Time Invariant Action Recognition with 3D Pose Information Based on the Generalized Hough Transformation

Master Thesis

by

Camilo Andres Ramirez Fandiño

17. May 2013

Supervisor: Prof. Dr.-Ing. Friedrich M. Wahl[1]
Advisor: Dr. rer. nat. Wolfgang Hübner
Advisor: David Münch

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

[1]Institute for Robotics and Process Control - Braunschweig University of Technology

Abstract

Motivated by the advances in computer vision and machine learning algorithms and the availability of more powerful computational resources, automatic human action recognition has emerged as an active research field in the recent years with several promising applications. Systems capable of extracting relevant information from human motion in video sequences could be applied in several domains such as video surveillance, automatic video indexing, analysis of sport events and human computer interaction. On the basis of a previously implemented framework (Zepf, 2012), this thesis presents an implementation of an automatic human action recognition system using a Hough-transform based approach. In this thesis, the recognition performance of the implementation is assessed for different actions using two different types of descriptors: joint-angle and geometric descriptors. Additionally, the time scale invariance and the robustness of the system concerning real data are evaluated. The results show that the evaluated actions can be classified successfully after a short time of observation.

Contents

Chapter 1

Introduction

Several video analysis tasks require human operators to monitor a huge amount of video data containing human motion. Such activities are not only tedious but demand high concentration, require high operational costs and are prone to errors. In some cases, it is also likely that relevant information is disregarded since the intervals during which motion occurs are rather rare. Consequently, the development of systems capable of analysing and understanding such video sequences has become an active research field in computer vision with many possible applications. Typical examples of applications include video surveillance systems, automatic video indexing, analysis of sport events and human computer interaction among others.

According to the hierarchy proposed in Moeslund *et al.*, 2006, recognition of human movements can be divided into action primitives, actions and activities. Action primitives refer to short movements that can be described at the limb level. Actions are constructed from several action primitives and represent a whole body movement whereas activities are sequences of actions that can be given a semantic interpretation. For example, hand or leg movements are action primitives. Jumping, running, and dribbling are actions while playing basketball is an activity containing these actions. Following this definition, the work presented in this thesis is focused on the recognition of individual actions. This, together with person tracking, is an integral part for the analysis of more complex interactions between people (Figure 1.1). Although, there are particular demands for each action recognition framework, three general steps in the implementation process can be identified. In the first place, the data is acquired by means of a single camera or more specialized 3D sensors. Secondly, the input data from the sensors is processed in order to extract relevant information from each frame and represent it in a suitable way: the descriptor. In the last step, characteristic patterns are found in the representation of data and they are classified in order to give a hypothesis of the type of action being performed. The extraction of useful information normally requires advanced image processing

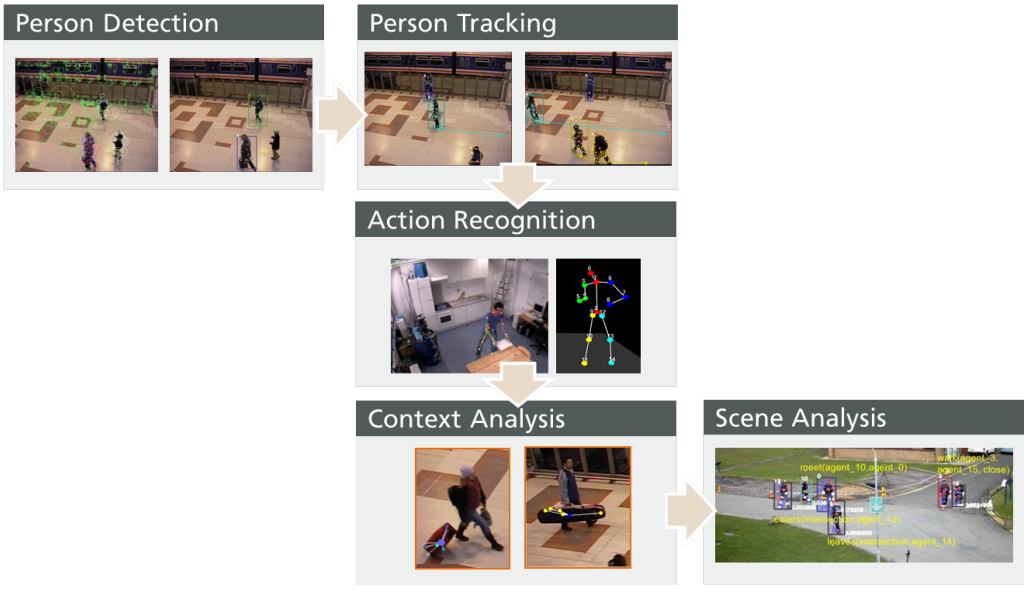


Figure 1.1: This figure depicts the pursued pipeline from images to semantic scene descriptors at the VCA group at the Fraunhofer IOSB.

techniques such as person detection and tracking whereas classification demands state of the art machine learning algorithms. As a result, human motion detection becomes a challenging and computational intensive task especially when real-time processing of video data is required.

Early developments in the field were only able to perform action recognition under an ideal environment in which, for example, only one person is present in each video frame and the background remains unaltered. However, more sophisticated techniques are needed in applications where cluttered video frames with several people, and possibly low video resolution, camera movements and partial occlusions are present. Previous research has shown that recognition of actions in such circumstances is possible with Hough-transform based methods (Yao, Juergen Gall, and L. V. Gool, 2010), (Yao, Juergen Gall, and L. Gool, 2012). Their ability to integrate information from different frames into a probabilistic framework makes them robust to small variations and missing information.

Motivated by the important role that action recognition plays and justified by the success of previous research, an action recognition framework based on the generalized Hough transform is developed in this thesis. In the following, a more detailed view of the objectives of this work is given.

1.1 Problem Definition and Aim of this Thesis

Automatically deriving semantic descriptors of an observed scene is a challenging task. The derivations of statements about actions a person is performing are high-dimensional and complex. In this work actions should be derived from the 3D pose of persons over time. It has been shown in previous work (Zepf, 2012), that the generalized Hough transform can be applied on action recognition. In this work actions are recognized on real data; on the one hand a depth sensor, the Microsoft Kinect, is available and on the other hand there are promising results in estimating the 3D pose of the human body in image sequences, see e.g. (Brauer and Arens, 2011). Additionally, the person-centric actions are a further step towards fully exhaustive video content analysis. Building on an existing framework, that has already been implemented to show the fundamental feasibility of using the generalized Hough transform for action recognition, an extension for temporal invariance should be developed and evaluated. Furthermore, the robustness of the action descriptors should be systematically increased. Subsequently, the framework should be examined for its applicability in real world scenarios. A real-time implementation is a must.

1.2 Thesis Outline

The thesis is structured as follows: Chapter 2 gives an overview of several recent approaches previously developed for action recognition. Chapter 3 introduces the main concepts for representing and classifying human body poses on which the implementation of this framework is based. Chapter 4 describes the improvements that were made in this work over the previously developed framework. Specifically, boolean geometric descriptors are introduced as a new type of human body pose representation that can improve robustness with respect to other descriptors. Additionally, the reciprocal nearest neighbour (RNN) procedure for clustering is introduced as a tool for improving robustness and reducing the computational time. Chapter 5 gives details over the full implementation of the framework, including the main modules of the program and specifications of the graphical user interface. Subsequently, Chapter 6 presents the main results obtained for different training and test sequences. Finally, Chapter 7 contains the summary and gives an outlook for future work.

Chapter 2

Related Work

The continuous development of more powerful computational devices has made rapid advance in computer vision possible. Human action recognition is no exception; a huge number of publications are nowadays available. These works differ mainly in the methodology used for extracting features and classifying them. Before proceeding to a more detailed description, this section intends to give a brief overview of recently developed methods in human action recognition. As mentioned before, methods for action recognition can be roughly classified according to how the image or video frame is represented and according to how this information is used to perform a classification. Given the extensive previous research on the topic, only a brief description of methods representing each category is given.

2.1 Image Representation

The way in which an image, depth image, video frame, or sequence of frames is represented is the initial step in any human action recognition approach. The chosen representation must be able to generalize over small variations in person *appearance*, *background*, *viewpoint* and *action execution*. Simultaneously, it must be robust enough to allow for accurate classification. Image representation can be categorized into global representations and local representations. The pattern followed in global representations requires a top down approach, that is, the person is initially localized by tracking or background subtraction. Subsequently, the whole localized person is used as the input descriptor for the classification procedure. On the contrary, local representations are characterized by a button-up approach. An image is described as a collection of individual patches where each patch is normally selected with help of a spatio-temporal interest point detector. The individual patches are later assembled to form the image representation. An important advantage of these

approaches is their robustness to noise and occlusions. In the following subsections, a few examples of the different methods are given.

2.1.1 Global Representations

As mentioned earlier, this kind of approaches normally require localization of the person. Once the region of interests (ROI) with the person in the center is found, different features such as silhouettes, edges or optical flow can be derived and used for the representation of the body.

An early example of such approaches was developed by (Bobick and Davis, 2001). They calculate a binary motion energy image (MEI) by aggregating the differences between silhouettes of consecutive frames, thus obtaining an approximation of where motion occurs. Similarly, silhouettes are used by (Y. Wang *et al.*, 2007) where an $\mathfrak{R}$ Transform is applied in order to achieve scale and translation invariance. In (L. Wang and Suter, 2006), silhouettes are also used. A representative silhouette is obtained by averaging the mean intensity among all centred frames, and in addition, the mean shape is calculated from the centered contour of all frames.

Instead of using a single camera, the approach adopted by (Weinland *et al.*, 2006) uses silhouettes from multiple views to construct a 3D voxel model. This work is an extension of the initially proposed approach in (Bobick and Davis, 2001) but extended to three dimensions. That is, the motion energy images and motion history images (MHI) are obtained by a set of calibrated cameras. The later, also known as motion template, represents a volume whose voxels indicate how recent a motion occurred. Since invariance with respect to position, orientation and size is required, the motion template is first transformed into cylindrical coordinates and a Fourier transform over the resulting function is calculated. The invariance results from the Fourier shift theorem which states that a function $f_0(x)$ and its translated counterpart $f_0(x - x_0)$ only differ by a phase modulation after the Fourier transformation:

$$F_t(k) = F_0(k)e^{-j2\pi kx_0} \tag{2.1}$$

Consequently, the magnitudes $|F_t(k)|$ are shift invariant. By choosing an appropriate cylindrical coordinate system, rotations around z axis are converted into translations, thus achieving rotation invariance as well.

When background subtraction is not available, optical flow can alternatively be used. However, difficulties with this approach may arise as optical flow is also influenced by camera movements. This can be compensated by tracking the person.

A variation of global representations are grid-based approaches. These approaches are implemented by dividing the spatio-temporal domain into a grid. This reduces

Figure 2.1: Space-time shapes for jumping, walking and running actions (Blank *et al.*, 2005).

difficulties that usually arise due to noise and partial occlusion. Although grid based methods are somewhat similar to local representations, they still need a global representation of the ROI. An example of such methods is described in (Kellokumpu *et al.*, 2008).

3D spatio-temporal volume (STV) also belongs to global representations category. In this method, different features in each frames are extracted and stacked over time to form a global function. In (Blank *et al.*, 2005), silhouettes from each frame are extracted and a volumetric space-time shape is constructed as shown in Figure 2.1. Given this volumetric shape, they compute a new function by assigning, for every internal point in the silhouette, a value reflecting the mean time required for a random walk beginning at the point to reach the boundaries. This function can be represented as a partial differential equation known as the *Poisson equation* with the boundary conditions specified by the silhouette contours. By computing the solution of this function, a wide variety of useful local shape features can be extracted. In order to obtain global features, weighted moments are calculated over the local features.

2.1.2 Local Representations

Instead of representing the video frame or group of frames as a whole, local representation methods use a collection of local descriptors or patches. They posses the advantage of being more robust to partial occlusions and changes in viewpoint. These types of methods can be categorized further into space-time interest point detectors and local descriptors.

Space-time detectors are operators capable of finding space-time points in a video sequence where important changes occur. (I. Laptev and Lindeberg, 2003) idealize

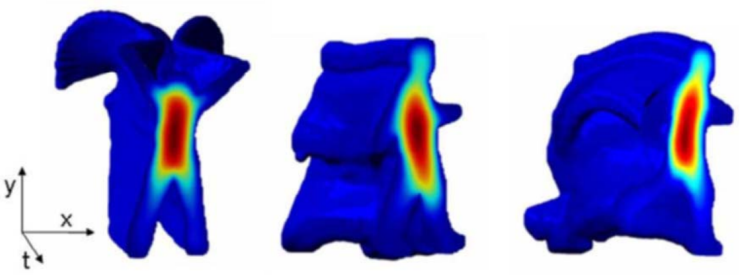


Figure 2.2: Solution of the Poisson equation for space-time shapes. Each space-time point represents the mean time required for a particle undergoing a random-walk process starting from the point to hit the boundaries (Blank *et al.*, 2005).

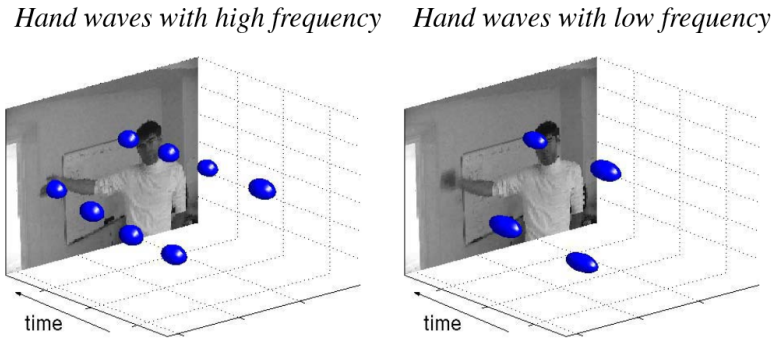


Figure 2.3: Result of Spatio-Temporal point detector for a wave action (I. Laptev and Lindeberg, 2003).

this approach by extending the Harris corner detector (Harris and Stephens, 1988). Instead of detecting significant spatial variations in an image, this new approach takes into account the time dimension. Points with high variations in the spatio-temporal domain will represent the location and time in a video sequence where non-constant motion occurred. Figure 2.3 illustrates the results of such a detector for a waving action.

Local descriptor representations abstract an image or video as a collection of small 2D or 3D windows. This descriptor should ideally be invariant to background clutter, appearances and occlusions. An example in this category is proposed in (Schuldt *et al.*, 2004) where space-time interests detectors are also employed. Alternatively, patches are described by local grid-based descriptors in (Willems *et al.*, 2008). In this work, a novel spatio-temporal feature detector is employed in which an improvement

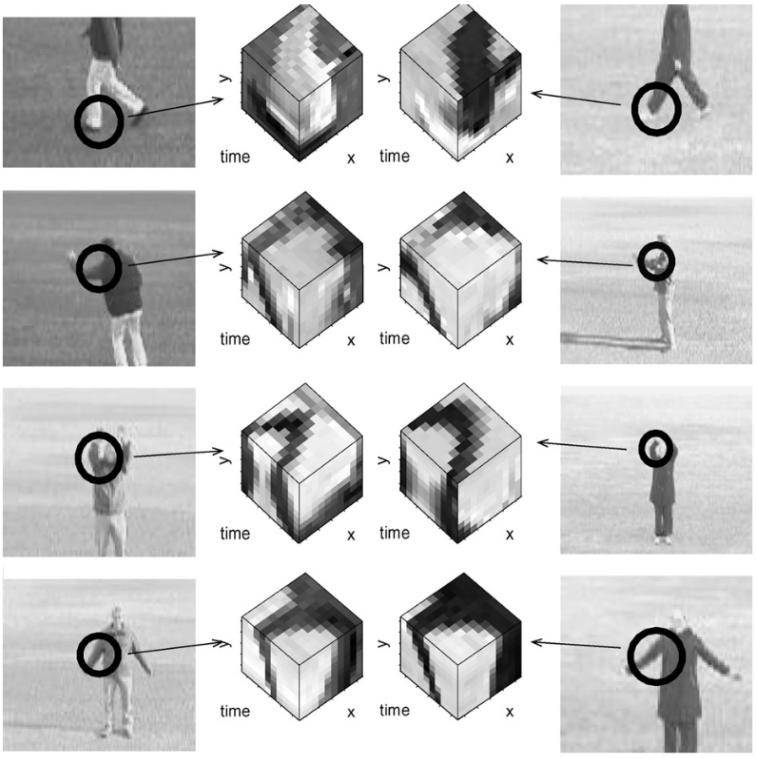


Figure 2.4: Cuboid regions around interest points for walking, boxing and hand waving actions (Ivan Laptev *et al.*, 2007).

in efficiency is achieved compared to the original detector introduced in (I. Laptev and Lindeberg, 2003). To describe the encountered points, an extended version of the SURF (Bay *et al.*, 2008) descriptor is implemented as illustrated in Figure 2.4. It consists of dividing the volume in the neighborhood of the interest point into a grid of $MxMxN$ bins. Each bin is filled by a weighted sum of uniformly sampled responses of the three axis-aligned Haar-wavelets d_x, d_y, d_t.

2.2 Action Classification

Having obtained a representation of an image or video frame, the next step is to perform classification. This can be done by directly classifying images without explicitly modeling variations in time as described in Section 2.2.1. Alternatively, temporal state-space models can be used, this is described in Section 2.2.2.

2.2.1 Direct Classification

Classifiers of this type typically compile an entire video sequence into a single representation or describe and classify each video frame individually without taking into account the temporal domain or relations between the frames. e.g. Each observed image sequence or single frame is then compared to a labeled dataset of actions by means of nearest neighbor (NN) classification. A different alternative is the use of discriminative classifiers in which a function that discriminates between two or more classes is learnt. In any case, dimensionality reduction is desired as typically image representations are highly dimensional and hence computationally expensive. Common linear dimensionality reduction methods include Principal Component Analysis (PCA), local linear embedding (LLE) and locality preserving projections (LPP).

In the subcategory of NN classifiers, a distance measure between the image representation and a training data set is used. The outcome of the classification is a set of nearest neighbors for each action class from which the most common action label is chosen as result. Another possibility is to calculate an action prototype for each class in the training set by finding the mean descriptor for each specific class label. This prototyping of classes is convenient in terms of computational performance as the search space is reduced. It is important to chose an appropriate training data set, distance metric, and image representation in order to cope with intra and inter-class variations.

The work of (Blank *et al.*, 2005) falls into the NN classifiers category. They use the Median Hausdorff Distance defined as:

$$D_H(s^1, s^2) = \underset{j}{median}(\underset{i}{min} \left\|c_i^1 - c_j^2\right\|) + \underset{i}{median}(\underset{j}{min} \left\|c_i^1 - c_j^2\right\|) \tag{2.2}$$

where c_i^1 and c_i^1 are space-time cubes belonging to the sequences s^1 and s^2 accordingly. In order to allow more flexibility, only partially overlapping sequences are compared instead of the whole sequence. In contrast, in the work by (Bobick and Davis, 2001), Hu moments are used and the Mahalanobis distance is used for comparison in order to take into account the variance of each dimension.

Another type in this category are discriminative methods. They attempt to separate two or more classes instead of modeling them. To this purpose, Support Vector Machines (SVM) are used in (Jhuang *et al.*, 2007).

2.2.2 Temporal State-Space Models

These types of classifiers try to model an action by calculating the probabilities of transitions between states which represent the image observations. While generative

models learn a joint distribution over observations and action labels, discriminative models learn the probabilities of the action classes conditioned on the observations.

Although Dynamic Time Warping (DTW) is used for comparison of two sequences, it can be classified in the category of generative models. It basically consists of finding the optimal match between the two sequences. In order to perform the optimization, a local distance measure is required. A long distance indicates that the difference between the two sequences is large, whereas a short measure indicates that the sequences are more similar.

Extensive research has also be done using Hidden Markov Models (HMM). In simplified terms, HMM predicts the probability of a sequence to have descended from a particular prior sequence or vice versa. Applied to action recognition, states in the model correspond to image representations. Additionally, two assumptions are made: state transitions are only dependent on the previous state and observations depend only on the current state (Yamato *et al.*, 1992),(Filipovych and Ribeiro, 2008) and (İkizler, 2008)).

2.3 Hough-Transform based Action Recognition

Hough Transform based-methods are another important type of classificators commonly used in both object and action recognition. The additive nature of these kind of methods make them more robust to loss of information. That is, in order to recognize an action, not every single pose which is part of this action is needed. Two relevant works using the Hough Transform Approach are reviewed.

2.3.1 A Hough Forest for Action Recognition

The work by (Yao, Juergen Gall, and L. V. Gool, 2010) addresses the problem of action recognition by a probabilistic voting framework using Hough Forests, (J. Gall *et al.*, 2011). Among the advantages of this method are the ability to use dense features over sparse features and the possibility to use only one classifier for all actions instead of separate classifiers for every single action. Hough Forests are based on Random Forests. They consist of a set of binary decision trees, where each non-leaf node is associated with a binary test and each leaf node is associated with a class distribution. The trees are referred as random because they are constructed by selecting a random subset of training samples and for each split in the non-leaf nodes of a tree, a random subset of possible splits are selected. Among this selection, the most optimal splitting is chosen.

Before the Hough Forest can be applied for action recognition, it has to be trained. Training sets are initially given for each class label. A requirement for each training sequence is that it should contain at least one complete action cycle. After applying a 2D Hough Forest to identify the person performing the action, a 2D bounding box around the person is drawn for each frame. Taking advantage of the high correlation of person scale and location between frames, a particle filter can be applied in order to obtain normalized action tracks. That is, a size normalized cuboid enclosing the object position in each frame. With such action tracks, the vote space is reduced in each video sequence. After the track generation step, 3D patches are extracted from the action tracks and given as input for each tree in the Hough Forest for the training process. Each of the extracted patches can be described as $P_i = (I_i, c_i, d_i)$ where P_i is a 3D patch of size $(x_{pixels} * y_{pixels} * frames)$, I_i are the extracted features at a patch such as grey scale, x and y time derivatives or absolute value of optical flow in x and y, c_i is the action label and d_i are the displacement vectors from the patch centre to the action track centre.
After the training phase, the classification and localization of actions is performed. In this step, as in the training, each video sequence undergoes the process of action track normalization and feature cuboid extraction. These cuboids serve as inputs for the already trained Hough Forest. Each cuboid which is passed through the forest, casts votes in a Hough space for action labels and spatio-temporal centres. After the whole video sequence has been passed through the forest, local maxima are searched in the Hough space, thus retrieving class label and temporal location of the respective action.

The implemented framework for action detection was evaluated with popular benchmarks consisting of large datasets containing a variety of action recognition scenarios. Including the UCF sports dataset, consisting of broadcast sport action videos and the UCR Video web Activities Dataset, which features multiple people interaction in surveillance scenarios. The result of applying the framework to this datasets shows an average better accuracy over other action recognition methods thus achieving state of the art performance even in challenging realistic video sequences (Yao, Juergen Gall, and L. V. Gool, 2010).

2.3.2 Coupled Action Recognition and Pose Estimation from Multiple Views

Human motion can be interpreted on a physical level by, for example, estimating the 3D coordinates of the body pose. Alternatively, it can be interpreted on a higher semantic level as action recognition by understanding the movements of the body over time. These two are directly connected given that the action being performed by

the subject restricts the physiological possible 3D poses of a human body to a subset of action specific poses. Conversely, 3D pose knowledge makes it simpler to identify specific actions as opposite to simple appearance based knowledge, given that the pose itself already contains the relevant high-level information of an image frame. The work developed in (Yao, Juergen Gall, and L. Gool, 2012) is based on the previous assumptions. In this order of ideas, a system is implemented in which a prior 2D appearance-based action classifier system firstly helps to estimate the 3D pose and subsequently, the pose information is used to refine the action label (Figure 2.5).

The Hough Forest classifier can be trained and used for recognition either with appearance based features or with pose 3D based features. In the first step, to have an initial guess of the action label, the appearance based features such as colour, optical flow and spatio-temporal gradients are determined. The action specific manifold obtained in this step is used as a prior distribution for the particle-based optimization for the 3D pose estimation. Once the 3D pose is estimated, the Hough Forest framework can be employed this time using pose 3D based geometric features as described in (Müller *et al.*, 2005) thus obtaining a more precise action label.

The framework has been tested on standard benchmark datasets on 3D human pose estimation containing subjects performing different actions. The results showed that optimizing the particle filter with a prior action estimate improve the pose estimation. It has also been observed that action estimation using 3D pose features is more efficient than 2D feature-based estimation. One important advantage is that due to the view invariance of the 3D pose data, training information can be obtained from different data sets. Finally, to obtain higher prediction accuracy and thus better results, the loop can be closed. It means that the last action labeling step can be fed back as input for the 3D pose estimation and so on.

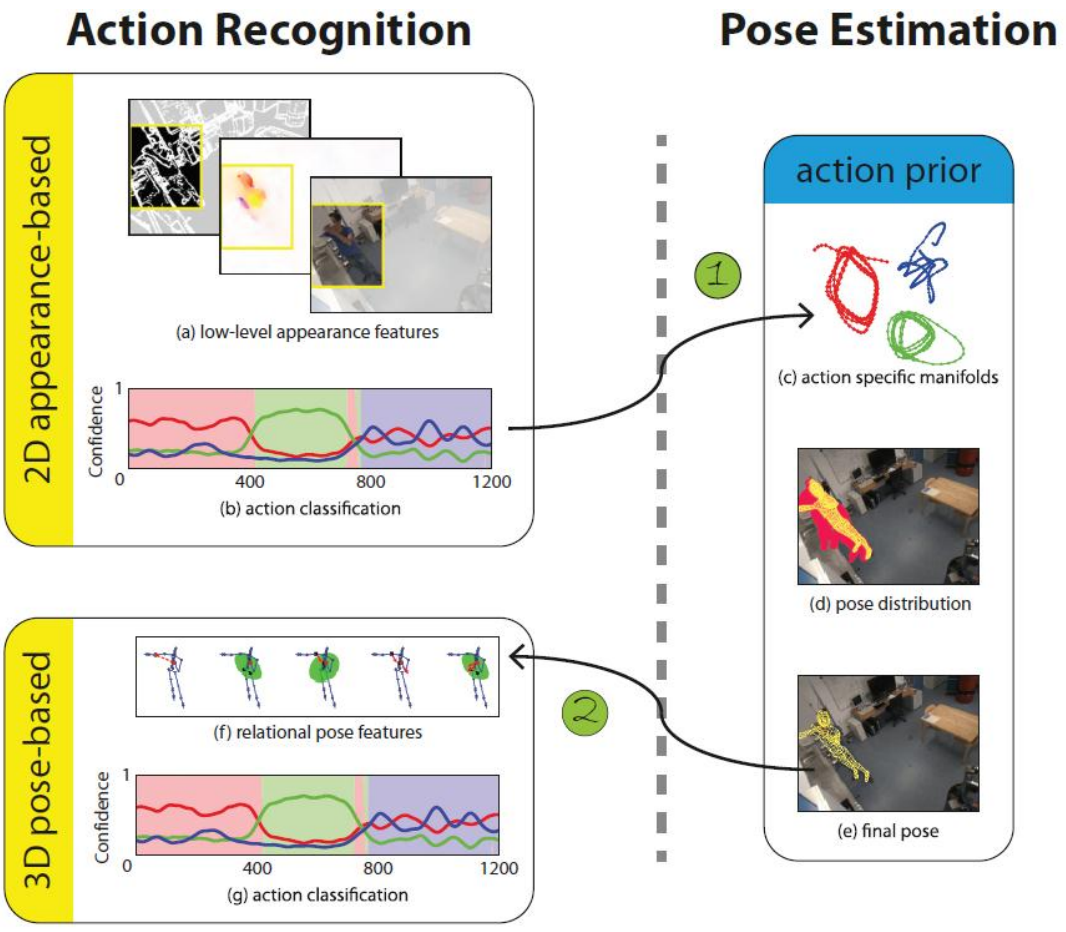


Figure 2.5: Action recognition and Pose estimation loop (Yao, Juergen Gall, and L. Gool, 2012). The loop begins by extracting the 2D image features from which an initial action statement is derived. This initial guess is used as a prior distribution for the particle-based optimization for 3D pose estimation. In the end, action recognition is performed based on pose features extracted from the estimated 3D pose.

Chapter 3

Basics of Human Body Representation and the Implicit Shape Model

Having explained some of the most relevant approaches in action recognition, it is now appropriate to describe the basic methods used in this work. Recalling from Section 2, action recognition approaches require finding a suitable representation of the video frames and subsequently classifying them. The following sections describe the details of how this is done. Namely, the human body model used for creating a descriptor and the Implicit Shape Model (ISM) used for classification.

3.1 Human Body Model

Instead of representing video frames with features directly extracted from an image such as edges, silhouettes, grey values or gradients, it is possible to use a human body model as descriptor. This model should be able to provide enough information that allows representation of a wide variety of body postures, thus making the recognition of different actions possible. A human body model is normally a set of two or three dimensional points representing the body parts. The model can additionally encode the relation between the body parts represented as a kinematic chain where each point is not only described by its coordinates in space but also their relations to the other body part.

A commonly employed technique for acquiring the 3D coordinates of the body limbs is the use of reflective markers around the body. A set of synchronized cameras under optimal illumination condition is placed to record the markers at a high sampling rate. Despite the high accuracy and relative simplicity of data acquisition

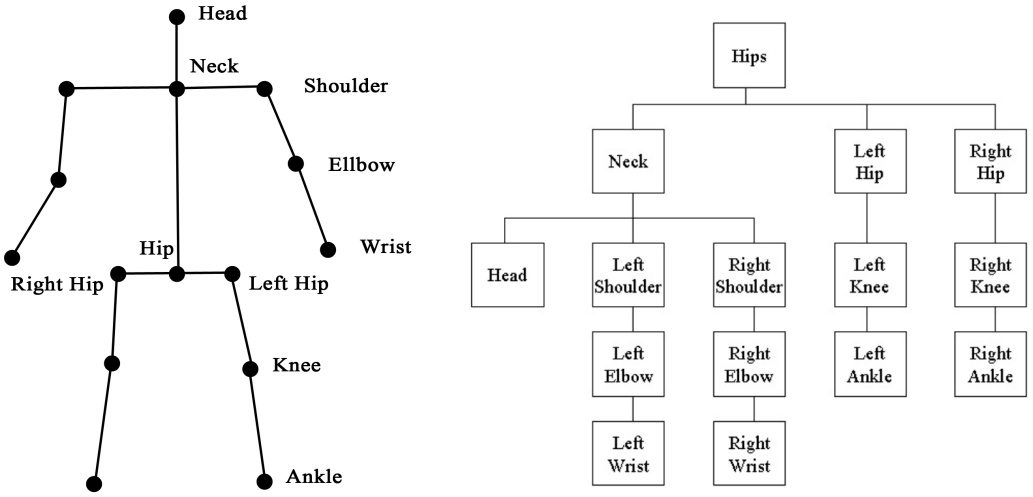


Figure 3.1: Human body model as proposed in (Gonzàlez, 2004). (a) Stick Figure model with fifteen joints and twelve limbs. (b) Hierarchy of the stick figure model.

procedure using this technique, such systems are only possible for recording motion in ideal circumstances. For applications such as video surveillance, human-robot interaction or automatic tagging and indexing of video sequences, another method for acquiring three dimensional information has to be used. To this purpose, a promising approach was developed in (Brauer and Arens, 2011). They extract body poses from simple 2D images and reconstruct the 3D body coordinates using a perspective projection camera model. Since the mapping from 2D to 3D coordinates can yield more than one solution, they additionally exploit anatomical constraints and joint angles probabilities in order to reduce the set of possible solutions and have a more accurate guess of the actual pose.

3.1.1 Stick Figure Model

In this work, a stick figure model is used to represent the human body as described in (Gonzàlez, 2004). In this setup, the body is composed of 15 markers corresponding to 12 limbs as seen in Figure 3.1(a). In addition to the body limbs, a hierarchical relation between each limb is established. In this way, the body can be seen as a kinematic chain where the hip joint is the parent and the rest of the joints can be reached from this point on.

The model reflects with enough accuracy the anatomical structure of humans, thus enabling a rich representation of body postures. However, expressing the body

posture directly with coordinates in space has important disadvantages. Positions of the body parts can widely differ between subjects performing the same action mainly due to differences in body length. Furthermore, the same action can be performed in several ways, implying variations in the movements of every limb (such as the stride of a walk action) and consequently in the space coordinates of every point. Given these conditions, a representation of the body by means of different coordinate systems is more convenient.

3.1.2 Joint-Angle Descriptor

Every limb in space could be easily represented as a local coordinate system by a transformation matrix $M \in \mathbb{R}^{3x3}$ with respect to a global coordinate system. However, such a matrix is not efficient because nine different parameters have to be set, where only three angles are necessary to express a rotation. This redundancy make this method more prone to computational errors. Alternatively, Euler angles can be employed. With this strategy, rotations are made in a specific order where the first rotation is always relative to the global coordinate system. While Euler angles are easy to compute, they exhibit some disadvantages. On one hand, the presence of singularities make it in some cases impossible to determine the order in which the rotations where made. On the other hand, discontinuities over time are a challenge.

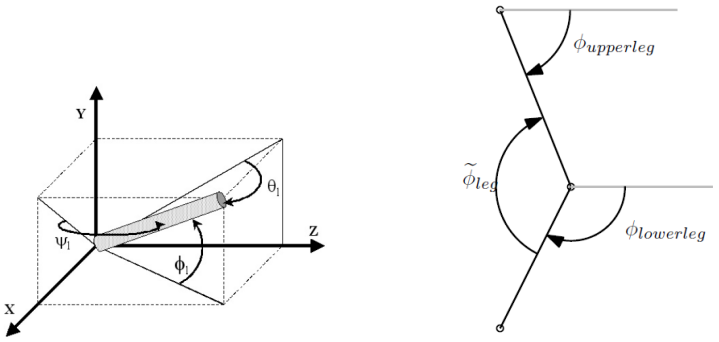
Bearing these difficulties in mind, an alternative representation is chosen in (Gonzàlez, 2004). They use a mapping to a higher dimensional space to solve the discontinuity problem. Each limb is represented by 3 different values: elevation ϕ, latitude θ and longitude ψ as shown in Figure 3.2a(a). These values are computed with the following equations:

$$\phi = tan^{-1}\left(\frac{y_i - y_j}{\sqrt{(x_i - x_j)^2 + (z_i - z_j)^2}}\right) \tag{3.1}$$

$$\theta = tan^{-1}\left(\frac{x_i - x_j}{\sqrt{(y_i - y_j)^2 + (z_i - z_j)^2}}\right) \tag{3.2}$$

$$\psi = tan^{-1}\left(\frac{z_i - z_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}\right) \tag{3.3}$$

Given this representation, angles lie in the range $[\frac{-\pi}{2}, \frac{-\pi}{2}]$ and the discontinuity problem is avoided. Although the representation with absolute angles is more robust to



(a) (b)

Figure 3.2: Joint-angle descriptors. (a) Visualization of the absolute angles in 3D polar coordinates. (b) Visualization of the relative angles between two limbs.

spatial variations than a three dimensional coordinates representation, it still lacks the ability to generalize over some limb movements that are semantically similar but have absolute angles that can significantly differ. For example, the movement of one hand from side to side could be performed with the upper arm positioned in different configurations, but it would still be a waving action. However, a numerical comparison using the absolute angles of such actions would regard them as different. To overcome this problem, the body poses are represented with relative angles between joints (see Figure 3.2(b)). These are calculated by subtracting the absolute angle values of corresponding parent-child pairs according to the established hierarchy (3.1b(b)).

3.1.3 Pose Normalization

Before calculating the relative angles, it is necessary to express the three dimensional points in a local coordinate system in order to achieve view invariance. The initial step in the normalization is to define a local coordinate system specified by a rotation, a translation, and a scaling from the global coordinate system. This can be achieved by employing affine transformations.

The local coordinate system is chosen as follows: the origin is located in the middle between the right hip, and the left hip coordinates, this point is taken as the translation vector $\vec{v}_t$. The x axis is placed along the hips and the y axis is calculated as the vector orthogonal to x and the vector along the torso. Finally, the z axis is found as the cross product between x and y. The vectors $\vec{b_x}$ $\vec{b_y}$ and $\vec{b_z}$ are computed as follows:

$$\vec{b_x} = \vec{v_t} - \text{rightHipCoordinate} \tag{3.4}$$
$$\vec{b_y} = \vec{b_x} \times (\vec{v_t} - \text{neckCoordinate}) \tag{3.5}$$
$$\vec{b_z} = \vec{b_x} \times \vec{b_y} \tag{3.6}$$

$$M_{rot} = \begin{bmatrix} \vec{b_x} & \vec{b_y} & \vec{b_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$M_{trans} = \begin{bmatrix} 0 & 0 & 0 & \vec{v_t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

$$M_{scale} = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

In addition, the scaling factor is chosen as the length of the vector formed by the hip and the neck markers. Once the above terms are calculated, the affine transformation from 3D world coordinates to normalized person coordinate systems can be defined as follows:

$$A = (M_{trans} * M_{rot} * M_{scale})^{-1} \tag{3.10}$$

Every joint in 3D world coordinates is then multiplied by the affine transformation matrix. With the local 3D coordinates available, the absolute and relative angles of the limbs are calculated for each video frame.

3.1.4 Building the Descriptor

Having the data normalized and represented according to the human body model presented above, it is now possible to create a simple descriptor. This can be embedded in a one dimensional vector $x_s = (\widetilde{\phi}_1, \widetilde{\theta}_1, \widetilde{\psi}_1, ..., \widetilde{\phi}_{12}, \widetilde{\theta}_{12}, \widetilde{\psi}_{12})$.

Information about the variation of the angles in each time step can also give important clues on the action being performed and improve the robustness of the

descriptor. Therefore, In addition to the relative angles, the first and second derivatives are calculated following the implementation proposed by (Zepf, 2012). They find the derivatives using the differential quotient between subsequent data points in three different ways:

$$forward : f'(x) = \frac{f(x+h) - f(x)}{h} \tag{3.11}$$

$$backward : f'(x) = \frac{f(x) - f(x-h)}{h} \tag{3.12}$$

$$symmetric : f'(x) = \frac{f(x+h) - f(x-h)}{h} \tag{3.13}$$

Besides the relative angles of each pair of joints, the global coordinates of the hip and its variations are also calculated in order to give a clue on the global motion of the body within the scene. Grouping all the terms together in one vector yields:

$$x_s =: D \tag{3.14}$$

This descriptor is used to represent the body pose in each time step. Hence, the classification can be performed using the Implicit Shape Model as explained in the following section.

3.2 Implicit Shape Model Based Action Recognition

The classification method employed in this work is based on an adaptation of the action recognition of the Implicit Shape Model (ISM) approach which was initially introduce by (Leibe *et al.*, 2008). The original idea was the development of a method capable of recognizing previously unseen objects in complicated scenes where large variation of features such as object color, texture, and shape are present. This is achieved by first creating a codebook of local appearances that learns the variability of an object category. That is, instead of defining explicitly all possible shapes of a class object, allowed shapes are defined implicitly by agglomerating similar features into clusters. Although the method was first used in object detection, an adaptation to action recognition is possible. The details of the method are described in the following subsections.

3.2.1 Codebook Generation and Clustering

The approach begins with the generation of a codebook. Namely, building a visual vocabulary of similar local appearances that are typical to a specific object class. In order to create the codebook, the input features have to be chosen. In the case of object recognition, an interest point detector is used to find relevant regions in an image. Afterwards, grey value patches and local shape context features are extracted from these regions (see (Leibe *et al.*, 2008) for details). Once the features from a training dataset have been obtained, they are used as input for the clustering algorithm.

A key element in the ISM approach is the clustering step. It consists of an unsupervised classification of patterns (usually feature vectors) into groups based on similarity. Once a dataset is clustered, patterns belonging to the same cluster are more similar to each other than they are to patterns associated to other clusters. The general steps in any clustering algorithm can be defined as: pattern representation, definition of a distance metric, grouping and optionally data abstraction.

The pattern representation step is related to the identification of appropriate variables to use as descriptors and their properties such as range, scale and domain. The next step is the definition of a suitable distance metric in order to have a measure of similarity and thus be able to group the patterns, commonly used distance metric include Euclidean distance, Mahalanobis distance or Hamming distance in the case of binary features. Subsequently, the grouping of similar patterns can be performed in a number of ways such as hierarchical clustering algorithms, partitional clustering algorithms or probabilistic methods among others. Lastly, data abstraction refers to the process of extracting a compact representation of each cluster in manner that it is efficiently representative of the features contained in the cluster.

3.2.2 The Training Procedure

After the construction of a codebook, the ISM method is followed by a second step in which the spatial probability distribution P_c is learned. An important design constrain is considered: the spatial probability distribution for each codebook entry is estimated in a non-parametric manner. Hence, the true distribution of the data is learned rather than making possibly oversimplifying Gaussian assumptions.

In order to learn the spatial probability distribution, the following procedure is performed: features of the training images are again extracted using an interest point detector and each of them is compared against the codebook entries. During the comparison, not only the best occurrence of the feature f to the codebook entry C_i is saved, but also all occurrences whose similarity is above a threshold value t.

The occurrences are saved in an array *Occ*. This array also saves all positions where the match occurred relative to the object centre which is known in advance for each training image (see Figure 3.3). By storing the occurrence locations in this manner, the spatial distribution of each sample is modelled in a non-parametric way.(See Algorithm 1 for details).

Algorithm 1: The training procedure . (Leibe *et al.*, 2008).

$F \longleftarrow \emptyset$ // *Initialize the set of feature vectors* F
forall the *training images* **do**
  Apply interest point detector.
  forall the *interest regions* $l_k = (l_x, l_y, l_s)$ *with descriptor* f_k **do**
    $F \longleftarrow F \cup f_k$
  end
end
Cluster F with cut-off threshold t and keep cluster centers C.
forall the *codebook entries* C_i **do**
  $Occ[i] \longleftarrow \emptyset$ // *Initialize occurrences for codebook entry* C_i
end
Compute ocurrences Occ **forall the** *training images* **do**
  Let (c_x, c_y) be the object center at a reference scale.
  Apply the interest point detector.
  forall the *interest regions* $l_k = (l_x, l_y, l_s)$ *with descriptor* f_k **do**
    forall the *codebook entries* C_i **do**
      // Record an occurrence of codebook entry C_i
      $Occ[i] \longleftarrow Occ[i] \cup (c_x - l_x, c_y - l_y, l_s)$
    end
  end
end

3.2.3 The Recognition Procedure

Once the codebook and its respective occurrence array have been obtained, new test images can be recognized. The recognition procedure is divided into two steps: Probabilistic Hough Voting and Scale Adaptive Hypothesis Search.

The first step consist of casting votes from each of the image features. Therefore, features from the test image are initially extracted. By comparing them to the

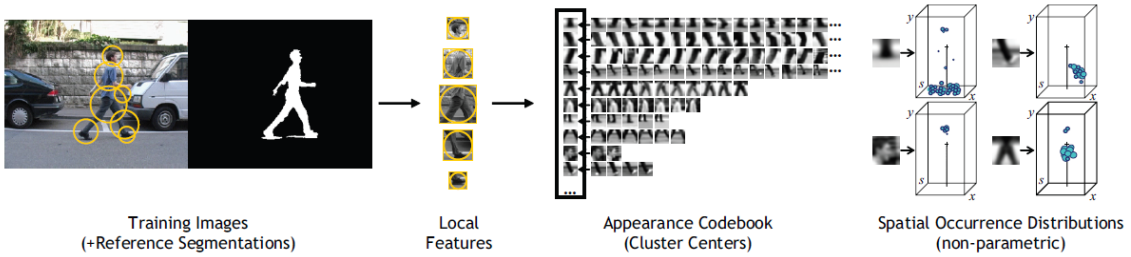


Figure 3.3: ISM Training procedure. Image local features are extracted with an interest point detector. Subsequently, the appearance codebook is created and, in a second iteration, a spatial distribution is learned (Leibe *et al.*, 2008).

codebook, a set of valid hypothesis C_i with a corresponding probability $p(C_i|f,l)$ are obtained. That is, for each entry C_i several valid object scales and centres can in principle be obtained according to the learned spatial distribution. In particular, the probability that the test image contains an object category o_n centred at x given that the feature f was found in the image at location l is given by:

$$p(o_n, x|f,l) = \sum_i p(o_n, x|f, C_i, l)p(C_i|f,l). \tag{3.15}$$

In the first term of Equation (3.15), dependency on f can be removed since the unknown image feature has been replaced by a known interpretation. Similarly, the dependency on l in the second probability term can be eliminated since the matching of features is done independent of the location of the image feature. After these simplifications, Equation (3.15) becomes:

$$p(o_n, x|f,l) = \sum_i p(o_n, x|C_i, l)p(C_i|f). \tag{3.16}$$

The first term in the above equation represents the probabilistic Hough vote for an object class and location given a feature interpretation. The second term indicates the quality of the match between the image feature and codebook cluster. Based on these definitions, the recognition procedure is performed as follows: each extracted feature from the test image is compared against all entries of the codebook. If the similarity between a feature and a codebook entry is bigger than the threshold t, a match between the feature f_k and codebook entry $C*_i$ together with the location offset is kept in a vector M. After having compared feature f_k to every entry in the codebook, the probability that the found matches are correct is estimated as $p(C*_i, f_k) = \frac{1}{\|M\|}$. For every single found match in M, votes are generated by

iterating through the occurrences of the codebook entries. The vote location is set at the position x calculated with the feature location and the offset of the occurrence.

The second step in the recognition is the Scale Adaptive Hypothesis Search. After the vote generation stage, the ideal is to find maxima in the voting space. In order to discard locations in the space with insignificant number of votes, a binned 3D Hough accumulator array is used to collect the votes. After having found those promising locations, a Mean-Shift search is performed. It is expressed as:

$$\hat{p}(o_n, x) = \frac{1}{V_b} \sum_k \sum_j p(o_n, x_j | f_k, l_k) K(\frac{x - x_j}{b(x)}) \tag{3.17}$$

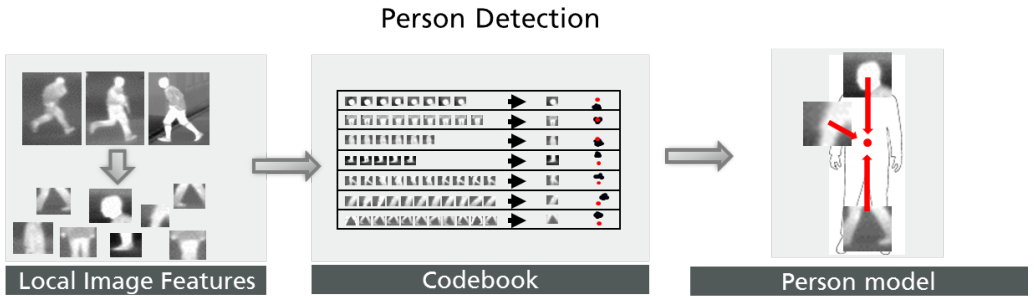
where K is a symmetric kernel function with bandwidth b. This search strategy is guaranteed to converge to local modes of the underlying distribution. The algorithm is adaptive because the bandwidth of the kernel is made dependent of the scale coordinate.

3.2.4 Adaptation from Object Recognition to Action Recognition

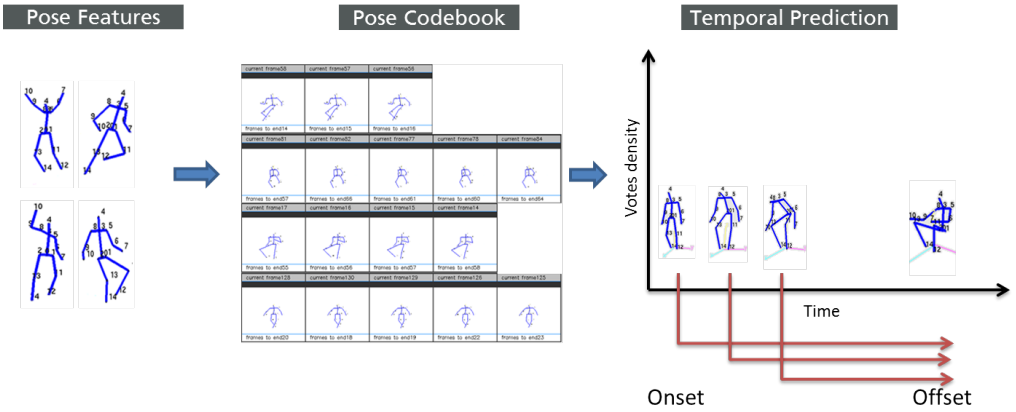
Action recognition in video sequences normally requires, in addition to the body pose estimation, finding the instant in time or time interval in which the action is being performed. In Hough transform based methods, the inclusion of the time parameter implies a higher dimensional Hough space. This requirement, together with the estimation of scale and position of the object, make the computational costs for such procedure high and inefficient. A solution to this problem is given in (Yao, Juergen Gall, and L. V. Gool, 2010). Nevertheless, The method developed in this work is only concerned with estimating the time interval where specific actions take place. This is based on the assumption that for each frame, the three dimensional coordinates of the body markers are already available.

The adaptation of the ISM developed in this work is based on the implementation in (Zepf, 2012). They also calculate the joint angle descriptor and perform a Hough-based classification that estimates action types and their respective intervals in time. During classification, they find similar poses of the input descriptor by performing a nearest neighbor search in the training data set. As an extension to their work, in this thesis, a clustering of the training data is performed before the nearest neighbor search, thus reducing the search space.

As in object recognition, the clustering consist of comparing and grouping similar descriptors in the training data set. In this case, instead of image features, each



(a)



(b)

Figure 3.4: Comparison Between ISM for Object and Action recognition.

descriptor represents the body pose per frame (see Section 3.1.3). Once the descriptors are grouped, each cluster can be interpreted as a set of similar body poses represented by the cluster centroid C_j. In addition, each cluster keeps a vector $O[j]$ containing the body poses with its respective action labels A_l and time offsets t_{offset} indicating where the action may end. Figure 3.4 depicts a comparison between object and action recognition.

Analogous to the ISM for object detection, after having learned a probabilistic distribution of the features, it is now possible to perform the recognition of actions with test video sequences as follows: initially, the descriptor D_i of a body pose (one video frame) at time step i is compared to every centroid C_j generated during clustering. This is done by searching the nearest neighbours of the current descriptor with respect to the cluster centroids. If the distance between the descriptor and its nearest neighbour is bigger than a threshold Thr, each of the entries in $Occ[k]$ of the matched centroid C_j* generates votes for the end of actions according to their

action label and offset (See Algorithm 2).

Once the whole sequence of poses has been iterated to cast votes, The mean shift mode seeking algorithm is performed on the resulting Hough-space in order to find maxima indicating when an action may end. It is important to note that not all the frames of an action have to be processed in order to produce an action statement. As frames are being processed, votes accumulate and peaks of maximum values start to develop. Therefore, it is possible in some cases to find clear maxima in the voting space after a few frames has been processed and thus generate an action statement.

Algorithm 2: Voting algorithm for action recognition

```
// Initialize the Hough voting space V ⟵ ∅ Create a pose descriptor for
each video frame
for i ⟵ to numberOfFrames do
    //Find nearest N neighbours of current fame descriptor in codebook
    NearestNeighbours ⟵ SearchNN(Descriptor[i], codebook)
    forall the Centroids C_j in NearestNeighbours do
        if sim(Descriptor[i], C_j > Threshold then
            forall the Cluster members Occ_k in C_j do
                voteIndex ⟵ i + Occ_k.FramesToEnd
                V ⟵ V ∪ (voteIndex, Occ_k.ActionLabel)
            end
        end
    end
end
```

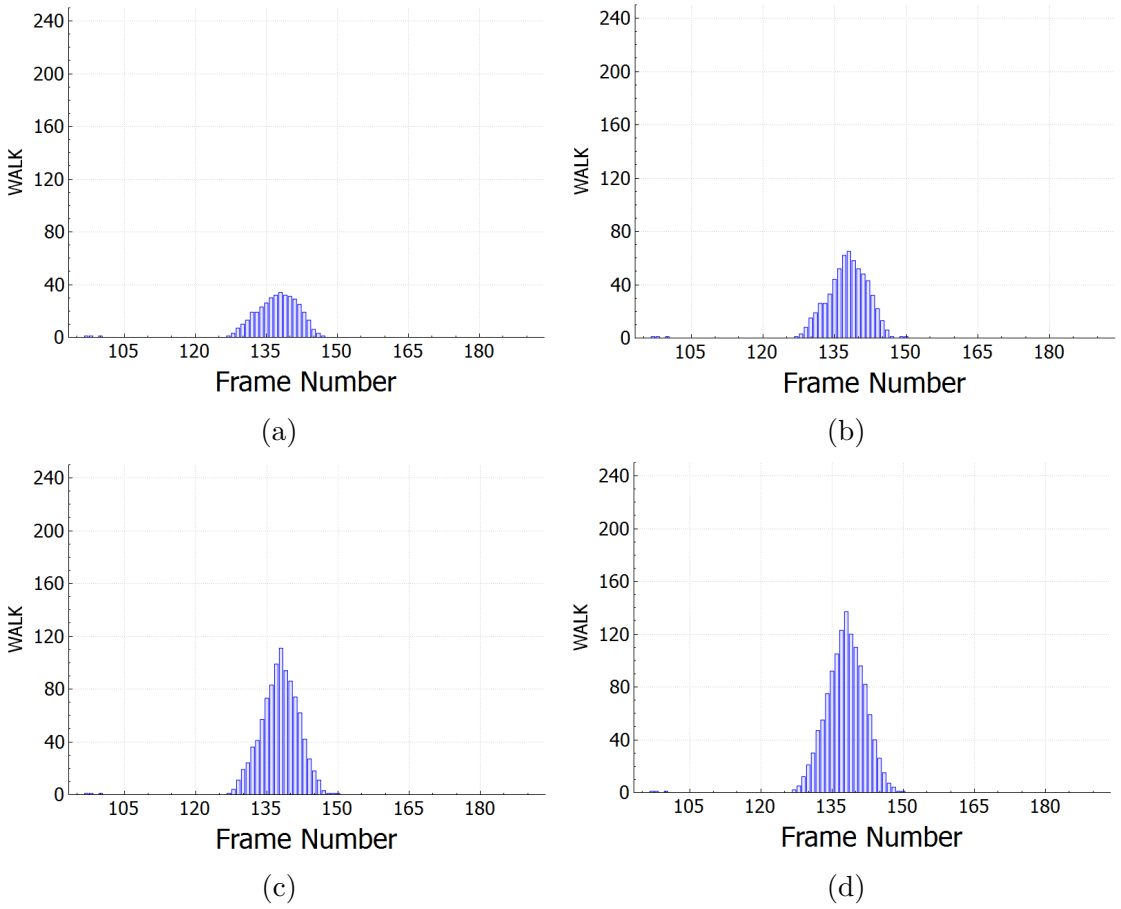


Figure 3.5: Development of the voting space over time for a walk action. The images from (a) to (d) show the Hough voting space for different instants in time in increasing order: (a) $t = 60$, (b) $t = 80$, (c) $t = 100$, and (d) $t = 120$. As the time advances, the peak around $t = 140$ keeps increasing. This indicates how the walking action ending at $t = 140$ can be predicted many frames in advance.

Chapter 4

Developed Methods

Having introduced the basics of the action recognition framework, it is now possible to incorporate some extensions in the initially developed framework in (Zepf, 2012). This section describes the implementation of an alternative descriptor that is intended to improve the robustness of the framework and reduces the dimensionality of the body pose representation. Additionally, as previously mentioned, a clustering algorithm to improve run-time is also described. At the end, a support vector machine is trained and applied to classify the detected actions.

4.1 Geometric Descriptors

The use of a joint angle model for representing a human body posture brings several advantages. Not only is it relatively easy to compute, but also overcomes the problem of differences in limb lengths. Before the angles are calculated and the actual descriptor is constructed, an affine transformation for dealing with the view invariance problem is additionally performed (see Section 3.1.3). Despite the advantages of this method, the joint-angle descriptor (Section 3.1.2) fails when it comes to a more general representation of similar poses. Namely, by numerically comparing two actions, they may be regarded as different even though they are logically similar. For example, the joint angles of a jump action could significantly differ between subjects depending on the body constitution or the length and height of the jump. In order to represent actions in a more semantic level, geometric boolean features descriptor are introduced in this work.

The implementation of the new descriptor follows the idea proposed in (Müller *et al.*, 2005). They introduce an action descriptor that expresses geometric relations between body parts, thus making the representation more robust to spatial variations

and closing the gap between logical similarity as perceived by humans and numerical similarity measures.

The geometric features can be implemented using the same human body model already described in Section 3.1. By establishing a few boolean geometric functions relating body parts, a rich set of meaningful body pose features is constructed. Before defining the most relevant features used in this work, the notion of boolean function is introduced. A boolean function is mathematically defined as $F : \rho \to \{0, 1\}$ where combinations such as $F_1 \wedge F_2$ or $F_1 \vee F_2$ are also valid boolean functions. Furthermore, a set or vector of functions can be defined as $F : \rho \to \{0, 1\}^F$. The application of the function over a pose $P \in \rho$ is then denoted as the feature $F(P)$. Before being able to establish such boolean relations, a meaningful continuous measure M of certain parameter and a threshold tr have to be chosen. Having this, the boolean feature can be defined as:

$$F^{(M)} = \begin{cases} 1 & \text{if } M \geq tr, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

4.1.1 Geometric Measures of Human Body Poses

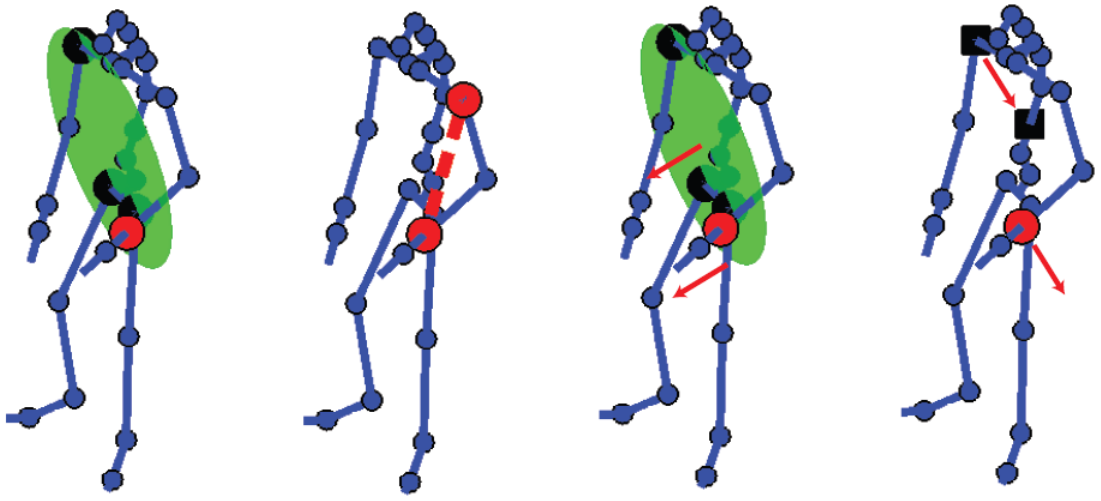
Following the above definition, the initial step in this approach is to define a set of measures. To this end, the position of joint j_i at a given time t is represented as three dimensional points $p_{j_i,t} \in \mathbb{R}^3, 1 \leq i \leq 4$. Some of these measures for specific sets of joints are depicted in Figure 4.1 as qualitative geometric relations between joints.

Firstly, the "distance to plane" is defined as the distance measure from one point to a plane. The function takes four joints as arguments, where the first three are used for the plane and the fourth is used as the point whose relation to the plane is to be found. Then the distance feature can be defined as:

$$D = dist(p_{j_1,t_1}, \langle p_{j_2,t_2}, p_{j_3,t_2}, p_{j_4,t_2} \rangle). \tag{4.2}$$

In the case of $t_1 = t_2$, the function becomes a simple measure of distance in space. By choosing different times, distances between planes and points separated in time are created. Alternatively, the plane can be specified by a point lying on it, and a normal vector.

Another possible feature is the measure of a joint velocity with respect to a plane. Similar to the distance feature, the plane can be either defined by three points or by a normal vector and one point. Given that the velocities $v_{j_i,t} \in \mathbb{R}^3$ of each joint j_i at a given time t are available, a velocity feature can be defined as follows:



(a) (b) (c) (d)

Figure 4.1: Geometric relations between joints (Yao, Juergen Gall, and L. Gool, 2012). (a) Distance from a joint (red) to a plane defined by three joints (black). (b) Distance between two joints. (c) Projection of joint velocity on plane normal. (d) Projection of joint velocity on vector.

$$Vn(j_1, j_2, j_3, j_4; t_1, t_2) = v_{j_1,t_1} \cdot \hat{n}_{\langle p_{j_2,t_2}, p_{j_3,t_2}, p_{j_4,t_2} \rangle}. \tag{4.3}$$

A simpler and equally powerful feature can be defined as the projection of the velocity vector of a joint over a vector defined by two other joints. It is expressed as:

$$Ve(j_1, j_2, j_3; t_1, t_2) = \frac{v_{j_1,t_1} \cdot (p_{j_2,2_1} - p_{j_3,t_2})}{\| (p_{j_2,2_1} - p_{j_3,t_2}) \|}. \tag{4.4}$$

For some kind of actions, it is useful to know whether two joints are connected or not, In this case, it is necessary to find the distance between them. This is accomplished with the following formula:

$$Jd(j_1, j_2; t_1, t_2) = \| p_{j_1,t_1} - p_{j_2,t_2} \| . \tag{4.5}$$

One more useful feature is made by defining the angle between a pair of limbs, where a limb is built by two joints according to the kinematic chain. This is expressed as:

$$Aj(j_1, j_2, j_3, j_4; t_1, t_2) = \cos^{-1} \frac{(p_{j_1,t_1} - p_{j_2,t_1}) \cdot (p_{j_3,t_2} - p_{j_4,t_2})}{\| p_{j_1,t_1} - p_{j_2,t_1} \| \| p_{j_3,t_2} - p_{j_4,t_2} \|}. \tag{4.6}$$

Based on the previous definitions, other kinds of measures can be constructed. Namely, by talking the first and second derivatives of distances, velocities, and angles, complementary information of an action is obtained. Having defined different measures, Boolean features can now be created by additionally specifying a threshold. This is described in the following subsection.

4.1.2 Building Boolean Geometric Descriptors

Building a descriptor becomes now a matter of choosing a set of the previously described measures and a threshold for each one. In order to obtain a meaningful representation, the features have to be selected according to which measure best characterizes a certain pose or sequence of poses. For example, in a typical walk action, the feet move alternately forward and backwards. To find such a sequence pattern, the following functions are defined using the notation previously described:

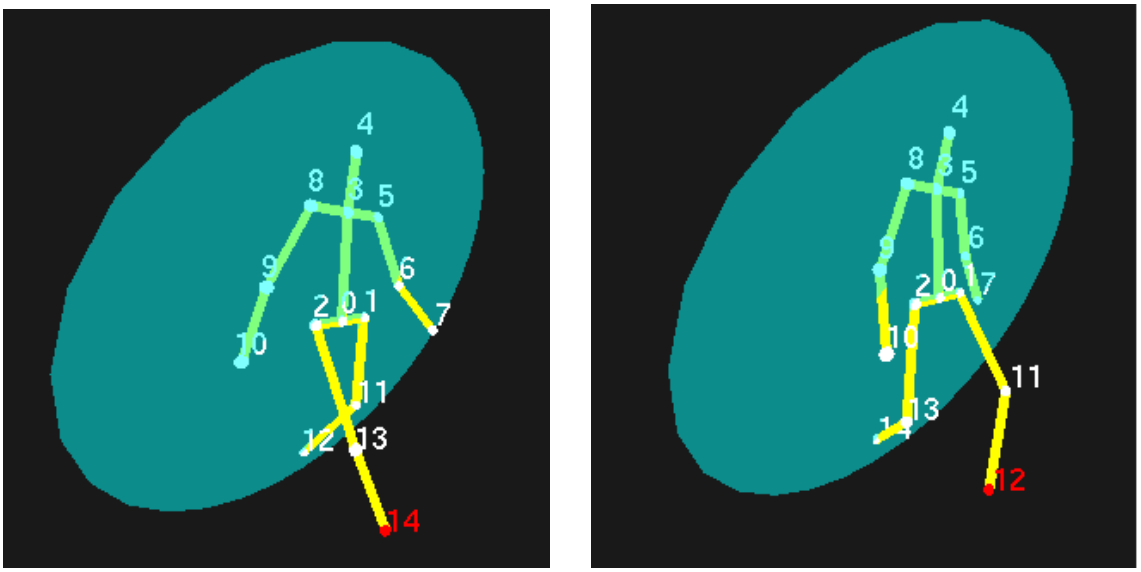
$$F^r = F(D^r) = F(D(HipCenter_{t1}, LeftHip_{t1}, LeftFoot_{t1}, RightFoot_{t1})). \quad (4.7)$$

$$F^l = F(D^l) = F(D(HipCenter_{t1}, RightHip_{t1}, RightFoot_{t1}, LeftFoot_{t1})). \quad (4.8)$$

$$F^2 = \begin{Bmatrix} F^r \\ F^l \end{Bmatrix} \quad (4.9)$$

In the above definition, each function indicates respectively whether the right foot or the left foot is in front or behind the body. By combining both functions in a function vector, a simple but powerful geometric feature capable of recognizing actions with strong spatial variations is built. Although it is already possible to recognize walk actions with simply two features, it is normally desired to have additional features that are characteristic of the action in order to achieve better discrimination results when identifying several types of actions simultaneously. For instance, It is useful to define two additional features for a walk action which indicate weather the hands move sequentially forward and backward.

While choosing the features, it is also important to thoroughly decide the correct threshold values in order to give a meaningful geometric relation. Since the subjects performing an action have different limb lengths, it is also desired to normalize the pose before applying the geometric features. This is specially needed when the



(a) (b)

Figure 4.2: Visualization of Equation 4.9. (a) Right foot in front. (b) Left foot in front. Two geometric features useful for the recognition of a walk action

measures are based on distances. However, when using velocity functions instead of distances, it is normally sufficient to establish the threshold value as zero removing the need for normalization. For example, with a zero threshold, the Vn function would simply indicate whether the joint is moving closer or moving away from the given plane.

4.2 Clustering with Reciprocal Nearest Neighbour

Once a valid representation of body poses is obtained, either as joint angle descriptor or as geometric feature descriptor, a grouping of similar poses can be performed. This is done using the Reciprocal Nearest Neighbor (RNN) algorithm originally introduced in (de Rham, 1980). The main idea behind the algorithm is the formation of pairs that are reciprocally nearest neighbors. That is, given two vectors x_i and x_j, x_i is the closest neighbor of x_j and vice versa. Once such pair of vectors is found, they can be merged into one cluster.

An efficient implementation of the RNN is done by using NN chains as described in (Benzécri, 1982). A NN chain is a list composed by vectors arranged so that the

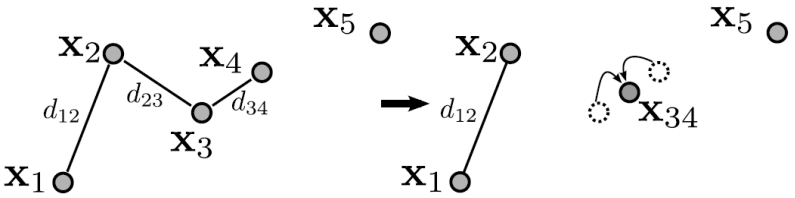


Figure 4.3: A simple representation of the RNN algorithm.(Leibe *et al.*, 2008).

next element in the list is always the NN of the previous element. Consequently, a NN chain of length l can be defined as $\{x_1, x_2 = NN(x_1), \ldots, x_{l-1}, x_l = NN(x_{l-1})\}$ where $NN(x_i)$ is the nearest neighbor of x_i. Given that the distances decrease from one element to another, the RNN pair is found at the end of the list.

Referring to Figure 4.3, the clustering algorithm proceeds as follows: Initially, an arbitrary vector x_1 is chosen from which the NN chain is built resulting in $d_{12} > d_{23} > d_{34}$. When the first fourth elements are in the list, it is found that $d_{45} > d_{34}$, therefore no more elements can be appended to the end of the list and a RNN pair is found. If the similarity between x_3 and x_4 exceeds a minimum threshold, both vectors are merged into a cluster. Otherwise, the chain is ignored.

By merging elements in this way, the Bruynooghe's reducibility property has to be fulfilled. This property states that when two clusters c_i and c_j are merged, the distance of the new cluster to any other cluster c_j may only increase:

$$D(C_i, C_j) \leq min(D(C_i, C_k), D(C_j, C_k)) \leq D(C_{i \cup j}, C_j). \tag{4.10}$$

It can be demonstrated that this condition is fulfilled for the group average criterion and the centroid criterion based on correlation. This condition guarantees that after the merging of clusters, The NN chain for the remaining elements stays unaltered and thus can be used for the next iteration. The process continues until the NN chain has no more elements or until the chain is ignored. In either case, a new arbitrary vector is chosen and a new NN chain is built.

Every time that a new cluster is built, its distances to the other clusters have to be recomputed. If the cluster similarities can be expressed in terms of centroids, the computation can be performed in constant time and only the mean and variance of each cluster need to be stored. This is the case for the group average criterion, whose similarity, using Euclidean distances, can be computed as $sim(X, Y) = ((\sigma_x^2 + \sigma_x^2) + (\mu_x - \mu_y)^2)$. Where $X = \{x^{(1)}, \cdots, x^{(N)}\}$ and $Y = \{y^{(1)}, \cdots, y^N\}$.

In this way, both the new mean and variance can be computed in an incremental manner:

$$\mu_{new} = \frac{N\mu_x + N\mu_y}{N+M} \tag{4.11}$$

$$\sigma^2_{new} = \frac{1}{N+M}\left(N\sigma_x + N\sigma_y + \frac{NM}{N+M}(\mu_x - \mu_y)^2\right) \tag{4.12}$$

The entire clustering algorithm results in a $O(N^2 d)$ time and $O(N)$ space complexity (Leibe *et al.*, 2008).

4.3 Time-Invariant Action Recognition

Since the duration interval of an action class can considerably differ between subjects, the time scale is an important issue that has to be confronted in human action recognition approaches. Thus, similarly to scale invariance in object recognition, time invariance in action recognition is a fundamental requirement for an optimal classification and performance.

In comparison to more complicated methods for achieving time invariance such as Dynamic time wrapping (DTW), the probabilistic nature of the ISM allows for a simple inclusion of action recognition at different time scales. The invariance is implicitly accomplished by using training data with different actions at of the same class at multiple time scales. During classification, the training sequence with the most similar time scale in relation to the test image, will generate higher peaks in the voting space with smaller variance.

Chapter 5

Implementation

This chapter gives an overview of the most important modules implemented in the action recognition framework. The whole framework can be seen as a pipeline composed of four main stages (see Figure 5.1):

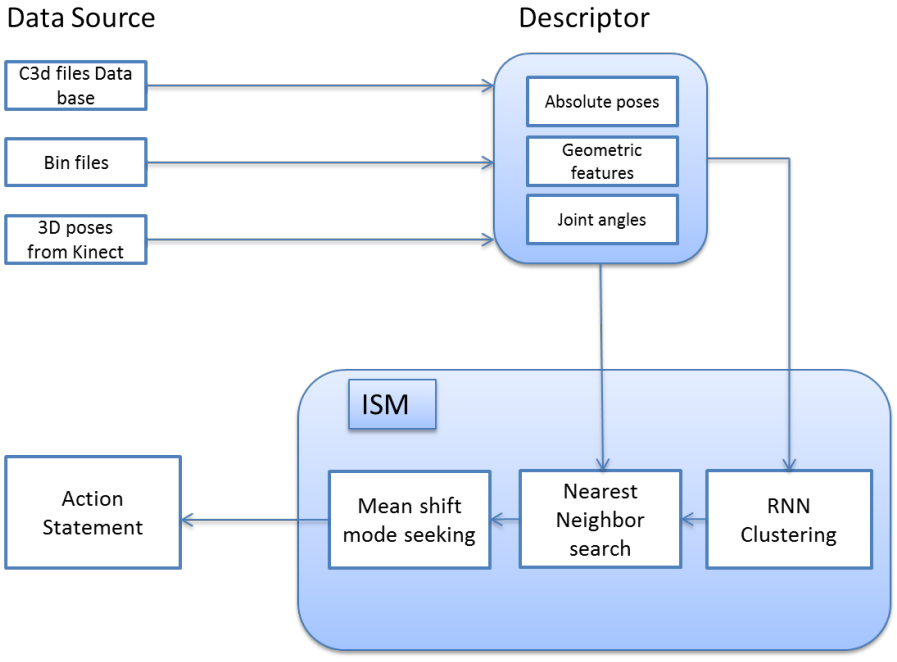


Figure 5.1: Stages of the action recognition framework. The input from different sources represents sequences of 3D coordinates of markers in the body. After processing each frame or a group of frames, an action statement is given.

The first module consist of the hardware used for the real time acquisition of three dimensional body markers and the software that interprets the input data and forwards it to the next stage. In second place, a module is implemented whose tasks is to create a descriptor out of the body markers. Subsequently, a module for loading and editing the training data is implemented. Having a descriptor of the body pose and the training data, the voting stage performs a comparison and generates votes in the Hough space for each of the possible actions being performed. Finally, a mean shift mode seeking is performed in order to find maxima in the Hough space and give a clear action guess. In the following sections, a more detailed description of the pipeline is given.

5.1 Data Acquisition

In this stage, the user has the possibility to chose the source of data for action recognition task. A database with different types of human actions represented as sequences of 3D coordinates is available in C3D format (*CMU: Carnegie-Mellon MoCap Database.* 2003). Additionally, The recognition can be performed in real time using the Microsoft Kinect depth sensor. The following sections give a detailed description of the two different data sources and their integration in the framework.

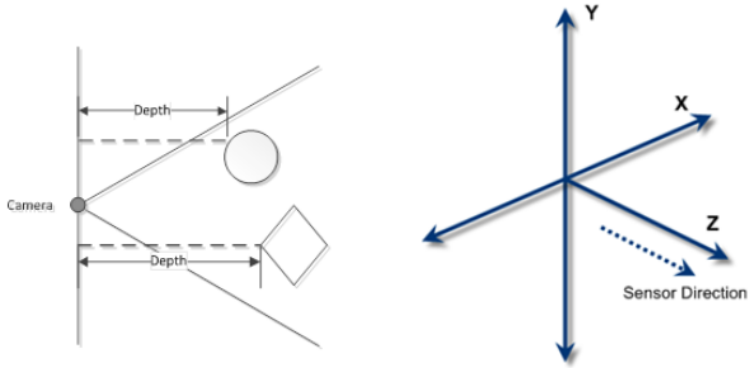
5.1.1 Motion Capture Database

The database used for training and testing consist of C3D files for different action types. The C3D is a binary file format typically used in Biomechanics, Animation and Gait Analysis laboratories to record synchronized 3D and analog data. The format provides a unique standard for storing raw 3D data and analog sample data, together with information that describes the stored data.

5.1.2 Integration with Kinect Depth Sensor

The real time data is acquired using the Microsoft Kinect depth sensor. This device provides three different types of data streams: color images, depth images and the skeleton coordinates.

The color image stream delivers 32 bit RGB images at a resolution of 640×480 pixels at up to 30 frames per second(FPS) or a resolution of 1280×1024 pixels at up to 10 FPS. The depth image streams provides data made up of pixels that contain, for each (x, y) coordinate in the depth sensor's field of view, the distance in millimeters from the camera plane to the nearest object. This stream additionally



(a) (b)

Figure 5.2: (a) Acquisition of depth image data with Microsoft Kinect depth sensor. The data contains (x, y) coordinates with the distance from the camera plane to the nearest object. (b) The 3D right-handed coordinate system used by Microsoft Kinect depth sensor for representing skeleton 3D coordinates.

contains player segmentation data indicating the index of a unique player detected in the scene. The depth image is available in 3 different resolutions: 640×480, 320×240, and 80×60 pixels. Finally, each frame of the depth image is processed by the Kinect runtime into skeleton data. This data contains the 3D coordinates of maximum two human skeletons that are visible in the depth sensor. The coordinates are expressed in meters and the x,y, and z axis are oriented as shown in Figure b.

The three different data streams are integrated into the framework. This is done using a separated thread from the main application which constantly reads the data from the sensor and sends it to the main window of the application where it can be visualized. The skeleton coordinates and the depth image are rendered in three dimensional frames (see Figure 5.3) in which the point of view may be translated and rotated. Additionally, the tilt of the sensor can be set for angles between -27 and +27 degrees in order to change the filed of view. Although the depth image and colour image are important for visualization purposes, only the skeleton data is sufficient for the action recognition procedure. If real time mode is selected, then the Kinect sensor is chosen as the source of the data. During real time recognition, k number of frames are collected into a batch and passed to the next stage. Subsequently, each video frame in the batch is transformed into a descriptor representation passed to the next stages. Each time a batch is processed, an action statement can be given. This procedure repeats for the next incoming frames from the sensor.

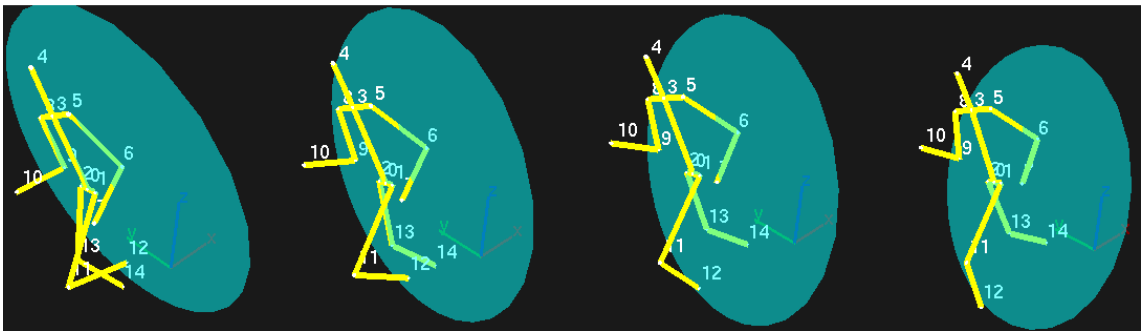


Figure 5.3: Sequence of a run action. A plane between the hips and the left foot is rendered in order to find characteristic patterns as relations between the plane an other joints

5.2 Descriptor Types

The GUI allows one to choose between three different type of descriptors: absolute, joint-angle and geometric descriptor. Having chosen one of the three descriptors, the values may be visualized in a table or as a grayscale image. The absolute descriptor represents the data as simple normalized 3D coordinates while the joint angle descriptor represents the body pose as a set of angles and optionally their first and second derivatives (see Section 3.1.2). Lastly, the geometric descriptor represents boolean features corresponding to geometric relations between the body limbs (see Section 4.1). Different geometric measures may be chosen by specifying their values in an xml file configuration. Here, for each geometric measure, a function type and its parameters is given. The functions in Table 5.1 are available.

Derivatives of each function in Table 5.1 are also available as geometric features. In order to create a boolean vector of features out of the measures, a threshold value has to be specified together with the function type and its parameters. Additionally, the skeleton rendering frame can be used as a visual aid for selecting geometric measures. This is done by drawing normal planes and vectors around the skeleton (see Figure 5.3).

Similarly, when the joint-angle descriptor is selected, an xml file specifies which relative angles are to be included in the descriptor and whether the first and/or second time derivatives of the angles between specific limbs are considered in the descriptor.

Table 5.1: Geometric functions definition.

Feature	Parameters	Description
DistanceToPlane	joint1, joint2, joint3, joint4.	Distance between $joint4$ and a plane formed by $joint1$, $joint2$ and $joint3$ (Equation 4.2).
DistanceToPlaneVector	joint1, joint2, joint3, joint4.	Distance between the fourth joint and a plane that is normal to the vector $\vec{n} = (joint2 - joint3)$ and contains $joint4$.
VelocityToPlane	joint1, joint2, joint3, joint4.	Velocity $\vec{v}$ of joint4 in the normal direction of the plane formed by $joint1$, $joint2$ and $joint3$ (Equation 4.3)
VelocityProjection	joint1, joint2, joint3	Projection of the velocity vector $\vec{v}$ of $joint1$ over the vector $\vec{b} = (joint3 - joint4)$. (Equation 4.4)
DistanceBetweenJoints	joint1, joint2	Euclidian distances between $joint1$ and $joint2$ (see Equation 4.5),
AngleBetweenJoints	joint1, joint2, joint3, joint4	Angle between a pair of limbs specified as: $\vec{limb}1 = joint1 - joint2$ and $\vec{limb}2 = joint3 - joint4$. (Equation 4.6).

5.3 Training Stage

In order to learn new actions, the framework must be able to generate a codebook out of training video sequences containing an action type and a pair of frames to indicate where the action starts and ends. To this end, a training stage is implemented in the GUI. The interface allows loading and editing of C3D files as well as clustering of the loaded data. The user can load a single file and manually edit the action label, start frame and end frame with the help of a rendering frame showing the file currently loaded. The user has also the option of loading an xml file which already specifies the set of training files to be loaded with their respective action labels, start and end fames. Such file is referred as a training dictionary in the following sections.

Once the training data is loaded. It is converted to a descriptor whose type has to be the same as the one used for representing the test sequence. Otherwise the comparison between descriptors is not possible. With the data of all the training sequences converted to a descriptor type, the RNN clustering can be performed. Here, it can be specified whether the voting stages uses clusters or not.

5.4 Voting Stage

With both the test and training sequences converted to a suitable descriptor, the comparison and vote casting can be performed. The implementation is based on the previously described approach (Algorithm 2). The search of NN is performed in a separate thread using the FLANN library. It performs a fast approximate nearest neighbour search that can reach an improvement in speed of several orders of magnitude compared to other algorithms. Different parameters have to be set for the search depending on the type of data being handled. In this implementation a suitable metric distance have to be chosen depending on the descriptor type. For the joint-angle descriptor, Euclidean distance is used, whereas for the geometric descriptor, Hamming distances is used.

The interface also shows a plot of the voting space for each of the actions contained in the training data. The x axes represent the time (frame number) while the y axes indicate the number of votes. If the real time mode is selected, The main thread collects a certain number of frames (batch) and sends them to the voting stage. This event causes the plots of the votes density to update. It is possible that the sequences of frames arrives at a higher rate than it can be processed by the voting stage. In order to cope with this, a frame buffer is implemented in the main thread. It keeps accumulating the incoming frames in a queue and only release a batch when it has already been processed by the voting stage thread.

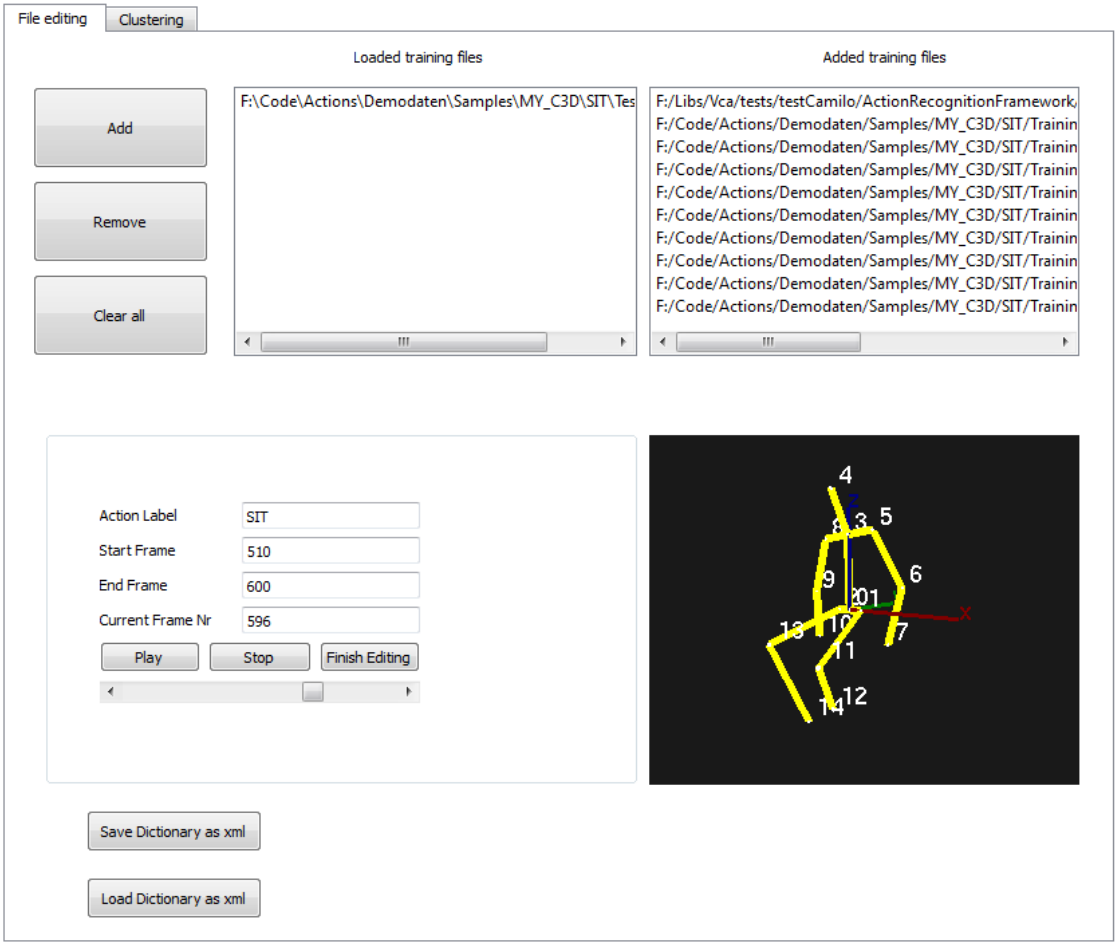


Figure 5.4: Training stage. Screen shot of the GUI for adding and editing training files. The 'Loaded training files' list on the left shows every single file that has been included with the 'Add' button for editing purposes. When a file of this list is selected, the corresponding skeleton poses are rendered allowing the user to choose the action label, start frame, and end frame. Once the file parameters have been edited, the 'Finish editing' button has to be clicked in order to add the file into the current training data set. When a file has been added, it is displayed in the 'Added training files' on the left side of the GUI. Additionally, the GUI provides the option of loading an entire set of training files with the 'Load dictionary as xml' button. Furthermore, The files that have been previously edited can be saved in a xml file with the 'Save dictionary as xml'.

5.5 Mean Shift Stage

The search for maxima in the voting space and its result is performed in this stage. Every time step the voting stage indicates when its results are ready and forwards them to the mean shift stage. This stage consist of approximating the vote density with Gaussian density kernels of a predefined variance value and subsequently finding the modes.

The resulting Gaussian approximation as well as the corresponding points where the maxima occur are shown in this stage. It is also possible to visualize the plots of the resulting maxima for each action that was present in the training data. With this information, it is already possible to give an action statement of where an action may end. As in the voting stage, the GUI allows the user to save the different plots.

5.6 Action Classification

While an action takes place, the mean shift algorithm estimates the rising modes of the vote distribution per action, see for example Figure 6.26. The final step to be performed is classifying the "good" modes, which vote for an end of an action, and the clutter. In order to do this a support vector machine (SVM) is trained per action class and during action recognition applied on its action class. The input features for the SVM are the amplitude of the nearest future mode and its current time. For example in Figure 6.20 both sit curves are the positive features, and all other values belong to the negative features. For simplicity we started with a linear kernel. After having trained a SVM on a training data set per action class in advance of the action recognition, they are applied to every next future mode during runtime.

Figure 5.5 visualizes the classifier score for a sit action with a SVM trained on sit data. The classification is obtained from that data by applying the signum function. That means, that after 40 frames (0.3 seconds) the classification raises the recognition of a sit action. All other classifiers (not shown) in that examples did not raise any positive action classification.

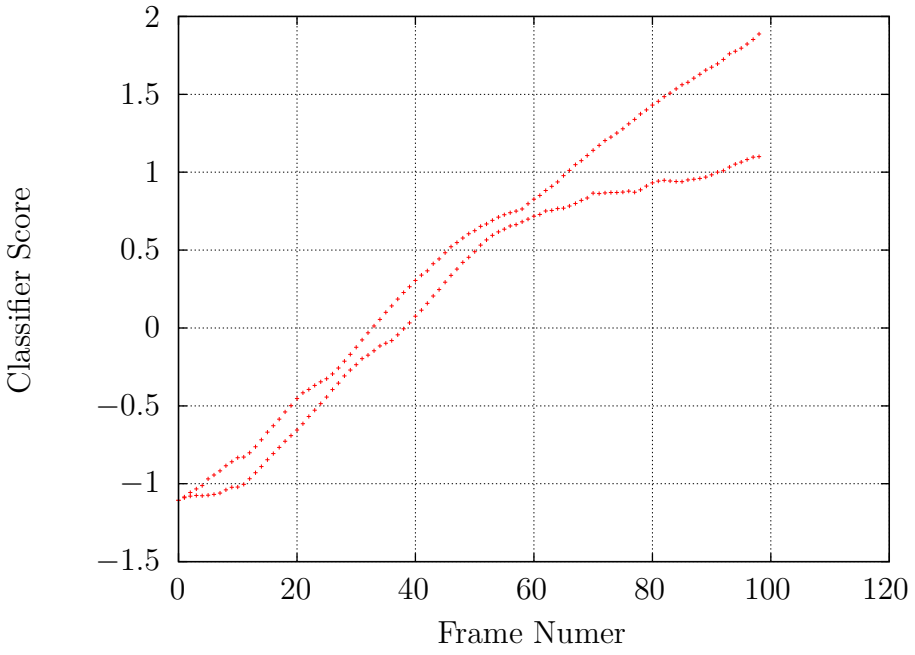


Figure 5.5: Classifier score for a sit action with a SVM trained on sit data. A positive classifier score indicates a sit action, a negative score not a sit action.

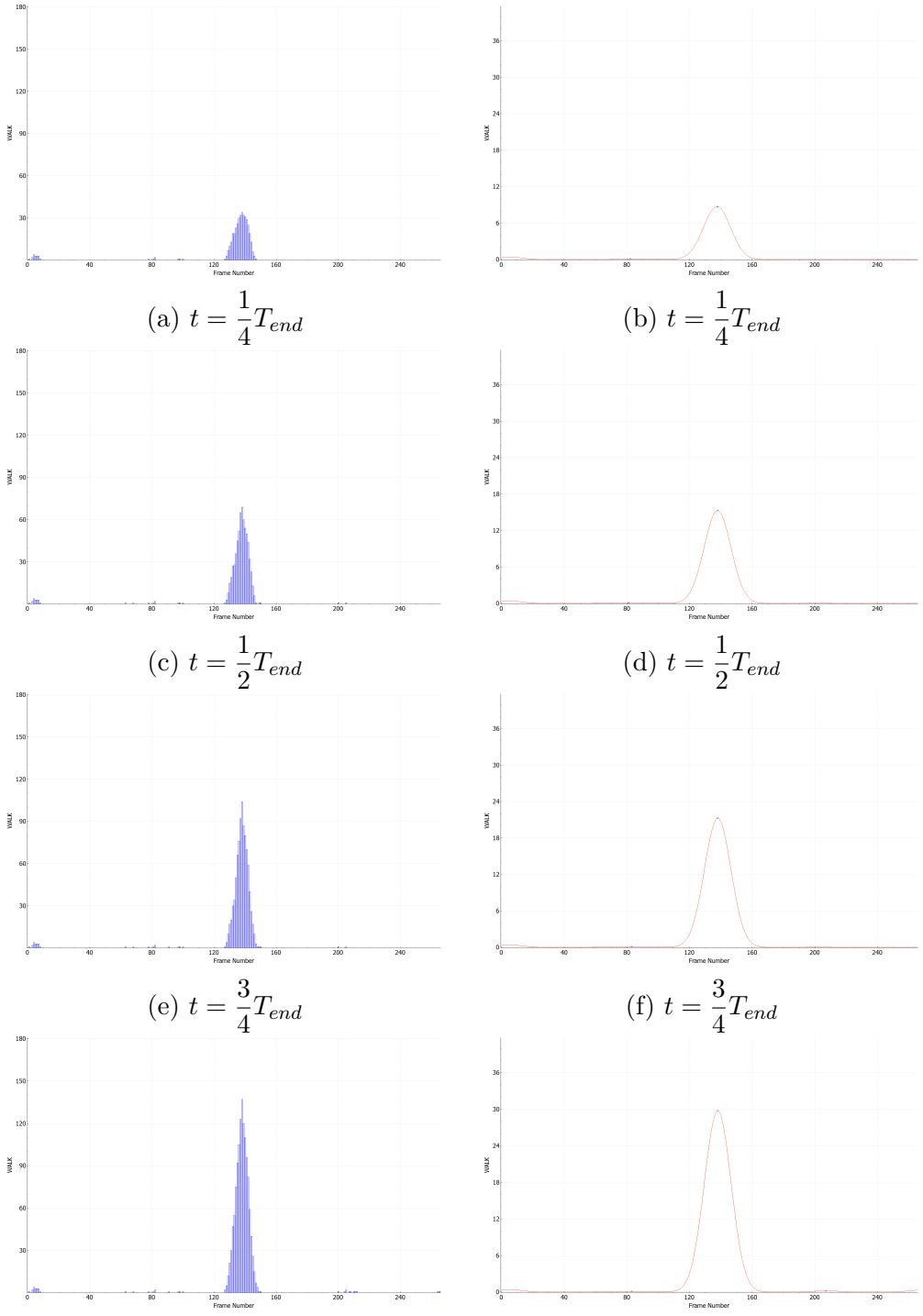
Chapter 6

Evaluation

In order to determine the performance of the developed method, different kinds of experiments were arranged: Section 6.1 presents experiments for verifying the correctness of the implementation of the descriptors. In Section 6.2, the intra-class variance is evaluated for five different kinds of actions. Section 6.3 shows the classification results for the case when the training data set consists of different types of actions. The experiments in Section 6.4 evaluate the time-invariance of the framework for different time scales of one specific action. Section 6.5 presents performance results on the data acquired with the Microsoft Kinect depth sensor. Finally, in Section 6.6 the real time capabilities of the framework are assessed.

6.1 Verification of the Correctness of the Implementation

Before performing any real action recognition tests, it is important to verify that the framework was implemented correctly. This is done as follows: a single video sequence V_i containing an action A_k is introduced in the dictionary as training data. Subsequently, the exact same video sequence V_i is chosen as the input data. The input sequence is compared to the training sequence frame by frame. For each processed frame, the descriptor votes for the end of the action. Subsequently, the mean shift mode seeking is performed on the generated voting space and the next future mode value is extracted (Figure 6.1). The point in time where this peak occurs, indicates where the action may end in the future, and the amplitude of the peak indicates how strong the prediction is. In this way, in every time step, the maximum amplitude can be visualized. This experiment was performed with different types of actions: walk, jump, run, sit, and wave. In each action class, the joint-angle descriptor as well as the geometric descriptor were tested.



(a) $t = \frac{1}{4}T_{end}$ (b) $t = \frac{1}{4}T_{end}$

(c) $t = \frac{1}{2}T_{end}$ (d) $t = \frac{1}{2}T_{end}$

(e) $t = \frac{3}{4}T_{end}$ (f) $t = \frac{3}{4}T_{end}$

(g) $t = T_{end}$ (h) $t = T_{end}$

Figure 6.1: Visualization of the mean shift density estimation. The set of figures from (a) to (h) show the progress of the voting densities (left column) with their corresponding mean shift mode estimations (right column) at four different time steps for a walk action. The values for the amplitude plots used in the following figures are derived from the modes of the mean shift density estimation in every time step.

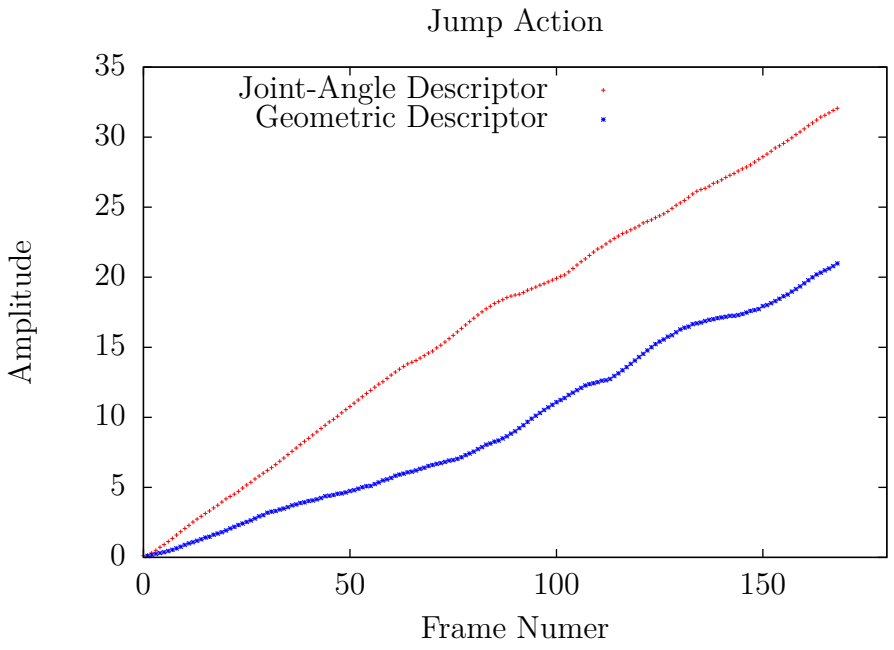


Figure 6.2: Verification of jump action for joint-angle and geometric descriptors with mean shift algorithm variance $v = 15$.

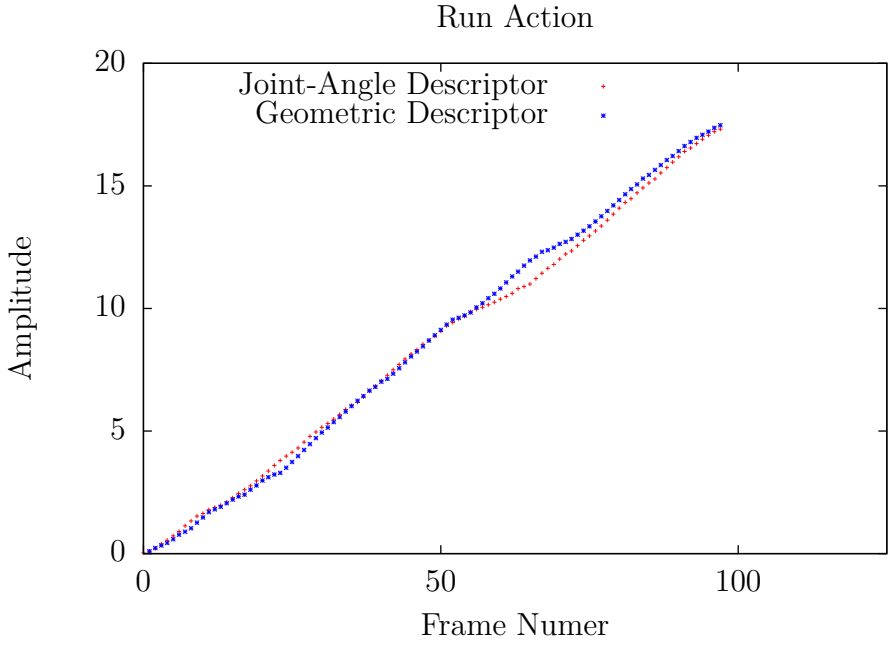


Figure 6.3: Verification of run action for joint-angle and geometric descriptors with mean shift algorithm variance $v = 15$.

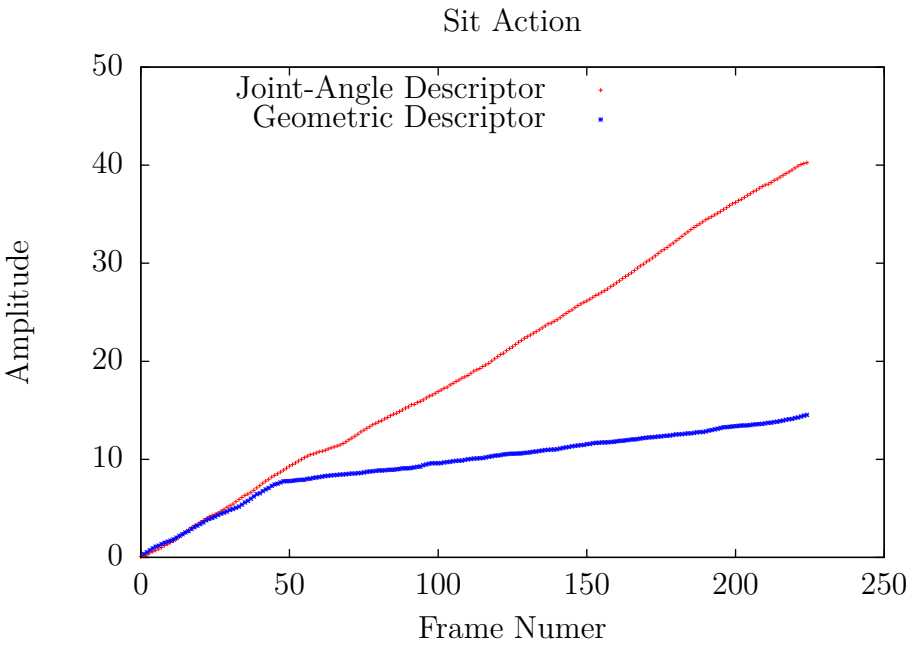


Figure 6.4: Verification of sit action for joint-angle and geometric descriptors with mean shift algorithm variance $v = 15$.

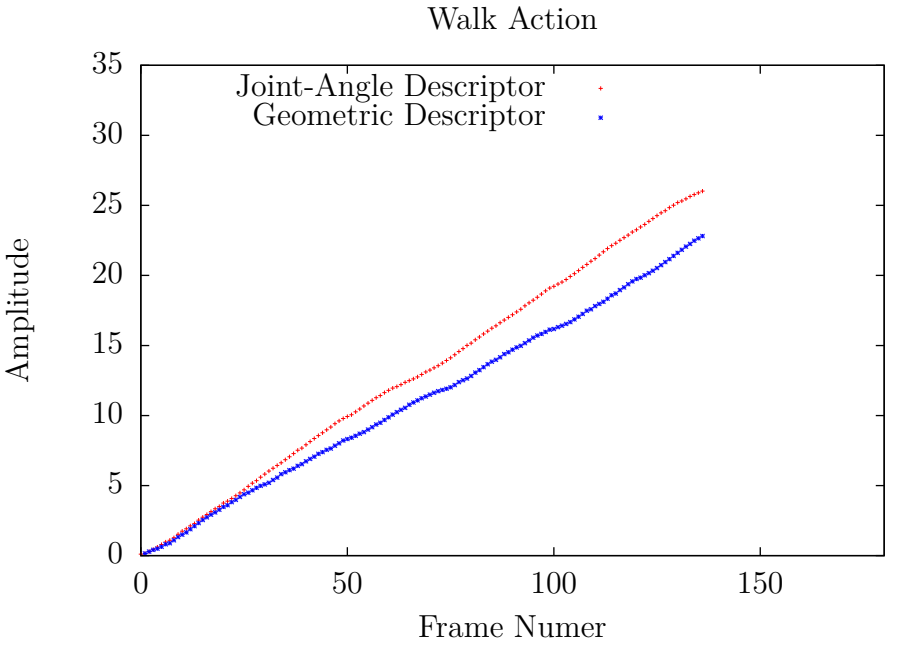


Figure 6.5: Verification of walk action for joint-angle and geometric descriptors with mean shift algorithm variance $v = 15$.

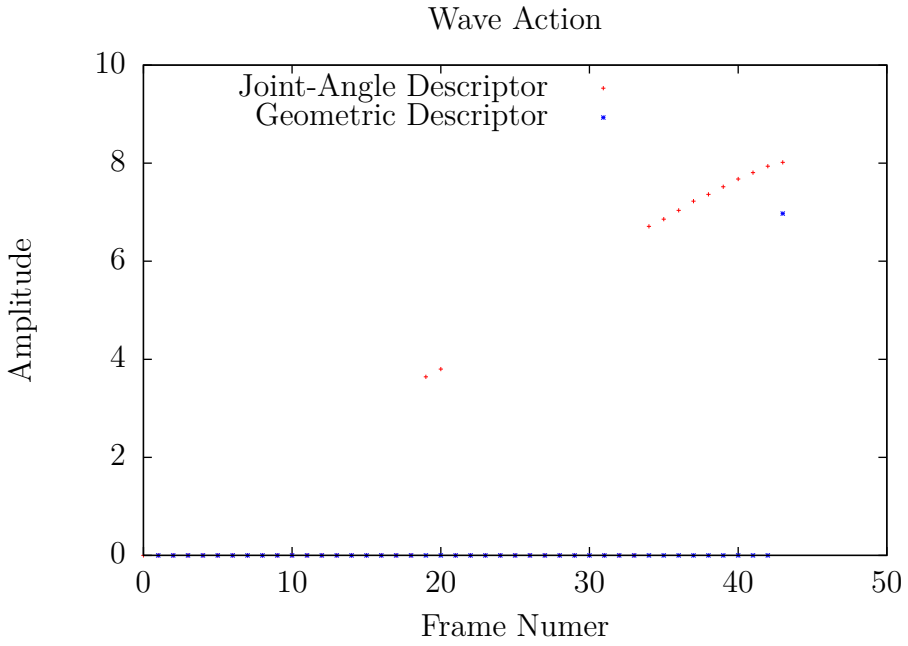


Figure 6.6: Verification of wave action for joint-angle and geometric descriptors with mean shift algorithm variance $v = 15$.

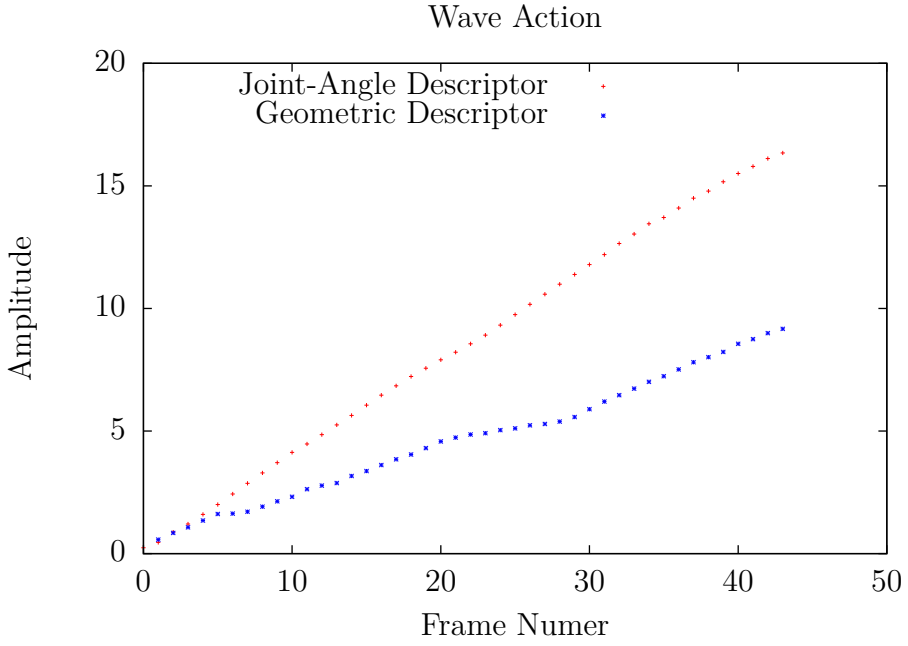


Figure 6.7: Verification of wave action for joint-angle and geometric descriptors with mean shift algorithm variance $v = 5$.

Each amplitude curve in Figure 6.2 to Figure 6.7 looks approximately as a straight line. These is the expected behaviour for a correct implementation. Since the sequence is being compared with itself, the nearest neighbour of each frame descriptor in the input sequence always matches its corresponding frame descriptor in the training sequence. Thus, one vote for the end of the action at t_{end} is generated every time step and the density of the votes increases linearly as more frames are processed. It can be seen that the joint-angle descriptor gains better performance whereas the geometric descriptor misses some matches due to quantization. The only exception for such behaviour is present for the wave action. Here, the amplitude values remain in zero for the majority of time steps and only exhibit a linear behaviour during a short time interval in the case of the joint-angle descriptor. This behaviour can be explained by the relatively big difference between the duration of a wave action with respect to other actions. The shorter an action is, the smaller the variance around the end time of the action becomes. It means that, the mean shift mode seeking will not always find maximum amplitude values of every density function since the algorithm only finds modes that correspond to an initially fixed variance value. If the variance present in the data cannot be modelled with the variance value being used by the algorithm, the amplitude value is simply set to zero. When the variance is set to a smaller value, a straight line in the amplitude diagram is also obtained for the wave action (see Figure 6.7).

Having verified the basic functionality of the framework, it is now possible to test the performance.

6.2 Intra-class Variation

This test has the purpose of evaluating the recognition of single actions classes in order to investigate how similar are all actions of the same type among themselves. The more similar actions are, the better results are obtained with smaller training data sets since the action can be represented with fewer samples.

For each action class, a training dictionary is created containing several samples of only the action class being tested. One of the actions present in the training dictionary is then used as input. As in Section 6.1, the maximum peak amplitude at each time-step is recorded and plotted. Furthermore, the mean shift density estimation for the last time step is shown in each experiment. This experiment was performed for pairs of actions for the joint-angle descriptor without clustering, joint-angle descriptor with clustering and geometric descriptor with clustering.

Figures 6.8, 6.9, and 6.10 depict the results of the intra-class variation experiments. In general, each experiment shows that the maximum peak amplitude of the mean

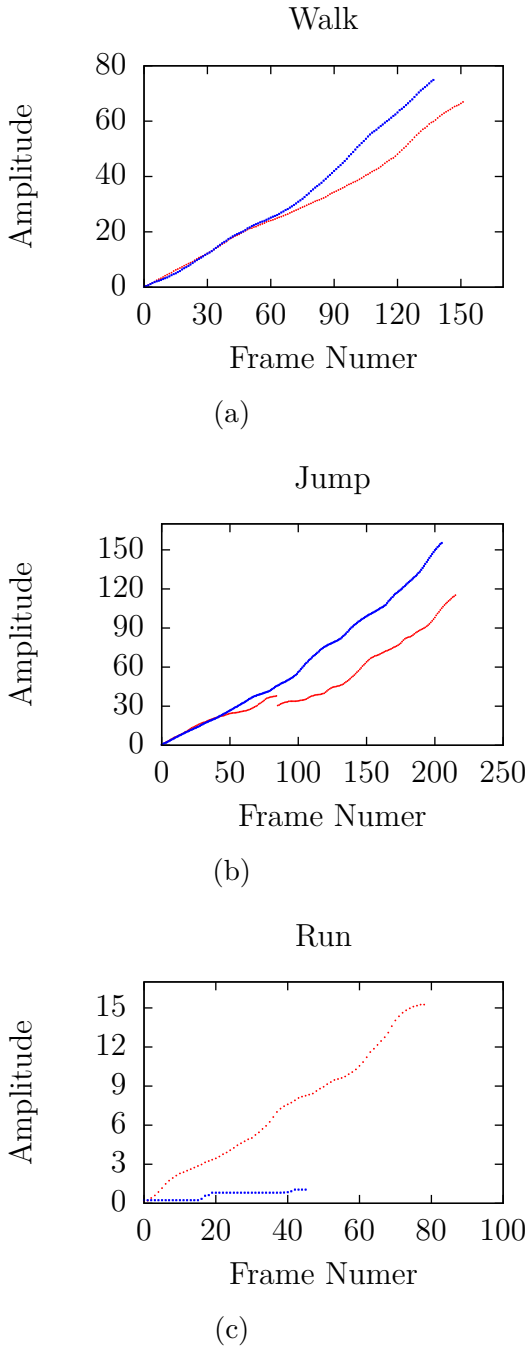


Figure 6.8: Intra class Experiment 1. Mean shift peak amplitudes of nine different actions using the joint-angle descriptor without clustering. Walk actions (a). Jump actions (b). Run actions: (c).

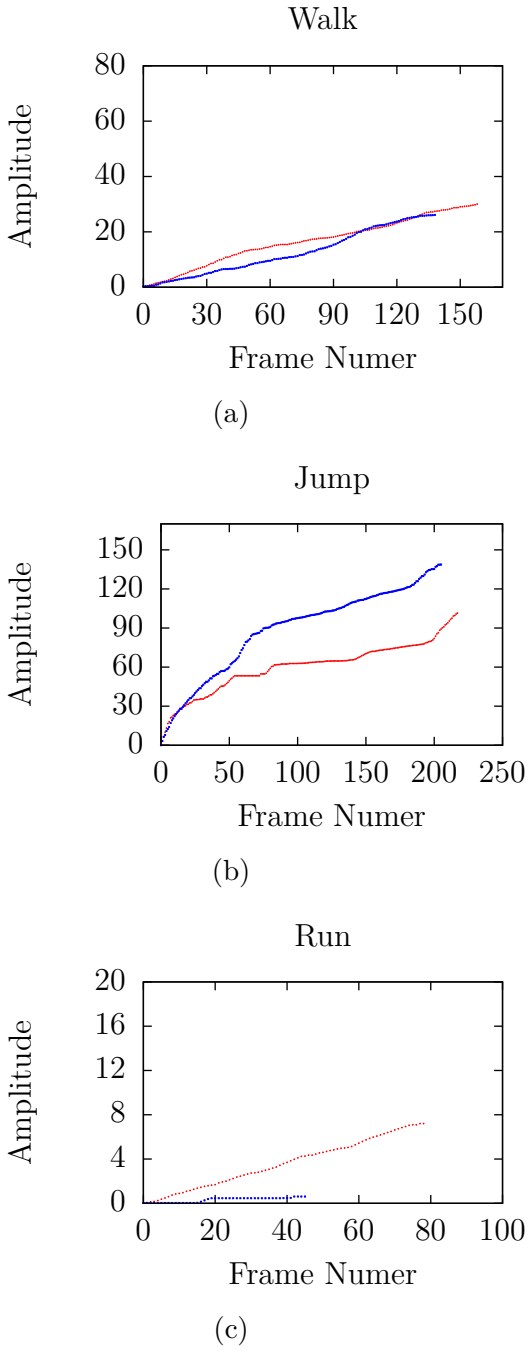


Figure 6.9: Intra-class Experiment 2. Mean shift peak amplitudes of nine different actions using the joint-angle descriptor with clustering. Walk actions (a). Jump actions (b). Run actions (c).

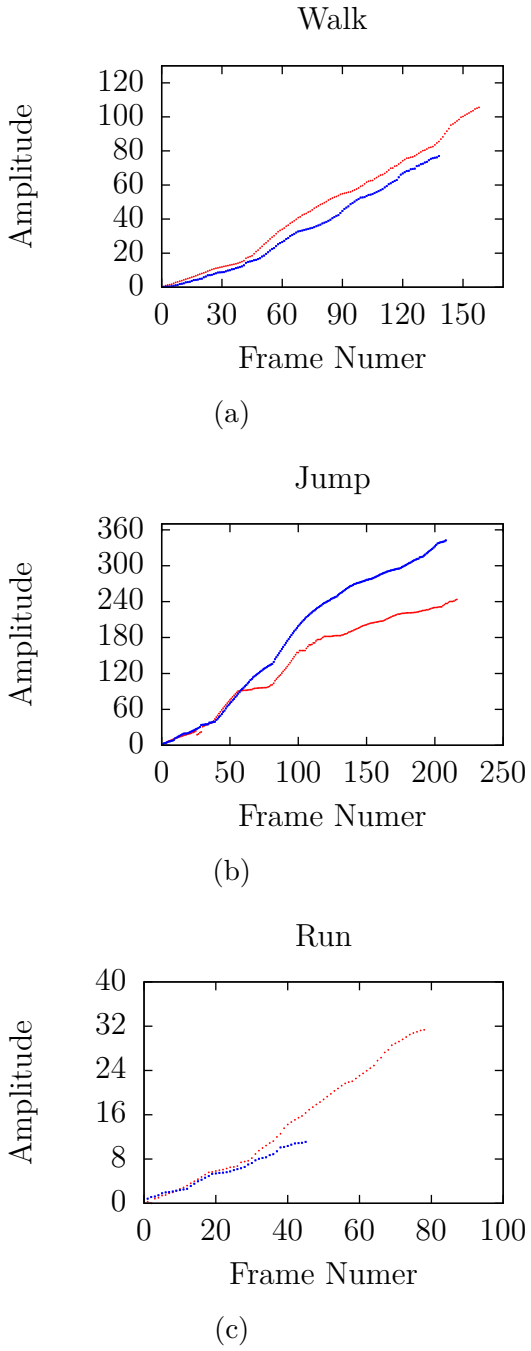


Figure 6.10: Intra-class experiment 3. Mean shift peak amplitudes of nine different actions using the geometric descriptor with clustering. Walk actions (a). Jump actions (b). Run actions (c).

shift increases along the time axis in the majority of the cases independently of the descriptor used. However, one can notice, that two actions of the same class exhibit a slightly different amplitude pattern. Such behaviour is normally expected since the descriptors employed for representing the frames of two different actions of the same class can significantly vary among themselves.

In the case of a 'run' action, the geometric descriptor outperforms the join-angle descriptor. This shows that in some cases, the geometric descriptor is capable of achieving a better generalization over bigger variations of the test data with respect to the training data. Another important advantage of the geometric descriptor is the generally shorter time required for processing each frame of the input sequence. This is partially explained by the relative simplicity of obtaining the geometric features. As opposite to the joint-angles, the calculation of geometric features does not require pose normalization since such features already express semantic relations between joints that are point-view independent. Another time advantage of the geometric descriptor is the reduced number of features that it normally has in comparison with the joint-angle descriptor. Consequently, the dimensionality of the search for similar poses is reduced, and thus the processing time becomes shorter.

6.3 Discrimination of Different Actions

Having evaluated recognition for individual actions, it is now necessary to find out how well the framework performs when it is tested under more realistic conditions. That is, when the training dictionary is composed by several different action classes and the input sequences can represent any of those action classes contained in the training data. In this case, an optimal performance of the framework requires that a correct action classification can be clearly given for any input video sequence as long as it represents one of the action classes given in the training data.

As in the previous tests, five action classes are evaluated: walk, run, jump, sit, and wave. However, this time, several video sequences of the five action classes are included in the dictionary. The maximum peak amplitudes of the density function are shown over time for pairs of actions. This experiment was performed for the joint-angle descriptor without clustering, joint-angle descriptor with clustering and geometric descriptor with clustering.

A number of conclusions can be derived from Figures 6.11 to 6.25. In general, the results for most of the cases show that actions can be discriminated. That shows, based on the mean shift amplitude data, it is possible to clearly discriminate between the action being performed and the other actions present in the training data set. A thorough analysis of the amplitude data shows some particularities for certain cases.

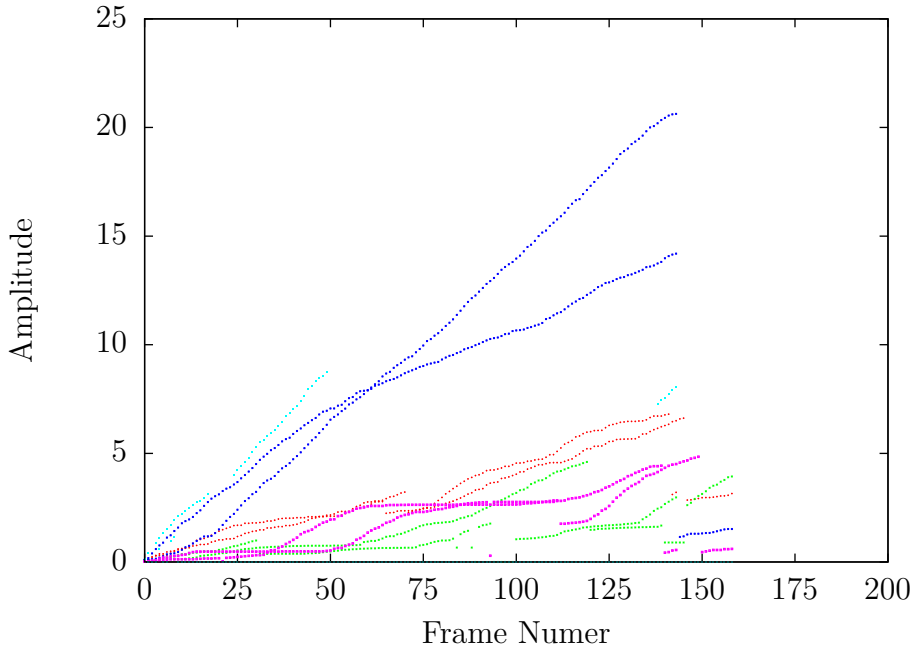


Figure 6.11: Mean shift peak amplitudes for 'walk' (blue) using joint-angle descriptor without clustering.

For example, in Figure 6.11, it can be observed that undesired amplitude levels appear. Having a closer look at Figure 6.11 unveils that the confusing wave action is performed during the person is walking. This noise levels, which correspond to actions different than the walk action, are higher when compared to the noise levels in Figures 6.12 and 6.13 where a clustering for codebook generation is applied. This statement is also valid for the jump action as well as for the sit action. Given that the use of clustering achieves a better discrimination (bigger difference between the amplitudes of the truth action and other actions), the SVM can more precisely classify an action. Another important remark can be made by comparing the joint-angle and geometric descriptors. In particular, the true amplitudes of action pairs with the same class label in Figures 6.13, 6.16, 6.19 appear more similar one to another than in the case of joint-angle descriptor. As mentioned before, such capability of generalization is one of the advantages of the geometric descriptor.

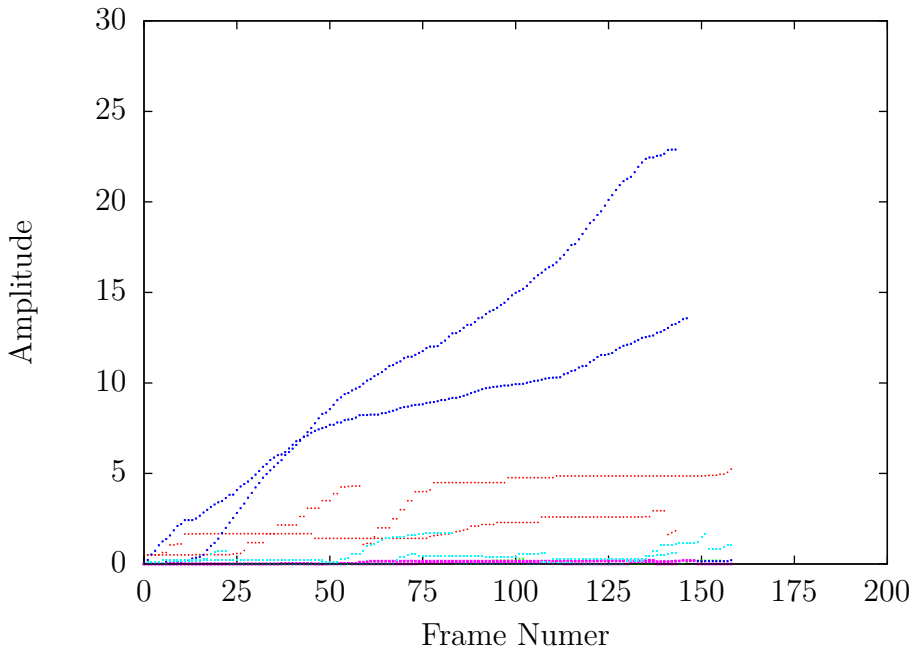


Figure 6.12: Mean shift peak amplitudes for 'walk' (blue) using joint-angle descriptor with clustering.

6.4 Temporal Invariance of the Descriptors

Another important performance indicator of an action recognition framework is defined by how well an action performed at different speeds can be recognized. In this experiment, tests were performed for the joint-angle descriptor with clustering for three different video sequences, each one containing several cycles of a walk action. A cycle of a walk action being defined as two strides. In each video sequence, cycles of the action were approximately of the same length, but the average cycle length Ca_i between video sequences varied considerably. Figures 6.26 to 6.27 show the results of this experiment. These figures depict the mean shift amplitude function over time for each case at four different time steps. Additionally, the ground truth of each action is shown as a blue box. Each box corresponds to the interval during which the action is being performed.

The performed experiments visualized in Figures 6.26 to 6.28 show that the proposed framework is capable to deal with action recognition at different temporal scales given that a proper training data set is provided.

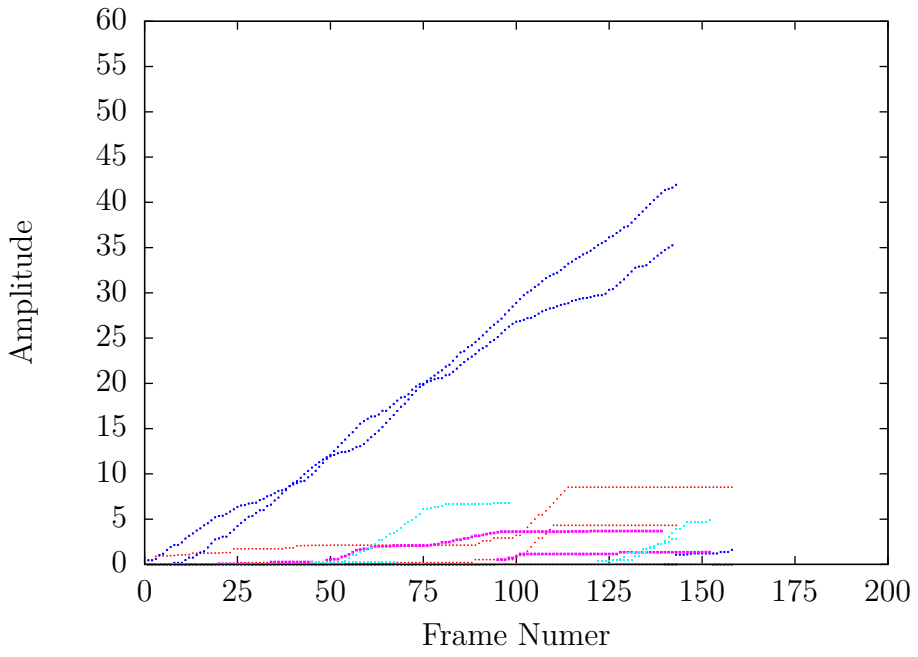


Figure 6.13: Mean shift peak amplitudes for 'walk' (blue) using geometric descriptor with clustering.

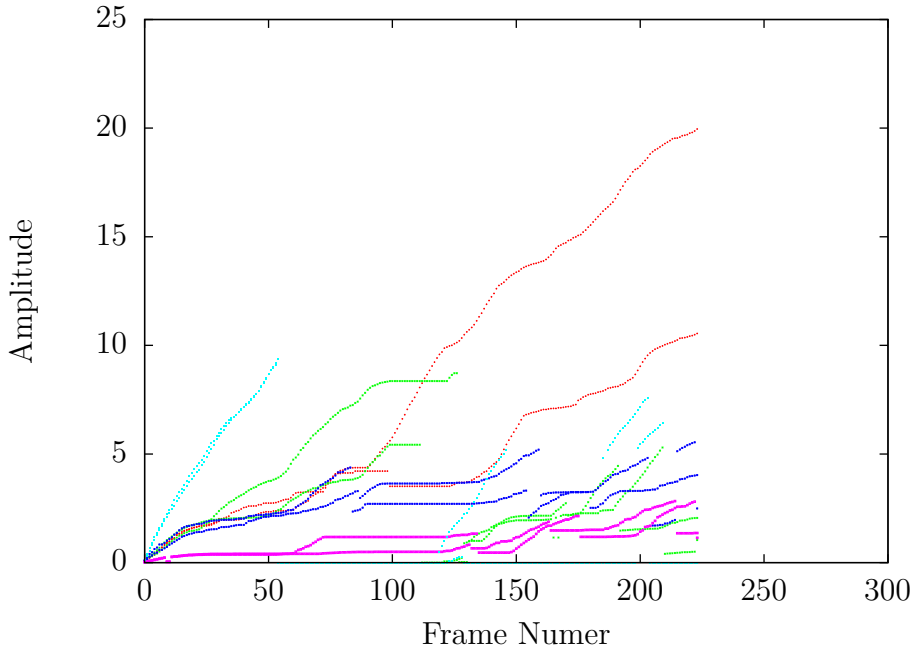


Figure 6.14: Mean shift peak amplitudes for 'jump' (red) using joint-angle descriptor without clustering.

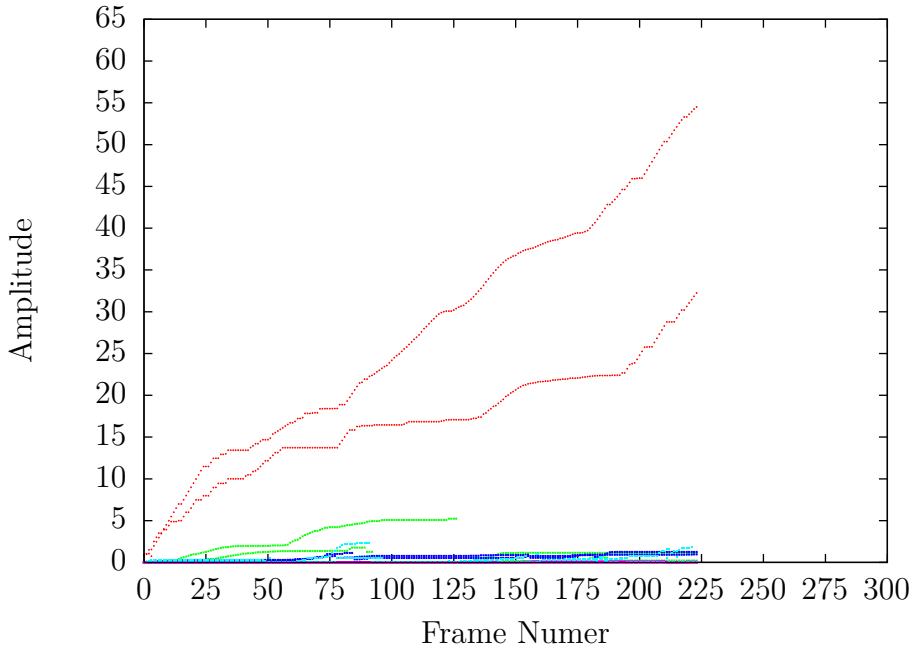


Figure 6.15: Mean shift peak amplitudes for 'jump' (red) using joint-angle descriptor with clustering.

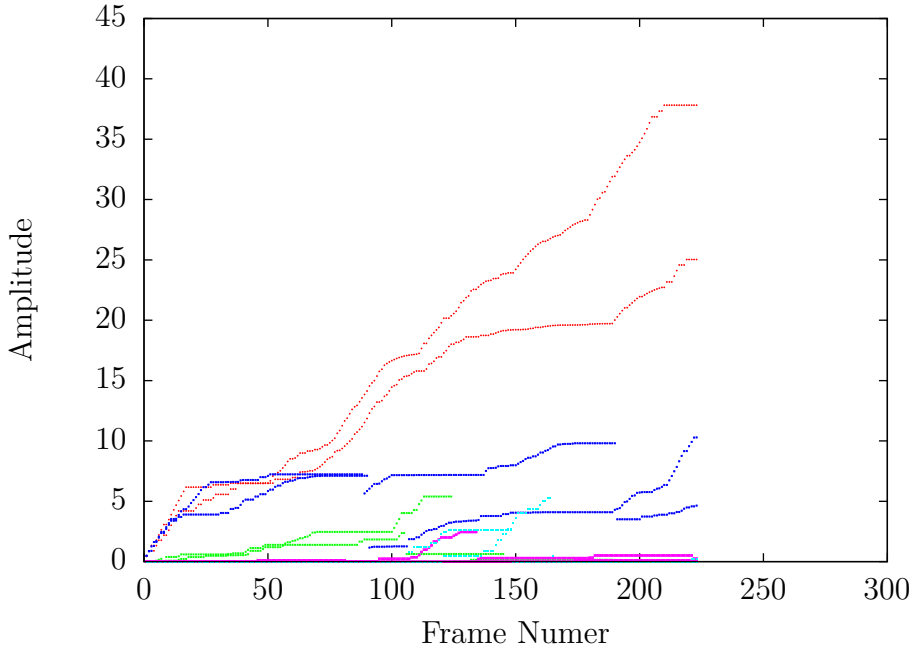


Figure 6.16: Mean shift peak amplitudes for 'jump' (red) using geometric descriptor with clustering.

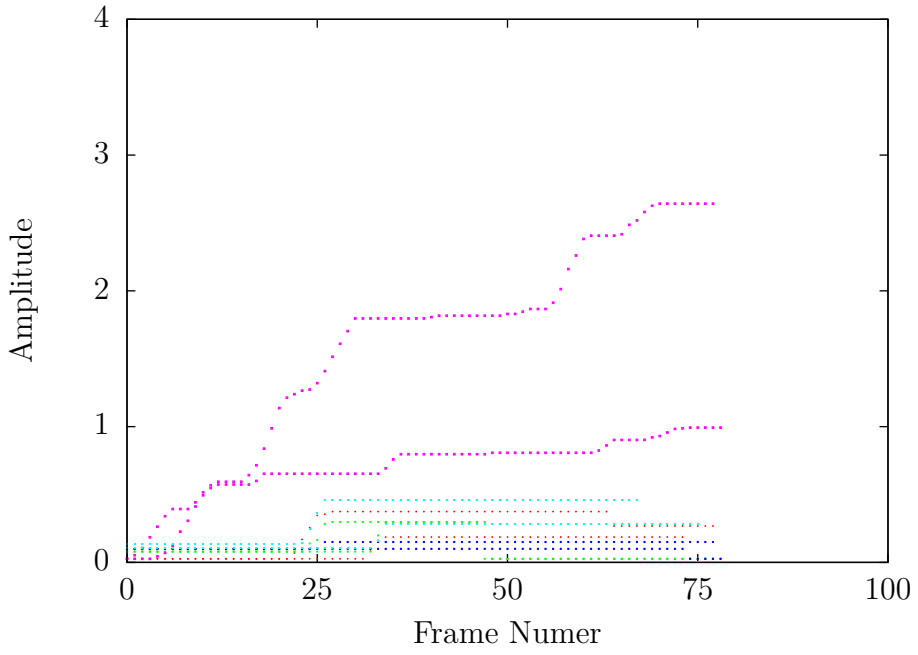


Figure 6.17: Mean shift peak amplitudes for 'run' (purple) using joint-angle descriptor without clustering.

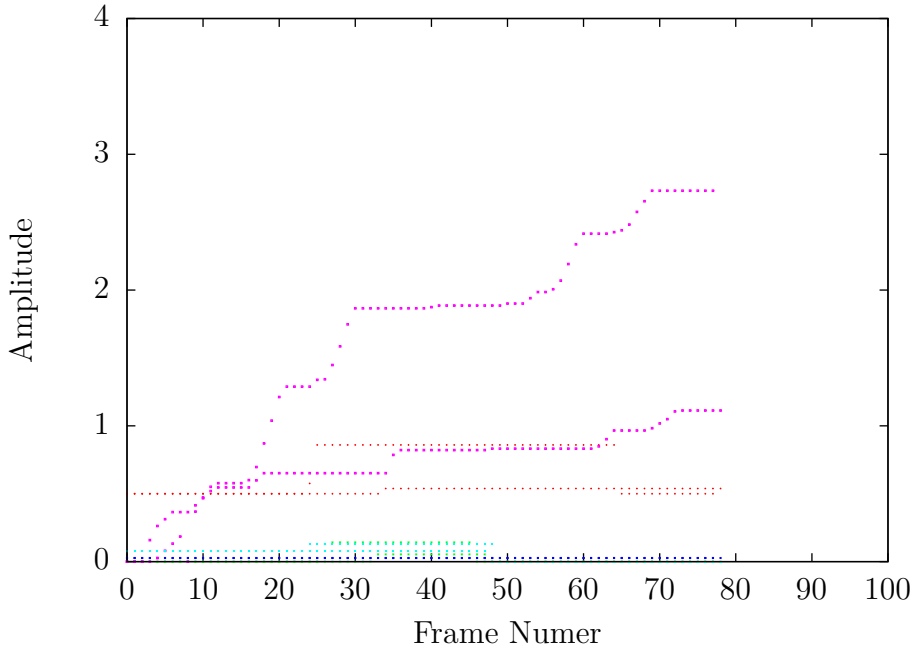


Figure 6.18: Mean shift peak amplitudes for 'run' (purple) using joint-angle descriptor with clustering.

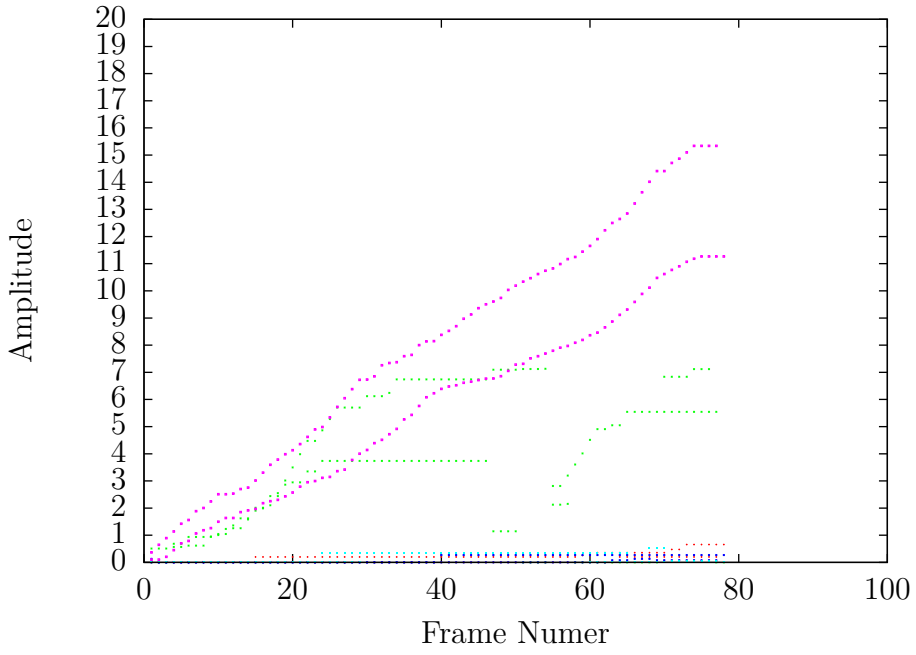


Figure 6.19: Mean shift peak amplitudes for 'run' (purple) using geometric descriptor with clustering.

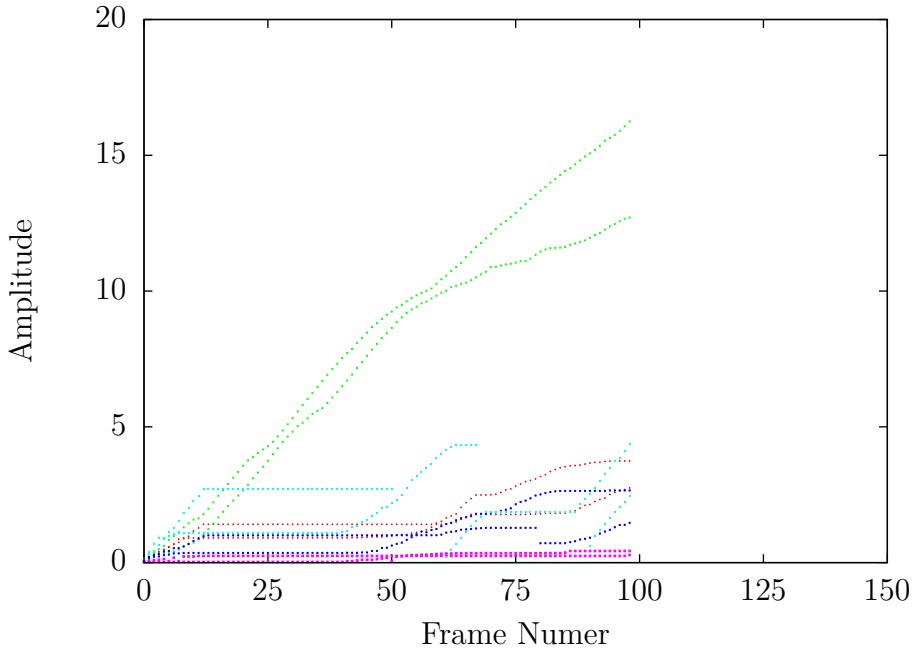


Figure 6.20: Mean shift peak amplitudes for 'sit' (green) using joint-angle descriptor without clustering.

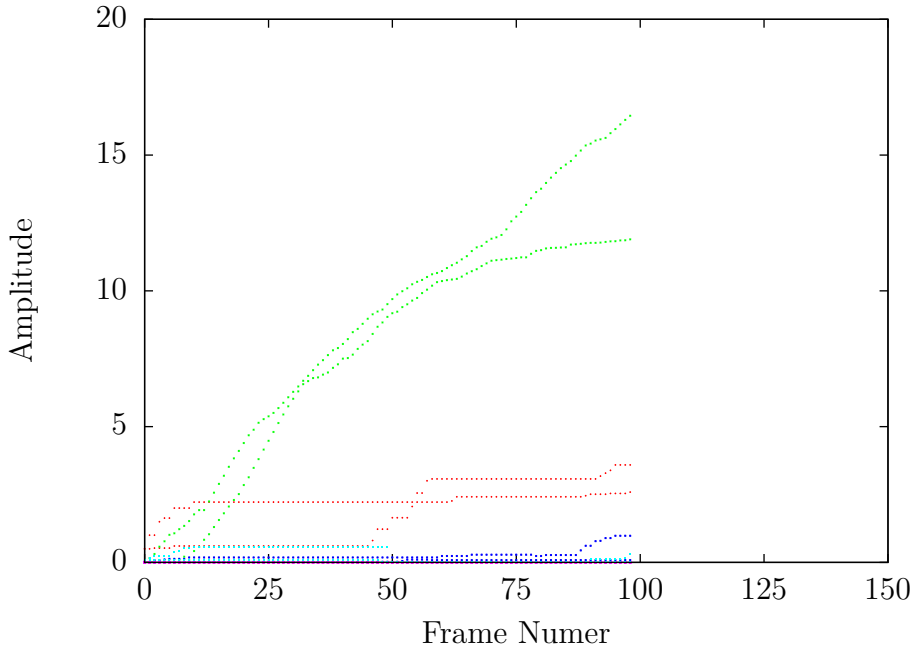


Figure 6.21: Mean shift peak amplitudes for 'sit' (green) using joint-angle descriptor with clustering.

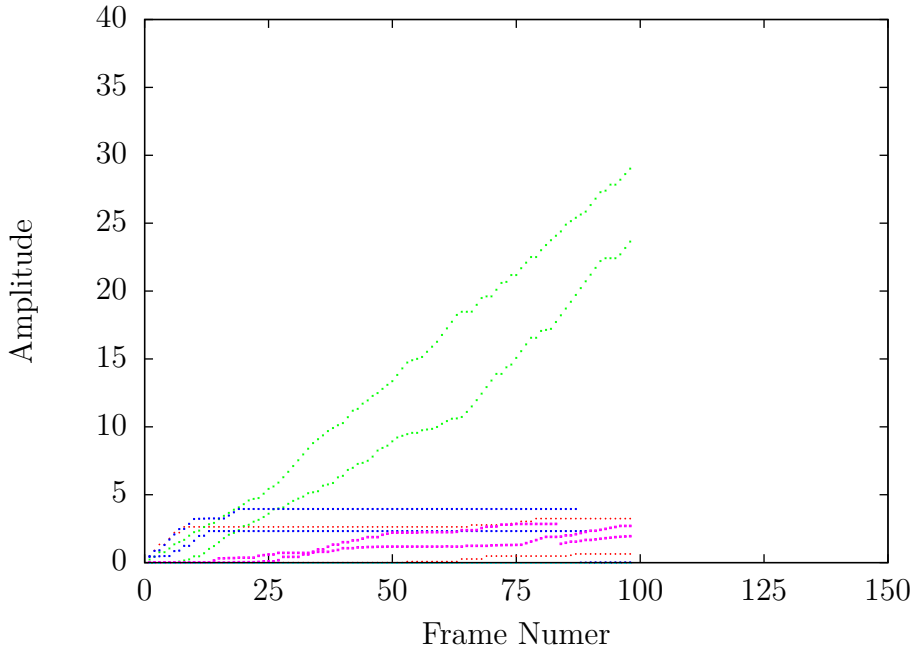


Figure 6.22: Mean shift peak amplitudes for 'sit' (green) using geometric descriptor with clustering.

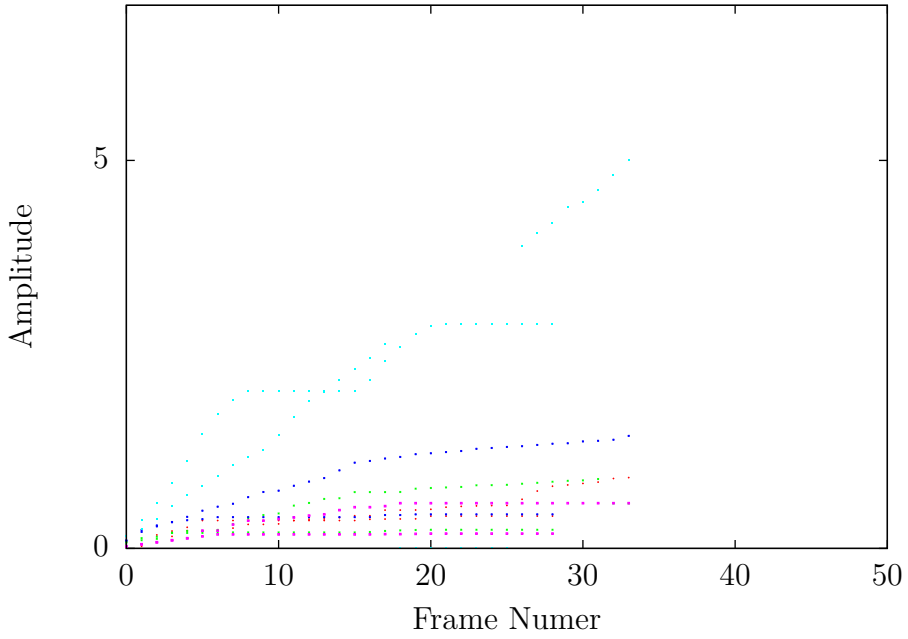


Figure 6.23: Mean shift peak amplitudes for 'wave' (turquoise) using joint-angle descriptor without clustering.

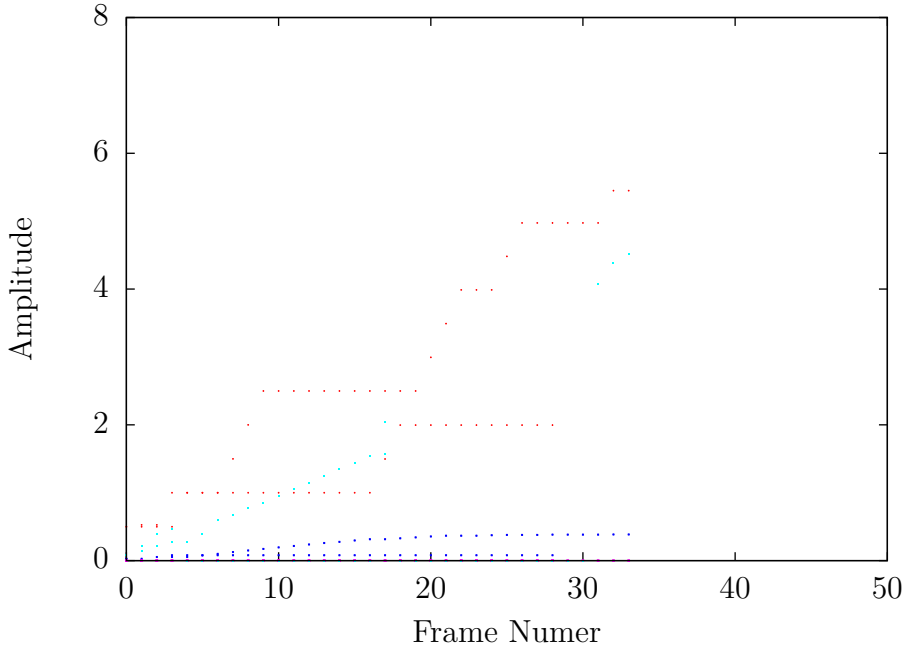


Figure 6.24: Mean shift peak amplitudes for 'wave' (turquoise) using joint-angle descriptor with clustering.

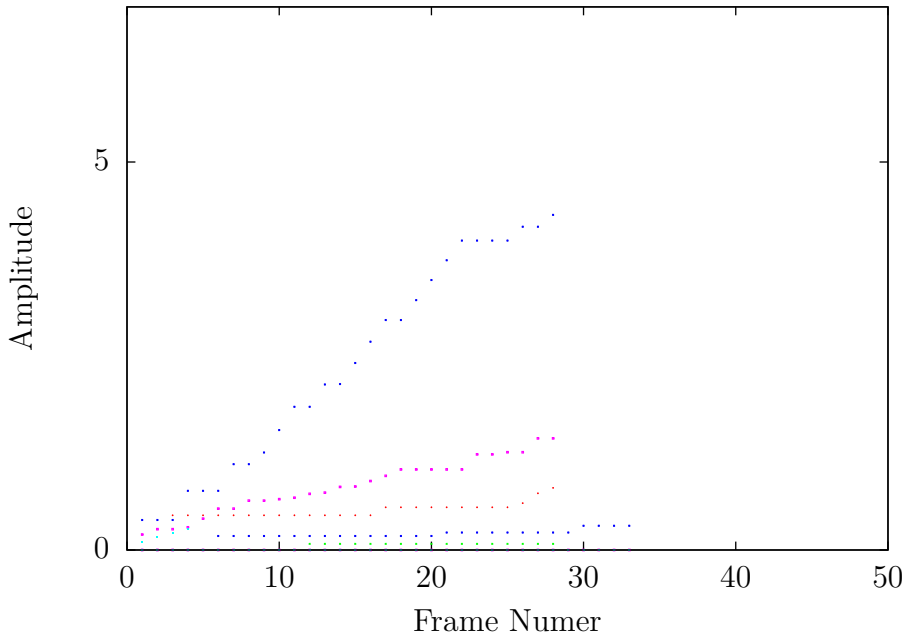


Figure 6.25: Mean shift peak amplitudes for 'wave' (turquoise) using geometric descriptor with clustering.

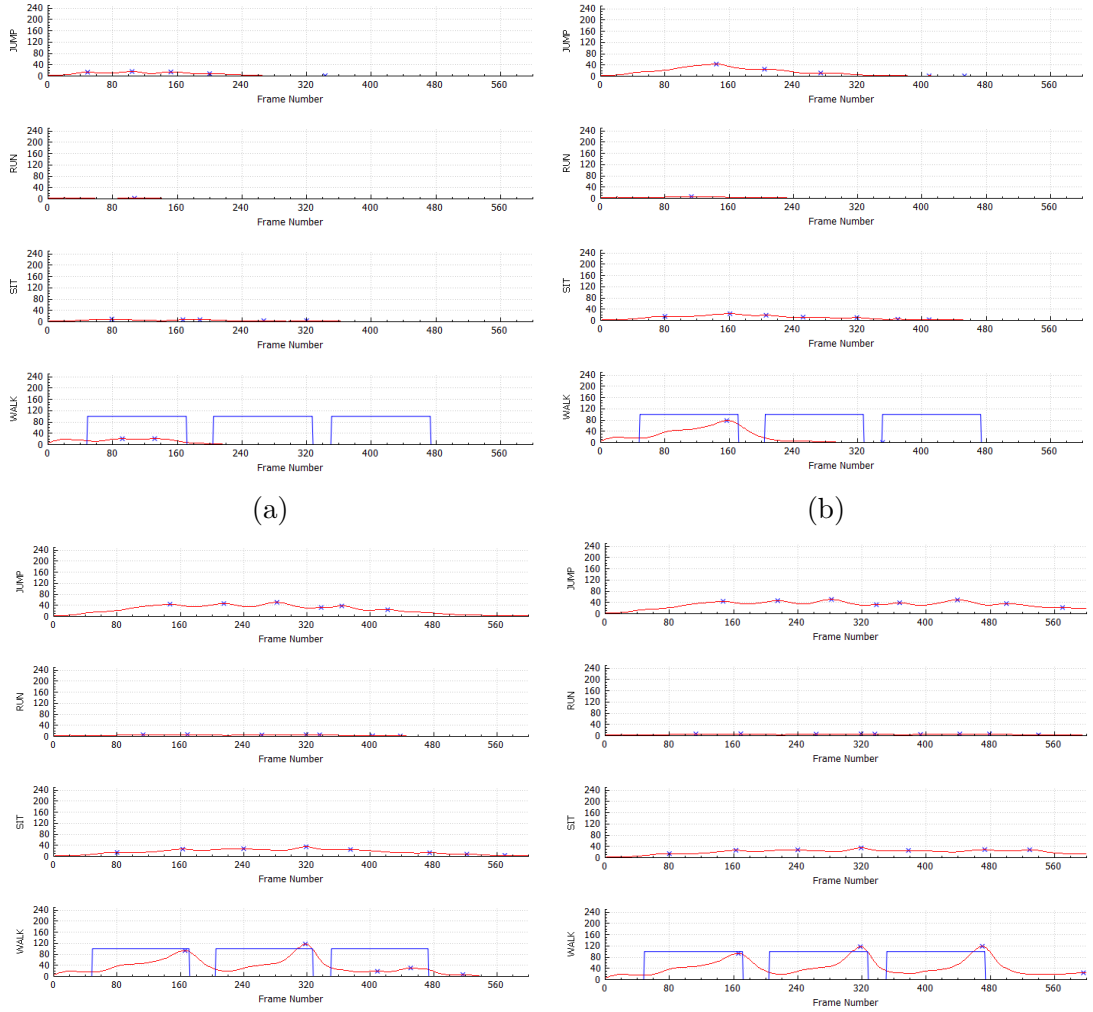


Figure 6.26: Mean shift amplitude for a slow walk action at four different time instants: $t_1 = 80$, $t_2 = 160$, $t_3 = 340$, and $t_4 = 480$. Average action cycle $Ca_1 \approx 160$.

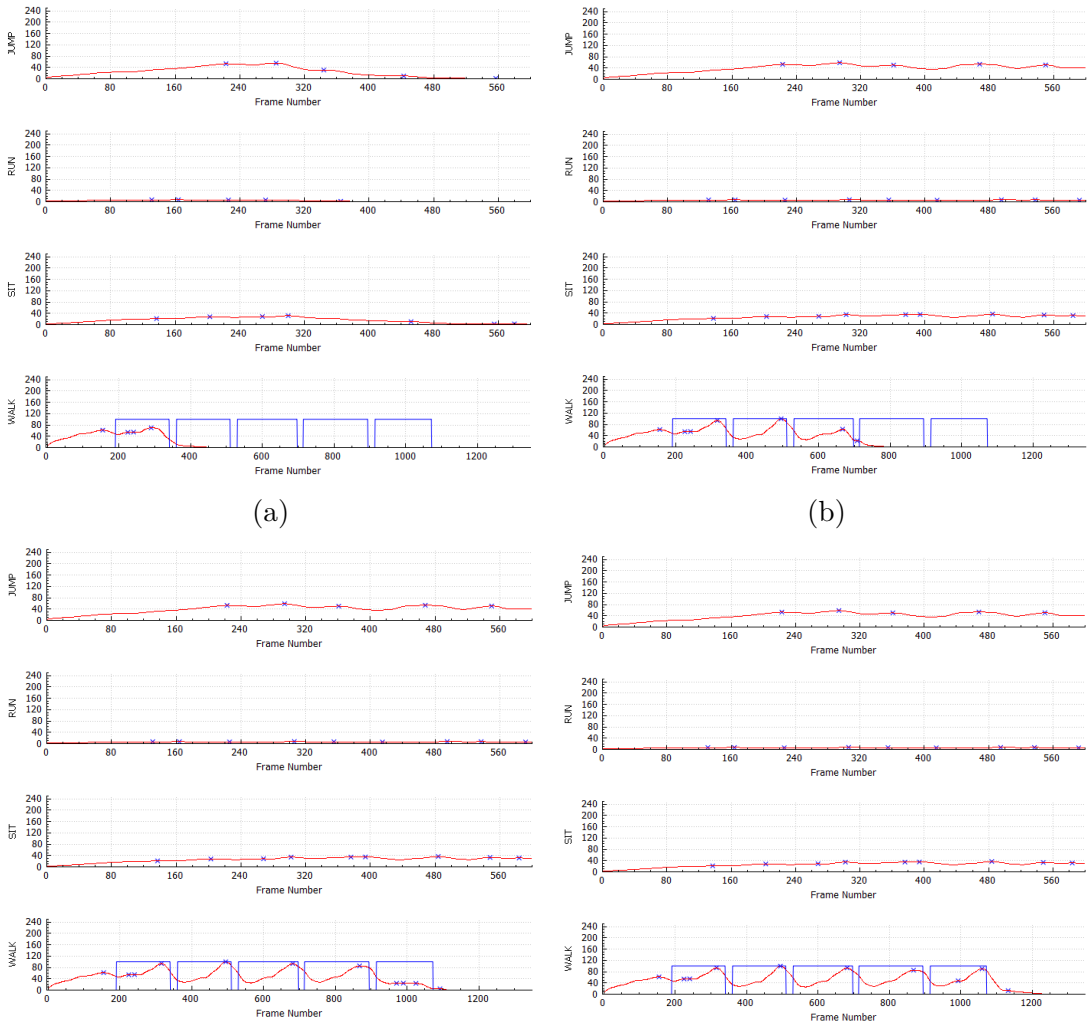


Figure 6.27: Mean shift amplitude for a slow walk action at four different time instants given in number of frames: $t_1 = 250$, $t_2 = 600$, $t_3 = 800$, and $t_4 = 1100$. Average action cycle $Ca_1 \approx 160$ frames.

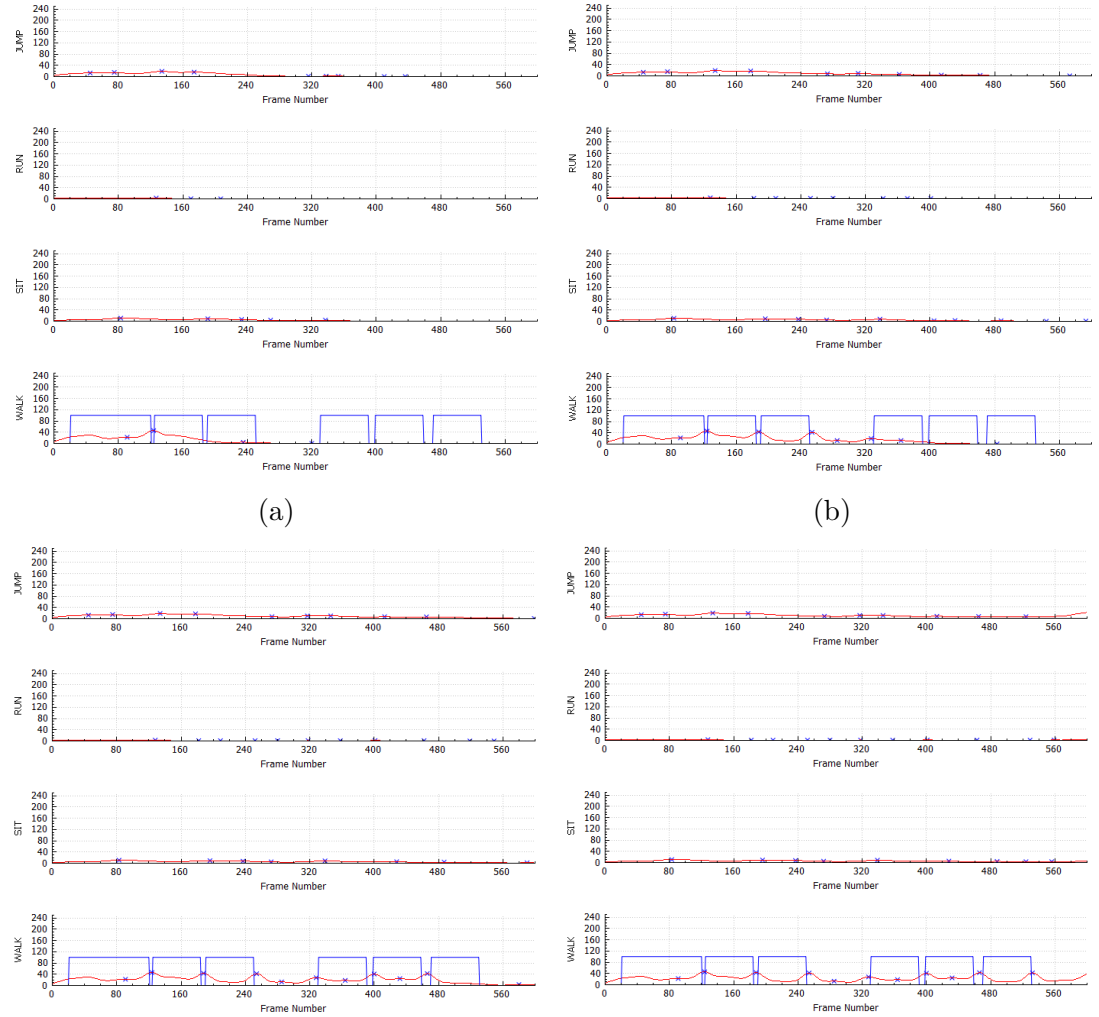


Figure 6.28: Mean shift amplitude for a slow walk action at four different time instants: $t_1 = 100$, $t_2 = 240$, $t_3 = 480$, and $t_4 = 560$. Average action cycle $Ca_1 \approx 160$.

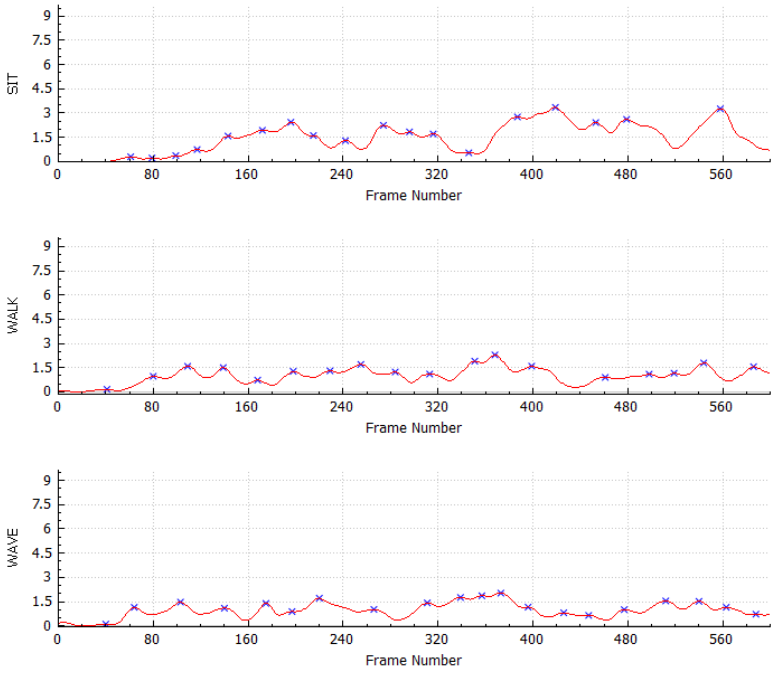
6.5 Robustness

In order to investigate the tolerance of the framework to noisy data, a robustness test for each descriptor was performed. Initially, different video sequences were recorded with the Microsoft Kinect depth sensor and used to create a training dictionary. Subsequently, in Figure 6.29 a video sequence containing two sit actions was recorded and used as the input. In (a) the joint-angle descriptor and in (b) the geometric descriptor were applied. In that case – surprisingly compared to the used data above – the joint-angle descriptor totally fails but the geometric descriptor provides a valid result. Looking closer to that issue reveals that the data recorded with the Kinect sensor is not as accurate as the data used above. That additionally leads to the conclusion that the geometric descriptor can handle small noise with ease.

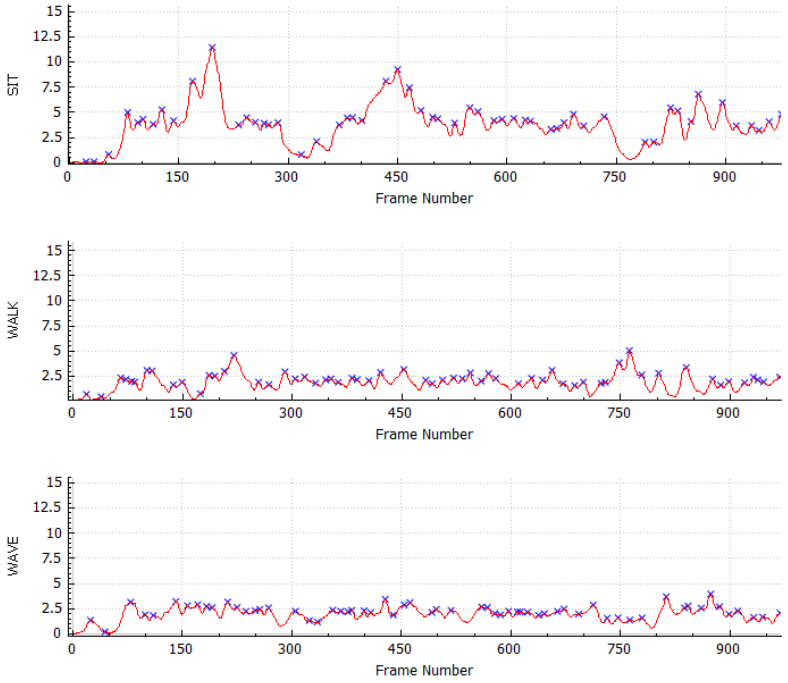
6.6 Real Time Evaluation

As a final experiment, the framework was tested using as input the real-time data streams provided by the Kinect depth sensor. As observed in Figure 6.30, the sensor delivers three different types of data streams. From these, the data stream containing the skeleton coordinates is used to create a pose descriptor per frame.

With an approximate frequency of 30 poses per second, the total time interval required by all the framework stages was short enough to allow for real-time action recognition. That means, the rendering of the skeleton appeared always fluently and the action statements were given with the latency of about one second, due to the evolution of the vote density, compare esp. Figure 5.5. The action recognition was faster when the geometric descriptor was used and not too many samples in the training were included.



(a)



(b)

Figure 6.29: Robustness Experiment. An action recorded with the kinect depth sensor was used as the input sequence. The sequence consisted of two cycles of a sit action finishing at $t = 170$ and $t = 450$. The framework was trained with 'sit', 'wave', and 'walk' actions. (a) and (b) show the results of the mean shift density employing a joint-angle descriptor and a geometric descriptor respectively.

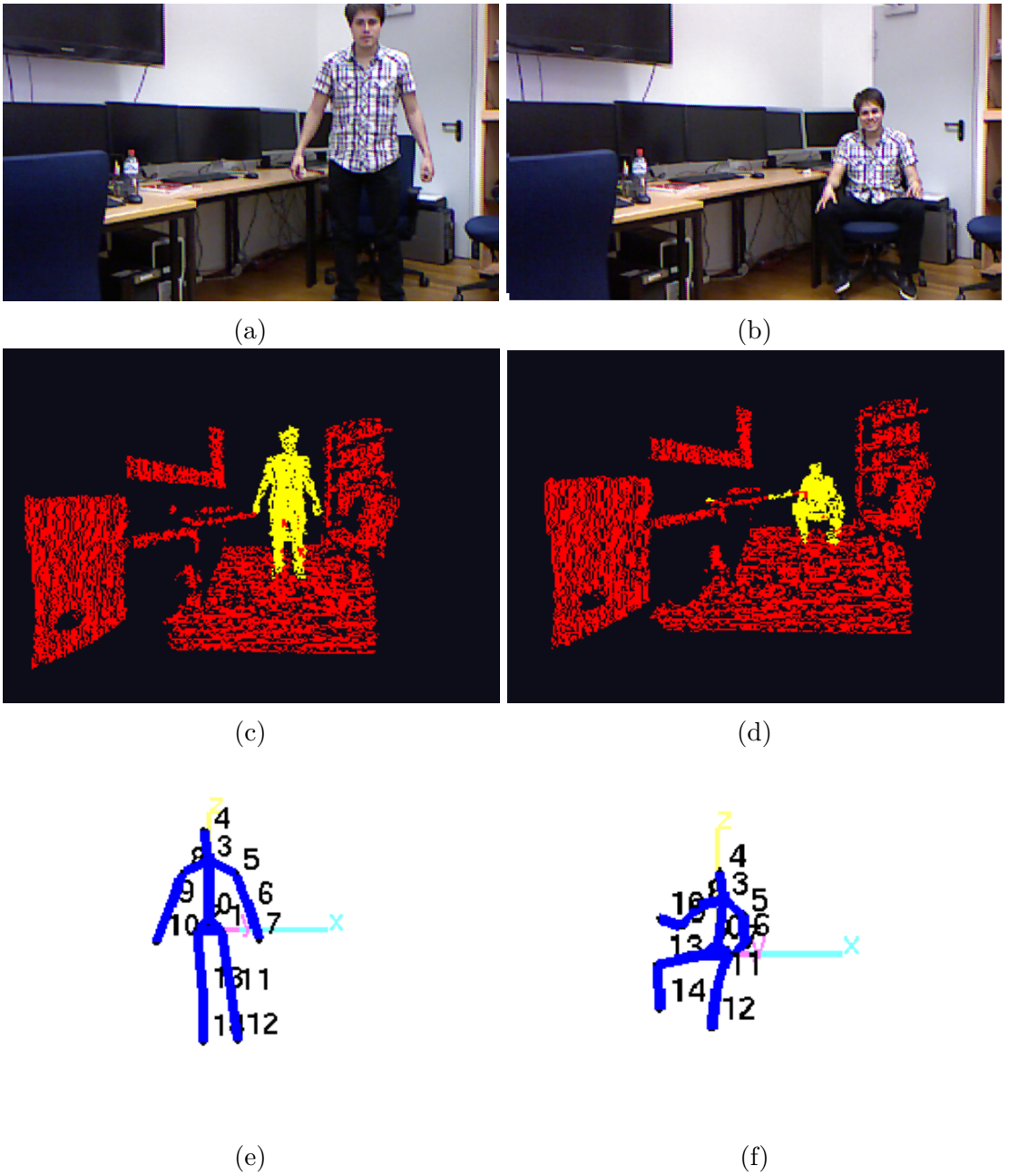


Figure 6.30: Real time data from the Microsoft Kinect sensor. (a,b) color images, (c,d) depth image, and (e,f) the estimated 3D pose of the person.

Chapter 7

Summary

This thesis presented a real-time approach for automatic human action recognition based on the Generalized Hough Transform . At the beginning, Chapter 2 briefly introduced state of the art approaches that have been recently developed for automatic action recognition. This gave a general overview of the main technical challenges found in the action recognition field. Namely, two basic pillars were identified: the selection of the features for representing an action and the respective classification method used for assigning those features into specific action classes.

Chapter 3 described the descriptor and the Implicit Shape Model approach used in the work by (Zepf, 2012) on which the framework developed in this thesis is based. Having given those initial basics, Chapter 4 presented the extensions of the initial framework that were included in this work. Particularly, the geometric descriptor was introduced as an alternative to the joint-angle descriptor. Furthermore, the RNN algorithm for descriptors clustering was explained.

The relevant details of the implementation were given in Chapter 5. Here, the different stages that constitute the framework were explained. These are the data acquisition, descriptor processing, voting, mean shift, and action classification stages.

Conclusively, Chapter 6 presented the different experiments performed on the action recognition framework. Initially, the correctness of the implementation was verified by testing video sequences of actions that were already present in the training data. Afterwards, experiments for detecting single actions were performed using the joint-angle descriptor and the geometric descriptor. From the results of this experiment, it can be concluded that the framework is capable of predicting the end of an action after a few number of frames has been processed with either descriptor, independently of whether the clustering is employed or not. However, the processing of each frame using geometric descriptors was generally faster than using joint-angles descriptors. Subsequently, the discrimination of several actions was

evaluated. In this experiment, the results showed promising results for the majority of actions using three different methods: joint-angle descriptor without clustering, geometric descriptor with clustering, and joint-angle descriptor with clustering. In almost every case, it was possible to correctly identify the action being performed and differentiate it from the other actions in the training set.

Experiments for the time invariance of the framework were also performed. They showed that actions performed at considerably different speeds were also able to be detected. This statement holds true as long as the training data set provided contains at least one action performed at a similar speed as the action under evaluation.

Additional experiments were performed for assessing the robustness of the framework considering real data from Microsoft Kinect sensor in real-time. To close the whole pipeline a classifier is automatically learnet and applied.

The contribution of this thesis is the implementation of the action recognition system in real-time, integration and evaluation of a new geometric descriptor, integration and test of the Microsoft Kinect sensor for using own data, and closing the pipeline with an automatically learnt classifier.

The time-invariance was demonstrated on training data with actions with different speeds. Future work should focus on time normalization of the actions. Another improvement is a descriptor which can be restricted to certain parts of the body allowing an action-specific adaption.

List of Figures

Bibliography

Bay, Herbert, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool

2008 "Speeded-Up Robust Features (SURF)", *Computer Vision and Image Understanding*, 110, 3 (June 2008), pp. 346-359, ISSN: 1077-3142. (Cited on p. 9.)

Benzécri, J.-P.

1982 "Construction d'une classification ascendante hiérarchique par la recherche en chaîne des voisins réciproques." *Les cahiers de l'analyse des données, 7 no. 2, p. 209-218.* (Cited on p. 33.)

Blank, Moshe, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri

2005 "Actions as Space-Time Shapes", in *The Tenth IEEE International Conference on Computer Vision.* Beiging, pp. 1395-1402. (Cited on pp. 7, 8, 10.)

Bobick, Aaron F. and James W. Davis

2001 "The Recognition of Human Movement Using Temporal Templates", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 3, pp. 257-267, ISSN: 0162-8828. (Cited on pp. 6, 10.)

Brauer, Jürgen and Michael Arens

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. (Cited on pp. 3, 16.)

CMU: Carnegie-Mellon MoCap Database. 2003 , http://mocap.cs.cmu.edu. (Cited on p. 38.)

Filipovych, R. and E. Ribeiro

2008 "Learning human motion models from unsegmented videos", in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-7. (Cited on p. 11.)

Gall, J., A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky

2011 "Hough Forests for Object Detection, Tracking, and Action Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33, 11 (Nov. 2011), pp. 2188-2202. (Cited on p. 11.)

Gonzàlez, Jordi

2004 *Human Sequence Evaluation: the Key-frame Approach*, PhD thesis, Universitat Autònoma de Barcelona, Spain. (Cited on pp. 16, 17.)

Harris, Chris and Mike Stephens

1988 "A combined corner and edge detector", in *In Proc. of Fourth Alvey Vision Conference*, pp. 147-151. (Cited on p. 8.)

İkizler Nazlıand Forsyth, David A.

2008 "Searching for Complex Human Activities with No Visual Examples", *Int. J. Comput. Vision*, 80, pp. 337-357, ISSN: 0920-5691. (Cited on p. 11.)

Jhuang, H., T. Serre, L. Wolf, and T. Poggio

2007 "A biologically inspired system for action recognition", in *In International Conference on Computer Vision*, pp. 1-8. (Cited on p. 10.)

Kellokumpu, Vili, Guoying Zhao, and Matti Pietikäinen

2008 "Human activity recognition using a dynamic texture based method", in *British Machine Vision Conference*, vol. 1, pp. 1-10. (Cited on p. 7.)

Laptev, I. and T. Lindeberg

2003 "Space-time interest points", in *Proceedings. Ninth IEEE International Conference on Computer Vision*, 432-439 Vol.1. (Cited on pp. 7-9.)

Laptev, Ivan, Barbara Caputo, and Tony Lindeberg

2007 "Local velocityadapted motion events for spatio-temporal recognition", *Computer Vision and Image Uderstanding*, pp. 207-229. (Cited on p. 9.)

Leibe, Bastian, Ale Leonardis, and Bernt Schiele

2008 "Robust Object Detection with Interleaved Categorization and Segmentation", *International Journal of Computer Vision*, 77 (1-3 2008), pp. 259-289, ISSN: 0920-5691. (Cited on pp. 20-23, 34, 35.)

Moeslund, Thomas B., Adrian Hilton, and Volker Krüger

2006 "A survey of advances in vision-based human motion capture and analysis", *Computer Vision and Image Understanding*, 104, 2-3 (Nov. 2006), pp. 90-126, ISSN: 10773142. (Cited on p. 1.)

Müller, Meinard, Tido Röder, and Michael Clausen

2005 "Efficient content-based retrieval of motion capture data", *ACM Transactions Graphics*, 24 (July 2005), pp. 677-685, ISSN: 0730-0301. (Cited on pp. 13, 29.)

De Rham, C.

1980 "La classification hi rarchique ascendante selon la m thode des voisins r ciproques", *Cahiers de l Analyse des Donn es 2 (5) 135 144*. (Cited on p. 33.)

Schuldt, Christian, Ivan Laptev, and Barbara Caputo

2004 "Recognizing Human Actions: A Local SVM Approach", in *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03*, pp. 32-36, ISBN: 0-7695-2128-2. (Cited on p. 8.)

Wang, Liang and David Suter

2006 "Informative Shape Representations for Human Action Recognition", in *Proceedings of the 18th International Conference on Pattern Recognition - Volume 02*, pp. 1266-1269, ISBN: 0-7695-2521-0. (Cited on p. 6.)

Wang, Ying, Kaiqi Huang, and Tieniu Tan

2007 "Human Activity Recognition Based on R Transform", in *IEEE Conference on Computer Vision and Pattern Recognition, 2007.* Pp. 1-8. (Cited on p. 6.)

Weinland, Daniel, Remi Ronfard, and Edmond Boyer

2006 "Free viewpoint action recognition using motion history volumes", *Computer Vission Image Understanding*, 104, 2 (Nov. 2006), pp. 249-257, ISSN: 1077-3142. (Cited on p. 6.)

Willems, Geert, Tinne Tuytelaars, and Luc Gool

2008 "An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector", in *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, Springer-Verlag, Marseille, France, pp. 650-663, ISBN: 978-3-540-88685-3. (Cited on p. 8.)

Yamato, J., Jun Ohya, and K. Ishii

1992 "Recognizing human action in time-sequential images using hidden Markov model", in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1992*, pp. 379-385. (Cited on p. 11.)

Yao, Angela, Juergen Gall, and Luc Gool

2012 "Coupled Action Recognition and Pose Estimation from Multiple Views", English, *International Journal of Computer Vision*, 100 (1 2012), pp. 16-37, ISSN: 0920-5691. (Cited on pp. 2, 13, 14, 31.)

Yao, Angela, Juergen Gall, and Luc Van Gool

2010 "A Hough transform-based voting framework for action recognition", in *IN: CVPR*. (Cited on pp. 2, 11, 12, 24.)

Zepf, Tobias

2012 *From the Articulated 3D Pose of the Human Body to Basic Action Recognition.* MA thesis, Fakultät f r Informatik. Institut f r Anthropomatik (IFA). (Cited on pp. iii, 3, 20, 24, 29, 73.)

Eidesstattliche Erklärung

Name: Camilo Andres Ramirez Fandiño
Matrikelnummer: 4116907

Hiermit versichere ich eidesstattlich, dass die vorgelegte Arbeit von mir eigenständig und ohne die unzulässige Hilfe Dritter verfasst wurde, auch in Teilen keine Kopie anderer Arbeiten darstellt und die benutzten Hilfsmittel sowie die Literatur vollständig aufgeführt sind.

Ettlingen, den 17. May 2013

Camilo Andres Ramirez Fandiño