

# From the Articulated 3D Pose of the Human Body to Basic Action Recognition

Diploma Thesis  
of  
Tobias Zepf

31. May 2012

Supervisor: Prof. Dr.-Ing. Rainer Stiefelhagen<sup>1</sup>

Advisor: Dr. rer. nat. Wolfgang Hübner

Advisor: David Münch

Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung  
Gutleuthausstr. 1, 76275 Ettlingen

<sup>1</sup> Computer Vision for Human Computer Interaction Lab - Karlsruhe Institute of Technology



# Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt und keine anderen, als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ettlingen, den 31. Mai 2012

---

Tobias Zepf

## Zusammenfassung

Die Ableitung von Aussagen über den Inhalt eines Videos, in diesem Fall die Art der Bewegung von Personen, ist eine Herausforderung und ein großes Forschungsgebiet. Die wesentliche Herausforderung liegt in der Komplexität menschlicher Bewegung. Beispielsweise kann eine Person am Bahnhof rennen um einen Zug zu erreichen oder sie kann vor etwas flüchten. In dieser Arbeit wird eine neue Methode vorgestellt, um Aktionen von Menschen zu erkennen die auf dem *Implicit Shape Model* basiert. Dieses Modell versucht anhand von Trainingssequenzen, die kurze und möglichst charakteristische Ausschnitte von Aktionen enthalten, eine Aussage darüber zu treffen wann eine Trainingsaktion im Eingabevideo auftritt.

Als Eingabedaten werden aufgezeichnete Kooordinaten von dreidimensionalen Motion Capture Verfahren verwendet. Diese Daten werden zunächst auf ein geeignetes Personenmodell übertragen. In dieser Arbeit wird dabei ein menschliches Modell mit 15 Gelenken und 12 Körperteilen verwendet. Basierend auf den Stellungen der Körperteile zueinander können mit diesem Modell Winkel zwischen den einzelnen Körperelementen berechnet werden. Diese Winkelinformationen werden anschließend zu einem charakteristischen Aktionsdeskriptor zusammengefasst, der die Form einer Matrix hat. Jede Zeile entspricht dabei einer Aufnahme und in den Spalten stehen die Winkelwerte sowie deren Ableitungen. Solch eine Matrix wird sowohl für die zu klassifizierende Eingabesequenz als auch für jede Trainingssequenz erstellt.

Jede Zeile dieser Matrix, die für eine Aufnahme aus der Eingabesequenz steht, wird mit allen Zeilen der Matrizen von Trainingssequenzen verglichen und die am besten passenden mit einer Radiusuche gefunden. Ausgehend von diesen nächsten Nachbarn werden Votes für das Ende der Aktion bestimmt, indem eine Assoziation *Descriptor*  $\rightarrow$  *Aktion*  $\rightarrow$  *Aktionsende* gelernt wird. Unter all diesen Votes werden Maxima mit dem Mean Shift Verfahren gesucht. Diese Maxima werden am Ende zu einer finalen Aussage über die aktuelle Aktion der Person im Video verarbeitet. Das Ziel dieser Arbeit ist dabei die Erkennung von Standardbewegungen wie Laufen, Springen und Rennen.

Das Ergebnis der Arbeit zeigt, das eine Erkennung von solchen Aktionen möglich ist. In der Evaluierung wurden zunächst Ergebnisse verglichen, um geeignete Parametereinstellungen zu finden. Mit Hilfe dieser Parameter wurden dann Versuche durchgeführt, die eine erfolgreiche und stabile Erkennung von verschiedenen Aktionen zeigen.



## Abstract

Analyzing video data and derivate statements about the actions of humans in videos is a very challenging task. The main difficulty is the complexity of human motions. In this work, a new approach for the process of action recognition, based on the *Implicit Shape Model*, is presented and evaluated. The *Implicit Shape Model*, a generalized Hough-transformation, is descend from the appearance-based object recognition. The aim of this work is the transfer of this method onto the problem of action recognition. For the characteristic description of actions, video data from 3D recording systems is transformed to a human body model from which angles between the human limbs are calculated. By calculating the angular data for each time step, we can express this information in a matrix where one row corresponds with one time step. Both the input and sample action sequences are transformed to such an action descriptor. The sample sequences represent action snippets like one step in walking or running actions. Correspondences between the input sequence and the sample sets are then searched with a radius search. This search takes one time step of the input sequence and finds the most similar time step in all sample sequences. With this information, offsets to the estimated actions represented in the sample set can be calculated. Votes for every estimated endpoint are generated and within these votes, maxima can be searched with Mean Shift mode seeking. The strongest of these maxima are then taken to output a final statement of the behavior of the human subject. The aim of this approach is the ability to detect basic human behaviors like walking and waving. The evaluation shows, that the approach is working robustly and the recognition of actions like walking is possible.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem Definition and Aim of this Thesis . . . . .	3
1.2	Thesis Outline . . . . .	3
<b>2</b>	<b>Related Work</b>	<b>4</b>
2.1	Temporal and Spatial Action Representations . . . . .	4
2.1.1	Spatial Action Representation . . . . .	4
2.1.2	Temporal Action Representation . . . . .	10
2.1.3	Segmentation . . . . .	13
2.1.4	View Indepence . . . . .	15
2.2	Human Sequence Evaluation: the Key-Frame Approach . . . . .	18
2.3	Implicit Shape Model based Action Recognition Methods . . . . .	19
2.4	Summary . . . . .	20
<b>3</b>	<b>Basics of Human Motion Modeling and Action Recognition</b>	<b>21</b>
3.1	Human Body Model . . . . .	21
3.1.1	Stick Figure Model . . . . .	22
3.1.2	From Motion Capture Data to Relative Angles . . . . .	28
3.1.3	Derivation of Relative Angles with Numerical Methods . . . . .	31
3.2	Implicit Shape Model based Action Recognition . . . . .	32
3.2.1	ISM - a Brief Overview . . . . .	33
3.2.2	Adaptions to the ISM . . . . .	34
3.2.3	Building a Descriptor for Action Recognition . . . . .	35
3.2.4	Codebook Entries and Voting . . . . .	36
3.2.5	Approximate Nearest Neighbor Search . . . . .	36
3.2.6	Mean Shift Mode Seeking . . . . .	39
3.2.7	Determination of Actions from Mean Shift Mode Seeking . . . . .	41
3.3	Summary . . . . .	42
<b>4</b>	<b>Implementation</b>	<b>43</b>
4.1	The Action Recognition Framework . . . . .	43
4.1.1	Training Phase . . . . .	43
4.1.2	Recognition Phase . . . . .	44

4.2	Steps Towards a Real-time Application . . . . .	46
4.3	Performance . . . . .	47
4.4	Summary . . . . .	47
<b>5</b>	<b>Experiments</b>	<b>48</b>
5.1	Parameter Settings . . . . .	48
5.1.1	Number of Trees in FLANN Search . . . . .	48
5.1.2	Number of Leafs to Visit when a Nearest Neighbor is Searched with FLANN . . . . .	50
5.1.3	Threshold for FLANN Results . . . . .	52
5.1.4	Approximative versus Exact FLANN Results . . . . .	54
5.1.5	Mean Shift Window Size . . . . .	55
5.1.6	Threshold for Getting the Action Statement . . . . .	58
5.2	Dictionary with Different Actions . . . . .	58
5.3	Influence of Body Parts . . . . .	59
5.4	Influence of the Derivations of Relative Angles in the Descriptor . . . . .	61
5.5	Robustness . . . . .	62
5.5.1	Results for a Walk Action . . . . .	62
5.5.2	Results for a Jump Action . . . . .	62
5.6	Summary . . . . .	62
<b>6</b>	<b>Conclusion</b>	<b>70</b>
6.1	Summary and Results . . . . .	70
6.2	Future Work . . . . .	71
	<b>List of Figures</b>	<b>76</b>
	<b>Bibliography</b>	<b>82</b>
	<b>A Graphs</b>	<b>83</b>
	<b>B Libraries</b>	<b>89</b>
B.1	Qt . . . . .	89
B.2	OpenCV . . . . .	90
B.3	OpenGL for Rendering of 3D Poses . . . . .	91
B.4	Fast Library for Approximate Nearest Neighbors . . . . .	92
	<b>C Timetable</b>	<b>94</b>

# Chapter 1

## Introduction

When video content has to be analyzed, the main task is to answer the questions what, where, when, how and why is something happening. The result should be understandable for humans and include semantic information. In person centric video analysis the answer should contain the action of a person, e.g. Person A is walking from X to Y. This can be achieved by analyzing the motion of a person on a formal basis. Such a basis was introduced first by Johansson ([Johansson 1973](#)). Following Johannsons claims, actions should be derived from the 3D pose of persons recorded over time.

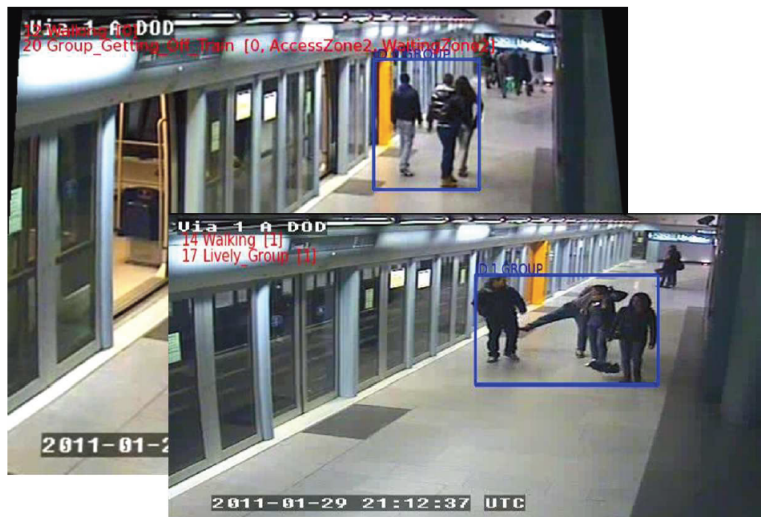


Figure 1.1: Example output of an automatic video surveillance system which is able to detect groups of persons and deduce their behaviors ([Zaidenberg et al. 2011](#)).

The visual analysis of human behavior is a challenging task. Machines which are able to understand and interpret human behavior would lead to major improvements in various applications like video surveillance, video indexing and searching, or robotics.

Especially in video surveillance applications there is a great need for automated video content analysis because human operators in front of many screens are not able to detect all objects and situations of interest. This human based surveillance is also very cost-intensive and in most scenarios, where many cameras are placed, these systems can be seen as passive systems where video data is analyzed after an event took place and often the videos are never watched. One can imagine, that an automated computer-based surveillance system, which is able to detect and recognize persons (as shown in Figure 1.1) would be a great step forward. If this system also is able to gain knowledge of certain actions and sending alerts if a suspicious object or human is found, it would have a huge market potential and therefore this topic is a very attractive field of research.

Another major application of analyzing video content arises with internet portals as YouTube<sup>1</sup> or Vimeo<sup>2</sup>. On these websites, users share huge amounts of videos among each other. All videos have tags and titles which were set by the uploader to allow searching for specific videos. But this tagging and indexing is error prone and imprecise. The ability of searching for a certain type of action automatically without adding additional information to a video would make indexing and searching in these databases more efficient and implicates enormous potential for new applications.

Understanding human behavior is very complex due to the nature of human movements. These movements are often ambiguous and context sensitive. For example in a station a person might be running to catch the train or also because of fleeing from something. One of the problems for action recognition are time differences in the execution of actions leading to segmentation problems, for example the same waving behavior can be performed very fast or very slow. Another challenge is that different actions like running or walking are very similar and seem difficult to distinguish. Despite specialized sensors, e.g. Microsoft Kinect<sup>3</sup> or multi-camera systems, most available videos are recorded under uncontrolled conditions with monocular sensor setups. This leads to side effects like occlusions and the loss of information. All of these tasks lead to the need of incorporating the uncertainty and complexity of human behavior during the design of a system which is able to recognize actions.

An action recognition system based on visual input can be divided into three essential steps (Gong & Xiang 2011). These are

1. **Representation and modeling** - Encode the information from the input.
2. **Detection and classification** - Searching for characteristics which allow the classification of behaviors.
3. **Prediction and association** - Being able to predict future events based on the information from past and current human motion data.

---

<sup>1</sup><http://www.youtube.com>

<sup>2</sup><http://www.vimeo.com>

<sup>3</sup><http://www.xbox.com/kinect>

In the following chapters an action recognition process based on the 3D pose of the human body is developed. All tasks named above can be found in this work including the selection of an appropriate body model, an action descriptor and the association between learned behavior examples and the current input data from videos. In the next section, a more detailed view on the definition and the aim of this work is given.

## 1.1 Problem Definition and Aim of this Thesis

In this thesis, an approach of basic human action recognition based on 3D input data of a human pose is designed and evaluated. Actions should be recognized from real data like motion capture methods. The first step is to get the 3D pose of the human body for every video frame. Therefore the motion capture data has to be transferred to an appropriate human model which is easy to compute. The proposed action recognition method is based on angles instead of absolute 3D positions, and these angles must represent continuous trajectories over time. With these information, an action descriptor can be built. By comparing each observed input descriptor with descriptors from learned and labeled motion sequences, a statement of what the subject is doing can be made. Throughout this thesis, the training sequences are snippets of actions and no clustering or vector quantization methods are used due to simplification reasons. The first task of the action recognition approach is to find correspondences between the input sequence and the trained action sequences with a radius search. These correspondences should generate votes and such votes are the main part of an *Implicit Shape Model* based action recognition process. On these votes, maxima can be searched with Mean Shift Mode seeking and then in the final step the most probable behavior of the human in the input video should be returned.

## 1.2 Thesis Outline

The following chapter gives an overview of recent and past studies in this wide field of research. Several approaches for the formal representation of actions are described and solutions to the problems described in Chapter 1 are presented. Chapter 3 deals with the basics of human motion modeling and action recognition. The theoretical and mathematical background of the human body model used in this work is explained as well as the input video format and the calculation of angles from the body model. Also, the most important step which is the action recognition approach based on the Implicit Shape Model is described in detail.

In the fourth chapter, an overview and details of the implementation are given including code snippets from all important parts. This comprehensive action recognition framework is then evaluated on several sample data sets in Chapter 5. Chapter 6 contains the conclusion and gives an outlook for future work.

# Chapter 2

## Related Work

This chapter gives a brief overview of existing methods for the process of action representation and recognition. As this topic is an active research field many different approaches exist, covering the demands of various applications, e.g. video surveillance systems or robotics. Therefore only a small extract of relevant related work can be given in this section. Action recognition is the genus of several techniques: feature extraction, action learning, action segmentation and action classification. All of these are based on video data and an action model database (Weinland et al. 2011). The result of these interacting parts should be a label for a specific action which can be interpreted by humans. This chapter gives an overview of recognition methods based on different representations of the temporal and spatial structure of actions. Further, approaches for action segmentation are presented and finally solutions for the view-invariance problem are mentioned.

### 2.1 Temporal and Spatial Action Representations

The difference between temporal and spatial action representations is the way how action recognition is done. In the spatial domain action recognition can be based on global image features which describe an image with only one vector (Lisin et al. 2005). Alternatives are the usage of image features like edges and corners or statistical models which describe the spatial distribution of local image features. Within the temporal domain, recognition is either based on global temporal signatures representing an action from the beginning to the end, on grammatical models which describe the action sequentially, or on statistical models which i.e. describe distributions of observations over time (Weinland et al. 2011). In the following two subsections some related methods are presented.

#### 2.1.1 Spatial Action Representation

A further distinction between spatial representation methods can be made by separating the approaches into body models, image models and local statistics. Body models



represent the whole human body in contrast to image models, where only regions of high interest from the body can be investigated. The third approach is independent from body parts or image coordinates and uses small image regions which are described by local features of an image. Some of these methods are described in the following sections.

### 2.1.1.1 Body Models

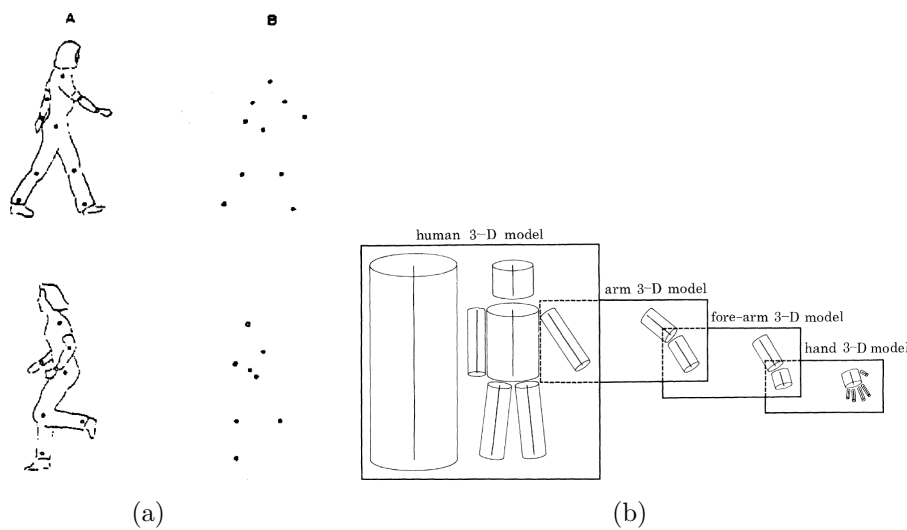


Figure 2.1: (a): Action recognition based on few light bulbs as dots by (Johansson 1973). (b): 3D Body model based on cylinders proposed by (Marr & Nishihara 1978).

When using a body model, a human pose is tried to be estimated. The pose can be used for action recognition and is based either on direct 2D body models generated from input data or on reconstructed 3D body models. Early work in this area was done by (Johansson 1973). He discovered that human actions can be recognized with information only about few characteristic points. Figure 2.1(a) gives an example of a walking and running subject. These points are also used in actual Motion Capture techniques. (Marr & Nishihara 1978) then introduced a 3D body model based on cylindrical primitives including a hierarchy which allows one to define the granularity as shown in Figure 2.1(b). It is possible to model one arm as a single cylinder as well as all human limbs including fingers. This model was adopted and extended (Rohr 1994) and other models based on quadrics (Gavrila & Davis 1995) and splines were also proposed. Motion capture techniques which are based on 3D coordinates, were used by (Campbell & Bobick 1995) to perform action recognition based on patterns for waving and twisting actions.

In contrast to 3D models several approaches using 2D data exist. Simple 2D stick

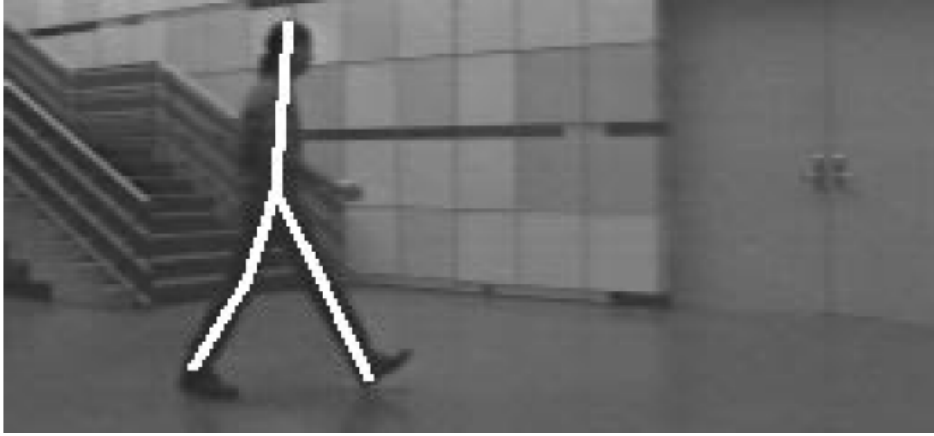


Figure 2.2: 2D stick figure generated by line fitting to a silhouette (Niyogi & Adelson 1994).

figures as shown in Figure 2.2 are used by (Niyogi & Adelson 1994) as well as tracked hand or head trajectories, where the 2D coordinates of these body parts are tracked over time (Brand et al. 1997)

#### 2.1.1.2 Image Models

The advantage of using image models instead of body models is that they are more efficient and simpler in most cases. Global, image based representations, which are also called *holistic representations* are based on features within a specific region of interest (ROI). Due to their simplicity they can be computed fast and efficient, while being as discriminative as body models (Weinland et al. 2011). One possibility of using an image model is to correlate the images directly without previous feature extraction. In (Darrell & Pentland 1993) this was done for some hand gestures with a static background by comparing the recorded image to existing training data.

Silhouette and contour based image models are also often used due to the fact that silhouettes as shown in Figure 2.3 and contours are easy to compute (Wang & Suter 2007). Two methods for modeling human motion have been used with silhouettes. Quantized silhouette images as features together with *Hidden Markov Models* (HMM) were used by (Yamato et al. 1992). Due to the assumption of independence between the states in HMM models which makes overlapping features or long-range dependencies difficult, (Wang & Suter 2007) uses *Factorial Conditional Random Field* (FCRF) based motion modeling. Also the problem with noisy backgrounds where silhouette extraction gets difficult was addressed by (Weinland & Boyer 2008) using the Chamfer distance (Gavrila & Philomin 1999).

Another important class of image models is based on the optical flow. (Cutler & Turk 1998) defines ‘motion blobs’ of the optical flow, as shown in Figure 2.4(a) and action recognition in this work is based on the number, motion, size and relative positions

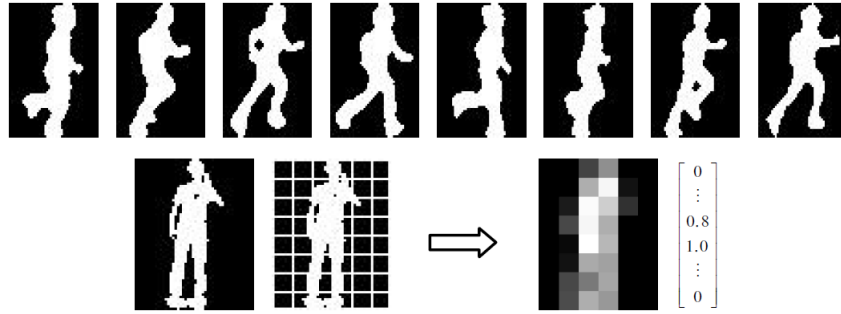


Figure 2.3: Silhouettes of a running person (top) and block-based feature representation which is used as a descriptor (Wang & Suter 2007).

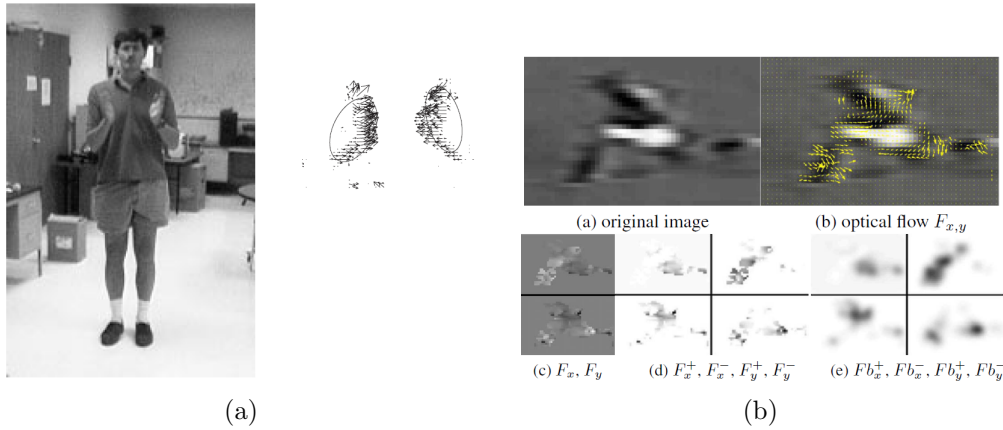


Figure 2.4: (a): The optical flow of a clapping action. The area where the main action takes place is marked as a blob (Cutler & Turk 1998). (b): Optical flow of a far away football player divided into its  $x$  and  $y$  components and then into four non-negative channels (Efros et al. 2003).

of these blobs. In (Efros et al. 2003) the flow field is separated into the horizontal and vertical optical flow components. These two fields are then half-wave rectified and blurred which gives the motion descriptor as it can be seen in Figure 2.4(b). The main advantage of optical flow based methods is that they are independent from background subtraction but they rely on the assumption that differences of images are movement and lighting and material changes are also detected as motion (Weinland et al. 2011). Our last example of image models which is described here are gradient based approaches. One possibility is to calculate gradient fields the spatio-temporal  $(x, y, t)$  direction and then build a histogram (Zelnik-Manor & Irani 2001) of these gradient fields. These distributions are then used as descriptors (see Figure 2.5). An alternative to gradient fields are the more robust *Histogram of oriented Gradients* (HOG) which are

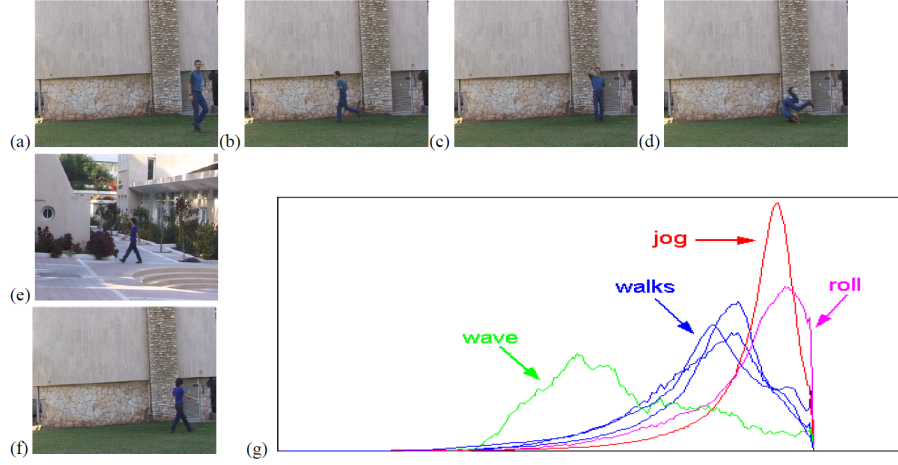


Figure 2.5: Distributions of gradients of different motion patterns (a) to (f) displayed as a smoothed histogram (g) (Zelnik-Manor & Irani 2001).

locally normalized gradient histograms computed for overlapping image blocks (Thurau & Hlavac 2008).

To summarize this chapter we can say that image models are simpler than body models but sensitive to changes in view directions, body sizes and rely on strong assumptions (Weinland et al. 2011).

### 2.1.1.3 Spatial Statistics

Action recognition can also be based on local features which are not body parts or image coordinates. If an image is divided into small regions and local features are calculated for every region, action recognition can be done using the statistics of these arbitrary features. As a result, these methods are independent of detecting and localizing whole humans or whole parts of humans because the video is decomposed in regions which are not linked to such objects (Weinland et al. 2011).

One very popular approach was proposed by (Laptev & Lindeberg 2003). He extended interest point detection in 2D  $(x, y)$  space, introduced by Harris, to 3D  $(x, y, t)$ . Interest points are spatio-temporal corners and can be used as features for object recognition, because they are characteristic and comparable. Figure 2.6 shows the result of interest point detection of waving action. With such an approach it is possible to detect walking people and perform pose estimation. Another spatio-temporal interest point based approach is based on 1D Gabor filters (Dollar et al. 2005). Also the very popular SIFT descriptor (Lowe 2004) was used for action recognition by searching nearest neighbors for the descriptors in training data and using a Hough transformation to identify clusters agreeing on the object pose.

All presented methods above have in common that the detected features are unordered and structureless. Therefore (Gilbert et al. 2008) proposes *compound features* which

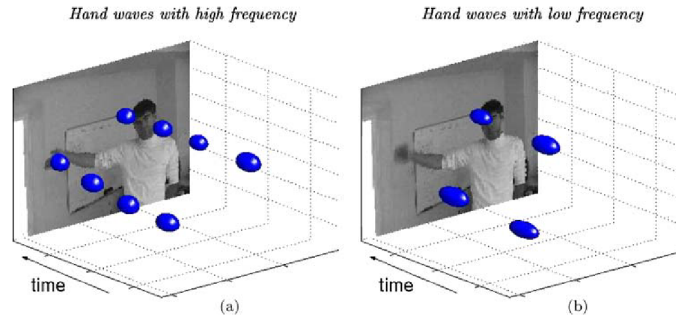


Figure 2.6: Local space time interest points of waving at different frequencies (Laptev & Lindeberg 2003).

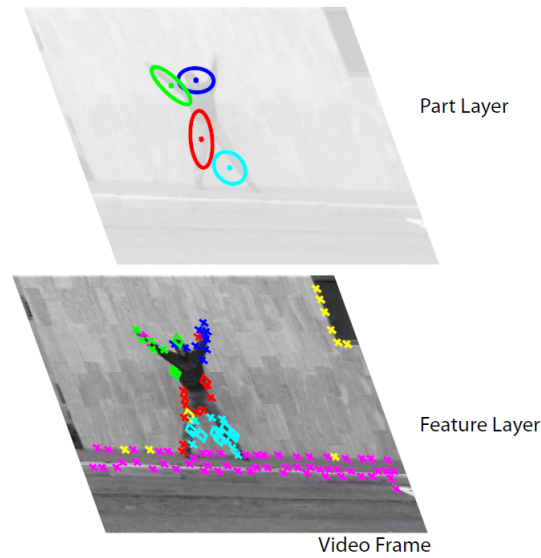


Figure 2.7: Adding structure to local features by introducing a part layer (Niebles & Fei-Fei 2007).

are in fact a set of features. Every local feature is grouped spatially to form such a set. Another possibility for structured local features is shown in Figure 2.7. The part layer introduced in (Niebles & Fei-Fei 2007) combines a large number of features and allows capturing similar body configurations or poses.

The advantage of statistical methods based on local features is their ability to adapt to difficult scenes. No complex body or image modeling is needed in advance. In (Weinland et al. 2011) however it is stated that the complexity of human actions will make it necessary to combine statistical methods with spatial and temporal models. Such combinations of spatial statistics and temporal models are presented in the next

section.

## 2.1.2 Temporal Action Representation

Despite the spatial structure of an action as introduced in Section 2.1.1, the temporal structure of an action is also a very important part in the action recognition processes. The representation of temporal structures can be divided into three main categories (Weinland et al. 2011): Action grammars which group features into categories and define transitions between them, action templates which complete temporal blocks of actions are learned, and temporal statistics which build statistical models of the appearance of actions without dynamics. In the following several approaches for these three categories are presented.

### 2.1.2.1 Action Grammars

*Hidden Markov Models* (HMM) are one of the most popular probabilistic graphical models. For example a silhouette based approach as described in Section 2.1.1.2 is combined with HMMs in (Yamato et al. 1992) to represent the sequences of silhouettes. Another example among many others for the usage of HMMs is (Starner & Pentland 1995) where American Sign Language is recognized.



Figure 2.8: Two person interaction and tracked body parts which can be analyzed with Bayesian Networks (Park & Aggarwal 2003).

Due to the fact that HMMs are sequential they lack the ability to model parallel and independent movements in body parts. With the use of *Dynamic Bayesian Networks* these limitations can be overcome. In (Park & Aggarwal 2003) such a model was used for the recognition of interactions between two persons by analyzing the evolution of poses of multiple body parts over time. An example is given in Figure 2.8. The body parts are then put into a tree structure and then concatenated in pairs for representing each of the two persons. Action recognition is then done with a Bayesian network. Another method to use features which are not independent are *Conditional Random Field* (CRF) based approaches as described in Section 2.1.1.2.

With action grammars one can reach very high modularity due to the properties of the used models. HMMs for example allow high granularity and Dynamic Bayesian

Networks are compositional. Therefore action grammars can handle large variations in action speeds. On the downside, learning and evaluation processes are still difficult when there are many action classes (Weinland et al. 2011).

### 2.1.2.2 Action Templates

Instead of using a layered and modular model as described in Section 2.1.2.1 a lot of work was investigated in action templates. Such a template describes a whole block of features for a sequence which is very long in contrast to previous described methods as the optical flow.

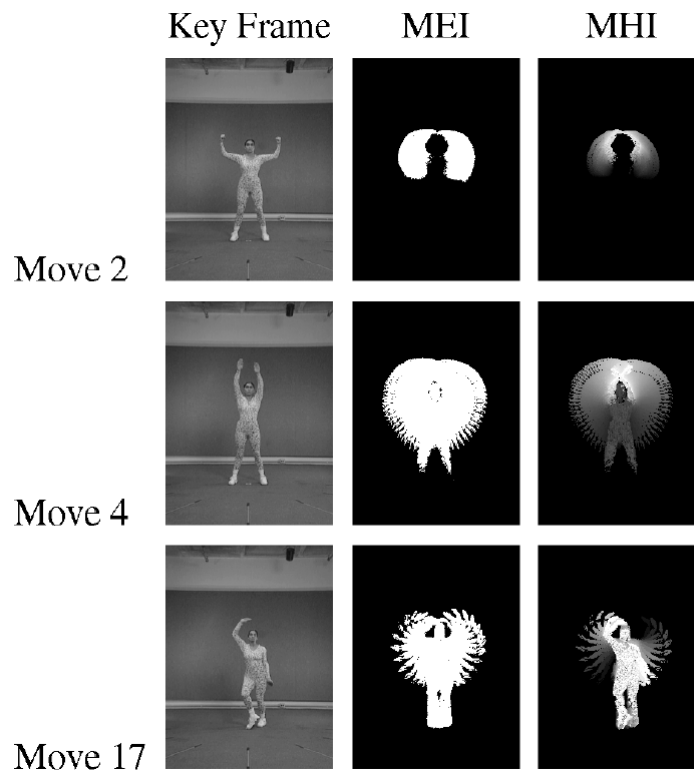


Figure 2.9: Bobick and Davis uses two images as action templates: The binary Motion Energy image (MEI) and the scalar valued Motion History Image (MHI) (Bobick & Davis 2001).

One very popular work on action templates was done by Bobick and Davis (Bobick & Davis 2001). The use two images for the representation of the history of an action as shown in Figure 2.9. One is the Motion Energy Image (MEI) which is a binary image that shows the areas where motion takes place are white. This image answers the question ‘Where is motion’. The second question ‘How does motion moves the image’ is answered by Motion History Images (MHI) which pixels represent the history of



movements. More recent movement areas are brighter than older ones. This concept of 2D action templates was extended to 3D Motion History Volumes (MHV) in (Weinland et al. 2006).

The main disadvantage of such template models is their inability to cover changes in speed and time. This can be resolved by multiple training sets, but this is not the best solution. Solutions to this problem are learning methods including neural networks (Guo et al. 1994) or normalizing the duration with dynamic time warping (DTW) as used in (Darrell & Pentland 1993).

### 2.1.2.3 Temporal Statistics

Another promising research direction are statistical models without an explicit model of dynamics (Weinland et al. 2011). This allows recognizing actions from still images. In (Carlsson & Sullivan 2001) this was successfully done for fore- and backhand strokes of tennis players. In that work, one *keyframe* for each action was defined and matched to frames from the input sequence.

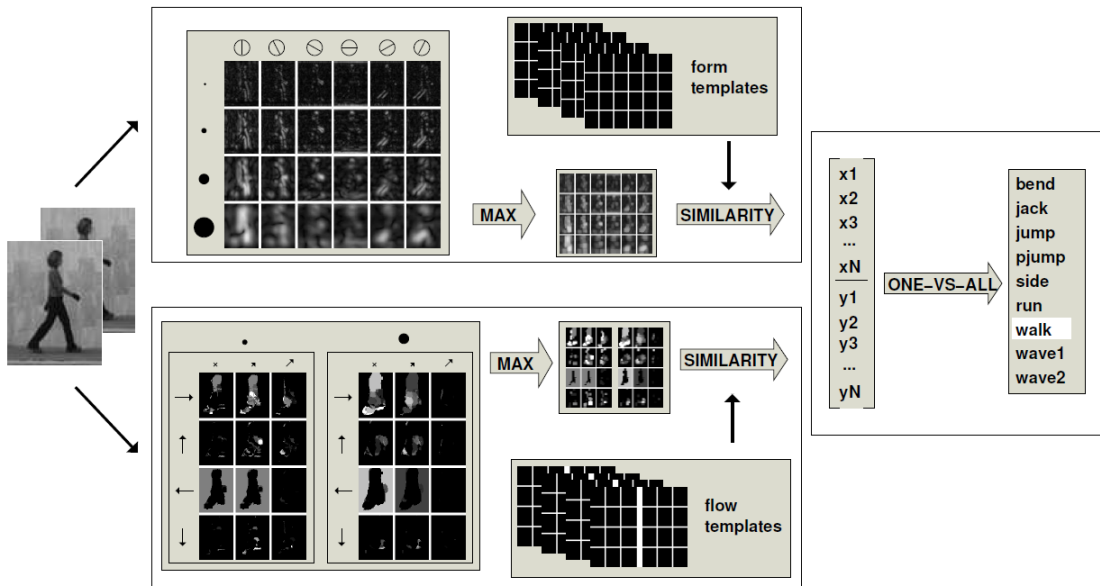


Figure 2.10: Action recognition process based on snippets (Schindler & van Gool 2008). This Figure shows the process of matching results from 1D Gabor filtering (upper half) and optical flow (lower half) to form and flow templates for a final action statement.

That keyframe based method was extended in (Schindler & van Gool 2008) to *snippets*, which are sequences of maximum 10 frames length. In this work it is stated that such *snippets* are sufficient to recognize actions. Figure 2.10 shows the recognition process. From the input sequence two descriptors are calculated. The upper half which is called



‘form pathway’ 1D Gabor filters are applied onto the detected features and matched to learned templates. In the lower half which is called ‘motion pathway’ optical flow extracted from different scales and speeds also matched to template data. MAX in the picture means MAX-pooling, i.e. that only the best features are taken into account to reduce complexity. Both similarity scores, which are the result of the template matching, are then concatenated and finally classified with a one-versus-all support vector machine.

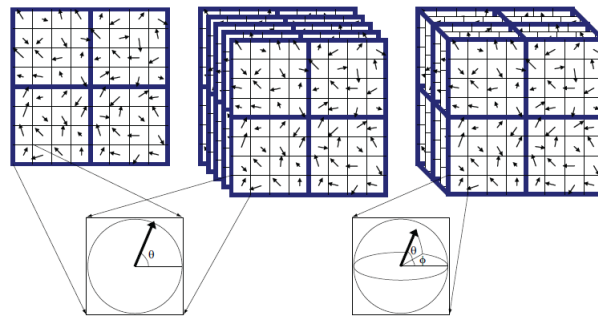


Figure 2.11: Different types of SIFT descriptors. From left to right: original 2D SIFT descriptor, 2D SIFT for videos, 3D SIFT descriptor (Scovanner et al. 2007).

Video sequences can be also described with histograms of feature occurrence over time. This can be done with a 3D SIFT descriptor as shown in Figure 2.11 which returns vectorized histograms (Scovanner et al. 2007). In general, action recognition with temporal statistics can be implemented very efficient and is very discriminative. One the other hand, not every action can be described with a model of the dynamics like two actions with the same poses but different temporal order. (Weinland et al. 2011).

### 2.1.3 Segmentation

Up to now, almost all mentioned techniques for action recognition work on short clips which include only one action. A video normally contains different kinds of actions which have to be separated. In general, this is almost the same problem as action recognition itself. The approaches for action segmentation can be classified into three main categories (Weinland et al. 2011) which are presented in the following sections.

#### 2.1.3.1 Boundary Detection

A common approach for segmentation is to divide the video at boundaries as special values in acceleration, curvature, and velocity of motions. The aim is to achieve segmentation at a lower processing level than action recognition. In (Marr & Vaina 1980) movements are segmented into *motion segments* and static rest states defined with

the body model in (Marr & Nishihara 1978). These static states, which can be local minima, can be interpreted as transitions between movements.

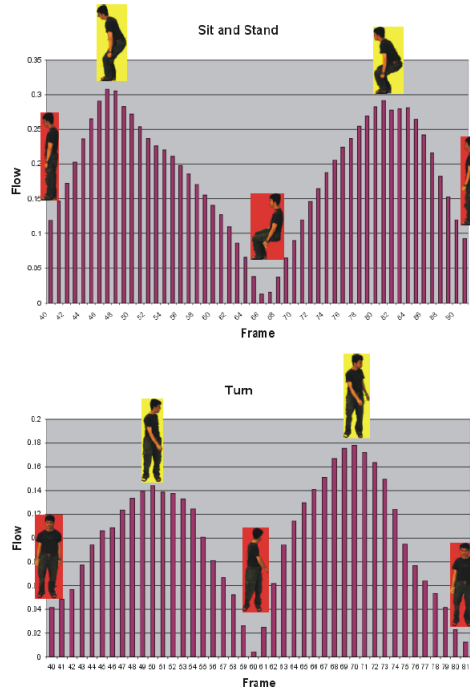


Figure 2.12: The vertical axis shows values of the characteristic difference between the optical flow and the mean optical flow of foreground pixels which can be taken into account for the segmentation of movements (Ogale et al. 2004).

Another possibility for boundary detection is to detect minima and maxima of the optical flow by comparing the optical flow of a foreground pixel which lies within the body with the mean optical flow at the same position, as shown in Figure 2.12. If that value is near to zero, an extreme body pose is indicated. Maximum values can be interpreted as large movements in all areas (Ogale et al. 2004).

All segmentation methods using boundary detection have in common that they have problems with simultaneous movements.

### 2.1.3.2 Sliding Windows

A very straightforward approach are sliding windows. Videos are divided into overlapping segments and the classification process is done on each of these segments. Then the most likely movement is assumed at the maximum of the cumulated classification scores. As a consequence the segmentation process relies on the results of the recognition process and is therefore inapplicable in the training stage. It is also expensive to

compute. But if a classification process is already implemented, these methods can be easily integrated (Weinland et al. 2011). Approaches based on sliding windows can be used with templates (Ke et al. 2005, Darrell & Pentland 1993) or grammars ((Wilson & Bobick 1999)).

### 2.1.3.3 Grammar Concatenation

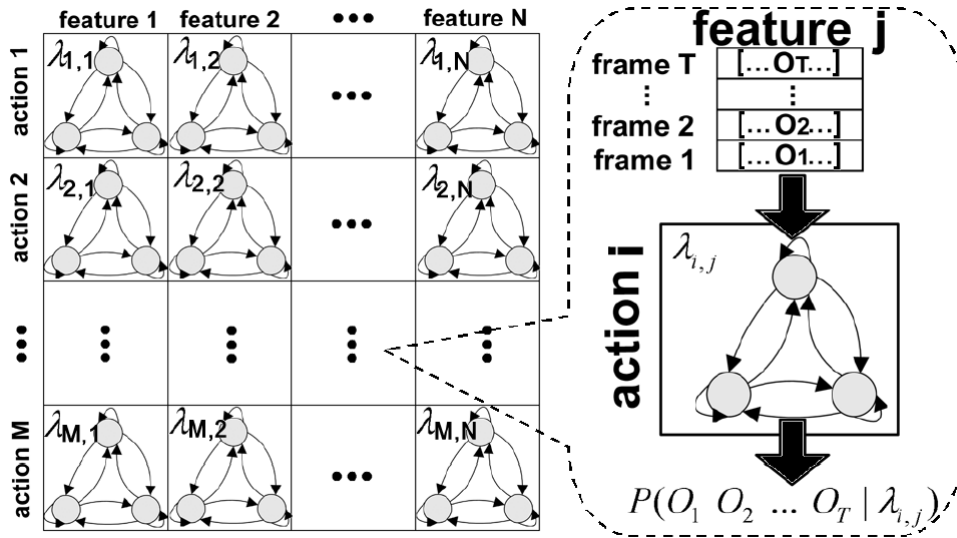


Figure 2.13: Network of HMMs used for action recognition. One HMM with 3 hidden states exist for every action and every feature. When the parameters of each HMM are learned, the most probable action sequence can be computed with the forward algorithm (Lv & Nevatia 2006).

In Section 2.1.2.1 the temporal representation of actions with grammars was described. Action grammars can be extended by making them also applicable to allow transitions between different actions itself instead of only different states within an action. Also, different actions can be concatenated. This leads to complex structures with similarities to HMMs and therefore efficient algorithms like Viterbi for finding the most likely sequence of states are applicable. Examples for the usage of such structures can be found in (Brand & Kettner 2000) or in (Lv & Nevatia 2006).

### 2.1.4 View Indepence

A desirable feature of an action recognition process is to be independent of the viewing angle. Strategies to achieve this aim can be classified into normalization, invariance and exhaustive search. In the following sections some examples for these strategies in 2D and 3D space are presented.

#### 2.1.4.1 Normalization

Normalization means that all results are transferred into a common coordinate frame to achieve comparable results. This is normally done by estimating the difference to the common frame and diminishing it. In 2D images this is often done by extracting a rectangular around the object followed by scaling and orientation as described in Section 2.1.1.2. Another possibility in 2D images is to estimate the 3D position of a person from its walking direction with knowledge of the camera calibration (Rogez et al. 2006).

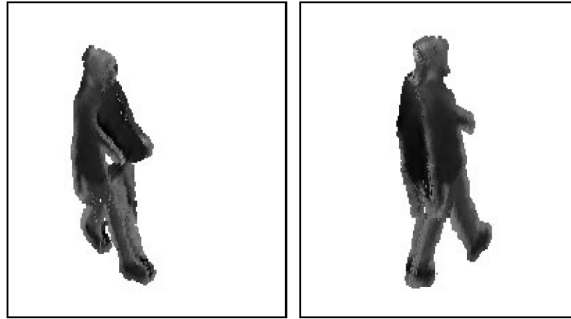


Figure 2.14: Generated new views from three completely different real camera views (Bodor et al. 2003).

In 3D the walking orientation was also used for orientation normalization. One example can be found in the work from (Bodor et al. 2003). As shown in Figure 2.14 this method creates new views out of real camera views. These views can be either used to generate orthogonal views for every input or to generate new views for additional training data. In general, these methods work well if the walking direction is available or other cues as a full body model can be computed (Weinland et al. 2011).

#### 2.1.4.2 Invariance

In contrast to methods which rely on normalization, view-invariant methods remove the view dependent information during the feature computation. In the 2D space, this can be done by regarding the frequency of occurrences instead of image features. This histogram based method was used by (Zelnik-Manor & Irani 2001) and described in Section 2.1.1.2.

In 3D, view invariance can be achieved with a different approach. In (Weinland et al. 2006) Motion History Volumes are used as action descriptors. Fourier-magnitudes and cylindrical coordinates are used to encode motion templates with invariant rotations around the z-axis.

### 2.1.4.3 Exhaustive Search

The third possibility for view invariant action recognition is exhaustive search. Instead of choosing a transformation for normalization or relying on invariant features it is possible to search over all transformations. This can be done either for 2D or 3D views (Weinland et al. 2011).

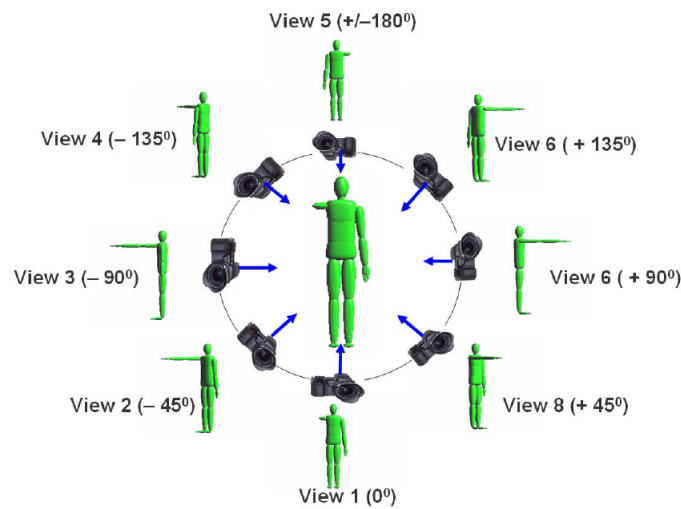


Figure 2.15: Generation of multiple views for training data. The exhaustive search is performed on such a dataset to find the best match (Ahmad & Lee 2006).

In 2D examples for exhaustive search methods can be found in (Ahmad & Lee 2006) where 8 cameras are used for recording the training data as shown in Figure 2.15. Then the input view is matched against all training views and then the best match is extracted with exhaustive search. In the work on Motion History Images (see Section 2.1.2.2) 7 cameras are used for recording and the training data. The recognition process takes the input of 2 cameras. Then the best match between multiple view represented by 90 degree rotations is determined (Bobick & Davis 2001).

The use of an internal 3D model with explicit variables for 3D position and orientation brings more flexibility to changes in the camera set-up (Weinland et al. 2011). Given the camera parameters all possible 2D view observations can be computed out of the 3D data. On these dataset the best match to the input 2D observation can be found with exhaustive search. An example is given in Figure 2.16, where image silhouettes generated from a 2D view are matched to silhouettes generated out of the 3D training data.

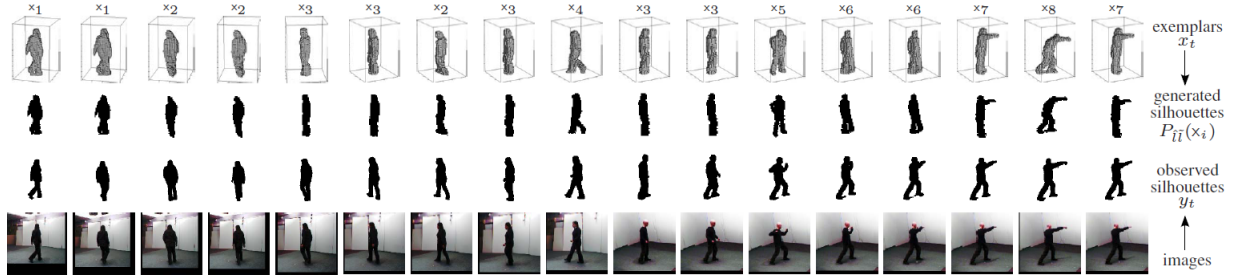


Figure 2.16: Examples for an exhaustive search method to match an observed silhouette to silhouettes generated from a 3D prototype. In this figure the best match is shown in the middle row (Weinland et al. 2007).

## 2.2 Human Sequence Evaluation: the Key-Frame Approach

In the work of Jordi González (González 2004) the problem of transforming image data into conceptual descriptors is addressed. The whole system is called *Human Sequence System*.

In the first part of that work, a human body model is proposed. The body model is used to describe human actions. This model is based on a stick figure with 15 joints and 12 limbs and is similar to the model used in later parts of this work. Procedures for acquiring data from 3D motion capture techniques and transferring them onto the body model are proposed. Furthermore different ways of describing the angles between the limbs are discussed.

These angles are then transformed into a pose descriptor. Such descriptors can be extracted from several training sequences in order to learn different human movements. The training actions are projected into the *aSpace* by Principle Component Analysis. This step reduces the dimensionality. The advantage of the *aSpace* is that the distance of the actions can be calculated faster. In the work from (González 2004) nine actions like sit, run, and walk are examined. For generating the *aSpace*, the required dimensionality has to be determined first. This is done by calculating mean postures and a covariance matrix of these. The dimensionality is then determined by eigenvalues of the covariance matrix. These eigenvalues express the mode of variation of a pose during an action. Then, each sample is projected into the *aSpace* and a mean manifold is determined. Some *key-frames* are selected out of the mean manifold by applying a distance function and finally, the prototypical action representation is an interpolation between these *key-frames*.

The so called *key-frames* are searched in order to reduce the number of frames which must be compared. Therefore, characteristic frames for an action are needed and this is done by determining frames which are the least likely body postures during an action. Examples for such frames are shown in Figure 2.17.

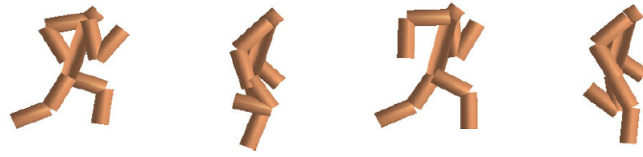


Figure 2.17: Detected key frames for a walk action. (González 2004).

By comparing such key frames from the training data with the input data from a video, human action recognition can be performed. Other applications are the synthesizing of virtual sequences which reproduce an action and comparisons of different human actions like walking of a male and female.

## 2.3 Implicit Shape Model based Action Recognition Methods



Figure 2.18: Pipeline for action recognition using the Implicit Shape Model (Thi et al. 2010).

The Implicit Shape Model (ISM) (Leibe et al. 2008) is a method for detecting objects in images and it is explained in Section 3.2 in detail. It has been adapted for the recognition of actions in (Thi et al. 2010). Instead of 2D interest points like Harris Corners, Space Time Interest Points (STIP) are searched to detect points with high motion change and also a region around this points is taken into account. Some of these points are affected by noise from the background and therefore a Sparse Bayesian Machine is incorporated to allow the distinction between relevant and background points. In contrast to the original ISM where the object center is detected, *local action centers* are used for the voting process in the approach from (Thi et al. 2010). These centers are found by decomposing videos into temporal slices or key frames and look for *local action centers* in each of them. The *global action centers* can be determined by searching the centroid of all *local action centers* along their trajectory. In that voting space a Mean Shift search is used to find the most possible action center.

## 2.4 Summary

In this chapter, an overview of relevant work in the different parts of an action recognition process has been given. This includes different methods for action representation like temporal and spatial approaches. Also methods for the segmentation of different actions are presented as well as solutions to achieve view independence. The last section covers a work where a similar body model as in our work was used.

Because of the fact, that the Implicit Shape Model only needs a small number of training examples and its flexibility, we decided to adapt the ISM, an object recognition approach, to human action recognition. Other advantages of the ISM are the ability of handling with incomplete data and its efficient implementation. To achieve view invariance, we have chosen to describe actions with a 3D human body model based on 3D input data.



## Chapter 3

# Basics of Human Motion Modeling and Action Recognition

In this chapter approaches of modeling human motion from sampled data and the action recognition approach which is used in this work are described. The structure of the following sections corresponds with the data processing in the implemented framework. For the representation of actions a human model is needed. One of these models is the stick figure model which is explained in the following section. After transferring input data from 3D Motion Capture techniques onto this model, a high dimensional descriptor based on relative joint angles and their derivations is created.

With such a descriptor the generation and comparison of training data with input data from a video is possible. This is done with a nearest neighbor radius search for each training sample and the observed action. Finally, the maxima of the nearest neighbor search results are determined with a Mean Shift search for the final statement of the action the subject in the input video performs.

### 3.1 Human Body Model

Selecting a sufficient human model is always a tradeoff between the realism and the computational complexity. On the one hand the model has to be accurate enough for representing all kind of human motion, on the other hand, it should be easy to compute. The model used in this work is a stick figure with 12 limbs and 15 joints, which is the same as in (González 2004). While allowing fast and efficient computations it still carries enough information about the pose of the human body. This model was adapted from (Cheng & Moura 1999).

The model as shown in Figure 3.1(a) has its root in the hip. The kinematic chains are starting from the hip to all endpoints. Kinematic chains in general are systems of rigid bodies connected with joints and are used for describing motion. As a consequence, every limb and joint is described with respect to its parent, where the parent is the next element in direction to the root. This topology is a tree structure allowing the analysis of specific moving body parts without having to take the full model into account. The

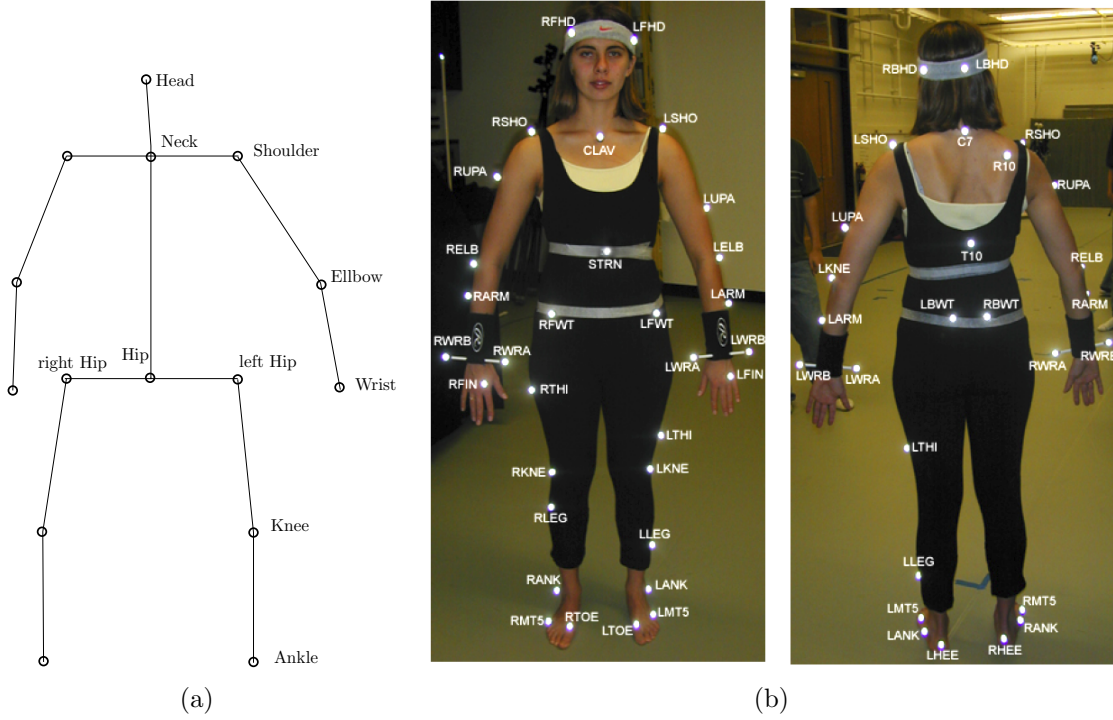


Figure 3.1: (a): Stick figure model with 12 limbs and 15 joints. (b): Placement of the 41 markers used for creating data for the CMU Graphics Lab Motion Capture Database (CMU n.d.c). These positions are transferred to the stick figure model (CMU n.d.a).

position of the ankle is for example expressed only relative to the position of the knee. The advantage of simple stick figures is that it is still possible to do segmentation and matching tasks because the geometric structure is kept sufficient (Ferrer 2005).

The original data comes from motion capture techniques, where several easy detectable reflecting markers on a human body are recorded by cameras. The marker positions have to be transferred to the stick figure model, because there are more markers than limbs in our model. This can be done by interpolation, for example one can take the mean between the right and left forehead marker (RFHD and LFHD in Figure (b)) and use it as the head in the stick figure model.

### 3.1.1 Stick Figure Model

A quite simple body model is to store 3D positions of the relevant parts. This method, which is easy to compute, has some disadvantages. One is, that the 3D positions of one person which repeats an action differ between the measurements due to different relative orientations between the person and cameras. Another reason is that the limb lengths are not the same between different persons. The results are incomparable due to these drawbacks and measuring the similarity of actions becomes a very difficult

task. A better and more robust way is to calculate the 3D orientation of adjacent limbs which are independent from the size of a limb (Ferrer 2005). Several techniques exist to express 3D orientations. They are discussed in the following sections.

### 3.1.1.1 Rotation Matrices

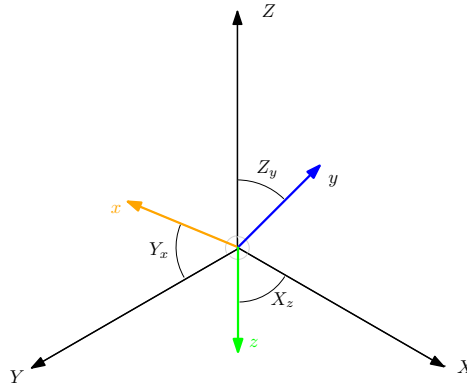


Figure 3.2: Visualization of the angles of the rotation matrix with a local coordinate system  $(x, y, z)$  and a global coordinate system  $(X, Y, Z)$ .

With a  $3 \times 3$  rotation matrix it is possible to express a rotation of a local system with regard to a global system. Each 3D unit vector  $(x, y, z)$  is expressed by components of the global reference system  $(X, Y, Z)$ . By dividing each component of the unit vector through its length we receive the cosine of the axes in respect to each axis of the global system. These cosines can be written in  $\mathbf{R} \in \mathbb{R}^{3 \times 3}$  as  $\cos_{Ab}$ , with  $A$  referring to the axis of the global system and  $b$  to the local respectively. Each row of this matrix is corresponding to one axis of the global system and each column to those of the local system (Zatsiorsky 1998).

The resulting matrix is

$$\mathbf{R} = \begin{bmatrix} \cos_{Xx} & \cos_{Xy} & \cos_{Xz} \\ \cos_{Yx} & \cos_{Yy} & \cos_{Yz} \\ \cos_{Zx} & \cos_{Zy} & \cos_{Zz} \end{bmatrix} \quad (3.1)$$

With such rotation matrices, several rotations can be expressed with one matrix  $\mathbf{R}$  by multiplying them in the right order. This order has to be preserved because the multiplication of matrices is not commutative. The disadvantage of  $\mathbf{R}$  is that 9 parameters have to be set. A 3D rotation can be expressed with only 3 parameters, so there is redundancy in this matrix which makes computations error-prone. A more compact way to express rotations are Euler angles, which are described in the following section.

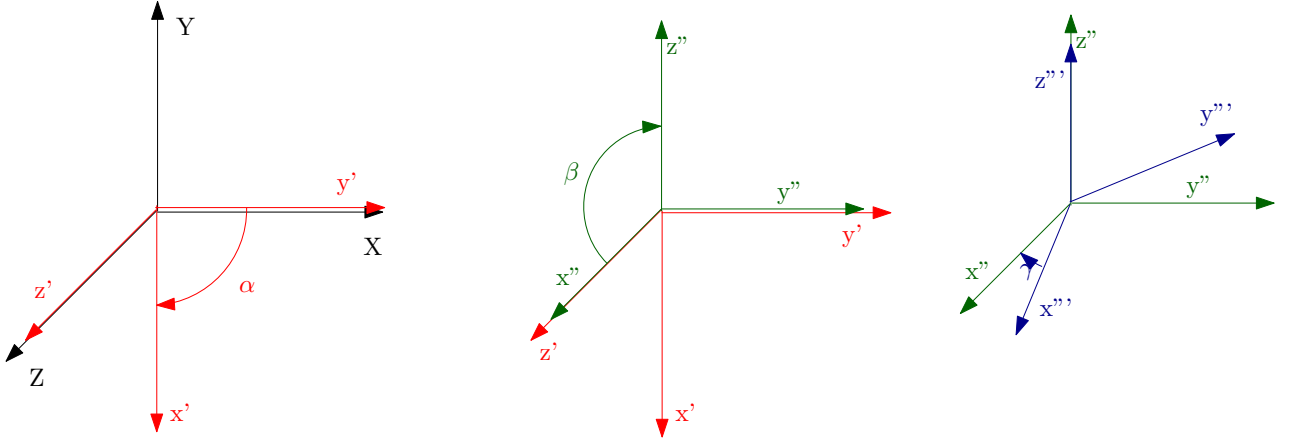


Figure 3.3: Visualization of a rotation with the angles ( $\alpha = 90^\circ, \beta = 90^\circ, \gamma = 15^\circ$ ) with the  $Zy'z''$  convention.

### 3.1.1.2 Euler Angles

With Euler angles, rotations are made in a specific sequence. There are 12 possible sequences of the 3 rotations which are possible in 3D space due to the fact that rotations are not commutative. The first rotation is always around an axis of the global coordinate system, the second and third rotations are expressed relative to already rotated axes. The notation of rotations with Euler angles is  $Ab'c''$  where  $A$  is the first rotation relative to an axis ( $X, Y, Z$ ) from the global reference system,  $b'$  is the second rotation with respect to a local axis ( $x', y', z'$ ) from the coordinate system resulting from the first rotation and  $c''$  is the final rotation around the local axis. The ' indicate the number of previous rotations. Allowed combinations of rotation axes are  $Ab'c''$  with different  $A, b$  and  $c$  and  $Ab'a''$  with different  $A$  and  $b$  (Zatsiorsky 1998). An example of one combination is shown in Figure 3.3. Euler angles can also be expressed as rotation matrices:

$$\mathbf{R} = [\mathbf{R}_z][\mathbf{R}'_y][\mathbf{R}''_x] = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \quad (3.2)$$

Despite the advantages of simplicity and easy geometrical interpretation of Euler angles, there are some implicit disadvantages. One is that in particular positions Euler angles cannot be defined. These are so-called singular or gimbal-lock positions and occur, when the  $x$ - and  $z$ -axis are in the same order after rotating the  $y$ -axis. The values of the first and third angle cannot be determined at this point because it is impossible to choose a rotation axis. As an example you can imagine a human doing a 90 degree left turn. It is impossible to distinguish if the rotation was around the global or the

local y-axis (Zatsiorsky 1998). For example, in a robot system singularities could lead to unexpected movements of an end effector.

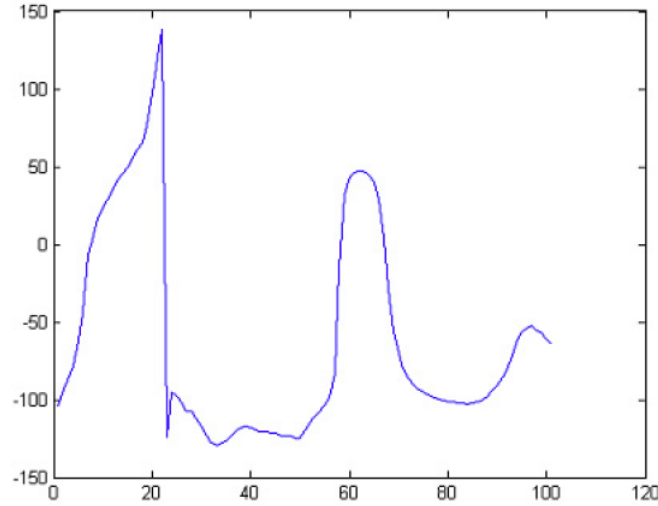


Figure 3.4: Angle values jump between  $-\pi$  and  $\pi$  which is a problem in the computation of a human body model and should be avoided (Ferrer 2005).

Other problems with Euler angles are the lack of interpolation facilities and difficulties with periodicity. If an object is rotated 360 degrees around an axis, there will be a leap between 0 and 360 degrees as shown in Figure 3.4. This is a problem if continuous data is required (Ferrer 2005).

### 3.1.1.3 Quaternions

Quaternions are a more sophisticated way to express rotations. A Quaternion is an extended complex number with one real and three imaginary parts (Stahlke 2009):

$$Q = [q_0, q_1, q_2, q_3] = q_0 + iq_1 + jq_2 + kq_3 = [q_0, \vec{q}] \quad (3.3)$$

The rules for mathematical operations are defined as follows:

$$i^2 = j^2 = k^2 = ijk = -1 \quad (3.4)$$

$$ij = -ji = k \quad (3.5)$$

$$jk = -kj = i \quad (3.6)$$

$$ki = -ik = j \quad (3.7)$$

Before one is able to rotate with Quaternions some definitions are needed. The complex conjugate is defined as  $Q^* = [q_0, -\vec{q}]$ , the norm as  $|Q| = \sqrt{Q^*Q} = [q_0^2, -\vec{q}^2]$  and the inverse element as  $Q^{-1} = \frac{Q^*}{|Q|^2}$ .

To perform a vector rotation the vector  $\vec{v}$  first has to be transformed into a Quaternion  $\hat{v}$  by  $\hat{v} = [0, \vec{v}]$ . With the rotation Quaternion

$$q = \cos\left(\frac{\theta}{2}\right) + v \sin\left(\frac{\theta}{2}\right) \quad (3.8)$$

where  $v$  is the rotation axis, e.g.  $(1, 0, 0)$  for the  $x$ .  $\theta$  is the angle with which  $v$  can be rotated by calculating

$$\vec{v}' = q\vec{v}q^* = q\vec{v}q^{-1} \quad (3.9)$$

regarding the fact that  $|q| = 1 \Leftrightarrow q^* = q^{-1}$ .

The main advantages of Quaternions are that there are no singularities, interpolation can be done by simple operations and they are very compact. However, Quaternions are not directly geometrically interpretable and we need 4 values to describe a 3D rotation. Another drawback is, that for translations more complex dual Quaternions are needed. In dual Quaternions, the 4 real numbers are exchanged by dual ones resulting in more costly calculations for multiplication. Therefore, we use an alternative way of describing angles which is described in the following section.

#### 3.1.1.4 Length Independent Human Body Modeling

In (Ferrer 2005), the human body and its pose are modeled as a tuple of 3D coordinates from the start and endpoints of the 12 limbs. Therefore 15 tuples  $(x_i, y_i, z_i)$  for start points and end points  $(x_j, y_j, z_j)$  of a limb exist. To achieve independence of the length of the limb and due to the fact that Cartesian coordinates are not the best solution when considering linear approaches for posture variation modeling, the limb locations should be expressed in angles. But the angles are not chosen direct. Instead, a polar coordinate system is used. This system is continuous within any symmetric range, e.g.  $[-\pi, \pi]$ .

$$\phi = \tan^{-1}\left(\frac{y_i - y_j}{\sqrt{(x_i - x_j)^2 + (z_i - z_j)^2}}\right) \quad (3.10)$$

$$\theta = \tan^{-1}\left(\frac{x_i - x_j}{\sqrt{(y_i - y_j)^2 + (z_i - z_j)^2}}\right) \quad (3.11)$$

$$\psi = \tan^{-1}\left(\frac{z_i - z_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}\right) \quad (3.12)$$

One limb is described in polar coordinates by the 3 angles longitude  $\psi$ , latitude  $\theta$  and elevation  $\phi$  (see Figure 3.5). These angles can be directly computed using the 15 tuples as presented in the Equations (3.10), (3.11) and (3.12).

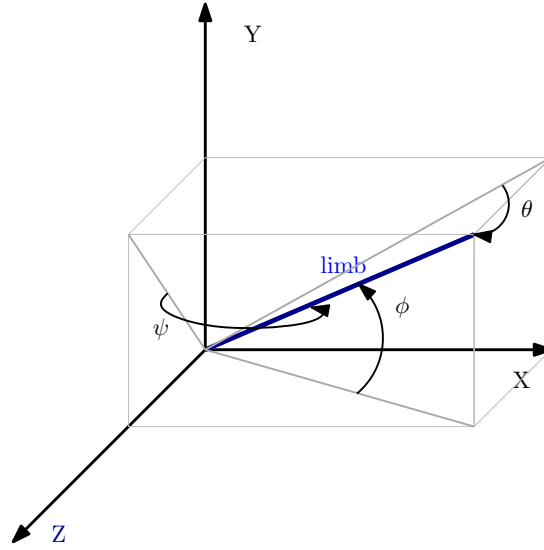


Figure 3.5: A limb of the stick figure model described with 3D polar coordinates as described in (Ferrer 2005).

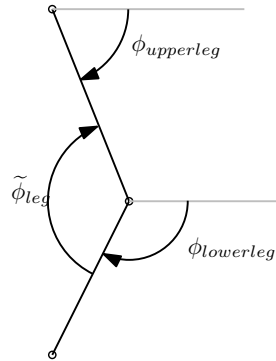


Figure 3.6: Visualization of a relative angle  $\tilde{\phi}_{leg}$  between two limbs.

With this method of describing the limbs orientation, angle values lie between  $[-\frac{\pi}{2}, \frac{\pi}{2}]$  and the discontinuity problem does not exist anymore. This is done by modeling the limb orientation by two different angles (Ferrer 2005). After calculating the angles with the Equations (3.10), (3.11) and (3.12), the relative angles  $\tilde{\psi}$ ,  $\tilde{\theta}$  and  $\tilde{\phi}$  of two adjacent limbs can be easily determined by subtracting the limbs angular value from the angular value of its parent. An example of a relative angle is shown in Figure 3.6, the angle  $\tilde{\phi}_{leg}$  is then  $\phi_{lowerleg} - \phi_{upperleg}$ . After doing this for all angles we have 36 relative angles. In addition to these values, the variation of the height of the hip is also added to the final body model, which is expressed as follows:

$$x_s = (u_s, \tilde{\phi}_1, \tilde{\theta}_1, \tilde{\psi}_1, \dots, \tilde{\phi}_{12}, \tilde{\theta}_{12}, \tilde{\psi}_{12}) \quad (3.13)$$

In order to calculate these angles and further use them for action recognition, the input data needs to be prepared to achieve view invariance. Necessary preparation steps are explained in the following section.

### 3.1.2 From Motion Capture Data to Relative Angles

In the following subsections the generation of a stick figure model from raw three dimensional data points is explained. Based on this model, the calculation of angles and their derivations is explained.

#### 3.1.2.1 Motion Capture and the C3D File Format

The 3D data source in this work is the CMU Graphics Lab Motion Capture Database (MoCap) (CMU n.d.b). This database contains several hundreds of datasets describing various actions of different persons in daily situations like walking, running, jumping etc. It is organized into Subjects and Trials and all files are indexed like  $x - y.c3d$  where  $x$  is the subject index and  $y$  is the trial number. The movies in this database are recorded at a sample rate of 120 Hz using 12 cameras around a rectangular area in which the actions are performed by people wearing markers as shown in Figure 3.1(b). The 3D coordinates from the 41 markers are stored directly in binary c3d (Dainis 1987) files.

Header section (512 bytes)
Parameter section (one or more 512 byte blocks)
Data section for 3D and analog data (one or more 512 byte blocks)

Table 3.1: Structure of a c3d file.

In these c3d files data can either be stored as 16-bit integers or floating point values. Each c3d file is structured into 512 byte blocks, where at least 3 blocks must exist to have a regular c3d file as shown in Table 3.1. The first mandatory block in each file is the 512 byte long header, which describes the format of the data. This header for example holds the number of frames and the number of data points per frame. It also includes a pointer to the parameter section. Within the parameter section, the data points are described in detail, e.g. that the 8th entry in each frame is the marker LFHD (left forehead) and also a pointer to the start of the data section is stored. This pointer must be used to locate the start of the data section. Within the data section, the data is stored frame by frame in 512 byte blocks and the last block is filled with zeros.



With the usage of such c3d files, we always receive a valid and complete 3D pose of our stick figure by converting the markers as described in the next section.

### 3.1.2.2 Transferring the CMU Database into the Stick Figure Model

The CMU Database contains 3D coordinates of 41 markers, which are named after their position as shown in Figure 3.1(b). It is noticeable, that these markers have an offset to the real position of the joint because they lie outside the body outline. But for our purposes this position error is tolerable. These markers have to be transformed onto our 15 joint stick figure model. Some of the stick figure joints can be initialized directly from the marker locations while others have to be calculated as the average of several marker locations. The formula for this calculation is

$$average(\vec{x}, \vec{y}) = \frac{1}{2}(\vec{x} + \vec{y}). \quad (3.14)$$

Stick figure Joint	Source data
left hip	$average(LFWT, LBWT)$
right hip	$average(RFWT, RBWT)$
left shoulder	LSHO
left elbow	LELB
left hand	LWRB
right shoulder	RSHO
right elbow	RELB
right hand	RWRB
left knee	LKNE
left foot	LHEE
right knee	RKNE
right foot	RHEE
neck	$average(LSHO, RSHO)$
head	$average(RFHD, LFHD)$
hip	$average(\text{left hip}, \text{right hip})$

Table 3.2: Correspondence of stick figure joints and Motion Capture markers.

The way of generating the coordinates of the 15 stick figure joints is described in Table 3.2.

### 3.1.2.3 Normalization of the Stick Figure to Achieve View Invariance

The stick figure, which was defined in Section 3.1.2.2, is expressed in a world coordinate system. To achieve view invariance, we need to express the stick figure in a local coordinate system, equal the orientation and also norm the size of the figure. This is done with an affine transformation of each coordinate of the stick figure. The transformation includes a translation, rotation and scaling.

The origin of the local coordinate system is placed in the middle of the left and right hip. This point is taken as the translation vector  $\vec{v}_t$  of the affine transformation.

The scale factor  $s$  is the length of the spine from the first available frame.

To rotate every figure to the same orientation, we generate a matrix  $(b_x^T, b_y^T, b_z^T) \in \mathbb{R}^{3 \times 3}$  with

$$\vec{b}_x = \vec{v}_t - \text{rightHipCoordinate} \quad (3.15)$$

$$\vec{b}_y = \vec{b}_x \times (\vec{v}_t - \text{neckCoordinate}) \quad (3.16)$$

$$\vec{b}_z = \vec{b}_x \times \vec{b}_y \quad (3.17)$$

$$M_{rot} = \begin{bmatrix} \vec{b}_x & \vec{b}_y & \vec{b}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.18)$$

$$M_{trans} = \begin{bmatrix} 0 & 0 & 0 & \vec{v}_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.19)$$

$$M_{scale} = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.20)$$

All these operations are expressed in three homogeneous  $\mathbb{R}^{4 \times 4}$  matrices  $M_{rot}$  (3.18),  $M_{trans}$  (3.19),  $M_{scale}$  (3.20). The affine transformation matrix from 3D world coordinates to a normalized person centric coordinate system is then calculated as

$$A = (M_{trans} * M_{rot} * M_{scale})^{-1} \quad (3.21)$$

A stick figures 3D coordinate  $(x, y, z)$  can then be transformed by generating a homogenous vector and multiplying it with  $A$ . The result is shown in Figure 3.7 which also shows that a person walking in different directions is normalized so that it is always oriented into the same direction.

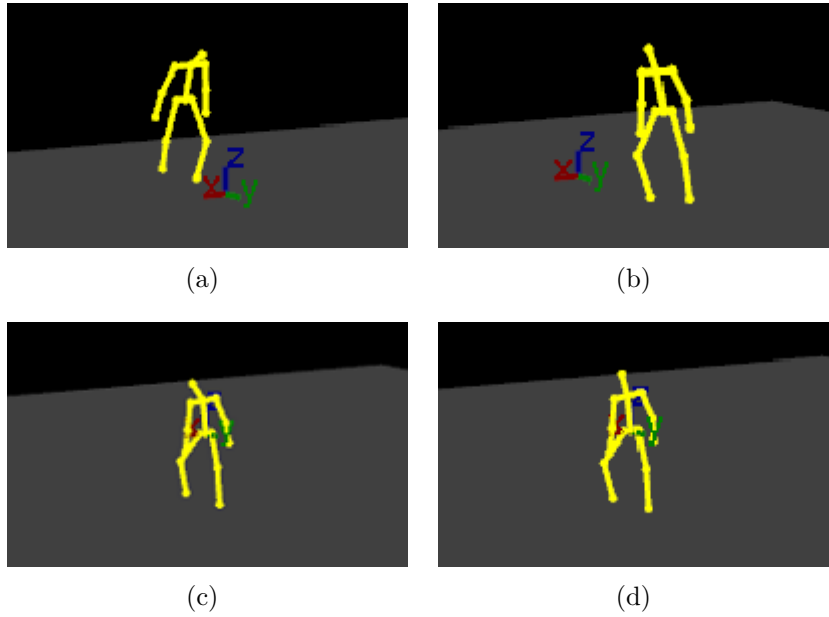


Figure 3.7: (a): Frame 276 of a walk action (Subject 139, Trial 30) from the CMU Motion Capture database. (b): Frame 833 of the same walk action in world coordinates. (c): Frame 276 of the same walk action in normalized coordinates. (d): Frame 833 of the same walk action in normalized coordinates.

#### 3.1.2.4 Calculating Absolute and Relative Angles

With a normalized stick figure, calculation of limbs angles can be done with the Equations (3.10), (3.11), and (3.12). As we have 15 joints in our model the resulting number of limbs is 12.

After calculating all angles we have 36 angular values which can be expressed as relative angles with respect to their parent limbs following kinematic chains, as described in Section 3.1.1.4. This is done due to the fact that relationships in movements between different limbs exist which we want to incorporate (González 2004). By calculating all relative angles from one time step, we receive the vector  $x_s$  holding all relevant values of every frame. An example of such angular values over time is depicted in Figure 3.8. It is noticeable that there are no leaps in the curve as shown in Figure 3.4. To incorporate the rate of change of these angular values into our action descriptor, we also derivate the values. The methods of derivations up to the order two are explained in the following section.

### 3.1.3 Derivation of Relative Angles with Numerical Methods

For the action recognition process it is also useful to have a look at the rate of change of the relative angles. Therefore the first and second derivations of the relative angles

Limb	start Joint	end Joint
hip limb	left hip	right hip
shoulder limb	left shoulder	right shoulder
spine	middle of the hip	neck
neck limb	neck	head
left upper arm	left shoulder	left elbow
left lower arm	left elbow	left hand
right upper arm	right shoulder	right elbow
right lower arm	right elbow	right hand
left upper foot	left hip	left knee
left lower foot	left knee	left foot
right upper foot	right hip	right knee
right lower foot	right knee	right foot

Table 3.3: Limbs of the stick figure model with their start end points.

over time are computed. For the discrete angular values this can be done by calculating the differential quotient between subsequent data points. The differential quotient can be calculated in three ways:

$$\text{forward : } f'(x) = \frac{f(x+h) - f(x)}{h} \quad (3.22)$$

$$\text{backward : } f'(x) = \frac{f(x) - f(x-h)}{h} \quad (3.23)$$

$$\text{symmetric : } f'(x) = \frac{f(x+h) - f(x-h)}{2h} \quad (3.24)$$

Since we have values for successive frames we can set  $h = 1$  to calculate the derivations.

## 3.2 Implicit Shape Model based Action Recognition

The action recognition process in this work is based on the Implicit Shape Model (ISM). The ISM approach is very popular for object recognition and tracking ([Jüngling 2011](#), [Leibe et al. 2008](#)) and will be explained in the following sections, including its adaptations to action recognition.

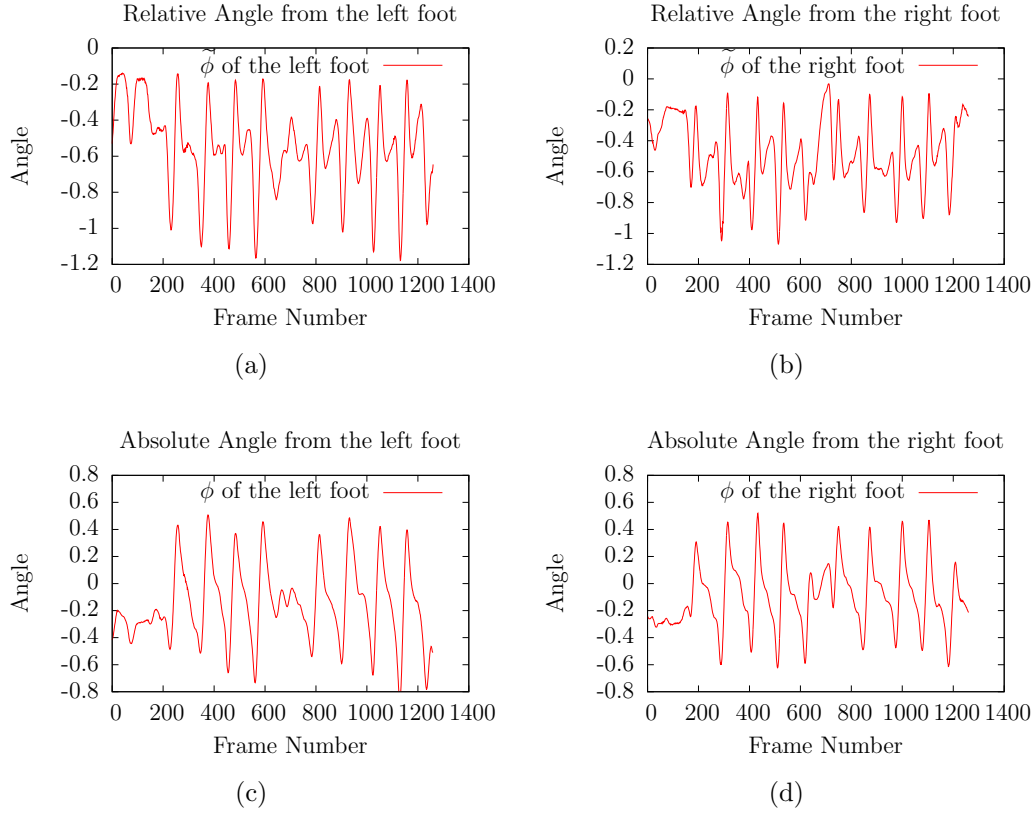


Figure 3.8: This figure shows the calculated absolute and relative angle values of  $\phi$  over time of the left and right foot from a walk action (Subject 139, Trial 30) of the CMU Motion Capture database (a): Left foot relative angle  $\phi$ . (b): Right foot relative angle  $\phi$ . (c): Left foot absolute angle  $\phi$ . (d): Right foot absolute angle  $\phi$ .

### 3.2.1 ISM - a Brief Overview

Object recognition in images based on the ISM has two phases. During the training phase which is shown in Figure 3.9, local and characteristic appearances of an object are learned and clustered to get an initial codebook with prototypes. This can be done in images for example by detecting characteristic local features like Harris Corners and extract a region around these corners from the image. Detected features are then clustered and a prototype is generated from each cluster by calculating the cluster mean. In the codebook, the prototype is stored together with its occurrence locations as offsets with respect to the objects center.

During the second phase which is the recognition phase, local features are detected with the same technique as in the training phase, as shown in Figure 3.10. For each feature, the best matching codebook entries are searched and stored and for each detection, a probability of the match is stored also. All of these matches vote for the center of

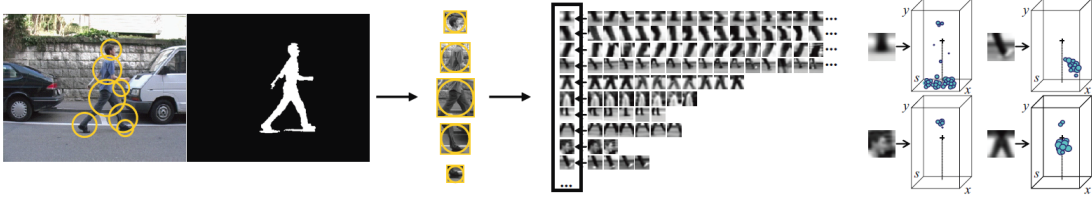


Figure 3.9: The ISM training phase with the extraction of interest points (yellow circles), the clustering and generation of mean samples called prototypes and the final codebook which contains the prototypes and the spatial occurrence of these prototypes. These positions are relative to the objects center (Leibe et al. 2008).

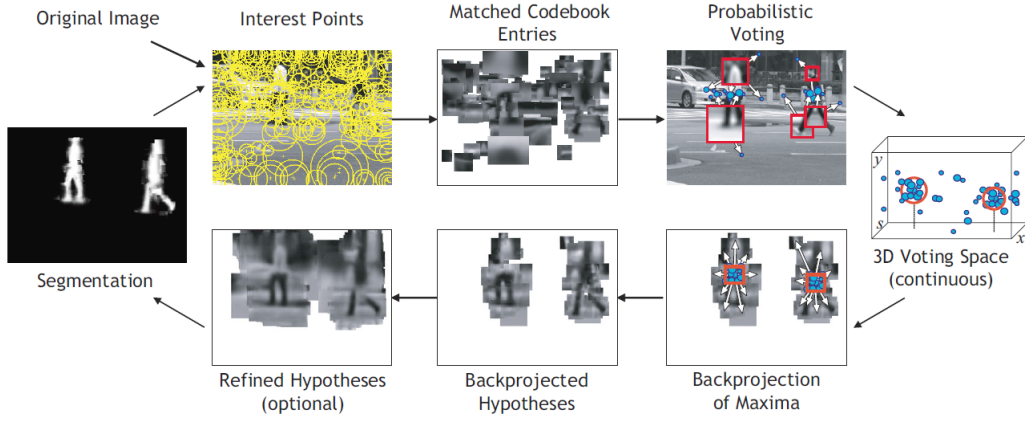


Figure 3.10: The ISM recognition process as proposed in (Leibe et al. 2008).

the object according to the stored offsets in the dictionary. This voting process results in a *Probabilistic Hough Voting* space, containing votes for 3D positions. The third dimension is the scale of the object. This voting space then is separated into bins to determine candidates for maxima. In regions around these candidates, the maxima are searched using Mean Shift mode seeking. With the found maxima, back projected hypotheses can be generated providing rough indications of the object location. An optional last step segments the background from the patches in the hypotheses to get a pixel based figure-ground segmentation.

### 3.2.2 Adaptions to the ISM

The general idea of the thesis is to apply the ISM method to the problem of action recognition. The dictionary for our ISM based action recognition approach consists of several actions which are described with action descriptors. This can be a walk action from the contact of the left foot on the ground until it reaches the ground again.

These action snippets are selected manually from CMU files. Codebook generation as described in (Jüngling 2011) is not part of this work, we treat every action sample independent from each other. Instead of saving the objects center, we use the end of an action as votes. All training samples for one action type represent exact the same action with the same start- and endpoint.

By searching the nearest neighbor from the actual time step of the whole input sequence in all time steps of a codebook entry, we receive an offset to the probable end of action represented by the codebook entry. It is not necessary to look for interest points in the input data as we only get relevant 3D coordinates. With these offsets which represent the number of time steps to the end of an action, votes for the end of an action are generated and saved into the voting space. Then, maxima of votes for ends are searched with Mean Shift. The details of this procedure are described in the following subsections.

### 3.2.3 Building a Descriptor for Action Recognition

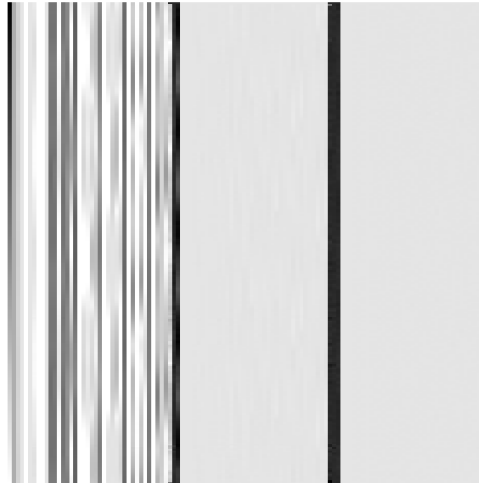


Figure 3.11: Visualization of the values in the descriptor for a walk action. The image is a greyscale image with values between 0 and 255. To show the hip values clearly, the colors of the hip and its derivations are set to darker values.

To describe actions a frame based descriptor is used. For every frame the descriptor holds 117 values including relative angles, their first and second derivatives and the hip position and its derivations. The final descriptor is then a matrix  $\in \mathbb{R}^{(NumberOfFrames) \times 117}$ . The 117 values in one row of the descriptor are composed of the following elements:

$$d_{frame} = (hip_x, hip_y, hip_z, \tilde{\phi}_1, \tilde{\theta}_1, \tilde{\psi}_1, \dots, \tilde{\phi}_{12}, \tilde{\theta}_{12}, \tilde{\psi}_{12}, \\ hip'_x, hip'_y, hip'_z, \tilde{\phi}'_1, \tilde{\theta}'_1, \tilde{\psi}'_1, \dots, \tilde{\phi}'_{12}, \tilde{\theta}'_{12}, \tilde{\psi}'_{12}, \\ hip''_x, hip''_y, hip''_z, \tilde{\phi}''_1, \tilde{\theta}''_1, \tilde{\psi}''_1, \dots, \tilde{\phi}''_{12}, \tilde{\theta}''_{12}, \tilde{\psi}''_{12})^T$$

These values include the hip position and their rate of change as well as all relative angles and their derivations. With this descriptor, both training samples in the dictionary and the input data as shown in Figure 3.11 are encoded. This allows the comparison of training data with input data. The details of this comparison are described in the next section.

### 3.2.4 Codebook Entries and Voting

Since codebook generation is beyond the scope of the thesis, we use the descriptor of a training sequence as an action prototype. Having a codebook of prototypes for a specific action and a descriptor -in our case some c3d files-, we cycle through the descriptor of the input video per frame. All action prototypes are represented with the action descriptor proposed in Section 3.2.3.

For each time step of the input video, we look for the nearest neighbors in each codeword of the training dataset whose distance is below a threshold. The results are positions in the analyzed training data descriptor which can be interpreted as the most probable point in time of what the subject in the video is compared to the action in the codeword. For each nearest neighbor which is valid, an offset between the actual time step in the input sequence and the estimated end of the action represented by the codeword is calculable. Using this offset, a vote for the end of the action can be made from every valid nearest neighbor. The threshold for the nearest neighbor search is set manually and evaluated later.

The whole algorithm is described in Algorithm 1. All votes are stored with the `voteForEndOfAction(codeword, voteIndex)` function, which allows the accumulation of votes and then the search of maxima. The threshold helps to sort out any nearest neighbors which do not match well. Within this function, a reference of the codeword is stored together with the `voteIndex`. The `FLANNSearch()` function returns a matrix which holds the nearest neighbor's indices of the actual frames descriptor in all frames of one training example. Details of the Fast Library for Approximate Nearest Neighbors (FLANN) are described in the following section.

### 3.2.5 Approximate Nearest Neighbor Search

FLANN is a library supporting C, MATLAB, and Python which performs fast approximate nearest neighbor search. It automatically chooses the best algorithm for a dataset and can reach an improvement in speed of several orders of magnitudes compared to other algorithms, e.g. linear search (Muja & Lowe 2009). There are two algorithms



---

**Algorithm 1:** Voting with approximative nearest neighbor search.

---

```

for  $i \leftarrow 1$  to  $numberOfFrames$  do
  foreach codeword  $c$  do
     $nearestNeighborIndices \leftarrow FLANNSearch(inputDescriptorRow(i), c);$ 
    foreach nearestNeighborIndex do
      if  $nearestNeighborDistance < THRESHOLD$  then
         $voteIndex \leftarrow i + ((c).length - nearestNeighborIndex);$ 
         $voteForEndOfAction(c, voteIndex);$ 
      end
    end
  end
end
 $searchForMaximaInAllVotes();$ 

```

---

which perform very well. One searches hierarchical *k-means trees* with a priority search order and the other one uses multiple randomized *kd-trees*.

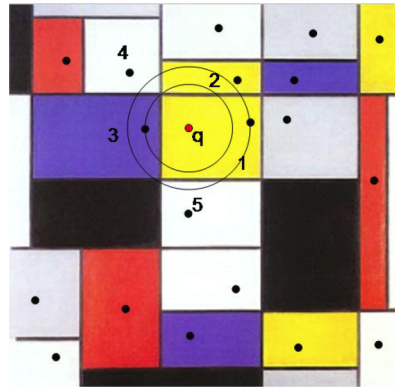


Figure 3.12: Priority based nearest neighbor search in a *kd-tree* as proposed in (Silpa-Anan & Hartley 2008). First, the tree is descended until the query point, which is labeled with a 1, is reached. Then, the neighbor cells are searched, which are the nodes 2 to 5 in this image. The boundaries of the search are spherical with an adjustable radius.

The concept of *kd-trees* is proposed in (Friedman et al. 1977) and extended in (Silpa-Anan & Hartley 2008), where multiple randomized *kd-trees* are created. *kd* stands for *k dimensions* and such trees organize points in a *k*-dimensional space. Instead of splitting the data in half at each level of the tree along the dimension with greatest data variance, the split dimension is chosen randomly from the first *d* dimensions with

greatest data variance. An example of a *kd-tree* and the nearest neighbor search within such a structure is shown in Figure 3.12.

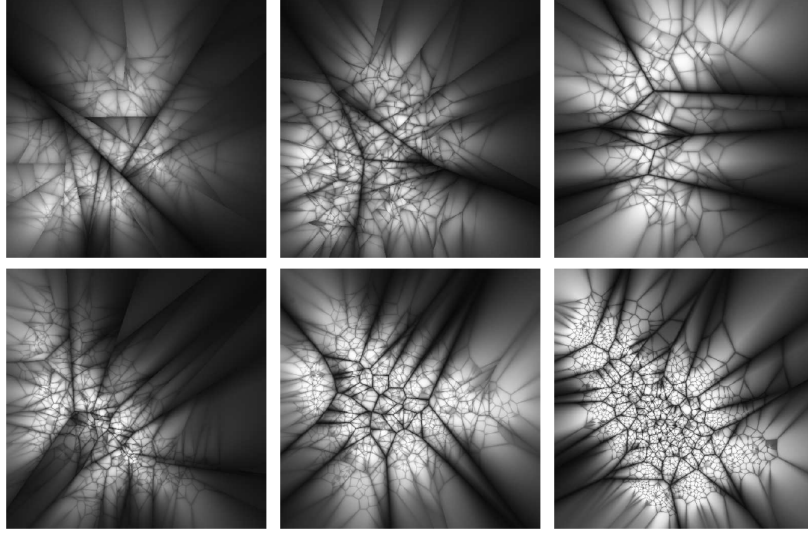


Figure 3.13: Visualization of hierarchical k-means trees of 100000 data points with different branching factors. Gray values indicate the distance between the nearest and second nearest center (Muja & Lowe 2009).

A hierarchical *k-means tree* is generated by splitting the data set into distinct regions using k-means clustering recursively. The data points are partitioned by choosing an initialization set and performing clustering recursively. The partitioning is done by selecting the data point which is nearest to the center of a cell and separating these centers from others with hyper planes that are bisecting the direct connection of the centers. In a 2D space this leads to a Voronoi-like decomposition. Then, new center points of all points within the new cluster are determined and the process continues until the number of points in a region is smaller than  $k$ . The result of such a tree is shown in Figure 3.13.

When the dataset is provided to the FLANN library, a cross-validation approach is used to determine the best algorithm and optimal parameters. The performance of the algorithms depends on the data structure, the desired search precision and also the parameters of the algorithm, e.g. the number of trees to use. Choice of the best algorithm is then an optimization problem in the parameter space trying to minimize a cost function including search time, tree build time, and memory usage. This problem can be computed only on a fraction of the dataset and the solution stays the same for similar datasets (Muja & Lowe 2009).

As mentioned in Section 3.2.4, we use one row of the descriptor as the input dataset which corresponds to one time step. With this dataset, we perform a radius search to find all nearest neighbors in all training samples. The indices and distances of these

nearest neighbors are returned by FLANN and all neighbors, whose distance is over a threshold, are disregarded. Given an input sequence at time step 100 and a codebook entry of the action walk which describes 80 time steps for example, the nearest neighbor search may return the index 50 with distance 5 among other indices. If this distance is below the threshold, the result indicates that the walk action ends in  $80 - 50 = 30$  frames. Therefore we vote for the end of the walk action at frame  $100 + 30 = 130$ . Within these voting results, we can search for maxima. These maxima indicate that an action is completed at this point. Such a search can be performed with Mean Shift maxima search, which is described in the following section.

### 3.2.6 Mean Shift Mode Seeking

Mean Shift is an iterative procedure which is used to determine the modes of a sample density function. By identifying dense regions in the feature space the maxima of this probability density are found. This also allows the identification of clusters by identifying the nearby maximum (Cheng 1995).

For every data point of the input values Mean Shift calculates the mean within a window around the data point and shifts the center of the window in the direction of the mean. This is repeated until convergence and the window is shifted towards a denser of the dataset. The size of this window also influences the size of clusters which Mean Shift will find. Examples of window center trajectories are shown in Figure 3.14.

$$K(x) = \begin{cases} 1, & \text{if } \|x\| \leq \lambda \\ 0, & \text{if } \|x\| > \lambda \end{cases} \quad (3.25)$$

$$K(x) = e^{-\frac{x^2}{2\sigma^2}} \quad (3.26)$$

A very important part is the selection of the window type. Such a weighted window is also called a kernel and can be a simple flat kernel as shown in Equation (3.25) or the often used Gaussian kernel (3.26). The Mean Shift is then calculated as follows where  $S$  is the dataset.

$$m(x) = \frac{\sum_{s \in S} K(s - x)s}{\sum_{s \in S} K(s - x)} - x \quad (3.27)$$

The density estimation window is moved by  $m(x)$  and the repeated application of this movement is called Mean Shift. Mean Shift is also called “a very intuitive estimate of the gradient of the data density” (Cheng 1995) and therefore applicable to optimization problems like finding local maxima. The movement is cancelled when the criterion  $m_i(x) - m_{i-1}(x) < \epsilon$  is reached.

A visualization of such an optimization is shown in Figure 3.15, where two maxima are found by Mean Shift. The Mean Shift optimization is comparable to genetic algorithms for such purposes (Cheng 1995).

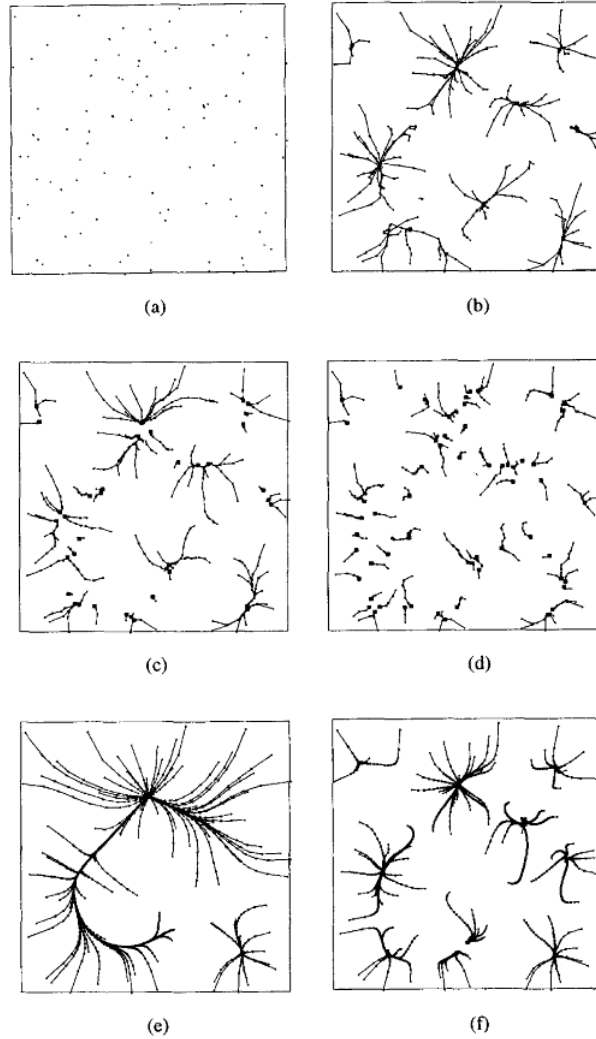


Figure 3.14: Mean Shift trajectories for different kernels and different procedures, each trajectory shows the process of shifting the window towards the local mode (Cheng 1995). (a) is the input dataset and b) - f) are the results with varying cluster size and number of iterations. In (c) and (d) truncated Gaussian kernels were used and in (e) and (f) non-truncated Gaussian kernels were chosen.

Applications of Mean Shift Searches in Computer Vision were introduced by Comaniciu in (Comaniciu & Meer 2002). They used Mean Shift on a pixel base and in each iteration, the density window is shifted towards the center of their neighborhood. The result is an image separated in homogeneous regions as showed in Figure 3.16. Such regions can be separated straight forward.

In this work, the result of performing a Mean Shift mode seeking on the voting results are maxima where ends of actions are estimated. For the final step, which is the statement of what the subject is doing at the moment, we look at the density of the

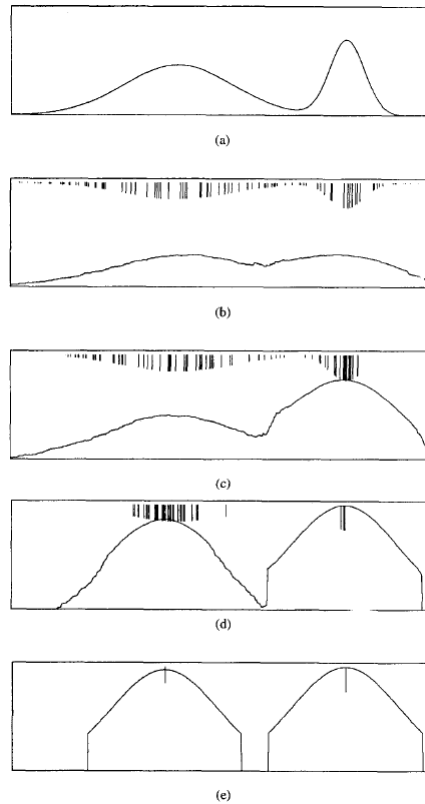


Figure 3.15: Iterations of the Mean Shift procedure to find global maxima. a) shows the function and b) - e) are the four iterations. The bars are the shifted data points from a) and the curve in b) - e) is a function whose local maxima approximate the maxima of the original function. In e) the two maxima are found (Cheng 1995).

Mean Shift result and separate the strongest maxima from weaker ones with a threshold. This operation is described in the next section.

### 3.2.7 Determination of Actions from Mean Shift Mode Seeking

The final step is the statement of the type of action which was recognized. For this purpose, we search for maxima in the density of the Mean Shift result which are over a threshold for every frame. The threshold is dependent on the number of training samples because this number influences the total number of votes and therefore the height of the maxima. In this work, the total number of training samples for a label is multiplied with a constant factor to get such a threshold. This constant factor is evaluated in Chapter 5. It is also noticeable, that we only look for maxima which lie in the future from our actual point of view because our votes are only pointing to future time points.



Figure 3.16: Example of an application of Mean Shift in Computer Vision. As a result, homogeneous regions are found within an image and these regions can be segmented very easy (Comaniciu & Meer 2002).

The result of comparing to a threshold is a simple *true* or *false* statement for each time step where true means that the action was recognized.

The value of this threshold is set for each label and directly dependent of the number of entries  $nSamples_{label}$  for each label in the training sample. Due to the fact, that the total number of votes  $nVotes(label)$  for one label in each time step is

$$nVotes_{label} \leq nSamples_{label} \quad (3.28)$$

$nVotes_{label}$  is equal or below the number of samples because only nearest neighbors found by FLANN whose distance is below the threshold result in a vote (see Section 3.2.5).

$$t_{label} = nVotes_{label} * z \quad (3.29)$$

The threshold  $t_{label}$  is defined as the product of  $nVotes_{label}$  with a constant factor  $z$  in Equation (3.29) and evaluated later.

### 3.3 Summary

In this chapter the theoretical background of the action recognition framework is presented. In the first parts, a human body model was introduced and methods for calculating angles between limbs of this model are described. Also, normalization techniques are explained to achieve view independence. These steps are also visualized in Figure 4.1. An action recognition framework based on the Implicit Shape Model was also described. As it can be seen in Figure 4.3, this includes all steps from nearest neighbor search on action descriptors, voting and the maxima search with Mean Shift to get a final action statement.

In the next Chapter, the implementation of the proposed action recognition framework is described.

# Chapter 4

## Implementation

In this chapter, the main steps of the implementation of the ISM based action recognition framework are described. Also, possibilities to speed up the implementation are presented.

### 4.1 The Action Recognition Framework

The action recognition framework is divided into the training phase and the recognition phase. The training has to be performed in advance and is explained in the following section.

#### 4.1.1 Training Phase

For the action recognition framework, first the training sequences have to be encoded to get a dictionary. In this dictionary, a variable number of descriptors from training sequences are stored together with a label for the action they represent. The dictionary is generated manually by selecting the start and end of the training sequence out of a file from the CMU Motion Capture database.

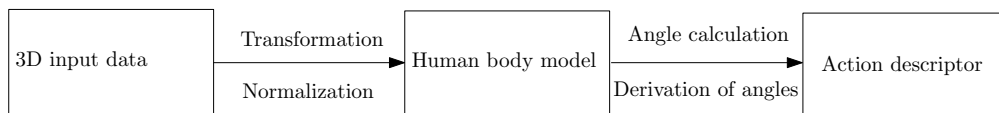


Figure 4.1: The processing pipeline for transferring 3D data to our human body model and the action descriptor.

In Figure 4.2, the tool for generating a dictionary entry is shown. First, the start and end frame of the action sequence are defined with the two sliders. Then a descriptor for this range has to be generated and then this descriptor can be added to an existing

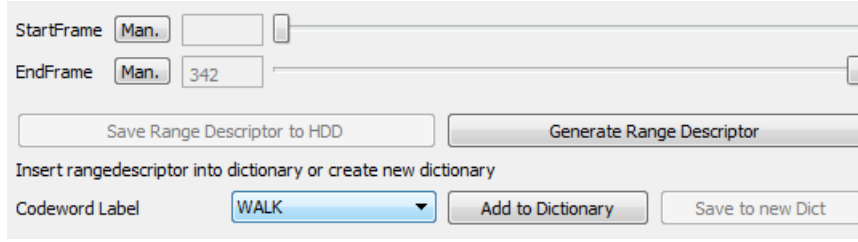


Figure 4.2: The tools for generating a dictionary entry.

dictionary or a new dictionary with a label like ‘WALK’. The processing pipeline for the generation of one training sample for the dictionary is shown in Figure 4.1 and explained in detail in the following enumeration:

1. 3D input data (c3d file)  $\rightarrow$  Human body model
  - The 41 3D marker positions for every time step are read from the c3d file and converted to the 15 joints of the human body model (see Section 3.1.2.2).
  - The human body model is normalized with an affine transformation to the same orientation and size. This is performed to achieve view invariance (see Section 3.1.2.3).
2. Human body model  $\rightarrow$  Action descriptor
  - First, the absolute and then relative angles of the 12 limbs of the normalized human body model are calculated. We use a polar coordinate system and the three angles (latitude, longitude and elevation) for each limb. The results are continuous angular values over time which can be derivated (see Section 3.1.2.4).
  - Derivations of the angular values are concatenated with the hip position and the relative angle values into an action descriptor for each time step. This results in a vector with 117 elements (see Section 3.2.3).
  - Each vector is written line by line into the action descriptor which has the form of a matrix with 117 columns and one row for every time step.
3. Action descriptor  $\rightarrow$  Dictionary
  - The action descriptor is stored together with a label into the dictionary. The final dictionary contains several labelled descriptors of different actions.

### 4.1.2 Recognition Phase

Having a dictionary we can start the action recognition process. The input sequences are also c3d files from the CMU Motion Capture database. The processing of the input



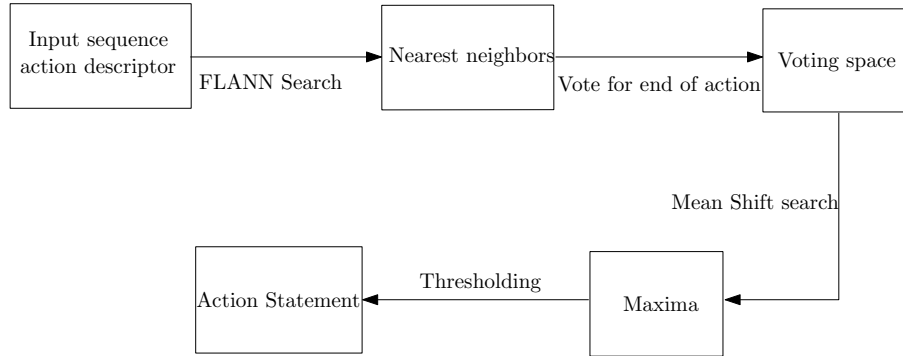


Figure 4.3: The processing pipeline for the action recognition framework.

file is time step based. A descriptor for every time step is generated in the same way as during the generation of a dictionary. Then the descriptor is processed as shown in Figure 4.3. In the following enumeration a more detailed description is given:

1. c3d input file  $\rightarrow$  Action descriptor

- For each time step of the input sequence an action descriptor matrix with dimensions  $\mathbb{R}^{1 \times 117}$  is build the same way as in the generation of the dictionary.

2. Action descriptor  $\rightarrow$  Association between input sequence and training sample

- To find the association between the descriptor of the input data and a training sample, a radius search is performed. This search is performed by the Fast Library for Approximate Nearest Neighbors (FLANN). The result is a list of nearest neighbors between the input descriptor and training samples together with distances. One example result could be  $Index = 30, Distance = 8; Index = 31, Distance = 9$ ; for the search in a walk action training sequence with a length of 70 time steps (see Sections 3.2.4 and 3.2.5).
- All results whose distance is below a defined threshold are valid and processed as a vote. The threshold defines the radial boundary of a prototype classifier.
- Each valid result of the FLANN search is taken as a correspondence between the time step of the input file and the training sample.

3. Nearest neighbors  $\rightarrow$  Voting space

- Each activated dictionary entry indicates the position of the current input action in the training sequence. Therefore the estimated end of the action represented by the training sample in which the nearest neighbor was found

is predictable. Taking the example from above and being at time step 100 of the input sequence, votes for the end of the action walk are stored at the indices  $100 + (70 - 30) = 140$  and  $100 + (70 - 31) = 139$ . From the actual point in time we only vote for ends of actions in the future.

- The voting space is the 1D time axis. One voting space is required for each action.

#### 4. Voting space $\rightarrow$ Action statement

- We are interested in maxima within the voting space. These maxima indicate the most probable end points of actions.
- To find these maxima we use Mean Shift Mode seeking (see Section 3.2.6).
- For the final action statement, all maxima in which are in the future and exceed a predetermined threshold are taken into account to get the action statement (see Section 3.2.7).

To optimize the runtime of the implementation, some of these steps can be executed in parallel. Therefore we used OpenMP, which is described in the next section.

## 4.2 Steps Towards a Real-time Application

Due to the fact, that both the voting and the Mean Shift loop are done independently for every codeword, it is possible to run these algorithms in parallel. For this purpose, the Open Multi-Processing (OpenMP) application interface is used. OpenMP can be used with C, C++ and Fortran programs and allows simple shared-memory parallelism. Parallelization with OpenMP takes place at loop level. OpenMP therefore has some directives which can be used to run a parallelized for-loop for example. OpenMP

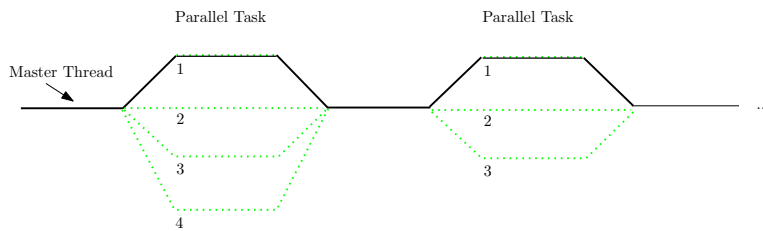


Figure 4.4: The multithreading concept which is used by OpenMP.

implements multithreading by using one master thread which executes instructions consecutively, see Figure 4.4. This master thread then forks a specified number of slave threads which execute the task. The command for setting the number of threads is `omp_set_num_threads(int numberOfThreads)`.

Another interesting property of OpenMP is that the threads are assigned to processors regarding usage and machine load (Board 2011).

The declaration of private and shared variables also has to be done in the directive. Shared variables can be accessed by all threads simultaneously whereas when a variable is private, each thread uses a local copy. Loop iteration variables are private per default in OpenMP.

The Mean Shift Mode seeking for example can be parallelized because the calculation of maxima of each label is independent from other labels. Other parts which were parallelized are the normalization of the stick figure and the calculation of angles in each time step.

### 4.3 Performance

The performance of the action recognition framework depends strongly on two factors:

- The similarity of the input sequence with training samples.
- The number of training samples in the dictionary.

The first factor has an influence especially on the performance of the Mean Shift Mode seeking. If the input sequence matches training samples, many nearest neighbors are found and the number of input values for Mean Shift increases. This also increases the runtime. In an extreme scenario, where all nearest neighbor distances exceed the threshold and no votes are made, Mean shift Mode seeking is not needed because we don't have any maxima.

As the number of votes also depends on the number of training samples in the dictionary, a very large dictionary would also result in longer runtimes. If there were three walk actions instead of one in the dictionary, we receive up to 3 votes instead of only one per time step.

The overall processing time for one time step in our experiments therefore was between 0.1 and up to 10 seconds with large dictionaries.

### 4.4 Summary

In this chapter an overview of the action recognition framework is given. This includes the generation of the dictionary as well as the recognition process. In the next chapter, the framework is evaluated and several experiments are performed.

# Chapter 5

## Experiments

In this chapter the evaluation of the developed approach is presented. In the first part, suitable parameter settings are determined. This includes FLANN and Mean Shift parameters as well as classifier thresholds and the selection of training sequences. After a well performing setting has been found, the action recognition approach is tested for its continuity and robustness.

### 5.1 Parameter Settings

In the following sections, the influence of different parameters on the action recognition performance is evaluated. First, the parameters for FLANN are evaluated. These are the number of trees which should be built and the number of leafs to visit while performing the radius search. For the Mean Shift mode seeking the window size has to be defined and results for different window sizes are presented. The aim is to find parameters which allow a robust discrimination between individual actions. In general, this is a tradeoff between robustness and

#### 5.1.1 Number of Trees in FLANN Search

One parameter which can be defined during a FLANN Search is the numbers of trees. The trees which FLANN uses are described in Section 3.2.5. In the example in the FLANN manual, a default value of 4 is given. Therefore we take this parameter as a start value and vary it from 1 to 4 with two different dictionaries and then choose the value which gives the best results.

##### 5.1.1.1 Results for a Walk Action

For this experiment, the input action sequence is trial number 6 from subject 7 in the CMU Motion Capture database. The entries in the dictionary are presented in the following table, where one entry is a walk action starting and ending with the left foot on the ground.

Subject	Trial	Start Frame Number	End Frame Number
07	07	77	215
07	08	32	180
07	09	15	138
07	10	26	157
07	11	28	136
07	12	13	135

Table 5.1: Dictionary of walk actions used for retrieving suitable values of the number of FLANN trees to use.

To get comparable results, all other parameters are not changed during the experiment and set to the following values:

- Number of leafs to visit=8
- Mean Shift window size=1.75
- Threshold for nearest neighbors=10

Figure 5.1 shows the result of the Mean Shift mode seeking for the walk action with the environment settings as mentioned above. The maximum on the left side of the figure is lower because the input sequence starts in the middle of a walk action and therefore we receive fewer votes compared to the 2 full walk actions with high maxima.

#### 5.1.1.2 Results for a Jump Action

For this experiment, the input action sequence is trial number 15 from subject 118 in the CMU Motion Capture Database. The entries in the dictionary are presented in the following table, where a jump action starts with the swinging of the arms and ends with a safe stand.

All parameters are set to the same value as given in Section 5.1.1.1.

Figure 5.2 shows the result of the Mean Shift mode seeking for the jump action using the dictionary and parameter setting above. In the input sequence the person performs one jump which has to be recognized here.

#### 5.1.1.3 Recommendation for the Number of Trees to Use

A Value of 3 for the number of trees to use gives good results regarding maxima from the Figures 5.1 and 5.2. In general, all evaluated parameter settings result in maxima that are sufficient for action recognition, but with a value of 3 we received the most promising results.

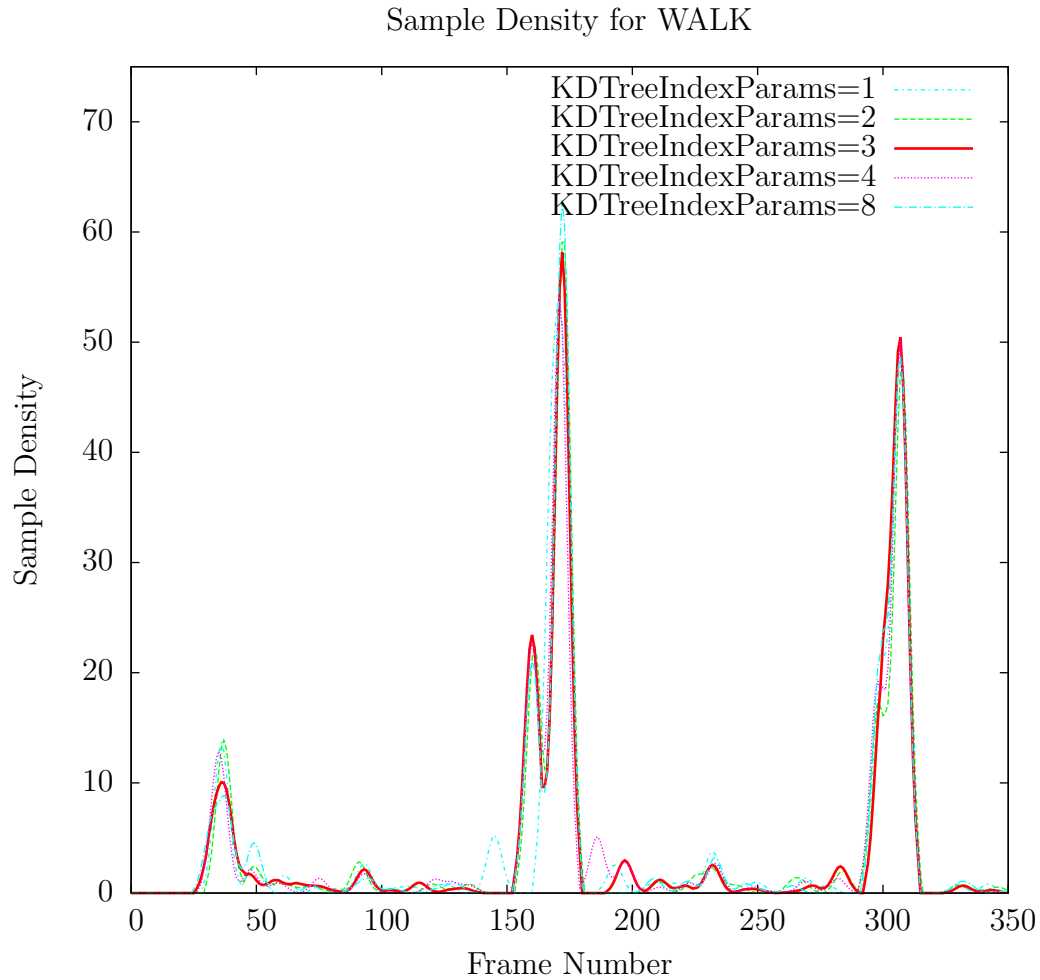


Figure 5.1: Sample density for different settings of the parameter `KDTreeIndexParams` for a WALK action.

### 5.1.2 Number of Leafs to Visit when a Nearest Neighbor is Searched with FLANN

One parameter of the `knnSearch()` function in FLANN is `SearchParams`. It can be set with the help of the `SearchParams()` function and the value in the FLANN example is 128. Higher values of this parameter would increase the precision but also the time for performing the radius search is increased. Therefore we vary the parameter from 1 to 256 and compare the results.

Subject	Trial	Start Frame Number	End Frame Number
118	6	55	280
118	7	195	435
118	8	186	417
118	10	90	287
118	11	150	438
118	12	232	454

Table 5.2: Dictionary of jump actions used for retrieving suitable values of the Number of FLANN trees to use.

#### 5.1.2.1 Results for a Walk Action

The input sequence and dictionary are the same as in Section 5.1.1.1. Figure 5.3 shows the result of the Mean Shift mode seeking for different numbers of leafs to visit. For other parameters, the following values were used:

To get comparable results, all other parameters are set to the following values:

- Number of trees to use=3
- Mean Shift window size=1.75
- Threshold for nearest neighbors=10

#### 5.1.2.2 Results for a Jump Action

For this experiment, the same parameter settings for the input sequence and dictionary as described in Section 5.1.1.2 are used. The parameter settings are equal to those in Section 5.1.2.1. Figure 5.4 also shows the sample density.

#### 5.1.2.3 Recommendation for the Number of Leafs to Visit

For later experiments a value of 8 leafs to visit while searching for nearest neighbor is used. As Figures 5.3 and 5.4 suggest, the parameter has influence on the result, especially the height of the maxima. It is also noticeable, that the influence on the height of maxima by different settings for this parameter is bigger than the influence of different number of trees to use. During a jump action, a parameter setting of 8 gives the highest maximum and for the walk action the maxima are very good too. The variance of the maxima depends on the action. This can be seen by regarding the result for 2 leafs to visit. In the walk action the results are smooth while during the jump action many small maxima occur. Therefore the setting for the number of leafs to visit was chosen by regarding the height of the maxima.

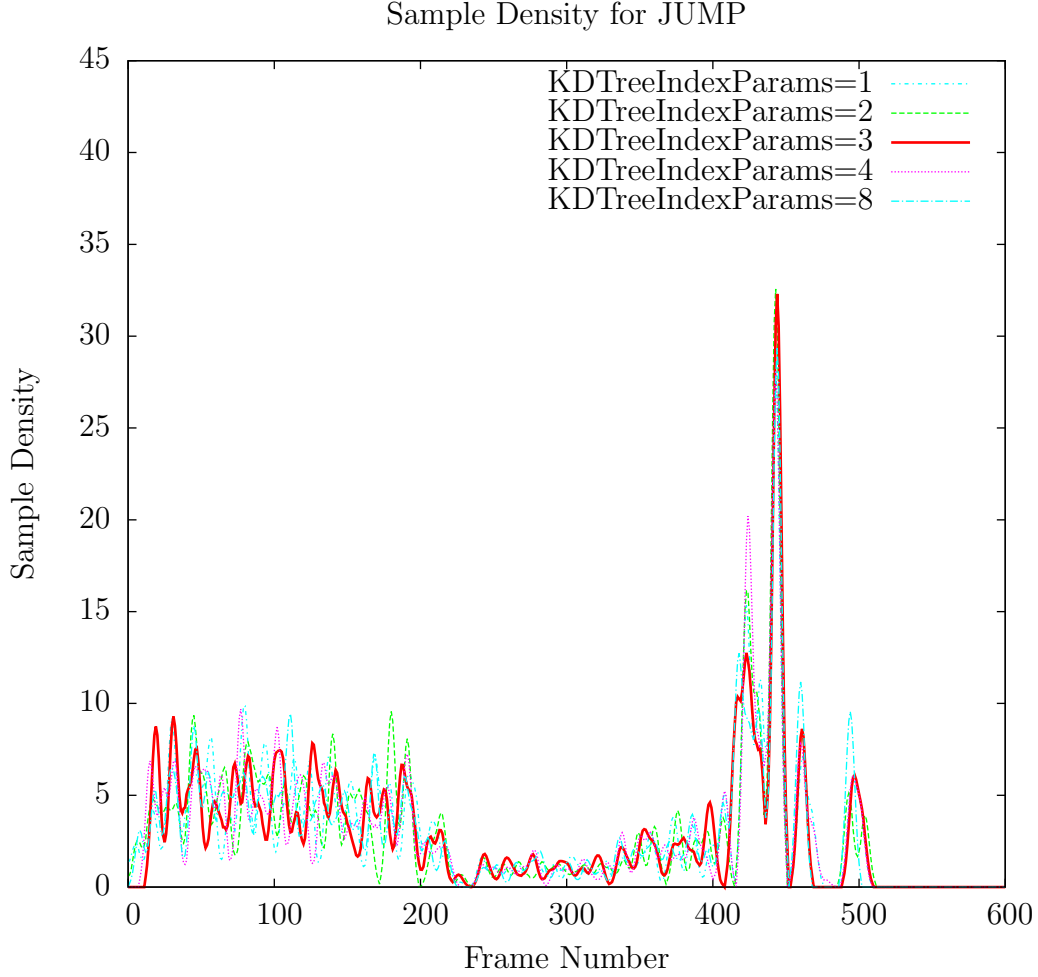


Figure 5.2: Sample density for different settings of the parameter `KDTreeIndexParams` for a JUMP action.

### 5.1.3 Threshold for FLANN Results

The nearest neighbor search with FLANN returns two values. One is the index of the nearest neighbor and the second one is the distance to this neighbor. When using *kd-trees* which is true in our case, this distance is the squared Euclidean distance between two vectors. To find a good value for this threshold, we compared the distances of actions which meet the action in the dictionary and others who are completely different. The dictionary for this experiment holds one action, namely a WALK action from frame 77 to 215 out of Trial 7 from Subject 7 in the CMU Motion Capture database. The parameter settings are described in the following enumeration.

- Number of Trees to use=3



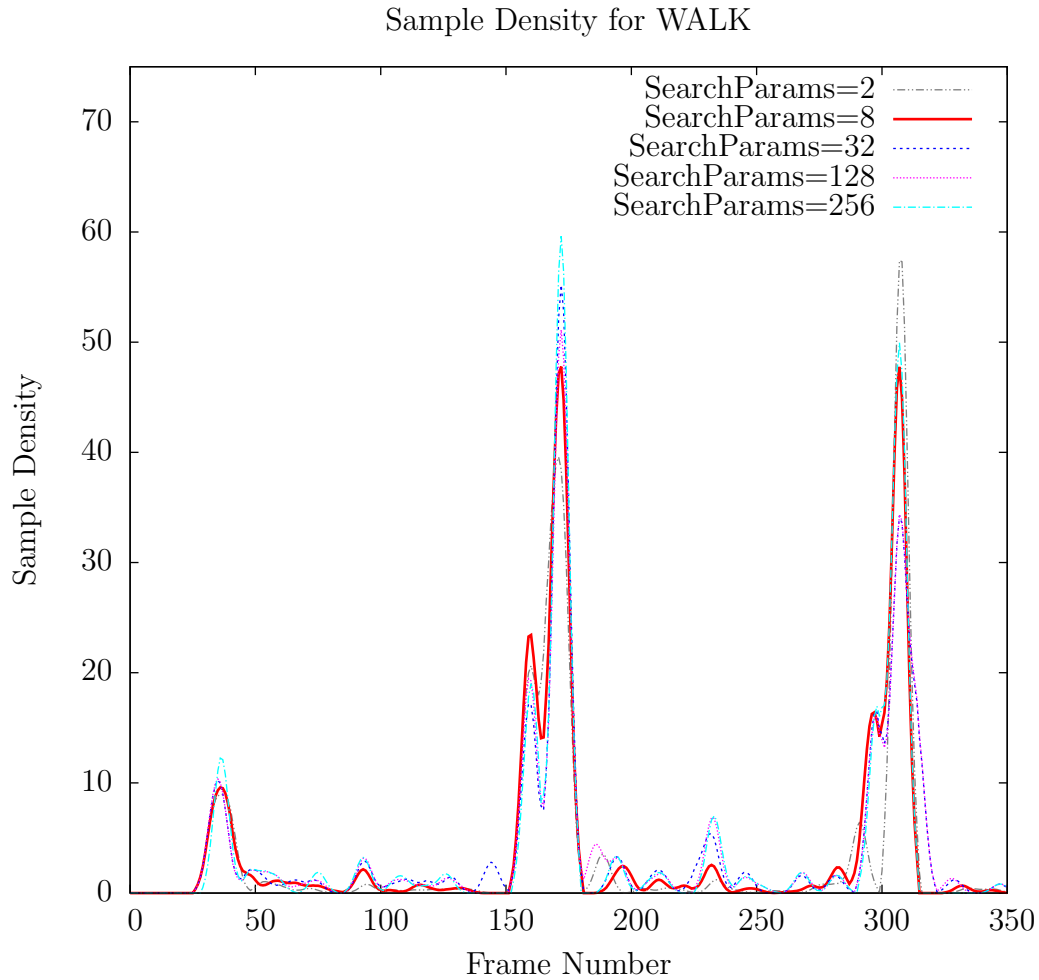


Figure 5.3: Sample density for different settings of the parameter SearchParams for a WALK action.

- Number of Leafs to visit=8
- Mean Shift window size=1.75

In Figure 5.5 the distances of four input sequences are drawn as dots. Two of these actions were walk actions with green (subject 2, trial 1) and black (subject 7, trial 6) dots and very low distances because of the walk action in the dictionary. For finding a good threshold, we have taken two samples with completely different actions. These are a jump action (red color, subject 128, trial 7) and a wave action (green color, subject 141, trial 16). The horizontal lines in the corresponding colors represent the mean of the distances for each action.

As a consequence of these results, we have set the threshold value to 10. The grey line in Figure 5.5 indicates that most of the distances from a walk action are below this

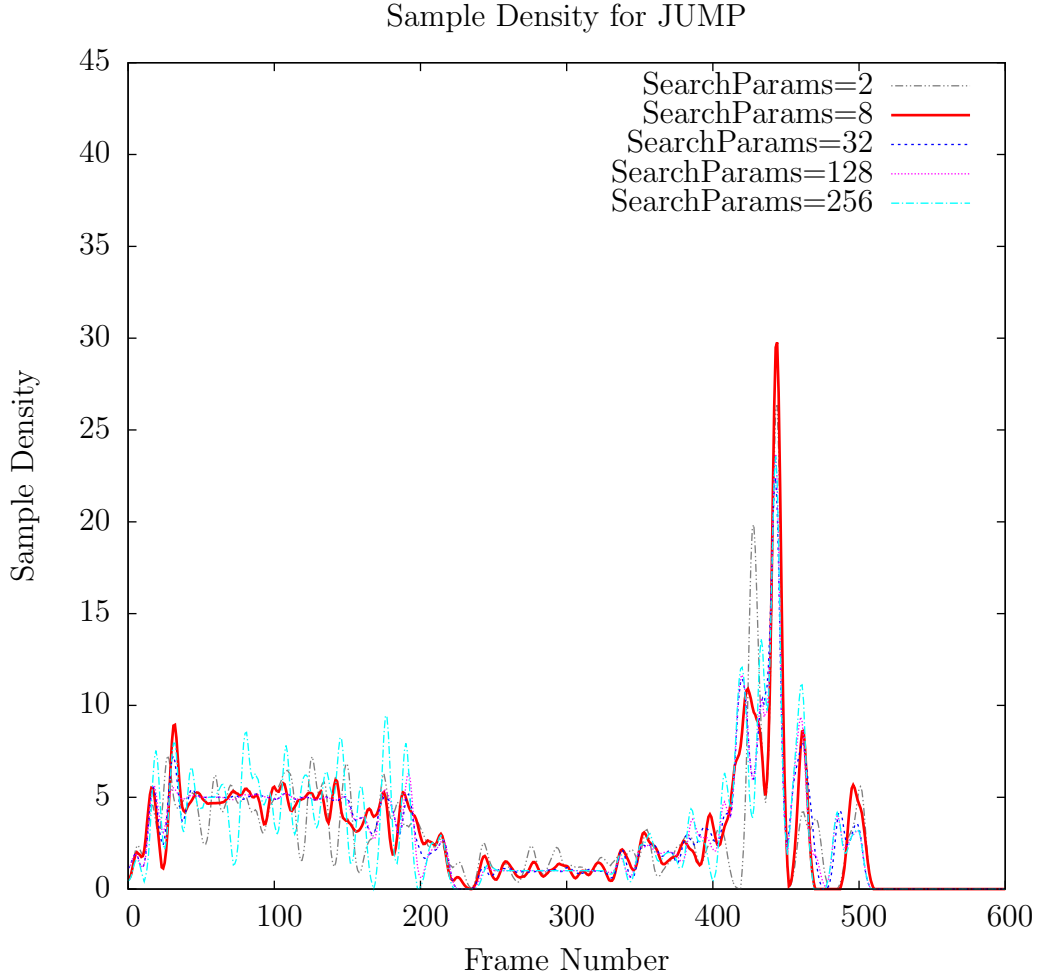


Figure 5.4: Sample density for different settings of the parameter SearchParams for a JUMP action.

threshold and the ones from different actions are over it, which is the desired outcome.

#### 5.1.4 Approximative versus Exact FLANN Results

The FLANN library also provides the option to search for exact nearest neighbors instead of using approximative algorithms. To compare the results, we use the following dictionary and parameter settings and run the action recognition process for both approximative and exact nearest neighbor results and compare the indices of the found nearest neighbors. The input file is Trial 6 from Subject 7 from the CMU Motion Capture database where 3 walking steps are performed.

To get comparable results, all other parameters are fixed during the experiment and set to the following values:

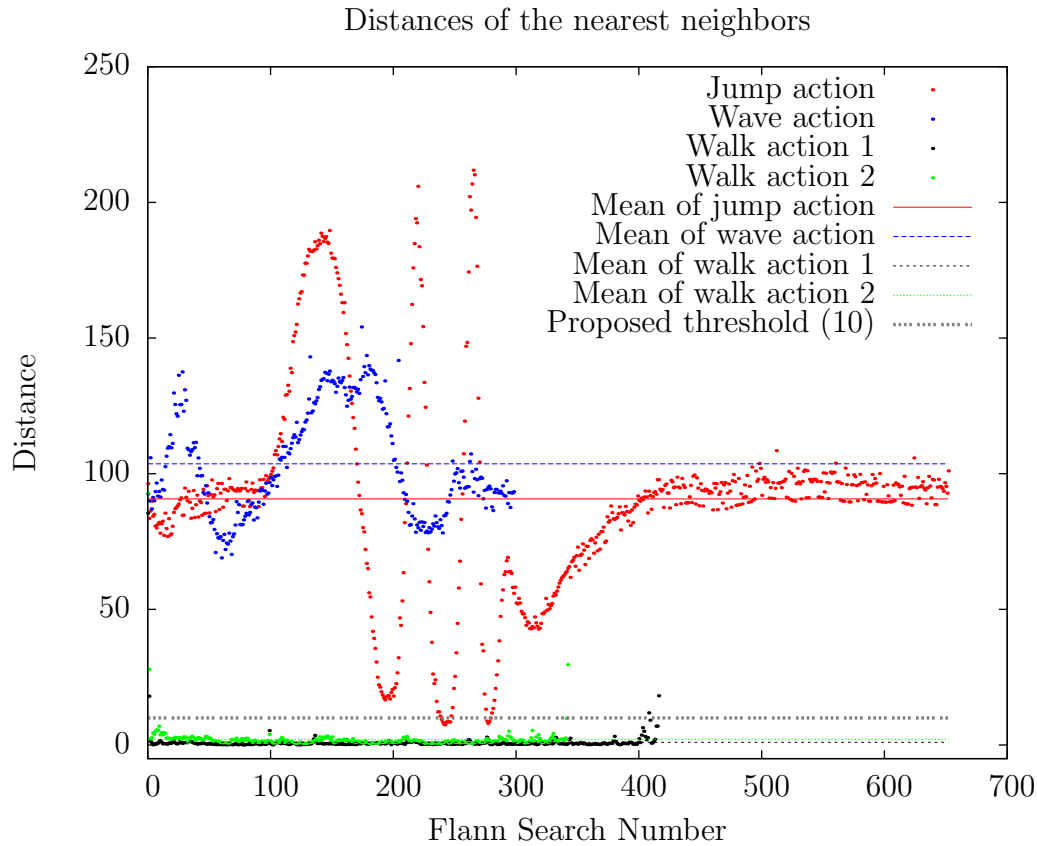


Figure 5.5: Distance of nearest neighbors with a walk action as training sequence and several input actions.

- Number of leafs to visit=8
- Number of trees to use=3

In Figure 5.6 the sample density based on both approximative and exact nearest neighbor search are compared. It can be seen, that the results are different but both methods generate characteristic maxima for the three walking actions. In this particular scenario, the approximative nearest neighbor search is even better than the exact one when regarding the last maxima. The measured speed up by using approximative nearest neighbor searches is in our case around 10 %.

### 5.1.5 Mean Shift Window Size

For the Mean Shift mode seeking a window size has to be set. This window size influences the maximum which is found. If it is very big, only one global maximum is found and if it is very small, many local maxima are detected. Therefore we have varied

Subject	Trial	Start Frame Number	End Frame Number
07	07	77	215
07	08	32	180
07	09	15	138
07	10	26	157
07	11	28	136
07	12	13	135

Table 5.3: Dictionary of walk actions used for comparing approximative and exact FLANN results

Comparison of FLANN results for approximative and exact nearest neighbor search

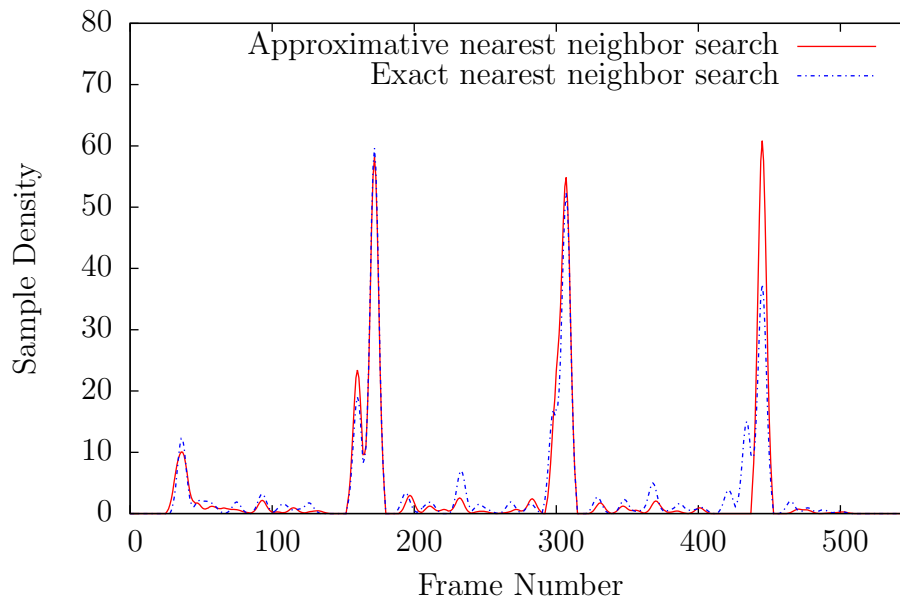


Figure 5.6: Sample density calculated with results of FLANN with an exact nearest neighbor search and an approximative nearest neighbor search.

the windows size from 0.1 to 5 to find a value which leads to good results regarding the sample density.

#### 5.1.5.1 Results for a Walk Action

Figure 5.7 shows the result of the Mean Shift mode seeking for different window sizes. For this purpose we have used the same training samples for walk as in Section 5.1.1.1. The other parameters were set to the following values:

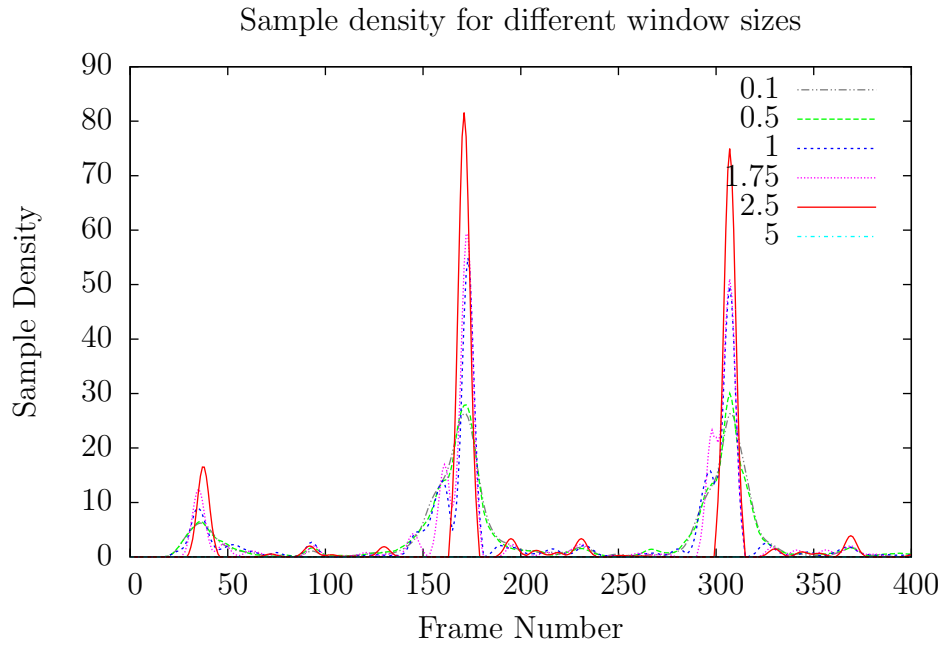


Figure 5.7: Sample density for different settings of the Mean Shift window size for a WALK action.

- Number of trees to use=3
- Number of leafs to visit=8
- Threshold for nearest neighbors=10

#### 5.1.5.2 Results for a Jump Action

For this experiment, the same environment settings for the input sequence and dictionary as described in Section 5.1.1.2 are used. The parameter settings are equal to those in Section 5.1.5.1. Figure 5.8 also shows the result of the Mean Shift mode seeking.

#### 5.1.5.3 Recommendation for the Mean Shift window size

One result of the comparison of the sample Density for different window sizes is, that a small window size of 0.1 leads to lower maxima which is not desired. In Figure 5.7 the result for a window size of 2.5 has the highest maxima and in Figure 5.8 it has the second highest maxima. A value of 2.5 for the Mean Shift Window size is therefore recommended. One side effect of a smaller window size is that Mean Shift converges faster because less iterations are needed.

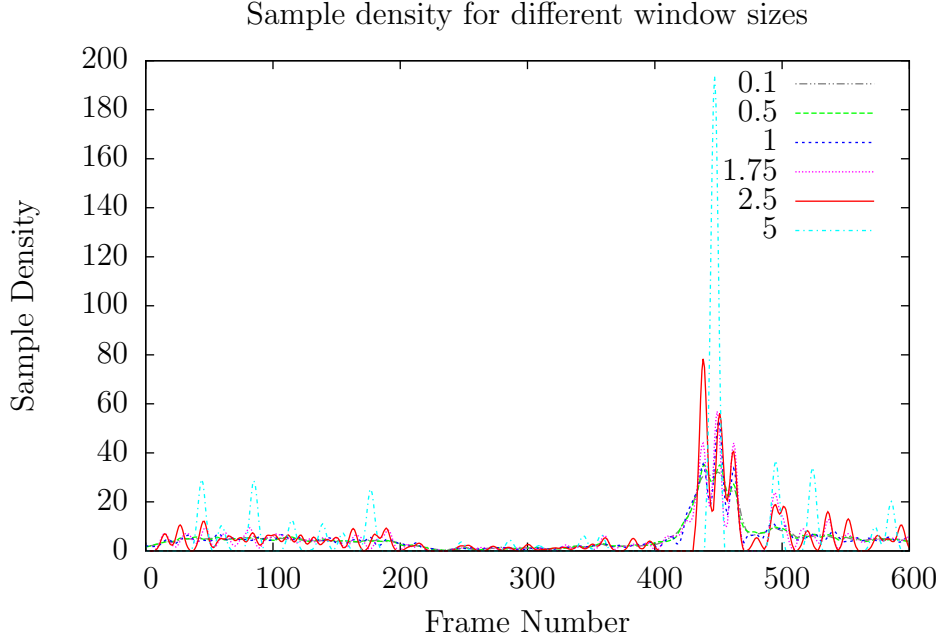


Figure 5.8: Sample density for different settings of the Mean Shift Window size for a JUMP action.

### 5.1.6 Threshold for Getting the Action Statement

For determining a threshold to select maxima which are sufficient to get an action statement, a value of 20 for the constant factor  $z$  (see Equation (3.29)) leads to good recognition results. It can be chosen this way because the maxima are very characteristic for actions as can be seen in Figure 5.9 and 5.10. With this value, also false detections are prevented because weaker maxima are disregarded. This can be seen in later experiments. In both scenarios six training samples were used and therefore the action statement threshold would be 120 and correct statements can be made.

The final action statements with a threshold as described above are shown in Figure 5.11(a) and 5.11(b).

## 5.2 Dictionary with Different Actions

In this experiment the action recognition approach is tested with a dictionary of four different actions (walk, run, jump and sit down). The results of the framework for input data which also includes all of the mentioned actions are compared and evaluated.

The images in Table 5.5 show, that walk actions are recognized very well. The gaps between the statements for walk correspond with the number of steps the subject performs and are due to the fact that the maxima increase with time. If one maximum

Subject	Trial	Start Frame Number	End Frame Number	Action label
118	6	55	280	jump
118	7	195	435	jump
118	8	186	417	jump
127	9	2	77	run
127	10	4	71	run
127	11	0	72	run
7	7	77	215	walk
7	8	32	180	walk
7	10	15	138	walk
86	9	986	1318	sit
114	5	1841	2170	sit
14	27	3177	3528	sit

Table 5.4: Dictionary of 4 actions which is used to compare results for different input actions.

is detected, the next one is below the action threshold for a time period. To avoid such alternating action statements, it is possible to hold the action statements a short period even if no maximum is above the threshold.

Other results like the one for the input from subject 118, file 29 of the CMU Motion Capture Database show that not all actions can be separated well from other actions which are in the training sample. This is caused by natural similarities of jump and sit actions for example. But the action statement for jump is still the strongest one because the maxima of the sample density for jump are three to four times higher than the one from sit and run. With a ‘winner takes all’ strategy for the action statement only the correct statement would be made.

### 5.3 Influence of Body Parts

In this experiment the influence of regarding the movement only of specified limbs and body parts is shown. To recognize run actions, we only incorporated the angular data from the feet into the descriptor because with values from all limbs action recognition fails. In Table 5.6 the results from both variants of the descriptor are compared.

The difference is caused by the various movements of the arms and upper body which is too different between the sample action sequences. This results in nearest neighbors which are disregarded due to their high distance to training samples. At the end of the

Input	Statement for walk	Statement for jump	Statement for run	Statement for sit
Subject 2 File 1 walk 3 steps				
Subject 7 File 1 walk 3 steps				
Subject 75 File 17 sit				
Subject 118 File 29 jump				

Table 5.5: Action Statements for various input files. The dictionary contains three samples of each action listed above. The first and second row contain the action statements for two jump actions with three steps in the input file. The third input action contains one sit action and the fourth one jump. Of course, the input actions are different from the learned samples.

input sequence for run also walk actions were detected. But as for the jump actions in Section 5.2, the maxima of the run actions value is  $1.5\times$  higher than the one from the walk action.



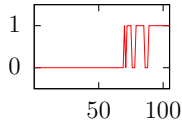
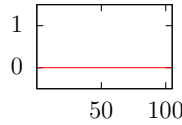
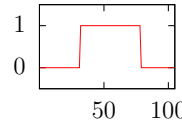
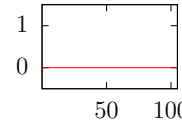
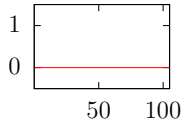
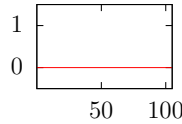
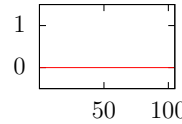
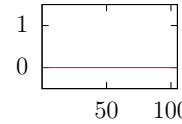
Input	Statement for walk	Statement for jump	Statement for run	Statement for sit
Subject 127 File 7 run only feet				
Subject 127 File 7 run				

Table 5.6: In the first row action statements for a run action where only values from the feet are in the descriptor are shown. The second row contains the action statements of the complete descriptor and it can be seen, that no statements were made if angles from all limbs are included.

## 5.4 Influence of the Derivations of Relative Angles in the Descriptor

In this experiment the influence of including the first and second derivation into the descriptor is shown. In this experiment the same dictionary as in Section 5.2 was used. The input file was file 1 from subject 7.

The result in Figure 5.12 shows that the maxima are influenced by the first and second derivatives. The best result is achieved if all derivatives are included in the descriptor. The influence of the first derivatives is higher than the second derivative. This is caused by the relative small numbers in the second derivative.

If no derivatives were included, also false positive action statements for run and jump actions were made. This can also be seen in Figure 5.13. The input sequence only contains walk actions and therefore, no recognitions for other actions should be made. With only relative angles in the descriptor, there were also maxima which lead to positive action statements for jump, run and sit which can be seen in the Figures 5.13(a), 5.13(b) and 5.13(c). This indicates that the derivatives of relative angles, especially the first derivative, are important and they are needed in the action descriptor.

## 5.5 Robustness

To test the robustness of the action recognition framework we use the same training sets and input data as in Section 5.2. The input data is corrupted with normally distributed additive noise with zero mean and different standard deviations. Because the values in the descriptor are expressed in radians, we can interpret a standard deviation of  $\sigma = 0.1$  as 5.7 degrees for the relative angles for example. The results are compared to those from undisturbed data and the standard deviation is increased until recognition fails.

### 5.5.1 Results for a Walk Action

For this experiment we have taken file 1 from subject 7 as the input file. The results of the action statements for the original data are shown in Table 5.5.

In Figure 5.14(a) and Figure 5.14(b), the different results of the action statement and sample density for walk are shown. With a noise of  $\sigma = 0.1$  ( $\approx 5.7^\circ$ ) the results are very similar to those from the original data. With  $\sigma = 0.2$  ( $\approx 11.5^\circ$ ) and  $\sigma = 0.25$  ( $\approx 14.3^\circ$ ), fewer and fewer action statements were made and also the height of the maxima decreases. Also there were no false classifications for other actions than walk in the action statements at all noise levels up to  $\sigma = 0.25$ . At a noise level of  $\sigma = 0.3$  ( $\approx 17.2^\circ$ ) there were no recognitions at all.

### 5.5.2 Results for a Jump Action

The recognition results for the jump action in Section 5.2 showed beside the correct recognition of sit also a short recognition of sit and run. Therefore we have also evaluated the influence of noise on such results. The results can be seen in the Figures 5.15(a) and 5.15(b). In this case with the results are comparable to those from Section 5.5.1. With increasing noise, the recognitions are getting shorter and shorter until a noise of  $\sigma = 0.3$  where no recognitions were made.

## 5.6 Summary

In the last sections, the action recognition framework was evaluated. Several parameters were determined to get satisfying results. It was shown, that the framework is able to detect actions like walk in input sequences. Also the robustness against a certain level of noise was evaluated. Special cases like running actions, where the action descriptor needs to be adjusted, were also considered. The next Chapter sums the work up and gives an outlook for possible future work.

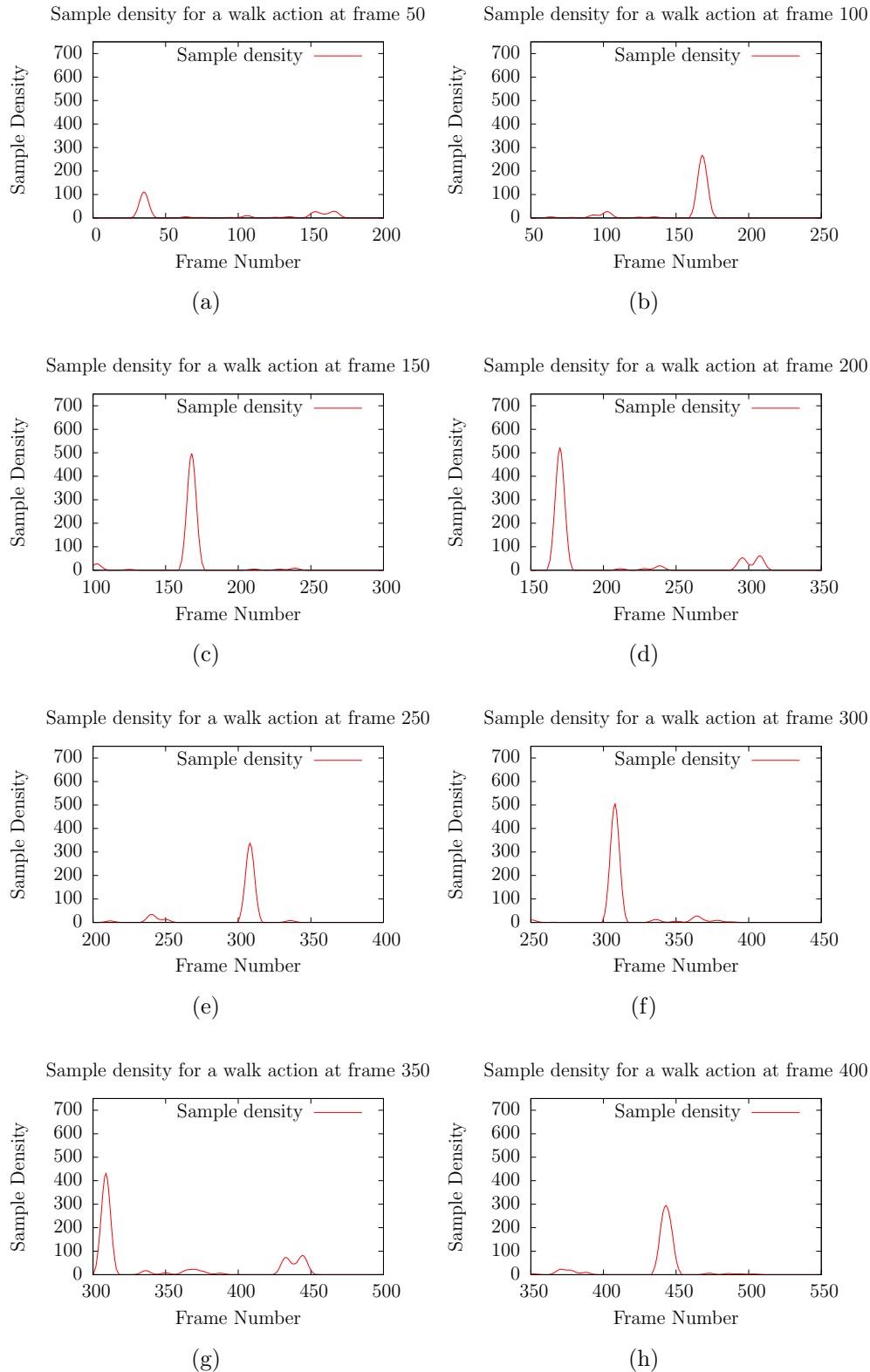


Figure 5.9: The Figures 5.9(a) to 5.9(h) are showing the sample density over time from a walk action (subject 7, trial 6) and the evolution of maxima. Here, only the part which is taken into account for detecting maxima above the threshold is shown (see Section 3.2.7). In this example, the subject performs three steps and therefore three maxima at frame 170, 310 and 440 where the walk action is finished are noticeable. The dictionary contained six walk actions and the action threshold would be 120 in this case.

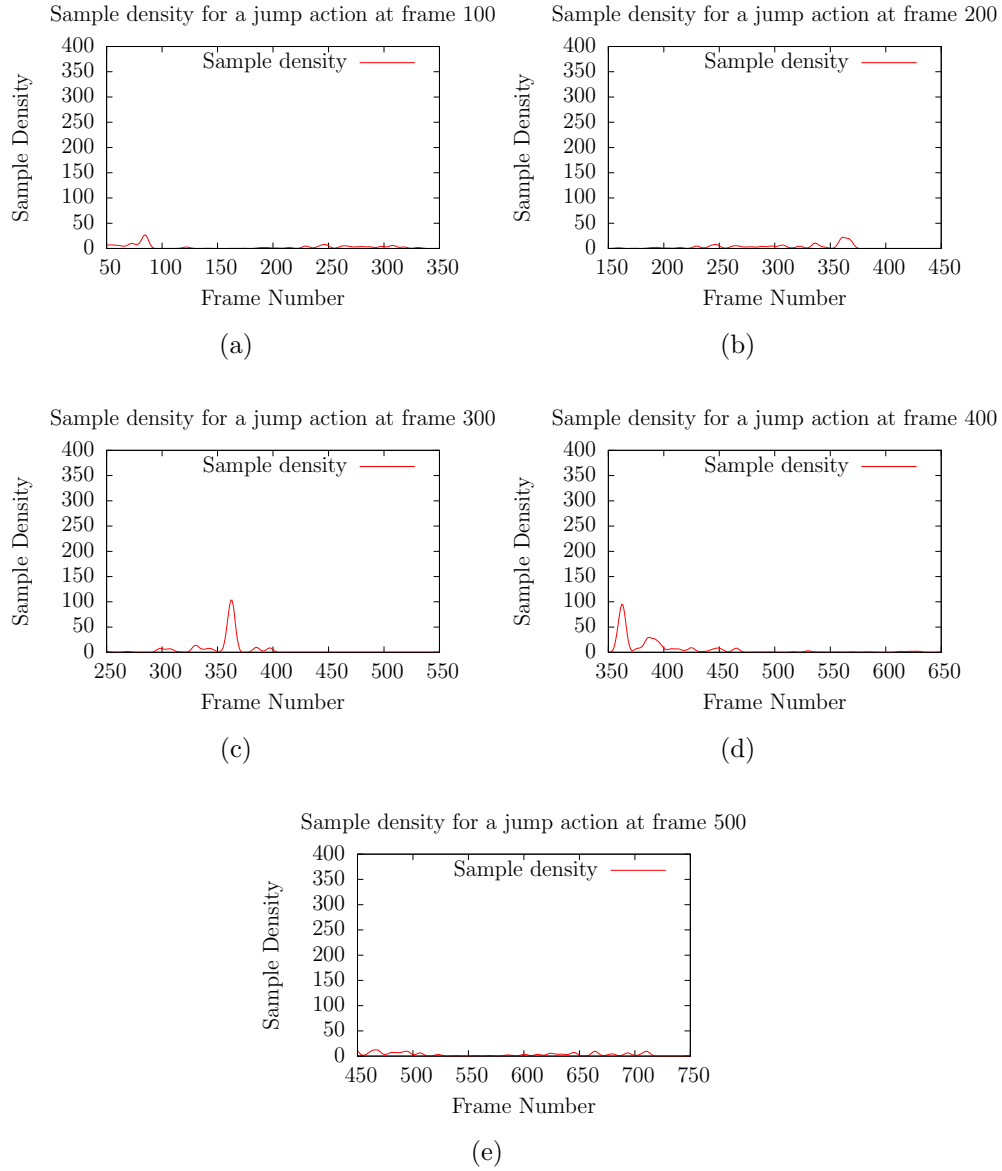


Figure 5.10: The Figures 5.10(a) to 5.10(e) are showing the sample density over time from a jump action (subject 118, trial 15) and the evolution of maxima. In this example, the subject performs one jump and therefore one maximum at frame 450 where the jump action ends is noticeable. The dictionary contained three jump actions and the action threshold would be 60 in this case.

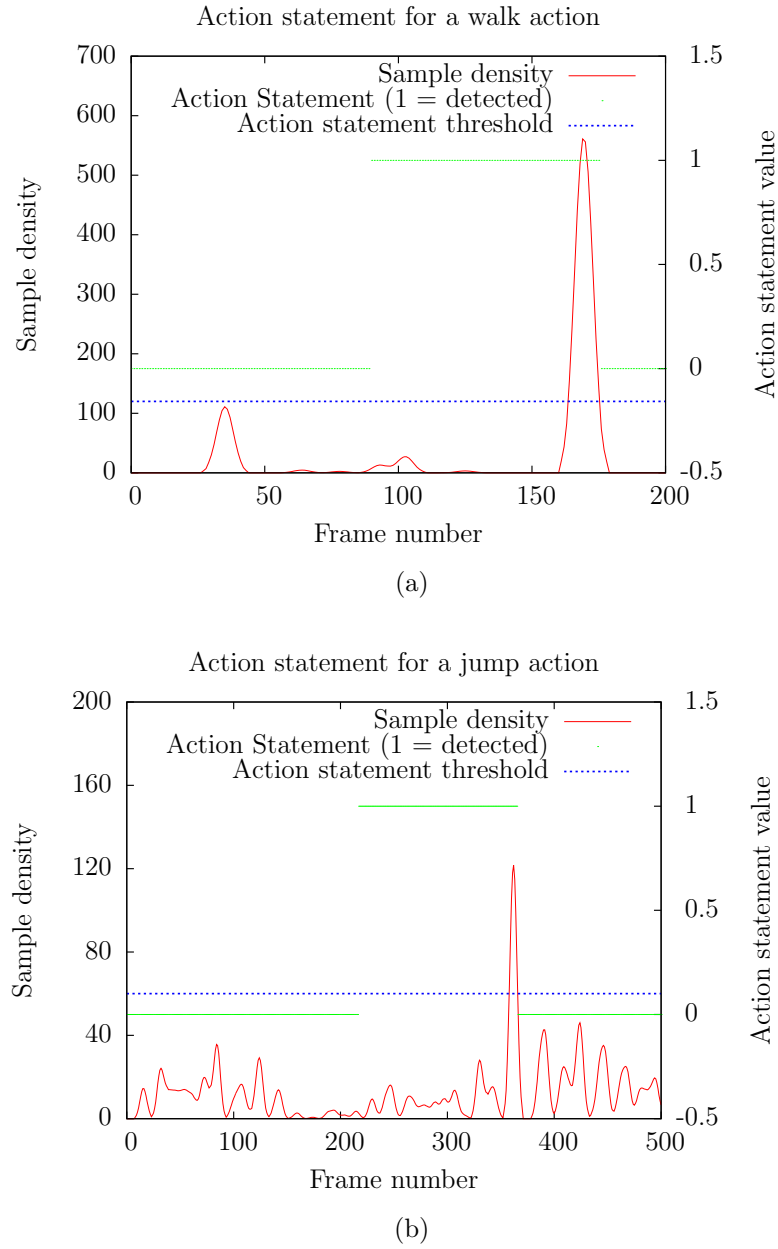


Figure 5.11: (a) shows the action Statement generated out of the results from frame 0 to 200 as shown in Figure 5.9. (b) shows the action Statement from the sample density as shown in Figure 5.10. In these Figures, also the Sample density is plotted to show the correspondence between action statement and Sample density.

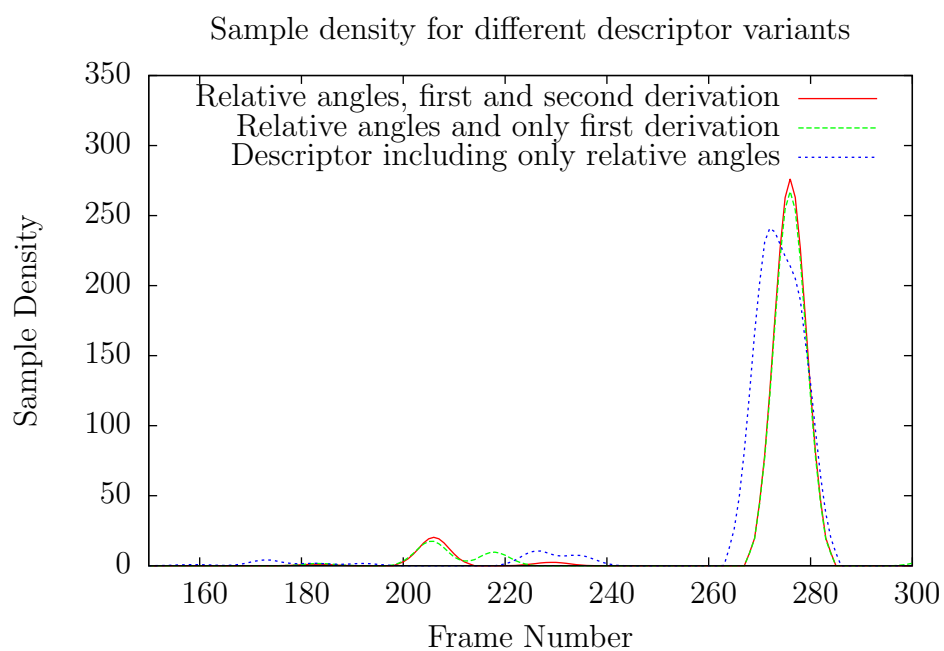
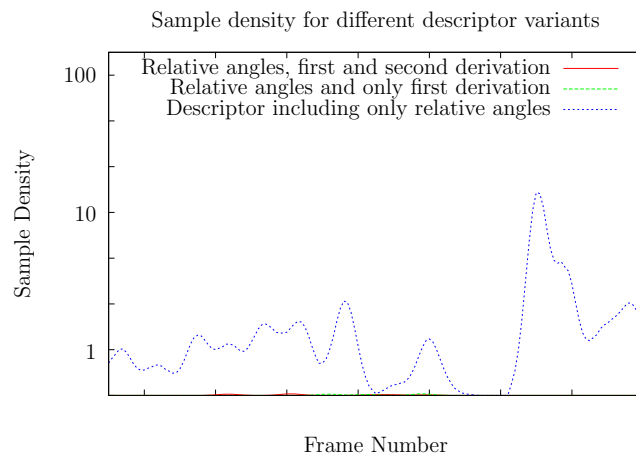
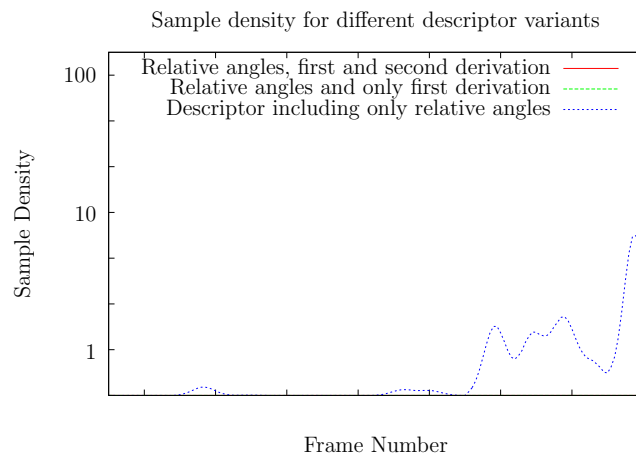


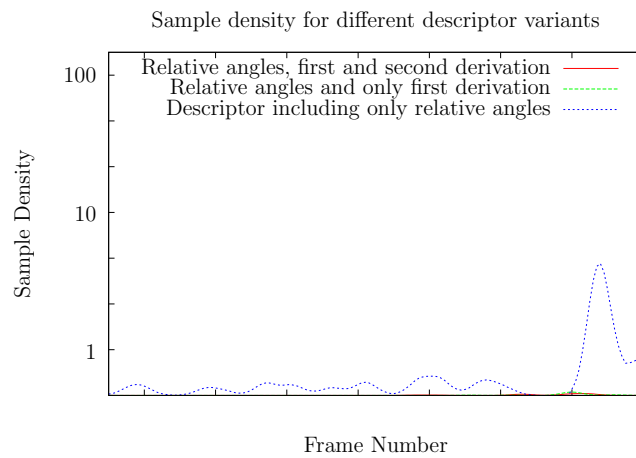
Figure 5.12: Sample density for walk with different descriptor variants. Because the input sequence is a walk action, higher maxima are better in this case.



(a)



(b)



(c)

Figure 5.13: (a): Sample density for jump. Lower values are better because the input sequence only contains walk actions. (b): The sample density for run, also lower values are better. (c): Mean Shift result for sit.

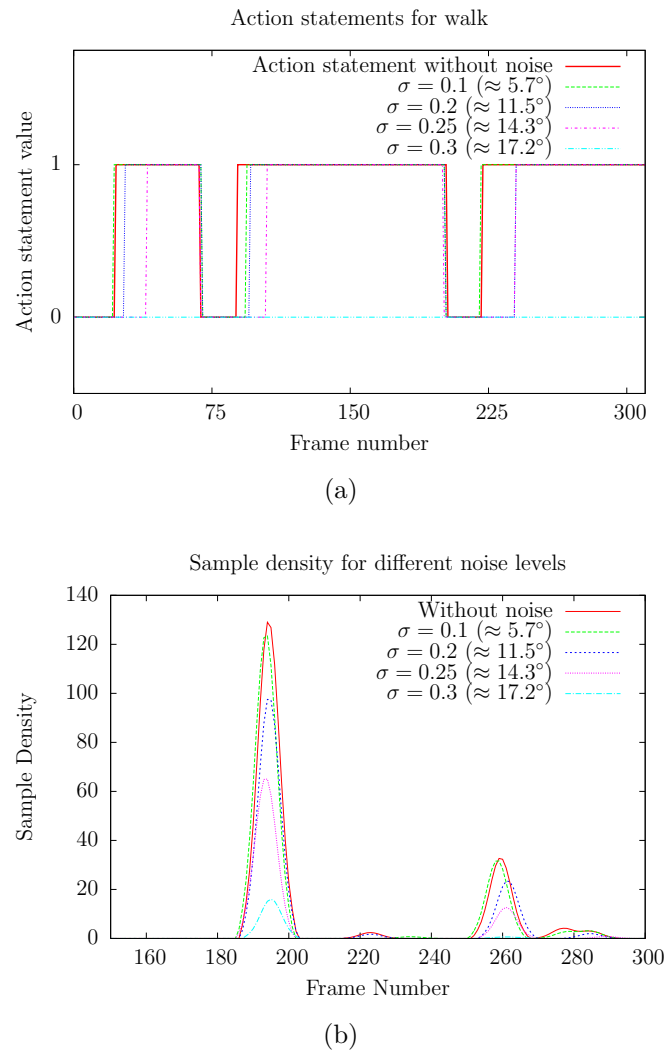


Figure 5.14: (a): Comparison of the action statements with different noise levels. (b): Comparison of the sample density with different noise levels.



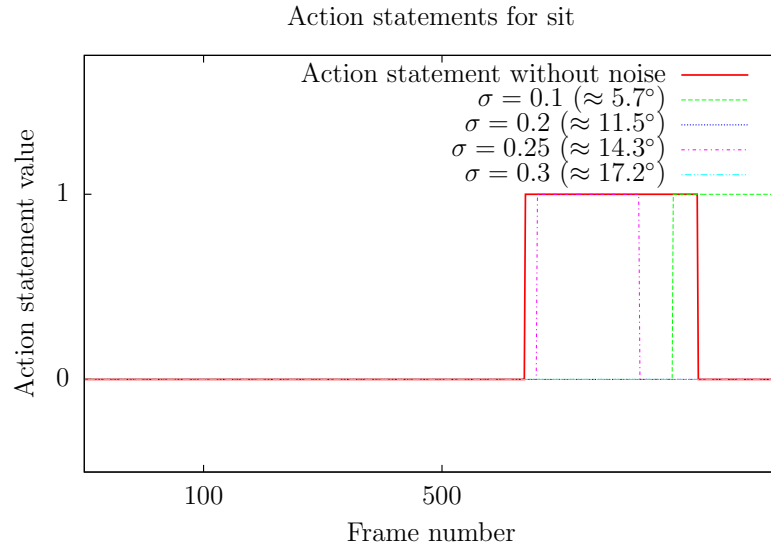
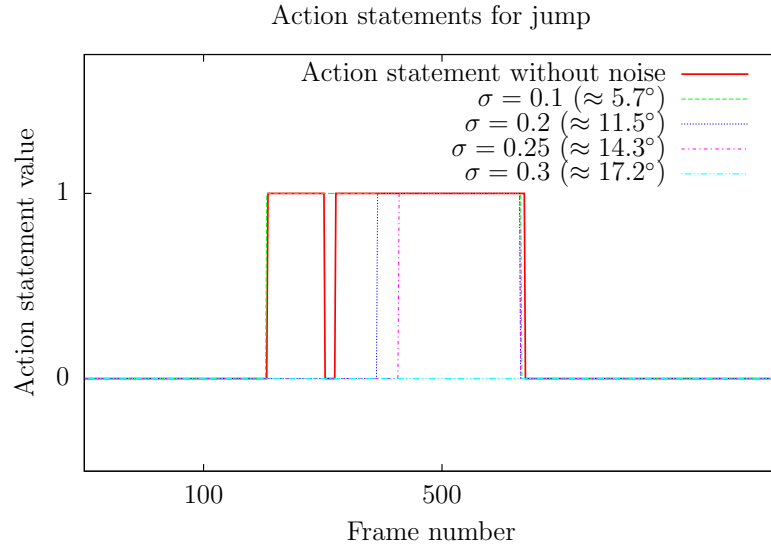


Figure 5.15: (a): Comparison of the action statements for the sit action with different noise levels. (b): Comparison of the action statements for the jump action with different noise levels.

# Chapter 6

## Conclusion

### 6.1 Summary and Results

In this thesis, an action recognition framework was described and evaluated. This framework consists of several parts which are the transformation of human poses as 3D input data into a human body model and the action recognition process based on the *Implicit Shape Model*.

In Chapter 2 an overview of the existing approaches was given. This includes the temporal and spatial action representations as well as the segmentation of actions. Also, methods to achieve view invariance are described.

This chapter is followed by Chapter 3 where the basics of the action recognition framework were presented. These are the human body model including the necessary steps for transforming 3D input data to the model and calculating angular values of the limbs. Based on these angular values, the *Implicit Shape Model*-based action recognition process was introduced in detail. This covers the descriptor for representing actions, the codebook and the voting process, the Mean Shift mode seeking and the generation of the final action statements. Details of the implementation of these steps were given in Chapter 4.

Subsequent to the basics and implementation, the approach was evaluated with several experiments. With a suitable parameter setting, basic action recognition was possible and very promising results for walking and jumping actions were received. Several illustrations of the Mean Shift mode seeking results and the consequential action statements were given. Also, the robustness and performance of the approach for separating between different actions and dealing with noisy data was investigated. During this phase, we also discovered that recognizing running actions works only if parts of the body are regarded due to the various ways of arm movements in the available training sequences. Also, the similarities of actions like jump and sit down are mentioned because they result in recognitions of both actions.

## 6.2 Future Work

The only source of 3D input data for action recognition in this work is the CMU Motion Capture database. Though this database provides a variety of actions, more data to test and evaluate the approach with different actions might be resulting in further improvements. For example, the framework can be tested with 3D poses provided by recording techniques as Microsoft Kinect. Other possible input data may come from 3D pose estimations out of 2D poses in real videos which was done in (Brauer & Arens 2011).

The action recognition approach is not yet implemented efficient enough to perform in real time. Up to now, the runtime depends heavily on the number of votes and the resulting Mean Shift iterations. The question is if it will perform in a real time mode on recent hardware or if we have to wait for faster devices. As the number of available CPU is increasing steadily and the implementation has still optimization potential more efforts can be taken into this direction.

Another idea is to consolidate several training samples into one by fitting them to the same length and calculating the mean for example and evaluate the performance of these training samples. This would also resolve the problem of temporal invariances in the training samples but then the input sequences can still be in different temporal scales.

The training samples may also be clustered which would also reduce the amount of votes and therefore improves the performance. The question is if the approach performs well with a lower amount of votes.

Also, the descriptor may be altered by using other values than the different ones to determine the strength of angular data for action recognition. The fact that running actions are only recognizable when the upper part of the body is disregarded (see Experiment 5.2) suggests, that using different descriptors may lead to better recognition results.

# List of Figures

1.1	Example output of an automatic video surveillance system which is able to detect groups of persons and deduce their behaviors (Zaidenberg et al. 2011).	1
2.1	(a): Action recognition based on few light bulbs as dots by (Johansson 1973). (b): 3D Body model based on cylinders proposed by (Marr & Nishihara 1978).	5
2.2	2D stick figure generated by line fitting to a silhouette (Niyogi & Adelson 1994).	6
2.3	Silhouettes of a running person (top) and block-based feature representation which is used as a descriptor (Wang & Suter 2007).	7
2.4	(a): The optical flow of a clapping action. The area where the main action takes place is marked as a blob (Cutler & Turk 1998). (b): Optical flow of a far away football player divided into its $x$ and $y$ components and then into four non-negative channels (Efros et al. 2003).	7
2.5	Distributions of gradients of different motion patterns (a) to (f) displayed as a smoothed histogram (g) (Zelnik-Manor & Irani 2001).	8
2.6	Local space time interest points of waving at different frequencies (Laptev & Lindeberg 2003).	9
2.7	Adding structure to local features by introducing a part layer (Niebles & Fei-Fei 2007).	9
2.8	Two person interaction and tracked body parts which can be analyzed with Bayesian Networks (Park & Aggarwal 2003).	10
2.9	Bobick and Davis uses two images as action templates: The binary Motion Energy image (MEI) and the scalar valued Motion History Image (MHI) (Bobick & Davis 2001).	11
2.10	Action recognition process based on snippets (Schindler & van Gool 2008). This Figure shows the process of matching results from 1D Gabor filtering (upper half) and optical flow (lower half) to form and flow templates for a final action statement.	12
2.11	Different types of SIFT descriptors. From left to right: original 2D SIFT descriptor, 2D SIFT for videos, 3D SIFT descriptor (Scovanner et al. 2007).	13

2.12	The vertical axis shows values of the characteristic difference between the optical flow and the mean optical flow of foreground pixels which can be taken into account for the segmentation of movements (Ogale et al. 2004).	14
2.13	Network of HMMs used for action recognition. One HMM with 3 hidden states exist for every action and every feature. When the parameters of each HMM are learned, the most probable action sequence can be computed with the forward algorithm (Lv & Nevatia 2006).	15
2.14	Generated new views from three completely different real camera views (Bodor et al. 2003).	16
2.15	Generation of multiple views for training data. The exhaustive search is performed on such a dataset to find the best match (Ahmad & Lee 2006).	17
2.16	Examples for an exhaustive search method to match an observed silhouette to silhouettes generated from a 3D prototype. In this figure the best match is shown in the middle row (Weinland et al. 2007).	18
2.17	Detected key frames for a walk action. (González 2004).	19
2.18	Pipeline for action recognition using the Implicit Shape Model (Thi et al. 2010).	19
3.1	(a): Stick figure model with 12 limbs and 15 joints. (b): Placement of the 41 markers used for creating data for the CMU Graphics Lab Motion Capture Database (CMU n.d.c). These positions are transferred to the stick figure model (CMU n.d.a).	22
3.2	Visualization of the angles of the rotation matrix with a local coordinate system $(x, y, z)$ and a global coordinate system $(X, Y, Z)$ .	23
3.3	Visualization of a rotation with the angles $(\alpha = 90^\circ, \beta = 90^\circ, \gamma = 15^\circ)$ with the $Zy'z''$ convention.	24
3.4	Angle values jump between $-\pi$ and $\pi$ which is a problem in the computation of a human body model and should be avoided (Ferrer 2005).	25
3.5	A limb of the stick figure model described with 3D polar coordinates as described in (Ferrer 2005).	27
3.6	Visualization of a relative angle $\tilde{\phi}_{leg}$ between two limbs.	27
3.7	(a): Frame 276 of a walk action (Subject 139, Trial 30) from the CMU Motion Capture database. (b): Frame 833 of the same walk action in world coordinates. (c): Frame 276 of the same walk action in normalized coordinates. (d): Frame 833 of the same walk action in normalized coordinates.	31
3.8	This figure shows the calculated absolute and relative angle values of $\phi$ over time of the left and right foot from a walk action (Subject 139, Trial 30) of the CMU Motion Capture database (a): Left foot relative angle $\phi$ . (b): Right foot relative angle $\phi$ . (c): Left foot absolute angle $\phi$ . (d): Right foot absolute angle $\phi$ .	33

3.9	The ISM training phase with the extraction of interest points (yellow circles), the clustering and generation of mean samples called prototypes and the final codebook which contains the prototypes and the spatial occurrence of these prototypes. These positions are relative to the objects center (Leibe et al. 2008). . . . .	34
3.10	The ISM recognition process as proposed in (Leibe et al. 2008). . . . .	34
3.11	Visualization of the values in the descriptor for a walk action. The image is a greyscale image with values between 0 and 255. To show the hip values clearly, the colors of the hip and its derivations are set to darker values. . . . .	35
3.12	Priority based nearest neighbor search in a <i>kd-tree</i> as proposed in (Silpa-Anan & Hartley 2008). First, the tree is descended until the query point, which is labeled with a 1, is reached. Then, the neighbor cells are searched, which are the nodes 2 to 5 in this image. The boundaries of the search are spherical with an adjustable radius. . . . .	37
3.13	Visualization of hierarchical k-means trees of 100000 data points with different branching factors. Gray values indicate the distance between the nearest and second nearest center (Muja & Lowe 2009). . . . .	38
3.14	Mean Shift trajectories for different kernels and different procedures, each trajectory shows the process of shifting the window towards the local mode (Cheng 1995). (a) is the input dataset and b) - f) are the results with varying cluster size and number of iterations. In (c) and (d) truncated Gaussian kernels were used and in (e) and (f) non-truncated Gaussian kernels were chosen. . . . .	40
3.15	Iterations of the Mean Shift procedure to find global maxima. a) shows the function and b) - e) are the four iterations. The bars are the shifted data points from a) and the curve in b) - e) is a function whose local maxima approximate the maxima of the original function. In e) the two maxima are found (Cheng 1995). . . . .	41
3.16	Example of an application of Mean Shift in Computer Vision. As a result, homogeneous regions are found within an image and these regions can be segmented very easy (Comaniciu & Meer 2002). . . . .	42
4.1	The processing pipeline for transferring 3D data to our human body model and the action descriptor. . . . .	43
4.2	The tools for generating a dictionary entry. . . . .	44
4.3	The processing pipeline for the action recognition framework. . . . .	45
4.4	The multithreading concept which is used by OpenMP. . . . .	46
5.1	Sample density for different settings of the parameter <code>KDTreeIndex-Params</code> for a WALK action. . . . .	50
5.2	Sample density for different settings of the parameter <code>KDTreeIndex-Params</code> for a JUMP action. . . . .	52

5.3	Sample density for different settings of the parameter SearchParams for a WALK action. . . . .	53
5.4	Sample density for different settings of the parameter SearchParams for a JUMP action. . . . .	54
5.5	Distance of nearest neighbors with a walk action as training sequence and several input actions. . . . .	55
5.6	Sample density calculated with results of FLANN with an exact nearest neighbor search and an approximative nearest neighbor search. . . . .	56
5.7	Sample density for different settings of the Mean Shift window size for a WALK action. . . . .	57
5.8	Sample density for different settings of the Mean Shift Window size for a JUMP action. . . . .	58
5.9	The Figures 5.9(a) to 5.9(h) are showing the sample density over time from a walk action (subject 7, trial 6) and the evolution of maxima. Here, only the part which is taken into account for detecting maxima above the threshold is shown (see Section 3.2.7). In this example, the subject performs three steps and therefore three maxima at frame 170, 310 and 440 where the walk action is finished are noticeable. The dictionary contained six walk actions and the action threshold would be 120 in this case. . . . .	63
5.10	The Figures 5.10(a) to 5.10(e) are showing the sample density over time from a jump action (subject 118, trial 15) and the evolution of maxima. In this example, the subject performs one jump and therefore one maximum at frame 450 where the jump action ends is noticeable. The dictionary contained three jump actions and the action threshold would be 60 in this case. . . . .	64
5.11	(a) shows the action Statement generated out of the results from frame 0 to 200 as shown in Figure 5.9. (b) shows the action Statement from the sample density as shown in Figure 5.10. In these Figures, also the Sample density is plotted to show the correspondence between action statement and Sample density. . . . .	65
5.12	Sample density for walk with different descriptor variants. Because the input sequence is a walk action, higher maxima are better in this case. . . . .	66
5.13	(a): Sample density for jump. Lower values are better because the input sequence only contains walk actions. (b): The sample density for run, also lower values are better. (c): Mean Shift result for sit. . . . .	67
5.14	(a): Comparison of the action statements with different noise levels. (b): Comparison of the sample density with different noise levels. . . . .	68
5.15	(a): Comparison of the action statements for the sit action with different noise levels. (b): Comparison of the action statements for the jump action with different noise levels. . . . .	69
A.1	The visualization of 3D input data. . . . .	84

A.2	The visualization of the relative angles and their derivations. . . . .	85
A.3	The management of the dictionary (upper part) and the starting of the action recognition process allowing the definition of the values which should be included (lower part). . . . .	86
A.4	The visualization of the distribution of the votes. . . . .	87
A.5	The visualization of the action statements. . . . .	88
B.1	The Qt Designer for creating interactive GUIs. . . . .	90
B.2	Example of the visualization of an OpenCV matrix with the <code>imshow()</code> function. The matrix in this image is a descriptor as mentioned in Section 3.2.3. . . . .	91
B.3	The OpenGL State Machine. Commands enter on the left and are processed through the pipeline. . . . .	92



# Bibliography

- Ahmad, M. & Lee, S.-W. (2006), Hmm-based human action recognition using multiview image sequences, *in* ‘18th International Conference on Pattern Recognition, 2006’, Vol. 1, pp. 263 –266. [17](#), [73](#)
- Blanchette, J. & Summerfield, M. (2006), *C++ GUI Programming with Qt 4*, Prentice Hall PTR, Upper Saddle River, NJ, USA. [89](#)
- Board, O. A. R. (2011), ‘Openmp application program interface’.  
**URL:** <http://www.openmp.org/mp-documents/OpenMP3.1.pdf> [47](#)
- Bobick, A. & Davis, J. (2001), ‘The recognition of human movement using temporal templates’, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **23**(3), 257 –267. [11](#), [17](#), [72](#)
- Bodor, R., Jackson, B., Masoud, O. & Papanikolopoulos, N. (2003), Imagebased reconstruction for view-independent human motion recognition, *in* ‘In Int. Conf. on Intel. Robots and Sys’, pp. 27–31. [16](#), [73](#)
- Brand, M. & Kettnaker, V. (2000), ‘Discovery and segmentation of activities in video’, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **22**(8), 844 –851. [15](#)
- Brand, M., Oliver, N. & Pentland, A. (1997), Coupled hidden markov models for complex action recognition, *in* ‘IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1997’, pp. 994 –999. [6](#)
- Brauer, J. & Arens, M. (2011), Reconstructing the missing dimension: From 2d to 3d human pose estimation, *in* ‘Proc. of REACTS 2011 - REcognition and ACTion for Scene Understanding - in conj. with CAIP 2011, 14th Intern. Conf. on Computer Analysis of Images and Patterns’, Spain, Málaga, pp. 25–39. [71](#)
- Campbell, L. & Bobick, A. (1995), Recognition of human body motion using phase space constraints, *in* ‘Fifth International Conference on Computer Vision 1995’, pp. 624 –630. [5](#)

- Carlsson, S. & Sullivan, J. (2001), Action recognition by shape matching to key frames, *in* ‘IEEE Computer Society Workshop on Models versus Exemplars in Computer Vision’. 12
- Cheng, J.-C. & Moura, J. M. F. (1999), Capture and representation of human walking in live video sequences, *in* ‘IEEE Transactions on Multimedia’, pp. 144–156. 21
- Cheng, Y. (1995), ‘Mean shift, mode seeking, and clustering’, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 17(8), 790–799. 39, 40, 41, 74
- CMU (n.d.a), ‘Cmu graphics lab motion capture database - frequently asked questions’.  
URL: <http://mocap.cs.cmu.edu/faqs.php> 22, 73
- CMU (n.d.b), ‘Cmu graphics lab motion capture database - informations’.  
URL: <http://mocap.cs.cmu.edu/info.php> 28
- CMU (n.d.c), ‘Marker placement guide’.  
URL: <http://mocap.cs.cmu.edu/markerPlacementGuide.pdf> 22, 73
- Comaniciu, D. & Meer, P. (2002), ‘Mean shift: A robust approach toward feature space analysis’, *IEEE Trans. Pattern Anal. Mach. Intell.* 24(5), 603–619. 40, 42, 74
- Cutler, R. & Turk, M. (1998), View-based interpretation of real-time optical flow for gesture recognition, *in* ‘Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on’, pp. 416–421. 6, 7, 72
- Dainis, A. (1987), ‘Original c3d specification’.  
URL: <http://www.c3d.org/pdf/C3D.pdf> 28
- Darrell, T. & Pentland, A. (1993), Space-time gestures, *in* ‘IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1993’, pp. 335–340. 6, 12, 15
- Dollar, P., Rabaud, V., Cottrell, G. & Belongie, S. (2005), Behavior recognition via sparse spatio-temporal features, *in* ‘Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on’, pp. 65–72. 8
- Efros, A., Berg, A., Mori, G. & Malik, J. (2003), Recognizing action at a distance, *in* ‘Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on’, pp. 726–733 vol.2. 7, 72
- Ferrer, I. R. (2005), Articulated 3d human motion modeling for tracking and reconstruction, Master’s thesis, Universitat Autònoma de Barcelona. 22, 23, 25, 26, 27, 73
- Friedman, J. H., Bentley, J. L. & Finkel, R. A. (1977), ‘An algorithm for finding best matches in logarithmic expected time’, *ACM Trans. Math. Softw.* 3(3), 209–226. 37

- Gavrila, D. M. & Davis, L. S. (1995), Towards 3-d model-based tracking and recognition of human movement: a multi-view approach, *in* ‘In International Workshop on Automatic Face- and Gesture-Recognition. IEEE Computer Society’, pp. 272–277. [5](#)
- Gavrila, D. & Philomin, V. (1999), Real-time object detection for smart vehicles, *in* ‘The Proceedings of the Seventh IEEE International Conference on Computer Vision 1999.’, Vol. 1, pp. 87 –93 vol.1. [6](#)
- Gilbert, A., Illingworth, J. & Bowden, R. (2008), Scale invariant action recognition using compound features mined from dense spatio-temporal corners, *in* ‘Proceedings of the 10th European Conference on Computer Vision: Part I’. [8](#)
- Gong, S. & Xiang, T. (2011), *Visual Analysis of Behaviour: From Pixels to Semantics*, Springer. [2](#)
- González, J. (2004), Human Sequence Evaluation: the Key-frame Approach, PhD thesis, Universitat Autònoma de Barcelona, Spain. [18](#), [19](#), [21](#), [31](#), [73](#)
- Guo, Y., Xu, G. & Tsuji, S. (1994), Understanding human motion patterns, *in* ‘Proceedings of the 12th IAPR International Conference on Pattern Recognition, 1994. Vol. 2’, Vol. 2, pp. 325 –329 vol.2. [12](#)
- Jüngling, K. (2011), Ein generisches System zur automatischen Detektion, Verfolgung und Wiedererkennung von Personen in Videodaten, PhD thesis, Fakultät für Bauingenieur-, Geo- und Umweltwissenschaften des Karlsruher Instituts für Technologie (KIT) (Institut für Photogrammetrie und Fernerkundung). [32](#), [35](#)
- Johansson, G. (1973), ‘Visual perception of biological motion and a model for its analysis’, *Attention Perception Psychophysics* **14**(2), 201–211. [1](#), [5](#), [72](#)
- Ke, Y., Sukthankar, R. & Hebert, M. (2005), Efficient visual event detection using volumetric features, *in* ‘Tenth IEEE International Conference on Computer Vision, 2005’, Vol. 1, pp. 166 – 173 Vol. 1. [15](#)
- Laganière, R. (2011), *OpenCV 2 Computer Vision Application Programming Cookbook*, Packt Publishing. [90](#)
- Laptev, I. & Lindeberg, T. (2003), Space-time interest points, *in* ‘Ninth IEEE International Conference on Computer Vision, 2003.’, pp. 432 –439 vol.1. [8](#), [9](#), [72](#)
- Leibe, B., Leonardis, A. & Schiele, B. (2008), ‘Robust object detection with interleaved categorization and segmentation’, *Int. J. Comput. Vision* **77**(1-3), 259–289. [19](#), [32](#), [34](#), [74](#)
- Lisin, D. A., Mattar, M. A., Blaschko, M. B., Benfield, M. C. & Learned-miller, E. G. (2005), Combining local and global image features for object class recognition, *in* ‘In Proceedings of the IEEE CVPR Workshop on Learning in Computer Vision and Pattern Recognition’, pp. 47–55. [4](#)

- Lowe, D. G. (2004), ‘Distinctive image features from scale-invariant keypoints’, *Int. J. Comput. Vision* **60**, 91–110. [8](#)
- Lv, F. & Nevatia, R. (2006), Recognition and segmentation of 3-d human action using hmm and multi-class adaboost, *in* ‘Proceedings of the 9th European conference on Computer Vision - Volume Part IV’. [15](#), [73](#)
- Marr, D. & Nishihara, H. K. (1978), ‘Representation and recognition of the spatial organization of three-dimensional shapes’, *Proceedings of the Royal Society of London Biological Sciences* **200**(1140), 269–294. [5](#), [14](#), [72](#)
- Marr, D. & Vaina, L. (1980), Representation and recognition of the movements of shapes, *in* ‘Proc. R. Soc. Lond. B 1982’, AI memo, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, pp. 501–524. [13](#)
- Muja, M. & Lowe, D. G. (2009), Fast approximate nearest neighbors with automatic algorithm configuration, *in* ‘International Conference on Computer Vision Theory and Application VISSAPP’09’, INSTICC Press, pp. 331–340. [36](#), [38](#), [74](#), [92](#), [95](#)
- Niebles, J. & Fei-Fei, L. (2007), A hierarchical model of shape and appearance for human action classification, *in* ‘IEEE Conference on Computer Vision and Pattern Recognition, 2007.’, pp. 1–8. [9](#), [72](#)
- Niyogi, S. & Adelson, E. (1994), Analyzing and recognizing walking figures in xyt, *in* ‘1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.’, pp. 469–474. [6](#), [72](#)
- Ogale, A. S., Karapurkar, A., Guerra-filho, G. & Aloimonos, Y. (2004), View-invariant identification of pose sequences for action recognition, *in* ‘In VACE’. [14](#), [73](#)
- Park, S. & Aggarwal, J. K. (2003), Recognition of two-person interactions using a hierarchical bayesian network, *in* ‘First ACM SIGMM international workshop on Video surveillance’, IWVS ’03, ACM. [10](#), [72](#)
- Rogez, G., Guerrero, J. J., Martínez, J. & Orrite-Uruñuela, C. (2006), Viewpoint independent human motion analysis in man-made environments., *in* M. J. Chantler, R. B. Fisher & E. Trucco, eds, ‘BMVC’, British Machine Vision Association. [16](#)
- Rohr, K. (1994), ‘Towards model-based recognition of human movements in image sequences’, *CVGIP: Image Underst.* **59**, 94–115. [5](#)
- Schindler, K. & van Gool, L. (2008), Action snippets: How many frames does human action recognition require?, *in* ‘IEEE Conference on Computer Vision and Pattern Recognition, 2008.’, pp. 1–8. [12](#), [72](#)
- Scovanner, P., Ali, S. & Shah, M. (2007), A 3-dimensional sift descriptor and its application to action recognition, *in* ‘Proceedings of the 15th international conference on Multimedia’, MULTIMEDIA ’07, ACM, New York, NY, USA. [13](#), [72](#)

- Segal, M. & Frazier, C. (2010), ‘The opengl graphics system : A specification’, *Group* **1**, 403.  
**URL:** <http://www.opengl.org/documentation/specs/> **91**
- Silpa-Anan, C. & Hartley, R. (2008), Optimised kd-trees for fast image descriptor matching, *in* ‘IEEE Conference on Computer Vision and Pattern Recognition, 2008.’, pp. 1 –8. **37, 74**
- Stahlke, D. (2009), ‘Quaternions in classical mechanics’.  
**URL:** <http://www.stahlke.org/dan/phys-papers/quaternion-paper.pdf> **25**
- Starner, T. & Pentland, A. (1995), Real-time american sign language recognition from video using hidden markov models, *in* ‘Computer Vision, 1995. Proceedings., International Symposium on’, pp. 265 –270. **10**
- Thi, T. H., Cheng, L., Zhang, J., Wang, L. & Satoh, S. (2010), Weakly supervised action recognition using implicit shape models, *in* ‘Pattern Recognition (ICPR), 2010 20th International Conference on’, pp. 3517 –3520. **19, 73**
- Thureau, C. & Hlavac, V. (2008), Pose primitive based human action recognition in videos or still images, *in* ‘IEEE Conference on Computer Vision and Pattern Recognition, 2008.’, pp. 1 –8. **8**
- Wang, L. & Suter, D. (2007), Recognizing human activities from silhouettes: Motion subspace and factorial discriminative graphical model, *in* ‘IEEE Conference on Computer Vision and Pattern Recognition, 2007.’, pp. 1 –8. **6, 7, 72**
- Weinland, D. & Boyer, E. (2008), Action recognition using exemplar-based embedding, *in* ‘IEEE Conference on Computer Vision and Pattern Recognition, 2008.’, pp. 1 –7. **6**
- Weinland, D., Boyer, E. & Ronfard, R. (2007), Action recognition from arbitrary views using 3d exemplars, *in* ‘IEEE 11th International Conference on Computer Vision, 2007.’, pp. 1 –7. **18, 73**
- Weinland, D., Ronfard, R. & Boyer, E. (2006), ‘Free viewpoint action recognition using motion history volumes’, *Comput. Vis. Image Underst.* **104**, 249–257. **12, 16**
- Weinland, D., Ronfard, R. & Boyer, E. (2011), ‘A survey of vision-based methods for action representation, segmentation and recognition’, *Computer Vision and Image Understanding* **115**(2), 224 – 241. **4, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17**
- Wilson, A. D. & Bobick, A. F. (1999), ‘Parametric hidden markov models for gesture recognition’, *IEEE Trans. Pattern Anal. Mach. Intell.* **21**, 884–900. **15**

- Yamato, J., Ohya, J. & Ishii, K. (1992), Recognizing human action in time-sequential images using hidden markov model, *in* ‘Computer Vision and Pattern Recognition, 1992. Proceedings CVPR ’92., 1992 IEEE Computer Society Conference on’, pp. 379–385. [6](#), [10](#)
- Zaidenberg, S., Boulay, B., Garate, C., Chau, D. P., Corvee, E. & Brémond, F. (2011), Group interaction and group tracking for video-surveillance in underground railway stations, *in* ‘International Workshop on Behaviour Analysis and Video Understanding (ICVS 2011)’, Sophia Antipolis, France, p. 10. [1](#), [72](#)
- Zatsiorsky, V. M. (1998), *Kinematics of Human Motion*, Vol. 1, Human Kinetics. [23](#), [24](#), [25](#)
- Zelnik-Manor, L. & Irani, M. (2001), Event-based analysis of video, *in* ‘Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001.’, Vol. 2, pp. II-123 – II-130 vol.2. [7](#), [8](#), [16](#), [72](#)

# Appendix A

## Graphs

In the following figures depict the GUI of the implementation. This includes the visualization of 3D Input Data, the calculated angles and also the generation of the dictionary as well as the options to start the recognition process. Also the visualization of the results from the voting and the action statements are shown.

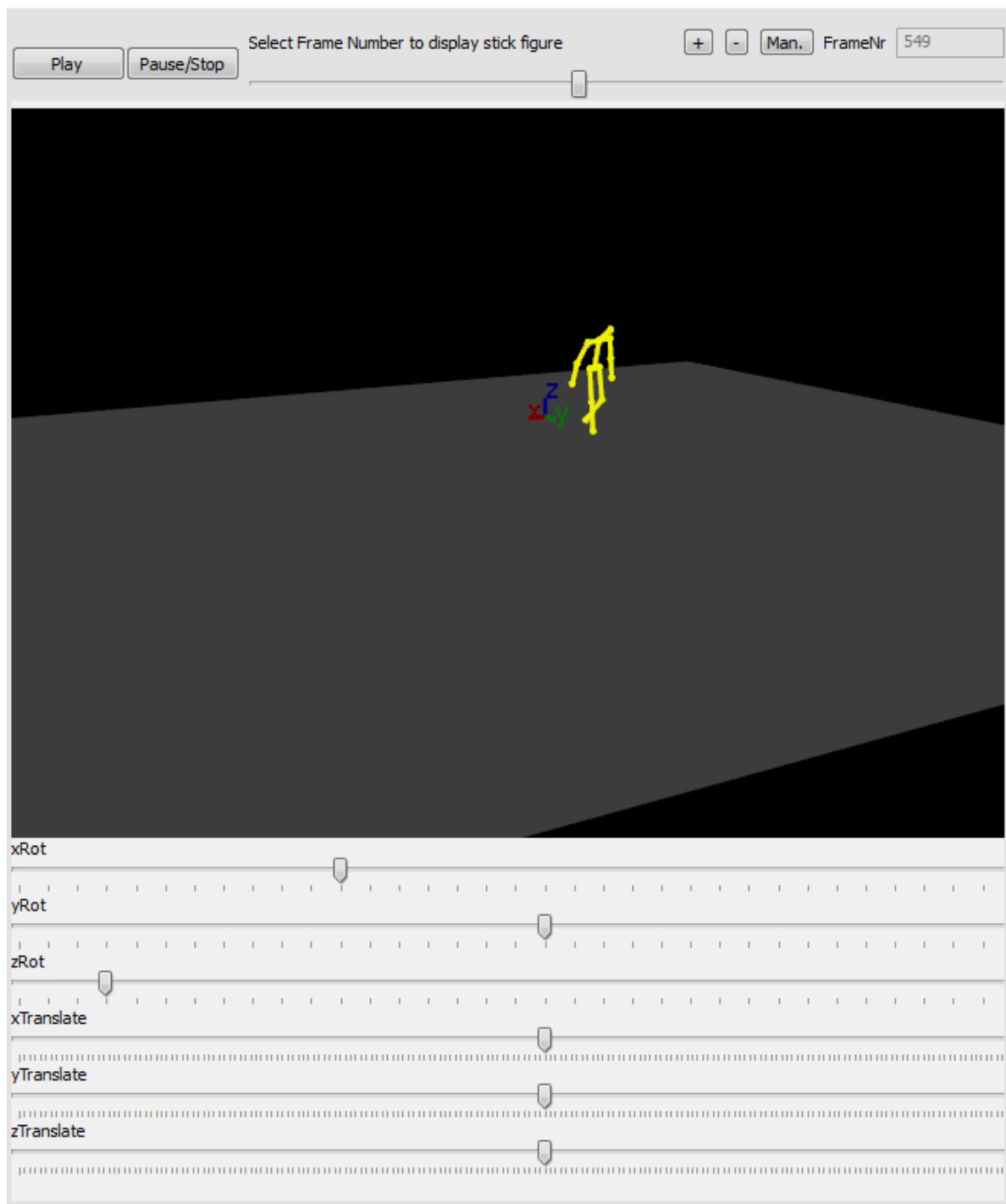


Figure A.1: The visualization of 3D input data.





Figure A.2: The visualization of the relative angles and their derivations.

**Generate descriptor for range of frames - this is needed for adding new entries to Dictionary**

StartFrame

EndFrame

Insert rangedescriptor into dictionary or create new dictionary

Codeword Label

---

**Details of the actual Dictionary:**

Current Dictionary:

Select codeword

Filename D:/MY\_C3D/ONLY\_WALK\_SUBJ07/Training/07\_07.c3d, Start Frame: 77, End Frame: 215

---

**Configure what should be included in nearest neighbor search**

Important: Angle value and derivation setting overrides the element setting

<input checked="" type="checkbox"/> Relative angle values	<input checked="" type="checkbox"/> First Derivations	<input checked="" type="checkbox"/> Second Derivation
<input type="checkbox"/> Hip World Coordinates	<input checked="" type="checkbox"/> Hip First and Second Deriv.	
<input checked="" type="checkbox"/> Hip angle (relative to world)	<input checked="" type="checkbox"/> Spine (rel. to hip)	<input checked="" type="checkbox"/> Shoulder (rel. to spine)
<input checked="" type="checkbox"/> Neck (rel. to shoulder)	<input checked="" type="checkbox"/> l. upper arm (rel. to shoulder)	<input checked="" type="checkbox"/> r. upper arm (rel. to shoulder)
<input checked="" type="checkbox"/> Left arm (rel. to l.u. arm)	<input checked="" type="checkbox"/> Right arm (rel. to r.u. arm)	<input checked="" type="checkbox"/> Left Upper foot (rel. to hip)
<input checked="" type="checkbox"/> Right upper foot (rel. to hip)	<input checked="" type="checkbox"/> Left foot (rel. to l.u. foot)	<input checked="" type="checkbox"/> Right Foot (rel. to r.u. foot)

Set the scale factor of the MeanShift Search: (float value)

Figure A.3: The management of the dictionary (upper part) and the starting of the action recognition process allowing the definition of the values which should be included (lower part).

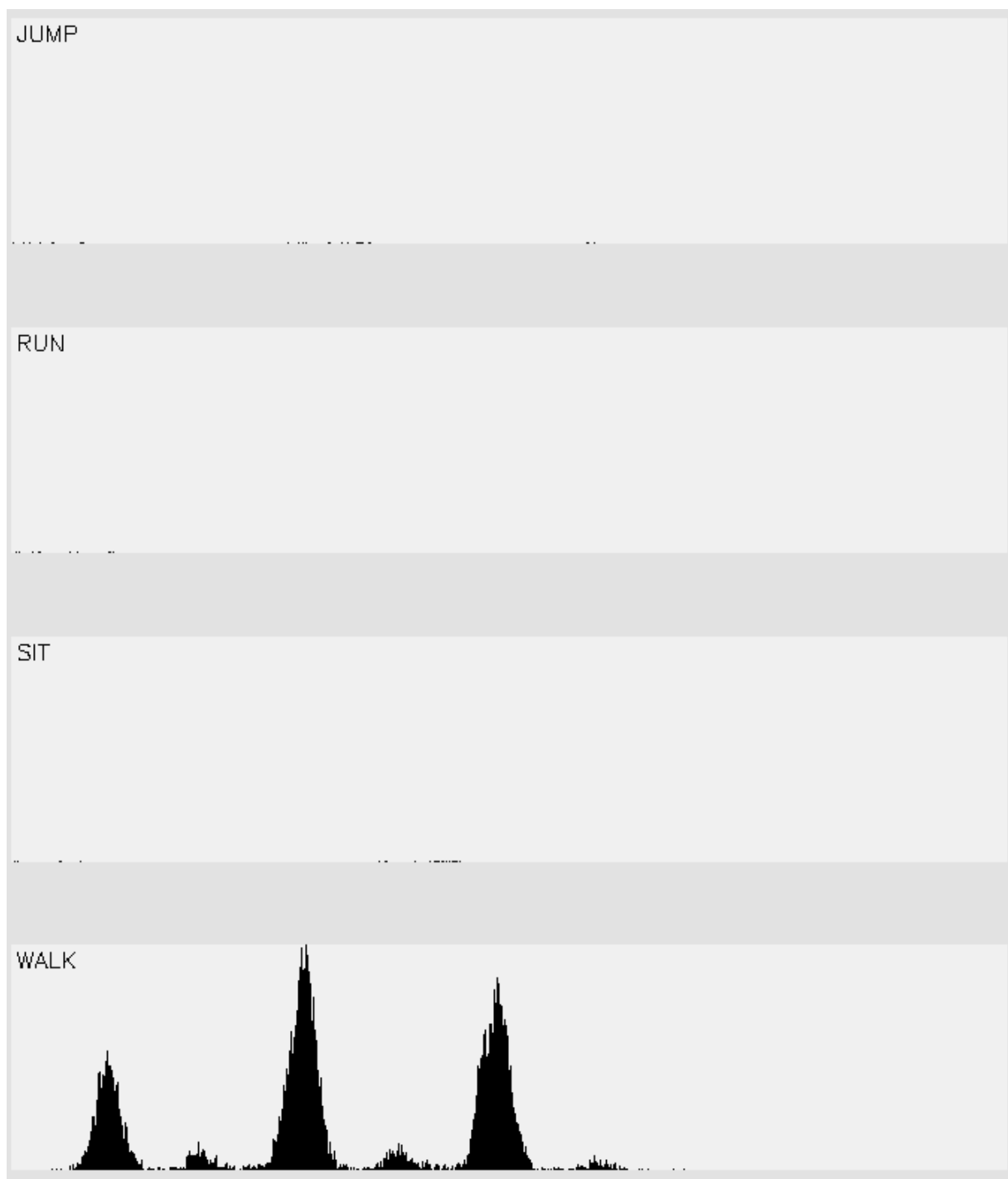


Figure A.4: The visualization of the distribution of the votes.

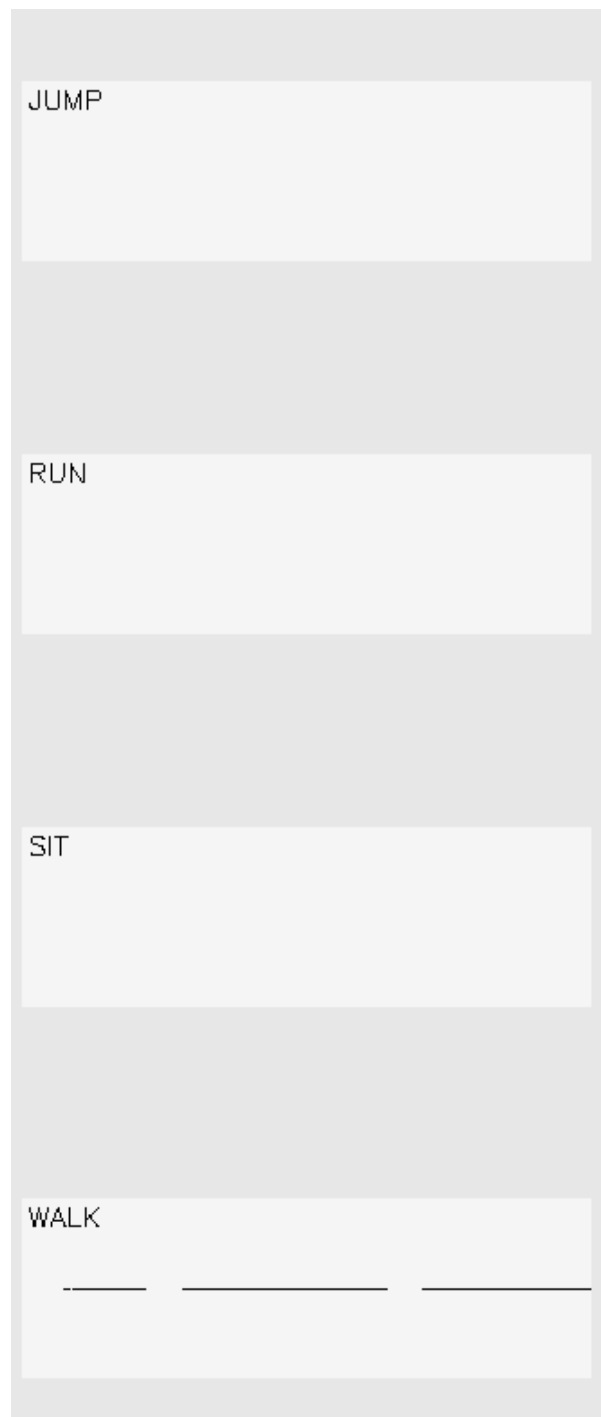


Figure A.5: The visualization of the action statements.

# Appendix B

## Libraries

For the avoidance of re-implementing already existing procedures and also performance issues, several libraries are used in the implementation. The graphical user interface (GUI) is realized with Qt, for matrix and image operations the OpenCV library is used. The rendering of 3D data is done with OpenGL, and as mentioned in Section 3.2.5, the FLANN library is utilized for performing approximative nearest neighbor searches. Details of these libraries are proposed in the following sections.

### B.1 Qt

Qt, which is pronounced ‘cute’ is a very popular platform independent framework for developing all kind of applications. The variety reaches from simple command-line tools to extended GUI Applications. The programming language is C++.

Qt was developed by the Norwegian company Trolltech and released 1995 as Qtopia. In 1997 Qt was used to build the KDE Desktop for Linux and was widely used for developing GUI Applications on Linux systems (Blanchette & Summerfield 2006). Since 2001 and version 3.0 Qt is also available for Windows and MAC platforms. In 2008 Nokia bought the Trolltech company.

The main advantages of Qt are a very simple and intuitive GUI design which can be done either with the built-in designer tool shown in Figure B.1 or in the code itself. Other useful tools are built-in features for XML editing, SQL database access, thread management and much more. Qt is licensed under the GNU Lesser General Public License and the usage is free. Popular applications which use Qt are Skype, Google Earth and the VLC Media Player.

The GUI of the implementation of the ISM based action recognition framework is shown in Chapter A. In the left third, the 3D Renderer for human poses can be seen. In the middle, the results from the conversions to relative angles are shown and the right third is for managing the dictionary and starting the recognition process.

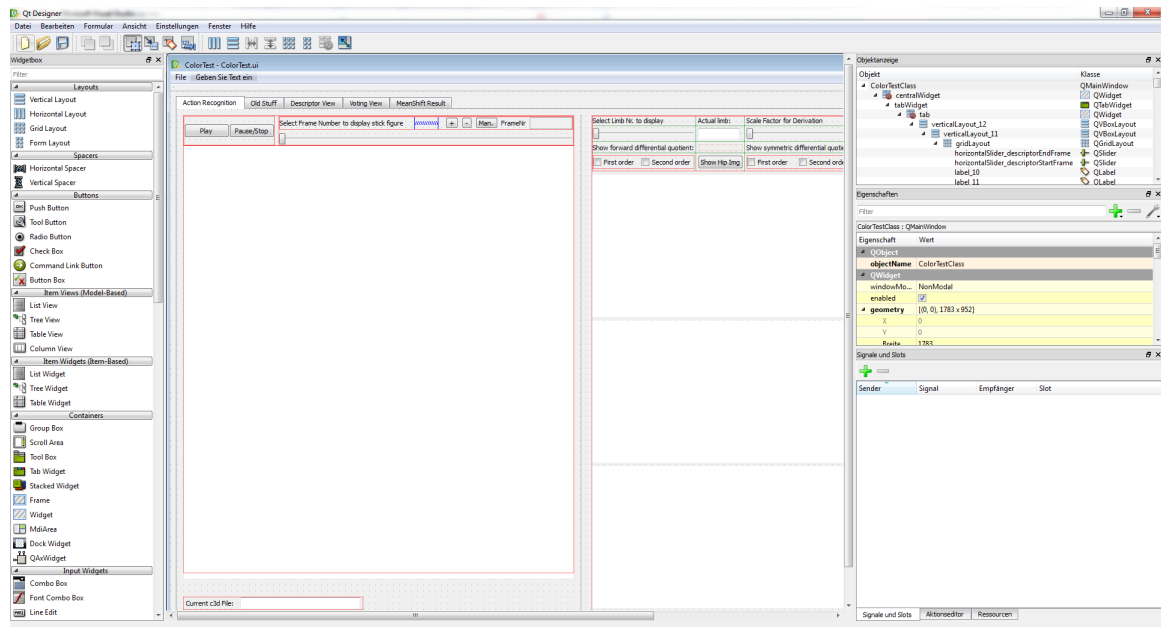


Figure B.1: The Qt Designer for creating interactive GUIs.

## B.2 OpenCV

Open Source Computer Vision (OpenCV) is a widely used library for image and video analysis. It contains several hundred algorithms for such purposes and was introduced by Intel in 1999. Today it is supported from Willow Garage<sup>1</sup>, a robotics research lab (Laganière 2011).

First versions were implemented in C but since version 2, it supports C++ as a programming language. The application areas of OpenCV cover a wide range from simple image loading and visualization as shown in Figure B.2 and manipulation over face recognition to advanced tracking technologies and even statistical learning methods. OpenCV is split in several modules from which the most important ones are described in the next enumeration:

- **core** module, which provides basic data structures as matrix operations.
- **imgproc** module for image processing including filtering, color conversions building histograms and much more.
- **video** module with algorithms for tracking and motion estimation for example.
- **calib3d** is a module for use with multi camera systems including methods for pose estimation and 3D reconstruction.

<sup>1</sup><http://opencv.willowgarage.com/>

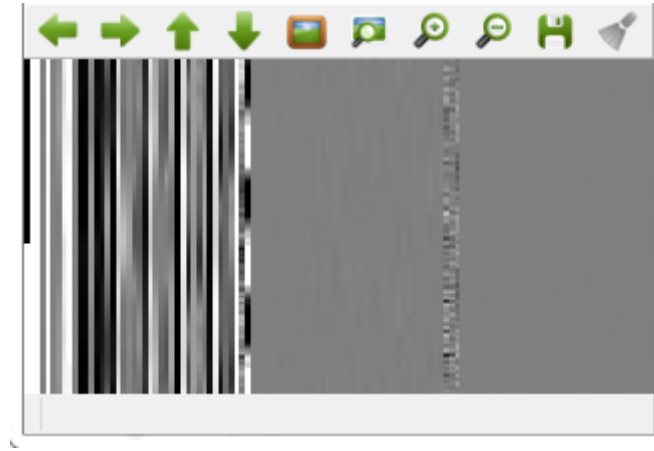


Figure B.2: Example of the visualization of an OpenCV matrix with the `imshow()` function. The matrix in this image is a descriptor as mentioned in Section 3.2.3.

- **features2d** module provides feature descriptors like SIFT, SURF among many others.
- **objdetect** module allows the detection of predefined objects like faces, eyes and cars.
- **highgui** provides an interface for capturing of video and image data.
- **gpu** includes several algorithms which are optimized for GPU devices.

In our implementation, OpenCV is used at many different points like displaying images of results and matrix operations with the `Mat` structures provided by OpenCV. For example, the descriptor matrix as described in Section 3.2.3 is implemented as an OpenCV matrix. OpenCV also allows easy storing and loading of matrices and XML files on the hard disk with the `FileStorage` interface.

## B.3 OpenGL for Rendering of 3D Poses

The Open Graphics Library (OpenGL) is a software which provides an interface to a graphic card for displaying images or 3D object. It was developed by Silicon Graphics (SGI) in the early 1990s and since 2006, The Khronos Group Inc.<sup>2</sup> has control of the OpenGL specification.

OpenGL defines a cross-language and cross-platform programming interface which provides several hundred functions like drawing points, lines or polygons for the creation of complex scenes (Segal & Frazier 2010). These basic primitives are converted

---

<sup>2</sup><http://www.khronos.org/>

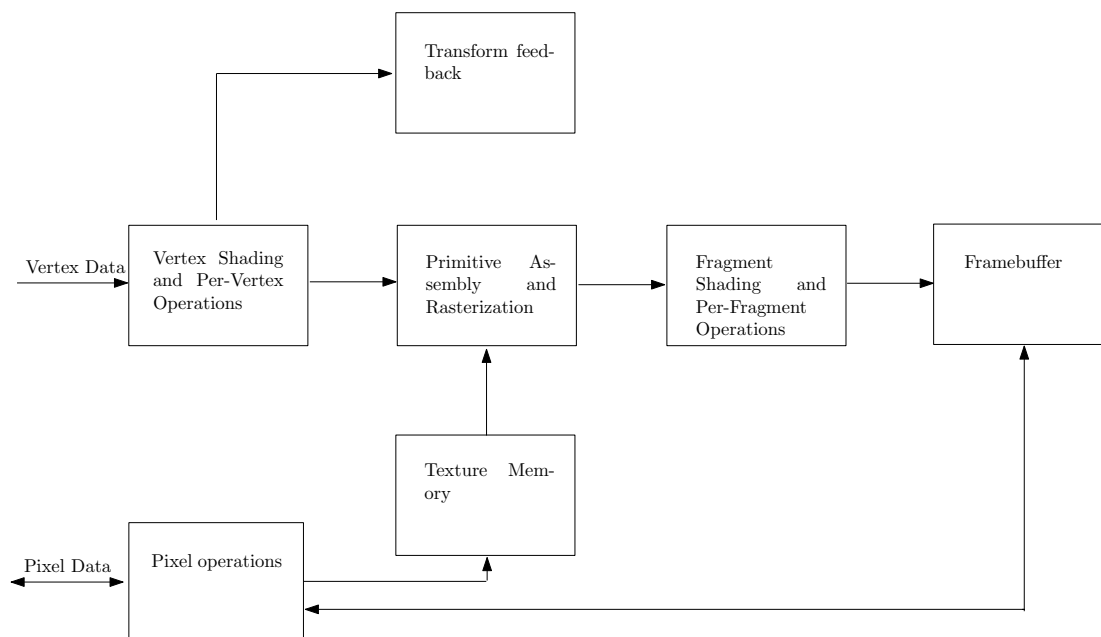


Figure B.3: The OpenGL State Machine. Commands enter on the left and are processed through the pipeline.

to pixels by passing a state machine called GL State Machine which is shown in Figure B.3.

In the first stage, vertex data as points or polygons are processed to geometrical primitives who can be processed by the following stage to other primitives. Then the data is rasterized into fragments and converted into pixels. On these fragments, operations like color blending can be performed before the last step follows, which is the insertion of fragments into the frame buffer. Elements in the frame buffer are then displayable on a screen. An example of the rendering of 3D poses is shown in Figure 3.7.

## B.4 Fast Library for Approximate Nearest Neighbors

The Fast Library for Approximate Nearest Neighbors (FLANN) was already described in Section 3.2.5. The library was developed by Marius Muja and David G. Lowe and released as open-source software under the BSD License<sup>3</sup>.

FLANN is written in C++ but also has bindings to C, Matlab, and Python. Compared to the brute-force technique of searching the nearest neighbor, the approximative approach of FLANN is several orders of magnitudes faster (Muja & Lowe 2009).

<sup>3</sup><http://www.opensource.org/licenses/bsd-license.php>



To run FLANN, a dataset and a query matrix have to be defined as `flann::Matrix`. Then, a randomized tree must be built by using the `flann::Index` interface and the `buildIndex()` function. The `knnSearch()` function then does the actual nearest neighbor search.

# Appendix C

## Timetable

Period	Work items
01.12.2011- 02.12.2011	Getting familiar with the geometric description of humans. Record advantages and disadvantages of Ferrers methods. Aim: formal basics for the correct geometrical description of humans.
05.12.2011- 10.12.2011	Setting up the environment for development and installing all needed frameworks (Qt, OpenCV, OpenGL). Familiarization with Qt and conception of a first GUI. Aim: Working GUI prototype
12.12.2011- 17.12.2011	Realization of an interface and a routine for converting 3D Motion Capture data from c3d format to C++ STL vectors. Implementing a routine for transferring 3D Motion Capture Data to a stick figure model. Aim: Ability to load .c3d files and converting them to the human model used in this work.
20.12.2011- 23.12.2011	Display a stick figure with OpenGL in the program. Retrieve how to calculate relative angles from 3D coordinates including affine transformations and normalizations of data. Aim: Visualizing stick figure and getting correct results for relative angles of the stick figures limb.
09.01.2012- 13.01.2012	Calculating first and second order derivations of relative angles with different methods. Display derivations in the program. Aim: Finding a sufficient method for derivating relative angles and ability to display the derivations over time.

Period	Work items
16.01.2012- 21.01.2012	Building a descriptor out of relative angles, first and second order derivations in a single matrix. Implementing methods for displaying and saving the descriptor. Find good data from the MOCAP Database for several actions as waving, walking and running. Aim: Ability to generate a descriptor which can be processed and used in later work.
23.01.2012- 28.01.2012	Reading papers for the related work chapter. Writing down the fundamentals for the work done till this date. Aim: Draft for the first chapters of the final thesis.
30.01.2012- 03.02.2012	Implementing the codeword dictionary and methods to create and edit a dictionary. Writing Chapter 2. Aim: Finished Chapter 2 and ability to generate and edit dictionaries.
06.02.2012- 10.02.2012	Implementing nearest neighbor search with FLANN (Fast Library for Approximate Nearest Neighbors) <a href="#">Muja &amp; Lowe (2009)</a> . Implementing appropriate methods for analyzing displaying the results of FLANN. Aim: Completing the workflow from c3d data until the search for nearest neighbors from descriptors in codewords.
13.02.2012- 17.02.2012	Testing the implementation with several dictionaries and large data sets. Implementing specializations of the descriptor to allow comparisons for specified limbs or datasets. Aim: Completing the workflow from c3d data to the search for nearest neighbors from descriptors in codewords.
13.02.2012- 17.02.2012	Improvement and Enhancement of the Gui. Basics of Mean Shift Search Aim: Well arranged GUI and understanding of the Mean Shift mode seeking.
27.02.2012- 03.03.2012	Testing the implementation of the Mean Shift Search with example data. Set the development environment to get Mean Shift working. Aim: Be able to run the MeanShift algorithm with sample data.
05.03.2012- 09.03.2012	Converting FLANN results to a Mean Shift readable format. Improving the displaying of the FLANN results. Aim: Correct displaying of FLANN results and first runs of Mean Shift Search on these results.

Period	Work items
12.03.2012- 16.03.2012	Improving the implementation of Mean Shift search. Implementing methods for displaying Mean Shift Results. Reimplementing methods for displaying in separate classes and restructuring of code. Aim: Being available to run Mean Shift search and visualize the results.
19.03.2012- 23.03.2012	Testing the application and removing bugs. Generation of a complete training data set for five actions. Aim: Correct implementation and appropriate training sets.
26.03.2012- 30.03.2012	Parallelization of the voting process and the Mean Shift Search. Create visualizations of the Mean Shift Search in MatLab. Aim: Being able to calculate Mean Shift results faster with larger dictionary.
09.04.2012- 13.04.2012	Finishing of Chapter 2. Writing Chapter 3. Aim: Finished Chapter 2.
09.04.2012- 13.04.2012	Preparing and first Experiments. Finishing Chapter 3. Aim: Outline and first results for Chapter 5.
16.04.2012- 20.04.2012	Code review to check for bugs and small errors. Writing Chapter 4. Aim: Stable implementation and complete Chapter 4.
16.04.2012- 20.04.2012	Generating graphs of the results for these experiments. Writing Chapter 'Evaluation'. Aim: Finished the Thesis to Chapter 4 and first parts of Chapter 5.
23.04.2012- 27.04.2012	Performing experiments for parameter settings. Aim: First experiments for Chapter 5.
30.04.2012- 04.05.2012	Review and correction of Chapters 3-4. Writing chapter 5. Aim: First parts of chapter 5.
07.05.2012- 11.05.2012	Corrections and improvements in all Chapters. Performing experiments for robustness. Aim: Correct text and more experimental data.
14.05.2012- 18.05.2012	Implementing methods to get noisy input data. Performing experiments for robustness with noisy input data. Aim: Tested robustness of the approach.

Period	Work items
21.05.2012- 25.05.2012	Finishing of the thesis. Aim: Finished all experiments and corrections.