ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.9, September 2007, pp.2356−2364
DOI: 10.1360/jos182356

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种虚拟人运动生成和控制方法*

高全胜[1,2+], 洪炳熔[1]

[1](哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

[2](常州纺织服装职业技术学院 信息技术系,江苏 常州 213164)

A Method for Character Animation Generation and Control

GAO Quan-Sheng[1,2+], HONG Bing-Rong[1]

[1](Department of Computer Science and Engineering, Harbin Institute of Technology, Harbin 150001, China)

[2](Department of Information Technology, Changzhou Textile Garment Institute, Changzhou 213164, China)

+ Corresponding author: Phn: +86-519-6456686,E-mail: gaoquansheng@msn.com

Gao QS, Hong BR. A method for character animation generation and control. *Journal of Software*, 2007, 18(9):2356−2364. http://www.jos.org.cn/1000-9825/18/2356.htm

Abstract: In this paper presents an approach for creating lifelike, controllable motion in interactive virtual environments. This is achieved through learning statistical model from a set of motion capture sequences. The method is based on clustering the motion data into motion primitives that capture local dynamical characteristics—Dynamic texture, modeling the dynamics in each cluster using linear dynamic system (LDS), annotating those LDS' which have clear meaning, and calculating the cross-entropy between frames of LDS' to construct a directed graph called a annotated dynamic texture graph (ADTG) which has two-level structure. The lower level retains the detail and nuance of the live motion, while the higher level generalizes the motion and encapsulates the connections among LDS'. The results show that this framework can generate smooth, natural-looking motion in interactive environments.

Key words: dynamic texture; motion synthesis; motion capture; LDS; character behavior

摘 要: 利用运动捕获数据,通过学习获得虚拟人运动的统计模型,从而创建真实、可控的虚拟人运动.提出了一种方法:通过对原始运动数据聚类,提取出局部动态运动特征——动态纹理,并用线性动态系统描述,有选择地注释有明确含义的线性动态系统,构建注释动态纹理图.利用这一统计模型,可生成真实感强、可控的虚拟人运动.结果表明,这种方法在交互环境中能够生成流畅、自然的人体运动.

关键词: 动态纹理;运动复合;运动捕获;线性动态系统;角色行为

中图法分类号: TP391 **文献标识码**: A

1 Introduction

Motion capture is a popular technique for synthesizing lifelike human motion. However, there are still a

* Supported by the National Natural Science Foundation of China under Grant No.69933020 (国家自然科学基金)
Received 2005-11-16; Accepted 2006-05-24

number of difficulties to overcome concerning how to use these motion data sets in a three-dimensional interactive virtual world. It is challenging to make balance between richness of character behavior and easiness of intuitive control for synthesizing lifelike motion.

Motion synthesis of the captured data largely follows two threads: basing synthesis on the frame level and basing it on the clip level. Early frame level efforts focus on editing a clip of the existing motion, and produce the desired motion by altering a base motion. This is done by using some editing tools, such as motion blending[1], motion warping[2], and retargeting[3]. Another alternative approach is to create a statistical model to provide the transition between frames. Schödl, *et al*.[4] use video textures to model a new motion as a first-order Markov process. Brand and Hertzmann[5] introduce stylistic HMM models to make stylistic motion synthesis. Kovar, *et al*.[6] present motion graphs for synthesizing seamless motion. Recent research of Lee and Lee[7] also uses frame-level transition methods in pre-computation method. While these frame level models retain the motion details and a broad variety of motion choices, the resulting data structure may be too complex for clear presentation in a user interface. The combinatorial explosion makes the searching at the frame level entirely intractable when the dataset is large.

In recent years, large motion datasets become commonplace. Datasets often contain many variants of the same kind of motion. People began to study how to synthesize motion at the clip level. Clip based approaches chop the long motion sequence into blocks, then use probabilistic method reassemble the blocks. Li, *et al*.[8] developed a two-level statistical model for motion synthesis. At the lower level, motion texton is represented by LDS, and at the higher level contain the transition probabilities among textons. Arikan and Forsyth[9] create a new motion by searching for plausible transitions between motion segments. Arikan, *et al*.[10] show that motion synthesis from a graph structure can be cast as a dynamic programming. The advantage of this method is that, in the synthesis phase, searching can be efficiently solved, but motion transitions can only take place at the end points of blocks. Our approach attempts to capture the strengths of the both by combining them in a two level structure. We provide high level control, and the searching is made at the block level; still we maintain transition at the frame level.

Arikan, *et al*.[10] annotate all frames with annotations draw from some fixed vocabulary. Our approach, in contrast, makes annotation at clip level. Furthermore, we only annotate those having clear meaning LDS. We believe a frame cannot represent an action, which is only a pose of the character, but an action is the process of pose changing. Therefore, the best unit of annotation is a clip of motion that completes the action. This is confirmed with our knowledge of what is action. Another drawback of making annotation at frame level is that it is too computationally expensive to solve the searching problem.

In this paper, we present a method for motion synthesis from motion data. Our goal is to provide a concise set of control commands, which is the same as the annotation vocabulary. User chooses the command to control the character's behavior in interactive environment by using input devices such as keyboard, mouse, or joystick. The transitions among different actions are automatically created by our system in real-time. In order to simplify the set of control commands, we pick to annotate those actions which have clear-cut meaning. Hence, we build a correspondence between the annotated LDS and the command in the control set. We demonstrate the power of our approach with character that reacts to the commands of the user and achieves specified tasks in an interactive environment (see Fig.1). The user can have a direct and immediate control over the motion of the character.

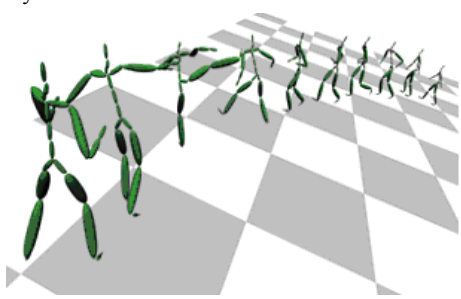
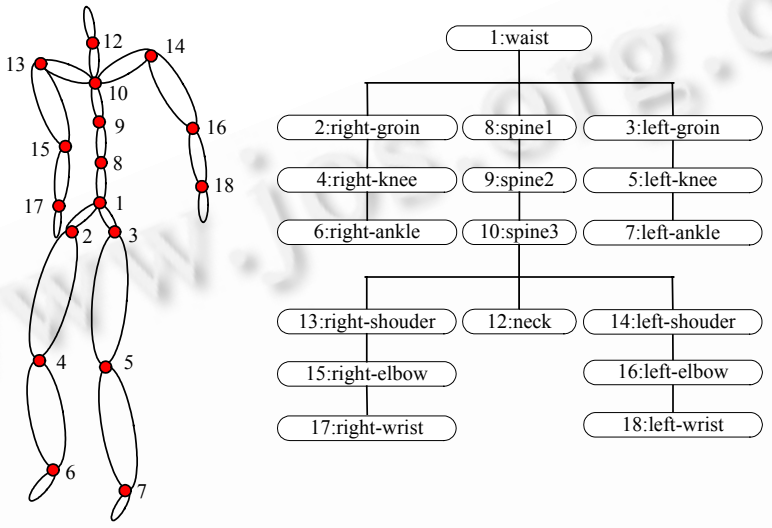


Fig.1 The real-time synthesized motion created using motion capture data

2 Extracting Dynamic Texture

We assume there is a dataset of motion data sequences of the same skeletal format; this is required to be able to extract motion texture and to be able to synthesize motion with the clips of motion texture. Every motion is represented as a time-series data that gives the configuration of an articulated figure. As illustrated in Fig.2, an articulated figure with n joints, we denote a motion by $Y_t=(y_t^0,y_t^1,...,y_t^n)^T$, where $y_t^o\in R^3$ describes the transitional motions of the root segment, $y_t^i\in S^3$ give the rotational motion of the ith joint for $1\le i\le n$, and here S^3 is unit quaternion space[11]. In our system, we model the configuration of an articulated figure with 18 joints. Therefore, Y_t is a 57-dimensional vector.



(a) Articulated figure model (b) Connection of figure joints

Fig.2 Human body model used

2.1 Model-Based cluster analysis

The human figure exhibits complex and rich dynamic behavior; a single model is inadequate to describe the evolution in configuration space. We use model-based cluster analysis[12] to generalize the motion data into clusters, which capture similarities in motion frames. In this approach, motion data are viewed as coming from a mixture of probability distributions, each representing a different cluster. Our goal is to capture the similar motion states within the same cluster, and different motion patterns belong to different clusters. The clusters are found to cut the trajectories into short sections, and all sections in a given partition have similar dynamics.

Given the observation sequences $Y=(Y_1,Y_2,\ldots,Y_t)$, let $f_k(Y_i|\theta_k)$ be multivariate normal of an observation Y_i from the kth cluster, where θ_k consists of a mean vector μ_k and a covariance matrix Σ_k,

$$f_k(Y_i\mid\mu_k,\Sigma_k)=\exp\{-1/2(Y_i-\mu_k)^T\sum_k^{-1}(Y_i-\mu_k)\}(2\pi)^{p/2}\mid\Sigma_k\mid^{1/2} \quad (1)$$

let K be the number of components in the mixture, $\theta=(\theta_1,\ldots,\theta_k)$, and $\tau=(\tau_1,\ldots,\tau_k)$, where τ_k is the probability that an observation belongs to the kth component. Our goal is to maximize the likelihood

$$L(\theta;\tau\mid Y)=\prod_{i=1}^{n}\sum_{k=1}^{K}\tau_k f_k(Y_i\mid\theta_k) \quad (2)$$

In practice, the EM algorithm estimates parameters θ of the clusters that maximize the mixture loglikelihood

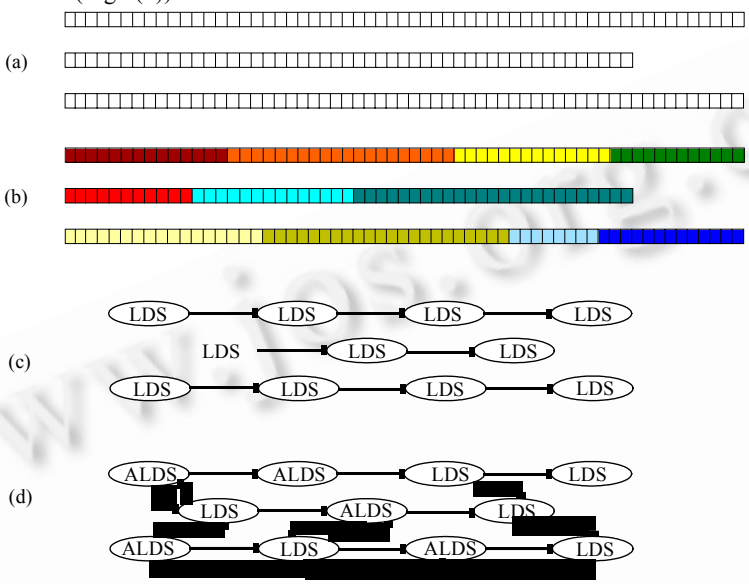
$$l(\theta_k,\tau_k,z_{ik}\mid Y)=\sum_{i=1}^{n}\sum_{k=1}^{K}z_{ik}[\log\tau_k f_k(Y_i\mid\theta_k)] \quad (3)$$

where z_{ik} is a indicator variable, $z_{ik}=1$ when anobservation point Y_i belongs to the kth cluster. The algorithm is

looped until convergence:

- E-step: Compute the expectation of z_{ik}, given the parameter estimates from the M-step.
- M-step: Maximize (3) given the expectation of z_{ik}.

To determine the number of clusters in the mixture model, we take different sizes of the cluster. Learning process is from big granularity to small one. We begin to cluster on *total length*/*number of cluster*=200 frames, till the smallest one—60 frames (Fig.3(b)).



(a) The motion database initially consists of many motion clips
(b) Our method clusters the motion data into motion primitives
(c) The dynamics is modeling in each cluster using linear dynamic system (LDS)
(d) We annotate those having clear meaning LDS and calculating the cross-entropy between frames of LDS' to construct a directed graph

Fig.3 Annotated dynamic texture graph (ADTG) built from initial clips

2.2 Learning dynamic texture

After the motion data are clustered, we learn local dynamics within each cluster. The motion data clip can be seen as time series produce by a system, which can be described using dynamic texture. A dynamic texture[13] is linear time-invariant dynamical systems (LDS). It can be described by the following two equations

$$X_{t+1}=A_1X_t+A_2X_{t-1}+W_t \tag{4}$$

$$Y_t=CX_t+V_t \tag{5}$$

time is indexed by the discrete index t, The Observation Y_t is a linear function of the state X_t , and the state at one time step depends linearly on the previous state. Both state and output noise, W_t and V_t, are zero-mean normally distributed random variables with covariance matrices Q and R, respectively. Only the output of the system is observed, and the state and all the noise variables are hidden. The parameters of an LDS can be represented by $\{A,C,Q,R,\pi_0,V_0\}$, where π_0 and V_0 are initial covariance and mean, respectively. There are standard ways of learning LDS parameters form training data, see Ref.[14] for details on calculating an LDS. We use EM algorithm to estimate the parameters of linear system. Figure 4 shows learned LDS of walking sequence and raw data measured in human subjects.

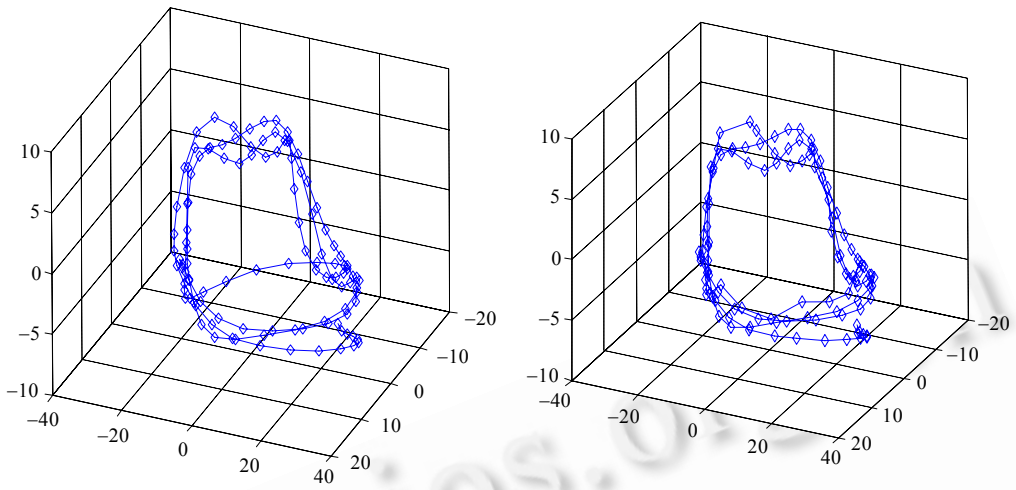


Fig.4 Learned LDS of walking sequence comprising 3.5 cycles (b) from raw data measured in human subjects (a)

After learning a LDS within each cluster, we evaluate the model prediction error, which is defined by

$$P_{error} = \sum_{i=1}^{l} D(Y_i, \hat{Y}_i) \tag{6}$$

where $\hat{Y}$ is the predicted sequence by the learned LDS, l is the length of the clip. $D(\cdot,\cdot)$ is a distance function defined by

$$D(Y_i, Y_j) = \sum_{k=1}^{n} \| \ln((y_j^k)^{-1} y_i^k) \|^2 \tag{7}$$

where $ln(\cdot)$ is the logarithmic map of unit quaternion[11]. If P_{error} is less than the given threshold, the LDS is retained, and the corresponding data clip is deleted from the data set. For the left data set, we cluster it with small granularity, and then learn LDS for each cluster. This process is repeated until all data sets are empty. Since joint rotation angle data are locally linear, LDS' can finally cover every data clips.

We add annotations to LDS to describe the motion. The description describes the qualitative property of the motion. It may be a word (such as walk, run, jump etc.) It can also be a composite action (such as walk and then run, run and turn right, stand and then walk etc.) In order to maintain a succinct command set, our method is only annotating those LDS' that have clear meaning. Hence, we make a correspondence between ALDS (annotated LDS) and the control command (Fig.3(d)).

3 Creating Texture Graph

3.1 Creating transitions

An important observation is that the synthesis block by LDS' is very similar to its original motion clip. We can tell what happens at a particular frame by knowing the original motion. This insight gives us a clue of how to make transition between LDS'. We can make transition between LDS' where the produced frames are similar. In order to preserve dynamics (orientations and velocities) we not only take frames about which we want to evaluate similarity, but also their adjacent frames within a temporal window as a slice of texture. The distance of two slices of texture is calculated by cross entropy.

3.1.1 Evaluation of the distance between LDS'

The observation sequences Y on the system are not exactly reproducible. For the same LDS and the same initial condition X_0, we may obtain different observation sequences Y due to the presence of various disturbances. In such cases it is natural to regard Y as a random variable of which we observe different realizations. That is, LDS can be formulated in term of the PDF for Y.

For D-dimensional Gaussians $\theta_i(Y)=N(Y;\mu_i,K_i)$ with mean μ_i and covariance K_i, i=1,2, the agreement between two PDF' (or two LDS') can be measured in terms of the cross-entropy

$$D(\theta_1 \| \theta_2)=\frac{1}{2}[\log|K_1|-\log|K_2|+\sum_{ij}(K_1^{-1})_{ij}(K_2^{-1})_{ij}+(\mu_1-\mu_2)^T K_1^{-1}(\mu_1-\mu_2)-d] \tag{8}$$

In order to compute the cross-entropy between LDS', we first represent state sequences $X=(X_1,X_2,\ldots,X_t)$ and observation sequences $Y=(Y_1,Y_2,\ldots,Y_t)$ as Gaussians, respectively. Hence, our goal is to express the PDF of an observation Y as multivariate normal.

Based on Eq.(4), we can write the conditional densities for the state:

$$p(X_t \mid X_{t-1})=\exp\left\{-\frac{1}{2}[X_t-AX_{t-1}]^T Q^{-1}[X_t-AX_{t-1}]\}(2\pi)^{-k/2}|Q|^{-1/2}\right\} \tag{9}$$

Assuming the initial state X_0 is constant, the state density of X_t is the linear combination of Gaussian random variables

$$p(X_t)=N(\mu_t,K_t) \tag{10}$$

with mean $\mu_t=A^tX_0$ and covariance $K_t=\sum_{i=0}^{t-1}A^iQ(A^i)^T$.

Let X be the sequence of t state vectors, the joint probability of X is also Gaussian

$$p(X)=N(\mu,K) \tag{11}$$

where $\mu=\begin{bmatrix}\mu_1\\ \mu_2\\ \vdots\\ \mu_3\end{bmatrix}$, $K=\begin{bmatrix}K_1 & (AK_1)^T & \ldots & (A^{t-1}K_1)^T\\ AK_1 & K_2 & \ldots & (A^{t-2}K_2)^T\\ \vdots & \vdots & \ddots & \vdots\\ A^{t-1}K_1 & A^{t-2}K_2 & \ldots & K_t\end{bmatrix}$.

Let the observation sequences be $Y=(Y_1,Y_2,\ldots,Y_t)$, then finally we get the probability of Y,

$$p(Y)=N(\gamma,\Sigma) \tag{12}$$

where $\gamma=C\mu$, $\Sigma=CKC^T+R$,

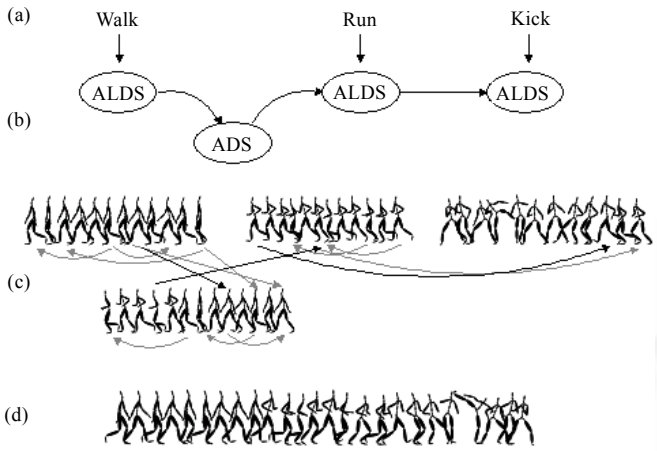
$$C=\begin{bmatrix}C & 0 & \ldots & 0\\ 0 & C & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & C\end{bmatrix},\quad R=\begin{bmatrix}R & 0 & \ldots & 0\\ 0 & R & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & R\end{bmatrix}.$$

The cross-entropy between two slices of LDS' measures the inconsistencies in their analysis of the data. Once the distance of two slices of texture has been computed by cross-entropy, the transition points are selected for those below an empirically determined threshold.

3.2 Creating dynamic texture graphs

We define the annotated dynamic texture graph (ADTG) structure on two levels. On high level a motion graph is a directed annotated graph with the nodes corresponding to LDS' and capturing the local dynamics. Those having correspondence to commands are described with labels. We collapse all the edges belonging to the same two LDS' in one direction. This yields a graph G where the nodes of G are motion clips and there is an edge from l_1 to l_2 for every pair of LDS' where we can cut from l_1 to l_2. We assume that the edges in G are attached to the detailed connection between the two LDS'. On low level, each node corresponds to the frame in LDS; each edge

corresponds to transitions occurred between LDS' or in the same LDS (Fig.5).



(A) User control stipulates what the sequence of action is
(B) High level structure is good for efficient searching and presentation of actions
(C) Low level retains the candidate transitions between frames
(D) New motion is synthesized based on high level order

Fig.5 An example of annotated dynamic texture graph (ADTG)

3.3 Two level data representation

Motion synthesis of motion-captured data can base on the frame level or on the clip level. Both of them have advantages: the former retains the detail and nuance of live motion, while the latter generalizes motion and encapsulates connections among clips. Our approach attempts to capture the strengths of both by combining them in a two level structure (Fig.5). Since there is correspondence between ALDS (annotated LDS) and control command, the high level structure is good for efficient searching and presentation in the interfaces. The low level is a detailed motion graph that retains the candidate transitions and generates new motion based on the high level demand of the user. The main advantages of our two level data structure are to offer high level control to the user by simply stipulating what the sequence of action is while low level provides smooth transition on frame level between two actions.

4 Results

We tested our approach on a data set containing 49 200 frames, or more than 15 minutes of motion sampled at 60Hz. These motion clips feature a variety of different kinds of actions, including walking, jogging, running, hopping, sneaking, and kicking; see Table 1 for a summary. All experiments were run on a machine with a 2.0GHz Pentium IV processor and 1GB of memory. User of our system can control the character to fulfill tasks interactively through interfaces such as mouse and keyboard.

Table 1 Motions learned and used in our experiments

Motion class	#Examples	Motion class	#Examples
Walk	46	Run	15
Kick	10	Jog	16
Sneak	8	hop	4

High level order synthesis. The example shown in Fig.1 suggests that the user can specify what kind of actions are to be performed at what times. When the user selects an action, the character performs that action. The real-time synthesized motion is created using motion capture data, the figure performs the actions that user

stipulated, which are walking, then running, and then hook-kicking.

Sketching path constraint synthesis. Given a path drawn by the user using a mouse and an action the character is required to perform, our system generates motion such that the character travels along the path. The path is projected onto the surface of the floor to provide desired travel trajectory. Our tracking algorithm fits the goal position along the desired trajectory and root position projected on the floor. In this example, we control the character sneaking along the indicated path. Refer to Fig.6 (The gray line is the path to be fit and the black line is an approximation of the actual path of the character).

Sketching path and action synthesis. Figure 7 illustrates that while the character travels along the path, the user can change the motion style. In this example, the user selects sneaking in the first section of the path, then changes the action to walking in the second section, finally selects running in the third section.

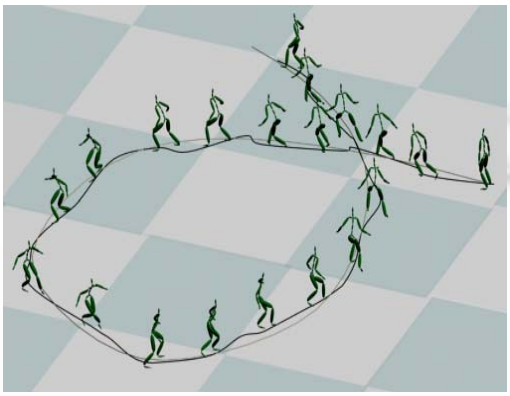


Fig.6 Image shows a motion synthesized to fit a path

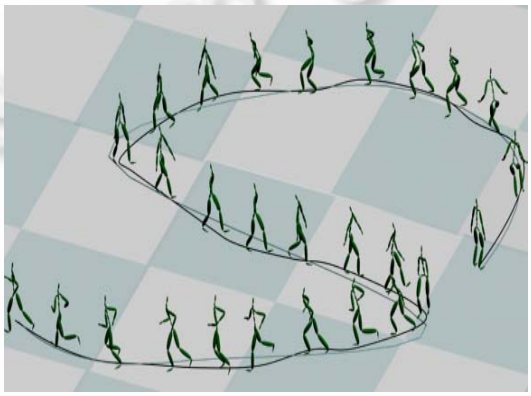


Fig.7 Image shows fitting a path wherein the character is required to sneak, then walk, and finally run

5 Conclusion

This paper has presented a general framework for creating natural-looking, controllable motion in interactive environments. A directed graph called annotated dynamic texture graph (ADTG) is proposed to encapsulate connections among local dynamics of character motion in the database, and to provide high level control of character behavior. This two-level data structure can be efficiently searched at run time to find appropriate paths to locations specified by the user. The good performance of our approach is supported by the experiment results.

References:

[1] Perlin K. Real time responsive animation with personality. IEEE Trans. on Visualization and Computer Graphics, 1995,(1):5−15.

[2] Witkin AP, Popović Z. Motion warping. In: Proc. of the SIGGRAPH. 1995. 105−108

[3] Gleicher M. Retargeting motion to new characters. In: Proc. of the SIGGRAPH. 1998. 33−42.

[4] Schödl A, Szeliski R, Salesin DH, Essa I. Video textures. In: Proc. of the ACM SIGGRAPH 2000. 2000. 489−498.

[5] Brand M, Hertzmann A. Style machines. In: Proc. of the SIGGRAPH 2000. 2000. 183−192.

[6] Kovar L, Gleicher M, Pighin F. Motion graphs. In: Proc. of the SIGGRAPH 2002. 2002.

[7] Lee J, Lee KH. Precomputing avatar behavior from human motion data. In: Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation 2004. 2004.

[8] Li Y, Wang T, Shum HY. Motion texture: A two-level statistical model for character synthesis. In: Proc. of the SIGGRAPH 2002. 2002.

[9] Arikan O, Forsyth DA. Interactive motion generation from examples. In: Proc. of the SIGGRAPH 2002. 2002. 83−490.

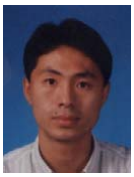
[10] Arikan O, Forsyth DA, O'Brien JF. Motion synthesis from annotations. ACM Trans. on Graphics (SIGGRAPH 2003), 2003,22(3): 402−408.

[11] Kim MJ, Kim MS, Shin SY. A general construction scheme for unit quaternion curves with simple high order derivatives. Computer Graphics (In: Proc. of the SIGGRAPH'95). 1995. 369−376.

[12] Fraley C, Raftery AE. How many cluster? Which clustering method? Answers via model-based cluster analysis. The Computer Journal, 1998,41(8):578−588.

[13] Soatto S, Doretto G, Wu YN. Dynamic textures. In: Proc. of the IEEE Int'l Conf. on Computer Vision. 2001. 439−446.

[14] Ljung L. System Identification, Theory for the User. 2nd ed. Prentice Hall, 1999.



GAO Quan-Sheng was born in 1968. He is a Ph.D. candidate at the Department of Computer Science, Harbin Institute of Technology. His current research areas are computer graphics and AI.

HONG Bing-Rong was born in 1937. He is a professor and doctoral advisor in Department of Computer Science, Harbin Institute of Technology. His research areas are robot soccer and AI.