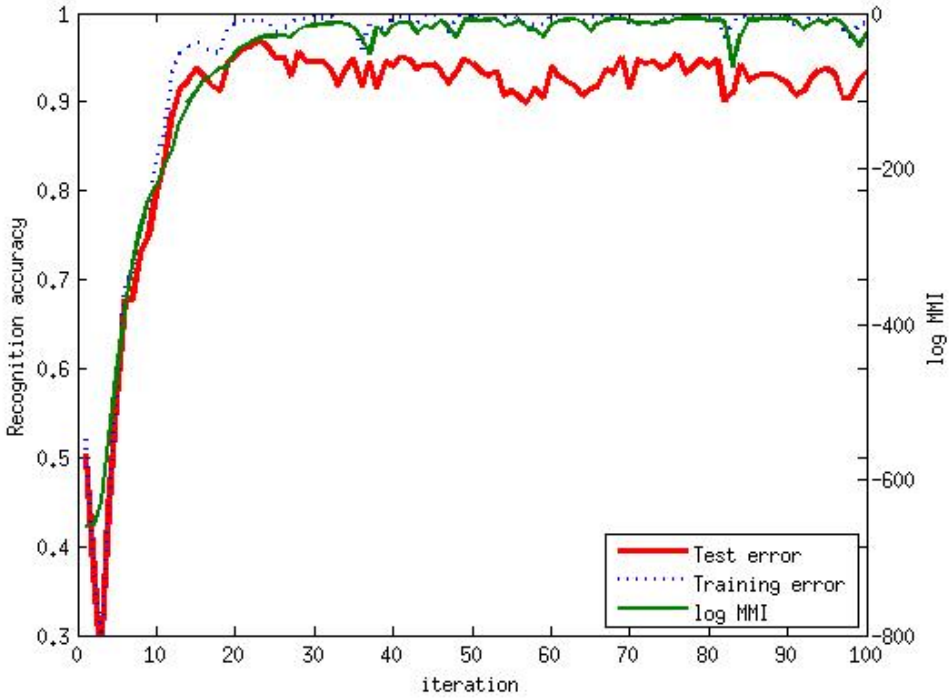


Figure 5.2 *Test error, training error and log-MMI criterion for EBW algorithm as a function of the number of iterations. Algorithm is ran from flat start with U=5.*

difficulties for very small datasets, since it will probably overfit (due to some highly discriminating key postures). However, both EM and EBW are able to learn more quickly for moderate dataset sizes, compared to the clustered model. This indicates both these methods do converge to different solutions than the clustered estimation. Note that the overall performance in figure 5.3 is slightly lower than the number reported earlier. All previous results used two out of three repetitions per action per individual for training. However, figure 5.3 makes a random pick out of all available training videos, which introduces more cross-subject variation. The clustered model seems to have most problems coping with this. However, for larger training set size all models reach performance around 95% again.

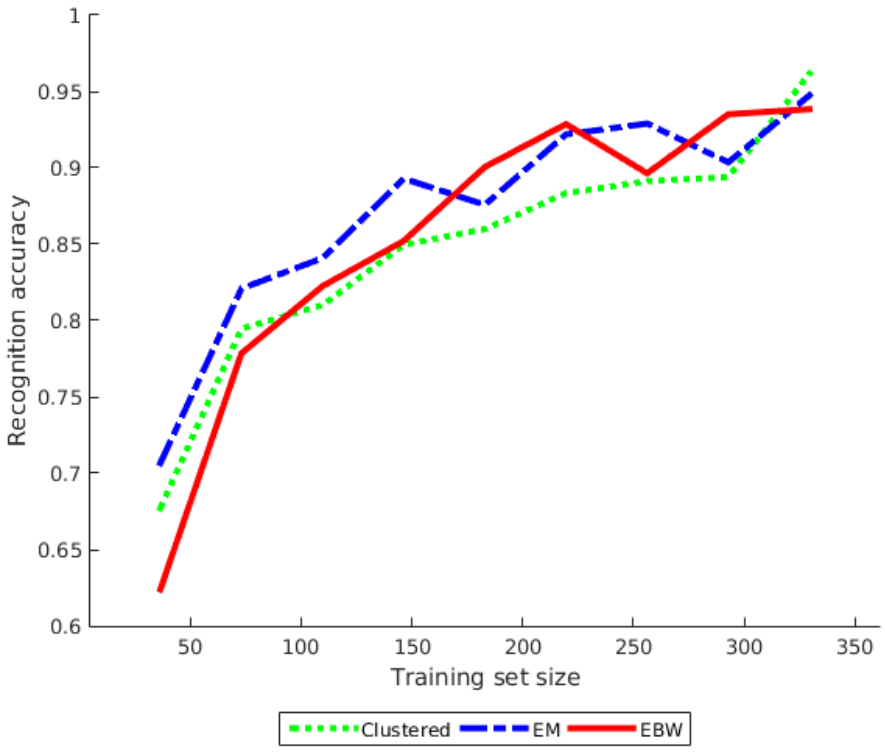


Figure 5.3 *Effect of training set size on recognition accuracy for the three different models. The EM and EBW model are able to learn more quickly, i.e. from smaller amounts of data, compared to the clustered model. In the long run they all reach between 90 and 95% accuracy. Note that the test results at 2/3 training set size are slightly lower compared to table 5.1. The setup for this plot is however not solely inter-subject (see section 4.1), but makes a random split over the data. Although performance slightly decreases, the results also indicate our method generalizes well for larger training set sizes.*

Chapter 6

Research Question 3: Novelty detection

An intelligent system is mostly characterized through its adaptivity. It should not only be able to use learned information on known terrain, but also detect and handle a new or unknown signal. An ordinary classification system, as described in the previous chapters, will assign each new video instance to one of the known classes (i.e. as a closed set problem). However, as with many applications of AI, real life does not comply to these assumptions. Therefore, we need a system which can also detect the occurrence of a new action class, which in machine learning research is known as *novelty detection*.

This chapter will quickly introduce novelty detection and identify methodology applicable to the HMM framework. Afterwards, we present novelty detection results for the previously discussed models (clustered, EM and EBW).

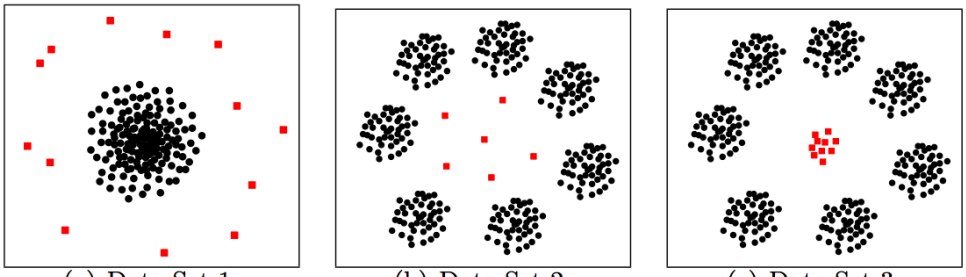
6.1 Overview of novelty detection

Novelty detection can be split into three steps [20]:

(a) *Anomaly detection (separation)*:
We first identify a new chunk of data as unknown/non-classifiable, i.e. as an instance occurring in a low density area under the known models. This might be due to an outlier or due to a novel class.

(b) *Overlap detection (cohesion)*:
The crucial extension of novelty detection (versus outlier detection) is the identification of an overlapping pattern among the outliers. Note that all anomalous videos from the first step should be buffered to detect this *cohesion*.

(c) *Retraining*:
After identification of a set of videos which potentially belong to a new action class, the robot might ask a human to provide the correct class label. As a final step, the



(a) Data Set 1 (b) Data Set 2 (c) Data Set 3

Figure 6.1 *Illustration of novelty detection, modified from [7]. Known and new data points are shown in black and red, respectively.*
a) Clear outliers, most anomaly detection methods will perform well.
b) Known classes showing clustered structure. Anomalies are still scattered, but appear in the middle area. Thereby, probabilistic methods might have trouble identifying the correct density boundaries, i.e. without generalizing to the middle area.
c) Anomalies now appear in a cluster themselves (cohesion), indicating a possible new class.

robot should be able to retrain the HMM system efficiently (i.e. without having to reconsider the full training dataset from scratch).

This chapter will mainly focus on the first step, anomaly detection. In the last part we will shortly investigate the possibilities for overlap detection.

6.2 Anomaly detection

Anomaly detection is a machine learning field studying the occurrence of non-conformant data patterns. A well-known example is fraud detection (for financial transactions). Various approaches to anomaly detection have been studied for fixed-length feature vectors. These are extensively reviewed in [7]. A few novelty detection examples are shown in figure 6.1.

Anomaly detection for HMM's has been specifically studied in speech recognition under the name of **Confidence Measures** (CM $\in [0, 1]$). The researchers in the field have tried for many years to develop methods identifying misrecognitions. A CM should reflect the reliability of the assigned class. Although this seems conceptually different from anomaly detection, the methodology turns out to be similar. A new instance with a low CM under the assigned class probably occurred in a low density area, and is therefore likely to be an outlier or novel video. We will therefore borrow from earlier work on CM in speech recognition. There are three general approaches to anomaly detection for HMM's (as is extensively covered in the review by Jiang on confidence measures [10]):

(a) *Combination of predictor features*
A general approach to anomaly detection is the use of a set of predictor features in

combination with a direct classifier. Any combination of predictor features with a probability density function separating known and unknown classes will be feasible. This approach has been extensively studied, with predictor features including frame-wise likelihood, number of competing hypothesis, posterior probabilities, etc.
However, results in speech recognition have shown that combining features hardly improves performance, probably due to high correlation among the individual features [33]. We will investigate the use of the raw frame-wise likelihood ($P(X|S)$). This quantity is an intuitive choice and will also serve as a baseline.

(b) *Posterior probability*
The second approach to confidence measures is to use the posterior probability. For class assignment we have used the MAP-assignment rule (equation 3.9), which we will expand here:

$$\begin{aligned} \hat{S} = & \operatorname*{argmax}_S P(S|X) \\ & \operatorname*{argmax}_S \frac{P(X|S)P(S)}{P(X)} \\ & \operatorname*{argmax}_S P(X|S) \end{aligned} \tag{6.1}$$

In the last simplification, we assume $P(S)$ uniform and note that $P(X)$ is equal across all actions. $P(X)$ is the marginal probability of our video. Although $P(X|S)$ is good enough to assign the highest class, it is only a relative measure of fit. Even if all models fit poorly, we will by definition find one class to fit best. However, for an open-set problem we will require $P(X)$ as a normalizing constant[8]. This will transform the relative measure of fit $P(X|S)$ into an absolute measure of fit $P(S|X)$.

The theoretical marginal probability $P(X)$ is obtained by summing over all possible classes in the model space. Note that this set also includes all unknown models, and is therefore both unknown and impossible to enumerate. We will somehow need to approximate the background of the model space, which will be discussed quickly.

(c) *Hypothesis testing*
The third approach to anomaly detection for HMM's was developed almost independently at Bell Laboratories [37] [31]. The researchers called their work *utterance verification*, which is actually a post-classification hypothesis test to validate the assigned class. The null and alternative hypothesis are defined as:

H_0: X_r is known and correctly recognized
H_1: X_r is novel and/or incorrectly recognized

A well-known choice, based on the Neyman-Pearson lemma, is to use the likelihood ratio test (LRT) statistic for testing:

$$LRT = \frac{P(X|H_0)}{P(X|H_1)} \geq \tau$$

For simple hypothesis, i.e. when the distributional assumptions are likely to be valid, we might derive the distribution of the test statistic analytically. In the HMM framework H_0 is still somewhat tractable (i.e. the probability under the assigned class), but H_1 is actually a very composite event. We can however determine the appropriate cut-off value τ from our training set (i.e. empirically). However, we still need to adopt an appropriate model for the alternative hypothesis.

Background models: a unified view on posterior likelihood and hypothesis testing

The posterior probability and hypothesis testing approach, as introduced above, are usually presented as separate entities in literature. However, we think both methods can actually be cast in the same framework of modelling the *background of the model space*. When an instance better fits into the background than into one of the models, we have encountered a possible novel video.

From the posterior probability perspective, this background is actually represented by the marginal probability of the video $P(X)$. Our final test statistic is thereby a normalizing constant:

$$\log\Big(P(X|S)\Big) - \log\Big(P(X)\Big)$$

In the likelihood-ratio test (LRT) framework, the alternative hypothesis should also represent the background of the model space. Our LRT test statistic has the form:

$$\log\Big(P(X|H_0)\Big) - \log\Big(P(X|H_1)\Big)$$

Now we can clearly see the overlap between two approaches. In both cases we correct the raw frame-wise likelihood $P(X|S) = P(X|H_0)$ by the probability of the video under the background model. As a confirmation of this similarity, both posterior probability and LRT approaches have independently used filler models ([13] and [31], respectively).

6.3 Background models

This section will present some general approaches to represent the background of the model space, where we have not yet identified any model. As illustrated in the previous section, this likelihood might be interpreted both as H_1 or $P(X)$. Therefore, we will from now on use the notation $C_s^{type}(X_r)$, to denote the background log-likelihood for video X_r under possible recognition of class s. We will omit the s to indicate the background model is not class-specific.

(i) *Sum over competing hypothesis*
In many speech recognition applications, the authors sum over the top-N competing hypothesis to approximate the alternative hypothesis. However, in action recognition we usually have a smaller number of classes, and it might actually be feasible to sum over *all* other classes.

$$C^{sum}(X_r) = \log\left(\sum_S P(X_r|\lambda_S)^{(\frac{1}{t_r})}\right)$$

(ii) *Filler and flat model*

Filler models are very generic background models. In speech recognition they are sometimes called garbage models or catch-all models [8]. Our filler models consist of new transition matrices (denoted by λ^{filler}) trained on all available videos. In speech recognition they have also used background noise to train the filler models. In the context of action recognition, they should reflect an average over all possible human movement.

We propose a generalization of filler models to flat models, which has not been discussed in literature. For flat models we initialize the transition matrix completely uniform over all possible transitions. Although this will permit impossible movements, it removes the dependency on the training data (i.e. filler models will never cover transitions not seen in known models).

$$C^{filler}(X_r) = \frac{1}{t_r} \log\Big(P(X_r|\lambda^{filler})\Big)$$
$$C^{flat}(X_r) = \frac{1}{t_r} \log\Big(P(X_r|\lambda^{flat})\Big)$$

(iii) *(Reweighed) Anti-model*

Anti-models are class-specific competing models. For each known action class s with model $\lambda_s^{(+)}$ we might also train an anti-model λ_s^-. In [31] the authors train anti-models for class s on all video's which do *not* belong to that class.

Now following Matejka [22] we might even extend this approach to more specific anti-models, by reweighing the videos not belonging to class s (denote them by X^{s-}) by their posterior likelihood under model λ_s. Thereby, we reweigh the competing videos by *how likely they are misrecognized as class s*. According to Matejka, we can thereby more closely approximate the direct neighbourhood of the correct class.

$$C_s^{anti}(X_r) = \frac{1}{t_r} \log\Big(P(X_r|\lambda_s^-)\Big)$$
$$C_s^{reweigh}(X_r) = \frac{1}{t_r} \log\Big(P(X_r|\lambda_s^{reweigh})\Big)$$

(iv) *Combinations of background models*

A final approach to the problem is to use a combination of the above models as background [31]. The rationale for combining these models is to borrow the different strengths of the above models. More general background models, like filler and flat models, are good for generalizing over the full probability space. On the other hand, anti-models are more class specific and can thereby more closely model the probability space surrounding the true action class. We will pool the anti-models with both filler and flat general background models, and furthermore adopt a reweighted mean of the likelihoods (as presented by Rahim [31]):

$$C_s^{comb1}(X_r) = \log\Big(0.5 \cdot P(X_r|\lambda^{filler}) + 0.5 \cdot P(X_r|\lambda_s^-)\Big)$$
$$C_s^{comb2}(X_r) = \log\Big(0.5 \cdot P(X_r|\lambda^{flat}) + 0.5 \cdot P(X_r|\lambda_s^-)\Big)$$

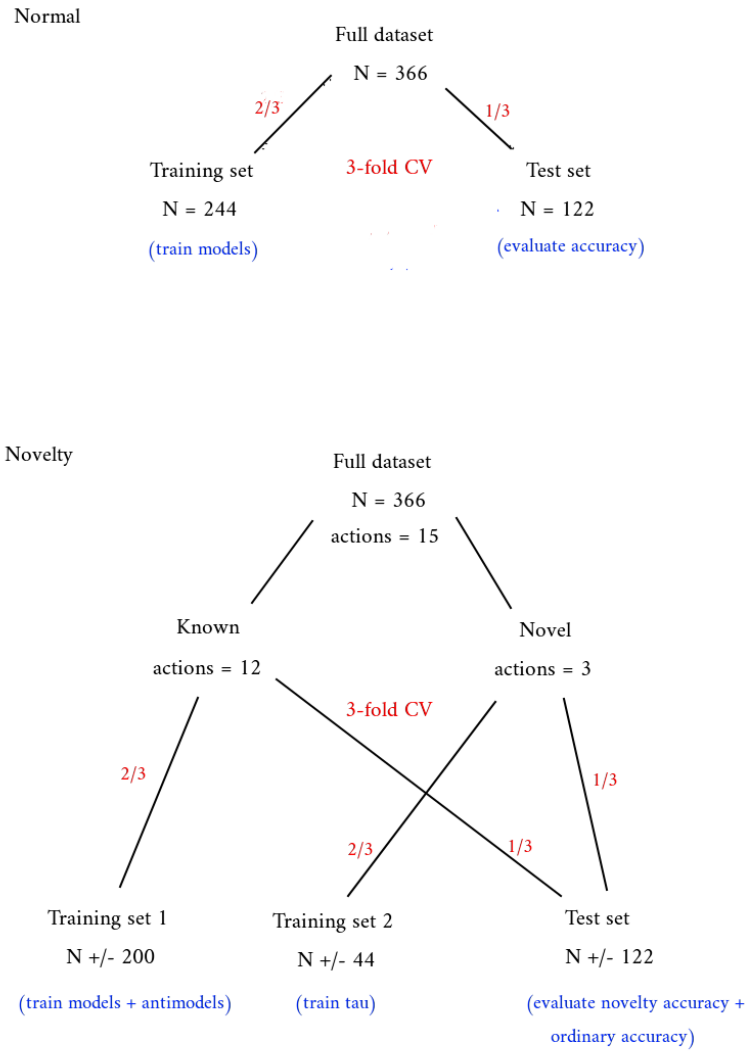


Figure 6.2 *Comparison of test set-up in ordinary action recognition and novelty detection. Top: in the previous chapters all results were reported over a random 3-fold cross-validation. Bottom: for novelty detection we first randomly select 3 videos as 'novel'. We then train our models and anti-models on the known videos (Training set 1) and subsequently tune a cut-off value τ on both known and novel videos (Training set 1 and 2). Novelty performance is evaluated on the independent test set. One full epoch makes a 5-fold novelty split with a nested 3-fold cross-validation over the train and test splits (i.e. 15 runs in total).*

6.4 Test set-up

We investigate the performance of 8 background models (none/raw, summed, filler, flat, anti-model, reweighed anti-model, combination of anti-model and filler, combination of anti-model and flat) in combination with 3 ordinary models (clustered, EM and EBW) on their novelty detection properties. For all combinations we use the background-model corrected posterior probability:

$$\log(P(X|s)) - C_s^{type}(X_r)$$

We will need to determine a threshold value τ as a cut-off for novel class identification. Below τ we identify the video as anomalous/new, since it has relatively low raw probability and relatively high background probability.

To determine τ we will need an adjusted test set-up, which is shown in figure 6.2. We initially split off 3 videos to represent the novel classes. The normal models and background models are then trained on 2/3 of the known videos. The background models are (when applicable) all trained through EM, using the key postures identified in the ordinary models. To select the optimal value of τ we use a second training set, which is split of from the novel actions. For all videos in Train 1 and Train 2 we decode their background-model corrected posterior probability, and subsequently identify the optimal τ. This value of τ is then evaluated on an independent test set (figure 6.2). The full procedure is ran in 5-fold cross-validation (novelty split-off) with a nested 3-fold cross-validation over the train and test set splits (i.e. a 15-fold run for one full epoch, which takes approximately 7.5 hours).

Two types of errors For any binary classification problem we usually make two types of errors (type-I and type-II). However, with the addition of novelty detection we can now make these mistakes on two levels. First of all, a recognition might be wrong on the novelty level (i.e. known identified as novel, or the other way around). However, we can also identify a known video correctly as known, but still assign it to the wrong class. Following [31], we will call this second type a *putative error*.
In practice we are mainly interested in two numbers: sensitivity (recognition accuracy, the proportion of known videos assigned to the correct class) and specificity (novelty accuracy, the proportion of new videos correctly identified as novel). We set τ to the value with the largest sum of sensitivity and specificity on the training set.

6.5 Results

Anomaly detection The sensitivity and specificity for all combinations of normal models and backgrounds models are reported in table 6.2. The models still use K=50 and P=10 for the number of key postures and feature vector length, respectively. The novelty performance of the raw frame-wise likelihood is around 60% for the different training procedures. Interestingly, the clustered model performs better compared to the EM and especially EBW model. The clustered model probably profits from its sparsity, i.e. many of the elements in the transition matrix are zero. Thereby, our known models

	True label	
Assigned label	**Known**	**Novel**
Known (correct)	78%	22%
Known (wrong)	1%	0%
Novel	21%	78%

Table 6.1 *Optimal performance, for clustered model with background model 'Anti-model & flat'. Sensitivity and specificity are also reported in table 6.2. Note that this table also includes the putative errors, i.e. known videos assigned to the wrong action class (middle-left).*

are bounded more tightly in the model space, making it easier to separate them from the background.

The various background models nearly all improve performance, although some do so only by a minor step. The anti-models consistently outperform their reweighed counterpart, which might indicate these reweighed variants model the directly surrounding model space to tightly (i.e. they do not generalize enough). The summed models are mainly strong at detecting novel classes. Optimal performance is achieved for the combination of flat and anti-model with the clustered model training, which identifies 78% of the known videos to the correct class, and is still able to detect 78% of the novel videos as well. The complete confusion table for this results is shown in table 6.1. This table also includes the putative errors, which turned out to be happen for only 1% of the known videos. The amount of putative errors is actually this small for all set-ups in table 6.2.

The previous results all used a PCA reduced feature vector of length P=10. This was a useful modification, as described in chapter 4, but in the context of novelty detection also poses a risk. New videos might also contain new key postures, but this information might hide *outside the PCA subspace*. Therefore, we might have trouble identifying the novel videos correctly. We therefore reran all experiments on the full-length feature vectors (P=30), for which the results are shown in table 6.3.

Interestingly, the performance of our EBW model is especially improved for the full-length feature vectors. The optimal performance is still obtained for the 'anti-model & flat' combination background model, with almost similar sensitivity and specificity. It remains unclear to us why the EBW model experiences most profit from the full-length features. However, overall novelty performance does not seem to be affected much by the PCA. We hypothesize that our training videos already contained a very large portion of the necessary key postures to model all gross dynamic actions.

A closer inspection of our scaling of τ is provided in figure 6.3. The figure presents the ROC-curve and test statistic distribution for the EBW model (P=30) with background

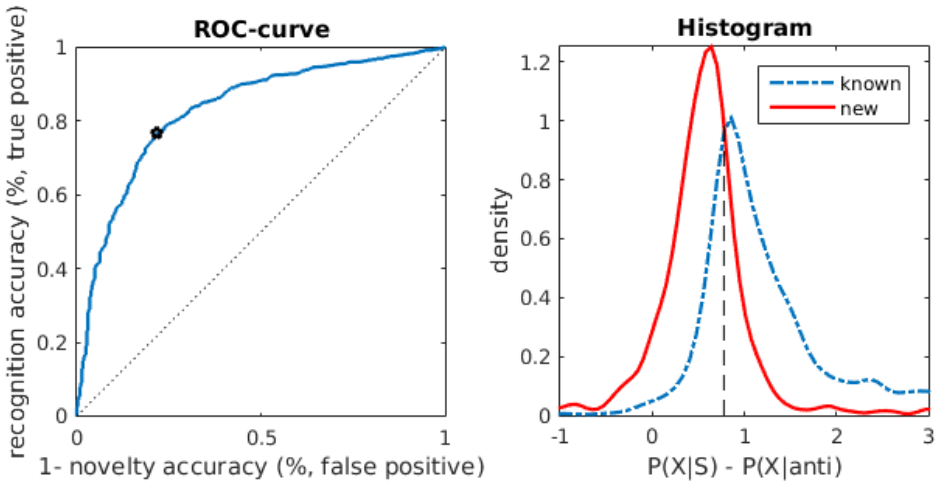


Figure 6.3 *Novelty detection for the EBW model with full feature length (P=30) for background model 'Combination of anti-model and flat model'. Left: The ROC plots the sensitivity and 1 minus specificity for various levels of τ. An ideal cut-off value approximates the top-left region. The best performing value of τ on our test set is marked with a star. Right: Histogram of the test statistic distribution (i.e. the background corrected posterior likelihood) for both the known and novel videos. The optimal value of τ is shown as a vertical dashed line.*

model 'anti-model & flat', which had optimal performance in table 6.3. We observe a decent shape of the ROC curve, where the optimal value of τ is marked with a star. On the right we show the density of the test statistic in both groups, with the distributions still partially overlapping. Note that the optimal value of τ in these plots is determined and evaluated on the same data. However, the sensitivity and specificity in table 6.2 and 6.3 are evaluated on a separate test set (i.e. with pre-determined τ).

The performance of τ in figure 6.3 is almost equal to the numbers reported in table 6.3, suggesting that the scaling of τ generalizes well (i.e. there is not much overfitting). To further investigate this consistency/generalization of τ, we plot the optimal values of τ for different background models in figure 6.4. We observe large variation in optimal τ for the raw frame-wise likelihood, which is the main reason we need background models. Also the reweighed anti-models still show quite some variation in optimal τ, indicating they do not generalize enough to the full model space. For the other background models, τ turns out to become quite consistent (although at different absolute values). This indicates consistency and therefore generalization of our methods to different dataset splits.

Cohesion detection: Multi-dimensional scaling This chapter mainly investigated the first step of novelty detection: anomaly detection. After buffering these videos, the

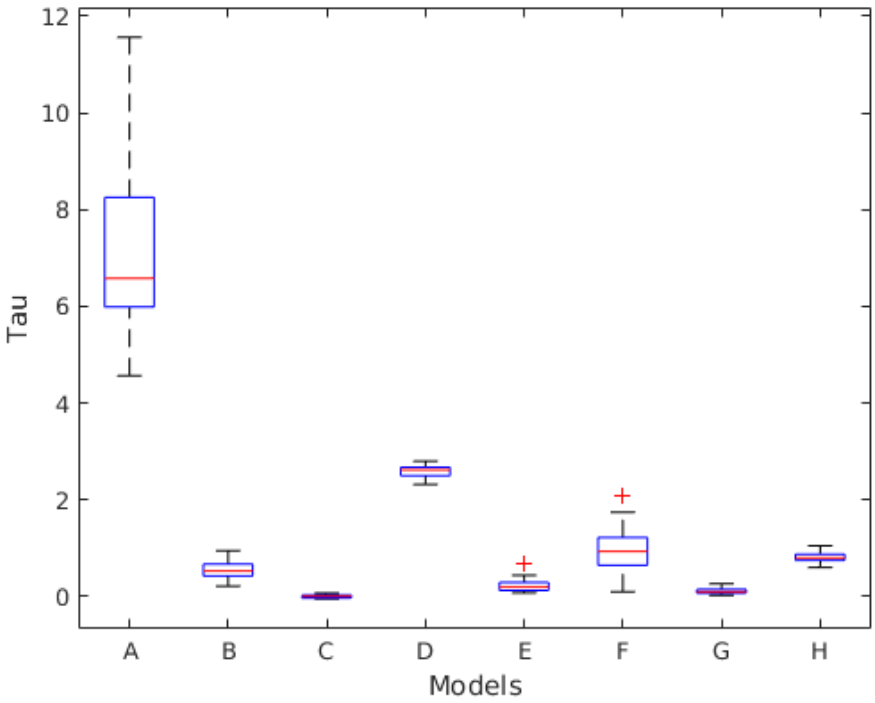


Figure 6.4 *Boxplots of distribution of τ for various background models. Results are obtained over 2 full epochs (30 runs) for the clustered model with P=10. On each run τ is determined as the cut-off value with optimal training set performance. Consistency of τ implies better generalization of our method to different splits.*

(A) Raw (none)

(B) Sum

(C) Filler

(D) Flat

(E) Anti-model

(F) Reweighted anti-model

(G) Combination of C and F

(H) Combination of D and F

next step should detect overlapping patterns (cohesion). This step is less explored in literature, with only one notifiable recent attempt by Masud et al. ([21], not in an action recognition context).
The set of buffered videos can contain various novel classes and misrecognized known classes. For each of these videos we known its posterior probability under the known action classes. This probability might serve as a distance measure between video and known class. A well-known statistical procedure to visualize such (dis)similarity datasets is multidimensional scaling (MDS).
An example MDS results on the buffered videos for one novelty detection split (figure 6.2) is shown in figure 6.5. The three novel classes are coloured, while the false-novel videos are shown as small black dots. We can clearly observe a grouping structure of the novel videos, indicating their similarity (cohesion). However, it will be hard to seperate them from the false-novel videos. Of course, the current representation is only two-dimensional, and we might obtain better separation in higher dimension. Due to time constraints, we were unable to further pursue cohesion detection here.

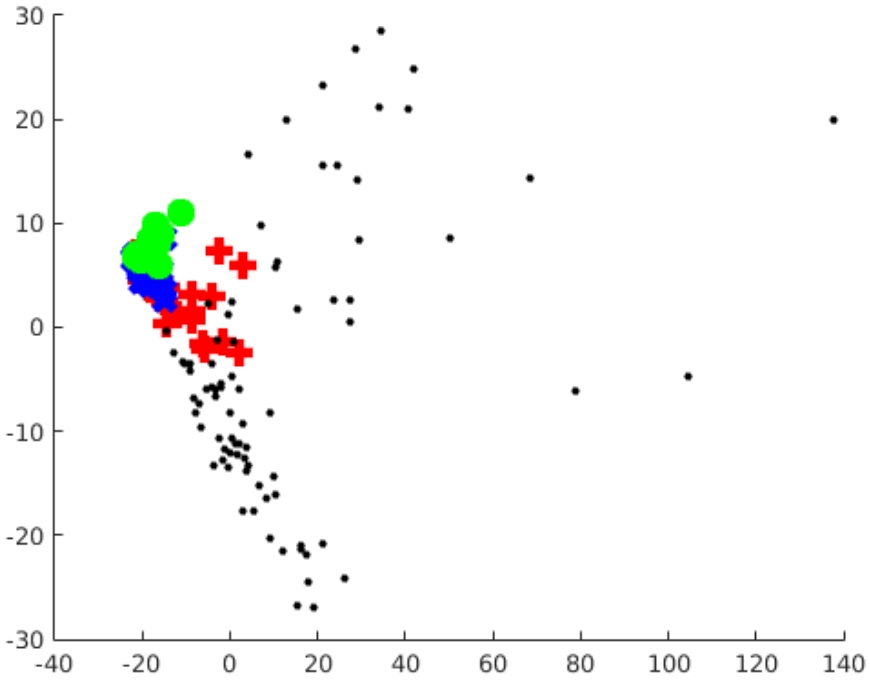


Figure 6.5 *Multi-dimensional scaling (MDS) on anomalous videos (in buffer). Distances between videos are calculated from the Euclidean distance between their posterior probability vector under the known classes. The figure shows the optimal 2D projection of the buffered videos. The three truly novel classes are plotted as blue crosses, red plusses and green balls. The false-novel videos are shown as small black dots. Novel class videos clearly group together (cohesion), but can not be clearly separated from the false-novel videos. (Note: the amount of false-novel videos might be exaggerated, since the prior probability of a known class was higher, as explained in figure 6.2)*

	Model					
	Clustered		*EM*		*EBW*	
Background model	*Sens*	*Spec*	*Sens*	*Spec*	*Sens*	*Spec*
Raw (none)	0.72	0.60	0.71	0.57	0.55	0.57
Sum	0.64	0.77	0.66	0.74	0.64	0.75
Filler	0.73	0.66	0.73	0.58	0.64	0.61
Flat	0.68	0.77	0.71	0.74	0.73	0.70
anti-model	0.77	0.77	0.73	0.69	0.71	0.73
Reweighed anti-model	0.63	0.75	0.70	0.68	0.68	0.72
Combination of filler and anti-model	0.76	0.75	0.73	0.68	0.67	0.70
Combination of flat and anti-model	**0.78**	**0.78**	0.73	0.73	0.67	0.79

Table 6.2 *P=10: Overview of sensitivity (known classes correctly recognized) and specificity (novel classes correctly recognized) for three types of models (clustered, EM and EBW) and various types of background models.*

	Model					
	Clustered		*EM*		*EBW*	
Background model	*Sens*	*Spec*	*Sens*	*Spec*	*Sens*	*Spec*
Raw (none)	0.73	0.61	0.70	0.63	0.61	0.72
Sum	0.54	0.77	0.59	0.73	0.69	0.81
Filler	0.73	0.67	0.75	0.52	0.68	0.64
Flat	0.66	0.78	0.73	0.69	0.74	0.67
anti-model	0.73	0.70	0.75	0.62	0.75	0.76
Reweighed anti-model	0.57	0.71	0.66	0.66	0.56	0.80
Combination of filler and anti-model	0.72	0.71	0.77	0.62	0.72	0.75
Combination of flat and anti-model	0.75	0.75	0.77	0.65	**0.76**	**0.78**

Table 6.3 *Similar to table 6.2, but with full-length feature vectors (P=30).*

Chapter 7

Discussion

Our work has two main contributions to the field of action recognition. First, we studied ordinary (closed-set) action recognition in a HMM framework, where we present the first open implementation of discriminative training in the field. Second, we are the first to study novelty detection for action recognition, which was a topic largely ignored in literature.

Closed-set action recognition and discriminative training We studied three methods (clustered, EM and EBW) to estimate the Hidden Markov Model classifier. Our results indicate the three methods perform similarly (table 5.1), with a mean recognition accuracy around 95%. This is very close to the state-of-the-art results in recent literature on this dataset (see table 2.2, Test 2). We did use a different test set-up, including more classes (15 in our work versus 8 in literature) but removing noisy videos an videos with less than three repetitions per individual per action. The former modification makes the classification task more challenging, while the latter simplifies it again. We expect both modifications balance, and our method does indeed compete with the state-of-the-art results.

Our work presents the first open implementation of the Extended Baum-Welch (EBW) algorithm. This estimation method did not change recognition accuracy compared to ordinary HMM estimation. However, we expect discriminative training will show benefit on more challenging datasets. This can partially be observed in figure 5.3, which shows the results of our estimation methods for different training set sizes. The EM and EBW implementation clearly learn quicker for smaller datasets, which is a more challenging task.

Since the three estimation methods showed the same recognition accuracy, it is interesting to question whether they converged to the same global solution. We identify three possibilities: the objective function has a clear global optimum (which is easily reached by all methods), the objective function is very peaked with many almost equal local optima, or the objective function has a large plateau around the optimum. Due to the large amount of parameters and the arbitrary ordering of key postures in the estima-

tion procedure, it is very hard to assess whether our solutions are actually similar. We do however observe a large difference in raw posterior probabilities, which suggest the models are different. Furthermore, we ran both EM and EBW estimation from clustered start (figures not shown), which showed a large increase in ML and MMI objective criterion, respectively. This also suggests our models are clearly diverging and parameter estimates are changing. We therefore expect the model space has many local optima with similar recognition accuracy around 95%. It is very hard to reach 100% accuracy on any classification task, and we might well approximate an upper bound for our dataset.

Of the three estimation methods, we had least expectations of the clustered model. It was actually incorporated only for baseline evaluation and initialization. However, the clustered model had similar accuracy compared to the iterated EM and EBW models. A possible explanation is the *sparsity* induced in the clustered model. Since the clustered model makes a *hard* key posture assignment, many elements in the transition matrices remain zero, and some key postures are not occupied at all for certain action classes.
On the other hand, EM and EBW estimated models smooth out the contribution of each frame over the key postures. We rewrite equation 3.3 here:

$$A_{s:ij} = \frac{\sum_{r=1}^{N_s} \sum_{t=2}^{t_r} \gamma_{r(t-1)i}\gamma_{rtj}}{\sum_{r=1}^{N_s} \sum_{t=2}^{t_r} \gamma_{r(t-1)i}}$$

For a very small value of the denominator, the estimates in the specific transition matrix row i become very unstable. Therefore, we expect the EM and EBW models will benefit from sparsity (as in the clustered model) as well. We could of course threshold the occupation, i.e. if $\sum_{r=1}^{N_s} \sum_{t=2}^{t_r} \gamma_{r(t-1)i} \leq \omega$ we set all transitions to zero (for that key posture). A more elegant approach is to add regularization to the HMM estimation, as has for example been done in [25].

Finally, the performance of our models on the ordinary recognition task is of course largely determined by the information retained in our features. We designed our own features from the skeleton sequences. Although these features are very compact and efficient, there are two limitations. First of all, we loose more subtle body information like hand positioning (i.e. we cannot discriminate 'thumbs up' from 'thumbs down'). Furthermore, we cannot discriminate object interactions, i.e. drinking (from a glass) and eating (an apple), since we lack information about the object.
A possible solution is to extend our skeleton features with occupation information from the depth map. This would results in a local occupancy pattern (LOP), as for example used in [48]. A second solution would be through deep learning. For example, Baccouche et al. [3] use deep recurrent neural networks (RNN) for action recognition, and report recognition accuracies around 90% on their dataset. Interestingly, the structure of a RNN actually highly resembles our HMM lay-out.

Novelty detection We are the first authors to study novelty detection for action recognition. Our optimal methods identified 78% of known videos to the correct class, while also identifying 78% of the unknown videos as novel. Although there are no recognition accuracies to compare, we do believe our results are a firm starting point. Just

the ordinary recognition accuracy of 78% was for a few years back around the state-of-the-art, and by introducing novelty detection we strongly increased the difficulty of the problem.

We do belief novelty detection is a necessary step in action recognition research, since all current papers confine their study to closed set problems. In our opinion, an interesting result in the novelty detection chapter is the good performance of the flat background model. This uniform initialization is completely data-independent, and was actually included for baseline comparison. On the contrary, all other background approaches utilize the dataset in a smart way to approximate the background. The good performance of the data-independent flat model does touch upon the fundamental challenge of novelty detection: you can not really model the unknown (new classes) from the known (observed data).

We studied discriminative training for HMM's because of its potential benefit for novelty detection (as suggested in speech recognition literature [10]). The advantage of discriminative training (better separating models) could also help to seperate the new classes better. However, in practice the EBW trained models did not change the novelty detection results. We think this might be due to the disruption of the probabilistic space in discriminative training. As already illustrated in figure 1.5, a discriminative training criterion can also estimate high function values in unoccupied model space (as long as the decision boundary is optimal). Therefore, the posterior probability of novel videos might get disrupted in discriminative training, which could explain why discriminative training did not improve novelty detection.

On a more conceptual level, novelty detection can be regarded as the joint occurrence of separation (from known data) and cohesion (among novel data). Our work mainly studied separation, which is a challenging problem for our highly structured models (HMM's). We extensively discussed background models as a way to discriminate truly novel videos (low raw and high background likelihood, i.e. low τ) from noisy known videos (low raw and low background likelihood, i.e. high τ). After this background correction we successfully identified a cut-off value for τ, which generalized well to different novelty splits (see figure 6.4).
It is in some respect surprising that τ generalized well to other dataset splits. In our opinion, τ should be related to the amount of within-class scatter (i.e. the amount of variation between video's of the same class). This process is illustrated in figure 7.1. We expect all our action classes to have similar within-class scatter and therefore our scaling of τ generalizes well over various splits. However, for different datasets or more subtle actions, we might need different values of τ. Ideally, the scaling of τ would be coupled to the within-class scatter.

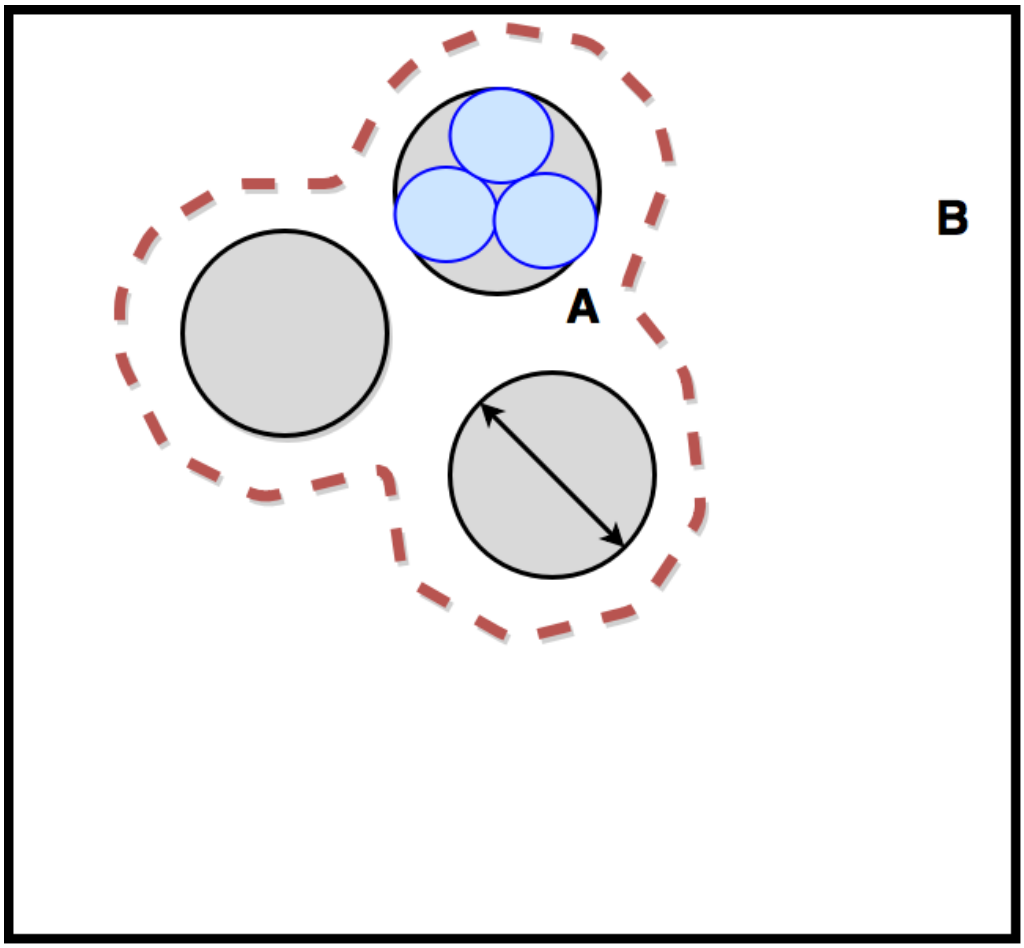


Figure 7.1 *Conceptual illustration of novelty detection. The full model space is shown as a black box. We currently know three action classes (1-3), shown as black circles. The optimal value of τ implicitly identifies a decision boundary surrounding the known classes, which is illustrated by the red dotted line. When we encounter a novel video we either assign to a known class (like for example video 'A') or buffer it as anomalous (like video 'B').*
We expect the scaling of τ is implicitly related to the within class scatter (black arrow). More specifically, for the particular size of the black circles, it is reasonable to assume that 'A' is just a noisy known video, while 'B' is truly anomalous/new.
On a lower hierarchical level we will however need a different value of τ, since the within-class scatter is also smaller. For example, if the top class is the action 'tennis swing', then the blue circles might indicate subclasses, like 'forehand swing' and 'backhand swing'. Now we reconsider video 'A': compared to the blue circles, this video is probably anomalous. This is quickly evident to the human eye (we excel at hierarchical classification), but is particularly due to the smaller within-class scatter of the blue circles. To discriminate video 'A' from the known classes on the blue circle level, we will need a different value of τ.

Applications and implementation Our results on novelty detection, in particular on anomaly detection, have important applications on a domestic care robot. First, we built an action recognition system with new features and new estimation method (EBW) which shows near state-of-the-art performance. More importantly, our anomaly detection framework enables the robot to filter out actions from unknown classes, which will make the system much more robust to an open environment.
A basic application of our work in the domestic care setting would be 'fall detection', i.e. determining whether the patient has fallen on the ground. This is a binary classification problem of a dynamic action, which should be well suited for our action recognition system. For more extensive implementation we would first need to identify a set of actions with relevancy for the care setting. We expect activities like 'eating' and 'sleeping' and signs like 'come closer' to provide significant information and add functionality to the care robot. A novel training set of these actions should ideally be recorded on elderly subjects with a recent Kinect model. As an alternative, an interesting public dataset on domestic activities is available from Cornell University [15].

We identify two final barriers to practical implementation. First, this work largely ignored the segmentation problem, i.e. separating a continuous data stream into seperate action instances. However, the advantage of our state-space models (HMM's) is their natural handling of segmentation. We propose a system like the graph in figure 2.7, which would perform segmentation by incorporating neutral pose nodes.
Another barrier to implementation is the limited testing of our results on cross-subject datasets. We largely ignored this problem, but results from other papers (see table 2.2) show the importance of between-subject variation. We could collect a larger training set or improve system robustness, but an interesting addition could be *speaker adaptation*. This sub-field of speech recognition (reviewed in [42]) studies the retraining of a HMM system with a small amount of subject-specific data, which has shown important improvement in speech recognition accuracy. In a similar way, our robot could also be retrained with some patient-specific action videos.

Conclusion This work is the first to assess novelty detection for action recognition. We were able to detect 78% of the videos from unknown classes (novelty accuracy), while also assigning 78% of the known videos to the correct class (recognition accuracy). We also wrote the first open implementation of the Extended-Baum-Welch algorithm (in Matlab, publicly available at `https://github.com/thomasmoerland/Thesis`. The interested reader can also reproduce all other results in this work from this repository). Although the results of the EBW model were comparable to the ordinary EM and clustered models, we expect EBW to increase performance on more challenging datasets. In conclusion, our action recognition system with novelty detection module might be an important step to move from the closed laboratory environment to the real-world domestic situation. We intend to implement our results on a real-world system in the near future. The work on novelty detection will be submitted as a conference paper.

Chapter 8

Acknowledgements

This thesis could not have been produced without the help of many researchers in the field. I want to thank Pieter Jonker for providing me the opportunity to work in his laboratory and the effort to engage me in all facets of his research group. I owe many thanks to my direct supervisor Aswin Chandarr for his input and ability to break up a full project in manageable weekly steps. I want to thank Maja Rudinac for her help in setting up this project and working out the relevant research goals. At the Mathematical Institute, I owe great thanks to my secondary supervisor Tim van Erven, for his overall commitment and ability to steer and reflect on the process of writing a thesis. Finally, halfway this work I received new input from David Tax, who's enthusiasm also sparked me. In general, my study period led me from a M.D. to the Robotics Institute, a road which's horizon was not always in full sight. I owe it most to the ever sincere support of my parents, that things worked out well.

August, 2015

Bibliography

[1] Jake K Aggarwal and Quin Cai. Human motion analysis: A review. In *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, pages 90–102. IEEE, 1997.

[2] Md. Atiqur Rahman Ahad, Joo Kooi Tan, Hyoungseop Kim, and Seiji Ishikawa. Motion history image: its variants and applications., 2012.

[3] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential Deep Learning for Human Action Recognition. In Albert Ali Salah and Bruno Lepri, editors, *HBU*, volume 7065 of *Lecture Notes in Computer Science*, pages 29–39. Springer, 2011.

[4] Lalit Bahl, Peter Brown, Peter V de Souza, and Robert Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, volume 11, pages 49–52. IEEE, 1986.

[5] Leonard E Baum, John Alonzo Eagon, et al. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc*, 73(3):360–363, 1967.

[6] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[8] Timothy J. Hazen, Stephanie Seneff, and Joseph Polifroni. Recognition confidence scoring and its use in speech understanding systems. *Computer Speech & Language*, 16(1):49–67, 2002.

[9] Xiaodong He, Li Deng, and Wu Chou. Discriminative learning in sequential pattern recognition. *Signal Processing Magazine, IEEE*, 25(5):14–36, September 2008.

[10] H. Jiang. Confidence measures for speech recognition: A survey. *Speech communication*, 45(4):455–470, 2005.

[11] Hui Jiang. Discriminative Training of HMMs for Automatic Speech Recognition: A Survey. *Comput. Speech Lang.*, 24(4):589–608, October 2010.

[12] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & psychophysics*, 14(2):201–211, 1973.

[13] Simo O. Kamppari and Timothy J. Hazen. Word and phone level acoustic confidence scoring. In *ICASSP*, pages 1799–1802. IEEE, 2000.

[14] Nationaal Kompas. Vergrijzing. `http://www.nationaalkompas.nl/bevolking/vergrijzing/toekomst/`.

[15] Cornell University Robot learning lab. Cornell Activity Datasets: CAD-60 & CAD-120. `http://pr.cs.cornell.edu/humanactivities/data.php`.

[16] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Expandable Data-Driven Graphical Modeling of Human Actions Based on Salient Postures. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11):1499–1510, Nov 2008.

[17] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3D points. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 9–14, June 2010.

[18] Microsoft Research; Zicheng Liu. Action 3D dataset. `http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc`.

[19] Montreal University machine learning lab. Research Vision. `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/ResearchVision`.

[20] Markos Markou and Sameer Singh. Novelty detection: a review–part 1: statistical approaches. *Signal Processing*, 83(12):2481–2497, December 2003.

[21] Mohammad M. Masud, Qing Chen, Latifur Khan, Charu C. Aggarwal, Jing Gao, Jiawei Han, Ashok N. Srivastava, and Nikunj C. Oza. Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams. *IEEE Trans. Knowl. Data Eng.*, 25(7):1484–1497, 2013.

[22] P. Matejka, L. Burget, P. Schwarz, and J. Cernocky. Brno University of Technology System for NIST 2005 Language Recognition Evaluation. In *Speaker and Language Recognition Workshop, 2006. IEEE Odyssey 2006: The*, pages 1–7, June 2006.

[23] Kevin P Murphy. HMM toolbox for Matlab. `http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`.

[24] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[25] Christoph Neukirchen and Gerhard Rigoll. Controlling the Complexity of HMM Systems by Regularization. pages 737–743, 1998.

[26] Sebastian Nowozin and Jamie Shotton. Action points: A representation for low-latency online human action recognition. *Microsoft Research Cambridge, Tech. Rep. MSR-TR-2012-68*, 2012.

[27] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal, and Ruzena Bajcsy. Sequence of the most informative joints (smij): A new representation for human skeletal action recognition. *Journal of Visual Communication and Image Representation*, 25(1):24–38, 2014.

[28] Ouderenfonds. Feiten en Cijfers. `https://www.ouderenfonds.nl/onze-organisatie/feiten-en-cijfers/`.

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

[30] Daniel Povey. *Discriminative training for large vocabulary speech recognition*. PhD thesis, University of Cambridge, 2005.

[31] Mazin G Rahim, Chin-Hui Lee, and Biing-Hwang Juang. Discriminative utterance verification for connected digits recognition. *Speech and Audio Processing, IEEE Transactions on*, 5(3):266–277, 1997.

[32] Michalis Raptis, Darko Kirovski, and Hugues Hoppe. Real-time Classification of Dance Gestures from Skeleton Animation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 147–156, New York, NY, USA, 2011. ACM.

[33] Thomas Schaaf and Thomas Kemp. Confidence measures for spontaneous speech recognition. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 875–878. IEEE, 1997.

[34] Ralf Schlüter and Wolfgang Macherey. Comparison of discriminative training criteria. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 1, pages 493–496. IEEE, 1998.

[35] Ralf Schlüter, Wolfgang Macherey, Stephan Kanthak, Hermann Ney, and Lutz Welling. Comparison of optimization methods for discriminative training criteria. volume 97, pages 15–18, 1997.

[36] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[37] Rafid Sukkar, Chin-Hui Lee, et al. Vocabulary independent discriminative utterance verification for nonkeyword rejection in subword based speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 4(6):420–429, 1996.

[38] Antonio W Vieira, Erickson R Nascimento, Gabriel L Oliveira, Zicheng Liu, and Mario FM Campos. Stop: Space-time occupancy patterns for 3d action recognition from depth map sequences. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 252–259. Springer, 2012.

[39] Jiang Wang, Zicheng Liu, Jan Chorowski, Zhuoyuan Chen, and Ying Wu. Robust 3d action recognition with random occupancy patterns. In *Computer Vision–ECCV 2012*, pages 872–885. Springer, 2012.

[40] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Learning actionlet ensemble for 3d human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(5):914–927, 2014.

[41] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241, 2 2011.

[42] Phil C Woodland. Speaker adaptation for continuous density HMMs: A review. In *ISCA Tutorial and Research Workshop (ITRW) on Adaptation Methods for Speech Recognition*, 2001.

[43] Philip C Woodland and Daniel Povey. Large scale discriminative training of hidden Markov models for speech recognition. *Computer Speech & Language*, 16(1):25–47, 2002.

[44] C. F. J. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

[45] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 20–27. IEEE, 2012.

[46] Xiaodong Yang and YingLi Tian. Effective 3d action recognition using eigenjoints. *Journal of Visual Communication and Image Representation*, 25(1):2–11, 2014.

[47] Xiaodong Yang, Chenyang Zhang, and YingLi Tian. Recognizing actions using depth motion maps-based histograms of oriented gradients. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1057–1060. ACM, 2012.

[48] Gang Yu, Zicheng Liu, and Junsong Yuan. Discriminative Orderlet Mining For Real-time Recognition of Human-Object Interaction. In *Asian Conference on Computer Vision*, 2014.