Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2014)
Vladlen Koltun and Eftychios Sifakis (Editors)

Efficient Unsupervised Temporal Segmentation of Human Motion

Anna Vögele[†], Björn Krüger[‡], Reinhard Klein[§]

Bonn University
Institute for Computer Science II

Abstract

This work introduces an efficient method for fully automatic temporal segmentation of human motion sequences and similar time series. The method relies on a neighborhood graph to partition a given data sequence into distinct activities and motion primitives according to self-similar structures given in that input sequence. In particular, the fast detection of repetitions within the discovered activity segments is a crucial problem of any motion processing pipeline directed at motion analysis and synthesis. The same similarity information in the neighborhood graph is further exploited to cluster these primitives into larger entities of semantic significance. The elements subject to this classification are then used as prior for estimating the same target values for entirely unknown streams of data.

The technique makes no assumptions about the motion sequences at hand and no user interaction is required for the segmentation or clustering. Tests of our techniques are conducted on the CMU and HDM05 motion capture databases demonstrating the capability of our system handling motion segmentation, clustering, motion synthesis and transfer-of-label problems in practice - the latter being an optional step which relies on the preexistence of a small set of labeled data.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Due to the rapid development of devices which record and process motion data, techniques for motion recognition, analysis and synthesis applications have become evermore important. Fields of application range from research to industry and further include entertainment, health care and consumer electronics. This diversity of applications as well as the sheer amount of recorded data makes it especially important to boost fully unsupervised motion analysis methods, while, at the same time, producing high quality results and becoming even more efficient.

Motion segmentation is one area where both automation and maximization of quality are of concern. For one thing, the quantity of captured data does not allow for time-consuming manual segmentation. For another thing, any further analysis of recorded data needs accurately defined data entities to build meaningful motion models. In particular, creating a statistical motion model to facilitate motion analysis calls for a data set consisting of perceptually well-defined 'atomic' activity segments. Methods which address the segmentation problem of isolating action primitives from motion data collections have been pursued by a number of previous works [ZlTH08,ZlTH13,MC12]. While they generally work within an unsupervised setup, the problem is still far from being solved. There are two major issues which deserve particular attention:

(C1) Distinct Activities: Separate subsequences which either contain divergent repetitive successions of poses or no such patterns at all.

(C2) Motion Primitives: Detect the presence as well as the precise number of action repetitions/motion cycles. Separate these motion primitives from each other and from the rest of the motion.

† voegele@cs.uni-bonn.de
‡ kruegerb@cs.uni-bonn.de
§ rk@cs.uni-bonn.de

The two major challenges outlined above are elaborated in the following. On the one hand, to the end of motion analysis, the motion entities are clustered in a motion model. One important condition on a clustering to be conclusive is that the intra-cluster variance should be low. To this end, the first step lies in producing perceptually well-defined, commensurable activity segments. In particular, repetitions of similar motion primitives which occur at any point in the motion sequences should be isolated by the segmentation such that they can aggregate to uniform structures. The similarity of the primitives as well as the frequency of their occurrence is paradigmatic for their segmentation. This is, if there are three similar instances of a single action, say A, in the data, the according pose-sequences of each instance of A should be separated from other poses which are dissimilar to all poses of the two other instances of the action A.
With the idea of motion synthesis in mind, a motion graph which allows for synthesis with some given constraints can be built from the clusters obtained by the outlined analysis step. The *motion transitions* which connect the motion primitives have a prominent role in the synthesis graph. They stand out due to their non-repetitive nature. Frames which are temporally located between but, as such, are neither similar to poses repeated within activities before or afterward, should be separated from both preceding and succeeding segments. The poses of a motion transition, for instance, occur as part of an intermediate section between walking and running without being similar to poses of either one of both adjacent actions. In case such transitions would not be isolated from other primitives by the segmentation, this would unavoidably impose a restriction on the motion graph, since any synthesized sequence which would otherwise contain one action, a transition and another action might be forced to include additional primitives even when this is neither convenient nor intrinsically motivated. To this end, we employ a Lazy Neighborhood Graph as found in [KTWZ10], originally introduced as a fast substitute for subsequence dynamic time warping. In particular, the analysis of the connected components in the neighborhood graph is an opportunity for insights which were not discussed by the authors or any other works so far. In addition, the use of neighborhood graphs vastly reduces the complexity of the motion segmentation task. Our main contributions are:

- Both activity separation task and a primitive detection are stated as well-known graph problems.
- Activity separation which allows to distinguish between periodic structures and non-periodic transitions.
- Subdivision of activities into self-similar units, i. e. one step in a walking sequence, while efficiently detecting the correct number of repetitions of each primitive.
- Keeping up with existing techniques in terms of efficiency.

Our approach concentrates on human motion sequences which contain repetitive patterns in the sense that similar poses reoccur in a cyclic sense or other repetitive fashion. In our systematic examination of all data given in the CMU and HDM05 data bases we observed that, by taking into account all poses of sequences of length more than 8 seconds which reoccur - after some intermission - within a time slot not wider than 10 seconds, we reach 81.89% of all motion sequences in these data bases.

The rest of this paper is organized as follows. As a first step, input sequences are investigated with regard to the detection of **distinct activities** (C1). The resulting motion segments are of the type *behavior* or *activity* and could be characterized by terms like 'walk', 'dribble a ball' or 'wipe the window'. This is discussed in Section 3.1. Special emphasis is on the separation of transition phases from the rest of behaviors. The second step is detecting **motion primitives** (C2) within motion segments resulting from the first segmentation step. This is done exploiting the same self-similarity information computed beforehand (see Section 3.2). In Section 4, the clustering of thus located motion segments is discussed, resulting in a set of activity clusters with motion primitives as representatives. The results of both segmentation and clustering are presented in Section 5. One important application of our entire approach is the transfer of given labels from a small set of motion primitives to entirely unknown motion data. We demonstrate the strengths of our method by solving the transfer of labels problem based on the meaningful motion primitives accomplished by the segmentation. The results of this step along with an evaluation is presented in Section 6.1. Examples of synthesis results - which represent another important area of application - along with a detailed overview of our segmentation results by comparison with other accomplished methods will be given in Section 6.2 followed by a discussion of limitations (Section 7) and a brief conclusion (Section 8).

2. Related Work

There has been a number of attempts to effectively solve the non-trivial problem of automatic motion segmentation. The following strategies are of great significance. A partition of motion sequences into behavior segments by a PCA-based method was proposed by [BP*04]. This segmentation focuses on detecting behavior segments which is, in some respect, related to the first step of our approach. The groundbreaking work of Zhou et al. [ZlTH08, ZlTH13] has examined a segmentation concept based on (hierarchical) aligned cluster analysis (H)ACA. This segmentation technique solves the segmentation task from an angle of temporal clustering which results in motion primitives which are assigned to classes of semantic discriminability. The classification of these small motion segments, in theory, enables the creation of longer motions as compositions while raising efficiency standards.

A different strategy is based on a clustering of poses which comprise the given time series. The works of [BCvdPP08] and [BWK*13] are based on this idea. The former of the two proposes the extraction of motion motifs

as representatives of motion clusters which are the building blocks of a graph structure useful for motion compression and the creation of blend spaces. The latter propose an exploratory search and analysis system called *MotionExplorer* which also depends on the organization of single time stamps according to similarity features. Even so, the exploratory system naturally accommodates segmentations of its input sequences in the hierarchical cluster graph structure developed as part of the associated motion processing pipeline. However, their main focus was on a new visual representation of motion data rather than analysis tasks as discussed here.

From a similar angle, [LMGCG12] create temporally meaningful pose clusters which reveals a weakly supervised way of training annotations for pose clusters to decide in which cases - and temporally defined by which timestamps - an isolated action has taken place when recording the data.

Needless to say, training a segmentation of unknown data from example segments or precomputed templates is a more straightforward approach which works well depending on the quality of motion templates used as training set, that is, at any rate also on the quality of preexisting motion clips from which the class templates may be derived. Müller et al. [MRC05, MR06, MBS09] have developed a framework based on geometric features in order to learn templates for solving - and accelerating solutions to - matching problems such as annotation and retrieval [MR06, MBS09]. Adaptive segmentation [MRC05] is one of the fundamental results induced by employing geometric feature vector sequences to compare motion capture data streams at the segment level rather than frame level. [LS13] present a very much related idea in their work on learning intrinsic regularities for segmentation purposes. They demonstrate that motion capture data can be segmented by using only a limited set of example motions to segment even different action types than the ones included in the training data. That is, the authors use Latent Dirichlet Allocation (LDA) as a generative topic model, thus accounting for 'missing' data when segmenting sequences which contain actions unknown to the training system.

The segmentation problem was also solved in conjunction with motion synthesis. Two fundamental approaches can be distinguished. The first approach is based on the idea of motion concatenation [KGP02]. The authors construct a *motion graph* from motion capture data which are given as clips of elementary human motions. The synthesis problem is solved as a motion extraction task on this directed graph. A second principle is motion parameterization [KG04]. This paper introduces a technique to retrieve motion elements similar to a query from larger motion data sets which might then be blended together according to user constraints. The novel distance relation designed for this task has become a paradigm of finding logically similar objects at interactive speeds. Later works combine both these ideas [HG07, SH07] to accomplish motion synthesis techniques for high quality interactive applications. A most advanced combination of the above-mentioned ideas is found in Min and Chai's paper on Motion Graphs++ [MC12]. The authors present a generative motion model which effectively enables a variety of applications such as motion segmentation, recognition and online synthesis.

3. Segmentation

The sensory input to our method is a prerecorded sequence of motion capture data which is split up in a two-step process. In the first step, the data are partitioned into distinct temporally coherent activity segments, the second step further investigates the structure of these activities in order to find recurring patterns and, in effect, shorter *motion primitives* potentially enclosed as part of the activities.

A motion sequence M is given as a collection of n poses $p_1, \ldots, p_n$ each of which is represented by a feature vector $F = (f_1, \ldots, f_N)$ according to a frame-based set of the geometric positions of head, wrists and ankles in $\mathbb{R}^3$ (in our case, $N = 15$). In order to create a neighborhood graph G_M, we first construct a kd-tree from all given poses p_i given in the input stream, then search for similar poses within a radius r. That is, we perform a similarity search with respect to the features F in the sense that the similarity d_{ij} denotes the Euclidean distance between the features F_i of p_i and F_j of p_j. This search depends on two parameters: a radius $r \in \mathbb{R}$ and the maximum number of nearest neighbors $k \in \mathbb{N}$ it is designed to detect. As a result of this search with a fixed radius r in the according feature space we obtain a set S_i of nearest neighbors for each pose p_i. Any of the nearest neighbors in S_i is specified as a pair (j, d_{ij}) of an index j to a frame in the input motion and the Euclidean distance d_{ij} between the query and this particular neighbor.

Evidently, the search is performed only once for every frame of the input motion whereas the neighborhood information resulting from the procedure is reused for the steps of **activity separation**, **detection for repetitions** and **clustering**. Our method is thus parameterized by the feature set retrieved from the captured poses, the search radius r and the maximum number of nearest neighbors k.

The sets of nearest neighbors are suitable to replace conventional dynamic time warping by a graph based approach (for more details on how G_M can be used instead of time-warping, see Krüger et al. [KTWZ10]). Dynamic time warping as a means to calculate an optimal match between two given time series M and N with certain restrictions, creates a path between these sequences in the following sense: The sequences are matched non-linearly in the time dimension to optimize for a similarity measure. Technically, a warping path $\mathcal{P}_{M,N}$ of length λ between two such sequences is given as a pair of vectors (v_M, v_N) where $v_M = (m_1, \ldots, m_\lambda)$ with $m_i \in M$ meeting constraints such as $m_i \leq \mu m_{i+1}$ for all indices and $v_N = (n_1, \ldots, n_\lambda)$ with $n_i \in N$ meeting $n_i \leq \nu n_{i+1}$. In our case, these were $\mu = \nu = 2$. The constraints on v_M could be seen as the **continuity** of the path $\mathcal{P}$, while the preconditions on v_N correspond to the **slope** of the path.

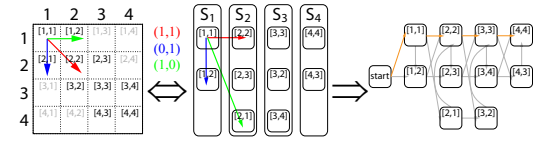

Figure 1: *Toy example illustrating the relationship between SSM and neighborhood graph G_M. a): SSM of four consecutive poses with allowed steps indicated by arrows. Red arrow: step $(1,1)$, blue arrow: step $(2,1)$, green arrow: step $(1,2)$. b): Neighborhoods of each of the four poses, e. g. S_i is the neighborhood of the pose corresponding to entry (i,i) in matrix a). c): Resulting neighborhood graph.*

All nearest neighbors p_j stored in the set of neighborhoods $S = \{S_i,\ i \in 1,\ldots,n\}$ are considered as nodes of the graph G_M. The criteria according to which different nodes are connected by edges in this graph, i. e. the accessibility between poses, can be characterized by the concept of dynamic time warping. Therefore, we will briefly recall this concept to clarify how to access one pose in the neighborhood graph from another. Consider two poses $p_j \in S_i$ and $p_{j^*} \in S_{i^*}$. A valid time warping step to access the pose p_{j^*} from p_j is defined as a pair $(a,b) \in \{(1,1),(0,1),(1,0)\}$ such that $p_{j^*} = p_{j+b}$ and $S_{i^*} = S_{i+a}$. In particular, the pose p_{j^*} is always an entry further below in the neighborhood list S_{i^*} than p_j is in the list S_i, while S_{i^*} is either identical to S_i or lies to its right hand side. Figure 1 shows a toy example to illustrate a possible scenario.

The local relations between poses p_i and p_j in the sequence M detected in the search can be visualized by a sparse-self-similarity matrix (SSM) $\mathcal{S}_M$ (Refer to Figure 2 for an example). Sparse self-similarity matrices are never actually **symmetrical** because the pose similarities which constitute the matrix block representation are restricted to a maximum number of admissible neighbors. Therefore it is possible, a neighbor p_l is taken into account for pose p_u, which, in turn, does not appear to be in the neighborhood of p_l because the number k cuts the list of p_l's neighbors before p_u is found. Nevertheless, those parts of the matrix where a minor diagonal has a fairly symmetrical match 'mirrored' by the main diagonal arise from more perceptually coherent matches in the nearest neighbor search. In short, preserving **symmetry constraints** up to a point leads to more stable results.

One important property of the graph G_M which we are looking to exploit lies in the following observation: Each individual diagonal part of $\mathcal{S}_M$ reflecting local similarities is one **connected component** of G_M.

In the following, it will be useful to remove all neighbors that belong to the same connected component cc as a given frame. Let the restricted graph which results from this be denoted by G_{cc} (Refer to Figure 3 for a visualization of connected components). A similar way of restricting G_M will also be convenient: restricting the neighborhood graph to nodes which are temporally situated between two given frames f_1 and f_2 results in a subgraph denoted $G_{\mathrm{f}_1,\mathrm{f}_2}$.

The construction of the neighborhood graph M_G, i. e. the computation of all retrieved pose based nearest neighbors for the whole motion sequence, is quite efficient being in in $O(kn\log n)$. Moreover, the whole graph structure will be reused in the later steps.

3.1. Segmentation into Distinct Activities

Referring to the typical SSM sketched in Figure 2 while taking into account that the heterogeneously structured diagonal blocks are representations of the activities we are looking to isolate, we decided to pursue **region growing** as a problem solving approach. There are three types of motions represented by the matrix in the example: the first activity, a shorter, intermediate action and the second activity. The repetitive property of each larger action is reflected by secondary structures parallel to the main diagonal in the upper left (respectively the lower right) part of the matrix, while the diagonal part connecting the two reflects a transition. The automatic detection of these three heterogeneous structures is a restatement of the segmentation task in terms of decomposing the SSM into three smaller blocks.

The general idea is to start growing a square connected region from a seed in the upper left corner of the neighborhood representation matrix. This region is gradually extended to adjacent rows and columns as long as the number of nearest neighbors in the updated region is increasing, too. If no new neighbors are found in the larger region, the current region is considered complete. A new region is then started from the upper left entry of the remaining matrix where every entry left of the last column respectively beneath the last row of the just completed region block is removed from the original matrix. This approach has one important precondition: the main diagonal part of the matrix needs to be removed.
In terms of the graph representation (which is the setting the actual solution is implemented in), the trick is to remove all nodes from the graph which belong to the same connected component as the first neighbor of the first (seed) frame. While we visualized the process by means of the self-similarity matrices for clarity and comprehensibility, we really count the number of indices in the neighborhood sets $S(i_\mathrm{start}),\ldots,S(i_\mathrm{end})$ where the index i is specified by $i_\mathrm{start} \leq i \leq i_\mathrm{end}$ where i_start and i_end are the respective start and end frames of the motion segment corresponding to the current region.
This region growing process is performed once as a **forward step**, seeding the first region at frame $\mathcal{S}_{11}$ of the matrix, to identify the **end frames** of repetitive patterns (see Fig. 2 (b)) and once as a **backward step** where the region growing process is seeded at the last frame $\mathcal{S}_{nn}$ of the input sequence. This second step identifies the **start frames** of the actions (Fig. 2 d)). Note that the worst case complexity of this step is

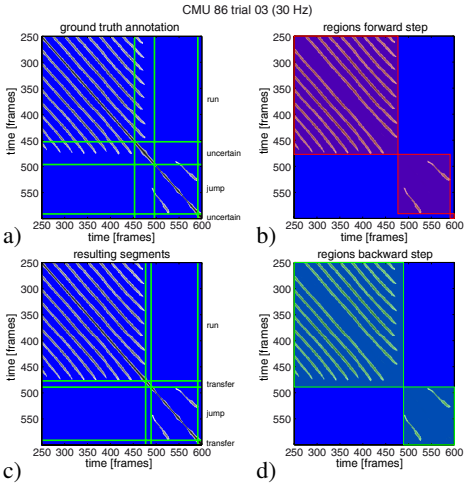

Figure 2: *Region growing for activity separation: a) Sparse self similarity matrix with ground truth annotations; the label 'uncertain' indicates an area of inconclusive user annotations. b) Same matrix without main diagonal; also results for forward step of region growing. c) Final outcome of segmentation. d) Results backward step of region growing.*

$O(k\frac{n\cdot(n-1)}{2})$, the worst case being that the first region grows from the first to the last frame of the input motion sequence.

3.2. Subdividing Activities into Motion Primitives

Once the input motion sequence is cut in to distinct activities, a search for recurrent motion primitives is performed for each of the resulting individual segments. Consider a given activity isolated by the first segmentation step, this could be a walk as in the example shown in Figure 3. We want to find the reoccurring units (such as steps) which the activity consists of. Such units are responsible for minor diagonals in the sparse self similarity matrix of the specific activity. Start and end frames of the primitives correspond to start and end frames of the minor diagonals. The basic idea is to use minimal cost warping paths in the matrix whose start and end positions are associated with the start and end frames of the minor diagonals. Thus, there is no need for an exhaustive search since every minor diagonal corresponds to a connect component in G_M.
To this end the neighborhood graph is restricted to contain only nodes associated with indices between the start and end frame of the considered activity and the search is carried out on this graph $G_{\text{sf,ef}}$.

All pairs of frames (v_{m_1}, v_{n_1}) associated with each of the retrieved warping paths are considered as candidates initial frames of a primitive. The set of all these warping paths is examined according to the following criteria: the **symmetry coherence** of the paths with respect to the similarity setting and the **projection coverage**. These criteria will be discussed by the next two paragraphs in that order.

As observed in the given examples (Figure 3), there

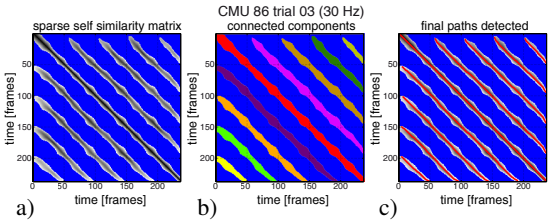

Figure 3: *Illustration of connected components in G_M. a) SSM corresponding to a walking. b) Same matrix with its color coded connected components. c) Optimal warping paths highlighted by red lines.*

is a conspicuous quasi-symmetrical structure in the self-similarity matrices which we should bestow consideration on to ensure the coherence constraints when detecting primitives: the minor diagonals in the matrix block representation encode important geometric information on the relationship between the match candidates they depict. For instance, two parallel stretches indicate there will be an warping path between the associated subsequences which has a slope near to 1, i. e. the distortion of the time dimension brought about by matching them will be small, in effect, pointing to a near-linear motion match.

There are two objectives of **path projection coverage**. Valid path candidates should not be shorter than 5 frames. Also, when projected to either the rows $i(m_1),\ldots,i(m_l)$ or the columns $j(m_1),\ldots,j(m_l)$ in the matrix index notation, the length of the respective projection of the path should not drop below the path length l by more than 50%. Note that this corresponds to admitting only a mean slope of between $\frac{1}{2}$ and 2.

Let n_i denote the number of frames of the activity and let n_{act} and e_{act} denote the number of nodes respectively of edges in the neighborhood graph G_{act} of the activity act. The number of nodes is $n_{\text{act}} \leq \min(n_i,k)\,n_i$ which in the worst case means $n_{\text{act}} = kn_i$. The number of edges e_{act} in G_{act} suffices $e_{\text{act}} \leq n_{\text{act}} n_{\text{steps}}$ where n_{steps} is the constant number of admitted warping steps. Note for the cases where the length of activity n_i is below the constant upper threshold k, the cost will be proportionally higher with $n_{\text{act}}|_{[0,k]} = n_i^2$, so graph construction can locally be even cubic in n_i but since this is only the case for $n_i < k$, there is still a valid upper threshold of $O(n_i)$. Searching warping paths with the DAG shortest path algorithm is in $O(n_{\text{act}} e_{\text{act}})$ which is equal to $O({n_i}^2)$ in our case.

4. Clustering of Motion Primitives

We may cluster activity primitives obtained from the segmentation according to their similarity. The unique characteristic is we do this without fixing a parameter to determine the number of clusters. The technical details can be found in Section 3.2, since the techniques for extracting motion primitives and grouping them together are strongly connected by their dependence on the neighborhood graph structure

paradigm. The frequency of occurrence of individual motion primitive types as well as the semantical and possibly temporal relation between them are valid information for later applications. Conveniently, all information needed to create a clustering in order to build such models is available in the neighborhood graph structure which we discussed before.

Consider a set of motion primitives $\mathbb{M} = \mathcal{M}_1, \ldots, \mathcal{M}_p$ obtained from our algorithm. We build a similarity graph denoted by $G_{\mathbb{M}}$ keeping track of the pairwise similarity of all primitives occurring in the set $\mathbb{M}$. Each node of the graph is one primitive $\mathcal{M}_\mathrm{q}$ with $\mathrm{q} \in \{1, \ldots, p\}$. An edge connecting the nodes $\mathcal{M}_\mathrm{h}$ and $\mathcal{M}_\mathrm{q}$ is added if a valid warping path can be found matching the first to the latter. Let sf and ef denote the respective start and end frames of the sequence $\mathcal{M}_\mathrm{q}$. Then the similarity comparison for a primitive $\mathcal{M}_\mathrm{q}$ against all other motion primitives is equivalent to a search for a path in the restricted similarity graph $G_{M,\{\mathrm{sf},\mathrm{ef}\}} \subset G_M$. Thus, we only need to resort to the graph G_M computed priorly to compute warping paths for each motion primitive, which can be carried out as efficiently as before (refer to Section 3.2 for the technical details). This yields a perceptually meaningful clustering of motion sequences (see Section 5 for results), where each cluster is represented by a connected component in $G_{\mathbb{M}}$.

5. Segmentation and Clustering Results

To represent the motion capture series for the experiments we used a frame-based feature set of the Euclidean positions of head, hands and feet in a normalized pose space, i.e. with respect to the skeleton's root node. Additionally, the same descriptors for the preceding and succeeding frames were combined with these to ensure a greater time-coherence of information. Enhancing the temporal consistency in this manner goes back to [KG04] in their works on extraction and parameterization of motions where they also introduced the congenial **point cloud dynamic time warping distance measure** we resort to for inter-sequence distance computation.

With this, we have a set $\mathcal{F}_{\mathrm{stacked}}$ of 45 features (15 per frame). Note that the time dependence parameterized by the feature space causes the streamlined diagonal structures in the sparse similarity matrices. A search radius of $r = 64$ centimeters in the described feature space and a number $k = 128$ of admissible nearest neighbors was fixed for the segmentation and clustering step. As opposed to Krüger et al. [KTWZ10], we enhance accessibility in the neighborhood sets by allowing $\{(2,1),(1,2),(2,2),(5,5)\}$ as additional warping steps when creating the graph G_M. This improves the stability of the segmentation. For each pose pair in G_M which suffices the outlined accessibility condition an edge is inserted between the two corresponding nodes. Since the PCA-based [BP*04] and the HACA-based [ZlTH13] method use motion sequences from subject 86 of the CMU database [CMU13] we also do this to admit a fair comparison between our's and the two methods introduced by the

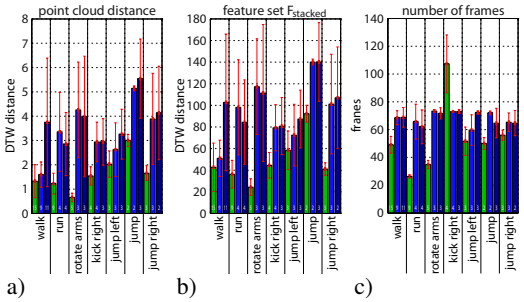

Figure 5: *Means (green bars - our method, (light)blue bars - (H)ACA) and variances (red glyphs) in the clusters of motion primitives. (a) Intra-cluster variance wrt DTW distance of point clouds. (b) Same for the DTW distance according to our feature set $\mathcal{F}_{\mathrm{stacked}}$. (c) Variance wrt the numbers of frames of the segments in each cluster. The numbers in the lower parts of the plot indicate the number of primitives in each cluster: E. g.* 15 *double steps (ground truth) detected correctly by our method, the other methods find* 9 *respectively* 11 *segments.*

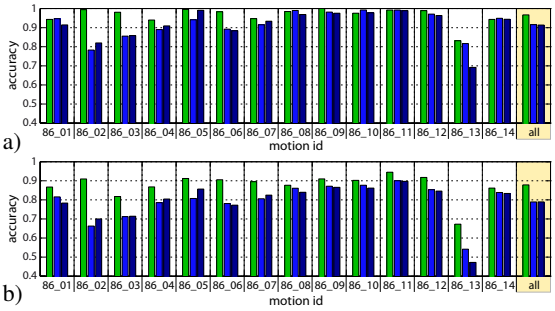

Figure 6: *Accuracy values of different segmentation methods: Green color bar is our method, light blue is ACA, dark blue HACA. Part (a): Results for the more tolerant accuracy computations, i. e. admitting the transitions as valid classes. Part (b): Evalutation wrt classes which are common to all three methods. This introduces a stronger notion of accuracy by counting strictly the classes both methods detect.*

works of Zhou et al. Refer to Figure 4 for a comparative tabulation of segmentation results. The same figure also shows the results of the clustering step for CMU 86 trials by color coding the resulting motion primitives according to the clustering discussed in Section 4 respectively by the (H)ACA methods. Note one important difference in the color codes of our method as opposed to the ground truth: Black primitives in our method are transfers between activities and are not in the same cluster (they are the segments we wanted to obtain by condition C1 on the list given in the introduction). Black areas in the ground truth annotation signify the area one standard deviation around the mean value of the three user annotations provided by Zhou et al. [ZlTH13]. Note that this standard deviation indicates there was a natural element of uncertainty in the human labeling. Our transition primitives pretty much coincide with these uncertainty areas.

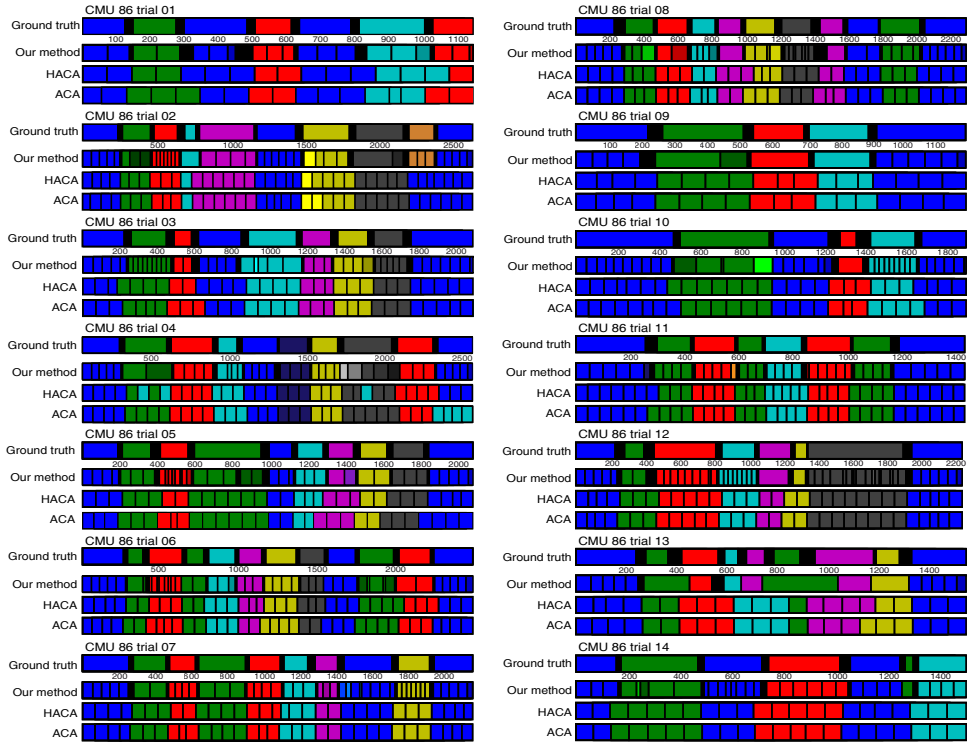

Figure 4: *Segmentation results for CMU 86 trial 01 to 14. For each of the 14 trials, the first row displays the human ground truth annotations, the second row compares them to our results. The results of the (H)ACA methods are given in the two lower rows. Note that there is a variety of units of different sizes indicating that the actual length of motion primitives may vary considerably.*

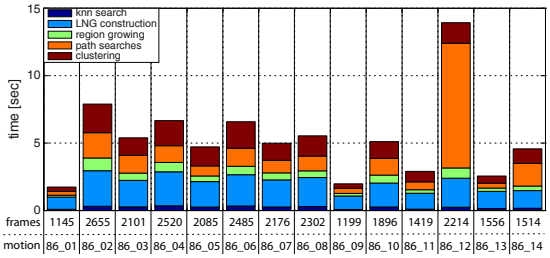

Figure 7: *Timings for all trials of subject* 86*. The color codes correspond to the different steps in the process (see Section 5.3).*

We further evaluated the performance of our approach on the HDM05 [MRC*07] database. We refer to the video for a presentation and discussion of appealing results.

5.1. Accuracy Comparison

Since the methods introduce exactly the same classes of motions found in the segmentation processes but with the addition of uncertainty classes/transitions in our case, we introduce two methods which both evaluate the achieved accuracy on a frame-wise level. For a segment s the first method checks whether its frames do belong to exactly the same motion class as the others in the same segment, the second method counts right for transion/uncertainty, too. This is done for all three methods. The results can be seen in part (a) of Figure 6. The first method introduces a stronger notion of accuracy by counting strictly the classes both methods detect, whereas the second is more tolerant towards overlaps of motion classes and preceding or succeeding transition classes. Part (b) of Figure 6 presents the results for the strict accuracy computations. Both plots show that we achieve significantly higher accuracy values. All in all, the stricter evaluation got 88% for our method, (H)ACA got 79%, the more tolerant evaluation for ours achieved 97%, HACA 92%, ACA 91%.

Figure 4 reveals that we have found a different number of primitives in all activity clusters than the other two methods. To evaluate the effect having more smaller segments has we compute the intra-cluster variance within each of the derived clusters.

5.2. Intra-Cluster Variance for Motion Primitives

The variance of a set of observed variables is commonly defined as follows. Consider a set $\mathcal{Y}$ of observed values of a variable Y with $\mathcal{Y} = (Y_1, \ldots, Y_\eta)$. The empirical variance is

$$\mathrm{Var}(\mathcal{Y}) = \frac{1}{n-1} \sum_{i=1}^{\eta} (Y_i - \mu)^2, \text{ with } \mu \text{ the mean of all values in } \mathcal{Y} \qquad (1)$$

The variance therefore depends on the distance measure used to compare different variables and to compute the mean μ. The established way to compute the distance between temporal sequences is dynamic time warping which computes the distance by accumulation of the local (frame-wise) distances from one segment to the warped version of the other. However, the mean motion needed for the conventional formula, is not defined: In probability theory mean and variance of discrete variables are defined for random variables which are functions to the *same* sample space. Finding a mean or average segment for motion segments of different lengths is a serious problem.
For a given cluster C the cumulative distance of each pair of segments s_i and s_j contained in the cluster:

$$D = \sum_{i=1,j\neq i}^{\|C\|} \left(\frac{\mathrm{DTW}_\alpha(s_i, s_j)}{\|s_i\|} \right) \tag{2}$$

where $\|s_i\|$ is the length of segment s_i and DTW_α is either the DTW distance of point clouds or of the features in $\mathcal{F}_{\text{stacked}}$. This yields a probability measure which is sensitive to outliers and will detect scattered or inconsistent clusters.

In our evaluation we computed the point cloud DTW distance and also the distance in the feature space to compare every pair of motion primitives of each activity. The accumulated distance is normalized by the number of frames of the reference motion segment. This distance is thus independent of the lengths of motion primitives. The finer distinctions result in a lower intra-cluster variance for the clusters: Figure 5 shows the variance between motion segments clustered by our, the HACA and the ACA method. Since the variance analytically depends on the employed distance features, it is fair to compare some of the most reasonable methods. The respective means are the green color bars for our method, blue color bars for (H)ACA and the variances are shown as red glyphs. Part a) shows the intra-cluster variance computed with respect to the DTW distance of point clouds, b) does the same for the DTW distance according to our feature set $\mathcal{F}_{\text{stacked}}$. Part c) shows the variance with respect to the numbers of frames of the segments in each cluster. Our method significantly reduces the variance in each of the DTW cases. This is an important effect of meeting the conditions C1 and C2 by segmenting and clustering motion primitives which are very similar. The variance in the numbers of frames is rather high in comparison which also highlights a positive fact: While the length of segments is strongly limited by both other methods (for the HACA between 45 and 80 frames regardless of the actual lengths of motions), our method does not put limits on the length of segments other than there should at least be a 5 frames which corresponds to 0.17 seconds in a segment. In fact, this facilitates finding the correct number of motion primitives in the first place as a double step in a run takes much less than a second, so the primitives in a run cluster are shorter than the ones of 'walk'. On the other hand, the slow kicks of the right leg each take longer than two seconds so the primitives are longer. Our clusters still preserve a certain variety of motion primitives but without containing transitions and other actions (C1). Also, our motion primitives exactly reflect the number of repetitions within activities (C2). For illustration, there are five repetitions of the action 'rotate arms' in one of the takes (subject 86, take 03, frames 1600-1800). Not accounting for this number but segmenting into only three primitives yields much higher DTW distances between the primitives which belong to this motion class.

5.3. Timings

On an Intel Core i7 4930K at 3.40GHz we were able to segment and cluster each motion in less than 15 seconds, using our single threaded Matlab implementation. Refer to Figure 7 for detailed timings for the CMU 86 examples. The figure also shows a case where the computation of the second segmentation step takes longer: there is an activity which is not broken down into smaller bits, so the runtime for the primitive detection (step 2) in the resulting 20 second activity segment is a few seconds.

6. Applications

At the outset, we defined a list of benefits we wanted our segmentation to entail. These features represent solutions to common problems of motion analysis and motion synthesis. Meeting the conditions on the list has a huge impact on both fields which is demonstrated by the applications we envision.

6.1. Application to the Label Transfer Problem

We show how a small set of semantic annotations that might be given for primitives or clusters of primitives can be used to transfer annotation labels to unlabeled data. The results illustrate that our segmentation method facilitates even the most straightforward automatic labeling method due to their simple and effective structure.

Consider a data base $S_{\text{unknown}} = [\mathcal{U}_1, \ldots, \mathcal{U}_n]$ of n unlabeled motion primitives as can be obtained by the segmentation in Section 3.2 and a set $S_{\text{labeled}} = [\mathcal{L}_1, \ldots, \mathcal{L}_j]$ of j labeled motion primitives. For all labeled primitives $\mathcal{L}_i, i \in [1, \ldots, j]$ we build a knowledge base of annotated primitives by computing a feature-space representation of S_{labeled}. Performing a nearest neighbor search for all unknown motion primitives in S_{unknown} and creating a similarity graph from this information admits finding optimal warping paths between any of the $\mathcal{L}$s and $\mathcal{U}$s by application of a subsequence DTW. If there is a valid correspondence between the query and a labeled primitive, the label associated with the primitive with the lowest warping path cost is transferred.
By this example we particularly wanted to show that primitives obtained by our method are well suited for the transfer of semantic annotations, while we are well aware that there are far more sophisticated machine learning techniques for the learning task. However, even by our simple strategy we achieved good results (see Table 1) which is chiefly due to

the simple structure of our motion primitives which provides comparability in the first place.
We carried out the following experiments: we build three different knowledge bases, consisting of the first, respectively the first two and the first three takes of subject 86 from the CMU data base. From these references we trained labels which were tagged by a human tester for takes 4 to 14 of the same subject. The labels were given an tested with respect to segments rather than single frames. It should be noted there is always a trade-off between achieving high precision and achieving high recall by learning methods. As is easily comprehensible, the precision of out learning step needs to be very high to enable correct synthesis results: Finding action primitives of an unwanted type in a synthesis result is highly inconvenient whereas not having a specific variation of a motion primitive represented in a result is acceptable. The afore-mentioned table demonstrates we can reach a very high precision. Note that a high level of recall was not so much in the focus of interest for the above-mentioned reasons. Transferring labels to 480 motion primitives needed approx. 35sec. we refer to Table 2 for details.

Table 1: *Results of the annotation transfer*

KB: CMU 86_01 Examples: jump 2, kick 4, punch 4, walk 7					
Class	found	correct	missed	precision	recall
punch	6	6	15	1.00	0.29
walk	117	117	26	1.00	0.83
KB: CMU 86_01 & 86_02 Examples: jump 6, kick 4, punch 7, run 7, squat 6, stretch 4, walk 23					
Class	found	correct	missed	precision	recall
punch	14	14	7	1.00	0.67
run	31	31	5	1.00	0.86
squat	3	3	0	1.00	1.00
stretch	3	3	0	1.00	1.00
walk	126	126	19	1.00	0.87
KB: CMU 86_01 & 86_02 & 86_03 Examples: jump 10, kick 7, punch 7, run 18, squat 7, stretch 4, walk 39					
Class	found	correct	missed	precision	recall
jump	1	1	21	1.00	0.087
kick	2	2	9	1.00	0.18
punch	14	14	7	1.00	0.67
run	31	31	5	1.00	0.86
squat	3	3	0	1.00	1.00
stretch	3	3	0	1.00	1.00
walk	131	129	19	0.98	0.87

Table 2: *Timings for the label transfer application. Note that we are interested in high precision rather than high recall.*

Annotation Transfer KB of 126 primitives, QDB of 480 primitives			
total time:	35.3 sec	knn search:	7.3 sec
construction of G_M:	18.2 sec	path search:	1.0 sec

6.2. Application to Motion Synthesis

Precomputed motion primitives can be used for motion synthesis with classical motion graph techniques or as input for statistical models as used in [MC12, KTMW08]. We claim that with our segmentation and clustering method we obtain motion segments that are superiorly suited for the application of such techniques. To substantiate this, we show a motion graph (see Figure 8) of motion primitives from the

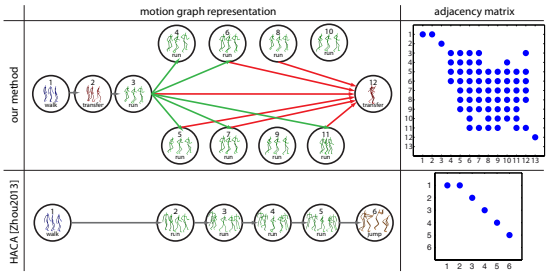

Figure 8: *Motion graphs for synthesizing running motions based on segments computed with our and the HACA method. Upper graph: green edges show possible transitions out of first run node, red edges represent transitions to transfer node. For clarity we omit all other edges within the run cluster. The given adjacency matrix encodes this information.*

CMU data base (subject 86, trial 03). The upper graph contains primitives we obtained by our method: one walk segment, all runs found in the entire sequence and the first segment after running. In our graph, there are nine primitives classified as 'run' (double step) and two transition phases. The HACA-induced graph has four primitives labeled 'run' (four steps each) and only a 'jump' as optional target node. This graph allows only for synthesis by revisiting the 'walk' segment and then linearly traversing the other nodes. As opposed to this, our graph creates much more variation: Multiple 'run' cycles may be visited from the first run node and from many of them, a transfer to 'jump' is a valid option. Every node labeled 'run' in our graph contains only one double-step while HACA-induced nodes contain four. Observe that all nodes in the 'run' cluster could be accumulated to form one super-node which corresponds to a single node in a **Motion Graph++**.

7. Discussion and Outlook

Even though the proposed method comes with a number of benefits, there also cases where it may not work in a straightforward way. Two scenarios seem particularly worth discussing. Figure 9 shows two blocks from self similarity matrices featuring two different motions. Part (a) is an example of three kicks, two of which are segmented to be in the same activity segment and the third is cut off the other two. This is a situation where the first segmentation step has failed. The result is still arguable since the second part of the segmentation would have separated three motion primitives (three kicks) which compensates the mistake of the first step. The first two kicks are similar, while the third was done with more force and different timing. While this is not a big problem, future work should investigate such issues: there might be helpful insight answering questions about how the method deals with different styles and moods which were not so much in the focus of this work but are also interesting angles. Part (b) shows an example of a motion for which the second segmentation step did not separate the primitives as

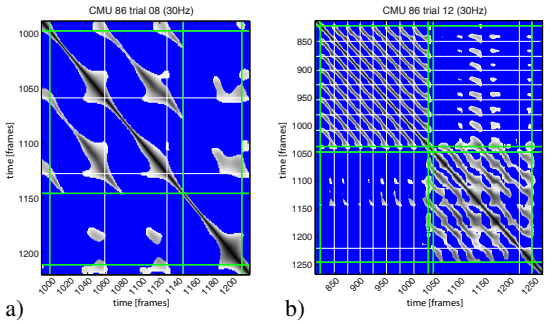

Figure 9: *a): SSM of* 3 *kicks,* 2 *of which are in the same activity segment, the third is separated (first segmentation step failed). b): Brushing dust off a dustpan (second segmentation step did not find primitives).*

expected. The activity 'brushing dust off a dustpan' which is limited to the upper extremities and has low range of motion. Effectively, the standard radius is too high for this particular activity. Making the algorithm more adaptive towards speed as well as defining more hierarchical approaches will help alleviate such problems. Taking into account the local contrary to full-body motion is a promising direction. Questions that also need answering regard the clustering of different moods and styles. So far the benefit of matching all walks over just walks with a particular character or style is not proven. Details on how to control this and what would be the outcome will yet have to be determined. The dependence on the periodicity of poses in the input motions is a minor drawback. Systematic evaluation on the used databases showed that only a small part of them is concerned by this limitation. The majority of the affected poses originate from transfers between periodic actions or stationary postures like sitting or t-poses. Another noteworthy limitation is given by over-segmentation as in example 5 (frames 400 to 600) of Figure 4. Future work will develop suitable re-clustering.

8. Conclusion

We have presented a solution to the segmentation problem of deriving motion primitives from sequences of human motion. We have created activity segments and motion primitives and clustered the latter mastering the problems C1 and C2 by exploiting self-similarities within input motions. The comparison of our results with the ones obtained by state-of-the-art segmentation techniques has pointed out the benefits of segments, as shown by motion graph based synthesis and annotations transfer. The evaluations show the profound improvements in quality.

Acknowledgements

We thank the authors of [ZlTH13] for kindly providing code and user annotations for comparison. This work was partially supported by Deutsche Forschungsgemeinschaft (DFG) under research grants KR 4309/2-1 and SPP1335.

References

[BCvdPP08] BEAUDOIN P., COROS S., VAN DE PANNE M., POULIN P.: Motion-motif graphs. In *Proc. of the 2008 ACM SIGGRAPH* (2008), SCA '08, pp. 117–126. 2

[BP*04] BARBIČ J., , PAN J.-Y., FALOUTSOS C., HODGINS J. K., POLLARD N.: Segmenting motion capture data into distinct behaviors. In *In Proc. of Graphics Interface 2004* (May 2004), pp. 185 – 194. 2, 6

[BWK*13] BERNARD J., WILHELM N., KRÜGER B., MAY T., SCHRECK T., KOHLHAMMER J.: Motionexplorer: Exploratory search in human motion capture data based on hierarchical aggregation. *IEEE Trans. on Visualization and Computer Graphics (Proc. VAST)* (2013). 2

[CMU13] CMU: Carnegie Mellon University Graphics Lab: Motion Capture Database, 2013. mocap.cs.cmu.edu. 6

[HG07] HECK R., GLEICHER M.: Parametric motion graphs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (2007), I3D '07. 3

[KG04] KOVAR L., GLEICHER M.: Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 559–568. 3, 6

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Trans. Graph. 21*, 3 (July 2002), 473–482. 3

[KTMW08] KRÜGER B., TAUTGES J., MÜLLER M., WEBER A.: Multi-mode tensor representation of motion data. *Journal of Virtual Reality and Broadcasting 5*, 5 (July 2008). 9

[KTWZ10] KRÜGER B., TAUTGES J., WEBER A., ZINKE A.: Fast local and global similarity searches in large motion capture databases. SCA 2010, pp. 1–10. 2, 3, 6

[LMGCG12] LÓPEZ-MENDEZ A., GALL J., CASAS J. R., GOOL L. J. V.: Metric learning from poses for temporal clustering of human motion. In *BMVC* (2012), pp. 1–12. 3

[LS13] LAN R., SUN H.: Automated human motion segmentation via motion regularities. *The Vis. Comp.* (2013), 1–19. 3

[MBS09] MÜLLER M., BAAK A., SEIDEL H.-P.: Efficient and robust annotation of motion capture data. SCA 2009, pp. 17–26. 3

[MC12] MIN J., CHAI J.: Motion graphs++: A compact generative model for semantic motion analysis and synthesis. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 153:1–153:12. 1, 3, 9

[MR06] MÜLLER M., RÖDER T.: Motion templates for automatic classification and retrieval of motion capture data. SCA 2006. 3

[MRC05] MÜLLER M., RÖDER T., CLAUSEN M.: Efficient content-based retrieval of motion capture data. *ACM Trans. Graph. 24*, 3 (July 2005), 677–685. 3

[MRC*07] MÜLLER M., RÖDER T., CLAUSEN M., EBERHARDT B., KRÜGER B., WEBER A.: *Documentation Mocap Database HDM05*. Tech. Rep. CG-2007-2, Universität Bonn, June 2007. 7

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. *ACM Trans. Graph. 26*, 3 (July 2007). 3

[ZlTH08] ZHOU F., LA TORRE F. D., HODGINS J. K.: Aligned cluster analysis for temporal segmentation of human motion. In *IEEE Conference on Automatic Face and Gestures Recognition* (September 2008). 1, 2

[ZlTH13] ZHOU F., LA TORRE F. D., HODGINS J. K.: Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Trans. on Pattern Analysis and Machine Intelligence 35*, 3 (2013), 582–596. 1, 2, 6, 10