JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 19, 1071-1085 (2003)

DOI:10.1688/JISE.2003.19.6.10

Short Paper

The Design and Implementation of Realistic Fish Animation Play System Using Video Sequence*

TONG-YEE LEE AND HUNG-YI CHEN
Computer Graphics Group/Visual System Laboratory
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, 701 Taiwan

In this paper, a novel system for rendering and animating fish is proposed. We captured a short video sequence of fish, and analyzed the similarities between these fish in video frames. In order to control the motions of the fish interactively as in computer games, we attempted to learn the motion properties of the fish in this video sequence. Some of the motion parameters of the fish were extracted from the video frames and then trained using a neural network. This trained information provided motion control for generating fish animation. In the proposed system, we also generate arbitrary video loop lengths for rendering and animating fish. It is not necessary that the video samples in these video loops be played in sequence. Furthermore, we can adapt both the video loop and neural network techniques in order to control the fish motions interactively and to maintain visual smoothness in fish animation.

Keywords: fish animation, neural network, video-loop, position table, gateway

1. INTRODUCTION

In the early days, computer animation appeared as successive 2D images, as in cartoons. Now, three-dimensional animation has taken over the commercial entertainment market. However, regardless of how much researchers have improved computer animation, it is still difficult and laborious to attain compellingly realistic objects in all cases. Although texture-mapping techniques increase the level of realism in modeling and rendering, the object still appears unnatural in most situations. Recently, researchers in computer graphics and computer vision have proposed efficient methods to generate novel views by analyzing captured images. These techniques, called image-based rendering, demand user interaction [3] and permit photo-realistic synthesis of static scenes. In this paper, we propose a new method is an image-based rendering technique. With it we have attempted to create natural and photo-realistic fish animation.

A sequential fish video captures the time-varying behavior of fish at very specific points in time. We took advantage of this characteristic of fish videos and analyzed fish

Received August 19, 2002; revised February 6, 2003; accepted February 27, 2003.
Communicated by Kuo-Chin Fan.
* This project is supported by National Science Council, Taiwan, R.O.C., under contract No. NSC-91-2213-E-006-077.

movement information within consecutive frames. We then generated realistic fish animation by re-ordering the samples from the original video sequence. The generated animation sequence needed to be both visually smooth and interactively controlled.

In our experiments, we placed our video camera on the upper side of a fish tank to capture a carp on video. A carp's body changes while it is turning. What a video sequence conveys through consecutive video images is an interesting topic of study. Because consecutive images are recorded on video, the fish's motion shows distinct changes frame by frame. There are various similarities among these consecutive frames, such as the object positions and colors. In a video sequence, the trajectory, regular behavior, and reflection caused by a swimming fish also have certain similarities. By exploring these properties, we can study the idiosyncrasies of fish behavior observed in the input video. In our design, these properties are parameterized using movement displacement, the degree of turn, head orientation etc. In order to understand the variations in fish locomotion, all of these parameters were trained in this study using a back-propagation neural network [4]. Based on these trained parameters, a fish motion control system was designed. By providing reasonable control parameters, the control system recreates realistic fish behavior. The control system is derived from the video itself. This control system truly represents the fish behaviors that were recorded in the video.

In [1], Schödl et al. proposed a new type of medium called video textures, which provide continuous, infinitely varying streams of video images. The basic concept of video textures can be extended in several different ways to further increase its applicability. Among the other concepts that Schödl et al. proposed, the most important one is that of video loops. Video loops are extracted directly from a video and can be played continuously without any visible discontinuities. However, this technique is not suitable when animators need to control specific objects with arbitrary movements. We solve this problem by introducing a table, called a video loop position map. This table records the orientations of the fish in each video loop. If the video is too short for us to extract enough video loops for rendering, or if the selected video loops have some unavoidable limitations, the proposed neural network method is used to compensate. Our main contributions are listed as follows:

1. Through our methods, the motion of fish within a video can be analyzed and parameterized. The neural network is utilized to learn the fish motion model. All of the procedures used in the proposed system are executed automatically.
2. Neural network training could be performed off-line. Animation from the trained motion model can be generated in real-time.
3. A position map is used to select suitable video loops. New animation can be generated using video loops as well. In contrast to [2], all video loops are extracted automatically without manual labeling.
4. By combining the neural network and video textures modes, we were able to design a complete fish animation play system. Our system attempts to animate fish interactively as well as to maintain visual continuity.

1.1 Related Work

As Schödl et al. pointed out in [1], there has been little previous work on automati-

cally generating motion using captured video. Fortunately, the authors proposed the video sprite [2], a type of video texture. In [1], they had generalized image-based rendering in the temporal domain, but, in their recent work [2], they also noted that the core algorithms used in video textures are not suitable for specific moving objects. In order to overcome this problem they proposed, video sprites with control [2] that can be rendered anywhere on the screen to create a novel animation. However, they manually labeled the training data to train the classifier. In most cases, manual classification is difficult to evaluate. Evaluating a method, which requires manual operations, is tedious and time consuming. For this reason, we propose an automatic learning method that could objectively represent fish motion. In [2], the captured video had to be long enough to provide enough frames for manual processing. However, the motions of fish in a tank are very monotonous, and sometimes, fish motion patterns are not sufficient to generate arbitrary animation. Therefore, in [2], the amount of time required to obtain adequate numbers of video frames was unpredictable.

In addition to [1, 2], a few studies have been done that are closely related to our research. In this paragraph, research work by others that inspired our study will be discussed. Tu et al. [5] proposed a framework for animation with minimal input from the animator. They presented a method that makes an artificial creature move by defining the principal motion patterns of fish and by providing the parameters that cause the motion. They defined a physically based, virtual marine world inhabited by artificial fish. These motion patterns are determined using a few motion state variables. There are, however, problems involved with the ease of use and generality of each variable. A method to automatically learn a fish's motion was proposed in [6]. Because the above methods use very complicated algorithms, it is difficult to generate motions in real time. Terzopoulos et al. [7] proposed a new method called NeuralAnimator that utilizes a neural network to simulate a physically based model. They replaced the conventional animation simulators with numerical simulations. Ziv Bar-Joseph et al. [8] and X. Wen et al. [9] introduced wavelet techniques for analyzing and synthesizing video. Finkelstein et al. [10] presented a new approach called multiresolution that provides a means of capturing time-varying image data produced at multiple scales, both spatially and temporally. Takahashi et al. [11] and Lipton [12] focused on video objects and studied object behaviors. Takahashi et al. proposed an estimation method for capturing fish positions and postures from video using an object-matching technique. Lipton presented a paradigm for data interaction called virtual video that can be interactively augmented in real-time.

1.2 System Overview

In the proposed system, there are two major play modes: the first is the neural network play mode. The second is the video loop play mode. The proposed system is shown in Fig. 1. As shown in this figure, an input source video clip is decomposed into single frames. The fish motion parameters are extracted from these images. The neural network uses these motion parameters to design a fish motion control model. By means of these frames, fish images can be synthesized for all orientations, and their similarities can be analyzed to determine possible video loops. The gateways between the synthetic images and source video clip, which are obtained by examining their resemblances, are used for switching between these two kinds of play modes.

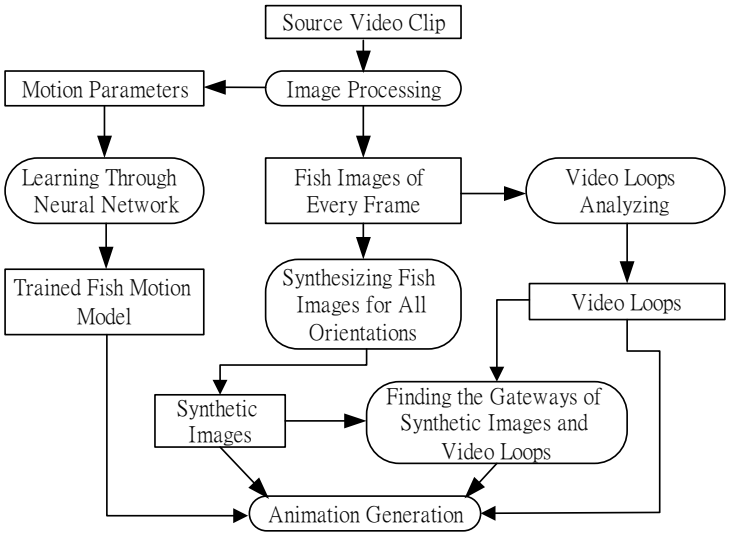


Fig. 1. System overview diagram.

The remainder of this paper is organized as follows. Section 2 describes methods used to obtain motion parameters from a fish video. In section 3, several parameters used to describe fish movement characteristics are given. Using these parameters, a fish motion model is trained by means of a neural network. In section 4, a method for automatically detecting video loops is proposed. We also discuss some limitations involved in using video loops. Solutions to these limitations are also presented. Our results are discussed in section 5. Our conclusions and future research direction are presented in section 6.

2. MOTION PARAMETERS

In our approach, fish images are segmented from a source video. The fish motions are then analyzed. Based on this analysis, the motion parameters are used to train a fish motion control system using a neural network. In this section, the segmentation method and the motion parameter measurements will be discussed.

2.1 Displacement

Many methods have been proposed for extracting objects for image segmentation [13]. In our case, a fish is in a fish tank. A simple background subtraction scheme is chosen for extracting the fish images. This method works well in general cases. When the fish swims near the fish tank walls, this method does not work. We locate the fish global coordinates by computing the center of mass of the fish. The fish's displacement can be measured using the difference in the fish coordinates in two consecutive frames. This is an important parameter of fish motion. The displacement varies as a fish moves. In addition to the above parameters, the positions of the fish's head and tail must also be located.

2.2 Motion Area

In our experiment, a carp video was captured. A carp swims in a manner similar to that of a horse mackerel. In this manner of swimming, the fish deforms very much when it turns around. Therefore, we define another measurement called the motion area and use it to parameterize the variations in the fish's motion. The motion area measurement method is described as follows. First, we apply the image-thinning method [13] to the original fish images. After thinning is performed, a thin line remains in the fish image, which looks like the backbone of a fish. Four points are placed at 1/6th, 1/3th, 1/2th and 5/6th parts of this thin line and on the head and tail. These six control points (shown as blue points) are then used to create a B-spline curve. The area enclosed by the B-spline is measured as the motion area that can objectively show the magnitude of the deformation of the fish. The choices for these six control points are based on observations of a carp's motion.

2.2.1 Branch pruning

In our experiment, if a fish unfolds its pectoral fin, after thinning is performed on the fish image, there will be several branches along the backbone.

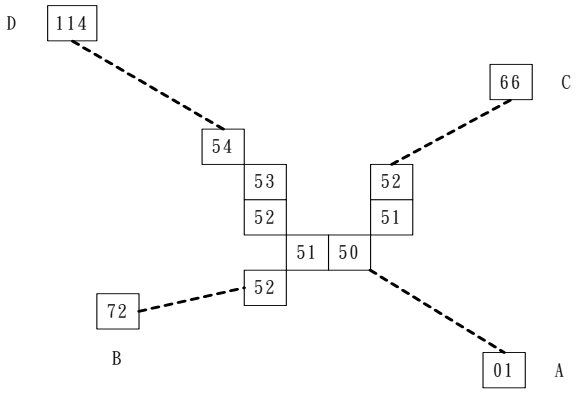


Fig. 2. Among these branches, only the one with highest number will be selected as the end point for the backbone of the fish. In this figure, the path from A to D is the backbone.

We prune the branches in two passes. In the first pass, a number is labeled as the initial at the beginning (i.e., fish head) of the backbone. Increasing numbers are assigned to the pixels along this backbone. Once this process encounters the branches, each branch is assigned an increasing and identical number. The labeling process continues along each branch until the end of the branch is reached. The end of each branch will receive a number. Among these end points, the end point with the highest number will be chosen. In the second pass, we start to trace from this end point back to the beginning point (i.e., the fish head). While tracing along the path, we trace from a pixel with a large number to a pixel with a small number. After we finish tracing, the traced path becomes the backbone of the fish. For example, in Fig. 2, the process starts at point A and assigns to it an initial number, 1. Numbers are assigned in an increasing order along the backbone. When

the number is 50, the process meets a branch and assigns 51 to each joint branch. The process continues until the end of each branch is reached. Point D is the final point, and the tracing process then proceeds backwards to point A. The path from D to A is the backbone.

2.3 Angle of Orientation

The motion area is used to measure the deformation of the fish's body when the fish turns. While the fish is turning, its shape always changes from almost straight to twist. Not only the posture, but also the orientation of the fish gradually varies. In Fig. 3, there are three images (3a, 3b, 3c) captured at different time slots. From these motion area images (3a', 3b', 3c'), we observe that both the motion area and head orientation keep changing. The angle between the current head orientation and previous orientation can represent the magnitude of the twisting that occurs while the fish is turning. The relationship between this angle and the fish's motion will be discussed in the next section.

(a) (a') (b) (b') (c) (c')

Fig. 3. The top three pictures, (a), (b), and (c), were taken at different time slots, t_0, t_0+k, and t_0+2k, and the bottom three, (a'), (b'), and (c'), show the motion area measurements, circled by a red line. If the fish is turning, the corresponding motion area is greater than that when the fish is swimming straight ahead.

3. NEURAL NETWORK PLAYING MODE

In section 2, we described several parameters related to fish motion. In this section, we will present a fish motion control model that employs these parameters. We adopt a neural network to analyze these parameters. The learning method and the method for applying the trained motion model to generate animation will be presented. We regard the trained model as an "artificial fish." This trained model returns the proper fish state for the given control parameters. Sometimes, unavoidable problems occur during rendering.

We solve these problems by synthesizing the fish images. In this paper, the kind of animation generated using this trained model is called "playing in the neural network mode."

3.1 Learning by Means of the Neural Network

Given the fish states and control values, a feasible reaction can be obtained from our fish motion control model. Our input parameters are separated into two classes: state inputs and control inputs. The state inputs are created using the current swimming type and motion area. We use the differences between the current state and previous state as the control inputs. In Eq. (1), for example, u and s stand for the control vector and state vector, respectively. In this equation, δ is the time interval between two sampling frames:

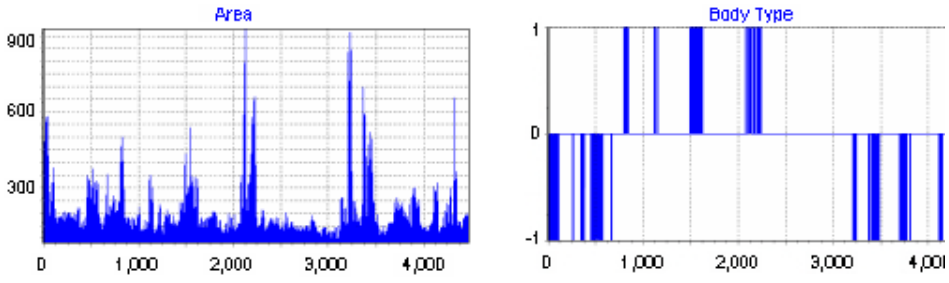
$$u_t = s_{t+\delta} - s_t \tag{1}$$

We used the back-propagation algorithm [4] in our experiment. There are three layers in our network. Back-propagation is a practical, recursive method used to calculate the component error derivatives of the gradient term. Applying the differentiation chain rule, the back-propagation algorithm first computes the derivatives with respect to the weights in the output layer and chains its way back to the input layer, computing the derivatives with respect to the weights in each hidden layer as it proceeds. The more complex mathematical principles of neural networks, such as the general delta rule and some other optimization methods, will not be discussed in this paper. More information about back propagation networks can be found in [4].

3.2 Selecting Parameters of Motion Model

Training the motion model using an input video clip is an innovative approach to computer animation. In addition to the above mentioned current states and control inputs that can be obtained from a video clip, the desired output is obtained from the next state of the fish that can be determined from the input video clip. In this way, the trained model truly reflects the characteristics of the fish motion depicted in the input video sequence. Parameter selection is quite important with respect to the neural-network training model. The training model will converge quickly if the selected parameters represent the features of the training data. The parameters chosen for our experiments were described in Section 2. The variations in all of these parameters objectively correspond to the fish motion, as shown in Fig. 4. The length of the input video clip in our experiment was about 3 minutes. There were 4480 frames in the source video clip. Fig. 4(a) shows the motion area measured on these frames. Fig. 4(b) shows the swimming mode of the fish. The remaining parameters show similarities within the motion area. During the period from frame 3150 to frame 3200, the fish turned right, and the corresponding motion area was much larger than that found when the fish was swimming straight ahead. During the period from frame 1600 to frame 2050, the fish swam straight, and its mean motion area was small.

With these parameters, which are highly relevant to fish motion, we could obtain an accurate fish motion control model. We chose the fish swimming mode and the motion area as the state parameters (see Eq. (2)). The angle of the fish's head orientation and the



(a) (b)

Fig. 4. (a) The motion area of the input video clip. The y-axis is the magnitude of the motion area, and the x-axis is the number of frames. (b) The swimming mode of the fish. The y-axis is the swimming mode of the fish. When the fish is swimming straight ahead, y is 0. When it is turning left, y is 1. When it is turning right, y is –1.

displacement were chosen as the control parameters (see Eq. (3)). The training model was designed using Eq. (4) with state inputs and control inputs. The training process convergence, which will be discussed in section 5, shows that this framework is practical and works well.

$$\text{State}_t = \begin{bmatrix} \text{Area}_t \\ \text{Type}_t \end{bmatrix} \tag{2}$$

$$\text{Control}_t = \begin{bmatrix} \text{Displacement}_t \\ \text{Angle}_t \end{bmatrix} = \begin{bmatrix} \left\| \text{Position}_{t+\delta} - \text{Position}_t \right\| \\ \text{Included Angle}(\text{Dir}_{t+\delta}, \text{Dir}_t) \end{bmatrix} \tag{3}$$

$$\text{State}_{t+\delta} = \text{BPN\_Model}(\text{State}_t, \text{Control}_t) \tag{4}$$

3.3 Synthesizing Fish Images

We expect to obtain a proper state output by giving reasonable control parameters. Because all of the parameters are observable features in video frames, this makes the control function simpler. In the next section, we will use appropriate fish images to illustrate rendering and animation. In the 2D plane, we define 24 orientations to organize the fish images. From our observations, a few fish images involve some orientations, while many fish images involve other orientations. It is important to have enough images for all fish orientations. Fish do not appear frequently enough in every orientation. For certain specific orientations, the fish only swims straight ahead or never turns right or left. For these reasons, it is sometimes difficult to render smooth animation because of the limited number of images for every orientation. Of course, if we continue capturing the fish for a longer period of time, enough motion information for every orientation can be captured. In order to solve this problem, we have devised an alternative approach. First, we capture successive fish images involving a fish swimming straight ahead and turning left and right. We then synthesize more images in every orientation by rotating these selected images. In this manner, we can capture enough fish images for rendering and animation.

4. VIDEO LOOP PLAYING MODE

A single transition i→j, from the source frame i to the destination frame j, is called a video loop if frame i is similar to frame j. During rendering, when we play frame j, we can then jump to frame i to replay a loop without causing visual discontinuity. This video loop can be repeated many times if required. Because the fish swims according to a regular pattern, there is a great possibility of extracting many video loops from a video clip. In this section, the video loop will be analyzed, and a method for using video loops to synthesize a novel animation will be proposed. However, several problems occur when video loops are used. We will present solutions to these problems.

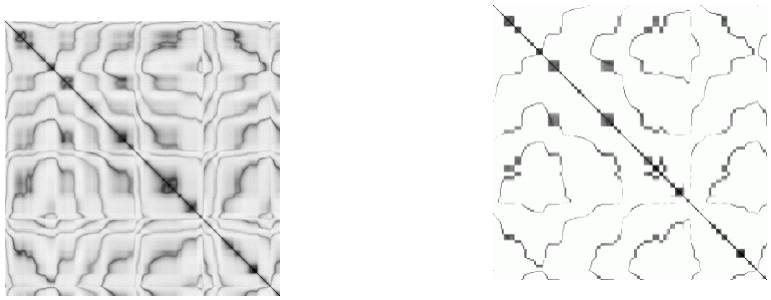
4.1 Finding Video Loops

The first step in finding a video loop is to compute the similarities between two fish

image pairs in the video sequence. These images can be nonconsecutive in the time domain. To easily select video loops, we compute a distance map (or difference map) for every pair of images in a video clip. This distance map is mathematically defined as follows:

$$D_{ij} = \left\| F_i - F_j \right\|_2 , \tag{5}$$

where F_i, F_j is the frame number and D_{ij} is the magnitude of the L_2 distance (Schödl [1]).



(a) (b)

Fig. 5. *Distance Map.* (a) is drawn by applying Eq. (5); that is, the distance between every two images is calculated to determine whether the fish properties are similar or not. (b) is drawn by applying only Eq. (6). In both maps, a darker area means a smaller distance and indicates greater similarity between the two fish images. A brighter area means greater dissimilarity between the two images. In this distance map is stored as a 2D array, and each element is D_{ij}, where i, j are video frame numbers.

Because there are several special fish motion characteristics, we can improve Eq. (5) to reduce the computing time. These characteristics include the head orientation and swimming mode. We do not calculate the distance D_{ij} if the fish motions are different. This approach is shown in Eq. (6). Before utilizing Eq. (6-a), we compute Eq. (6-b) and (6-c) first. If the return value is not equal to 1, this pair is immediately discarded. Because the computing time for D_{ij} is greater than that for computing Eq. (6-b) and (6-c), building a distance map using Eq. (6-a) can save more than 90% of the required time compared to using only Eq. (5). This result is shown in Fig. 5.

$$\begin{aligned} D'_{ij} &= State(i, j) \bullet D_{ij} \\ &= (OH(i, j) \otimes ST(i, j)) \bullet MA(i, j) \bullet D_{ij} \end{aligned} \tag{6}$$

where $\bullet$ *is a scalar* multiplication

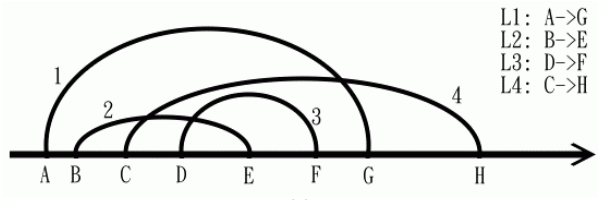
$$OH(i, j) \otimes ST(i, j) = \begin{cases} MAX & if\ OH(i) \neq OH(j)\ \ or\ ST(i) \neq ST(j) \\ 1 & if\ OH(i) = OH(j)\ \ and\ \ ST(i) = ST(j) \end{cases} \tag{6-b}$$

$$MA(i, j) = \begin{cases} MAX & if\ \left| MA(i) - MA(j) \right| > Threshold \\ 1 & otherwise \end{cases} \tag{6-c}$$

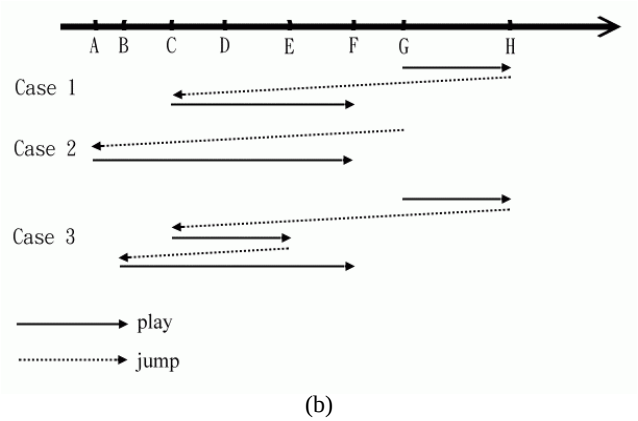
In the above equations, OH (i) stands for the "Head Orientation", ST (i) stands for the "Swimming Mode", and MA (i) represents the "Motion Area" of the fish in frame i.

4.2 Video Loop Position Table

After feasible video loops are selected, synthetic fish animation can be created by reusing or compounding the video loops. Fig. 6 shows three possible ways to get from G to F. In this figure, the fish is moving in a limited 2D plane instead of an infinite area. In Fig. 6(a), there are four video loops, L1, L2, L3, and L4. If we wish to generate synthetic animation and play a transition from G to F, there are many ways to do. We show three possible ways to play video frames in Fig. 6(b). The policy for case 1 is GHCDEF. For case 2, it is GABCDEF, and for case 3, it is GHCDEBCDEF. The black line means "play the fish images successively." The dotted line means "jump to other fish images."



(a)

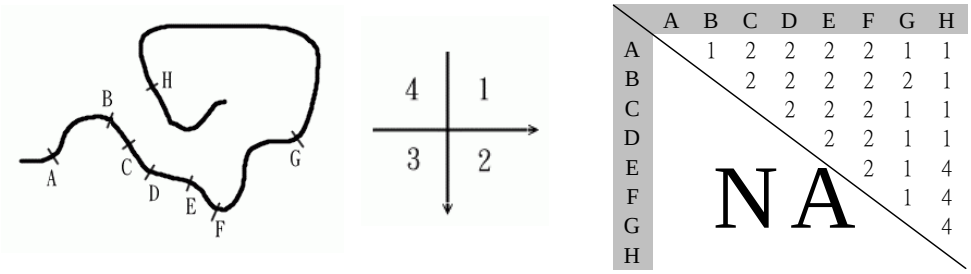


(b)

Fig. 6. The relationship between video loops in the time domain.

In Fig. 7(a), the trajectory of a fish is shown at the beginning and the end of each video loop. This figure shows the movement trajectory for a fish using several possible video loops from Fig. 6(a). Obviously, for each case in Fig. 6(b), the fish will swim to a very different place in a 2D plane. If the animation is generated using only the relations in the video loops in the time domain, we will never know where the fish will swim. In other words, finding an additional relationship in the fish orientation/position is necessary for fish animation. In order to manage video loops better, a position table was designed that records the relative quadrant position between the later and the earlier fish

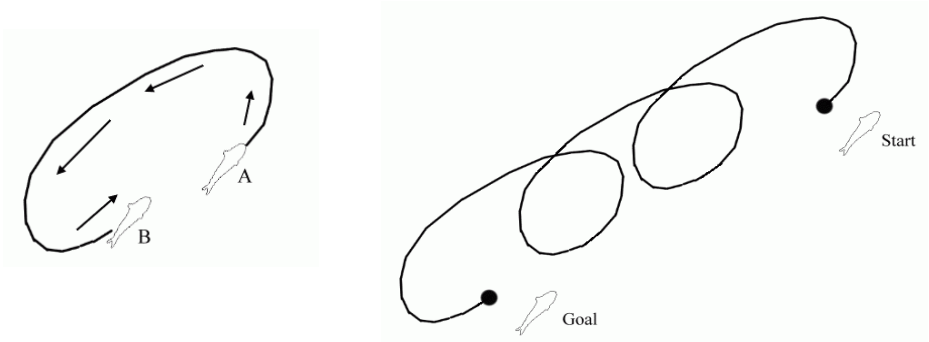
images. The quadrant is shown in Fig. 7(b), and an example of a position table is shown in Fig. 7(c), where the label for each cell represents the relative position quadrant between the later fish position and the former position. For example, the value of (A, B) is 1; that is, B is in the first quadrant of A. Therefore, we assume that the current fish image is at F, the end of Loop 3 (L3), and that the fish will swim toward the first quadrant. From this table, we can continue playing the fish images from F to G because the fish position in G is in the first quadrant of F.



(a) (b) (c)

Fig. 7. Video loop position table.

Fig. 8. In the time domain, the video loops are divided into two groups because the fish does not swim normally, or because the fish motion does not appear to be uniform.



(a) (b)

Fig. 9. Improper video loops.

The video loop position table helps us to determine the proper video loops. However, there are some problems with the video loop technique. In Fig. 7, although the current fish should swim toward the third quadrant, a video loop bringing the fish toward

this goal is not found. There are no video loops in the third quadrant that lead to other loops. Furthermore, in Fig. 8, for example, there are six video loops in the video clip. The first three loops in this video clip overlap one another, and the last three loops also overlap. However, these video loops are separated into two groups. Once we play the video loops in the later group, we cannot play the video loops in the former group. Moreover, if a video loop leads the fish to some other orientation that is entirely opposite to the current orientation (see Fig. 9), then using this kind of loop to generate animation will produce odd-looking results. Therefore, we must have some constraint.

4.3 Switching to Neural Network Play Mode

The problems mentioned in section 4.2 can not be solved if we insist on generating animation using only video loops. For this reason, in our experiment, we left the video loop mode when these problems occurred and we switched to the neural-network play mode instead, which is defined in section 3. To smoothly switch between these two play modes, we had to find gateways between them. For the neural network mode, some of the original video frames were selected and synthesized into new fish images. We built a synthetic fish database which consisted of these two types of images. These two types of images were further classified according to the fish swimming mode and orientations in the synthetic fish database. Computing the gateways between the fish database and the original video source was a very straightforward process. All the gateways were determined according to the image similarity between them. If a fish image *m* in the fish database was very similar to a fish image *n* in the original video source, then there was a gateway between images *m* and *n*. When we played fish image *m*, we could jump to image *n* and then play a video loop that contained *n*. The visual quality of the fish animation was maintained because the video loops were from the original source instead of from image synthesis. When we played an image *n* that was in a video loop and we wanted to interactively move the fish to another place, no video loop image was available for this purpose. In this situation, we could jump to image *m* in the fish database and start to animate the fish in the neural network mode. Later, if we had a chance, we would jump back to the video loop mode again. In this manner, we could compensate the disadvantage of the video loop method.

5. EXPERIMENT RESULTS

In this section, we will present several diagrams and discuss some important results for each major process. More experiments and demos can be found at our web site: *http://couger.csie.ncku.edu.tw/~vr/fish.html*.

5.1 Computing Time

As described previously, there were 4480 frames in the source video clip. In Table 1 we list the computation times for the major steps shown in Fig. 1. All of the processes listed in Table 1can be accomplished automatically without any user intervention.

Table 1. The computation times for the major steps

Process		Computing Time
1	Extract Fish Images from a Video Clip	Several Hours
2	Build a Motion Model by Learning Through a Neural Network	About 70 Seconds (200 epochs)
3	Video Loops Analysis	About 30 minutes
4	Synthesize Fish Images for All Orientations	About 15 minutes
5	Find the Gateways of Synthetic Images and Video Loops	Several Hours
6	Generate Animation (Hybrid Playing Mode)	Real-Time

5.1.1 Learning through a neural network

After the fish motion data were extracted from the video, the motion model was trained using a BPN algorithm. The network architecture was a 3-layer network with 4 inputs, 20 hidden units and 2 outputs. The momentum value was set to 0.5, and the learning rate was set to 0.5. The error rate for each epoch is shown in Fig. 10. This figure shows that the proposed fish motion model could be converge very quickly. The X-axis is the number of epochs. The Y-axis is the amount of error in the motion area prediction. In this experiment (where both the learning rate and momentum were 0.5), after about 20 epochs, convergence is approaching. We will not address the criterion for terminating the training process in this work. Information about neural network training can be found in [4]. For experiments on the other parameters, please see our web site.

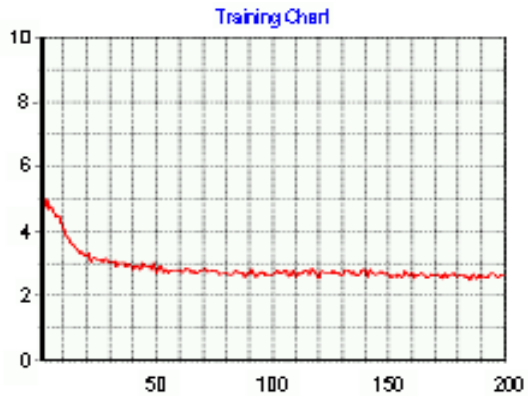


Fig. 10. The error rate for each epoch in the training process.

5.2 Swimming Through a Specific Goal

We can use not only the neural network play mode (see Fig. 11) or the video loop mode, but also a hybrid of these two modes (see Fig. 12). In Fig. 11, we show that the fish could chase specific goals (A, B and C) when we applied the trained motion model. Fig. 12 shows the results for the hybrid play mode. When the fish was at A in the video loop play mode, the program switched to another play mode because of the lack of a

proper video loop. This brought the fish into the legal area. At B, the orientation of the fish head was entirely different from that in the video loop. There were not enough video loops to produce this piece of animation. For this reason, we had to abort the video loop play mode and switch to the neural network mode. For more experimental results and animated sequences, please visit our website.

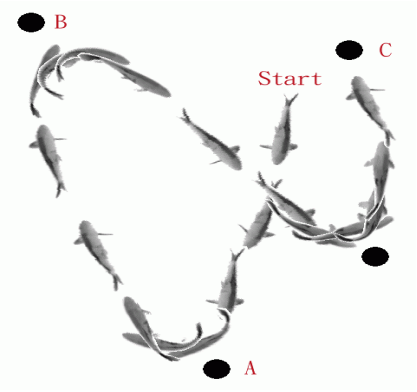


Fig. 11. Fish chases specific goals.

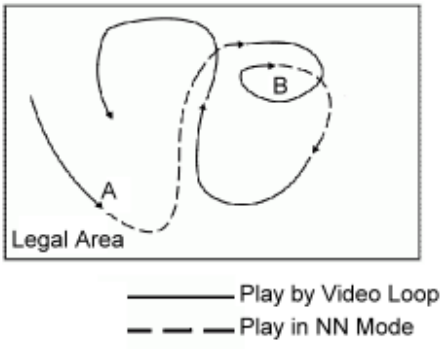


Fig. 12. Hybrid play mode.

6. CONCLUSIONS

In this paper, a novel system for implementing realistic fish animation has been presented. We have proposed two realistic fish play modes. The first is the neural network play mode. The second is the video loop play mode. More significantly, these two rendering methods have been combined to eliminate the otherwise unavoidable constraints encountered when playing fish animation. The neural network motion model play mode is trained by using fish motion parameters that are extracted from the video. That is, this motion model is trained by using actual fish behavior in the input video. The advantage is that this mode allows animators to interactively control fish animation as in a computer game. We have proposed a video loop position table that greatly helps the animator determine the proper video loop when playing in the video loop mode. The advantage of video loops is that they guarantee visual smoothness in the fish animation. We also combine play mode, in which the modes compensate for one another. In the near future, we will turn our attention to finding a more accurate method for extracting motion parameters. We will also apply our methodologies to other targets, such as human expressions and other objects that exhibit regular idiosyncrasies.

REFERENCES

1. A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa, "Video textures," in *Proceedings of SIGGRAPH 2000, ACM Annual Conference Series*, 2000, pp. 489-498.
2. A. Schödl and I. Essa, "Machine learning for video-based rendering," in T. K. Leen, T. G. Dietterich, and Volker Tresp, ed., *Advances in Neural Information Processing*

Systems, Vol. 13, MIT Press, USA, 2001, pp. 1002-1008.
3. P. Debevec, *Image-Based Modeling, Rendering, and Lighting*, SIGGRAPH'99 Course 39, August 1999.
4. S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed., Prentice Hall, 1999.
5. X. Tu and D. Terzopoulos, "Artificial fishes: physics, locomotion, perception, behavior," in *Proceedings of SIGGRAPH 94*, *ACM Annual Conference Series*, 1994, pp. 43-50.
6. R. Grzeszczuk and D. Terzopoulos, "Automated learning of muscle-actuated locomotion through control abstraction," in *Proceedings of SIGGRAPH '95*, *ACM Annual Conference Series*, 1995, pp. 63-70.
7. R. Grzeszczuk , D. Terzopoulos, and G. Hinton, "NeuroAnimator: fast neural network emulation and control of physics-based models," in *Proceedings of SIGGRAPH 95*, *ACM Annual Conference Series*, 1998, pp. 6 pp. 9-20.
8. Z. Bar-Joseph and D. Lischinski, *Statistical Learning of Multi-Dimensional Textures*, Master's Thesis, Institute of Computer Science, The Hebrew University of Jurusalem, Israel, 1999.
9. X. Wen, T. D. Huffmire, H. H. Hu, and A. Finkelstein, "Wavelet-based video indexing and querying," *Multimedia Systems*, Vol. 7, 1999, pp. 350-358.
10. A. Finkelstein, C. E. Jacobs, and D. H. Salesin, "Multiresolution video," in *Proceedings of SIGGRAPH 96*, *ACM Annual Conference Series*, 1996, pp. 281-290.
11. H. Takahashi, J. Hatoya, N. Hashimoto, and M. Nakajima, "Animation synthesis for virtual fish from video," in *Proceedings of the 10th ICAT* (*International Conference on Artificial reality and Telexistence*), 2000, pp. 90-97.
12. A. J. Lipton, "Virtual postman-real-time, interactive virtual video," Technical Report, CMU-RI-TR-99-12, the Robotics Institute, Carnegie Mellon University, 1999.
13. J. R. Parker, *Algorithms for Image Processing and Computer Vision*, Wiley Computer Publishing, 1997.

Tong-Yee Lee (李同益) was born in Tainan county, Taiwan, Republic of China, in 1966. He received his B.S in computer engineering from Tatung Institute of Technology in Taipei, Taiwan, in 1988, his M. S. in computer engineering from National Taiwan University in 1990, and his Ph.D in computer engineering from Washington State University, Pullman, in May 1995. Now, he is a Professor in the department of Computer Science and Information Engineering at National Cheng-Kung University in Tainan, Taiwan, Republic of China. He serves as a guest associate editor for IEEE Transactions on Information Technology in Biomedicine in 2000, 2001, 2002 and 2003, respectively. His current research interests include computer graphics, visualization, virtual reality, 3D Game Design, surgical simulation, distributed & collaborative virtual environment.

Hung-Yi Chen (陳弘毅) was born in Chang-Hua county. He received his BS degree in Computer Science from I-Shou University, and M.S. degree in Department of Computer Science and Information Engineering within National Cheng-Kung University, in 1999 and 2001 respectively. His research is concerned with wavelets and multi-scale analysis and its application in computer graphics. Furthermore, he is also interested in the video-based animation, computer image, and computer vision.