Action Graph: A Versatile Data Structure for Action Recognition

Jan Baumann, Raoul Wessel, Björn Krüger and Andreas Weber
Department of Computer Science II, Bonn, Germany
{*baumannj, wesselr, kruegerb, weber*}*@cs.uni-bonn.de*

Keywords:
Animation, Action Recognition, Accelerometer, Motion Capture

Abstract:
This work presents a novel and generic data-driven method for recognizing human full body actions from live motion data originating from various sources. The method queries an annotated motion capture database for similar motion segments, capable to handle temporal deviations from the original motion. The approach is online-capable, works in realtime, requires virtually no preprocessing and is shown to work with a variety of feature sets extracted from input data including positional data, sparse accelerometer signals, skeletons extracted from depth sensors and even video data. Evaluation is done by comparing against a frame-based Support Vector Machine approach on a freely available motion capture database as well as a database containing Judo referee signal motions and concludes by demonstrating the applicability of the method in a vision-based scenario using video data.

1 INTRODUCTION

Consumer motion capture systems (like Kinect, WiiMote, EyeToys, accelerometers) have received a lot of attention in recent years, primarily because they enable the user to interact with an application in a very natural way using low cost hardware. The field of usage exceeds replacing the classic game controller in computer games, as new applications beyond the field of computer games are emerging. This paper is motivated by such a novel example application: The automated detection of Judo referee signals, i.e. the recognition of full body movements as belonging to a set of small motion segments which are detected as certain referee signals (usually denoted by their Japanese names). Taking the developed method to the gym would allow for cost-effective automatic score counting and time keeping and greatly reduce the administrative overhead required at Judo competitions.

Technically, a fully data driven action recognition scheme is devised, where motion sequences can be detected in real time. The method is very flexible concerning the used sensor input data, which can range from high quality optical motion capture data, over medium quality Kinect skeletons to highly noisy accelerometer readings. Adding robust feature extraction from video data to the recognition pipeline even enables the approach to detect actions from video input. All this various input data can be compared in realtime with previously recorded sample motions in a motion database. The framework detects if the performed motion is similar (and possibly time-warped) to one of the annotated clips contained in the database.

The method requires very little preprocessing—only sample motions for each action to be recognized have to be labeled by the name of the action. No further explicit learning phases are required. Additional flexibility comes from the ability to add action templates to the used action database in an online manner, requiring only minimal processing.

For the purpose of evaluating different aspects of the method it is applied to prerecorded high quality motion capture data, to live captured low quality motion data obtained by a Microsoft Kinect sensor, to features extracted from video data and to a sparse accelerometer sensor setup with only four sensors attached to the actor's body. Interestingly, even for these very sparsely distributed accelerometers, the method is able to detect some actions, making it very effective in a low-cost sensor setup.

2 RELATED WORK

Related work for the method can be divided into four groups, image-based action recognition, 3D point trajectory action recognition methods, methods using accelerometers as sensors and data-driven techniques in the field of computer animation.

The first group of techniques uses 2D information such as images coming from a video camera to infer information about the actions taking place. The work by (Bobick et al., 2001) presents a view-based approach to action recognition using *temporal templates*, which are static vector-images where the vector value at each point is a function of the motion properties at the corresponding spatial location in an image sequence. (Schuldt et al., 2004) use local space-time features in combination with SVM classification schemes for action recognition.

The second group works directly on 3D point trajectory data. (Barbič et al., 2004) show methods for automatically segmenting motion capture data into distinct behaviours. The work by (Campbell and Bobick, 1995) presents techniques for representing movements based on space curves in subspaces called *phase spaces*, recognizing actions by calculating distances between these curves at every time step.

(Arikan et al., 2003) use an interactively guided *Support Vector Machine* to generalize example annotations made by a user to the entire motion capture database. Their approach works well on the small (7 minutes) motion capture database presented in their paper. The method presented in this paper uses a similar SVM approach for comparison with the developed method.

Data-driven k-nearest neighbor approaches have been quite popular in the field of computer animation in recent years. In the context of synthesizing motions, (Chai and Hodgins, 2005) show how to transform the positions of a small number of markers to full body poses. For nearest neighborhood pose searches, they construct a *neighbor graph*, allowing approximate NN-searches and requiring a quadratic preprocessing time in the size of the number of poses in the database. (Krüger et al., 2010) improve the method presented in (Chai and Hodgins, 2005) by querying a kd-tree for determining the neighborhood of a query pose resulting in exact neighborhoods for arbitrary query poses.

A novel and very intuitive puppet interface is used by (Numaguchi et al., 2011) to retrieve motions from a motion capture database. By sketching actions with the 17-degree of freedom puppet, the method matches the puppet's sensor readings retargeted to human motion to behaviour primitives stored in the motion database.

(Raptis et al., 2011) develop a method to classify human dance gestures by using a special angular skeleton representation designed for recognition robustness under noisy input. They use a cascaded correlation-based classifier for multivariate time-series data in combination with a dynamic-time warping based distance metric to evaluate the difference in motion between a performed gesture and an oracle for the matching gesture. Although the classification accuracy of their approach is very good, it assumes that the input motion adheres to the underlying musical beat, whereas the approach presented here does not rely on such assumptions.

Another class of methods is about using accelerometers for activity recognition. (Bao and Intille, 2004) present a system designed for context-aware activity recognition detecting everyday physical activities from acceleration data. They focus on a semi-naturalistic data collection protocol to train a set of classifiers, and find this is best evaluated by decision tree recognition algorithms. Along the same lines, (Maurer et al., 2006) study the effectiveness of activity classifiers also within a multi-sensor system. Their analysis of the proposed activity recognition and monitoring system concludes it is able to identify and record a subject's activity in real-time.

While (Ravi et al., 2005) also study the activity recognition techniques, they present a solution using only a single triaxial accelerometer worn within different data collection setups. Within this context, they analyze the quality of known classifiers for recognizing activities with particular emphasis on the importance of selected features and level of difficulty of recognizing specific activities. The system developed by (Khan et al., 2010) is capable of recognizing a broad set of human physical activities using only a single triaxial accelerometer. The approach is of higher accuracy than the previous works due to a novel augmented-feature vector. Additionally, they provide a data acquisition protocol using data collected by the subjects at home without the researcher's supervision.

Since activity-aware systems have inspired novel user interfaces and new applications, recognizing human activities in smart environments becomes increasingly important. In this spirit, (Choudhury et al., 2008) propose an automatic activity recognition system using on-body sensors. Several real-world deployments and user studies show the relevance of using the results to improve the hardware, software design, and activity recognition algorithms to context-aware ubiquitous computing applications.

This paper presents a method which is data-driven and uses motions from a motion capture database to construct a prior-database. Publicly available datasets, like the Carnegie Mellon University motion capture database (Carnegie Mellon University Graphics Lab, 2004) and the HDM05 (Müller et al., 2007) library, recorded at the Hochschule der Medien in Stuttgart, contain large amounts of motion capture data.

In this paper we used data from the HDM05 library, which contains more than three hours of systematically recorded and well documented motion capture data. Of great benefit in the evaluation of the action recognition method are the manually cut out motion clips that were arranged into 64 different classes and styles. Each such motion class contains 10 to 50 different realizations of the same type of motion covering a broad spectrum of semantically meaningful variations. The resulting motion class database contains 457 motion clips of a total length corresponding to roughly 50 minutes of motion data.

3 OVERVIEW

The workflow of the proposed action recognition method can be divided into three distinct processes. First, in an offline step, the motion capture database is created from motion data, where the quality can range from sparse and noisy data such as of a sensor setup using only a single accelerometer to high accuracy optical motion capture data. All such data sets can easily be handled and are manually or automatically annotated by specifying start and end frames as well as a keyword for labeling. This is followed by a preprocessing phase in which a kd-tree is created using a specific feature set allowing fast k-nearest neighborhood searches on the poses in

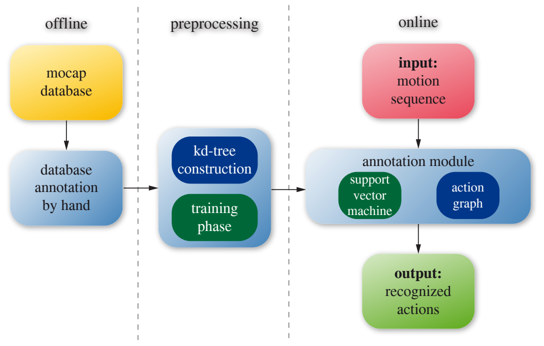

Figure 1: Workflow of the proposed method. Starting out with a motion capture database annotated with actions of interest in an offline phase, the method builds up a kd-tree from this data in the preprocessing phase. The online phase then consists of feeding new frames of the input motion into the annotation module, recognizing actions as soon as the actor finishes executing them.

the database. This feature set depends on the application which, in turn, is interdependent on the specific type of motion capture system respectively the employed sensor setup.

As of now, the workflow can be split into the *action graph*-based (dark blue in Fig. 1) and the SVM-based (green in Fig. 1) action recognition workflow. The SVM-based method is implemented for comparison with the *action graph* during evaluation. In the online phase of the *action graph* method, actions are recognized from any type input motion sequence by feeding new frames of the input motion into the annotation module. This module then uses similar poses retrieved from the kd-tree built up from the motion capture database in an action aware neighborhood graph (see Subsection 4.3) to output all recognized actions as soon as they are detected. Within the SVM-based situation, the preprocessing phase consists of learning SVM parameters on a training set, whereas in the online phase, the SVM classifier checks whether a frame derived from the input query motion belongs to a previously annotated action.

Since the approach at hand is generic, the input need not be of a specific data type and may even cover cross-modal signals.

4 ACTION RECOGNITION METHODS

Evaluation is done by comparing two different action recognition methods side-by-side, the online-capable *action graph*, where the input motion sequence is efficiently compared to annotated motions in the motion capture database and an offline *Support Vector Machine* (SVM) approach similar to one that was introduced for motion capture data by (Arikan et al., 2003).

4.1 Data Preparation

Since the preparation of motion capture data for our method is highly dependent on the application and sensors used, one cannot give general rules for preparing the data in the database or for processing the poses of the query motion. Various applications presented in the results in Section 5 show realistic examples.

4.2 Data Annotations

For the training phase of the action recognition methods, as well as for evaluations during the testing phase, accurate annotations are needed. Annotations inform the system at which time in a motion sequence specific actions are performed by the actor. For this reason, in this method, all used datasets were annotated by hand. Another possibility would be to use automatically annotated mocap data, e.g. by methods presented in (Arikan et al., 2003) or (Wyatt et al., 2005). In this work, the decision was made on a complete, reliable, manual annotation procedure to ensure that the results are not affected by false, automatically computed annotations. For each relevant action, an annotator gives a start frame, an end frame and a keyword that describes the performed motion, ultimately creating a mapping from database frame f to the annotations stored in f.

4.3 Action Graph Based Recognition

The presented action recognition method searches for motion segments that are similar to annotated actions in the motion database by taking into account the temporal continuity of the underlying motion. This is in contrast to the SVM-based approach presented in Subsection 4.4 for comparison, which ignores this information and decides whether a frame belongs to an action on a frame-by-frame basis, leading to many possible ambiguities.

To this end, the *action graph* detects if an action ends at the current frame and then tries to find possibly time-warped motion segments spanning the action in its entirety. Looking at the individual pose neighborhoods of the knn-search alone can lead to possibly many different annotations. By using the *action graph*, paths representing motion segment matches can be found through the annotated neighborhoods, resolving the ambiguity.

Basically, the method presented in this paper extends the *Lazy Neighborhood Graph* (LNG) proposed by (Krüger et al., 2010) to find motion segments similar to the currently performed motion, enabling its use for action recognition. In its original implementation, the LNG first retrieves the k nearest neighbors from a motion database for every pose in the query motion. To bridge the gap from these locally matching poses of the retrieved pose neighborhoods to globally matching similar motions in the database, their method constructs a directed acyclic graph by regarding the retrieved local neighboring poses from the motion database as vertices in the graph. Now, an edge connects a pair of neighbors of temporally adjacent pose neighborhoods, if certain stepsize conditions are satisfied, similar to *Dynamic Time Warping*. In its simplest form, the step tuple $(step_{pose}, step_{time}) = (1, 1)$ connects pose p at time t to pose $p + 1$ at time $t + 1$. By allowing additional step tuples, the results could also possibly be time warped. After having made all possible connections, a single source vertex s is connected to all the pose neighbors in the first neighborhood. The problem of finding a motion contained in the database which is most similar to the query motion can now be reduced to solving a single-source shortest paths problem. Starting the search at vertex s, one only has to check whether there exists a path which terminates at a vertex in the last neighborhood. The entire global matching can be solved in $O(km \log(n))$, where k is the number of retrieved nearest neighbors, m the number of frames contained in the query motion and n the number of frames in the motion capture database.

In contrast to the original implementation of the LNG, the developed action recognition framework tries to find motion segments which start close to the beginning of an annotated action having the currently processed frame close to the terminating frame of an annotation (see Fig. 2). This is accomplished by first inspecting the pose neighborhood of the current frame f_n for annotated ending poses. For each of these found action ending poses, a search for a starting pose annotated with the same action is performed in all neighborhoods up to frame f_{n-1}. All of these starting poses are then connected with the single source vertex mentioned above. If an annotated action in the database is similar to the currently performed motion and is contained in the pose neighborhood queue of length w, the single-source shortest paths algorithm is able to find paths from the start of an action to the end of the action, containing only the specified annotation. The found actions are possibly time-warped according to the allowed time-steps, making the method very flexible and robust to time-variations in motion performance.

In the technical part of the motion detection scenario, we test by considering a query motion take, e. g. from a Kinect recording, against a set of query action classes A, that is, annotated classes which are present in the data base. As a result of searching the *action graph* for motions associated with these class annotations we get a set of path candidates $C = \{s_i\}$ consisting of similar motion segments s_i. The size of this set depends on the employed data base, but may range from zero to several thousand retrieved segments. Consequently, we compute the percentage of detected paths s_i from the *action graph* which agree with the query set A, thus automatically addressing the fundamental question whether we came across any action contained in A. We collect the annotations which are represented most strongly, i. e. make for more than 10% of the whole set C and computing the respective start and end frames of these as follows. We calculate the frame window which contains the intersection of the annotation ground truth and a minimum of 75% of all according paths retrieved from the *action graph*. The start and end of this window marks the start and end frame of the annotation at hand. Note that in case we aim at evaluation rather than detection, we follow a slightly different protocol. This will be addressed in Section 5.

4.4 SVM-Based Action Recognition

We additionally evaluate frame classification based on a Support Vector Machine (SVM) with a standard Radial Basis Function kernel (RBF). To this end we use the LibSVM implementation (Chang and Lin, 2011). Optimal SVM parameters C balancing hyperplane minimization and the influence of slack variables as well as the RBF kernel width γ are determined using grid search with cross validation. To reduce the time consumption for training, we use only 30% of the frames of each training sequence that are chosen randomly. To take possible influence of this random selection into account, we conduct four runs each time using a different training frame selection and average the resulting classification accuracies.

5 RESULTS

Applications used for evaluation

For evaluation purposes, we considered five applications. First, in Section 5.2.1, we performed action recognition tests on a cut dataset taken from the HDM05 motion capture database. For this reason we separated the cut sequence database into a training part that contained exactly nine realizations of each motion class, and a testing part that contained at least 3 realizations. The same motion capture database is then used to test our algorithm on a sparse accelerometer setup, detecting actions using a total of four simulated accelerometers on the wrists and ankles.

In Section 5.2.2, we tested the behavior of the methods with Judo referee signal movements in an online scenario, using query motions coming from an optical mocap system. In this scenario, the database contained typical referee signals performed by three different actors with at least three repetitions. This database was captured with a Vicon motion capture system and the motion capture data was stored in the skeleton based .v file format.

We also modified the previous scenario to a cross-modal scenario, where the query motion was captured with a Microsoft Kinect sensor, obtaining the skeletal data using the Microsoft Kinect SDK. For this reason, the Judo database had to be resampled to the native frequency of the Kinect sensor (30Hz).

In the last application example (Section 5.2.3), interest points extracted from video data serve as input for the proposed algorithm, demonstrating applicability in a vision-based context.

For each of the evaluated applications, we build up two databases, one including motion clips of the actor and the other with clips of the actor excluded.

Some applications which require poses in the database to be comparable need to perform a normalization step on each pose, making them scale-

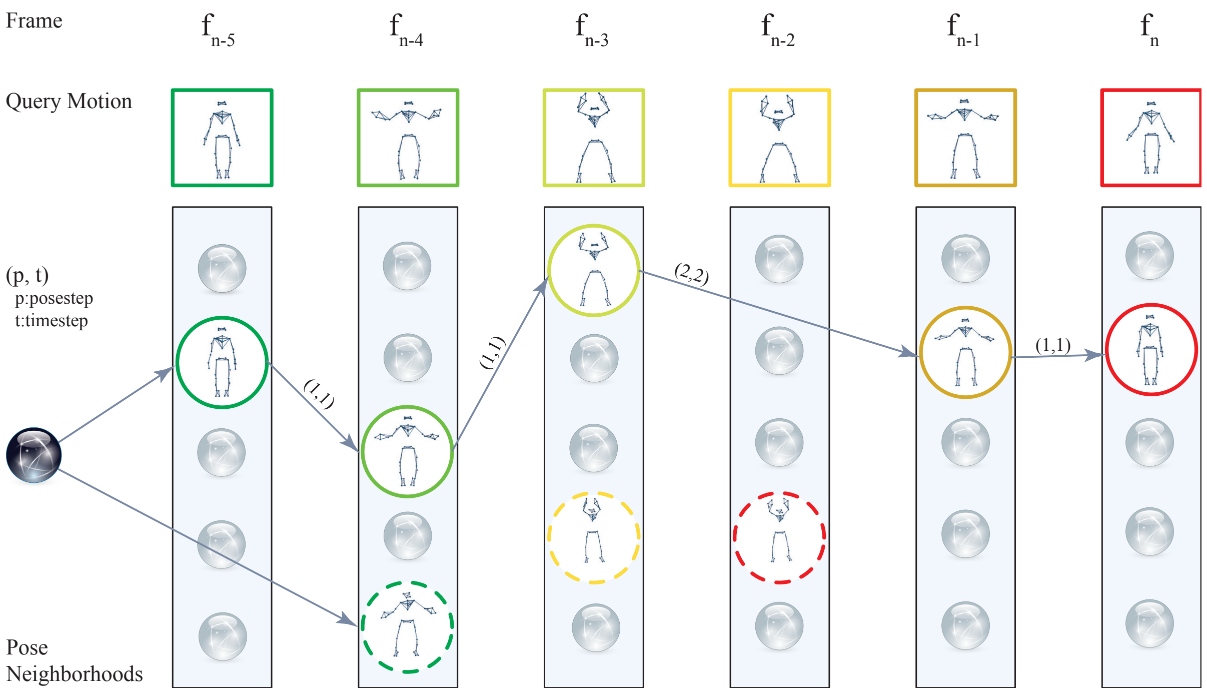

Figure 2: Detecting actions in current frame f_n using the *action graph*. In this example, we illustrate detecting a 'Jumping Jack' motion which is performed in the last six frames (f_{n-5} to f_n) using a window of $w = 6$ frames. The poses of the 'Jumping Jack' are color coded, ranging from green to red, representing the start and end of the action, respectively. First, all poses annotated with starting poses of actions (green) are connected to the single source vertex required for the single-source shortest path algorithm, regarding all past neighborhoods up to window size w. Now, for every neighborhood, poses are connected with edges according to the allowed time and pose steps S ($S = \{(1,1),(2,2)\}$ in this example). After running the single-source shortest path algorithm, we check for every candidate path terminating at an action ending pose (red pose in neighborhood f_n) whether the nodes on the path are consistently annotated with the same action, in which case this action is reported as found. Note that every 'JumpingJack' motion contains a 'ClapAboveHead' in its middle, as can be seen in the pose neighborhoods (dashed circles). Consequently, this clap is also detected, but at an earlier stage (frame f_{n-2}).

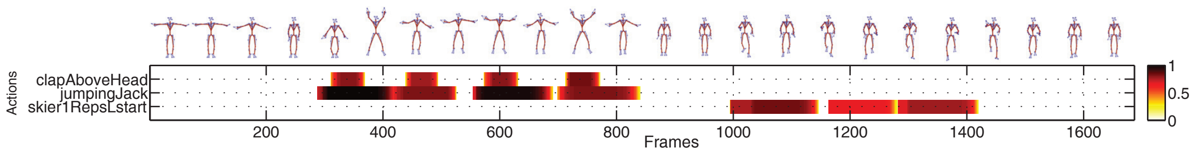

Figure 3: Example action recognition run on frames $0-1686$ of motion HDM_bd_03-05_02_120 from the HDM05 motion capture library, comprising four jumping jack motions followed by three complete and one half skiing motion starting with the left foot. Note that our method also detects sub-actions like the clap above the head, which is contained in the middle of each jumping jack. The half-executed skiing motion at the end is not detected, because the *action graph* is unable to find an annotated end frame in this case.

and view-invariant. Along the lines of (Krüger et al., 2010), the root node of the skeleton is transformed such that the skeleton faces forward and is anchored at the global coordinate frame origin. If the skeleton is given in a hierarchical representation (e.g. HDM05 and Judo database skeletons), the root node's position is translated to $(0,0,0)^T$ and its orientation is set to the multiplicative identity quaternion, followed by a forward kinematics calculation to update the remaining skeleton nodes. When normalizing skeletons where joint positions are given in absolute world coordinates with no rotational information (e.g. Kinect skeletons), the orientation of the root node is estimated by exploiting rigid connectivity between the pelvis and its neighboring joints, similar to the normalization step used for raw optical marker data in (Baumann et al., 2011).

To obtain scale-invariance, the bones of any query skeleton are resized to match the skeleton that was used to build up the database.

Description of the Evaluation

Allowing detection of more than one particular action at a time does not make sense for evaluation of the detection method, especially when this is done by means of confusion matrices. The decision criteria presented in Section 4.3 which allow for several strongly represented action classes to contribute to the detection results, is clearly not suitable for evaluation purposes. Instead, we decide for the single most strongly represented action class found in the *action graph* paths. To evaluate the quality of the decision method we distinguish the following cases:

1. The retrieved motion paths lie completely within the relevant ground truth interval, in which case the method is regarded as properly working.
2. The motion paths lie outside the ground truth interval as a whole, in which case the method is dismissed as incorrect (this was rarely observed to be the case).
3. The retrieved path set intersects the ground truth interval, in which case we further differentiate: if this intersection includes more than 90% of the total retrieved paths, the method is considered to work well, otherwise this hypothesis is dismissed.

According to the above, matrices similar to confusion matrices are used to visualize the performance of the action recognition algorithm. The columns of the matrix represent instances of the recognized actions while the rows represent the actual actions. Taking into account the cases in which the algorithm fails to detect any action, a column labeled *none* is added. A perfect action recognition would have a confusion matrix with 1 on the diagonal and 0 for every other element.

5.1 Details on knn search

Choosing a feature set for the HDM05 cut database was straightforward. The results from (Krüger et al., 2010) indicated that feature set $\mathcal{F}_{\mathrm{E}}^{15}$, which includes the positions of the head, hands and feet, would work very well on this database. The confusion matrices presented below (Fig. 4) confirm that this assumption holds. Instead of encoding temporal information (e.g. velocities) in the feature set directly, these are represented in the structure of the *action graph*: Edges are inserted between successive database indices, according to the allowed step size conditions.

According to (Krüger et al., 2010) we use the step sizes (1,1), (2,1), (1,2) and (2,2). Thus, the *action graph* easily runs in real-time when searching for 256 nearest neighbors, achieving an average frame rate above 75Hz in a multi-threaded implementation on the regarded motion capture databases. The described results were obtained using a system with an Intel hex core CPU with 3.33GHz and 24Gb of memory.

The knn search used in our approach can be replaced by a fixed radius search. This variation does not produce convincing results, due to the following reasons: First, a fixed radius can mean that we do not find any neighbors. Second, we found in our tests of this variant, that the variability between motions in some classes (cartwheel) is larger than variability in other classes (walk two steps). Therefore it is not possible to specify a uniform radius for all regarded motion classes.

5.2 Discussion of Results

5.2.1 Action Recognition Tests on HDM05 Motion Classes

We conducted action recognition tests on the HDM05 cut library, which contains manually cut out motion clips that were arranged into several different classes and styles, having multiple realizations of the same motion. These motions

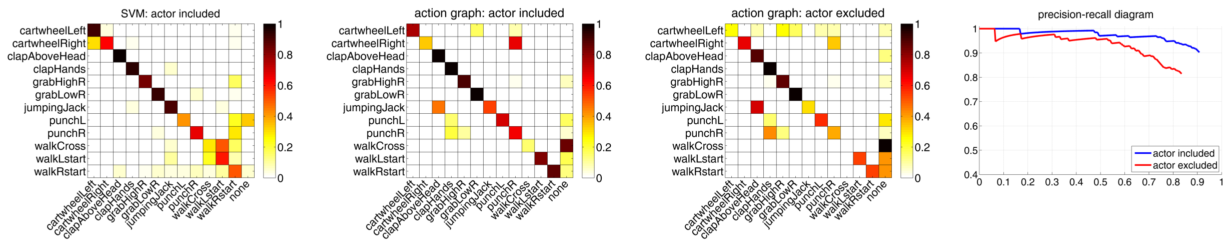

Figure 4: Confusion matrices for SVM-based and *action graph* recognition methods, calculated on the HDM05 cut database using feature set $\mathcal{F}_{\mathrm{E}}^{15}$ and the precision-recall diagram. We further distinguish the results between two different databases. In the *actor included* scenario, motion clips of the actor of the currently tested action are allowed to be present in the database. In the *actor excluded* scenario, the database is cleared of all clips of the actor in a preprocessing step.

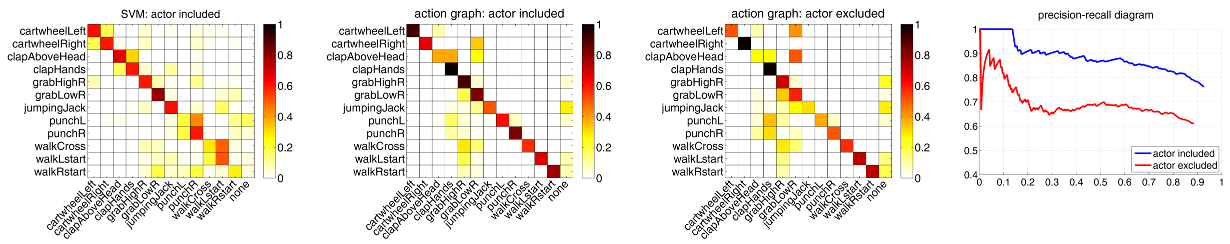

Figure 5: Confusion matrices for SVM-based and *action graph* recognition methods, calculated on the HDM05 cut database using data obtained from accelerometers attached to the wrists and ankles. On the right we show the precision-recall diagram.

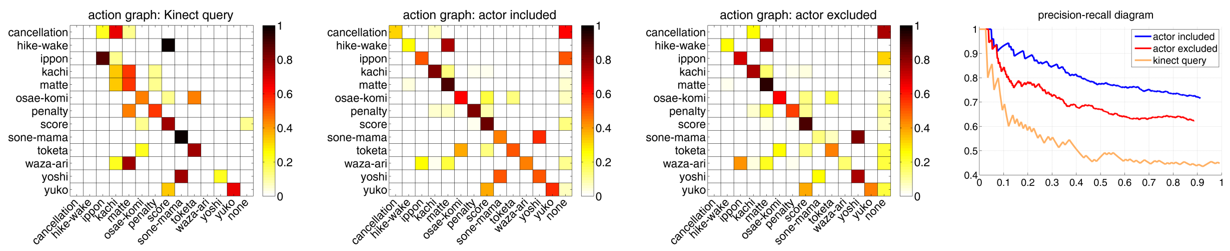

Figure 6: Confusion matrices for SVM-based and *action graph* recognition methods, calculated on Judo Motions using feature set $\mathcal{F}_{\mathrm{E}}^{15}$ for Kinect and V-Files and the corresponding precision-recall diagram.

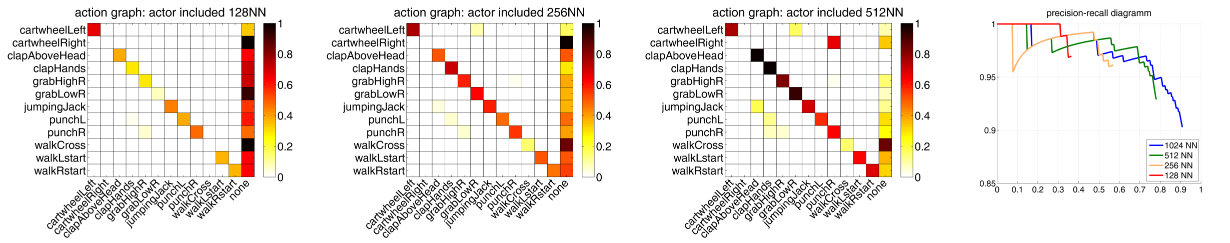

Figure 7: Confusion matrices for different values of the parameter k (128,256,512) using feature set $\mathcal{F}_{\mathrm{E}}^{15}$ on the HDM05 database and the corresponding precision-recall diagram.

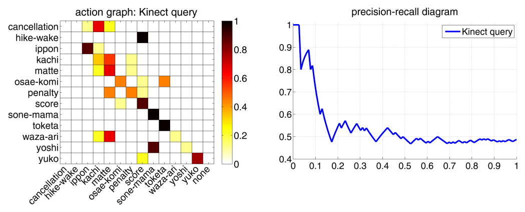

Figure 8: Confusion matrix and precision-recall diagram for Kinect queries in the Judo scenario where larger step sizes were used.

were divided into a training set, containing 142 motions, and a test set, containing 273 motions. The confusion matrices for the two action recognition methods on this dataset using $k = 1024$ in the k-nearest neighborhood search can be seen in Fig. 4. Examining the confusion matrices shown in Fig. 4, the SVM-based approach shows a good performance for a pose-based approach, having a clearly visible diagonal with a few outliers, primarily confusing walking motions. The *action graph* shows a crisp diagonal, with only two major outliers, namely recognizing a sideways punch instead of a cartwheel starting with the right hand and recognizing a clap above the head instead of a jumping jack. However, in both cases the correct actions are sub-actions of the recognized one, which is illustrated in Fig. 2 and Fig. 3 for the jumping jack motion. Also, when visually comparing the cartwheel instances with the sideways punches, the starting phases of the cartwheels show huge similarities with the sideways punches, leading to false recognitions. This indicates the method is broadly suitable. Inspecting the accuracy plot for the HDM05 library in Fig. 7, the recognition method detects 90% of actions correctly and its accuracy peaks at approximately 90% using $k = 1024$.

Testing of the method was also done by evaluating performance on a sparse accelerometer setup consisting of only four accelerometers attached to the wrists and ankles. Again, the HDM05 cut library was used for this experiment. As can be seen in Fig. 5, the pose-based SVM approach mislabels many action classes, whereas the *action graph* method shows a dark diagonal with fewer mislabelings, indicating that the method works well on this sensor setup. Taking a closer look at the results also shows that the method often confuses very similar actions, from which many are not distinguishable from each other using accelerometers alone. One such example are the clap above head and clap hands actions, which produce the same sensor reading in the given sensor setup.

5.2.2 Judo Referee Signals

In a cross-modal scenario, we used skeletons extracted from a Microsoft Kinect device to query for similar motions in the optical motion capture database containing the Judo referee signal motions. Skeletal data obtained from this sensor contains positional data only and is of much lower quality than the optical mocap data, meaning the positional noise is much more noticable and the accuracy of the system is not on par with optical systems. Since the Microsoft Kinect delivers skeletal motion data at 30Hz, whereas the optical motion capture system has a frame rate of 119.88Hz, the Vicon data is downsampled to the lower rate to obtain temporal comparability.

In order to improve the probability of finding paths through the pose neighborhoods using the *action graph*, we ran additional tests with an increased number of allowed steps (see Fig. 8). Interestingly, feature set $\mathcal{F}_{\mathrm{E}}^{15}$ gains in accuracy when allowing 8 steps and 2^{10} neighbors.

5.2.3 Action Recognition from Video Data

To show that the action recognition concept easily applies to other sorts of data, we refer to the example of video data. In order to keep emphasis on our action recognition method, we use a simple setup to demonstrate the concept. To this end, we annotate positions of hand, feet and the head in the first frame of video data and use standard feature detection methods (MSCR and SURF) to track the relevant features used in our algorithm. Based on these we obtain a ten dimensional feature set $\mathcal{F}_{\text{E-2d}}^{10}$ consisting of five two dimensional positions. Camera parameters are derived by incorporating knowledge about the scene and actors.

Since the motion database consists of three-dimensional positional data in this example and the feature extraction from video yields two-dimensional interest points, we perform parallel projections of all poses contained in the database. To handle different viewing directions, projections were performed from different viewing angles in 20 degree steps. All resulting two-dimensional features were used to construct a kd-tree for knn search. The back-projection of kd-tree indices results in database indices in the motion's original space, enabling the use of the *action graph* to detect the performed actions. Since the tracked features are very noisy in our case, the *action graph* does not return paths in all relevant cases. To alleviate this we adapted step size conditions for this scenario to allow for steps (1,3),(3,1),(4,1) and (1,4) in addition to the previously mentioned ones.

5.2.4 Comprehensive Analysis of the Results

As demonstrated by the confusion matrices in Fig. 4 and Fig. 5, the results of the above-mentioned tests show that the proposed detection method works very well on optical motion capture data and still well for accelerometer data. However, the Judo results lag far behind this good score both for motion capture data as well as data from Kinect recordings (see Fig. 6). In both cases this is partly due to the fact that the recorded referee motion repertoire turns out to be a challenge for the method in itself: For one, most of the gestures typical for Judo referees are fairly static and do not display the continual movement a sensible action recognition method is based on. Additionally, referee gestures with different meanings often differ only marginally, especially for the noisy Kinect data and its poorly aligned skeletons. This causes conditions to deteriorate.
Fig. 7 illustrates the transition of the resulting precision respectively recall for increasing choices of the number k of nearest neighbors in the action recognition test. As can be seen, the results for $k = 512$ already display satisfactorily high precision. Achieving this is obviously easy if we require as little recall. A more reliable framework forces the recall to be higher by employing a parameter $k = 1024$, although this effects in some loss of precision.

6 CONCLUSION AND FUTURE WORK

This paper examined methods to automatically detect human full body motions using motion capture data obtained from various setups. This includes working with high quality optical motion capture data, skeletons associated to the Microsoft Kinect within a cross-modal setup as well as sparse and noisy data obtained from accelerometers. Moreover, the method extends to features extracted from video data. In particular, the presented data-driven motion based detector was found to be superior to support vector machines in terms of their performance.

The approach at hand is parameterized by the employed feature sets, hence will work with other capabilities. It will therefore be a matter of future work to use and to evaluate the very recently proposed more robust feature sets (Ofli et al., 2012) within our framework. There are certain areas which turn out to serve as fertile grounds for future work: From one point of view, the application of the fixed radius search method has revealed there is a striking amount of variation in the respective retrieved pose neighborhoods of certain queries. In particular, the gaps occurring within these neighborhoods seem noteworthy. Analyzing neighborhood variation phenomena should provide interesting new insight.

Although our method is already real-time capable in many scenarios, it allows for modifications to increase this capability to scenarios with many allowed time steps and large pose neighborhoods: It is clearly not necessary to create a complete graph structure for every single frame throughout the process. Working out a more efficient solution which avoids discarding previously acquired information in the spirit of (Tautges et al., 2011) will contribute significantly to greater efficiency.
Moreover, since all significant processes involved in our method are easy to parallelize, they come with even more advantages when executed on highly parallel units. In particular, implementing the proposed techniques on a GPU seems a logical step which shall be taken in the future.

Another line of future research is the exploration of other consumer electronic devices—such as contact sensors, simple 1-or-2 axes accelerometers, altimeters, etc.—and their combinations. We refer to (Latré et al., 2011) for a recent survey of wireless body area networks and to (Khan et al., 2010) for a recent accelerometer based physical activity system. Although not all of these are suitable for our approach, many of them present promising perspectives. Especially cell phones come with an increasing variety of sensors and hence become popular objects of study. Combining the information from different sensors at different body locations combined with Bayesian a priori knowledge on the temporal evolution of human motions taken from databases—as our approach can be summarized—might be beneficial in this more general context as well.

REFERENCES

Arikan, O., Forsyth, D. A., and O'Brien, J. F. (2003). Motion synthesis from annotations. *ACM Trans. Graph.*, 22:402–408.

Bao, L. and Intille, S. (2004). Activity recognition from user-annotated acceleration data. In Ferscha, A. and Mattern, F., editors, *Pervasive Computing*, volume 3001 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin / Heidelberg.

Barbič, J., Safonova, A., Pan, J.-Y., Faloutsos, C., Hodgins, J. K., and Pollard, N. S. (2004). Segmenting motion capture data into distinct behaviors. In Heidrich, W. and Balakrishnan, R., editors, *Proceedings of the Graphics Interface 2004 Conference*, pages 185–194. Canadian Human-Computer Communications Society.

Baumann, J., Krüger, B., Zinke, A., and Weber, A. (2011). Data-driven completion of motion capture data. In *Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*. Eurographics Association.

Bobick, A. F., Davis, J. W., Society, I. C., and Society, I. C. (2001). The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:257–267.

Campbell, L. W. and Bobick, A. F. (1995). Recognition of human body motion using phase space constraints. In *Proceedings of the Fifth International Conference on Computer Vision*, ICCV '95, pages 624–, Washington, DC, USA. IEEE Computer Society.

Carnegie Mellon University Graphics Lab (2004). CMU Motion Capture Database.

Chai, J. and Hodgins, J. K. (2005). Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24:686–696.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Choudhury, T., Borriello, G., Consolvo, S., Haehnel, D., Harrison, B., Hemingway, B., Hightower, J., Klasnja, P., Koscher, K., LaMarca, A., Landay, J. A., LeGrand, L., Lester, J., Rahimi, A., Rea, A., and Wyatt, D. (April 2008). The mobile sensing platform: An embedded system for activity recognition. *Appears in IEEE Pervasive Magazine - Special Issue on Activity-Based Computing*, 7(2):32–41.

Khan, A. M., Lee, Y.-K. L. Y.-K., Lee, S. Y., and Kim, T.-S. K. T.-S. (2010). A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. *IEEE transactions on information technology in biomedicine a publication of the IEEE Engineering in Medicine and Biology Society*, 14(5):1166–1172.

Krüger, B., Tautges, J., Weber, A., and Zinke, A. (2010). Fast local and global similarity searches in large motion capture databases. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 1–10, Madrid, Spain. Eurographics Association.

Latré, B., Braem, B., Moerman, I., Blondia, C., and Demeester, P. (2011). A survey on wireless body area networks. *Wireless Networks*, 17(1):1–18.

Maurer, U., Smailagic, A., Siewiorek, D., and Deisher, M. (2006). Activity recognition and monitoring using multiple sensors on different body positions. In *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pages 4 pp. –116.

Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., and Weber, A. (2007). Documentation: Mocap Database HDM05. Computer Graphics Technical Report CG-2007-2, Universität Bonn.

Numaguchi, N., Nakazawa, A., Shiratori, T., and Hodgins, J. K. (2011). A puppet interface for retrieval of motion capture data. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Ofli, F., Chaudhry, R., Kurillo, G., Vidal, R., and Bajcsy, R. (2012). Sequence of the most informative joints (smij): A new representation for human skeletal action recognition. In *CVPR Workshops*, pages 8–13. IEEE.

Raptis, M., Kirovski, D., and Hoppe, H. (2011). Real-time classification of dance gestures from skeleton animation. In *Symposium on Computer Animation*, pages 147–156.

Ravi, N., Dandekar, N., Mysore, P., and Littman, M. L. (2005). Activity recognition from accelerometer data. In *Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3*, IAAI'05, pages 1541–1546. AAAI Press.

Schuldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: A local svm approach. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03*, ICPR '04, pages 32–36, Washington, DC, USA. IEEE Computer Society.

Tautges, J., Zinke, A., Krüger, B., Baumann, J., Weber, A., Helten, T., Müller, M., Seidel, H.-P., and Eberhardt, B. (2011). Motion reconstruction using sparse accelerometer data. *ACM Trans. Graph.*, 30:18:1–18:12.

Wyatt, D., Philipose, M., and Choudhury, T. (2005). Unsupervised activity recognition using automatically mined common sense. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 1*, AAAI'05, pages 21–27. AAAI Press.