HUMAN MOTION SYNTHESIS BASED ON ITERATED FUNCTION SYSTEMS

Stephen Wang-Cheung Lam and Powis Lai-Yin Leung

Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon.
Email: cswclam@comp.polyu.edu.hk *Fax: 2774 0842* *Tel: 2766 7301*

ABSTRACT

This paper describes a novel approach to automatic motion sequence generation. The basic algorithm underlying our approach is the *iterated function systems* (IFS) which have already found many applications in image compression and pattern generation. In our system, a set of *key-frames* are first specified by the user. Subsequently, IFS based interpolation scheme is employed to generate a sequence of motions having self-similar characteristics. Our ultimate purpose is to produce realistic repetitive motion, which possesses a main theme but the details varies over time. Our algorithm can find applications in generating human motion like dancing or walking which involves long sequence of repetitive movements of a puppet's limbs and body.

Keywords*: Iterated function systems, puppet, motion.*

1. INTRODUCTION

As stated in [Aus95], automatic motion synthesis is the problem of determining how an animated character should move in a plausible way that meets the animator's goals. Animated characters are often modeled as articulated figures, which are constructed by rigid rods connected by flexible joints. Generally speaking, physical realism is a good way to guarantee visual plausibility. Hence, traditionally, the automatic motion-synthesis paradigm requires the animator to specify both the physical structure of an articulated figure and the mechanism for driving it. The computer then evaluates a physically realistic motion sequence. Based on this paradigm, many techniques or algorithms have been developed. For example, human and animal character simulation based on inverse dynamics has been developed in [Gir85]. In [Bro88], researchers model the emotion of objects and their environment by differential equations obtained in classical mechanics. External control is then applied to satisfy the constraints imposed by the key-frames. Smooth and natural looking interpolations are obtained by minimizing a combination of the control energy and the roughness of the trajectory of the objects in 3D space. In [Cha89], artificial muscle layers are used for creating animated figures. The muscle deformations determine the resulting geometric surface of the character. In [Pan93], a sensor-actuator network (SAN) approach is proposed to control an articulated figure. The user needs to supply the configuration of a SAN, which is composed of a small non-linear network of weighted connections between sensors and actuators. Subsequently, a set of possible modes of locomotion of that particular object are evaluated. Further experiments and refinements of this method are described in [Aus95]. In [Grz95], a learning technique was developed which automatically synthesizes realistic locomotion for the animation of physics-based models of animals is developed. In [Bad93], simulated actors that embody true physical constraints are described in details. Finally, a survey of controlling articulated character animation and its related issues can be found in [Bad91].

Besides creating motion through simulating the underlying mechanics, recently other approaches are also proposed by researchers. For example, in [Unu95], a method for modeling human figure locomotion with emotions is presented. In that paper, Fourier expansions of experimental data of

actual human behaviors serves as a basis upon which the method can interpolate or extrapolate the human locomotion. In [Per95], rhythmic and stochastic noise functions are used to define time varying parameters that drive computer generated puppets. The ultimate purpose is to convey the texture of motion. With this algorithm, the computation of dynamics and constraint solvers is avoided.

As stated in [Hau96], fractal dynamics were recently detected in the apparently noisy variations in the stride interval of human walking. This inspires us to employ the *iterated function systems* (IFS) to generate repetitive animated motion sequence with textual characteristics. IFS were proposed in [Dem85] for generating irregular graphics patterns and have been applied for generating Chinese fonts [Ip94] and extended to three-dimensional space [Che97]. In [Per93], stochastic IFS and their applications in image coding are explored in details.

The remainder of this paper is structured as follows. In section 2, we present a brief review of iterated function systems (IFS). In section 3, we apply the IFS in the context of human character animation. Similar to the noise adding approach stated in [Per95], our algorithm can produce noisy motion sequence. Additionally, the core structure of the repeating motion pattern can be specified by the user through key-frames. In section 4, experiments based on our approach are presented. Finally, in section 5, we summarize the contributions in this research and suggest some topics for future work.

2. ITERATED FUNCTION SYSTEMS

Iterated function systems (IFS) can be classified into deterministic and stochastic ones [Hau96][Per93]. In this section, we briefly review the fundamental concepts and definitions developed in *deterministic IFS* which will be employed in our interpolation scheme[1].

2.1 Hausdorff Distance

Given a metric space (X,d), a new metric space (H(X), h) can be constructed where H(X) is the collection of non-empty, compact, subsets of X, and h is the Hausdorff metric defined as follows:

For A and B in H(X) and x in A, we define the distance from x to B as:

$$d(x,B) = \min_{y \in B} d(x,y) \quad (2.1.1)$$

where $d(x,y) = \left\{ \sum_{i=1}^{n} (y_i - x_i)^2 \right\}^{1/2}$. Since $y \mapsto d(x,y)$ is a continuous function and B is compact and nonempty, the above minimum is able to attain. Then, we define the distance from A to B as:

$$d(A,B) = \max_{x \in A} d(x,B) \quad (2.1.2)$$

For the same reason as above, this maximum is also attained. Finally, the Hausdorff distance between A and B is defined as:

$$h(A,B) = \max(d(A,B), d(B,A)) \quad (2.1.3)$$

2.2 Contractive Transformation

Given two metric spaces (X,d_1) and (Y,d_2), a transformation $w: X \mapsto Y$ is said to be a contraction if and only if there exists a real number s, with $0 \le s \le 1$, such that:

$$d_2(w(x_1), w(x_2)) \le s\, d_1(x_1, x_2),$$

for any $x_1, x_2 \in X$. (2.2.1)

Any such number s is called a *contractivity factor* for w. w is said to be a strict contraction, if s is strictly less than one. Note that, when the two metric spaces coincide, w acts on pairs of points in X by bringing them closer together, their distance being reduced by a factor of at least s.

2.3 Defining IFS

Given a complete metric space (X,d), we shall consider the associated space (H(X), h) of nonempty, compact subsets of X, endowed with the Hausdorff metric, and define a contractive transformation of W of H(X) into itself. By the constraction mapping theorem, W has a unique fixed point in H(X). In particular, for the case $X=\mathbf{R}^2$, this construction will enable us to identify certain fixed points of constractive transformation $H(\mathbf{R}^2)$ into itself.

Firstly, we consider a metric space (X,d) and a finite set of strictly contractive transformations $w_n: X \mapsto X$, $1 \le n < N$, with corresponding contractivity factor S_n. Then, we proceed to define a transformation $W: H(X) \mapsto H(X)$, where H(X) is the

[1] Most of the materials presented in this section are taken from [Per93].

collection of nonempty, compact subsets of X by:

$$W(B) = \bigcup_{n=1}^{N} w_n(B) \quad \text{for any } B \in H(X) \quad (2.3.1)$$

It is easily shown that W is a strict contraction, with contractivity factor $s = \max_{1 \le n \le N} s_n$. According to the contraction mapping theorem, if (X,d) is complete, W has a unique fixed point A in H(X), satisfying the remarkable self-covering condition

$$A = W(A) = \bigcup_{n=1}^{N} w_n(A) \quad (2.3.2)$$

Finally, we give the following definition.

Definition 2.1. A hyperbolic iterated function system (IFS) $\{X; w_1,\ldots,w_n\}$ consists of a complete metric space (X,d) and a finite set of strictly contractive transformations $w_n : X \mapsto X$ with constractivity factors s_n, for n=1,..,N. The maximum s among $s_1,\ldots,s_N$ is called a contractivity factor for the IFS. The unique fixed point in H(X) of the transformation W defined by relation (2.3.1) is called the contractor of the IFS.

For examples, please refer to [Per93].

3. MOTION SEQUENCE GENERATION

In this section, we will explain the configuration of our puppet and the interpolation scheme for generating motion sequences.

3.1 Puppet Configuration

As shown in figure 3.1, eighteen components are used in constructing our puppet. We have included the head, neck, shoulder and so on. Ideally it should be more; this is the minimum to provide the best emotional expressiveness. Each component is represented by a rod, which connects two points in three-dimensional space. The coordinates of these end-points, which change at every frame, drive the puppet's movement. We provide a motion script interface, which resembles the linear-list notation [Fol90]. A statement such as **4 LUpperarm RotateZ 3,** means "*in frame 4, the puppet's left upper arm should be rotated about Z axis for 3 degrees*". Figure 3.2 illustrates a segment of a script, which specifies the initial six frames, with key-frames 1, 3 and 6. The puppet's posture in each key-frame is specified by a group of statements. The first statement of a group is preceded by a number, which identifies the corresponding key-frame.

3.2 Motion Sequence Generation Scheme

In this section, we present our motion sequence generation algorithm.

3.2.1 Transformation

Each component of the puppet can move in two opposite directions, i.e. positive and negative. Figure 3.3 illustrates the posture requirements of some of the components in a key-frame. For example, the left upper arm should attain a position effectively equivalently to after rotating about x-axis for 40 degrees. Figure 3.3(a) shows a bar chart, which represents the initial set of data D_0. We define $T^+(D_i)$ as the positive transformation function for making all the data values positive and $T^-(D_i)$ doing the reverse. D_i is the data set of key-frame i.

3.2.2 Generating a New Key-frame

Our motion sequence generation algorithm is shown as follows:

for each data value α_{ij}^{p} in a set D_i,

if $\alpha_{ij}^{p} > 0$ then $D_i' = S(S_N(T^+(D_i)))$

if $\alpha_{ij}^{p} < 0$ then $D_i' = S(S_N(T^-(D_i)))$

set α_{ij}^{p+1} to the mean of $D_i' \cup \alpha_{ij}^{p}$.

The whole sequence is composed of a set of frame sequences. Each frame sequence is indexed by p. j is the index of the data value of each key-frame i. S_N is a scaling function for normalizing the transformed value. Figure 3.4(a) shows the normalization process. S is a scaling function for scaling the normalized data values. The scaling magnitude is specified by the user: S ranges from R_{min} to R_{max}, where R_{min} and R_{max} are the parameters input by the user. For example, R_{min} and R_{max} could set to 0.6 and 1.4 respectively. S is then generated randomly within the range. Figure 3.4(b) shows the effect of this function.

In order to avoid impossible or abnormal movements, we need to check α_{ij}^{p+1}. The following is the checking rules:

if $\left|\alpha_{ij}^{p+1}\right| > \left|\alpha_{ij}^{p}\right|$ and $\left|\alpha_{ij}^{p+1} - \alpha_{ij}^{p}\right| > \left|\alpha_{ij}^{p} \times \lambda^{u}\right|$

then α_{ij}^{p} is not acceptable and is set to $\alpha_{ij}^{p} \times \lambda^{u}$

if $\left|\alpha_{ij}^{p+1}\right| < \left|\alpha_{ij}^{p}\right|$ and $\left|\alpha_{ij}^{p+1} - \alpha_{ij}^{p}\right| > \left|\alpha_{ij}^{p} \times \lambda^{l}\right|$

then α_{ij}^{p} is not acceptable and is set to $\alpha_{ij}^{p} \times \lambda^{l}$

where λ^{u} and λ^{l} are user specified percentage values.

As shown in the above rules, the user is required to specify an upper and a lower limit on the change of movement. If the new evaluated value exceeds the range, it is replaced by the limit value. An example is shown in table 3.1

3.3 User Interface Design

Figure 3.5 shows the user interface of our system. The user can switch on and off the viewing windows and the motion sequence window through the main controller panel. The motion parameters are input through the motion controller panel. The user can select different motion sequence like swimming, walking or dancing. Play, pause, stop and repeat buttons are provided for controlling the playing of animation interactively. Furthermore, the user can select different viewing effect from the viewing effect block. If the capture button is pressed, the sequence of motion is captured and displayed in the motion sequence motion window. Finally, the user can also input different parameters for controlling the IFS based motion synthesis if the IFS button is pressed.

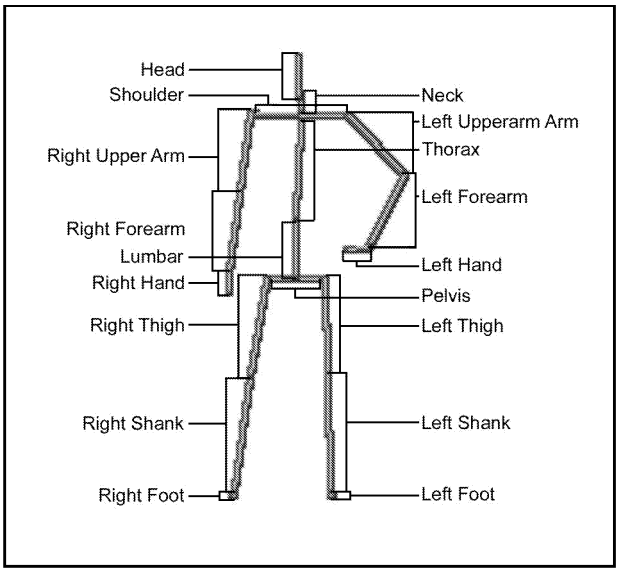


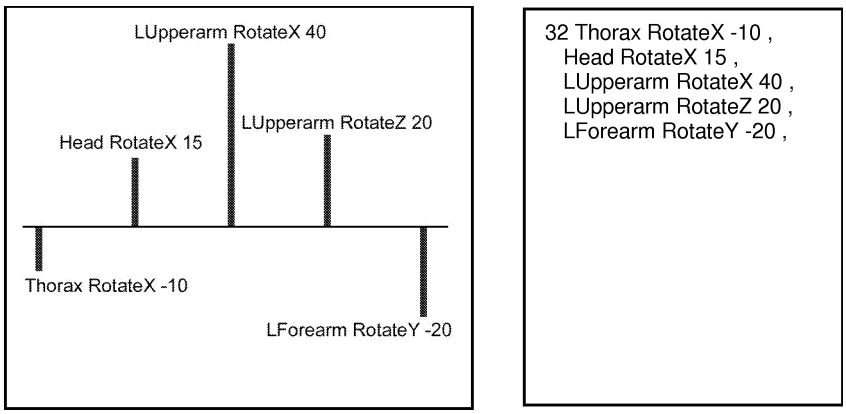
Figure 3.1. A puppet with eighteen components.

```
1 Puppet MoveY -3 ,
  Puppet MoveX -2 ,
  Shoulder RotateY 5 ,
  RUpperarm RotateX -15 ,
  LUpperarm RotateX 25 ,
  Pelvis RotateZ -4 ;
3 Puppet MoveX 1 ,
  Puppet MoveY 2.5 ,
  Shoulder RotateY -2.5 ,
  RUpperarm RotateX 20 ,
  RForearm RotateX -20 ;
6 RShank RotateX 7 ,
  RFoot RotateX 8 ,
  LThigh RotateX -7 ,
  LShank RotateX -3 ,
  LFoot RotateX -3 .
```

Figure 3.2. A motion script example.

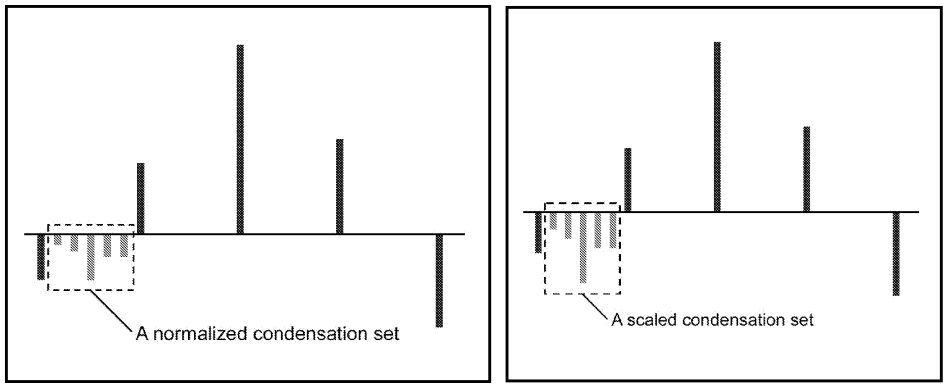
Component	Action	Original value	Evaluated value	Upper Limit	Limit Value	Final value
Lupperarm	RotateZ	30	40	120%	30x120%=36	36

Table 3.1. An example of value adjustment.



(a) (b)

Figure 3.3. An example key-frame. (a) A bar chart. (b) The script of the key-frame.



(a) (b)

Figure 3.4. Construction of interpolated values. (a) Normalization and (b) Scaling of the interpolated values.

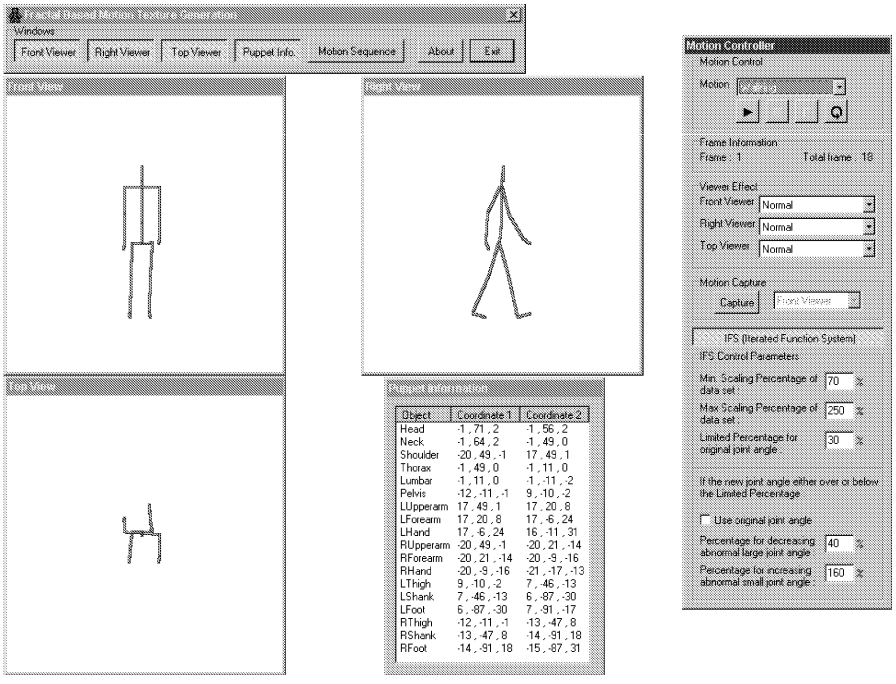


Figure 3.5 This figure shows the user interface of our IFS based motion generation system.

4. EXPERIMENTS

Two experiments have been carried out to test the effectiveness of our algorithm.

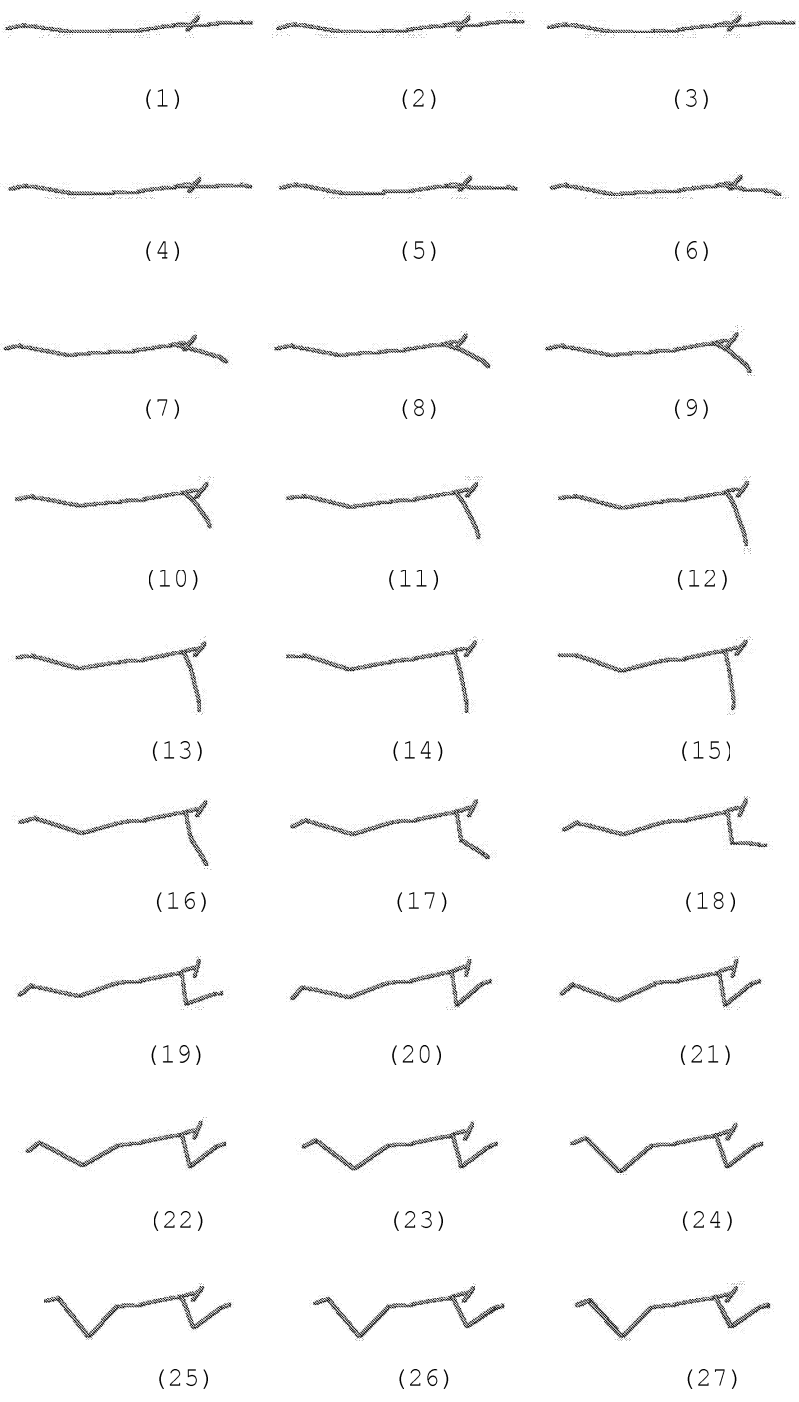


Figure 4.1. Swimming. Key-frames are 1,5,10,15,20,25, and 27.

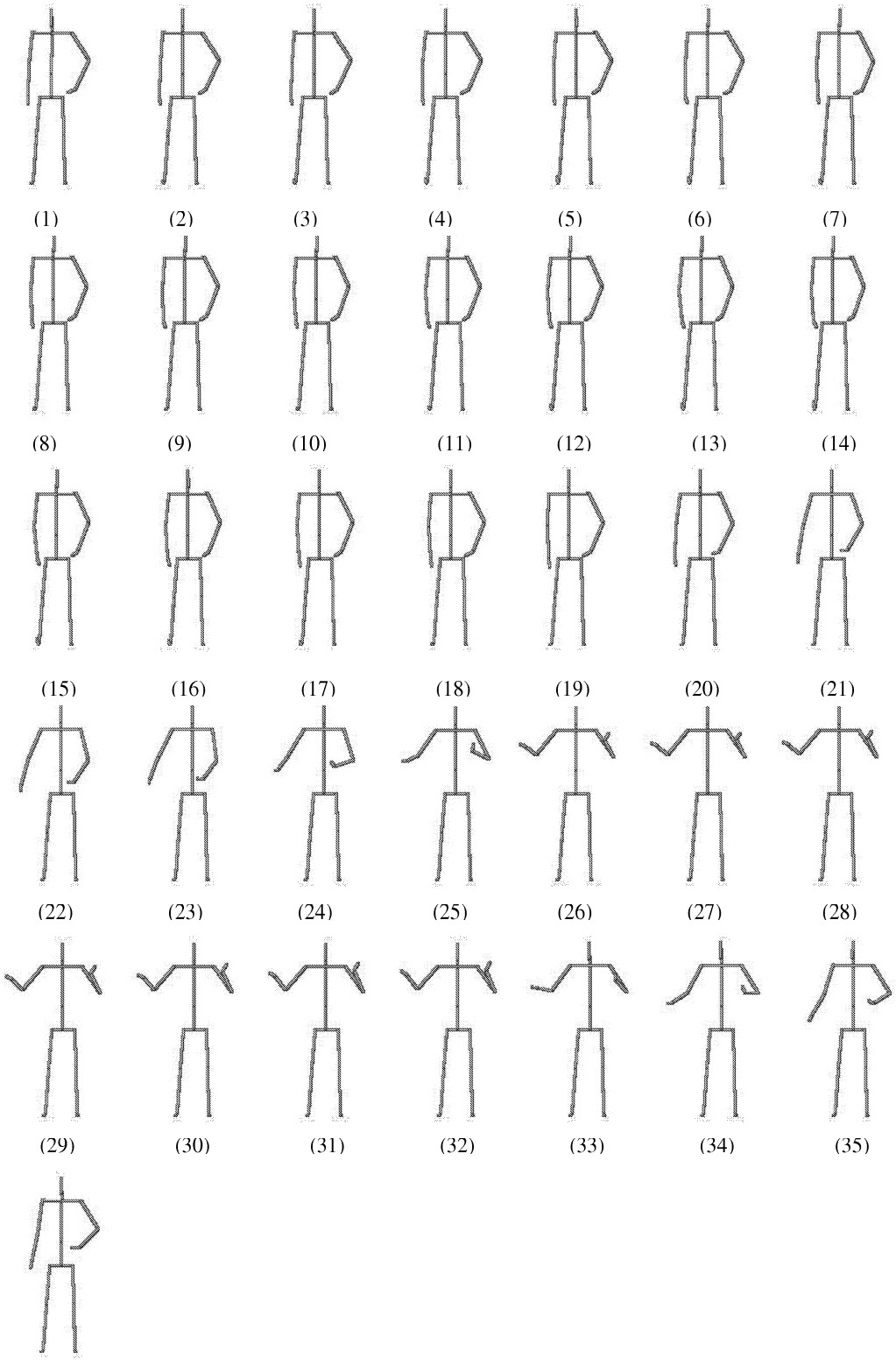


Figure 4.2. Character shows the message “I don’t know” (more obvious in frame 26-32). Key-frames are 1,10,14,18,23,28,32 and 36.

We present the first set of generated frames in figures 4.1 and 4.2. Some of the consecutive frames may look idle. This is owing to the fact the motion is in three-dimensional space and the corresponding motions can only be visible from other viewing directions. Although we have not depicted the non-IFS versions owing to the limit of the length of this paper, the overall control mechanism of our algorithm can still be observed from the figures.

5. CONCLUSIONS

Currently, repetitive motion like dancing is generally created by simply repeating a short motion sequence many times. Most likely, the resulting motion sequence looks boring and unnatural. We conjecture that, this is because fractal characteristics (i.e. irregular and self-similar) are not possessed by the sequence. Based on the idea of motion texture [Per95] and inspired by the recent research of human walking behavior [Hau96], we attempt to employ the notion of *iterated function systems* in the image sequence interpolation process. (Owing to the limitation of this paper's length, the details of the motion computations are not shown here.) The result is a sequence of motion, which appears more nature to the viewer. Similar to [Per95], by conveying just the texture of motion, we are able to avoid computation intensive dynamics and constraint solvers. We believe this approach has the potential to create many lifelike repetitive motion sequences, for instance, swimming, walking, jogging and many others. It is difficult to compare our approach to others as, to the authors' knowledge, we are the pioneers in this topic. Finally, in the future, we plan to enhance our algorithm by employing anti-aliasing methods, which are tailor-made for fractal objects (e.g. [Har91]) to remove aliasing effect in the motion sequence generated by us.

REFERENCES

[Aus95] Auslander, J., A Fukunaga, H Partovi, J Christensen, L Hsu, P Reiss and A Shuman, Further Experience with Controller-Based Automatic Motion Synthesis for Articulated Figures, *ACM Transactions on Graphics*, Vol. 14, No. 4, 1995, 311-335.

[Bad91] Making Them Move: Mechanics, Control, and Animation of Articulated Figures, *San Mateo, Calif: Morgan Kaufmann Publishers*, edited by Badler, N. J., B. A. Barsky and D. Zeltzer, 1991.

[Bad93] Badler, N. J., C. Philips, and B. L. Webber, Simulating Humans: Computer Graphics, Animation, and Control, *Oxford Univ. Press*, 1993.

[Bro88] Brotman, L. S. and A. N. Netravali, Motion Interpolation by Optimal Control, *Computer Graphics*, Vol. 22, No. 4, 1988, 309-315.

[Cha89] Chadwick, J. E., D. R. Haumann and R. E. Parent, Layered Construction for Deformable Animated Characters, *Computer Graphics*, Vol. 23, No. 3, 1989, 243-252.

[Che97] Chen, Y Q and G Bi, 3-D IFS Fractals As Real-Time Graphics Model, *Computers & Graphics*, Vol. 21, No.3, 367-370, 1997.

[Dem85] Demko, S., L. Hodges and B. Naylor, Construction of Fractal Objects with Iterated Function Systems, *Computer Graphics*, Vol. 19, No. 3, 1985, 271-278.

[Fol90] Foley, Van Dam, Feiner and Hughes, Computer Graphics: Principles and Practice, 2nd edition, Addison Wesley, 1990.

[Gir85] Girard, M. and A. A. Maciejewski, Computational Modeling for the Computer Animation of Legged Figures, *Computer Graphics*, Vol. 20, No. 3, 1985, 263-270.

[Grz95] Grzezcuzuk, R. and D. Terzopoulos, Automated Learning of Muscle-Actuated Locomotion Through Control Abstraction, *Computer Graphics*, 1995, 63-70.

[Har91] Hart, J. C. and T. A. DeFanti, Efficient Antialiased Rendering of 3-D Linear Fractals, *Computer Graphics*, Vol. 25, No. 4, 1991, 91-100.

[Hau96] Hausdorff, J M, et. al., Fractal Dynamics of Human Gait: Stability of Long-range Correlations in Stride Interval Fluctuations, *Journal of Applied Physiology*, 80(5), 1996, 1448-1457.

[Ip94] Ip, H H S, H T F Wong and F Y Mong, Fractal Coding of Chinese Scalable Callographic Fonts, *Computers & Graphics*, Vol. 18, No. 3, 343-351, 1994.

[Pan93] Panne, M. V. D. and E. Fiume, Sensor-Actuator Networks, *Computer Graphics*, 1993, 335-342.

[Per93] Peruggia, M., Discrete Iterated Function Systems, *A K Peters*, 1993.

[Per95] Perlin, Ken, Real Time Responsive Animation with Personality, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1, No. 1, March 1995, 5-15.

[Unu95] Unuma, M., K. Anjyo and R. Takeuchi, Fourier Principles for Emotion-based Human Figure Animation, *Computer Graphics*, 1995, 91-96.