Accelerating statistical human motion synthesis using space partitioning data structures

Erik Herrmann1,2 | Martin Manns3 | Han Du1,2 | Somayeh Hosseini1,2 | Klaus Fischer1,2

1 Deutsches Forschungszentrum für Kunstliche Intelligenz Standort Saarbrücken, Agents and Simulated Reality, Saarbrücken, Germany
2 Department of Computer Science, Universität des Saarlandes, Saarbrücken, Germany
3 University of Siegen, Siegen, Germany

Correspondence
Erik Herrmann, Agents and Simulated Reality, Deutsches Forschungszentrum für Kunstliche Intelligenz Standort Saarbrücken, Saarbrücken, Germany.
Email: erik.herrmann@dfki.de

Funding information
European Union in FP7, Grant/Award Number: 611007; Horizon 2020 under the Marie Skłodowska-Curie, Grant/Award Number: 642841

Abstract
Statistical model-based motion synthesis enables the constrained generation of natural human motions. In this paper, we present a combination of statistical model-based motion synthesis with a motion retrieval method. We fit a statistical motion model to the low-dimensional latent space obtained by applying principal component analysis to a functional representation of example motion data. In order to accelerate the synthesis, we construct a space partitioning tree on a dense set of samples from the model projected into a feature space. At run-time, the tree is traversed based on an objective function to find the best sample fitting user-defined constraints. After finding a good initial guess, we apply numerical optimization to reach constraints exactly. We evaluated the accuracy and efficiency of the search in the tree using different space partitioning methods on synthetic and motion capture data. In our experiments, the most stable and accurate search results were achieved using the k-Means++ algorithm for the tree construction.

KEYWORDS
character animation, human motion synthesis, motion retrieval, clustering

1 INTRODUCTION

Statistical model-based motion synthesis methods such as the method proposed by Min et al.1 can synthesize realistic human motions according to spatial and time constraints by optimizing latent motion parameters that were found using dimensionality reduction methods. The resulting motions are ensured to be natural using a statistical model of training examples as prior knowledge.

Depending on the size of the training set and the variation in the data, these methods have to search in a large motion space to find the parameters that are close to given user constraints. Using random samples to initialize the optimization, can lead to a guess far away from the global minimum. This increases the computation costs of the optimization as well as the likelihood that it leads to a local minimum. This problem becomes more noticeable when multiple paths in a large graph of statistical models have to be enumerated and evaluated to find the best sequence of steps in the graph or when multiple agents have to be simulated simultaneously.

The problem of searching in a large motion latent space is similar to the problem of motion retrieval. Motion retrieval methods such as the method presented by Krüger et al.2 search inside space partitioning data structures instead of evaluating the whole sample space. We apply a similar methodology for the evaluation of samples of a statistical motion model, in order to reduce the search space of the optimization and thus accelerate the motion synthesis.

In this work we present a motion synthesis algorithm that builds on the statistical model-based motion synthesis method proposed by Min et al.1
approach presented by Min et al.\textsuperscript{1} We extend their work by a search in space partitioning data structures to find the initial guess for numerical optimization instead of random sampling.

In the following sections, we give a brief overview of the related work on data-driven motion synthesis and motion retrieval, before we describe the motion synthesis approach. We then evaluate different algorithms for the partitioning of the space of low-dimensional motion models. The evaluation is done using synthetic data and motion capture data of manual assembly tasks.

## 2 RELATED WORK

Many data-driven motion synthesis approaches can be categorized into concatenation-based, interpolation-based and statistical model-based approaches. Concatenation-based approaches Kovar et al.\textsuperscript{3} Arikan et al.\textsuperscript{4} and Ren et al.\textsuperscript{5} construct graphs of poses and add transitions based on pose similarity. New motions can be generated by traversing the graph based on objective functions. The variation of the synthesis is restricted to the existing example data. Interpolation-based approaches such as the work by Rose et al.\textsuperscript{6} and Kovar et al.\textsuperscript{7} can create new variations via interpolation of time aligned example clips. The clips are placed in a manually defined parameter space. Interpolation weights are found by applying scattered data interpolation methods. A recent approach by Lee et al.\textsuperscript{8} also integrates motion blending with motion graphs. Statistical model-based approaches such as the work by Min et al.\textsuperscript{1,9} and Chai et al.\textsuperscript{10} model the intrinsic variation of sets of example motions as a statistical distribution in the latent parameter space found via dimensionality reduction methods. The statistical model is then used as prior knowledge in a continuous optimization to define the natural parameter range. Recently, deep learning has been applied on the modeling of motion data. Holden et al.\textsuperscript{11} propose to use a combination of a convolutional autoencoder and a feedforward network to learn a motion manifold and generate natural motion from high-level parameters. In order to support a wide range of behaviors different interpolation spaces and statistical models can be combined as nodes in a graph as done by Heck et al.\textsuperscript{12} and Min et al.\textsuperscript{1} For a more detailed review of data-driven motion synthesis approaches see the survey by Guo et al.\textsuperscript{13}

Some motion synthesis approaches such as the methods proposed by Kovar et al.\textsuperscript{7} and Heck et al.\textsuperscript{12} make use of space partitioning data structures for motion retrieval. Space partitioning data structures on low-dimensional feature representations have been explored extensively for the application of retrieving high-dimensional time series data such as motion capture data. Krüger et al.\textsuperscript{2} evaluate different feature representations for pose retrieval using kd-trees and present a fast motion retrieval method based on graphs constructed using the result of the pose retrieval queries of individual frames. Kapadia et al.\textsuperscript{14} present a motion retrieval approach that allows the definition of arbitrary key features by experts which are then used to construct a novel trie-tree data structure. A node in the trie-tree represents a key and the leaves of the tree contain references to motion clips that fit the sequence of keys which is obtained by traversing the tree from the root to the leaf. Bernard et al.\textsuperscript{15} present a visual motion data base exploration tool that makes use of k-Means clustering of poses represented as feature vectors containing 3D positions of important joints to generate interactive dendograms of pose accumulations at different levels of detail. Yoo et al.\textsuperscript{16} present a system for motion retrieval and editing with a sketching interface that uses a fast search in a motion database implemented on the GPU. Zhou et al.\textsuperscript{17} propose a method called spatial temporal pyramid matching to compare 2D histograms of a motion capture database with query motions. Deng et al.\textsuperscript{18} present an approach in which the example human poses are divided into different body parts. Using the k-Means clustering algorithm, the motion segments are then grouped into different patterns that are stored in a k-d-tree for body-part-based matching of query motions.

In this paper we evaluate the use of space partitioning data structures on the latent motion spaces of a statistical model-based motion synthesis approach. Similar to motion retrieval methods presented by Krüger et al.\textsuperscript{2} and Bernard et al.\textsuperscript{15} we construct a space partitioning data structure on a motion feature representation. However, in contrast to these methods we do not search for a motion similar to an input motion but a motion fitting to arbitrary user-defined spatial constraints based on an objective function. Compared to the work of Chai et al.\textsuperscript{9} our work differs by the use of the result of PCA as low-dimensional signal and the use of motion spaces instead of pose spaces. Therefore, in contrast to using pre-computed features, the constraints are evaluated online instead of offline. This allows constraints on individual joints and frames of a motion primitive such as in the work by Min et al.\textsuperscript{1} but more efficiently.

## 3 MOTION SYNTHESIS METHOD

In this section, we present a fast algorithm for data-driven motion synthesis based on the method presented by Min et al.\textsuperscript{1} We extend their approach by making use of a k-Means tree on the latent feature space of motion clips in order to reduce the number of samples that need to be evaluated to find a good initial guess for optimization. At run-time we use trajectory and keyframe constraints derived from a text-based user interface and an annotated 3D scene as input for the motion synthesis algorithm. An overview of the offline and online pipeline of the motion synthesis method is shown in Figure 1.

### 3.1 Motion preprocessing

We represent motion data using a sequence of frames defined by relative joint orientations. Each frame consists of the translation of the root and orientation of the joints of the skeleton relative to their parent joints. We represent orientations using quaternions.
The preprocessing of motion capture data for the construction of statistical motion models involves four steps: segmentation, temporal and spatial alignment, normalization and smoothing.

We first segment recordings of motions into smaller clips and categorize them into different classes called motion primitives. Each motion primitive represents a part of a specific action e.g. left stance of walking. For segmenting the motion capture data into smaller clips we use a semi-automatic process based on manually defined example keyframes. For cyclic motions such as walking the repeated keyframes are automatically identified and extracted.

After the segmentation, we apply dynamic time warping to align the example motions of each motion primitive into a canonical time line. For this purpose, we select the canonical time line from one example motion primitive which has the minimal average distance to the other examples. The frame similarity is measured in a transformation-invariant way using point cloud alignment based on correspondences of the joints according to Kovar et al.\(^3\) The spatial alignment of the example motions of each motion primitive is also performed using point cloud alignment.

In order to be able to project the motion primitives into low-dimensional space, the aligned motion clips need to be smoothed and parameterized as vectors. In the first step of the smoothing process, the quaternions that represent the orientation of the joints have to be aligned for all examples of one motion primitive. This is necessary to be able to use a linear dimension reduction method on the non-linear quaternion representation. In the second step of smoothing, we create a functional representation of each example motion by fitting cubic B-splines to the aligned motion clips and time warping functions. This will smooth the noisy motion capture data and moreover reduces the dimensionality by representing the motion clips using the coefficients of the B-splines.

### 3.2 Motion model construction

From this preprocessed data, we construct a latent space, in which each point represents one motion example, and similar motions are located close to each other. We apply PCA on the coefficients of the aligned motion splines and time splines separately to project the motions into a low-dimensional parameter space that captures the implicit variation of the example motions. For this purpose, we stack the coefficients of the motion spline and time spline as two high dimensional vectors and apply PCA separately resulting in spatial \(\alpha\) and time parameters \(\gamma\) which are stacked into a single vector \(s\). The parameters of the motion primitive resulting from the application of functional PCA on the training data are then modeled using the expectation-maximization algorithm as a Gaussian mixture model (GMM). The optimal number of Gaussians is selected based on the best Bayesian information criterion score.

Using this statistical model, any number of variations of the original training motions can be generated by back projecting the low-dimensional parameters of a random sample from the model into motion spline coefficients and time spline coefficients. The latent parameters can be back projected by linear combination of space and time parameters with the space and time principal component matrices resulting from PCA. The motion spline is parameterized by the canonical time line of the motion primitive and is warped by evaluating the motion spline using the time spline which maps from sample time to canonical time.

### 3.3 Space partitioning via clustering

The latent space defined by a GMM motion model results in the separation of the data into clusters. We observed that clusters identified in the latent space also correspond to clusters in the Euclidean space for interesting joints that we want to
constrain during the synthesis. We assume this is also the case for any kind of motion primitive constructed using the linear dimensionality reduction method PCA. Furthermore, we can use a projection of samples into a feature space that encodes the variation we are interested in to influence the clustering result.

In order to accelerate the synthesis from the statistical motion model we use those clusters to hierarchically partition the space and search for a sample fitting the given constraints. This sample is then used as a close initial guess for further numerical optimization.

We construct a space partitioning tree by applying the k-Means algorithm recursively on a projection of the random samples of the motion model in a latent feature space. The feature space is constructed by back projecting the latent variables into motion space and then converting the motion samples into point clouds in Euclidean space. We then apply PCA on those point clouds to project each sample into a latent feature space. Similar to Krüger et al.\(^2\) we select a subset of joints to reduce the number of dimensions. We select ten joints to allow the use of the same tree for constraints on different joints: left and right hip, two joints for each foot, and two joints for each hand. We selected the joints heuristically to represent the variation we are interested in.

The k-Means algorithm finds a partitioning of the space by assigning points to their nearest cluster centers. In multiple iterations, the positions of the cluster centers are optimized to minimize the average distance to the points in their clusters. We use the k-Means++ algorithm\(^{19}\) for the partitioning of the data into clusters on each level. This algorithm converges faster than the standard k-Means algorithm by optimizing the initial seed of the clusters in order to separate the initial clusters in a larger area. The algorithm for the tree construction is shown in Algorithm 1 where \(C\) are cluster indices resulting from k-Means.

### Algorithm 1: Recursive cluster tree construction

**Input**: data \(S \in \mathbb{R}^{n \times p}\), features \(F \in \mathbb{R}^{n \times d}\), number of subdivisions \(k\)

**Output**: tree \(t\) over \(F\) and \(S\)

1. **Function** \(\text{DIVIDE}(S, F, k)\)
2. \(\mu \leftarrow \text{mean}(S)\)
3. \(C \leftarrow \text{kMeans}(F, k)\)
4. \(N \leftarrow \emptyset\)
5. for \(\text{cluster} \in C\) do
6. \(S' \leftarrow S[\text{cluster}]\)
7. \(F' \leftarrow F[\text{cluster}]\)
8. \(N \leftarrow N \cup \text{DIVIDE}(S', F', k)\)
9. end
10. return \(t = (\mu, N)\)

Each node in the k-Means tree represents its child nodes \(N\) using the mean latent motion space parameter vector \(\mu\) of the cluster in latent motion space \(S\) but the clusters are found in latent feature space \(F\). In case the number of samples is less than the number of subdivisions, a leaf node is created for each sample.

### 3.4 Online sample search

At run-time, the k-Means tree is traversed to find the optimal sample \(s\) based on an objective function that evaluates spatial constraints on joints at one or more frames of the motion primitive. The objective function is shown in Equation (1).

\[
O(s, C) = \sum_{i=0}^{N} \sqrt{f(M(s)) - C_i^2}
\]  

(1)

Here \(s\) is the concatenated low-dimensional parameter in latent space of the spatial and time variation, \(C\) is a set of constraints on the joints of the motion at a frame of the canonical time line, \(N\) is the number of constraints, \(f\) is the forward kinematics function and \(M(s)\) is the projection from latent space to motion space.

### Algorithm 2: Candidate search

**Input**: tree root \(t\), objective function \(O\), number of candidates \(k\)

**Output**: sample \(\in \mathbb{R}^m\)

1. \(R \leftarrow \emptyset\)
2. \(G \leftarrow \{t\}\)
3. while \(|G| > 0\) do
4. \(G' \leftarrow \emptyset\)
5. for node \(\in G\) do
6. if node is not leaf then
7. Evaluate child nodes using \(O\)
8. Add child nodes to \(G'\)
9. else
10. \(R \leftarrow R \cup \{\text{node}\}\)
11. end
12. end
13. \(G \leftarrow k\) best candidates from \(G'\)
14. end
15. return best candidate from \(R\)

The algorithm for the candidate search in the tree is shown in Algorithm 2 where \(G\) and \(G'\) are priority queues of candidates and \(R\) is a priority queue of the results. The search tries to avoid ending up in a local minimum by keeping parallel candidates that traverse the tree using different branches. Using this method, the number of the evaluations is reduced from \(O(N)\) to \(O(l \times k \times c)\) where \(l\) is the number of levels, \(k\) the number of subdivisions and \(c\) is the number of candidates. The search result is a low-dimensional vector \(s\) that is back projected into a warped motion spline as described in Section 3.2.

The search in the k-Means tree can only be used to optimize spatial constraints. In order to exactly fit the spatial constraints
TABLE 1  Information on the evaluated motion models

| Model       | Examples | Keyframes | Spatial Dimensions | Time Dimensions | Feature Dimensions |
|-------------|----------|-----------|-------------------|----------------|-------------------|
| walk left   | 760      | 46        | 9                 | 9              | 4                 |
| packRight   | 29       | 103       | 10                | 6              | 6                 |
| placeRight  | 37       | 106       | 10                | 6              | 4                 |
| screw reach | 47       | 246       | 16                | 6              | 9                 |
| synthetic   | 225      | 103       | 4                 | 0              | 4                 |

and also apply time constraints the resulting sample vector needs to be optimized numerically as described in the next section.

3.5  Synthesis of constrained motion primitive sequences

We use a list of actions with constraints on joints of the skeleton as input for the motion synthesis algorithm, e.g. walk, pick, carry, and place. The constraints can be either defined in a graphical user interface or generated via a text-based user interface in combination with an annotated 3D scene. The details of the text-based interface are given by Busemann et al.20

The list of actions is sequentially converted into a single motion by generating a graph walk. In each step, we iteratively enumerate all optional transitions from the current motion primitive looking a fixed number of steps \( T \) ahead and select the best option for each step by evaluating a set of samples from the motion primitive model using Equation (1) as objective function. For the evaluation of samples we use the directed search in the k-Means tree described in Section 3.4 instead of random sampling.

In order to take transition models modeled as Gaussian Processes (GP) into account as proposed by Min et al.¹ we can modify the objective function passed into the tree search to include a predicted distribution from the GP. Equation (2) shows the extended objective function.

\[
\text{arg} \min_{s_i, C_i} \sum_{s_i} O(s_i, C_i) - \ln(p_{r}(s_i|s_{i-1}))
\]  

Here \( p_{r}(s_i|s_{i-1}) \) gives the likelihood of the GP and \( s_i \) is the parameter of the motion primitive at step \( i \) and \( s_{i-1} \) is the parameter of the previous step. However, in practice we found that a pose constraint at transitions results in satisfying transitions in combination with smoothing. The motions in the supplementary material were generated using pose constraints instead of a transition model.

Based on the initial guess found via the k-Means tree search, we numerically optimize \( T \) steps at once by stacking the parameter vectors of all steps \( s_1, \ldots, s_T \) into one vector. Equation (3) shows the extended objective function that is optimized numerically using the Levenberg-Marquardt algorithm.²¹

\[
\text{arg} \min_{s_1, \ldots, s_T} \sum_{i=1}^{T} O(s_i, C_i) - \ln(p_{r}(s_i))
\]  

Here \( s_i \) is the low-dimensional parameter in latent space for step \( i \) and \( p_{r} \) gives the likelihood from the GMM of the motion primitive that prevents the synthesis of unnatural motions. \( C_i \) is a set of constraints for step \( i \) and \( O \) is Equation (1). The resulting graph walk is back projected and concatenated into a single motion spline that can be discretized into a keyframe animation.

4  EXPERIMENTAL RESULTS

We evaluate the efficiency and accuracy of the search algorithm for different motion types for spatial constraints inside of the range of the training data. In order to investigate the effect of the space partitioning method, we compare the search result in the k-Means tree with the search result in trees constructed using other space partitioning methods. As alternative space partitioning methods, we tested clustering using GMM and a median split heuristic. We construct the GMM trees by applying the expectation-maximization algorithm recursively. The spatial median heuristic splits the data according to the median of the dimension with the maximum variance. This is one of the strategies that can be used for the construction of kd-trees.²² However, we apply a different search algorithm on the resulting tree because the query data differs from the data on which the tree is constructed. Furthermore, we use multiple subdivisions per node in the tree.

FIGURE 2  Effect of the parameters of the k-Means tree search on the search error and evaluations. Each box represents the normalized sum of the median error and median evaluations for 1000 random constraints using 10,000 synthetic training samples. The best trade-off between accuracy and efficiency is achieved using 4 clusters and 4 candidates.
We use four motion primitives in the experiments that were created using motion capture data of manual assembly tasks: walk, pickRight, placeRight and screwing. The motion was retargeted to a skeleton with 19 rotational joints in addition to the root translation. The rotation is represented by quaternions resulting in 79 pose parameters per frame. Additionally, we generate a motion primitive using synthetic data in order to evaluate the effect of the number of clusters and the number of different parameters on an example set with high density. We generate the synthetic data by modifying a limited set of example motions using inverse kinematics and smoothing to reach a 2D grid of target coordinates.

For each motion primitive we apply PCA on a point cloud representation of 10,000 samples from the statistical model to construct the low-dimensional feature space. The number of components are selected to keep 95% of the variation. Information on the number of training samples and latent dimensions of the motion primitives are listed in Table 1.

The effect of the parameters on the accuracy and the number of evaluations of the k-Means tree search is shown in Figure 2 for a tree constructed on 10,000 samples of the synthetic reach model using 1000 random position constraints on the right hand inside the range of the motion primitive. The value of each box in the figure is given by Equation (4)

\[ V(c, k) = \frac{\Delta(c, k)}{\max(\Delta)} + \frac{N(c, k)}{\max(N)} \]

where \( c \) is the number of candidates, \( k \) is the number of clusters, \( \Delta \) is the median distance to the target in cm and \( N \) is the median number of evaluations.

Figure 3 shows the results of the experiment that compares the k-Means tree search with different space partitioning methods and the ground truth. The optimal parameters for the k-Means tree and the search algorithm are set to 4 subdivisions and 4 candidates based on the result of the parameter evaluation shown in Figure 2. A brute force search, which evaluates the 10,000 samples that were used to construct the tree, is used as ground truth in the experiments. Furthermore, the result is compared with random sampling using 100 samples. The number of random samples was selected based on the average number of evaluated samples during the search in the k-Means tree using the selected parameters.

For the evaluation with motion models we generate 1000 random position and orientation constraints for one joint and one keyframe of each motion primitive sampled from the models. Additionally, we evaluate 1000 random pose constraints for each motion primitive which constrains the root, both shoulders, both hands and both feet of one frame. The distance for the position constraint is measured using the Euclidean distance. In case of the pose constraints the distance is divided by the number of constrained joints. The orientation constraints are defined by a quaternion representing a rotation in the global coordinate system. The distance between orientations is measured by the angle between reference vectors rotated by the global orientation of the joint. The constrained joints were selected based on the use case of the simulation of manual assembly workers.

The results in Figure 3 show that the k-Means tree search reduces the median error for all evaluated motion primitives in comparison with random sampling. For position constraints the median difference to the ground truth is reduced by
84 – 97% compared to random sampling, for orientation constraints by 34 – 88% and for pose constraints by 89 – 98%. For position and pose constraints, the search result reaches close to the ground truth, while requiring less than 120 evaluations in most cases. This shows that the search in the same k-Means tree supports different spatial constraint types and constraints on multiple joints. However, the accuracy of our method differs depending on the motion primitive and constraint type.

The accuracy of the search in the GMM tree is close to the accuracy of the k-Means tree, however, the median error is slightly higher by at least 10% for all tested motion primitives and constraint types except for orientation constraints. Contrary to that, the result using the median split strategy is close to the result of random sampling.

Using a Python implementation running on an Intel i7-4770K the median duration of the brute force search for a position constraint on the root, a hand and the entire pose is 0.558, 1.952 and 6.344 seconds respectively for all tested motion primitives. Using the k-Means tree the median duration of queries is 0.007, 0.020 and 0.068 seconds respectively. The time is similar for the other tree types. For orientation constraints on the root and hand the median duration of a query is 0.016, 0.030 seconds respectively and using brute force 1.502 and 2.983 seconds respectively.

5 | DISCUSSION

The comparison in Figure 3 shows that both the search in the k-Means tree and the GMM tree lead to a result that is closer to the ground truth compared to random sampling for a similar amount of evaluated samples. The median tree approach is less accurate compared to both the k-Means and the GMM-tree approach. Furthermore, the result for the median tree also does not show a significant improvement compared to random sampling. We assume this is because the k-Means algorithm and the GMM training algorithm find structures in the data while the median heuristic does not, because it does not take all dimensions into account during partitioning. Therefore, the choice of the space partitioning method matters when improving the efficiency of finding constrained motions in the latent space.

The k-Means tree leads to smaller median errors for all motion primitives compared to the GMM tree. However, both methods have varying results depending on the motion primitive and the constraint type. The most stable and accurate results were observed for the position constraint on the hand joints. For orientation constraints the result is improved as well but does not get close to the ground truth.

However, as shown in Figure 2, the accuracy can be increased by increasing the number of candidates of the search. Furthermore, the search in the k-Means tree is restricted to the variation of the discrete samples. By increasing the density of the number of samples in the tree, the accuracy of the search can be also improved. The number of candidates used during the search and the sample density is a trade-off between the accuracy and efficiency. In practice, we found that the parameters used in the experiments provide a good trade-off.

For practical applications, we suggest that further numerical optimization is used to reach constraints exactly and to be able to combine spatial and time constraints. However, due to the close initial guess provided by the search in the k-Means tree, the optimization time is reduced. Furthermore, the chance of falling into a local minimum is reduced.

In general, if the constraint space is known during the model construction, the best results can be achieved when the constraint space is used as feature space. In this case, a more efficient method would be the construction of a k-d tree as used by Krüger et al. for motion retrieval. However, our method allows for the use of arbitrary spatial constraints using the same tree and thus keeps the general applicability of the statistical motion synthesis approach.

6 | CONCLUSIONS AND FUTURE WORK

We have accelerated the statistical model-based motion synthesis method originally presented by Min et al. by replacing the random sampling step in the motion synthesis algorithm with a directed search in a space partitioning structure constructed on a discretized motion model. To find a close guess for the numerical optimization, we search in k-Means trees constructed on the latent feature space of the motion primitives. For all evaluated motion primitives, we show that the search in the k-Means tree in the latent feature space is more efficient compared to random sampling and a search in trees constructed using a median split heuristic and GMM classification. Our method was successfully tested in practice using motion capture data that was recorded for manual assembly tasks.

In applications for domains, which need fast motion synthesis of high-quality motion capture data for multiple agents, we see a performance gain compared to using random sampling for the initialization of the optimization of latent motion parameters.

As future work, we plan to evaluate other feature spaces constructed using nonlinear dimensionality reduction methods. Furthermore, another direction for future work is to improve the speed of the synthesis further by parallelization of the tree traversal.

ACKNOWLEDGEMENTS

This work was funded by the European Union in FP7 under grant agreement 611007 and in Horizon 2020 under the Marie Skłodowska-Curie grant agreement 642841. We would like to thank anonymous reviewers for improving this work.
REFERENCES

1. Min J, Chai J. Motion graphs++: A compact generative model for semantic motion analysis and synthesis. ACM Trans Graph. 2012;31(6):153:1–153:12.

2. Krüger B, Tautges J, Weber A, Zinke A. Fast local and global similarity searches in large motion capture databases. Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, Madrid, Spain; 2010. p. 1–10.

3. Kovar L, Gleicher M, Pighin F. Motion graphs. Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’02. New York, NY, USA: ACM, 2002. p. 473–482.

4. Arikan O, Forsyth DA, O’Brien JF. Motion synthesis from annotations. Proceedings of ACM SIGGRAPH 2003, San Diego, California; 2003. p. 402–408.

5. Ren C, Zhao L, Safonova A. Human motion synthesis with optimization-based graphs. Comput Graph Forum. 2010;29(2):545–554.

6. Rose C, Bodenheimer B, Cohen MF. Verbs and adverbs: Multidimensional motion interpolation using radial basis functions. IEEE Comput Graph App. 1998;18:32–40.

7. Kovar L. Automated methods for data-driven synthesis of realistic and controllable human motion. Ph.D. Thesis, University of Wisconsin at Madison, Madison, WI, USA; 2004. AAI3155067.

8. Lee Y, Wampler K, Bernstein G, Popović J, Popović Z. Motion fields for interactive character locomotion. ACM Trans Graph. 2010;29(6):138:1–138:8.

9. Min J, Chen Y-L, Chai J. Interactive generation of human animation with deformable motion models. ACM Trans Graph. 2009;28(1):9:1–9:12.

10. Chai J, Hodgins JK. Performance animation from low-dimensional control signals. ACM Transactions on Graphics (TOG), vol. 24, ACM; 2005. p. 686–696.

11. Holden D, Saito J, Komura T. A deep learning framework for character motion synthesis and editing. ACM Trans Graph. 2016;35(4):138:1–138:11.

12. Heck R, Gleicher M. Parametric motion graphs. Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, I3D ’07. New York, NY, USA: ACM, 2007. p. 129–136.

13. Guo S, Southern R, Chang J, Greer D, Zhang JI. Adaptive motion synthesis for virtual characters: A survey. Vis Comput. 2015;31(5):497–512.

14. Kapadia M, Chiang I-k, Thomas T, Badler NI, Kider JT Jr, et al. Efficient motion retrieval in large motion databases. Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, ACM, 2013. p. 19–28.

15. 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 Transactions on visualization and computer graphics (Proc. VAST), Atlanta, Georgia; USA; 2013. p. 2257–2266.

16. Yoo I, Venek J, Nizovtsvea M, Adamo-Villani N, Benes B. Sketching human character animations by composing sequences from large motion database. The Visual Comput. 2014;30(2):213–227.

17. Zhou L, Lu Z, Leung H, Shang L. Spatial temporal pyramid matching using temporal sparse representation for human motion retrieval. The Visual Computer: International Journal of Computer Graphics. 2014;30(6):845–854.

18. Deng Z, Gu Q, Li Q. Perceptually consistent example-based human motion retrieval. Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games, ACM, Boston, Massachusetts; 2009. p. 191–198.

19. Arthur D, Vassilvitskii S. K-means++: The advantages of careful seeding. Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms; 2007. p. 1027–1035.

20. Busemann S, Steffen J, Herrmann E. Interactive planning of manual assembly operations: From language to motion. In Procedia CIRP. 2016;41:224–229.

21. Marquardt DW. An algorithm for least-squares estimation of non-linear parameters. J Soc Ind Appl Math. 1963;11(2):431–441.

22. Muja M, Lowe DG. Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2014;36:2227–2240.

Erik Herrmann received his MSc Degree in Media and Communication Informatics from Reutlingen University of Applied Sciences, Germany, in 2013. He is currently a researcher in the Agents and Simulated Reality group at the German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany and a PhD candidate at the Department of Computer Science at Saarland University. His research interests include computer graphics, data-driven animation, and physics-based animation.

Prof. Dr.-Ing Martin Manns is a Professor at the Institute of Production Technology at the University of Siegen. He holds a chair in manufacturing automation and assembly since 2016. Professor Manns obtained his degree in business and industrial engineering at the University of Karlsruhe in 2000. In 2005, he received his doctorate in mechanical engineering from the University of Hanover on the topic of predicting material-flow bottlenecks in automated assembly systems with artificial neural networks. He continued his research as Post-doctorate Fellow at the University of Windsor, Ontario, Canada. Professor Manns has 10 years of industrial experience in production engineering and research working at the Daimler AG, Henkel AG & Co. KGaA and SIG positec BERGER-LAHR. His work focused on production automation and assembly optimization methods. He has coordinated the EU funded project INTERACT, in which a method for data-driven human motion synthesis from textual work descriptions has been developed and prototypically presented. Professor Manns research interests include Industry 4.0 production technologies, data-driven modeling of work processes, and assembly process chains for additively manufactured parts.
Han Du received his BSc degree in Electronic Information Science and Technology from Hefei University of Technology, Hefei, China, in 2010, and the MSc degree in Computer Science from Saarland University, Germany, in 2014. He is currently a PhD candidate at the Department of Computer Science at Saarland University and a researcher at the German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany. His research interests are data-driven character animation.

Somayeh Hosseini received the BSc Degree in Electrical Engineering from Sadjad University, Mashhad, Iran, in 2004, and the MSc Degree in Electrical and Computer Engineering from Colorado State University, Fort Collins, USA, in 2014. Since 2015, she has joined the University of Saarland and German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany, where she is currently a PhD student in computer science under the European Union’s Horizon 2020 research and innovation project DISTRO ITN. Her main areas of research interest are machine learning and data-driven animation.

Dr. Klaus Fischer is DFKI Research Fellow and head of the Multi-agent Systems (MAS) research Group at DFKI. His main research areas are intelligent agent systems and multi-agent systems, model driven system design, service-oriented architectures, and agent technologies for the Future Internet with special focus on 3D Internet. He has published 4 books and about 100 papers in scientific journals, conferences and workshops. At DFKI, he has a long record of successfully managed projects for industrial partners as well as for public authorities (e.g. EU, BMBF).

SUPPORTING INFORMATION
Additional supporting information may be found online in the Supporting Information section at the end of the article.

How to cite this article: Herrmann E, Manns M, Du H, Hosseini S, Fischer K. Accelerating statistical human motion synthesis using space partitioning data structures. Comput Anim Virtual Worlds. 2017;28:e1780. https://doi.org/10.1002/cav.1780