Human Motion Database with a Binary Tree and Node Transition Graphs

Katsu Yamane
Disney Research, Pittsburgh
kyamane@disneyresearch.com

Yoshifumi Yamaguchi
Dept. of Mechano-Informatics
University of Tokyo
yamaguti@ynl.t.u-tokyo.ac.jp

Yoshihiko Nakamura
Dept. of Mechano-Informatics
University of Tokyo
nakamura@ynl.t.u-tokyo.ac.jp

***Abstract*— Database of human motion has been widely used for recognizing human motion and synthesizing humanoid motions. In this paper, we propose a data structure for storing and extracting human motion data and demonstrate that the database can be applied to the recognition and motion synthesis problems in robotics. We develop an efficient method for building a human motion database from a collection of continuous, multi-dimensional motion clips. The database consists of a binary tree representing the hierarchical clustering of the states observed in the motion clips, as well as node transition graphs representing the possible transitions among the nodes in the binary tree. Using databases constructed from real human motion data, we demonstrate that the proposed data structure can be used for human motion recognition, state estimation and prediction, and robot motion planning.**

I. Introduction

Using a collection of human motion data has been a popular approach for both analyzing and synthesizing human figure motions, especially thanks to recent improvements of motion capture systems. In the graphics field, motion capture data have been widely used for producing realistic animations for films and games. A body of research efforts have been directed to techniques that allow reuse and editing of existing motion capture to new characters and/or scenarios. In the robotics field, the two major applications of such databases are building a human behavior model for robots to recognize human motions and synthesizing humanoid robot motions.

Databases for robotics applications are required to perform at least the following two functions: First, they have to be able to categorize human motion into distinct behaviors and recognize the behavior of a newly observed motion. This is commonly achieved by constructing a mathematical model that bridges the continuous motion space and the discrete behavior space. The problem is that it is often difficult to come up with a robust learning algorithm for building the models because raw human motion data normally contain noise and error. The efficiency of search also becomes a problem as the database size increases.

Another function is to synthesize a robot motion that adapts to current situation, which is computationally demanding because of the large configuration space of humanoid robots. Motion database is a promising approach because they can reduce the search space by providing the knowledge on how human-like robots should move. For this purpose, however, motion capture data must be organized so that the planner can effectively extract candidates of motions and/or configurations. A database should also be able to generate high-quality motion data, which is also difficult because sample motion data are usually compressed to reduce the data size.

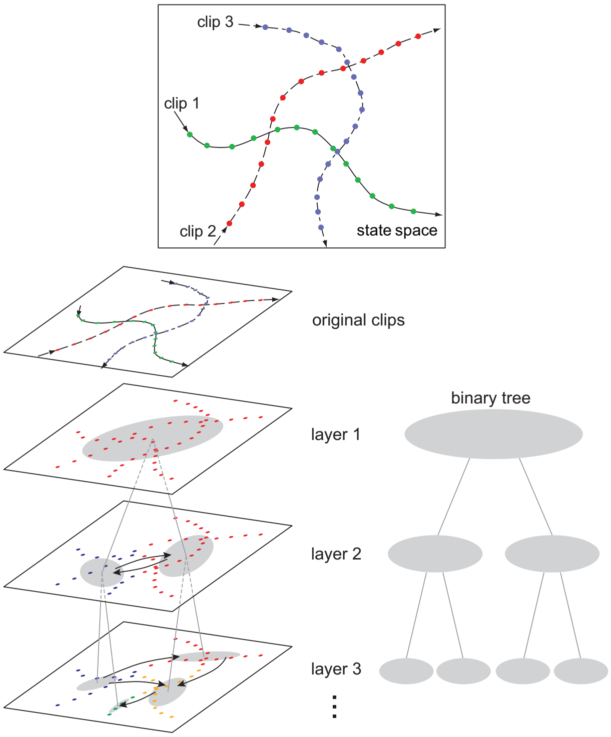

Fig. 1. Constructing a binary tree database from human motion data. Top: the three sample motion clips depicted as trajectories in the state space. Each dot represents a frame in the motion clips. Bottom left: the frames are iteratively divided into two descendant nodes, starting from a single node including the whole dataset. Each oval represents the distribution of the states in the node. The arrows represent the possible transitions between the nodes. Bottom right: the binary tree consisting of the nodes generated in the left figure. Each node contains the information on state distribution (typically the mean and covariance) and the node transitions.

In this paper, we propose a highly hierarchical data structure for human motion database. We employ the binary-tree structure as shown in Fig. 1, which is a classical database structure widely used in various computer science applications because of search efficiency. However, constructing a binary-tree structure from human motion data is not a trivial problem because there is no straightforward way to divide the multi-dimensional, continuous motion data into two descendant nodes. Our first contribution is a simple, efficient clustering algorithm for dividing a set of sample frames into two descendant nodes to construct a binary tree from human motion data.

We also develop algorithms for basic statistical computations based on the binary tree structure. Using these algorithms, we can recognize a newly observed motion sequence, estimate the current state and predict future motions, and plan new sequences that satisfy given constraints.

Another minor but practically important aspect of the proposed database is the ability to incorporate motion data from different sources. For example, we may want to include motion data captured with different marker sets, or include animation data from a 3D computer graphics (CG) film. It is therefore important to choose a good motion representation to allow different marker sets and/or kinematic models. In this paper, we propose a scheme called *virtual marker set* so that motion data from different sources can be represented in a uniform way and stored in a single database.

The rest of this paper is organized as follows. In Section II, we review the related work in graphics and robotics. We then present the proposed data structure and associated algorithms in Sections III and IV respectively, and provide several application examples in Section V. Section VI demonstrates experimental results using human motion capture data, followed by concluding remarks.

II. Related Work

In the graphic field, researchers have been investigating how to reuse and edit motion capture data for new scenes and new characters. One of the popular approaches is *motion graphs*, where relatively large human motion data set is analyzed to build a graph of possible transitions between postures. Using the graph, it is possible to synthesize new motion sequences based on simple user inputs by employing a graph search algorithm. Kovar et al. [1] proposed the concept of motion graphs where similar postures in a database are automatically detected and connected to synthesize new motion sequences. They presented an application of synthesizing new locomotion sequence that follows a user-specified path. Lee et al. [2] employed a two-layered statistical model to represent a database, where the higher-level (coarse) layer is used for interacting with user inputs and the lower-level (detail) layer is used for synthesizing whole-body motions. Arikan et al. [3] also proposed a planning algorithm based on a concept similar to motion graphs. The work in the graphics field mostly focuses on synthesizing new motions from simple user inputs using, for example, interpolation and numerical optimization [4], but such systems cannot handle the motion recognition problem.

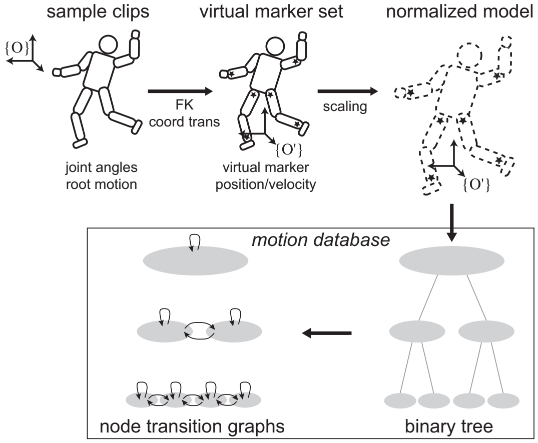

Fig. 2. The process for building a database.

In robotics, learning from human demonstration, or imitation, has been a long-term research issue [5]–[7] and a number of algorithms have been developed for storing human motion data and extracting appropriate behaviors. Because human motion varies at every instance even if the subject attempts to perform the same motion, it is necessary to model human behaviors by either statistical models [7], [8], nonlinear dynamical systems [9], [10], or a set of high-level primitives [11]. Related work relevant to this paper includes the Hidden Markov Model (HMM) representation of human behaviors [8] and the hierarchical motion database based on HMM [12]. Another hierarchical motion categorization method is also proposed using neural network models [10]. However, most work in robotics is still focused on robust learning of human behaviors. Scalability of the motion database or synthesizing transitions between different behaviors have not been investigated well. HMMs have also been used in the graphics context by, for example, Brand et al. [13] where they used HMMs to model human motions with styles. Their work is focused on learning relatively few behaviors with various styles, while we are interested in constructing a database with a variety of behaviors.

A binary-tree structure database of human motions is used in Sidenbladh et al. [14] for efficient sampling of human poses with application to human tracking in videos. They use the result of principal component analysis (PCA) of the entire motion dataset to iteratively divide the data into two nodes. In addition to the differences in the applications, there is a difference in the database construction method because we run PCA every time a new node is obtained, which would generally result in better clustering performance.

III. Building the Database

The process for building the proposed database is summarized in Fig. 2. The user provides one or more sample motion clips represented as a pair of root motion and joint angle sequences, typically obtained by motion capture or hand animation. The joint angle data are then converted to virtual marker data through forward kinematics (FK) computation to obtain the marker positions and velocities, coordinate transformation to remove the trunk motion in the horizontal plane, and scaling to normalize the subject size. The positions and velocities of the virtual markers are used to represent the state of the human figure in each sample frame.

To construct a binary tree, we first create the root node that contains all frames in the sample motion clips. We then iteratively divide the frames into two descendant nodes using the method described in Section III-B. After the tree is obtained, we count the number of transitions among the nodes in each layer to construct the node transition graphs as described in Section III-C. The binary tree and node transition graphs are the main elements of the proposed motion database.

A. Motion Representation

The source motion capture data may be represented with different marker sets or skeletons. To obtain a uniform representation of the motions, we define a *virtual marker set* with N_v virtual markers, and represent all motions by their trajectories (positions and velocities). This representation is similar to the point cloud representation [1] except that all virtual markers have been labeled.

If a motion is represented by joint angle trajectories of a skeleton model, it can be easily converted to the virtual marker set representation by giving the relationship between each marker in the virtual marker set and the skeleton used to represent the original motions. The relationship between a virtual marker and a skeleton is defined by specifying the link to which the marker is attached, and giving the relative position in its local frame. Although this approach requires some work on the user's side, it allows the use of multiple skeleton models with simple distance computation.

After converting the motion data to the virtual marker set representation, we perform a coordinate transformation to remove the horizontal movement in the motion and scaling to normalize the subject size. Each marker position is represented in a new reference coordinate whose origin is located on the floor below the root joint, z axis is vertical, and x axis is the projection of the front direction of the subject model onto the floor. The marker velocities are also converted to the reference coordinate.

Formally, a sample motion data matrix $\boldsymbol{X}$ with N_S frames is defined by

$$\boldsymbol{X} = (\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \ldots \boldsymbol{x}_{N_S}) \tag{1}$$

where $\boldsymbol{x}_i$ is the state vector of the i-th sample frame defined by

$$\boldsymbol{x}_i = \left(\boldsymbol{p}_{i1}^T \ \boldsymbol{v}_{i1}^T \ \boldsymbol{p}_{i2}^T \ \boldsymbol{v}_{i2}^T \ \ldots \boldsymbol{p}_{iN_v}^T \ \boldsymbol{v}_{iN_v}^T\right)^T \tag{2}$$

and $\boldsymbol{p}_{il}$ and $\boldsymbol{v}_{il}$ are the position and velocity of marker l in sample frame i. If multiple motion clips are to be stored in a database, we simply concatenate all state vectors horizontally and form a single sample motion data matrix.

B. Constructing a Binary Tree

A problem of constructing a binary tree for motion data is how to cluster the sample frames into groups with similar states. Most clustering algorithms require a large amount of computation because they check the distances between all pairs of frames. This process can take extremely long time as the database size increases.

Here we propose an efficient clustering algorithm based on principal component analysis (PCA) and minimum-error thresholding technique. The motivation for using PCA is that it determines the axes that best characterize the sample data. In particular, projecting all samples onto the first principal axis gives a one-dimensional data set with the maximum variance, which can then be used for separating distinct samples using adaptive thresholding techniques developed for binarizing images.

Note that the primary purpose of using PCA is not dimensionality reduction. We only use the first principal component to divide the motion capture frames into two groups and the original motion data are stored in the database, although it is certainly possible to reduce the database size by using the results of PCA. The three-dimensional space shown in the results section is used only for visualization purpose.

The process to divide the frames in node k into two descendant nodes is as follows. Assume that node k contains n_k frames whose mean state vector is $\bar{\boldsymbol{x}}_k$. Also denote the sample motion data matrix of node k by $\boldsymbol{X}_k$. We compute the zero-mean singular value decomposition of $\boldsymbol{X}_k$ as

$$\boldsymbol{X}_k^{'T} = \boldsymbol{U\Sigma V}^T \tag{3}$$

where each column of $\boldsymbol{X}'_k$ is obtained by subtracting $\bar{\boldsymbol{x}}_k$ from the original state vectors, Σ is a diagonal matrix whose elements are the singular values of $\boldsymbol{X}_k$ sorted in the descending order, and $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices. The columns of $\boldsymbol{V}$ represents the principal axes of $\boldsymbol{X}_k$. We obtain the one-dimensional data set $\boldsymbol{s}_k$ by projecting $\boldsymbol{X}'_k$ onto the first principal axis by

$$\boldsymbol{s}_k = \boldsymbol{X}_k^{'T} \boldsymbol{v}_1 \tag{4}$$

where $\boldsymbol{v}_1$ denotes the first column of $\boldsymbol{V}$.

Once the one-dimensional data is obtained, the clustering problem is equivalent to determining the threshold that minimizes classification error. We shall use a minimum-error thresholding technique [15] to determine the optimal threshold. After sorting the elements of $\boldsymbol{s}_k$ and obtaining the sorted vector $\boldsymbol{s}'_k$, this method determines the index m such that the data should be divided between samples m and $m+1$ using the following equation:

$$m = \underset{i}{\operatorname{argmax}} \left\{ i \log \frac{\sigma_1}{i} + (n_k - i) \log \frac{\sigma_2}{n_k - i} \right\} \tag{5}$$

where σ_1 and σ_2 denote the variance of the first i and last $n_k - i$ elements of $\boldsymbol{s}'_k$, respectively.

We obtain a binary tree by repeating this process until the division creates a node containing fewer frames than a predefined threshold. To make the statistical distribution of each node meaningful, we also avoid nodes with small number of sample frames by setting a threshold for the minimum number of frames in a node. If Eq. (5) results in a node with fewer number of frames than the latter threshold, we do not perform the division. Therefore, a node may not be divided if it contains many similar frames.

All branches must have the same depth so that all frames appear in all layers. Some branches may be shorter than others because we extend each branch as much as possible and some of them may hit the thresholds earlier. In such cases, we simply extend shorter branches by attaching a copy of the leaf node so that the length of all branches become the same. Each node therefore can have 0 (leaf nodes), 1 or 2 descendant nodes.

C. Node Transition Graphs

After constructing the binary tree, we then build the node transition graphs based on the transitions observed between nodes in each layer. Because we know the set of frames included in each node, we can easily determine the transition probabilities by dividing the number of transitions to a specific node by the total number of frames in the node.

We build two kinds of node transition graphs at each layer. The *global transition graph* describes the average node transition probabilities observed in all sample clips. The transition probability from node m to node n in the same layer is computed by

$$p_{m,n} = \frac{t_{m,n}}{\Sigma_i t_{m,i}} \tag{6}$$

where $t_{k,l}$ denotes the total number of transitions from nodes k to l observed in all sample clips. A *clip transition graph* describes the node transition probabilities observed in a specific sample clip. We can use the same equation (6) to compute the transition probabilities, except that $t_{k,l}$ only considers the transitions within the same sample clip. Figure 3 shows examples of global and clip transition graphs for the motion clips shown in Fig. 1.

The global transition graph at each layer is similar to motion graphs [1] in the sense that all possible transitions between nodes are included. However, the way we construct the graph is different from existing motion graph techniques, resulting in a more efficient database construction. Our method generally requires $O(N \log N)$ for a database with N sample frames because the depth of the tree is typically $O(\log N)$ and dividing the frames at each layer requires $O(N)$ computations, while most motion graph and other clustering techniques require $O(N^2)$ computations because they compute the distance between each pair of frames in the database.

The clip transition graphs are similar to human behavior models using HMMs [8]. In most HMM-based approaches, a simple left-to-right model or single-chain cyclic model with fixed number of nodes is assumed because it is difficult to train an HMM with arbitrary length or arbitrarily interconnected nodes. In our method, we do not assume the structure of the node transition or the number of nodes used to represent a sample clip. If a sample clip includes a cyclic pattern, for example, our method automatically models the cycle by producing a single transition loop, while a left-to-right model tries to encode the whole sequence within a given number of nodes.

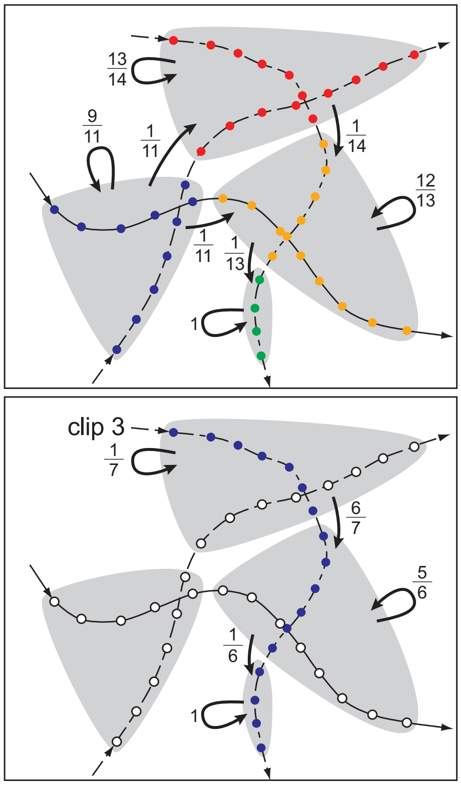

Fig. 3. Examples of node transition graphs. Top: global transition graph; bottom: clip transition graph for clip 3 (see also Fig. 1).

IV. Algorithms

For a given tree and node transition graph, we should be able to perform the following two basic operations:

- find the optimal node transition to generate a newly observed motion clip, and
- compute the probability that a newly observed motion clip is generated by a node transition graph.

In the rest of the section, we shall denote the newly observed motion comprising M frames by $\hat{\boldsymbol{X}} = (\hat{\boldsymbol{x}}_1 \; \hat{\boldsymbol{x}}_2 \; \ldots \; \hat{\boldsymbol{x}}_M)$ where $\hat{\boldsymbol{x}}_i$ is the state vector at frame i. Here we assume that both positions and velocities of virtual markers are given.

A. Optimal Node Transition

The probability that the observed motion $\hat{\boldsymbol{X}}$ was produced by a node transition $\mathcal{N} = \{n_1, \; n_2, \; \ldots, \; n_M\}$ is given by

$$P(\mathcal{N}|\hat{\boldsymbol{X}}) = \prod_{i=1}^{M} P_t(n_{i-1}, n_i) P_s(n_i|\hat{\boldsymbol{x}}_i) \tag{7}$$

where $P_t(k,l)$ is the transition probability from node k to l ($P(n_0, n_1) = 1$) and $P_s(k|\boldsymbol{x})$ is the probability that the state was at node k when the observed state was $\boldsymbol{x}$. $P_s(k|\boldsymbol{x})$ is obtained by the Bayesian inference:

$$P_s(k|\boldsymbol{x}) = \frac{P_o(\boldsymbol{x}|k)P(k)}{P(\boldsymbol{x})} = \frac{P_o(\boldsymbol{x}|k)P(k)}{\sum_i P_o(\boldsymbol{x}|i)} \tag{8}$$

where $P_o(\boldsymbol{x}|k)$ is the likelihood that state vector $\boldsymbol{x}$ is output from node k and $P(k)$ is the *a priori* probability that the state is at node k.

The actual form of $P_o(\boldsymbol{x}|k)$ depends on the probability distribution used for each node. In this paper, we assume a simple Gaussian distribution with mean $\bar{\boldsymbol{x}}_k$ and covariance $\boldsymbol{\Sigma}_k$, in which case $P_o(\boldsymbol{x}|k)$ can be computed by

$$P_o(\boldsymbol{x}|k) = \frac{1}{(\sqrt{2\pi})^N |\boldsymbol{\Sigma}_k|} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \bar{\boldsymbol{x}}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x} - \bar{\boldsymbol{x}}_k)\right).$$

$P(k)$ can be either a uniform distribution among the nodes, or weighted according to the number of sample frames included in the nodes. In our implementation, we use a uniform distribution for the global transition graph.

Obtaining the optimal node transition $\mathcal{N}^*$ is the problem of finding the node sequence that maximizes Eq. (7). A common method for this purpose is to perform forward-backward algorithm or dynamic programming. However, such algorithms can be computationally expensive for long sequences or densely connected graphs. We could omit nodes far enough from each observed frame using a threshold, but it is difficult to determine the threshold so that enough number of candidates are left for the search.

Instead of searching the entire node sequence at a single layer, we utilize the binary tree data structure by starting from the top layer. Because the top layer only contains the single root node r, the trivial optimal sequence at the top layer, $\mathcal{N}_1^*$, is to visit the root node M times, i.e., $\mathcal{N}_1^* = \{r, r, \ldots, r\}$. Starting from this initial sequence, we perform a dynamic programming to find the best way to trace the descendants of the nodes in the sequence all the way down to the bottom layer. We could also terminate at an intermediate layer if we do not need precise results and/or we have to obtain a result faster.

Figure 4 illustrates the algorithm using a simple example where we try to find the optimal node transition corresponding to a motion consisting of three frames (red circles connected by the thick arrow). At the top layer, the only possible node transition is $N_{11} \rightarrow N_{11} \rightarrow N_{11}$. Because N_{11} is divided into two nodes N_{21} and N_{22} in the second layer, there are $2^3 = 8$ possible node transitions for the three frames. However, most of them are eliminated because the first and second frames are too far from N_{21}, leaving only two possible node transitions. The two transitions expand to 16 transitions in the third layer, most of which are again eliminated due to low generation likelihood (for example, generating the first frame from N_{34}), or zero transition probability (for example, N_{34} to N_{33}). This process can be continued until the search reaches the bottom layer, or any of the layers in the middle if the application does not require high precision.

B. Motion Generation Probability

Motion generation probability is defined as the probability that a node transition graph generates the observed motion. This probability can be used for identifying the type of behavior. We can compute the motion generation probability by summing up the probability of generating the motion by all possible node transitions. However, there may be huge number of possible node transitions for long motions or large transition graphs.

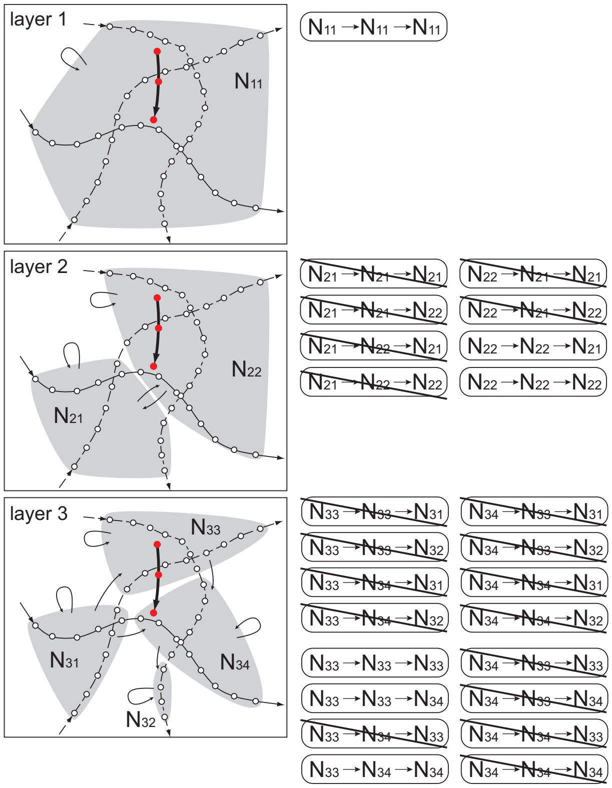

Fig. 4. Example of obtaining the optimal node transition where we find the optimal node transition at layer 3 corresponding to the motion represented by the thick arrow connecting the three red circles. Left column depicts the three layers with possible node transitions. Right column shows all node transition candidates examined in each layer, where the candidates with diagonal lines are eliminated due to either the small likelihood of generating a frame from a node (e.g., first frame from N_{34}) or missing node transitions (e.g., N_{34} to N_{33}).

An alternative used in this paper is to use the dynamic programming described in the previous subsection to find multiple node sequences. Because the algorithm returns node sequences in the descending order of probability, we can approximate the total motion generation probability by using the top few node sequences.

V. Applications

A. State Estimation and Prediction

Estimating the current state is accomplished by taking the last node in the most likely node transition in the global node transition graph. Once the node transition is estimated with high probability, we can then predict the next action by tracing the node transition graph. By combining the probability of the node transition and the probability of the future transition candidate, we can also obtain the confidence of the prediction.

B. Motion Recognition

Motion recognition is the process to find a motion clip in the database that best matches a newly observed motion sequence. This is accomplished by comparing the motion generation probability from the clip transition graphs.

C. Motion Planning

The global transition graph can also be used for planning a new motion sequence subject to kinematic constraints. In addition, because the tree has multiple layers that model the sample clips at different granularities, motion planning can also be performed at different precision levels. The only issue is how to compute the motion of the root in the horizontal plane that has been removed from the original data during database construction.

Our solution is to use the marker velocity data to obtain the root velocity, and then integrate the velocity to obtain the root trajectory. The velocity can be obtained by employing a numerical inverse kinematics algorithm, which essentially solves the following linear equation:

$$\boldsymbol{v} = \boldsymbol{J}\dot{\boldsymbol{\theta}} \tag{9}$$

where $\dot{\boldsymbol{\theta}}$ is the joint velocities including the root linear and angular velocities, $\boldsymbol{v}$ is the vector of all marker velocities extracted from the mean vector of a node, and $\boldsymbol{J}$ is the Jacobian matrix of the marker positions with respect to joint angles. This is usually an overconstrained inverse kinematics problem because the virtual marker set contains more markers than necessary to determine the joint motion. We therefore apply the singularity-robust (SR) inverse [16] of $\boldsymbol{J}$ to obtain $\dot{\boldsymbol{\theta}}$.

VI. Results

A. Sample Data Set

We use two different sets of data to demonstrate the generality of our approach. The minimum number of sample frames in a node is set to 16 in both databases. The properties of the databases are summarized in Table I.

The first set (*Database 1*) consists of 19 motion clips including a variety of jog, kick, jump, and walk motions, all captured by the authors in a motion capture studio at University of Tokyo. The motions were captured using our original marker set consisting of 34 markers (Fig. 5). This marker set is also used as the virtual marker set to construct all the databases in the experiments. *Database 1* is used to demonstrate the analysis experiments.

The second set (*Database 2*) is generated from selected clips in a publicly available motion library [17] and will be used for the planning experiment. The database includes two long locomotion sequences with forward, backward and sideward walk of the same subject. Although *Database 2* contains twice as many sample frames as *Database 1*, it resulted in relatively small number of layers and nodes probably because they consist of similar motions and hit the minimum frame number threshold earlier.

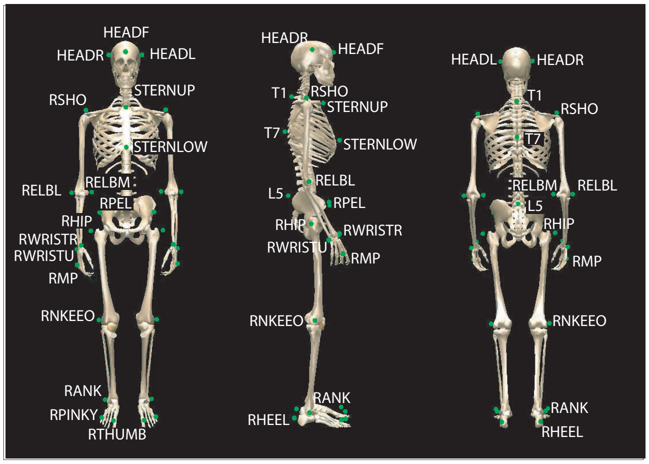

Fig. 5. The marker set used to obtain the samples in *Database 1*. This is also used as the *virtual marker set*.

TABLE I

Properties of the two databases.

	Database 1	*Database 2*
# of frames	5539	11456
# of layers	17	16
# of nodes	1372	1527
# of nodes in bottom layer	167	205

Because the virtual marker set consists of 34 markers, the dimension of the state vector is 204.

B. Properties of the Database

We first visualize *Database 1* to investigate its properties. In Fig. 6, the spheres (light-blue for readers with color figures) represent all nodes in the bottom layer and the red lines represent the transitions among the nodes. The white mesh represents the plane of first and second principal axes. The projection onto first-second and first-third principal axes planes are also shown in Figures 7 and 8 respectively. Figure 7 also includes the mean poses of the nodes at the both ends of the first two principal axes. These poses clearly show the geometrical meaning of the first and second principal axis: the first axis represents the vertical velocity and the second axis represents which leg is raised. Apparently there is no verbally explainable meaning in the third axis. Figure 9 shows the nodes and global transition graphs at layers 4 and 7, with the top layer numbered layer 1.

The location of each node is computed by projecting its mean vector onto the first three principal components of the root node of the tree. The size of a sphere is proportional to the maximum singular value of the sample motion data matrix of the node.

Figure 10 shows the node transition graphs of four representative clips. As expected from the above observation, motions such as jump that involves mostly the vertical motion stay in the first and third axes plane, while jog, kick and walk motions stay in the second and third axes plane.

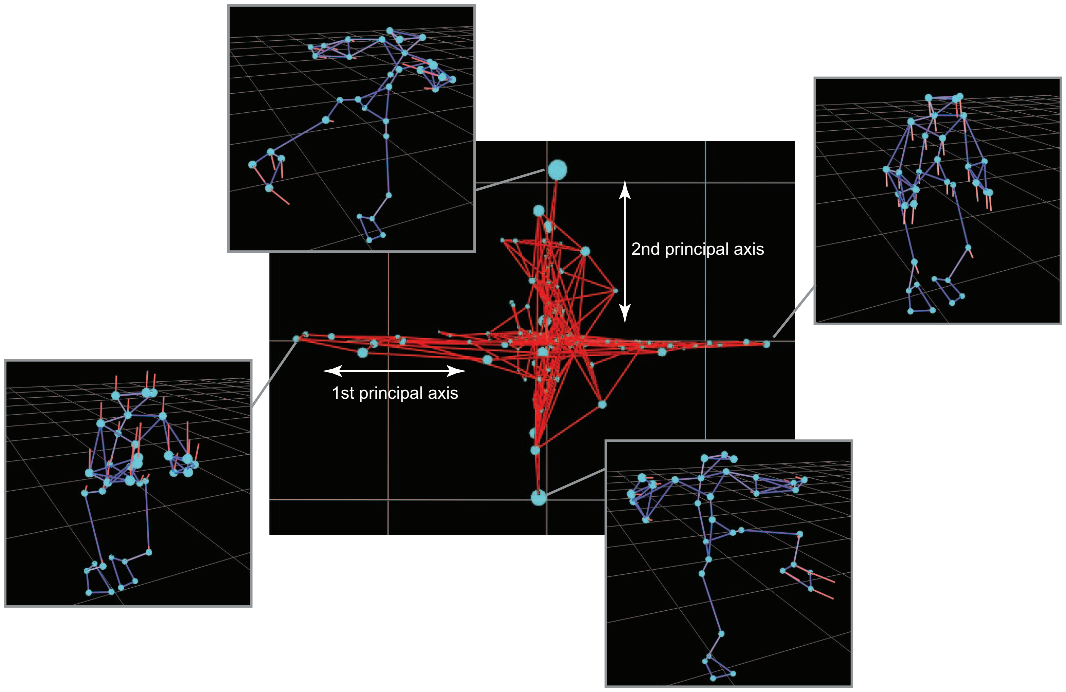

Fig. 7. Visualization of the database in the first (horizontal) and second (vertical) principal axes space. The four insets show the mean marker positions (cyan balls) and velocities (red lines) of the nodes at both ends of the two axes.

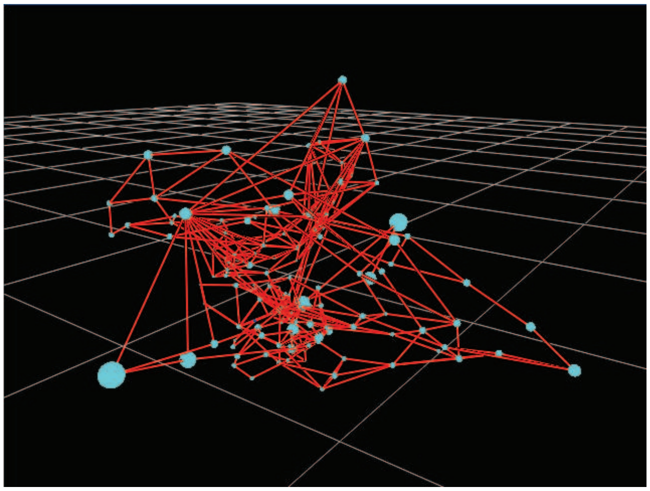

Fig. 6. Visualization of the database in the three principal axes space.

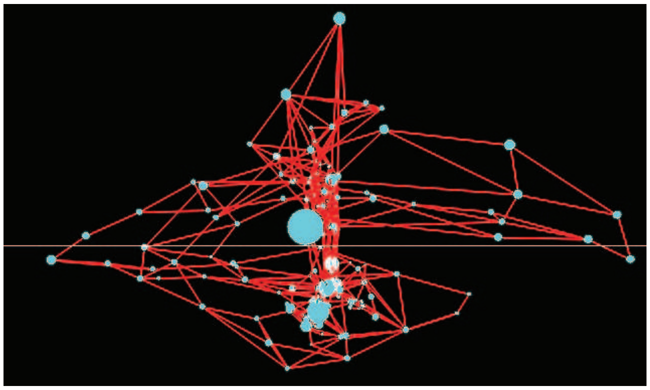

Fig. 8. Visualization of the database in the first (horizontal) and third (vertical) principal axes space.

C. State Estimation and Prediction

We experimented the state estimation ability by providing the first 0.2 s of a novel kick motion by the same subject and a jump motion by a different subject. The results are shown in Figures 11 and 12 respectively. Note that the global node transition is used to compute the best node sequence, although only the node transition in relevant sample clip is drawn in each figure for clarity.

In both cases, the database could predict the subjects' actions before they actually occurred. In Fig. 11, all nodes on the identified node sequence (drawn as yellow spheres) are included in the transition graph of the kick samples. The last node in the sequence is likely to transit to the dark blue node in the left figure with the probability of 0.36, which corresponds to the marker positions and velocities in the right figure. Similarly, the result of Fig. 12 indicates that the database can correctly identify that the subject is in preparation for a jump.

D. Motion Recognition

Figures 13–14 show the results of motion recognition experiments. The three graphs in Fig. 13 show differences between layers for the same observation. We computed the motion generation probability of new motions with respect to the node

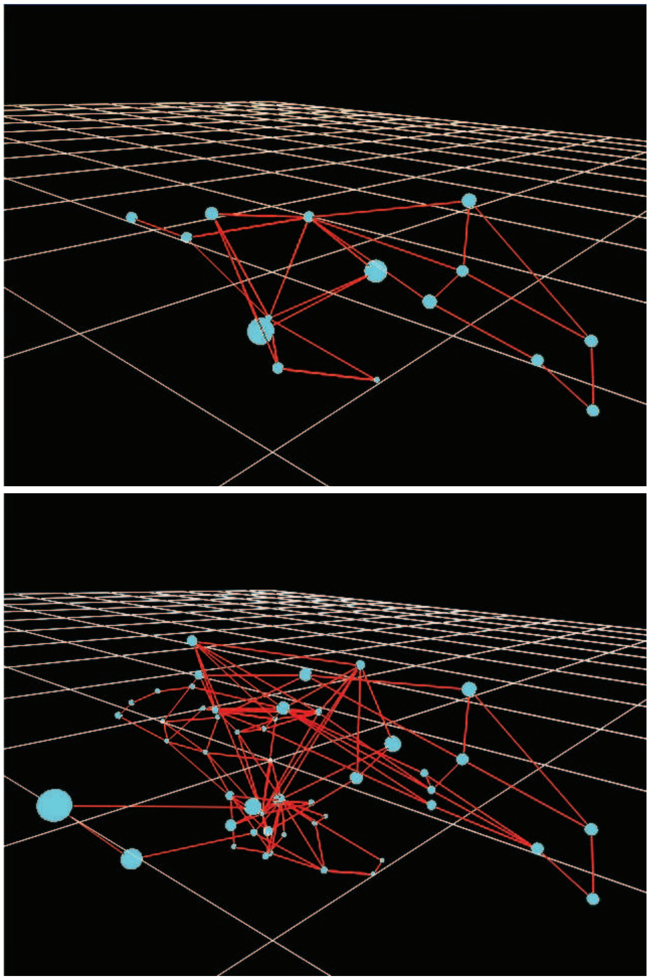

Fig. 9. Nodes and global transition graphs at layers 4 and 7.

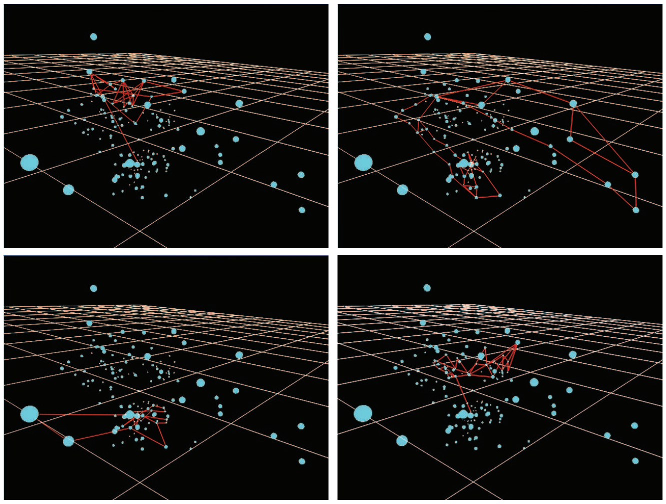

Fig. 10. Node transition graphs of selected clips. Top row: jog and jump, bottom row: kick and walk.

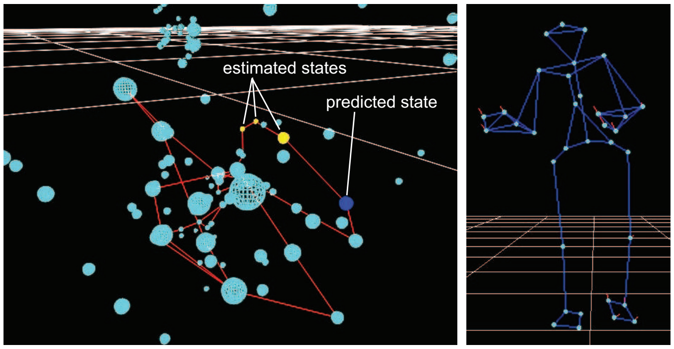

Fig. 11. Result of state estimation and prediction for a kick motion of the same subject. Left: the nodes on the identified node transition are drawn in yellow. Right: the marker positions and velocities corresponding to the dark blue node in the left figure.

transition graph of each sample motion clip. Because the new motion is 2.5 s long and computation of the probabilities takes a long time, we computed the probability of generating the sequences within a sliding window of width 0.4 s. The graphs depict the time evolution of probabilities when we moved the window from $[0.0\mathrm{s}, 0.4\mathrm{s}]$ to $[2.0\mathrm{s}, 2.4\mathrm{s}]$. The lines are colored according to the type of motions. We use same line color and type for very similar sample clips for clarity of presentation.

The first three graphs show the probability of a new walk motion by the same subject as the database, while a walk motion from a different subject was used for Fig. 14. These results show that the model can successfully recognize the type of observed motion even for different subjects. It is also suggested that the statistical models at different layers have different granularity levels because the variation of probabilities is smaller for upper layers. In particular, similar motions such as walk and jog begin to merge at layer 7.

Figure 15 show the result at the bottom layer for a completely unknown motion (lifting an object from the floor). It is clear that the database cannot tell whether the motion is jump or kick, which is intuitively reasonable because these three motions start from a small bending.

E. *Motion Planning*

We performed a simple motion planning test using the second database. Figure 16 shows the root trajectories in the planned motions where the goal position in both examples is 2.0 m front and 1.0 m left of the start position, and the goal headings are 0 rad (same as initial) in the first example and 1 rad in the second. The blue circle and green line represent the position and heading direction respectively every 1/6 s. The trajectories appear to be reasonable under the available samples.

We first search for a feasible node transition at the 13-th layer of the 16 layers in *Database 2* by randomly sampling node transitions based on the node transition graphs, until the root position and orientation come close to the given goal. We then find a node transition at the bottom layer by the algorithm described in Section IV-A. An inverse kinematics algorithm then converts the mean marker positions at each node to joint angles, which are interpolated to generate a

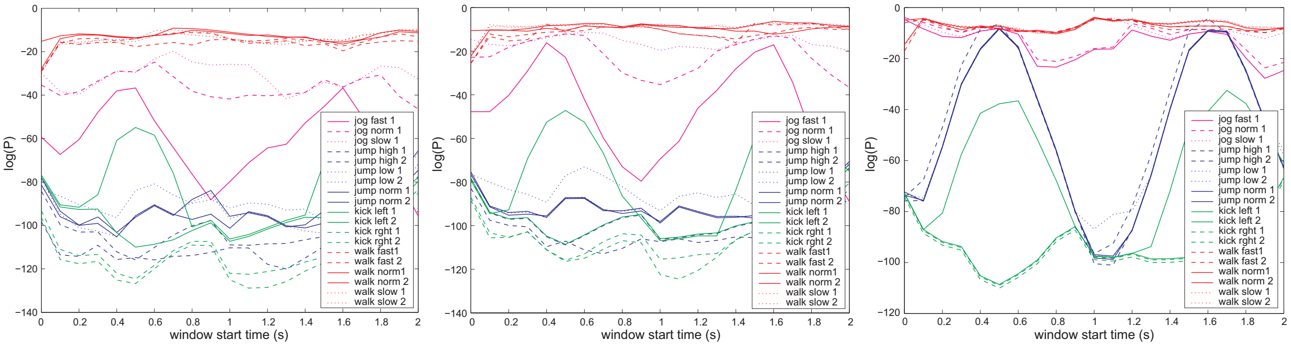

Fig. 13. Time profile of generation probability of a new walk motion of the same subject. From left to right: bottom layer, layer 10, layer 7.

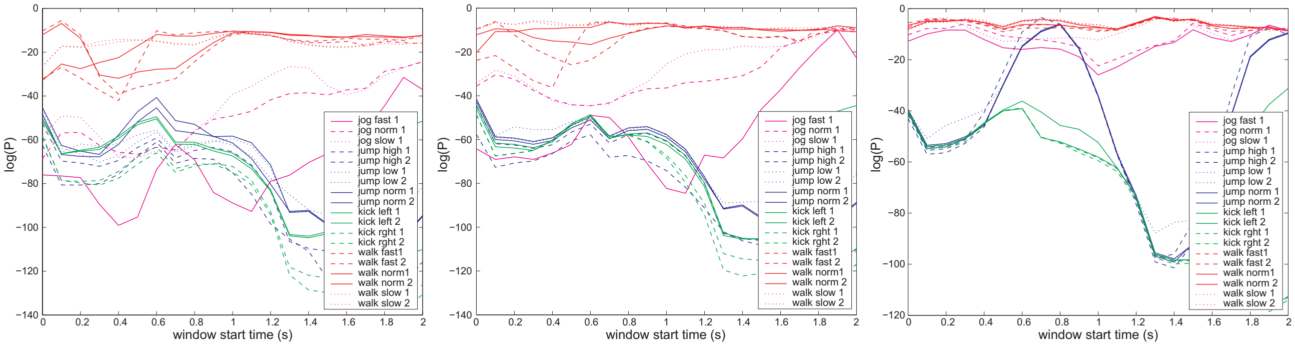

Fig. 14. Time profile of generation probability of a walk motion of a different subject. From left to right: bottom layer, layer 10, and layer 7.

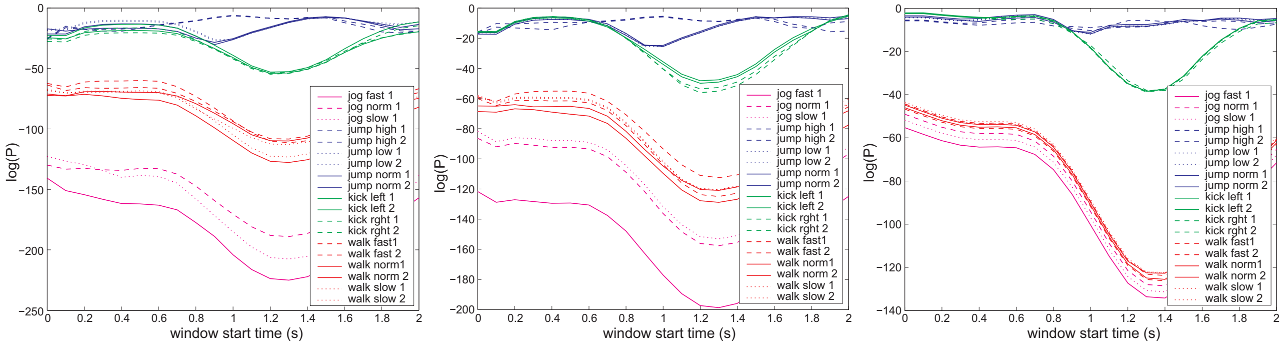

Fig. 15. Time profile of generation probability of a lifting motion. From left to right: bottom layer, layer 10, and layer 7.

whole-body animation.

The supplemental movie includes animations of the planned motions although the quality of the whole-body motions is not the main focus of this paper. Obviously, the motions are not smooth because the nodes represent multiple frames from the motion capture sequences. We also observe that the feet occasionally slip and penetrate the floor. These issues can be potentially solved by averaging over multiple planning results and performing additional inverse kinematics computation using contact status information from the original motion squences.

VII. Conclusion

In this paper, we proposed a new data structure for storing multi-dimensional, continuous human motion data, and demonstrated its basic functions through experiments. The main contributions of the paper are summarized as follows:

1) We proposed a method for constructing a binary tree data structure from human motion data. We applied PCA and a minimum-error thresholding technique to efficiently find the optimal clustering of sample frames.
2) We proposed to build a global node transition graph representing the node transitions in all sample clips, as well as a clip transition graph representing the node transitions in each sample motion clip.

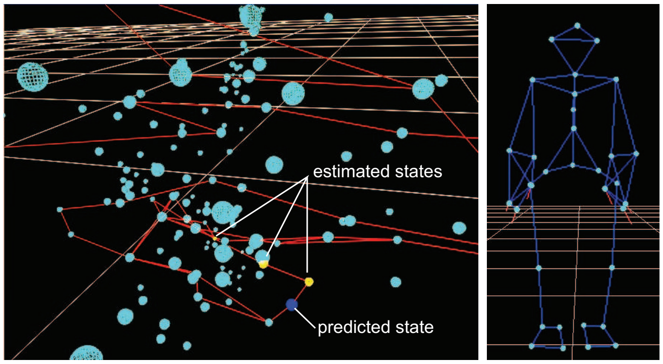

Fig. 12. Result of state estimation and prediction for a jump motion of a different subject. Left: the nodes on the identified node transition are drawn in yellow. Right: the marker positions and velocities corresponding to the dark blue node in the left figure.

3) We developed two algorithms for computing the most probable node transition and generation probability for a given new motion sequence, based on the binary tree data structure.
4) We demonstrated three applications of the proposed database through experiments using real human motion data.

There are several functions yet to be addressed by our database. We currently do not support incremental learning of new motions because all sample frames must be available to perform the PCA. If the new sample clips do not drastically change the distribution of whole samples, we could apply one of the extensions of PCA for online learning techniques [18]. Some techniques for balancing binary trees for multi-dimensional data could also be employed to reorganize the tree [19].

It should be relatively easy to add segmentation and clustering functions because the sample motion clips are abstracted by node transition graphs. We can easily detect same or similar node transitions in different motion clips, which could be used for segmentation. Clustering of sample clips can be achieved by evaluating the distance between motion clips based on their node transition graphs and applying standard clustering algorithms [20].

Our current database only contains the marker position and velocity data. It would be interesting to add other modalities such as contact status, contact forces, and muscle tensions. In particular, we would be able to solve the contact problem in our planning experiment if we have access to the contact status information. Contact force and muscle tension information would also help generating physically feasible motions for humanoid robots.

Acknowledgements

Part of this work was conducted while the first author was at University of Tokyo. The authors gratefully acknowledge the support by the Ministry of Education, Culture, Sports, Science and Technology, Japan through the Special Coordination Funds for Promoting Science and Technology, "IRT Foundation to Support Man and Aging Society."

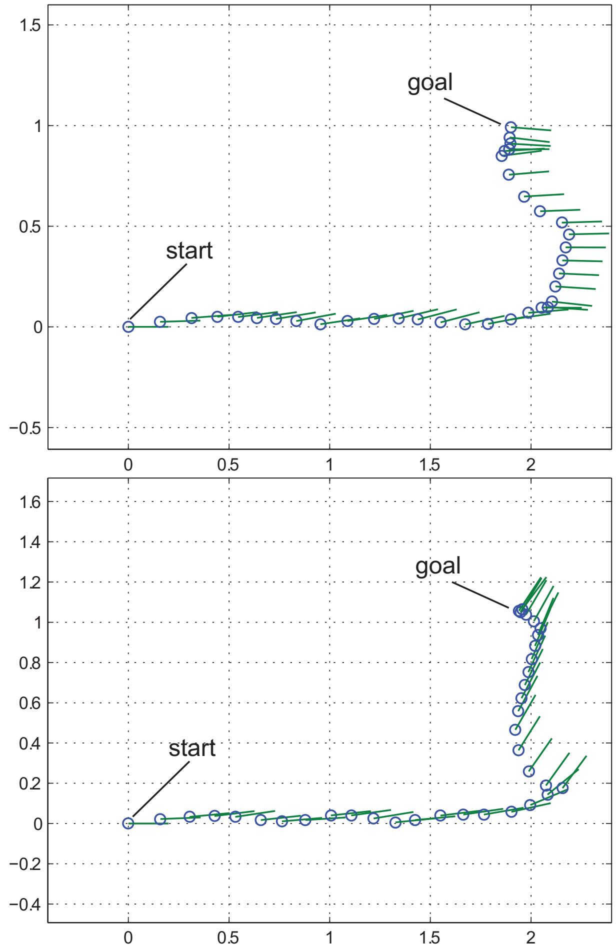

Fig. 16. The root trajectories in the motion planning results with goal position (2.0, 1.0). The goal headings are 0 in the top and 1 rad in the bottom figure. The blue circles and green lines denote the position and heading respectively.

References

[1] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 473–482, 2002.

[2] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard, "Interactive Control of Avatars Animated With Human Motion Data," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 491–500, July 2002.

[3] O. Arikan and D. A. Forsyth, "Synthesizing Constrained Motions from Examples," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 483–490, July 2002.

[4] A. Safonova and J. Hodgins, "Interpolated motion graphs with optimal search," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 106, 2007.

[5] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends in Cognitive Science*, vol. 6, no. 11, pp. 481–487, 2002.

[6] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Phylosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 358, pp. 537–547, 2003.

[7] A. Billard, S. Calinon, and F. Guenter, "Discriminative and adaptive imitation in uni-manual and bi-manual tasks," *Robotics and Autonomous Systems*, vol. 54, pp. 370–384, 2006.

[8] T. Inamura, I. Toshima, H. Tanie, and Y. Nakamura, "Embodied symbol emergence based on mimesis theory," *International Journal of Robotics Research*, vol. 24, no. 4/5, pp. 363–378, 2004.

[9] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings of International Conference on Robotics and Automtation*, 2002, pp. 1398–1403.

[10] H. Kadone and Y. Nakamura, "Symbolic memory for humanoid robots using hierarchical bifurcations of attractors in nonmonotonic neural networks," in *Proceedings of International Conference on Intelligent Robots and Systems*, 2005, pp. 2900–2905.

[11] D. Bentivegna, C. Atkeson, and G. Cheng, "Learning tasks from observation and practice," *Robotics and Autonomous Systems*, vol. 47, no. 2–3, pp. 163–169, 2004.

[12] D. Kulić, W. Takano, and Y. Nakamura, "Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains," *International Journal of Robotics Research*, vol. 27, no. 7, pp. 761–784, 2008.

[13] M. Brand and A. Hertzmann, "Style machines," in *Proceedings of SIGGRAPH 2000*, 2000, pp. 183–192.

[14] H. Sidenbladh, M. Black, and L. Sigal, "Implicit probabilistic models of human motion for synthesis and tracking," in *European Conference on Computer Vision*, 2002, pp. 784–800.

[15] J. Kittler and J. Illingworth, "Minimum error thresholding," *Pattern Recognition*, vol. 19, no. 1, pp. 41–47, 1986.

[16] Y. Nakamura and H. Hanafusa, "Inverse Kinematics Solutions with Singularity Robustness for Robot Manipulator Control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 108, pp. 163–171, 1986.

[17] "CMU graphics lab motion capture database," http://mocap.cs.cmu.edu/.

[18] M. Artač, M. Jogan, and A. Leonardis, "Incremental pca or on-line visual learning and recognition," in *Proceedings of the 16 th International Conference on Pattern Recognition*, 2002, pp. 30 781–30 784.

[19] V. K. Vaishnavi, "Multidimensional balanced binary trees," *IEEE Transactions on Computers*, vol. 38, no. 7, pp. 968–985, 1989.

[20] J. Ward, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, pp. 236–244, 1963.