Hybrid animation with spatial keyframes and motion capture

Bernardo Fortunato Costa, Claudio Esperança
Laboratório de Computação Gráfica/PESC
COPPE/UFRJ
Rio de Janeiro, Brazil
Email: {bfcosta,esperanc}@cos.ufrj.br

***Abstract*—We propose a hybrid scheme for authoring character animations where key poses are collected from motion capture data and projected on a plane using dimension reduction techniques. Animations can then be created by drawing a curved trajectory on that plane, where each point along the path corresponds to a new pose interpolated from neighboring projected poses using Igarashi's spatial keyframing approach[1]. Furthermore, timing information contained in motion capture timestamps are also projected on the plane, so that they can guide the trajectory parametrization. By allowing the animator to edit the curve path and also its temporal parametrization, we expect the system to produce natural animations including the original motion capture animations, but not restricted to them.**

***Keywords*-character animation; motion capture; keyframing; spatial keyframe;**

I. Introduction

The traditional technique known as *keyframing* is a popular way to produce animations. When used in computer-assisted character animation, it consists of specifying the character pose at certain time instants and then using interpolation functions to obtain in-between poses. Since poses are usually specified with the help of rigid articulated skeletons, computing in-between poses is tantamount to interpolating joint rotations between successive keyframes. The careful specification of the poses and interpolation functions, together with correct timing of a character's actions is essential to the final quality of the result. It requires time and effort from skilled animators to produce good animations.

On the other hand, the technique known as *motion capture* or *"mocap"* produces animations by recording actions of an actor by means of a set of cameras. This data is translated into skeleton poses to animate a character in a virtual environment. This produces a wealth of detail, both in poses and timing, thus producing realistic character animations. Although the acquisition process is expensive, there is a large amount of data available for reuse. Unfortunately, unless one is trying to reproduce a motion which is very similar to some originally recorded one, mocap data reuse is still a challenge.

The aim of the research discussed in the following sections is to build a system that allows reusing mocap data in the context of a keyframing application. Rather than using temporal keyframes, however, we propose adopting the *spatial keyframes* of Igarashi et al. [1], where key poses are associated with points on a 2D plane. The animation itself is produced in real time by moving a controller cursor on the plane, thus describing a curved trajectory, where each point on the curve corresponds to a pose interpolated from nearby key poses. That system was originally proposed for creating quick and non-professional animations, typically using no more than a few key poses manually specified. The scheme, however, can be extended by selecting key poses from mocap data and projecting them onto the spatial keyframing plane using dimension reduction techniques. Moreover, the original mocap animation will, in this case, be equivalent to a trajectory which can be edited by the animation, thus yielding significantly different motions.

In the following sections, we review related work, state our project goals, describe some achievements and point to further investigation paths.

II. Related work

Since motion capture data started to become available for reuse, several ideas have been proposed to make an alternative use of this data, rather than simply reproduce it in the same context it was originally built. Kovar et al. [2] had the idea of breaking the data into smaller units which could be connected, given that there is a plausible transition between them. A directed graph is built and the animation is produced by traversing this graph and applying an algorithm to smooth transitions between nodes. They achieve good animation quality but the sketching space available to the animator is limited to the existing database clips.

Motion capture databases can be quite large and this also has led to research focusing on data compaction. Safonova et al. [3] show that motion capture data has much correlation between its degrees of freedom, making it possible to compress it by employing principal components analysis. Another strategy is to discard some frames from the database, replacing them with frames reconstructed from the remaining keyframes by means of blending functions. Xiao et al. [4] use a strategy based on curve simplification to choose keyframes. This is an iterative strategy aimed at finding the most "distant" frame in the motion, compared to the reconstructed frames, and then adding it to the set of keyframes. Huang et al. [5] use a strategy called matrix factorization. Here, character poses are described as vectors and put all together in a matrix representing the full set of frames. The keyframe extraction problem is posed as

a constrained matrix factorization problem to be solved with least-squares optimization. Halit and Capin [6] implemented another strategy called clustering. In this case, character poses are grouped in clusters of poses according to some definition of distance, and a representative pose is elected for the group. Zhang et al. [7] use genetic algorithms to find keyframes in motion databases. A search through the solution space is done by combining previous solutions, giving priority to best fits.

The keyframe animation workflow consists, in essence, of defining poses for given timestamps and using interpolation functions to synthesize poses for the remaining time instants. This process can be quite laborious, especially for informal animations. Igarashi et al. [1] proposed an interface and technique for authoring these, called spatial keyframing. Here, the user records poses of an articulated virtual character, associating each pose to a marker on the screen, typically a visible point. The animation is built in real time by moving a controller, another visible point, between the markers. The controller trajectory creates the animation by blending the recorded poses, using radial basis functions (RBFs) [8], such that the pose for a given point is more heavily influenced by the closest pose markers.

The use of motion capture data in keyframe based animation is not spread out due to the difficulty of finding ways to make both workflows work together to produce good animation. Pullen and Bregler [9] use mocap data to enhance parts of a keyframed animation, by matching the frequency analysis of the keyframed and motion capture data. Their goal is similar to ours but their work is aimed at traditional temporal keyframing.

III. Reusing mocap data with spatial keyframing

As stated earlier, we propose to reuse poses extracted from mocap data and set them as keyframe markers within the spatial keyframing pose plane. This requires addressing two important problems, namely, how to choose a small but comprehensive set of poses, and how to lay them out on the pose plane.

A. Pose extraction

Motion capture pose extraction is a well known problem. It considers motion as a function with time as the domain and poses as the range. In order to extract the most relevant poses and omitting redundant poses, most schemes define some metric whereby the dissimilarity between two poses can be evaluated in the form of a distance function. Two main groups of distance functions have been proposed. One group defines pose distance by measuring degrees of freedom in a positional format while the other defines pose distance mostly as a measure of rotational variance. Let A and B be two poses, consisting of J joints (degrees of freedom). Let $P(\cdot)$ be a function that maps a joint to its position, and $Q(\cdot)$ a function that maps a joint to its rotation as a quaternion. Then, equations (1) and (2) show two typical positional and rotational definitions of pose distance functions, respectively. Weights w_j are used to assign importance to individual joints, typically by assuming that limb extremities should have more importance than joints closer to the skeleton's center of mass.

$$E_p(A,B) = \sum_{j \in J} w_j \|P(A_j) - P(B_j)\|^2 \tag{1}$$

$$\begin{aligned} E_r(A,B) = \; & w_0 \|P(A_0) - P(B_0)\|^2 \\ & + \sum_{j \in J} w_j \left(1 - \langle Q(A_j), Q(B_j) \rangle^2\right) \end{aligned} \tag{2}$$

Pose distance functions can be used as error-measuring devices. Since pose extraction consists of selecting a set of frames that are used to reconstruct the missing ones, it stands to reason that a good selection minimizes the error between synthesized motion and the original captured motion. At any rate, the evaluation of the pose extraction is usually conducted with the aid of such functions.

Only positional and rotational data might not be able to fully characterize the motion. Dynamic information such as velocity or acceleration is used by some authors to better choose keyframes in a motion frame set. Assa et al. [10] builds a framework which takes into consideration the spatial distance in both formats, together with velocity distance also taken in positional and rotational format. Bulut and Capin [11], Halit and Capin [6] and Jin et al. [12] use the motion curvature for choosing keyframes. Motion curvature is guessed from velocity and acceleration.

The strategy for choosing keyframes set also plays a role in our context. Two promising strategies are curve simplification [4] and clustering [6]. Curve simplification gives us the ability of customizing the keyframe set size. But some adaptation has to be considered as all of the reviewed algorithms assume a canonical motion curve. This means that whereas in conventional keyframing the domain is a set of time instants, in the spatial keyframe environment the domain is the 2D set of points on the screen. As a consequence, pose reconstruction is based on a 2D neighborhood, rather than a one-dimensional neighborhood.

B. Pose projection

Traditional temporal keyframing requires a careful positioning of poses along the time axis. In spatial keyframing, markers are positioned on the screen. This is usually done manually. If poses are imported from motion capture, a nice feature is to lay out the corresponding markers on the screen automatically. However, this placement needs not be final. One feature of spatial keyframing is the ability to reposition markers as user wishes. As a first guess, it is possible to treat the character pose as a multidimensional vector and model this problem as a projection of these vectors to a lower dimensional space. There are algorithms capable of this task, but we have to take in consideration some issues such as time performance, support for repositioning and the availability of multiple distance metrics.

A complete review of possible algorithms is beyond the scope of this work. We shall only cite a small set of popular

solutions. Assa et al. [10] use multiple distance metrics for comparing poses and proposes the use of a type of MDS[1] called RMDS[2]. Jin et al. [12] use LLE[3] for working with pose data in low dimension. The main difference between them is that LLE does his positioning just considering the surrounding neighborhood while MDS algorithms like RMDS use all available data. However, none of them gives support for data repositioning in low dimension. LAMP[4] [13] is a technique for data projection in low dimension which implements data relocation. However, LAMP needs a seed projection to work. Usually, this is done by another approach known as force projection [14].

C. Motion timing

Given a marker set positioning from motion capture poses, the user has to be able to recover part of motion capture realism given by its timing data. However, using this information is not trivial. Animation timing in spatial keyframing is done in real time by the user. To make both ends meet, we need to find a parameterization which makes the controller trajectory match the timing from the motion capture. Our current idea is use a two-step approach. In a first step, the user would produce the raw animation giving a trajectory to the controller. This trajectory can be edited in a second step to adjust the motion timing. The system should ensure that a controller trajectory between two markers, which are separated by a given time interval in the mocap data, follow the same animation timing. Proposing a solution for editing the controller trajectory using motion capture data is a key issue, which we hope to be able to offer in an animation authoring environment with enhanced realism.

IV. Current status

Until the present moment, most of the work targeted the reconstruction of the motion capture animation inside a spatial keyframe environment. In a first experiment, we tried to rebuild motion capture frames using a curve simplification strategy and the spatial keyframing interpolation scheme. Markers were laid out along a single line, mimicking a time slider. The idea of this experiment was to investigate the adequacy of positional and rotational pose distance functions. As a whole, the positional distance function seems to produce a more visually faithful animation than the rotational distance function. Fig. 1 illustrates reconstruction result with a positional pose distance. The number of keyframes used was fixed in 10 % from the total of frames. Weights were chosen as the level of hierarchy for each degree of freedom, being the root joint equal to one.

In a second experiment, we tried to capture the effects of placing markers on the screen, using a pose projection scheme in low dimension. The scheme used was force projection [14] and keyframes are the same chosen in previous experiment.

[1]Multi Dimensional Scaling
[2]Replicated Multi Dimensional Scaling.
[3]Local Linear Embedding
[4]Local Affine Multidimensional Projection

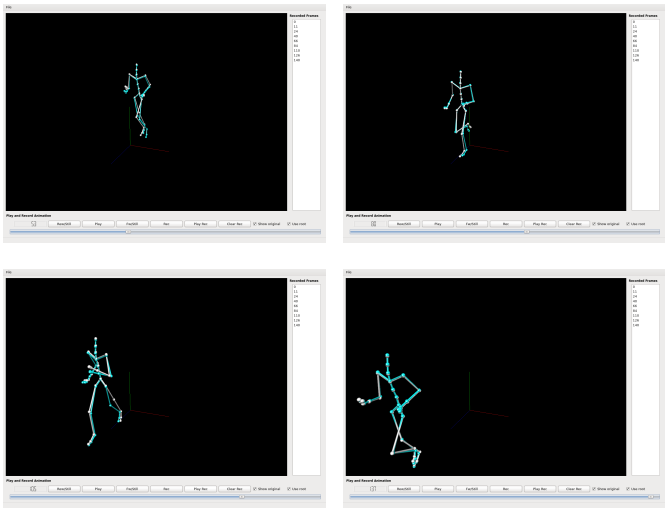

Fig. 1. Screenshots of a motion run rebuilt by curve simplification with 9 keyframes from a total of 141. White skeleton shows the motion capture pose for the current frame and blue skeleton the reconstructed pose.

Fig. 2 illustrates a motion run being reproduced in this scheme. The controller trajectory is automatically updated without user intervention. With the same amount of keyframes we are able to reach the same basic result of our previous reconstruction. However, if a big number of markers are used, animation quality degrades since the distribution will produce clusters over-representing similar poses, which causes the RBFs to skew the interpolated pose towards these clusters. This is to be expected, since the pose selection was conducted in the time domain rather than in the spatial domain.

V. Future work

The initial tests using force projection approach seem promising but there are other low dimension projection approaches to be tested. In fact, we only listed a few possible projection approaches we are aware of and a more complete review of them might show more suitable algorithms to be used. Up to now, LAMP seems the most suited for its ability to reposition all markers given a set of control points. In fact, manual positioning of markers is one of the next experiments.

We should also test the current approach also more complex motion capture data or with more than one file as input. A more complex pose set will have a great impact on marker positioning and also in the final result.

Most importantly, user intervention has to be considered. All issues related to the reuse of timing information have yet to be modeled. User input such as creating new poses or removing some should also be taken into consideration. One interesting feature we hope to achieve is to automatically propose a time transition between synthetic poses, given that they have a motion capture transition information for their surrounding markers. User feedback should play a crucial role in the evaluation of the system.

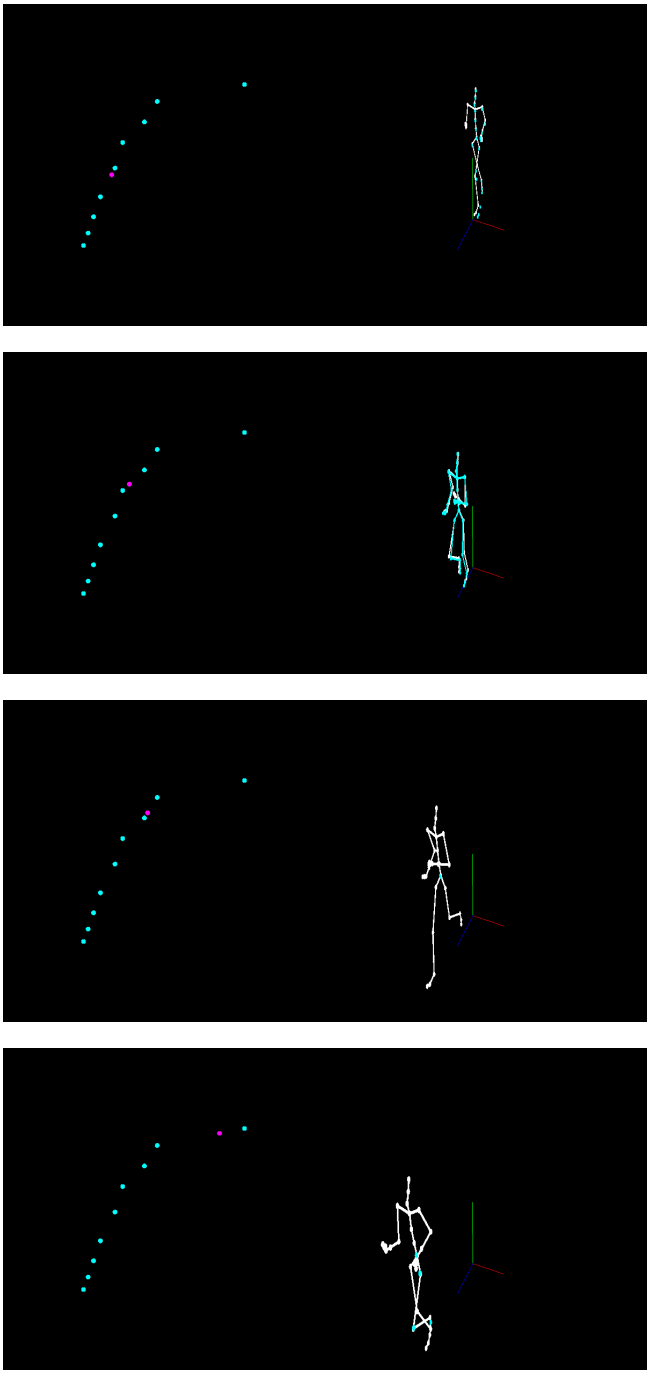

Fig. 2. Motion run with markers automatically positioned by force projection scheme. On the right, white skeleton shows motion capture pose for current frame and blue skeleton the same pose rebuilt with curve simplification. There is a huge overlap between them. On the left, blue points represent markers and purple point represents the controller.

ACKNOWLEDGMENT

The authors would like to thank reviewers for their valuable feedback.

REFERENCES

[1] T. Igarashi, T. Moscovich, and J. F. Hughes, "Spatial keyframing for performance-driven animation," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '05. New York, NY, USA: ACM, 2005, pp. 107–115. [Online]. Available: http://doi.acm.org/10.1145/1073368.1073383

[2] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 473–482, Jul. 2002. [Online]. Available: http://doi.acm.org/10.1145/566654.566605

[3] A. Safonova, J. K. Hodgins, and N. S. Pollard, "Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 514–521, Aug. 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015754

[4] J. Xiao, Y. Zhuang, T. Yang, and F. Wu, "An efficient keyframe extraction from motion capture data." in *Computer Graphics International*, ser. Lecture Notes in Computer Science, T. Nishita, Q. Peng, and H.-P. Seidel, Eds., vol. 4035. Springer, 2006, pp. 494–501. [Online]. Available: http://dblp.uni-trier.de/db/conf/cgi/cgi2006.html#XiaoZYW06

[5] K.-S. Huang, C.-F. Chang, Y.-Y. Hsu, and S.-N. Yang, "Key probe: a technique for animation keyframe extraction," *The Visual Computer*, vol. 21, no. 8, pp. 532–541, 2005. [Online]. Available: http://dx.doi.org/10.1007/s00371-005-0316-0

[6] C. Halit and T. Capin, "Multiscale motion saliency for keyframe extraction from motion capture sequences," *Computer Animation and Virtual Worlds*, vol. 22, no. 1, pp. 3–14, 2011. [Online]. Available: http://dx.doi.org/10.1002/cav.380

[7] Q. Zhang, S. Zhang, and D. Zhou, "Keyframe extraction from human motion capture data based on a multiple population genetic algorithm," *Symmetry*, vol. 6, no. 4, p. 926, 2014. [Online]. Available: http://www.mdpi.com/2073-8994/6/4/926

[8] H. Q. Dinh, G. Turk, and G. Slabaugh, "Reconstructing surfaces by volumetric regularization using radial basis functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 10, pp. 1358–1371, Oct 2002.

[9] K. Pullen and C. Bregler, "Motion capture assisted animation: Texturing and synthesis," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 501–508, Jul. 2002. [Online]. Available: http://doi.acm.org/10.1145/566654.566608

[10] J. Assa, Y. Caspi, and D. Cohen-Or, "Action synopsis: Pose selection and illustration," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 667–676, Jul. 2005. [Online]. Available: http://doi.acm.org/10.1145/1073204.1073246

[11] E. Bulut and T. Capin, "Key frame extraction from motion capture data by curve saliency," *Computer Animation and Social Agents*, p. 119, 2007.

[12] C. Jin, T. Fevens, and S. Mudur, "Optimized keyframe extraction for 3d character animations," *Computer Animation and Virtual Worlds*, vol. 23, no. 6, pp. 559–568, 2012. [Online]. Available: http://dx.doi.org/10.1002/cav.1471

[13] P. Joia, F. Paulovich, D. Coimbra, J. Cuminato, and L. Nonato, "Local affine multidimensional projection," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2563–2571, Dec 2011.

[14] E. Tejada, R. Minghim, and L. G. Nonato, "On improved projection techniques to support visual exploration of multidimensional data sets," *Information Visualization*, vol. 2, no. 4, pp. 218–231, Dec. 2003. [Online]. Available: http://dx.doi.org/10.1057/palgrave.ivs.9500054