Human Action Recognition using Global Point Feature Histograms and Action Shapes

Radu Bogdan Rusu\textsuperscript{1}, Jan Bandouch\textsuperscript{1}, Franziska Meier\textsuperscript{1,2},
Irfan Essa\textsuperscript{2}, and Michael Beetz\textsuperscript{1}

\textsuperscript{1}Intelligent Autonomous Systems Group, Technische Universität München
Boltzmannstr. 3, 85748, Garching bei München, Germany
\{rusu, bandouch, meierfr, beetz\}@cs.tum.edu

\textsuperscript{2}College of Computing, Georgia Institute of Technology
Atlanta, GA, USA
irfan@cc.gatech.edu

Abstract

This article investigates the recognition of human actions from 3D point clouds that encode the motions of people acting in sensor-distributed indoor environments.

Data streams are time-sequences of silhouettes extracted from cameras in the environment. From the 2D silhouette contours we generate space-time streams by continuously aligning and stacking the contours along the time axis as third spatial dimension.

The space-time stream of an observation sequence is segmented into parts corresponding to subactions using a pattern matching technique based on suffix trees and interval scheduling. Then, the segmented space-time shapes are processed by treating the shapes as 3D point clouds and estimating global point feature histograms for them. The resultant models are clustered using statistical analysis, and our experimental results indicate that the presented methods robustly derive different action classes. This holds despite large intra-class variance in the recorded datasets due to performances from different persons at different time intervals.

keywords: action recognition, point cloud, global features, action segmentation

1 Introduction

As robots and humans are to share the same workspaces and cooperate with each others, the robots must be capable of understanding the actions and movements of the humans. This requires in many cases the recognition and interpretation of full body motions. Robots must hand over objects to humans and receive objects from them, and they must approach people differently depending on what they are currently doing. Recognizing an action is not only necessary to understand the current behavior of a
human but it is also a pointer to its future intentions (e.g. recognize the situations where he might need help).

In this paper we present a combination of methods that together form a complete pipeline that tackles all necessary steps from the camera based observation of humans to the recognition of the actions performed by them. These methods do not rely on the estimation of model parameters that might be hard to come by in the case of human full body motions. We furthermore propose the use of methods originally developed for the acquisition of 3D environment models from laser scans to improve the performance of vision based action recognition methods[42].

![Figure 1: Silhouette extracted from a camera image using background subtraction (left); Silhouette extracted from a thermal camera using thresholding (center); Space-time shape generated from a sequence of silhouettes (right).](image)

A number of approaches to action recognition [7, 16, 54] propose the interpretation of space-time shapes as depicted in Figure 1. The problem of action recognition then becomes the problem of classifying such a given shape with respect to a set of a priori learned classes of actions. By treating the space-time shapes as 3D objects, we require techniques for solving the problem in a different domain: that of 3D object recognition. Thus, we propose to apply methods for robustly identifying sets of informative features, which can solve the recognition problem.

An important step that is often neglected is the segmentation of space-time shapes from an ongoing stream of camera observations, i.e. the recognition of start and end frame of an action. Only then is an automated system able to correctly classify current actions that may vary in length and style, and thus are difficult to recognize using e.g. sliding window approaches. We propose methods related to string matching and pattern recognition to solve this subtask, and thus to automatically obtain meaningful action shapes for classification.

Among the main contributions of this paper is the application of a robust feature histogram representation [43, 50] to the problem of action recognition. We assume that the feature histogram representation of space-time shapes of the same actions are sufficiently similar and those of different actions sufficiently different that we can classify the actions reliably based on them. We describe how this method can be implemented and applied to the visual data and empirically evaluate the recognition results. In partic-
ular, we propose a system for action recognition from cameras placed in the environment. Given only silhouette data that is relatively easy to obtain from static camera images, we detect actions by comparing features extracted from a time-sequence of silhouettes to exemplars stored in a training database. Treating the silhouette-sequences as 3D point cloud data, we propose the use of robust point-based geometry techniques which have proven to be reliable for solving numerous problems [43]. We show that such features can be successfully applied also to the action recognition domain.

Another important contribution is the automatic segmentation of the initial aligned time-sequence of silhouettes (space-time stream) into individual consistent action components. We do this by retrieving all possible patterns in the motion data using suffix trees. The combination of patterns that best describes the data stream and thus the start and end frames of likely subactions corresponding to these patterns are then found using a dynamic programming technique known from the weighted interval scheduling problem.

Figure 2: Action Recognition pipeline as described in this article.

The remainder of this paper is organized as follows. The next section gives a brief overview on related work. We then address each step of our action recognition pipeline in the subsequent sections, closely following the system overview given in Figure 2. In Section 3 we describe the acquisition and alignment of silhouette data to create space-time streams. We then describe how to segment the start and end frames of subactions from the space-time streams in Section 4. We treat the resulting space-time shapes as 3D point clouds and present several optimizations to these action representations in Section 5. These help to optimally align noisy silhouette data, to minimize redundant information for fast processing, and to implicitly account for variance in the execution speed of an action. Section 6 then describes our implementation for computing robust feature histograms for such action shapes. In Section 7 we present our optimizations for the computation of global point feature histograms (GPFH). Finally, we discuss experimental results in Section 8 and conclude in Section 9.

2 Related Work

Action recognition or interpretation of human motions aims at assigning higher-level semantic labels to human motion patterns. This facilitates the recognition of ongoing activities and intentions when combined with context information. Video-based action recognition can be roughly divided into two research directions.
The first class of methods is model-based, where parameters of a representative model are determined in a first step, and then used to perform action classification in phase space. The most commonly used parameters for action classification are joint angles or joint positions of articulated human models [46, 27]. Such parameters are invariant to translations, scale and rotations, and form a rich and comprehensive representation of human movements that is well suited for recognition, providing high recognition rates. However, the model parameters are difficult to extract by unintrusive means, so most of the methods rely on commercial marker-based motion capture systems for retrieving the joint parameters. Markerless motion capture systems [12, 41, 23, 20] provide an alternative, but are yet computationally demanding and lack robustness and general applicability.

The second class of methods provides a direct mapping of image cues to action classes without the intermediate estimation of model parameters. Such methods can further be divided into global/holistic approaches and patch-based approaches. Holistic approaches consider image information as a whole, e.g. by using silhouettes [8], edge images [51] or optical flow fields [13]. Bobick et al. [8] have introduced motion-history-images as a 2.5D representation of actions, where silhouettes are overlayed on a single image with brightness corresponding to the distance in time. Blank et al. [7, 16] create spacetime-shapes from extracted silhouettes by using time as the third dimension. A similar technique was independently proposed by Yilmaz and Shah [54] as spatiotemporal volumes. Weinland et al. [53] create view-independent representations of human motions by fusing silhouettes from multiple cameras to obtain visual hulls, that are then augmented with temporal information to form 3.5D motion-history-volumes (extending motion-history-images [8] by another dimension). Both silhouettes [2, 12, 3] and their multi-view extensions as visual hulls [10, 11, 23, 36] are also frequently and successfully used in markerless human motion tracking or pose detection approaches, which showcases their usefulness in a research area closely related to human action recognition. In contrast to holistic approaches, patch-based approaches extract only selected salient features and group them for classification, which helps to overcome problems in the case of partial occlusions [37].

Besides the feature extraction step, classification plays an important role. Here, an action label or a probability distribution over multiple labels is associated with the observed features. Nearest neighbor classification is probably the simplest method for selecting class membership. Sometimes, dynamic time warping is used to achieve invariance with respect to the speed of the motion [49]. Better recognition rates can be achieved by using discriminative classifiers such as support vector machines [45]. Another class of methods is modeling an action in state space as a set of states and transitions. Hidden markov models e.g. can be used for recognition as well as for generation of actions [25, 30, 38, 9]. Closely related are probabilistic grammars [32]. Ogale et al. [35] use context-free grammars to describe actions as a sequence of keyframe silhouettes. Such generative models are usually learned separately for all action classes. Sminchisescu et al. [47] use conditional random fields as discriminative graphical models to achieve better separation of action classes.

In our work we use a holistic approach related to Blank et al. [7, 16] and Yilmaz and Shah [54] to create 3D space-time representations of actions from 2D silhouette contours. Several features have
been proposed for classifying 2D contours, e.g. Fourier descriptors [39] and shape contexts [6, 33].

Three-dimensional extensions of these features have also been proposed for the classification of 3D shapes such as the space-time shapes we are interested in [53, 17]. The 3D object recognition community has developed different methods for computing multi-value features which describe complete models for classification: curvature based histograms [18], spin image signatures [22], or surflet-pair-relation histograms [50]. All of them are based on the local estimation of surface normals and curvatures and describe the relationships between them by binning similar values into a global histogram. A high number of histograms per object is required by [18], but the method can cope with up to 20% occlusions. The 4D geometrical features used in [50] and the spin image signatures in [22, 29] need a single histogram and achieve recognition rates over 90% with synthetic and CAD model datasets, and over 80% with added uniformly distributed noise levels below 1% [50]. All of the above show promising results, but since they have only been tested against synthetic range images, it’s still unclear how they perform when used on noisier real-world datasets.

The task of temporally segmenting a continuous video stream into action components has received little attention so far, and most approaches on action recognition still rely on manually presegmented actions. Frade et al. [28] propose a two-step method for temporally segmenting facial behaviors comprised of clustering shape and appearance features using spectral graphs and then grouping these clusters into a set of dynamical facial gestures. Zelnik and Irani [55] design a simple distance measure for the event-based analysis of video sequences of different lengths, that is used to obtain a temporal segmentation into event-consistent subsequences. Barbić et al. [4] propose three methods to perform the temporal segmentation task on motion capture data. The first two approaches are based on Principal Component Analysis (PCA), where sudden changes in the intrinsic dimensionality of the PCA model are detected, and motion sequences are decomposed into action components based on these changes. Aligned Cluster Analysis (ACA) for the task of temporal segmentation of human motion is proposed by Zhou et al. [57]. On that account the temporal segmentation problem is formulated as a clustering problem, and standard clustering algorithms are extended such that they achieve temporal invariance. The solution to the segmentation problem is posed as an energy minimization problem. Weinland et al. [52] split sequences into action components at points of minimal motion energy. Kulic et al. [27] present an integrated approach for learning and grouping of motion primitives for both recognition and generation using adaptive hidden markov chains, but they assume that motion primitives are presegmented.

A more detailed survey on action recognition is provided by Krüger et al. [26].

3 Space-Time Streams from Silhouettes

We take a holistic approach to action recognition, where the goal is to find a direct mapping from the observed image information to action classes without the estimation of intermediate model parameters.

One of the most informative visual features when perceiving humans is their silhouette, which provides a rich source of information regarding the pose of a human. People are capable of correctly
guessing certain poses by simply looking at the silhouette shapes. However, actions are hard to recognize by looking at only a single silhouette, as the temporal alignment of poses becomes an important cue. We therefore create a stack of 2D silhouette contours by treating time as the third dimension and shifting each incoming silhouette along that direction as time progresses (Figure 1 right). Such a strategy has been independently proposed by Blank et al. [7] and Yilmaz and Shah [54]. As we are dealing with continuous data, and the start and end position of an action is yet unknown, we will refer to this structure as space-time stream.

The extraction of foreground regions from static camera images is a well-studied problem in computer vision [48, 15, 24]. Foreground-background segmentation is often achieved using statistical models of the background, that are trained on static background images, and that are updated over time to account for changing lighting conditions. The models are usually learned independently per pixel, making these algorithms easy to implement and computationally tractable. Special variations have been proposed to account for shadows, that are often erroneously attributed to foreground regions [40].

We use the method proposed by Kim et al. [24] for foreground extraction. It can be interpreted as a fast approximation to a multimodal mixture of Gaussians model of the background under the assumption that the largest eigenvector of each Gaussian points along the luminance direction. After segmenting the foreground, noise is reduced by applying morphological opening and closing operations on the foreground mask. This also joins adjacent regions that have been split (possibly due to noise) to form a single connected component. The silhouette contours are extracted after selecting the biggest connected component in the image and filling all holes in the component (Figure 1 left).

To obtain a good classification accuracy later on as well as for scale invariance, the sizes of the silhouette contours are normalized with respect to the maximum distance between points along the vertical direction (i.e., the shape’s height). Furthermore, contours are aligned by their centers of gravity before stacking them in the third dimension. Another optimization is to filter frames with no significant change in appearance as described in more detail in Section 5. This way, an invariance regarding the speed of an action is achieved, and the computational load is reduced.

In this work we make the following assumptions to assure a straightforward segmentation of the silhouette contours. First, we assume that only one person is visible in the camera images, so that a single contour is easily extractable by the aforementioned methods. More sophisticated techniques (e.g., [14]) would be needed to distinguish between several persons, which is out of the scope of this article. Second, we assume that the camera used for acquisition is statically placed so that we are able to train a model of the background. A method that is capable to extract silhouettes from moving cameras has been proposed by Zhang et al. [56]. And last, we assume that the appearance of the humans differs sufficiently from the background. This is difficult to assure in practical scenarios, as people wearing a white shirt in front of a white wall would not be segmented correctly. To overcome the last restriction, we propose to use thermal cameras, where a segmentation of humans is straightforward and independent of human appearance. Due to the large differences between human body heat and the rest of the objects in thermal space, a simple thresholding operation is sufficient to obtain a foreground
mask of the human person (Figure 1 center), followed by the contour extraction as described above.

All experiments in this article have however be conducted using standard cameras, which is a sufficient setup in most situations.

It should be noted that space-time streams generated from 2D silhouettes are view-dependent, and sometimes miss relevant information due to self-occlusions of body parts. To overcome this limitation in practical applications, it might be necessary to use multiple cameras to capture the motions from several viewpoints. The best perspective could then be selected online by detecting viewpoints with the largest variation of silhouette shapes.

4 Segmenting Space-Time Streams

A space-time stream as generated in the last section encodes information about the motions performed by a person. However, there is no information which parts of the motion correspond to a single action (such as picking an object), and where the transitions between actions are (such as from picking an object to walking). It is also possible that no action is currently being performed (standing still or no one present). This makes it difficult to classify individual actions, as the position and duration of the action in the stream is unknown. A further difficulty arises when actions have seamless transitions between each other, so that it is difficult to tell exactly when one action ends and the other starts.

Figure 3: Overview of the proposed segmentation process.

In this section we propose a method that automatically segments the sequence of extracted silhouettes that form a space-time stream. The task is stated as the problem of finding a set of patterns that optimally cover the sequence without being too short or too long. A pattern in this context is a sequence of poses, where similar poses get assigned the same symbols. Likewise, the whole sequence is encoded
as a string of symbols, so that a pattern is just a subsequence thereof. Figure 3 illustrates the proposed segmentation process. In the feature selection step, a vector of features encoding the shape of the human silhouette is extracted for each frame of the sequence. These vectors are then clustered, so that similar shapes that are likely to correspond to similar human poses get the same symbol assignment, and a sequence of these symbols is generated. In the pattern extraction step all possible patterns are extracted from the sequence by building a suffix tree that implicitly contains all occurring patterns including their start and end frame in the sequence. An optimal selection of patterns and thus segmentation of the sequence is then obtained in the pattern selection step, where we use dynamic programming to find the sequence of patterns that best describe the data given a yet to be described value function. The start and end frames of these patterns are then used to create the space-time shapes for the final action classification as stated in Section 5ff.

**Feature Selection:** The goal of this first step is to obtain one descriptive feature vector for each extracted silhouette. For this, we have chosen to utilize the Hu invariant moments [21] of the silhouette contours. These are computed from the normalized centralized moments up to order three and are invariant with respect to translations, scale and rotations. Thus for each frame, a 7-dimensional vector comprising of the 7 Hu invariant moments is created to encode the silhouette shape. To ensure good results in the clustering step that comes next, each of the Hu moments is additionally discretized into 5 equally sized intervals.

It should be noted that although we achieved good results using the Hu moments, it has been shown by Poppe and Poel [39] that Fourier descriptors or 2D shape contexts [6] can be more robust in the presence of noisy observations.

**Clustering:** This phase aims at clustering the feature vectors, so that similar silhouette shapes that likely correspond to similar poses are grouped together.

Because of it’s simplicity, and because it performs sufficiently well for our task, we chose the basic K-means algorithm over other, more sophisticated (and complex) clustering techniques. Let $K$ denote the number of clusters that are to be formed. The cluster centers $c_i$, with $i = 1, ..., K$, are initialized randomly and the Euclidean distance serves as the distance function within the K-means procedure. Note, that the K-means algorithm has the property that the quality of the final solution strongly depends on the initial set of cluster centers. Thus, the solution to which one complete run of K-means converges is not guaranteed to be optimal. To avoid this problem the K-means algorithm is run several times with randomly chosen cluster centers, thus several clustering results are obtained. Among these results, the one with the smallest error is returned.

As a result of this process, each feature vector has been assigned to one cluster center $c_i$. Hence, each vector can now be mapped onto one symbol, with the constraint that all vectors belonging to the same cluster center are represented by the same symbol. Therefore we have achieved the goal of this phase: A string that describes the motion data. To achieve invariance with respect to the speed of
motions, we remove all duplicates from the resulting string. This way, the string only represents the order of transitions between different pose states, without enforcing temporal equality.

A major drawback of the K-means technique is the necessity of knowing the parameter $K$ - the number of clusters to be found - beforehand. To get a feel about a good choice for $K$ we ran the whole segmentation process on several training sequences with $K \in \{6, ..., 29\}$ keeping the rest of the segmentation pipeline the same for each test run. Thus, the only parameter influencing the final segmentation result was $K$. The segmentation results were manually inspected for oversegmentation or undersegmentation. We found that with a $K$ of 8 or 10 the best results were achieved on the training sequences. Furthermore, we noticed that $K$ is not proportional to the number of actions within a sequence but rather depends on the type of actions performed. This indicates that for a given data set, the $K$ could be learned on a labeled training set. For our purposes, we proceeded with a $K$ of 10 as it seemed to give the best segmentation of in most of the test cases.

![Figure 4: Pattern Extraction Overview](image)

**Pattern Extraction:** Given the string representation of the observed motion, we now describe how to extract patterns within that sequence. Consider e.g. the following sample sequence:

$$s = '232324545'.$$

Note that the subsequence '3' only appears in the context of subsequence '32'. In other words, '3' does not present any additional structural information given the pattern '32'. Thus, for the purposes of this work a subsequence is regarded as **pattern** if it encodes unique structural information within a given cluster sequence. To this end, we propose the usage of suffix tree $T$ to facilitate the extraction of patterns. Through this, the pattern extraction phase consists of the following two steps:

1. Building the suffix tree for a given cluster sequence.
2. Traversing the suffix tree to obtain patterns.

Let $s$ denote the cluster sequence created in the previous steps. With the help of McCreight’s algorithm [31], the suffix tree $T$ is built in time linear to $\|s\|$. The suffix tree for the sample cluster sequence is visualized in Figure 4. Starting from the root node, every pattern in $s$ can be generated
by traversing $T$. Thus, the set of all patterns in $s$ can be obtained by traversing suffix tree $T$ using depth-first search. While traversing suffix tree $T$ to extract all patterns, we also keep track of the start and end position of each pattern, since this data is needed for the final segmentation of the motion sequence.

Figure 5: (a) Information obtained in the pattern extraction step. (b) Illustration of the Pattern Selection phase.

Pattern Selection: Figure 5a shows the information obtained so far. Now the phase of pattern selection utilizes this information to actually segment the full sequence by choosing some of these patterns that will represent the segments. Within this work the pattern selection problem is stated as Weighted Interval Scheduling Problem. There, a list of tasks given as a set of time intervals associated with weights, needs to be scheduled such as to achieve the highest value possible, or in other words an optimal distribution of tasks.

More formally, there are $n$ intervals labeled $1, \ldots, n$ and each task $i$ can be represented by its start time $s_i$ and its finish time $f_i$. Also, each task $i$ has a value $v_i > 0$ associated with it. Naturally, it holds that $s_i < f_i$ for all $i$. Furthermore, two intervals $i$ and $j$ are compatible if they do not overlap. Thus, if intervals $i$ and $j$ are compatible, either interval $i$ ends before interval $j$ starts, $f_i < s_j$ or the other way around. The goal is to select a subset $S \subseteq 1, \ldots, n$ of mutually compatible intervals, such that the total value of all intervals in the subset $S$, $\sum_{i \in S}$, is maximized.

Before attempting to find a optimal solution for a given instance of the Weighted Interval Scheduling Problem, the intervals need to be sorted after their finish time in ascending order: $f_1 \leq f_2 \leq \ldots \leq f_n$. Having established this order, we are now interested in the following: For each interval $j$ we want to find the largest index $i$ is such that intervals $i$ and $j$ do not overlap. Thus, each interval $i$ is additionally associated with a second value $p_i$, with $p_i$ being the leftmost interval that ends before interval $j$ begins.

One can now find a solution for the Weighted Interval Scheduling by using Dynamic Programming. Let $S(i)$ denote the maximum value of any set of compatible intervals, all of which finish by $f_i$, the finish time of job $i$. We can make the following observation about the optimal solution for $S(i)$: Either interval $i$ is used or it is not. Let’s look at both cases in more detail:
Case 1 Interval \( i \) is not used: We have a maximum value set of compatible intervals on intervals 1, ..., \( i - 1 \), which is \( S(i - 1) \) by definition. Since interval \( i \) is not used the value of \( S(i) \) does not increase, so \( S(i) = S(i - 1) \).

Case 2 Interval \( i \) is used: Since all intervals from \( p_i + 1 \) through \( i - 1 \) overlap with job \( i \). Hence all other intervals selected for \( S(i) \) are drawn among the intervals from 1 through \( p_i \). This also means, that if interval \( i \) is removed from the optimal solution we obtain an optimal solution on intervals 1, ..., \( p_i \), which is \( S(p_i) \) by definition. Since interval \( i \) is used, the value of \( S(i) \) increases by the value of \( v_i \), so \( S(i) = S(p_i) + v_i \).

Putting these two cases together, one can come up with the following equation for \( S(i) \):

\[
S(i) = \begin{cases} 
S(i - 1) & \text{if } S(i - 1) > S(p_i) + v_i \\
S(p_i) + v_i & \text{if } S(p_i) + v_i > S(i - 1)
\end{cases}
\]

This equation can be transformed further into

\[
S(i) = \max(S(i - 1), S(p_i) + v_i),
\]

which is the main component of the dynamic programming solution to our problem. We can now iteratively compute the entries of \( S \), by initializing \( S \) with \( S(0) = 0 \) and then solve the Equation 4 for \( i = 1 \ldots n \). The value of the optimal solution can then be extracted from \( S(n) \). The pseudo code for the complete algorithm is given in Algorithm 1.

<table>
<thead>
<tr>
<th>Algorithm 1: Compute-Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>input</strong> : n intervals, each interval ( i ) is associated with ( f_i ), ( s_i ), ( v_i ) and ( p_i )</td>
</tr>
<tr>
<td><strong>output</strong> : Value of optimal solution</td>
</tr>
<tr>
<td><strong>begin</strong></td>
</tr>
<tr>
<td>( S(0) = 0 )</td>
</tr>
<tr>
<td>for ( i \leftarrow 1 ) to ( n ) do</td>
</tr>
<tr>
<td>( S(i) = \max(S(i - 1), S(p_i) + v_i) )</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

Note, that so far we have only computed the maximum value of a optimal solution but do not know which intervals were actually chosen. To obtain this information one could keep track of the chosen intervals while computing the maximum value.

Transferring this definition to our problem, we state the following:

- The patterns constitute the intervals, given their start and end frames.
- Each pattern \( pat_i \) is assigned a value \( v_i \) that depends on the patterns length.

In our work we define the value \( v_i \) for pattern \( pat_i \) as follows, although alternative definitions are possible:
\[ v_i = \log(length(pat_i)). \]

As a result, longer patterns have a higher value than shorter ones and are more likely to be chosen, while it is still possible for two successive shorter patterns to outdo the longer pattern with their combined values. Thus, it is also ensured that very long patterns are less likely to get selected. The optimal solution for our sample sequence is shown in Figure 5b.

Since a motion segment can comprise of several patterns, one needs to make sure that similar neighboring patterns are merged into one segment. This can be accomplished by simply comparing the cluster consistency of two adjacent patterns. If two adjacent patterns primarily consist of the same clusters, it is likely that they encode the same motion, thus they are merged.

Resulting from this step, we have estimated a collection of start and end frames inside the space-time stream, that segment it into single parts corresponding to subactions. In the next sections we will show how to represent the segmented shapes as space-time shapes and how to use these for classifying actions based on a training set.

5 Optimizing Space-Time Shapes

Having estimated good segmentation points for the space-time streams, we now proceed to the creation of space-time shapes. We represent a space-time shape as a 3D point cloud created from 2D point clouds representing silhouette contours by stacking them in the time dimension. Furthermore, space-time shapes should correspond to a single action, so start and end frame must have been segmented. We introduce the following two optimizations to space-time shapes. First, adjacent contours are aligned using an iterative registration process, to suppress the influence of noise in the silhouette extraction when aligning using the centers of gravity. Second, redundant adjacent contours are removed from the final shape by considering a metric for the shape-change. Therefore, space-time shapes become independent of the speed of an action, and the computational complexity of subsequent steps is reduced along with the size of the point cloud. The representation of actions as 3D point clouds enables us to use sophisticated methods developed in the domain of 3D object recognition.

Our method assumes that during a complete human motion, the resultant silhouettes can be continuously extracted. This does not mean however, that the space-time shapes are susceptible to fail in the presence of occlusions. Because we do not incorporate any information about body parts and poses in our framework, occlusions do not affect the end results as long as they are consistent over the entire motion space for all recorded datasets. For example, if a part of the human body is not visible at a certain location \( L \) by a given camera \( c \), because there is fixed object \( O \) (e.g. a table) occluding the view, then all recorded actions will contain the same occlusions as long as the pose of \( O \) is unchanged with respect to the pose of \( c \). Simply put, the model for this particular space-time shape will be constrained on the presence of the occlusion for future model validations.

To formulate the space-time shape computational model, we introduce the following notations:
• \( p_i \) is a 3D point having \( \{x_i, y_i, z_i\} \) geometric coordinates;
• \( \langle p_i, p_j \rangle \) describes the dot product between \( p_i \) and \( p_j \);
• \( n_i \) is a surface normal estimate at point \( p_i \) having a \( \{nx_i, ny_i, nz_i\} \) direction;
• \( \mathcal{P}_n = \{p_1, p_2, \cdots\} \) is a set of nD points (also represented by \( \mathcal{P} \) for simplicity);\(^1\)
• \( f_t \) is a silhouette data frame acquired at time \( t \) represented by a 2D point cloud \( \mathcal{P}_2 \);
• \( \mathcal{P}^k \) is the set of points \( p_j \), \( (j \leq k) \), located in the \( k \)-neighborhood of a query point \( p_i \);
• \( \| \cdot \|_x \) is the \( L_x \) norm (e.g. \( \| \cdot \|_1 \) is the Manhattan or \( L_1 \) norm, \( \| \cdot \|_2 \) is the Euclidean or \( L_2 \) norm).

For every frame \( f_t \) inside the current segment, our method takes an extracted silhouette and converts it to a 2D \( \{x, y\} \) point cloud \( \mathcal{P}_2 \), annotated with a frame number \( t \). The final goal is to acquire a stack of silhouettes, build 3D point clouds for every segmented action with \( z \) as the time axis, then exploit the existing geometry in the data to create a Global Point Feature Histogram (see Section 6) for each motion that we want to classify. Figure 6 presents an example of such a silhouette stack.

Figure 6: Examples of extracted silhouettes for human poses (left) and motions (right).

**Alignment:** Due to small errors and noise in the segmentation process, the resultant silhouettes from two consecutive frames do not always perfectly overlap by aligning them solely using their gravity centers. This results in a small displacement of the 2D point clouds from each other, and can potentially propagate further errors in the space-time shape model creation.

To solve the displacement problem and to make the algorithm more robust with respect to imperfect segmentation, we introduce an iterative registration loop for every two consecutive frames, meant to minimize the Euclidean distance metric error between them. In particular, for every frame pair \( \langle f_t, f_{t+1} \rangle \), with \( f_t \) having \( p \in \mathcal{P} \) and \( f_{t+1} \) respectively \( q \in \mathcal{Q} \) data points, we solve in a 2D space for:

\[
\min \sum_i \| p_i, q_i \|_2
\]  

\(^1\)Two different point clouds will be referred as \( \mathcal{P}_n \) and \( \mathcal{Q}_n \).
The solution of Equation 1 is efficiently computed using a Levenberg-Marquardt non-linear optimization method, and the algorithm usually requires only 1-3 iterations to converge to the final solution. The results before and after the registration step are shown in Figure 7.

![Figure 7: Example of two raw consecutive frames before (left) and after (right) fine registration.](image)

After registration, every new segmented frame will be stacked along the \( z \) axis to the current 3D point cloud as shown in Figure 8.

![Figure 8: A 3D point cloud representing a waving human motion comprised of several 2D stacked frames with \( z \) as time.](image)

**Minimization:** Because of the stacking process however, double frames can appear, mostly because the subject did not move fast enough or at all between two consecutive frames. These “double frames” represent redundant data with respect to our problem and create an additional computational burden. Therefore, we proceed to remove them as follows. For each two consecutive frames, the computational steps include:

1. for each point \( p_i \in \mathcal{P} \) belonging to frame \( f_t \), search for its closest corresponding point \( q_i \in \mathcal{Q} \) in frame \( f_{t+1} \);
2. compute the Frobenious norm in distance space of $f_t$ and $f_{t+1}$ as:

$$\|f_t, f_{t+1}\|_F = \sqrt{\sum |q_i - p_i|^2}$$

(2)

3. select $f_{t+1}$ as being different (i.e. unique) than $f_t$, if $\|f_t, f_{t+1}\|_F \leq d_{thresh}$

In our implementation, we search for the nearest $p_i$ point neighbors by employing a fast kd-tree structure [34]. This has proved to work faster than brute force searches for our datasets.

![Figure 9: Dataset before (left) and after (right) the pruning of duplicate frames. The number of points got reduced from 101520 to 53697.](image)

The results after pruning the duplicate frames are shown in Figure 9, where relevant frames are colored in green (and shown again as the resulted sequence in the rightmost part of the image), while the frames to be pruned in black. The computational time decrease obtained by removing irrelevant frames is highly dependent on the input sequence and can vary greatly. In our experiments, we achieved reduction rates from 24.15% up to 91.47%.

In addition, we achieve an implicit invariance regarding the speed of an action, which helps to improve the recognition rates in task-oriented action recognition. Note that sometimes invariance to speed might be undesirable, e.g. when concerned about the mood of a person. In such cases we still propose to use our minimization as it reduces computational complexity without loss of relevant information. Instead, the number of duplicate frames should be taken into account when reasoning about stylistic details of an action.

6 From Space-Time Shapes to Global Point Feature Histograms

Once a set of silhouettes are acquired and a 3D point cloud is built, our goal is to transform the geometric information into a compact and informative representation. This representation needs to be able to efficiently cluster similar motions or gestures together as well as distinguish between different ones.
In our previous work [43, 44], we have successfully adopted the surflet-pair-relation histograms proposed in [50] for the problem of classifying dishware and other kitchen objects from sensed data using Support Vector Machines, as well as computing initial registration solutions for the problem of 3D point cloud registration [44]. We extend the formulation of the Point Feature Histograms from [44], and give insight on a series of optimizations to reduce their computational complexity. In the new framework, a Global Point Feature Histogram (GPFH) is a set of n-D feature vectors describing the underlying object model using a statistical pyramidal approach. Furthermore, entries in the GPFH are weighted differently based on their importance in the space-time shape. We show that our approach works reliably for different levels of details and provide an in-depth analysis of the method’s breaking point.

The input data consists of segmented space-time shape models in the form of \( P_3 \) (3D \( \{x, y, z\} \) point clouds). In the following we describe the computational steps needed for estimating the GPFH.

Since the proposed feature space depends on surface normal estimates at each point \( p \), our method will first attempt to estimate them by performing Principal Component Analysis (PCA) in the neighborhood of \( p \). Thus, for a given radius \( r \), we select a sphere centered at \( p \), and compute the principal directions of \( P^k \), the set of \( k \)-neighbors around \( p \). The eigenvector corresponding to the smallest eigenvalue approximates the surface normal at point \( p \). Due to the fact that there is no way to guarantee a consistent normal orientation from the eigenvalue decomposition, we proceed at re-orienting them as follows\(^2\). In the process of building \( P \), we estimate a set of 2D centroids for each frame \( f \), and attach them to the resultant 3D points in \( P \). After computing the normal \( n_i \) at a given point \( p_i \), we compute an angle \( \theta \) between \( n_i \) and the direction towards the point’s attached centroid. The direction of \( n_i \) will be changed if \( \theta > 90^\circ \), that is \( n_i = -n_i \).

Figure 10 presents a space-time shape and its associated \( P \) point cloud before and after consistently re-orienting normals.

In a general sense, for every pair of points \( p_i \) and \( p_j \) in the shape and their estimated normals \( n_i \) and \( n_j \), we select a source \( p_s \) and a target \( p_t \) point, the source being the one having the smaller angle between the associated normal and the line connecting the two points:

\[
\text{if } (n_i, p_j - p_i) \leq (n_j, p_i - p_j); \text{ then } p_s = p_i, p_t = p_j; \text{ else } p_s = p_j, p_t = p_i
\]

and then define the Darboux frame with the origin in the source point as (see Figure11):

\[
u = n_s, \quad v = (p_t - p_s) \times u, \quad w = u \times v.
\]

To reduce the dimensionality of the data represented by the four selected vectors \( p_i, p_j, n_i, \) and \( n_j \), we proceed by computing a 4D feature value \( f_{v_x} \) with \( x \in \{0, 1, 2, 3\} \) that measures the angle differences between the point normals and the distance vector between them, and place the resulted values into a global histogram.

For each feature \( f_{v_x} \) we define its theoretical definition range: \((f_{v_{x_{\min}}}, f_{v_{x_{\max}}})\), and divide it in \( d \) subdivisions labeled with 0, 1, 2, ..., \( d-1 \). Therefore, for a given feature value \( f_{v_x} \), we can find out the

\(^2\)see [19] for a general algorithm for consistent normal orientation propagation
Figure 10: From top to bottom and left to right: a) space-time shape in dark blue and the computed centroids in red; b) estimated surface curvatures (low curvatures with red, and high curvatures with blue); estimated point normals (only the 100th displayed) c) before and d) after re-orientation.

Subdivision label in which it is located, by computing the integer part operation on its value's proportion in its definition interval. The combination of all four subdivision labels yields an index in the histogram \( i_{\text{hist}} \) in which the point pair falls.

\[
\begin{align*}
fv_0 &= \langle v, n_t \rangle \\
fv_1 &= \langle u, p_t - p_s \rangle / \| p_s, p_t \|_2 \\
fv_2 &= \| p_s, p_t \|_2 \\
fv_3 &= \arctan(\langle w, n_t \rangle, \langle u, n_s \rangle) \\
i_{\text{hist}} &= \sum_{x=0}^{x\leq3} \left\lfloor \frac{fv_x \cdot d}{fv_x_{\text{max}} - fv_x_{\text{min}}} \right\rfloor \cdot d^x
\end{align*}
\]

where the \( \langle \rangle \)-operator denotes the scalar product, and \( \lfloor \rfloor \) the floor function. For each point-pair and its \( i_{\text{hist}} \) index, we increment the histogram value at that index by 1, and at the end, normalize each bin with the total number of point pairs to achieve point density invariance.
Figure 11: The computed Darboux frame (vectors u, v and w) placed at the source point.

Figure 12: Example of a resulted histogram for a 3D space-time shape.

The number of histogram bins that can be formed using these four geometric features is \(d^4\). For example, by dividing the feature definition range in 2 parts (smaller or greater than \((f_{v_{x_{max}}} - f_{v_{x_{min}}})/2\), we obtain a total of \(2^4 = 16\) bins as the total number of combinations between the 4 features.

To find out the minimum and maximum values of each feature, we have to consider that they are a measure of the angles between the points’ normals and the distance vector between them. Because \(f_{v_1}\) and \(f_{v_2}\) are dot products between normalized vectors, they are in fact the cosine of the angles between the 3D vectors, thus their value is between \(\pm 1\), and 0 if they are perpendicular. Similarly, \(f_{v_3}\) is the arctangent of the angle that \(n_t\) forms with \(w\) if projected on the plane defined by \(u = n_t\) and \(w\), so its value is between \(\pm \pi/2\), and 0 if they are parallel. Finally \(f_{v_2}\) is the length of the segment between the two points, which is always positive and for which a maximal value can be set as the diameter of the shape’s bounding box.

Because the number of bins is increasing exponentially with the number \(d\) of feature categories, we have to select a high enough number for capturing detail, but low enough to reduce the algorithm’s
computational complexity. Figure 12 presents a feature histogram example for a given 3D space-time shape.

The effects of different movements in the feature histogram space for two space-time shapes is shown in Figure 13.

The development of Global Point Feature Histograms was motivated by the fact that they provide a high dimensional feature space that is fairly robust to noise and variations in data, such as different persons having different heights. In the next section we present a histogram resolution analysis which motivates the creation of a repertoire of models with different levels of details, in extremely compact formats (from a few ten to a few hundred feature values per model).

While techniques based on Iterative Closest Point implementations give reasonable results for some applications, we discovered that they are unable to guarantee good solutions with the datasets used in our experiments, due to a large number of local minima and in general large variations in the 3D data (e.g., the person’s height can negatively influence the 3D registration/matching results unless a more complicated special model is created that takes body parts into account). In our experiments, such registration algorithms are also highly sensitive with respect to an initial estimate, that is difficult to provide for most of our datasets, and can take a long time to converge to a minimum.

Employing local point features, such as spin images [22], or any other similar 3D features, could potentially lead to results of similar quality when compared to ours, but would render the model computation and matching times more expensive. In addition, the model size might also grow considerably. In general, our experiments lead us to the conclusion that the proposed Global Point Feature Histograms, if parameterized and used efficiently, provide extremely compact yet powerful models for our application.

Figure 13: Two different space-time shapes and their estimated feature histograms.
7 Global Point Feature Histograms Optimizations

The computation of a GPFH involves building a fully interconnected mesh with a $O(N^2)$ computational complexity, based on the relationships of every point $p_i$ and its associated normal $n_i$ with all the other points and normals in $\mathcal{P}$, with $N$ being the total number of points. That is why a straightforward computation of all the point to point relationships in $\mathcal{P}$ can be extremely costly. To speed up the GPFH creation, we propose two different optimizations:

1. select a small subset $\mathcal{P}^m$ of interesting points from $\mathcal{P}$ and perform combinations between them and all the points in $\mathcal{P}$ to build the GPFH, thus bringing down the theoretical complexity to $O(N \cdot M)$ where $M \ll N$;

2. use spatial decomposition techniques to reduce the number of points in $\mathcal{P}$ by averaging and smoothing the point values along all dimensions.

We will show that the two optimizations given above are extremely important for bringing down the computational complexity, and if used correctly can lead to near-realtime performance results.

Figure 14: Depiction of the weighting scheme between two subsequent frames in the sequence. Points belonging to parts of the shape which do not change are marked with red, with large differences being marked with blue. The left part shows a 3D view of the point-to-point correspondences between the frames, while the right side shows the two frames overimposed in 2D.

In general, global descriptors can easily differentiate between shapes which are very different, but they are unable to accentuate smaller differences. This is due to the fact that these differences constitute only a small percentage of the total number of points present in a space-time shape, and the usual consensus in global shape descriptors is to treat all points equal with respect to their impact on the descriptor. Therefore it is recommended to use these methods only for a rough classification, followed by more detailed comparisons through the combination of different techniques or the usage of local shape descriptors [1]. This however leads to an increased algorithmic complexity and requires additional time for the final descriptor computations.
In our proposed framework, we alleviate additional computational burdens by providing a simple, but powerful weighting scheme for each point in the shape. The resultant point weights are then used in the GPFH descriptor calculus, either by changing a point’s influence in the histogram or by disregarding it completely. We achieve the latter using the first optimization given above. Its implications are therefore twofold: besides boosting small local changes by weighting the points of interest more than the rest, it also reduces the number of points that are to be considered in the GPFH computations. This contributes to a reduced computational complexity of $O(N \cdot M)$, and therefore a faster overall runtime.

In detail, our method estimates a point weight as follows. For each pair of frames $f_t$ and $f_{t+1}$, we assign the weight as the L2 Euclidean norm $\|p_i, q_i\|_2$, where $p_i$ is a point in $f_t$, and $q_i$ is its closest 2D neighbor in $f_{t+1}$. This means that points which have a large distance to their neighbors in a subsequent frame are more probably to belong to a part of the shape that contains an active movement with respect to the previous frames. Figure 14 presents a snapshot of our weighting scheme between two frames. The assigned weight for each point is displayed in color, with red being a low weight and blue a high weight. A complete weighting interpretation for the space-time shape given in Figure 10 is presented in Figure 15.

![Figure 15: Estimated point weights for the space-time shape presented in Figure 15, here shown from two different 3D positions. The colors are displayed in logarithmic scale, with red being a low weight and blue a high one.](image)

To disregard points with very low weights, we make use of simple statistics. By building a distribution over the weights space, our algorithm simply selects those weights larger than some $\mu_w + \alpha \cdot \sigma_w$, where $\mu_w$ and $\sigma_w$ represent the distribution parameters and $\alpha$ can be chosen empirically.

In a second optimization step, we attempt to reduce the data dimensionality by downsampling the number of points present in the space-time shape. Figure 16 presents the effects of the reduction both in terms of the number of cells (points) left in the shape as well as the total computation time of the GPFH.

The computation of feature histograms for different data scales – and thus different levels of detail – adds the possibility of introducing *pyramidal* results in the system, in the sense that the shape
classification can be performed on various levels from rough to fine. Besides studying the computational aspects however, we need to investigate the breaking point of a certain GPFH at a given scale. This scale thus becomes an important factor in the parameterization of the space-time shape.

A secondary factor is somewhat explicitly implied by the resolution of the GPFH itself. Due to how the feature space is defined, a decision has to be made on how many histogram bins should each of the 4D descriptors be split into.

We have studied the impact on both the above mentioned factors, and concluded that for different histogram resolutions, at different scales, the breaking points are similarly defined. Figures 17, 18 and 19 present the aforementioned analysis results for 3 different scales (low to high resolutions). Different GPFH are displayed on the left side of each figure, each plot holding 10 histograms, one for every level of detail. The number of remaining points in each plot is expressed as a measure of the size of each individual cell with respect to the overall bounding box of the dataset. Thus, a resolution of 5% implies that the size of each cell in the downsampled version of the data is approximately 5% of the entire dataset. On the right hand side, the plots describe the Kullback Leibler distances to the least downsampled (i.e. highest resolution) histogram in the dataset, 1%. The distance is defined as:

$$\text{KL divergence} = \sum_{i=1}^{16} (p_i^f - \mu_i) \cdot \ln \frac{p_i^f}{\mu_i}$$  \hspace{1cm} (4)

The values presented are correlated with the previous plots shown in Figure 16.

Using the plots on the right side of Figures 17, 18, and 19, we can estimate the breaking point of each combination of scale and resolution. Values within some user-set tolerance are exemplified with the blue color, and values outside with red.
Global Feature Histograms for different resolutions

Figure 17: Analysis over the resolution and scale factors for the computation of a GPFH for a 81D scale. Top: the computed histograms for 10 different resolutions. Bottom: the KL distance of each GPFH to the highest level of resolution detail (1% downsampled) from the presented set.

8 Discussions and Experimental Results

We evaluated the performance of our approach by taking several datasets in our distributed-sensing kitchen environment [5]. To allow more variation in the data, our subjects were asked to perform several actions and motions, without knowing explicitly from us the exact instructions on how to do them. Furthermore, each individual action of a subject was performed without the other subjects seeing it, so they could not attempt to replicate the movements exactly. The list of performed actions included: (i) opening and closing a cupboard; (ii) opening and closing a drawer; (iii) opening and closing a vertical-door top cabinet; (iv) picking up one object from a table and moving it to another; (v) opening and closing the oven door; and (vi) unscrewing a bottle and drinking from it (see Figure 20 for examples). Each action was performed several times by a subject, at different time intervals. After recording several experiments and looking at the data, we noticed a high degree of variability between the way the movements were performed, from person to person, but also between the same actions of the same
Figure 18: Analysis over the resolution and scale factors for the computation of a GPFH for a 256D scale. Top: the computed histograms for 10 different resolutions. Bottom: the KL distance of each GPFH to the highest level of resolution detail (1% downsampled) from the presented set.

**Segmentation:** We first evaluate our segmentation approach from Section 4. For this, we have manually analyzed the recorded sequences to find a meaningful ground truth segmentation of the datasets with respect to the actions we are interested in. To account for transitions between different activities, each segmentation point has been marked over 10 frames instead of choosing one single frame as breaking point. These transition phases are marked black in Figure 21, where we present the results. The performance of the proposed algorithm was evaluated based on this human segmentation.

In Figure 21, two characteristic segmentation results are depicted. Note that with few exceptions, all the motion segments in the two characteristic test-sequences have been identified by our approach. In two cases a segment is completely missing (*close drawer* and *close oven* in Zoltan # 3) and in several cases the borders of the segments were misplaced by a few frames. For both error types, intermediate results of the segmentation process show that patterns encoding the correct borders of the motion segments
exist. These patterns were not successfully extracted by our interval scheduling phase. Future work will try to address this issue by improving the values assigning formula for each of the patterns. However, our classification approach has been designed in such a way, that a slightly imprecise segmentation of an action does not inhibit correct classification.

Space-Time Shape Classification: Figure 22 depicts the processed histograms for one space-time shape representing the opening of a cupboard. Three persons were asked to repeat the same action 3 times, at different time intervals, and a fourth person once. Notice that the histograms are matched almost perfectly, even though the subjects participating in the experiment performed the action differently.

To evaluate the histograms of different actions, we computed several distance metrics and compared the resulted values. As indicated by [18], the metrics which gave the best, but also similar, results were the Chi-Square ($\chi^2$) divergence and the Kullback-Leibler (KL) divergence (see Equation 4). The curve

![Global Feature Histograms for different resolutions](image)

**Figure 19:** Analysis over the resolution and scale factors for the computation of a GPFH for a 625D scale. Top: the computed histograms for 10 different resolutions. Bottom: the KL distance of each GPFH to the highest level of resolution detail (1% downsampled) from the presented set.
Figure 20: Some examples of manipulation actions recorded in a kitchen environment. Actions were performed several times by different actors, and recorded by 4 static cameras.

Presented in the upper part of Figure 23 shows the values obtained by computing the KL divergence between a set of histograms representing different action-shapes and the mean $\mu$-histogram for one action (opening cupboard). Each of the values shown on the $x$-axis of the plot is encoded as $Nt_{act}$, where $N$ is the first initial of the person performing the action, $t$ represents the trial number (i.e. the number of the experiment), and $act$ is an acronym describing the action name. The 3 evaluated actions are: a) OC for opening the cupboard; b) O for opening the oven door; and c) MB for moving a bottle from a table to another table. The first 10 values on the $x$-axis are the KL distance values for the histograms in Figure 22, from which the mean $\mu$-histogram was computed. The following three values are obtained by computing the divergence between three histograms representing the action of opening the oven, and the last three values for the action of moving the bottle. Note how the distances appear to be in the same clusters, in the sense that their values are very close to each other, thus demonstrating the robustness of our method to categorize actions efficiently.

The second part of Figure 23 presents 2 histograms obtained by: subtracting a histogram for a different action (opening the oven door) from the $\mu$-histogram of the action opening the cupboard (top plot), and subtracting a histogram for the same action but performed by a different person from the same $\mu$-histogram (bottom plot).

The acquisition process of a space-time shape is clearly dependent on the position of the camera in the environment with respect to the action performed. It is therefore obvious that certain gestures or movements will not be captured in the most relevant way by all cameras, thus their resulted his-
Figure 21: Automatic segmentation result (AS) for the trials Zoltan # 3 (top) and and Dejan # 1 (bottom) as compared to a manual segmentation (Human). The action labels are shown for illustration purposes only and are not provided by the automatic segmentation.

tograms might contain a high degree of ambiguity between classes of actions. Since our environment is instrumented with several cameras, we performed one experiment to see the differences in the resulted histograms for a gesture using two different cameras, one located behind the subject at a ≈ 55° angle, and one in front at ≈ 80° angle.

Figure 24 show the results of our test, which indicate that the differences between the actions of two subjects (Z1 and J2) captured using camera cam1 are much smaller than the ones capture by cam0. This demonstrates our above analysis, and suggest that using a single camera in such an environment is inadequate for capturing all the aspects of a given action accurately. We plan to extend our processing pipeline to deal with data coming from all cameras in our future work.

For a comprehensive classification of the recorded 3D space-time shapes, we have gathered several data sets from multiple subjects, extracted their histograms and used them to train a SVM (Support Vector Machines) classifier. Preliminary results yielded very good result, above 96% in most of the cases.
Figure 22: Feature histograms for the space-time shape representing the opening of a cupboard. The experiment involved 3 subjects, out of which 3 were asked to repeat the action 3 times.

9 Conclusions

In this paper we have presented a system for the acquisition, segmentation, and classification of human motions out of 2D silhouettes extracted from image data. We introduced a combination of methods that cover all necessary steps of an automatic action recognition system, from the feature extraction to the temporal segmentation of actions and the final classification. The key novel aspect of our work is the reasoning about motion recognition at a different level, that is, by transforming and solving the problem into a different space: that of 3D point cloud based geometry. Preliminary results show a good applicability of our approach for datasets coming from indoor environments where mounting stationary cameras is relatively easy and cheap.

The input data is acquired using either standard video cameras or any other sensor capable of extracting human silhouettes. Extracted silhouette contours are stacked in time, creating a continuous 3D shape structure encoding the observed motions. An automatic segmentation algorithm based on suffix trees and weighted interval scheduling is used to segment the continuous stream of silhouettes. The segmented silhouettes are then transformed into 3D space-time shapes, i.e. a 3D point cloud representation of actions. By minimizing the number of relevant frames and computing informative Global Point Feature Histogram (GPFH) descriptors, classes of actions can be robustly identified and categorized.

Though our proposed approach already showed that it can deal with variations in the data, such as actions performed by different persons or at different time intervals, future work will include additional experiments on larger datasets using more variate actions. Furthermore, we want to address one of the main drawbacks of the current method, namely its view dependency, by extending it to multiple cameras. As we have shown in our experiments, actions that are indistinguishable from one camera perspective can often be distinguished from another perspective (Figure 24). Problems with view-dependency usually occur when the body motions that are characteristic for an action are not visible in the extracted silhouettes due to self-occlusions of body parts. As the resulting silhouette shapes have small variance,
we believe that good camera views for action recognition could automatically be found by selecting views where the silhouette shapes show the highest variance. We have already presented a method to calculate similarity of silhouette shapes in Section 5. In addition, multiple camera perspectives could be used as independent experts, and their votes could be combined for the final classification, possibly weighting them based on the silhouette shape variance of each perspective.

Acknowledgements

This work is supported by the CoTeSys (Cognition for Technical Systems) cluster of excellence. We would also like to thank our colleagues Zoltan Marton and Nico Blodow for fruitful discussions.

REFERENCES


Figure 24: Comparing the same action using different cameras for two subjects. Notice the differences in the histogram plots for the data captured using one camera (cam0) against the ones from another camera (cam1).


