Motion Synthesis and Editing in Low-Dimensional Spaces

Hyun Joon Shin

Div. of Digital Media, Ajou University,

San 5, Woncheon-dong, Yungtong-Ku

Suwon, Korea

Tel. (+82)31 219 1837 Fax. (+82)31 219 1797

email: joony@ajou.ac.kr

Jehee Lee

School of Computer Science and Engineering, Seoul National University

Seoul, Korea

email: jehee@cse.snu.ac.kr

Abstract

Human motion is difficult to create and manipulate because of the high dimensionality and spatiotemporal nature of human motion data. Recently, the use of large collections of captured motion data has added increased realism in character animation. In order to make the synthesis and analysis of motion data tractable, we present a low-dimensional motion space in which high-dimensional human motion can be effectively visualized, synthesized, edited, parameterized, and interpolated in both spatial and temporal domains. Our system allows users to create and edit the motion of animated characters in several ways: The user can sketch and edit a curve on low-dimensional motion space, directly manipulate the character's pose in three-dimensional object space, or specify key poses to create in-between motions.

Keywords: Multi dimensional scaling, Character animation, User interface

Introduction

Creating animated characters that move realistically is an important problem in computer graphics. One appealing approach is to collect a large amount of human motion data and analyze those data for constructing a behavior model of animated characters. We expect that this model not only contains many primitive actions available to the characters but also is able to provide assorted variants of each primitive action in a parametric form. It is also expected that a set of actions are provided in a connected way so that transitions from one action to another are allowed. Constructing such a behavior model is quite difficult especially if we want to make use of relatively unstructured motion data for behavior generation.

With a relatively small collection of motion data, it is possible to look through the entire motion set thoroughly to manually construct a connected set of character behaviors. Through the manual process, one can have an in-depth understanding of what behaviors are available at any given situation and how behaviors are organized. However, this is unrealistic in practical applications because the input motion set must be large enough to accommodate a rich variety of natural human motion. Recently developed methods are able to analyze a large motion set automatically to identify a connected set of distinctive behaviors and even parameterize variants of each behavior. The resulting data structure can be searched in order to create a sequence of motions that allows animated characters to track a target or travel along a sketched path. However, this data structure may be too complex for animators to understand how behaviors are organized. From the animator's point of view, it is very important to have direct and immediate control over the character's motion. Animators want to be able to select a set of appropriate behaviors at a given situation and to produce a carefully-crafted motion of the character interactively by precisely adjusting parameters of the behaviors.

In this paper, we show that a rich, connected set of human behaviors can be projected onto a low-dimensional space (mostly two-dimensional space in our experiments) so that its structure can be effectively visualized and exploited for interactive motion synthesis and editing. The motion data is preprocessed to construct a motion graph, which is then projected onto a low-dimensional space. The two-dimensional projection explicitly depicts the connectivity of the motion graph and also provides a parametrization among similar actions. This parametrization allows any path in the low-dimensional space sufficiently close to the projected graph to be mapped to a motion of the character.

Even with state-of-the-art dimension reduction methods, projecting large high-dimensional motion data onto a drastically low-dimensional space is still difficult particularly when the intrinsic dimension of the data is higher than the dimension of the target space. Instead of projecting the entire motion sets onto a single space, our system provides an ability to choose a local neighborhood from input motion sets and generate its low-dimensional projection on the fly while a new motion is created and edited by users.

We demonstrate the power of our approach in the context of several applications:

- **Motion creation and editing.** The user is allowed to create a new motion by sketching a path on two-dimensional motion space.

- **Direct manipulation.** Our system allows the user to edit the motion in three-dimensional space through direct manipulation.
- **Data-driven keyframing.** The user specifies a sequence of key poses, which are interpolated by natural in-between motions generated from input motion data.

Background

Understanding and representing the intrinsic structure of human motion has been an active research topic in a wide range of disciplines. The solutions of these problems have often relied on manual processing because they are extremely capable of identifying and recognizing distinctive human movements. Badler et al. [1] suggested representing human behavior by a directed graph of which nodes correspond to full-body poses and edges correspond to manually-crafted motion segments. This graph exhibits connections in a repertoire of given motion data. Similar graph structures have widely been used in game industry. A number of researchers have developed automatic and semi-automatic methods to create a graph-based representation of motion from unstructured, unlabeled motion sequences [2, 3, 4, 5].

A number of previous efforts have established the technique of building parameterized motions from blends of motion examples [6, 7]. Statistical models have frequently been used to capture the underlying structure of a large collection of motion data. A typical approach is to exploit PCA to simplify the data, cluster for identifying similar motions, and use a

Markov process model to allow transitions between clusters [8, 9, 10].

Dimensionality reduction allows us to visualize, categorize, or simplify large data sets. Several techniques for dimension reduction have been exercised in the context of human motion analysis for capturing the intrinsic dimensionality of the desired behavior and efficient computation. Jenkins and Mataric [11] generalized a well-known dimension reduction technique, called *Isomap*, so that it can account for temporal coherency among motion frames. Safonova et al. [12] constructed a low-dimensional space of human motion via PCA and showed that an articulated figure motion can effectively be optimized in the low-dimensional space to satisfy user-specified constraints. Chai and Hodgins [13] built an animation system that uses low-dimensional control signals from the user's performance to control the motion of animated characters. Assa et al. [14] employed a dimensionality reduction technique for selecting keyframes from motion data. Grochow et al. [15] provided a method to solve inverse kinematics (IK) problems in a low dimensional space.

Sketching has widely been investigated for motion synthesis and steering interactive characters. Given a sketched path on the ground, a number of researchers explored many different ways to allow characters to move along the path [3, 4]. The capability of sketch interfaces has not been limited to walking trajectories, but can specify many more details. The sketch interface of Davis et al. [16] allows animator to sketch 2D poses at keyframes, then infers 3D poses from the sketches, and finally interpolates the reconstructed 3D poses to produce 3D character animation. Li et al. [17] presented a similar idea to exploit 2D drawings for stylizing 3D shapes and animation. Thorne et al. [18] developed a sketch-based user interface that translates a sequence of cursive strokes to character animation.

Low-Dimensional Motion Space

Dimensionality Reduction

We collected five sets of human motion data using an optical motion capture system. Each motion set includes many similar motions that form a parameterized family of motions. Through dimension reduction, we expect that those motions would form a stream of parallel curves in a low dimensional space and those curves would be well separated from each other (rather than collapsed to a single curve) so they are parameterized along orthogonal axes.

To select an appropriate dimension reduction method, we examined three well-known methods with our motion data: PCA, multi-dimensional scaling (MDS), and Isomap. The distance metric proposed by Kovar et al [3] was adopted to compute the dissimilarity between motion frames. In our experiments, we observed that PCA parameterizes motion data in a quite different way from MDS and Isomap do and its parametrization often fails to reveal the nonlinear structure of motion data. In many cases, MDS and Isomap are comparable with respect to our requirements. In most of our experimental results, MDS was used because MDS is simpler and more efficient than Isomap.

Local Projection and Transition

Existing dimension reduction methods work well with a small set of motion data. However, those methods do not scale easily to large motion sets (see Figure 2). It is probably because the intrinsic dimension of large data sets is higher than that of the target 2D space. We address this problem by allowing the user to select a small portion of data interactively and project that portion on-the-fly using an existing method. It is like using a magnifying glass to closely examine a specific portion of data sets. In this way, the structure of motion data can be clearly presented in an uncluttered display. The user is allowed to select a seed frame and the length of time he/she want to look forward and backward from the seed frame. To identify its temporal neighbors efficiently, we construct a neighborhood graph at the preprocessing phase. In the neighborhood graph, each node corresponds to a motion frame and maintains the connections to nearby nodes within a user-specified distance threshold.

The projection by aforementioned dimension reduction methods is in general non-linear. Our magnifying glass is actually showing a deformed image of motion data from a certain viewpoint. An appropriate viewpoint is automatically selected for visualizing the structure of motion data (see Figure 2). As the user shifts the magnifying glass by selecting a different seed frame, the projected data on the display should be shifted accordingly. This shifting function is also non-linear and computed as follows. Suppose that frames $\{F_0, \cdots, F_n\}$ are currently selected and displayed in the screen and the 2D screen coordinates $\{p_0, \cdots, p_n\}$ of the frames are computed through dimension reduction. Shifting the magnifying glass will select a new overlapping set of frames $\{F_k, \cdots, F_m\}$, where $0 < k < n$ and $n < m$, for display and their 2D coordinates $\{p'_k, \cdots, p'_m\}$ can also be computed through dimension reduction. Note that the common frames $\{F_k, \cdots, F_n\}$, in general, have different coordinates before and after. We compute a smooth shifting function such that it aligns $\{p_k, \cdots, p_n\}$ with $\{p'_k, \cdots, p'_n\}$ (see Figure 3). Our shifting function first translates and rotates the former set to align with the latter set as closely as possible and then uses Radial Basis Function (RBF) interpolation for precise matching. The amount of translation and rotation is determined by solving a least squares problem, which allows an analytic solution.

Motion Reconstruction

Given motion data projected onto low-dimensional space, sketching is an intuitive, versatile way of specifying desired motions. Though our magnifying glass interface shows only a small portion of data in a single screen, the user can specify a long path through the entire data by repeatedly sketching and shifting the viewpoint (see Figure 4). Converting a two-dimensional sketched curve to a much higher-dimensional articulated figure motion is a major challenge of our work. Our basic idea for synthesizing a motion corresponding to the curve is to blend motion streams nearby in the low dimensional space. That is, any point on the curve can be represented as an affine combination of projections of original motion frames. The same weights are used to determine a full-body pose at that point by blending original motion frames. We first show how a pose is constructed at a point on the curve and the corresponding tangent, and then address issues arise while constructing the motion.

A pose from a point. To produce quality motion, the motions to be blended must be aligned temporally. Roughly speaking, in the low dimensional space, the axis parallel to a motion stream corresponds to poses changing over time, while the axes perpendicular to the curve correspond to a change within a parameterized space of motions at a fixed time instance.

Given a point on a sketched curve, we select two surrounding samples from the original motion streams such that the point on the curve is represented as a weighted sum of two samples. All points on the parametrization axes except the ones heading for directions opposite to the curve tangent can be blended to produce a quality motion. In two-dimensional space, we choose two nearest surrounding samples among the intersections between the line perpendicular to the tangent at the point and the motion curves. The blending weights for the samples are inversely proportional to distance in two-dimensional space. If only one similarly directed stream intersects the line, the corresponding pose to the intersection is selected without blending since there is no similar motion to blend. If no surrounding streams along the parametrization axes exist within a user-specified range, the point on the sketched-curve is *invalid*. In our implementation, the system automatically detects invalid points on the curve and omits the points in motion synthesis.

A motion from a curve. To produce a motion rather than a sequence of poses, we have to decide the time parameter along the curve. The time parameter at any point on the curve is determined by integrating the ratio $\frac{\Delta t}{||\Delta\mathbf{x}||}$ of time increment to the distance traveled along the curve. This time/space ratio at a point is computed by interpolating the ratio of surrounding motion streams with the weights determined above. Integrating this ratio along the curve allows us to sample motion frames on the curve at desired rates (typically 30 fps).

The motion obtained through pose blending above may have unwanted high frequency components due to jitter of the curve or sudden changes of surrounding streams. Instead of blending each pair of poses separately, we create a short motion clip at each frame on the curve by blending consecutive frames selected from two matching streams (see Figure 5). This generates a sequence of motion clips overlapping front and back. At each time instance, all overlapping clips are convoluted with a Gaussian kernel to produce the final pose.

Motion Synthesis and Editing

As far, we discussed how a character motion is reconstructed from a curve drawn on the two-dimensional space. Motion synthesis and editing often require the inverse mapping that projects a novel pose or motion onto the two-dimensional space. To compute this mapping efficiently, we sample the low dimensional space using a grid of locations and quantized tangent directions. For two-dimensional space, we typically quantize tangent directions into eight directions. Each sample on the grid is indexed by 2D location (x,y) and tangent direction θ. A sample is valid if it has two surrounding motion streams along the normal direction. The pose at each valid sample is computed by the method described in the previous section. This 3D grid forms a directed graph, which has an edge from one valid node (x_1, y_1, θ_1) to another (x_2, y_2, θ_2) if two nodes are adjacent and aligned along direction θ_1.

Data-driven IK. The graph of 3D sample nodes provides us with capability to solve an inverse kinematics problem by blending existing example data. Given a user-specified constraint, we first find the node on the grid of which corresponding pose matches the constraint best. Typically, the constraint specifies the location of an end-effector or the root segment. The pose is further refined by linearly interpolating the neighboring samples assuming that the pose space and the low dimensional space are locally linearized through dense sampling.

Direct manipulation. Once a motion is created, the user can edit the motion either by tweaking the curve in two-dimensional space or by directly manipulating the motion in the pose space. For direct manipulation, the user can select any frame of the motion and change the pose of the character at that frame by dragging a body segment. Then, our system makes a smooth change to the low-dimensional motion curve such that the curve passes through the corresponding point in the low-dimensional space, thus the user-specified pose is interpolated in the motion. In this case, the distance from the initial point to the target point is also taken into account while choosing the best matching sample to limit unwanted large deformation of the curve.

Data-driven keyframing. Given start and end poses, an appropriate in-between motion can be found by planning a path through the 2D projection of motion data. We first find start and end nodes that correspond to start and end poses, respectively. The octilinear path between start and end nodes is found with A^*-search algorithm. We approximate this octilinear path with a smooth curve to generate the desired in-between motion exploiting low-pass filtering.

Experimental Results

Walk. We captured our subject walking in various steering angles for about 20 seconds. Each cycle of walk is projected into a circle in two dimensional space (see Figure 6). At a point on the circle, the tangential direction corresponds to time and its orthogonal (normal) direction parameterizes steering angles.

Walk and sneak. To collect motion data, our subject walked and sneaked around in an environment for 20 seconds(see Figure 2). The projection of the entire motion data do not show the cyclic nature of locomotion and the parametrization along the normal direction cannot capture the variation of motions. Each individual projection of walk and sneak provides better visualization and intuitive parameters for creating quality blends of motions.

Motion editing. To collect walk-and-pick data, our subject walked around and picked balls at various heights. As shown in Figure 7, we can pick any point on the 2D curve and drag the point to edit the curve and its corresponding motion. Editing the curve allows us to create a continuous span of picking motions with the hand aiming different heights.

For direct manipulation, we sampled 2D space with a regular 20×20 grid. As the user drags the right hand of the character in the pose space, the characters's pose is projected into one of the grid points (blue dots in Figure 7) and the shape of the curve is modified accordingly.

Keyframing. The data-driven keyframing capability of our system was demonstrated with two sets of motion data: walk-and-pick and boxing. In the walk-and-pick example, we use a regular 20×20 grid (see Figure 8 (Top)). In the start pose, the character was swing the right leg in a walk cycle. In the end pose, the character was reaching the right hand forward. Our algorithm found a natural in-between motion in which the character took a short step forward, stopped, and reached the right hand forward.

Our boxing data was about one minute long and included a variety of actions such as punching, dodging, ducking, and blocking punches. The boxing example required a denser 40×40 grid because the boxing data has a narrow passage in the valid region.

Discussion

Our approach scales relatively well with the size of motion data due to the local projection strategy. There is one issue to be addressed for achieving better scalability. Both MDS and Isomap requires $O(n^2)$ memory space for maintaining the all-frames-to-frames dissimilarities, where n is the number of frames in the database. We observed that motion data tends to be partitioned into sets of coherently parameterizable motions and narrow connections between them. We envision the use of graph cut techniques to partition motion data automatically. With this partitioning, the dissimilarity matrix could be stored compactly.

Acknowledgements

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment). This research work was partly supported by the Young investigator's award grant from the KOSEF.

References

[1] N. I. Badler, R. Bindiganavale, J. P. Granieri, S. Wei, and X. Zhao. Posture interpolation with collision avoidance. In *Proceedings of Computer Animation '94*, pages 13–20, 1994.

[2] Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, 2002.

[3] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, 2002.

[4] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, July 2002.

[5] Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen. Snap-together motion: assembling run-time animations. In *Proceedings of 2003 ACM Symposium on Interactive 3D Graphics*, pages 181–188, April 2003.

[6] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics*, 23(3):559–568, 2004.

[7] Tomohiko Mukai and Shigeru Kuriyama. Geostatistical motion interpolation. *ACM Transactions on Graphics*, 24(3):1062–1070, 2005.

[8] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of SIGGRAPH 2000*, pages 183–192, July 2000.

[9] Tae-hoon Kim, Sang Il Park, and Sung Yong Shin. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics*, 22(3):392–401, 2003.

[10] Eugene Hsu, Kari Pulli, and Jovan Popovic. Style translation for human motion. *ACM Transactions on Graphics*, 24(3):1082–1089, 2005.

[11] O. C. Jenkins and M. J. Mataric. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the International Conference on Machine Learning*, pages 225–232, 2004.

[12] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics*, 23(3):514–521, 2004.

[13] Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics*, 24(3):686–696, 2005.

[14] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. Action synopsis: Pose selection and illustration. *ACM Transactions on Graphics*, 24(3):667–676, 2005.

[15] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522–531, 2004.

[16] James Davis, Maneesh Agrawala, Erika Chuang, Zoran Popović, and David Salesin. A sketching interface for articulated figure animation. In *Proceedings of 2003 ACM SIGGRAPH / Eurographics Symposium on Computer animation*, pages 320–328, 2003.

[17] Yin Li, Michael Gleicher, Ying-Qing Xu, and Heung-Yeung Shum. Stylizing motion with drawings. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 309–319, 2003.

[18] Matthew Thorne, David Burke, and Michiel van de Panne. Motion doodles: an interface for sketching character motion. *ACM Transactions on Graphics*, 23(3):424–431, 2004.

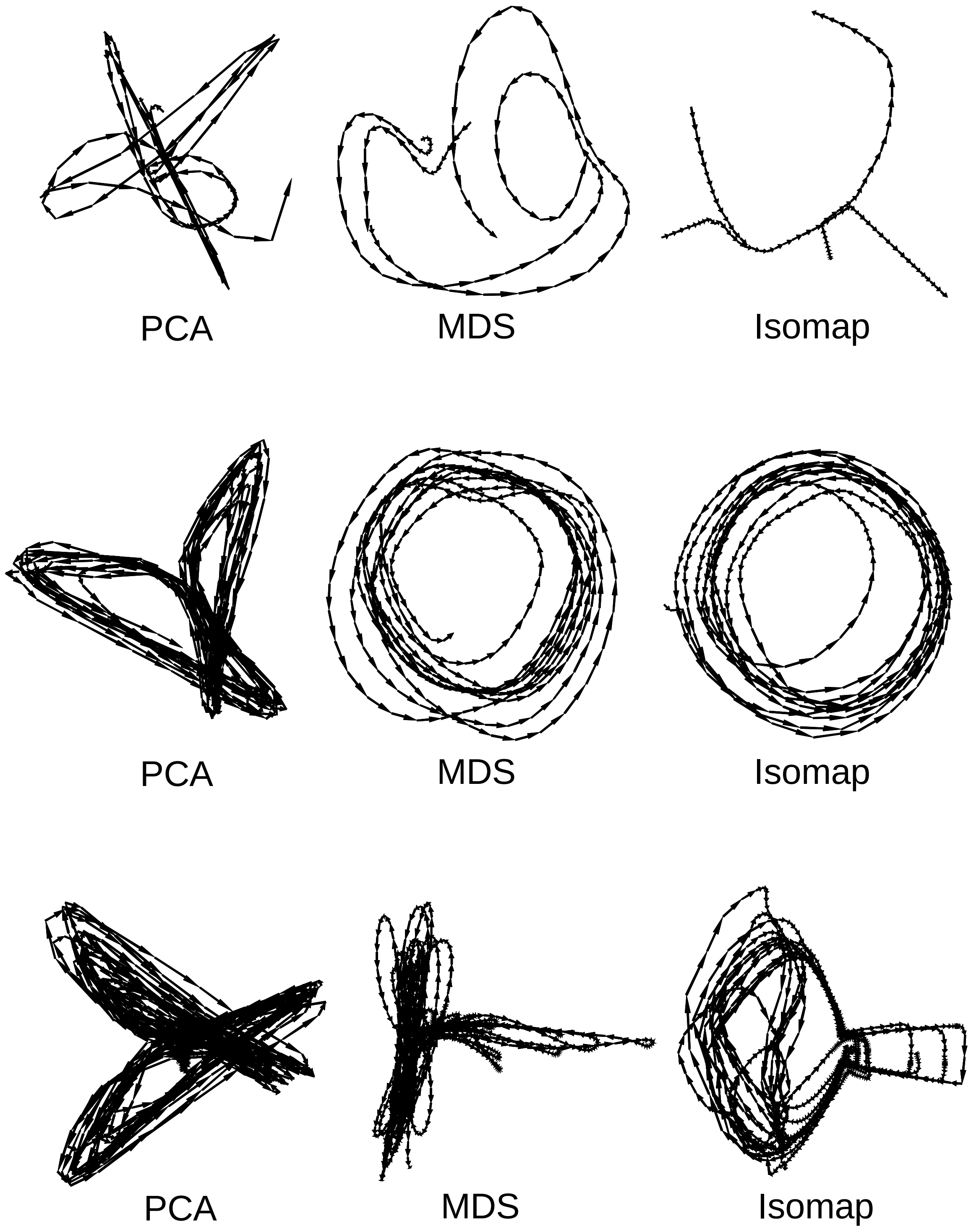

Figure 1: Dimensionality reduction of motion data: A boxing motion (top row), walking motion (middle row), and walking and picking motion (bottom row) are projected on two-dimensional space by PCA (left), MDS (middle), and Isomap (right).

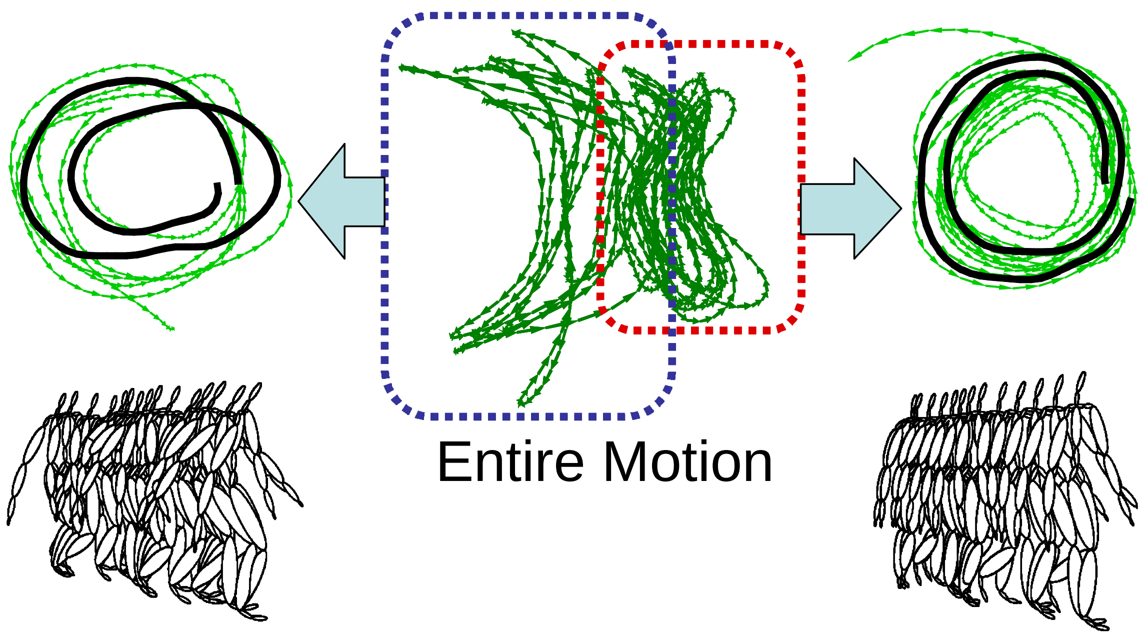

Figure 2: The projection of the entire motion including walk and sneak cannot show the structure of motion data (middle). Projecting a portion of motion data selects a different viewpoint to uncover the cyclic nature of sneak and walk (left and right).

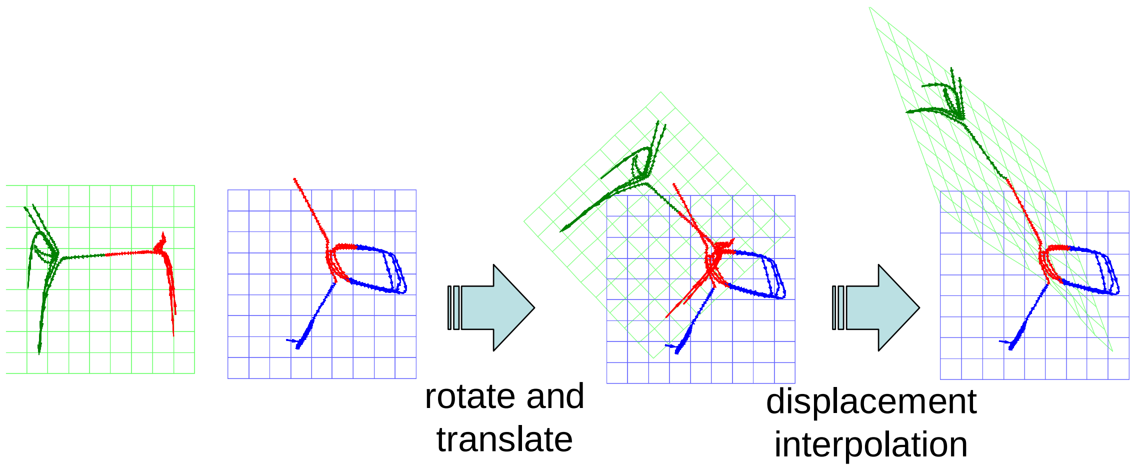

Figure 3: Aligning 2D point sets projected from different viewpoints. Overlapping (red) points are created from the same set of motion frames. The shifting function from the green set to the blue set is a composition of two transformations. The first transformation rotates and translates the green set to align the overlapping points as closely as possible. The second transformation is a non-linear transform represented by radial basis functions. Using RBF interpolation, the coordinates of every pair of overlapping points can be matched precisely and the other points can be transformed accordingly.

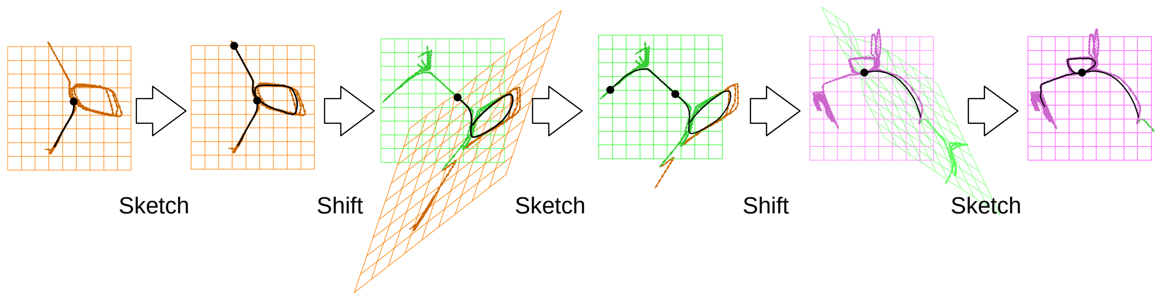

Figure 4: Sketching a long path needs for a user to shift the viewpoint. Local dimensionality reduction selects an appropriate viewpoint for visualizing the structure of motion data.

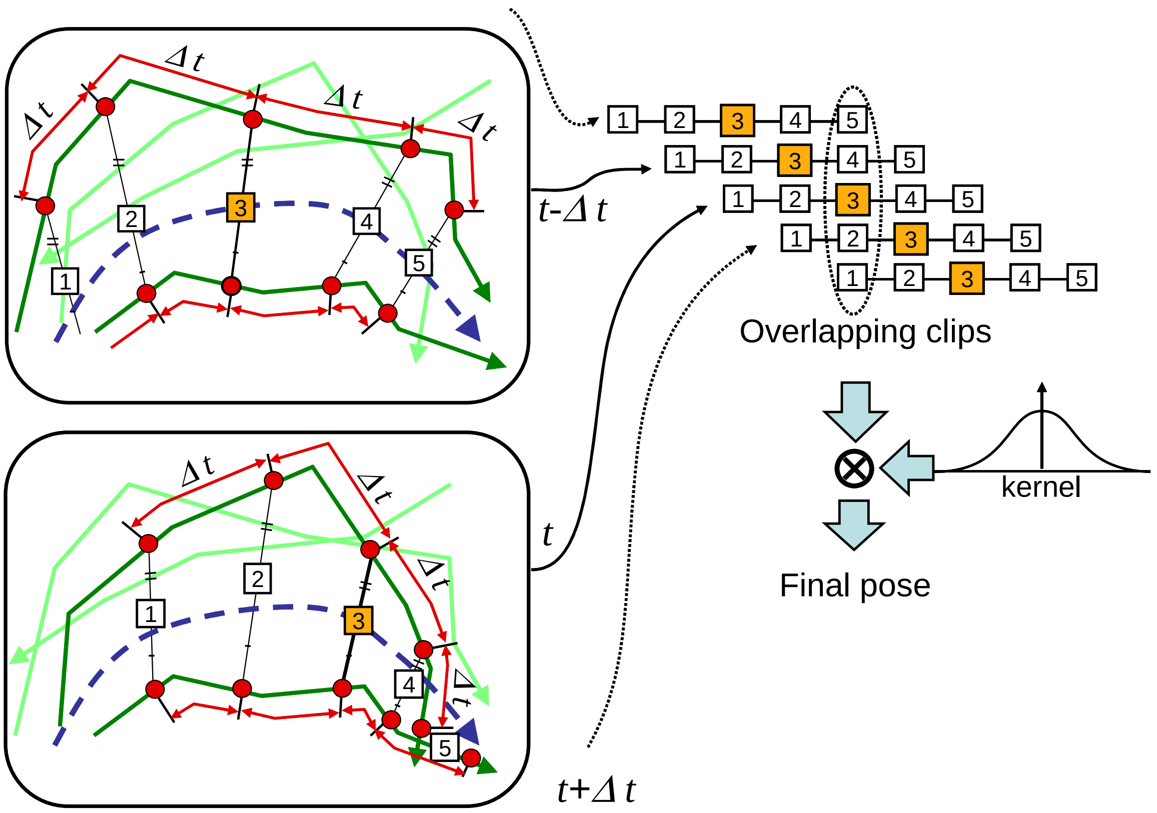

Figure 5: Motion reconstruction. Temporally-overlapping motion clips are generated along the curve. The corresponding poses from the overlapping clips are convolved with a smoothing kernel to produce the final pose.

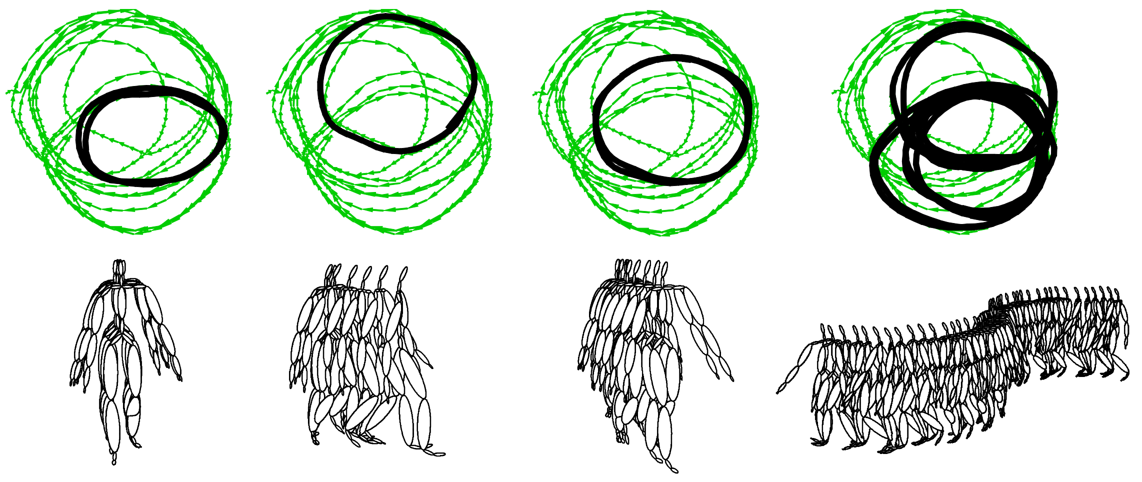

Figure 6: Walking in various steering angles. (Left to right) straight walk, turn left, a blend of straight walk and turn, and a random walk

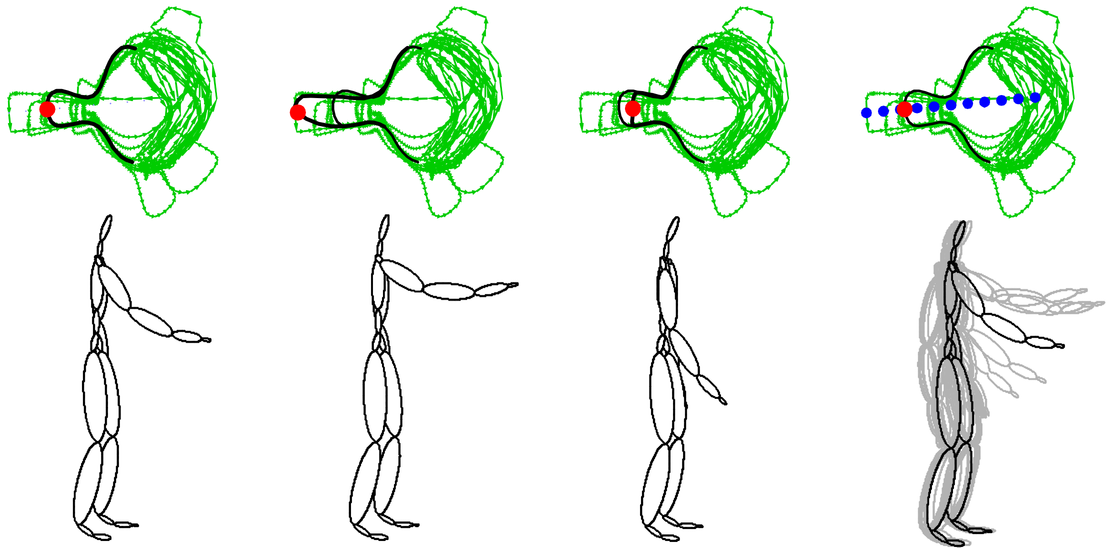

Figure 7: Motion editing. (Left) The original motion created through our sketch interface. (Center) The user tweaked the 2D curve by interactively dragging the red point in two-dimensional space. (Right) The user directly manipulated the motion in three-dimensional space by dragging the right hand of the character.

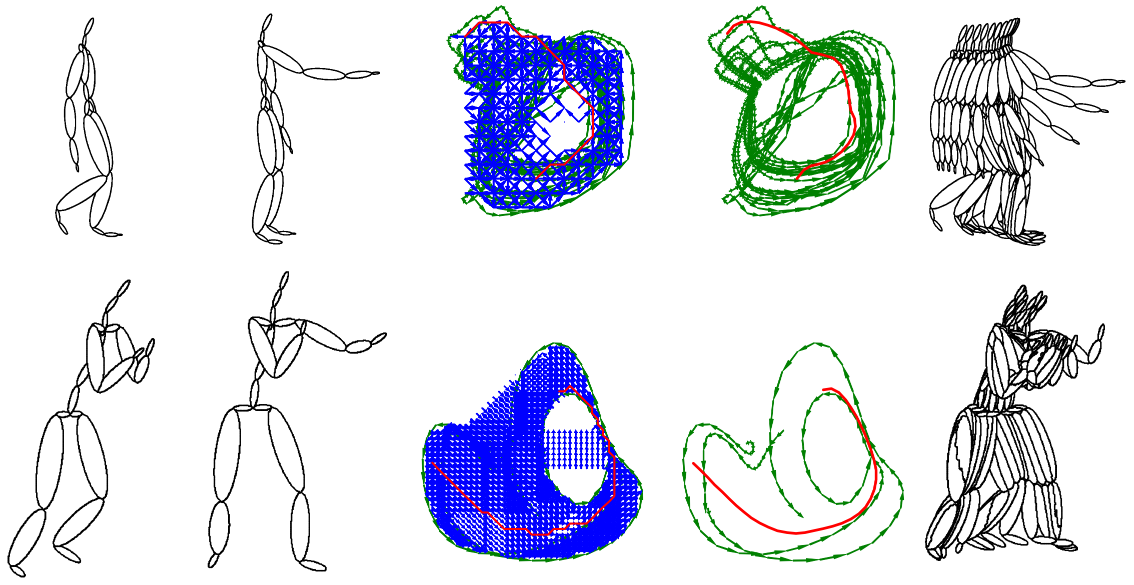

Figure 8: Data-driven keyframing using data sets of walk-and-pick (top) and boxing (bottom). (From left to right) A start pose, an end pose, a grid of valid locations and tangent directions, a searched path after smoothing, and a final in-between motion.