Motion Path Synthesis for Intelligent Avatar

Feng Liu[1] and Ronghua Liang[1,2]

[1] College of Computer Science, Zhejiang University
`lffred@yahoo.com.cn`
[2] Institute of VR and Multimedia, Hang Zhou Inst. of Electronics Engineering
`rhliang2000@eyou.com`

Abstract. In this paper, we present a new motion generation technique, called motion path synthesis. The objective is to generate a motion for an intelligent avatar to move along a planned route using a pre-existing motion library. First, motion primitives, each defined as a dynamic motion segment, are extracted from the motion library. Then a motion graph is constructed based on the motion primitives and their connectivities. Within this motion graph, the desired realistic motion for avatars can be synthesized through a two-stage process: search an optimal motion path within the motion graph, joint the motion path and adapt it to the route. The experiment shows the effectiveness of the presented technique.

1 Introduction

Creating a realistic motion along a planned route is often an important task in making an intelligent virtual avatar. As people are well qualified to discern the artificiality in the motion of human-like avatars, both effort and expertise are needed to create realistic motions by key framing. And a large number of DOFs (Degree of Freedom) of avatars make it even more difficult.

A recent popular solution to this problem is motion capture [1]: the required motion along a given route is performed by an actor and then recorded to drive avatars. However, motion capture data is hard to be adapted to different routes. Two methods have been presented to improve the re-use of the motion capture data. One is to adapt existing motions to new requirements through interactive control, such as constraint based methods [2, 3], motion retargeting [4], etc. A more attractive way is to synthesize new motions from example. That is to generate motions through selecting and jointing existing motions along a specified path or according to statistical distributions. Most presented techniques, however, are limited by the size of a motion library, and the results lack variation.

In this paper, we present a new example based motion synthesis technique for creating motions for intelligent avatars to move along a planned route. Each motion is considered to be composed of a series of motion primitives, the minimal element that embodies the dynamic of a motion, and is modelled as a first-order Markov process. Then a directed graph, called motion graph, can be constructed from the motion library, with each motion primitive as a vertex. New motion can be generated by synthesizing primitives along a constrained or specified path within the motion graph. Since each new motion is created by segmented motion capture data (motion primitives), the reality of each motion is preserved. And as motion primitives are used in synthesizing new motions instead of full-length motion clips, the promising motion space is well enlarged.

The remainder of the paper is organized as follows: in the next section, we give an overview on related work. In Section 3, we describe motion graph construction. In Section 4, we propose motion path synthesis for intelligent avatars. We show the result in Section 5 and conclude the paper in the last section.

2 Related Work

Fruitful work has been carried out to adapt motion capture data for avatars to new application. The early research mostly aimed at providing convenient tools for interactive motion editing. A constraint based method was proposed for editing a pre-existing motion such that it meets new needs yet preserves the original quality as much as possible [2, 3]. A similar technique was presented for adapting a motion from one character to another [4]. Other researchers, such as Bruderlin and Williams [5], apply techniques from image and signal-processing domain to designing, modifying and adapting animated motion. Similarly, Liu et al.[6] provided a series of tools for editing motion at a high level by introducing wavelet transformation into motion analysis and synthesis. Unuma et al.[7] described a method for modelling human figure locomotion with emotions. Brand and Hertzmann [8] proposed a style machine to produce new motion with the desired feature.

More recently, a fantastic approach, motion synthesis by example, is presented. Molina-Tanco and Hilton [9] presented a system that can learn a statistical model from motion capture data and synthesize new motions by specifying the start and end key-frames, and sampling the original captured sequence according to the statistical model to generate novel sequences between key-frames. Pullen and Bregler [10] presented a motion capture assisted animation, which can derive realistic motions from partially key-framed motion using motion capture data. Arikan and Forsythe [11] constructed a hierarchical graph from a motion library and adopted a randomized search algorithm to extract motion according to user constraints. Kovar et al. [12] build a similar motion graph that encapsulates connections among the motion library and synthesize motions by building walks on the graph. In the work of Li et al.[13], motion data is divided into motion textons, each of which can be modelled as a linear dynamic system. Motions are synthesized by considering the likelihood of switching from one texton to the next.

3 Motion Graph

Each motion is represented as a frame sequence, with each frame defining a posture using a popular hierarchical posture description, in which the posture

of an articulate figure is specified by its joint configurations together with the orientation and position of **Root** as follows:
$\boldsymbol{F}_t = (\boldsymbol{T}_{Root}(t),\ \boldsymbol{Q}_{Root}(t),\ \boldsymbol{Q}_1(t),\ \boldsymbol{Q}_2(t) \ldots \boldsymbol{Q}_n(t))$
where $\boldsymbol{T}_{Root}(t)$ and $\boldsymbol{Q}_{Root}(t)$ are the translation vector and rotation vector of **Root** at time t, and $\boldsymbol{Q}_i(t)$ is the rotation vector of joint i around its parent joint at time t.

A directed graph, called motion graph as shown in Fig. 1, is built to capture the connection among motions (motion segments) in a motion library. Motion graph is a newly arisen structure for motion description, plan and generation. Each vertex is associated with a motion segment, varying from a single frame to a full-length motion, and the edge from vertex i to vertex j is associated with a weight denoting the connectivity from the motion segment associated with vertex i to that with vertex j. In Arikan and Forsyth [11], each vertex is a single frame, which enlarges the motion graph and the promising motion space. Its drawback, however, is obvious in that a large number of vertices in the graph cause high calculation overhead, and the realism of the original motion is damaged severely for the selected neighbor frames do not necessarily have smooth transitions. Whereas it comes to associating each vertex with a long motion segment or even a full-length motion, the realism of the original motion is preserved at the cost of small promising motion space. We adopt a similar scheme to Li et al.[13], and associate each vertex with a motion segment. Unlike Li et al.[13], in which each motion segment is replaced completely with a dynamic model, called motion texton, we preserve all the original frames in the motion segment/primitive and synthesize new motions with them. Thus the detail and reality of motion capture data is well preserved.

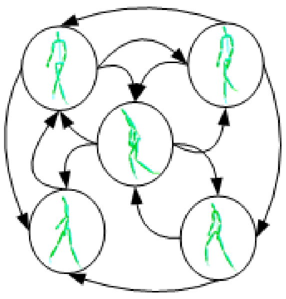
Fig. 1. Motion graph.

3.1 Motion Primitive Extraction

Motion primitive is defined as a fundamental segment that captures the dynamics of a motion and can be well fitted to a quadratic dynamic model in the following way: $\boldsymbol{F}_t = \boldsymbol{F}_0 + \boldsymbol{A}t^2 + \boldsymbol{B}t$, where $\boldsymbol{F}_0$ and $\boldsymbol{F}_t$ are the initial and t^{th} frames, $\boldsymbol{A}$ and $\boldsymbol{B}$ are dynamic parameters.

In this paper, we model an articulate avatar with the global position, orientation and other 17 joints. Thus the posture can be represented as a 57-dimension vector and a motion $\boldsymbol{M}$ with T frames can be described as a T×57 matrix. To reduce the high dimensions, we adopt **SVD** to decompose $\boldsymbol{M}$ and get the principal motion components in the following way: $\boldsymbol{M}_{T\times 57} = \boldsymbol{U}_{T\times T}\Lambda_{T\times 57}\mathrm{V}_{57\times 57}$. Let $\boldsymbol{U}_{1-q}$ be the first q columns of $\boldsymbol{U}$, it comprises the principal components of $\boldsymbol{M}$. Greedy algorithm is employed to extract motion primitives.

1. Fetch a motion $\boldsymbol{M}$ from the library, decompose it using SVD and get $\boldsymbol{U}_{1-q}$.
2. Fetch T_{min} frames from $\boldsymbol{U}_{1-q}$, and fit them to a dynamic model using the least square method. Add subsequent frames to them until the fitting error exceeds a given threshold.

3. Fetch the next T_{min} frames from $\boldsymbol{U}_{1-q}$, and extract a new motion primitive $\boldsymbol{MP}_i$ in the similar way to that described in step 2.
4. Repeat step 3 until all frames in $\boldsymbol{M}$ are processed.
5. Repeat step from 1 to 4 until all motions in the library are processed.

3.2 Motion Graph Construction

Inspired by the previous work, such as Li et al.[13] and J.Lee et al.[14], we model each motion as a first-order Markov process. Each motion primitive is a state and the next state only depends on the current state. Unlike J.Lee et al.[14], each state is a motion primitive, instead of a single frame, which makes the previous states have little effect on the current state and thus makes this motion model more plausible. Each motion in the library can be modelled as a first-order Markov process, and thus the motion library can be represented as a motion graph. Each vertex is a motion primitive, and the directed edge e_{ij} from $\boldsymbol{MP}_i$ to $\boldsymbol{MP}_j$ is defined as the connectivity from $\boldsymbol{MP}_i$ to $\boldsymbol{MP}_j$, $\mathrm{P}(\boldsymbol{MP}_j|\boldsymbol{MP}_i)$, which can be calculated as the similarity between the last L frames from $\boldsymbol{MP}_i$ and those from the precedence of $\boldsymbol{MP}_j$.

The similarity between two frame sequences can be defined as the sum of the similarity between the two corresponding trajectories of **Root** and that between two posture sequences as follows.

$$sim(\boldsymbol{M}_i, \boldsymbol{M}_k) = \beta sim_t(\boldsymbol{M}_i, \boldsymbol{M}_k) + (1-\beta) sim_p(\boldsymbol{M}_i, \boldsymbol{M}_k)$$

$$sim_t(\boldsymbol{M}_i, \boldsymbol{M}_k) = \exp(-\min_{\boldsymbol{R},\boldsymbol{T}} \sum_{t=1}^{L} (\alpha((\boldsymbol{R}\boldsymbol{M}_{pit} + \boldsymbol{T}) - \boldsymbol{M}_{pkt})^2 + (1-\alpha)(\boldsymbol{R}\boldsymbol{M}_{oit} - \boldsymbol{M}_{okt})^2)/L)$$

$$sim_p(\boldsymbol{M}_i, \boldsymbol{M}_k) = \exp(-\sum_{t=1}^{L} \sum_{j} \omega_j (\boldsymbol{M}_{jit} - \boldsymbol{M}_{jkt})^2/L)$$

where $\mathrm{sim}(\boldsymbol{M}_i,\boldsymbol{M}_k)$ is the similarity between $\boldsymbol{M}_i$ and $\boldsymbol{M}_k$, $\mathrm{sim}_t(\boldsymbol{M}_i,\boldsymbol{M}_k)$ is the similarity between the two **Root** trajectories, $\mathrm{sim}_p(\boldsymbol{M}_i,\boldsymbol{M}_k)$ is the similarity between the two posture sequences. $\boldsymbol{M}_{jit}$ is the rotation of joint j in frame t in $\boldsymbol{M}_i$, and ω_j is a weight indicating the significance of joint j. $\boldsymbol{M}_{pit}$ and $\boldsymbol{M}_{oit}$ are the position and orientation of **Root** in frame t in $\boldsymbol{M}_i$ respectively. To avoid the disturbance of the initial orientation and position of a motion, we calculate the optimal $\mathrm{sim}_t(\boldsymbol{M}_i,\boldsymbol{M}_k)$ by transforming $\boldsymbol{M}_i$ globally with rotation $\boldsymbol{R}$ and translation $\boldsymbol{T}$. The optimal $\boldsymbol{R}$ and $\boldsymbol{T}$ can be found using the algorithm detailed in [15]. The result is normalized such that $\sum_{j=1}^{N} e_{ij}=1$. If no precedence of $\boldsymbol{MP}_j$ is found, the connectivity is set 0.5 empirically.

4 Motion Synthesis

We generate the desired motion along a planned route $\boldsymbol{R}$ through a two-stage process. Search a motion path within the motion graph, joint the motion primitives on the extracted motion path and adapt it to the route $\boldsymbol{R}$.

4.1 Motion Path Finding

The optimal motion path shall satisfy the following two criterions, fitting for the route and having natural transition. We define the motion path **MS** as follows:

$$\mathbf{MS} = \arg\max_{\Pi} P(\boldsymbol{MP}_{k1}, \boldsymbol{MP}_{k2} \cdots \boldsymbol{MP}_{kn} | \boldsymbol{MP}_{k1} = \boldsymbol{MP}_s, \boldsymbol{G}, \boldsymbol{R})$$

where $\prod$ is the set of motion primitive sequences beginning with $\boldsymbol{MP}_s$ within the motion graph $\boldsymbol{G}$. We can transform this problem into a path finding problem in a graph based on the first-order Markov process as follows.

$$\begin{aligned}
\mathbf{MS} &= \arg\max_{\Pi} P(\boldsymbol{MP}_{k1}, \boldsymbol{MP}_{k2} \cdots \boldsymbol{MP}_{kn} | \boldsymbol{MP}_{k1} = \boldsymbol{MP}_s, \boldsymbol{G}, \boldsymbol{R}) \\
&= \arg\max_{\Pi} P(\boldsymbol{MP}_{k2}|\boldsymbol{MP}_s) P(\boldsymbol{MP}_{k2}|\boldsymbol{R}, \boldsymbol{S}_{k2}) \cdots P(\boldsymbol{MP}_{kn}|\boldsymbol{MP}_{kn-1}) P(\boldsymbol{MP}_{kn}|\boldsymbol{R}, \boldsymbol{S}_{kn-1}) \\
&= \arg\max_{\Pi} lgP(\boldsymbol{MP}_{k2}|\boldsymbol{MP}_s) P(\boldsymbol{MP}_{k2}|\boldsymbol{R}, \boldsymbol{S}_{k2}) \cdots P(\boldsymbol{MP}_{kn}|\boldsymbol{MP}_{kn-1}) P(\boldsymbol{MP}_{kn}|\boldsymbol{R}, \boldsymbol{S}_{kn-1}) \\
&= \arg\min_{\Pi} -(lgP(\boldsymbol{MP}_{k2}|\boldsymbol{MP}_s) P(\boldsymbol{MP}_{k2}|\boldsymbol{R}, \boldsymbol{S}_{k2}) + \cdots P(\boldsymbol{MP}_{kn}|\boldsymbol{MP}_{kn-1}) P(\boldsymbol{MP}_{kn}|\boldsymbol{R}, \boldsymbol{S}_{kn-1}))
\end{aligned}$$

where $\boldsymbol{MP}_s$ is the specified start motion primitive, P$(\boldsymbol{MP}_{kj}|\boldsymbol{MP}_{ki})$ is the transition from $\boldsymbol{MP}_{ki}$ to $\boldsymbol{MP}_{kj}$, P$(\boldsymbol{MP}_{ki}|\boldsymbol{R},\boldsymbol{S}_{ki})$ is the fitness of $\boldsymbol{MP}_{ki}$ to the path segment on $\boldsymbol{R}$ starting at the position $\boldsymbol{S}_{ki}$ and is defined as a decreasing function of the total change needed to adapt $\boldsymbol{MP}_{ki}$ to the path segment. The process of adapting is detailed in Section 4.2.

We construct a hierarchical directed graph as shown in Fig. 2. The start motion primitive is selected as vertex V0. Let the number of all the motion primitives be n, we set the other vertices V$_{ij}$=$\boldsymbol{MP}_j$, where V$_{ij}$ is the j^{th} vertex at level i, $j\epsilon[1, n]$. The weight of each edge is -lgP$(\boldsymbol{MP}_{ki}|\boldsymbol{MP}_{ki-1})$-lgP$(\boldsymbol{MP}_{ki}|\boldsymbol{R},\boldsymbol{S}_{ki})$, to be calculated in the runtime. According to this graph, the problem can be transformed to finding a shortest path. Because no evident end vertex is present and the number of vertices in the graph is infinite, the traditional Dijkstra algorithm can not be adopted directly. We devise an adapted Dijkstra algorithm. Let $\boldsymbol{S}$ be the set of vertices whose final shortest path weights starting at V0 have been determined, and $\boldsymbol{Q}$ be the set of vertices whose best estimations of the shortest path weights have been calculated, the adapted Dijkstra algorithm can be outlined as follows:

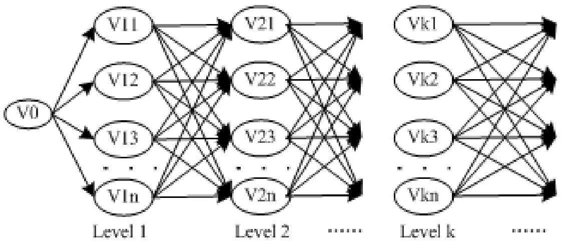

Fig. 2. Hierarchical graph for motion path finding.

```
1)Initialization.
  V0.d=0      //set the shortest path weight starting at V0 to this vertex
  V0.p_end=P0//P0 is the start location of the route R
  Add(V0,Q)  //add V0 to Q
2)While the end location of R has not been approximated enough
  Vc=Extract-Min(Q)//fetch Vc with the minimal shortest path weight in Q
  Add(Vc,S)        //add Vc to S
  Remove(Vc,Q)     // remove Vc from Q
  for each vertex Vi in the next level to Vc
   Vi.pre=Vc        // set Vc as the precedence of Vi
   Vi.d=Vc.d+(-lg(P(Vi|Vc)P(Vi|R,Vc.p_end))// calculate the shortest path
                                       // weights from V0 to Vi
   Vi.p_end=Reach-end(R,Vc.p_end,Vi)//calculate the end location after
                                                          the shortest
                            //path from V0 to Vi has been adapted to R
    if Vi has already been in Q,
      Update the Vi in Q
    else
      Add(Vi,Q)
    endif
 endfor
endwhile
```

The calculation of Reach-end(***R***,Vc.p_end,Vi), the end location after the shortest path from V0 to Vi has been adapted to ***R***, is similar to the process for jointing motion primitives in Section 4.2. The motion path can be obtained through starting from the last vertex added to ***S***, and tracing its precedence in ***S*** back up to the start vertex.

4.2 Motion Primitives Jointing and Adapting

Let ***M*** be the motion resulting from jointing all the motion primitives on the motion path before $\boldsymbol{MP}_{i+1}$, the current motion primitive to be connected to ***M***, ***PS*** be the end location of ***M***, we devise the following motion transition and adaptation algorithm to append ***M*** with $\boldsymbol{MP}_{i+1}$.

1. Fetch the first frame from $\boldsymbol{MP}_{i+1}$ as an appended frame.
2. Translate the appended frame to locate its **Root** at ***PS***.
3. Calculate the tangent of ***R*** at ***PS***, and orientate the appended frame parallel to the tangent. If the appended frame is the first frame of $\boldsymbol{MP}_{i+1}$, replace the last frame of ***M*** with it. Otherwise append ***M*** with this appended frame.
4. If no frame left in $\boldsymbol{MP}_{i+1}$, end. Otherwise fetch the next frame as the appended frame, and move ***PS*** along ***R*** with the distance from the appended frame to its previous frame in $\boldsymbol{MP}_{i+1}$. Goto Step 2.
5. Smooth ***M*** with a Gauss convolution template ***G***.

$$\begin{bmatrix} M(m-T_b) \\ M(m-T_b+1) \\ \cdots \\ M(m+T_b) \end{bmatrix} = \begin{bmatrix} G_0(0) & G_0(1) & \cdots & G_0(2\mathrm{T}_b) \\ G_1(\text{-}1) & G_1(0) & \cdots & G_1(2\mathrm{T}_b\text{-}1) \\ \cdots & \cdots & \cdots & \cdots \\ G_{2T_b}(\text{-}2\mathrm{T}_b) & G_{2T_b}(1\text{-}2\mathrm{T}_b) & \cdots & G_{2T_b}(0) \end{bmatrix} \begin{bmatrix} M(m-T_b) \\ M(m-T_b+1) \\ \cdots \\ M(m+T_b) \end{bmatrix}$$

$G_i(t)=p(t)/\sum_{j=-i}^{2Tb-i} p(j)$

where m is the number of frames in ***M*** before appending it with $\boldsymbol{MP}_{i+1}$,[m-T$_b$,m+T$_b$] is the range for smoothing, and p(t) is the Standard Normal Distribution.

Table 1. Composition of the motion library

Motion	The number of frames	The number of Motion Primitives
Normal Walk	136	7
Cat Walk	121	8

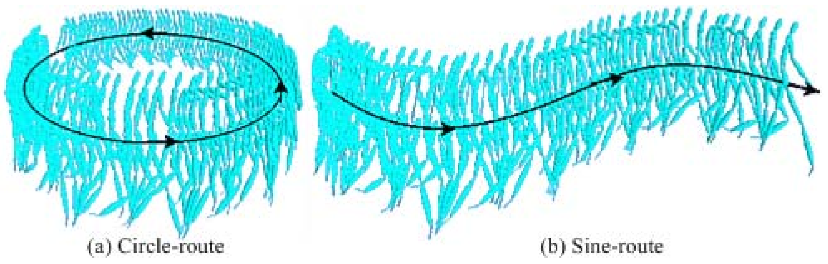

Fig. 3. Path synthesis.

5 Experiment

To verify the effectiveness of the presented technique, we build a small motion library as shown in Table 1. We extract motion primitives from this library, and the number of motion primitives extracted from each motion is also shown in Table 1. A motion graph is constructed using these motion primitives. Below we show some motions synthesized within this motion graph.

Firstly, a circle route is specified, and a natural motion along this route is synthesized. This motion consists of 477 frames. The result is shown in Fig. 3(a). Secondly, a sine curve is sketched. Again, a smooth motion along this route is generated. This motion consists of 1020 frames. The result is shown in Fig. 3(b).

From these two examples, we can see that there are far more frames in each generated motion than those in the original motion library. Some motion primitives are frequently used to generate the desired motion, which demands new transitions for every two motion primitives which are not connected in the original motions. Within the presented motion generation framework, we propose two strategies to achieve the smooth transition, selecting the motion primitive sequence, which is most likely to have natural transitions among its every two nearby motion primitives, and smoothing every two nearby motion primitives with a Gauss convolution template. The results above show that using these two strategies does help to produce realistic motions. We adopt a frame by frame strategy to adapt the motion primitive sequence to the route. To preserve the realism of the motion as much as possible, we incorporate a least change strategy into the motion primitive sequence extraction. The results show that the generated motions fit the route well; meanwhile, these generated motions are realistic though something different from the original one.

6 Conclusion

In this paper, we present a new example based motion synthesis technique to create motions for intelligent avatars to move along a planned route. The main contribution of this paper is to propose a new motion graph using motion capture data. Because we adopt motion primitives, segments from the original motions, as the vertices, and use them to create new motions, our motion graph preserves the reality of the original motion yet provides a large promising motion space. Another contribution is that we present an efficient way to generate motions for intelligent avatars. Given a planned route, the motions can be produced automatically.

The problem in this technique is that we use the motion primitives to construct the motion graph directly, which results in the problem of high computational overhead in searching the optimal motion path when the motion library is large. This drawback may affect the application of our method in real-time fields when the motion library is large. We will focus our future work on it.

References

1. Michael Gleicher.: Animation from observation: Motion capture and motion editing. Computer Graphics, 1999. 4(33):51–55
2. Michael Gleicher.: Motion editing with spacetime constraints. In: Proceedings of the 1997 Symposium on Interactive 3D Graphics. Providence,1997. 139–148
3. Jehee Lee, Sung Yong Shin.: A hierarchical approach to interactive motion editing for human-like figures. In: Proceedings of SIGGRAPH 99. Los Angeles,1999. 39–48
4. Michael Gleicher.: Retargeting motion to new characters. In: Proceedings of SIGGRAPH 98. Orlando, Florida, 1998. 33–42
5. Bruderlin. A, Williams. L.: Motion signal processing. In: Proceedings of SIGGRAPH 95. Los Angeles, 1995. 97–104
6. Feng Liu, Yueting Zhuang, Zhongxiang Luo, Yunhe Pan.: A hybrid motion data manipulation: Wavelet based motion Processing and spacetime rectification. In: Proceedings of IEEE PCM 2002, Hsinchu, Taiwan, 2002. 743–750
7. Unuma. M, Anjyo. K, Takeuchi. R.: Fourier principles for emotion-based human figure animation. In: Proceedings of SIGGRAPH 95. Los Angeles, 1995. 91–96
8. Matthew Brand, Aaron Hertzmann.: Style Machine. In: Proceedings of SIGGRAPH 2000. New Orleans, 2000. 183–192
9. L. Molina Tanco, A. Hilton.: Realistic synthesis of novel human movements from a database of motion capture examples. In: Proceedings of IEEE Workshop on Human Motion. Austin, Texas, 2000. 137–142
10. Katherine Pullen, Christoph Bregler.: Motion capture assisted animation: Texturing and synthesis. In: Proceedings of SIGGRAPH 2002. San Antonio, Texas, 2002. 501–508
11. Okan Arikan, D.A. Forsyth.: Interactive Motion Generation From Examples. In: Proceedings of SIGGRAPH 2002. San Antonio, Texas, 2002. 483–490
12. Lucas Kovar, Michael Gleicher, Frdric Pighin.: Motion graphs. In: Proceedings of SIGGRAPH 2002. San Antonio, Texas, 2002. 473–482
13. Yan Li , Tianshu Wang, Heung-Yeung Shum.: Motion texture: A two-level statistical model for character synthesis. In: Proceedings of SIGGRAPH 2002. San Antonio, Texas, 2002. 465–472

14. Jehee Lee , Jinxiang Chai , Paul S. A. Reitsma , Jessica K. Hodgins, Nancy S. Pollard.: Interactive control of avatars animated with human motion data. In: Proceedings, SIGGRAPH 2002. San Antonio, Texas, 2002. 491–500
15. A. Eden.: Directable Motion Texture Synthesis. technical report, Harvard University, April 2002.