Synthetic Motion Capture for Interactive Virtual Worlds

Qinxin Yu and Demetri Terzopoulos
Department of Computer Science, University of Toronto
10 King's College Road, Toronto, ON M5S 3G4, Canada
E-mail: {qyu|dt}@cs.toronto.edu

Abstract

The numerical simulation of biomechanical models enables the behavioral animation of realistic artificial animals in virtual worlds. Unfortunately, even on high-end graphics workstations, the biomechanical simulation approach is at present computationally too demanding for the animation of numerous animals at interactive frame rates. We tackle this problem by replacing biomechanical animal models with fast kinematic replicas that reproduce the locomotion abilities of the original models with reasonable fidelity. Our technique is based on capturing motion data by systematically simulating the biomechanical models. We refer to it as synthetic motion capture, because of the similarity to natural motion capture applied to real animals. We compile the captured motion data into kinematic action repertoires that are sufficiently rich to support elaborate behavioral animation. Synthetic motion capture in conjunction with level-of-detail geometric modeling and object culling during rendering has enabled us to transform a system designed for the realistic, off-line biomechanical/behavioral animation of artificial fishes, into an interactive, stereoscopic, virtual undersea experience.

1. Introduction

Artificial life modeling is an exciting new trend in computer graphics [13]. It has yielded impressive animations such as Tu and Terzopoulos' artificial fishes [15]. Seen in pre-recorded action, these lifelike virtual animals beckon active involvement. One feels compelled ultimately to interact with artificial fishes in their virtual marine environment as scuba divers would interact with the marine life inhabiting a coral reef. On the level of motion synthesis and control, however, the realistic, artificial life modeling of animals typically relies on biomechanical modeling techniques, which unfortunately require intensive numerical computation. In addition to those employed in artificial fishes, physics-based locomotion models also form the basis of Miller's snakes and worms [11], the virtual humans of Hodgins *et al.* [8], and other realistically self-animating characters.

This paper proposes an approach that brings us closer to developing engaging virtual environments populated by lifelike characters. Our goal is to develop fast derivatives of biomechanics based animation capable of supporting the interactive animation and rendering of virtual worlds inhabited by numerous lifelike creatures. We would like to achieve this goal without necessarily relying on costly, specialized virtual reality equipment, such as flight simulators [19, 12] or CAVE-like installations [3]. Unfortunately, in addition to the burden of photorealistic rendering, the dynamic simulation of biomechanical animal models of any complexity is usually too computationally intensive to run at interactive rates on current desktop or deskside graphics workstations.

Our solution is to replace computationally expensive biomechanical animal models with fast kinematic replicas that preserve as much as possible the lifelike appearances, locomotions, and behaviors of the fully dynamic originals. In particular, we capture motion data through the systematic biomechanical simulation of locomotion in the original models. We refer to this technique as *synthetic motion capture* since it is in principle not unlike natural motion capture applied to real animals, particularly human actors. We appropriately process the recorded data and compile the captured actions into *action repertoires*. The action repertoire implements motion synthesis in a kinematic creature, and it is rich enough to support natural looking locomotion and complex behavior.

To demonstrate our approach, we have transformed the non-realtime world of artificial fishes presented in [15] into an interactive virtual undersea experience. The user pilots a submarine, navigating in a 3D virtual world populated by lifelike marine fauna (see Figs. 5, 6, and 7). The user may maneuver the submarine into a large school of fishes, chase a fleeing fish, or simply look around and observe colorful marine life unfold. Our interactive virtual marine world is inhabited by 60 artificial fishes of 7 different species. Each fish is an autonomous behavioral agent that interacts with other fishes. Our virtual marine world runs at interactive rates on a deskside graphics workstation and also in a large-scale "Reality Theater".

Section 2 reviews related work. Section 3 explains how we apply synthetic motion capture to artificial fishes. Tu's biomechanical fish models provide motion data for the action repertoires of our kinematic fishes. Section 4 presents techniques for processing the motion data to produce a functional motor control system. Section 5 discusses how we have adjusted Tu's artificial fish behavioral model to deal with the new, fully kinematic motor system. Section 6 reports on how we accelerate rendering by culling objects relative to the view frustum and geometrically modeling visible objects with a suitable level of detail based on their distance from the viewpoint. Section 7 discusses the performance that our approach achieves, and Section 8 presents conclusions.

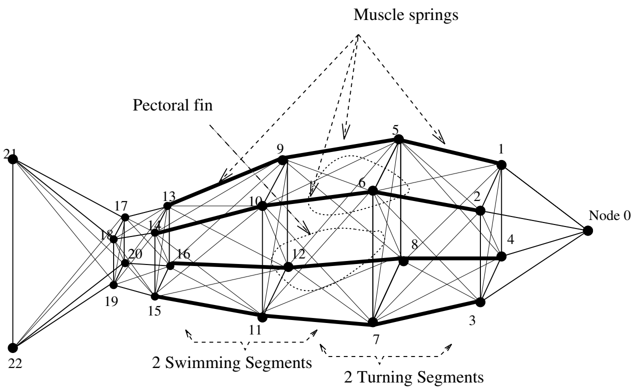

Figure 1. The artificial fish biomechanical model (reproduced from [14]).

2. Related work

Researchers and VR system developers have employed various techniques to increase the complexity of models in virtual environments while maintaining realism and fast frame rates.

Carlson and Hodgins [2] used simulation levels of detail (LOD) for the real-time animation of single-legged hoppers, switching between a full dynamic model of a hopper, a less expensive hybrid dynamic/kinematic model, and a simple point-mass model. We too exploit multiple levels of detail in animation, but the intrinsically higher biomechanical complexity of artificial fishes makes it infeasible to maintain an acceptable frame rate using any reasonable dynamic model. Instead, we propose synthetic motion capture as a means of achieving interactive frame rates without excessively compromising the quality of the animation.

Granieri et al. [5] used an off-line process to record posture graphs for a human model. The recorded posture graphs were played back to animate human figures in a distributed simulation. They also used motion levels of detail, but concentrated more on procedurally generated motion. Van de Panne [16] used footprints as a basis for generating locomotion for bipedal figures at interactive rates.

In creating action repertoires for virtual creatures, we were motivated by motion capture techniques [1, 6, 10]. Wiley and Hahn [17] applied an interpolation synthesis process to motion captured data to generate new motions for articulated figures. Lamouret and van de Panne [9] discussed various problems associated with the use of motion databases to create novel animations. They implemented a prototype system for a planar three-link articulated hopping figure. We have addressed some of the problems outlined in their paper and have successfully built a much more elaborate system.

3. Compiling action repertoires

In this section, we explain the application of synthetic motion capture to artificial fishes. The artificial fish model is described in [15]. Additional details are available in [14].

The original biomechanical fish model is a dynamic mass-spring-damper system consisting of 23 nodal point masses as illustrated in Fig. 1. Under the action of the 12 contractile muscle springs, at each time frame t the numerical simulator performs an implicit time integration of the system of 69 coupled, second-order ordinary differential equations of motion that govern the biomechanical model. This involves first computing the external hydrodynamic forces at time t, then solving a sparse 69×69 system of linear algebraic equations for the 23 nodal velocities at $t + \Delta t$, and finally integrating explicitly in time to obtain the 23 nodal positions $\mathbf{n}_i$, for $i = 0, \ldots, 22$, at the next time frame $t + \Delta t$.

3.1. Motion data capture and processing

To eliminate this computationally intensive numerical simulation, we capture and compile into action repertoires the nodal positions computed over sequences of time frames.

The numerical simulator computes nodal positions $\mathbf{n}_i$ with respect to a fixed world coordinate system. To compile an action repertoire and facilitate multiple level-of-detail modeling, we express these nodal positions with respect to a body-centered coordinate system $\mathbf{B}$, illustrated in Fig. 2, that translates and rotates in accordance with the dynamic fish model. At each time frame, we record the incremental translation (i.e., the change in position) and rotation (i.e., the change in orientation) of $\mathbf{B}$, as well as the "body deformation", or the nodal positions with respect to this body coordinate system.

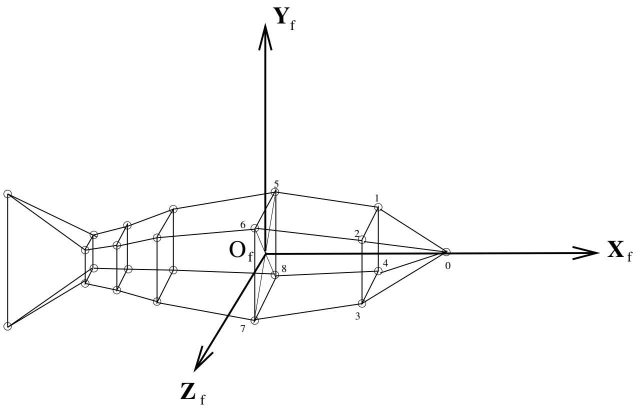

Figure 2. The body coordinate system of a fish.

Referring to Fig. 2, the origin $\mathbf{o} = [o_1\ o_1\ o_3]^T$ (center point of the fish) and the three unit vectors that define the body coordinate system $\mathbf{B}$ are computed as follows:

$$\begin{aligned} \mathbf{o} &= \frac{1}{2}(\mathbf{n}_5 + \mathbf{n}_7) &(1)\\ \mathbf{x} &= \frac{\mathbf{n}_0 - \mathbf{o}}{||\mathbf{n}_0 - \mathbf{o}||} &(2)\\ \mathbf{y} &= \mathbf{x} \times \frac{\mathbf{n}_5 - \mathbf{n}_6}{||\mathbf{n}_5 - \mathbf{n}_6||} &(3)\\ \mathbf{z} &= \mathbf{x} \times \mathbf{y}. &(4) \end{aligned}$$

The $\mathbf{x} = [x_1\ x_2\ x_3]^T$ axis points to the anterior of the fish, the $\mathbf{y} = [y_1\ y_2\ y_3]^T$ axis points in the dorsal direction, and the axis $\mathbf{z} = [z_1\ z_2\ z_3]^T$ points in the right lateral direction. This body coordinate system can be represented by the homogeneous matrix

$$\mathbf{B} = \begin{bmatrix} x_1 & y_1 & z_1 & o_1 \\ x_2 & y_2 & z_2 & o_2 \\ x_3 & y_3 & z_3 & o_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (5)$$

within which the upper-left 3×3 submatrix $\mathbf{R} = \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} \end{bmatrix}$ indicates the rotation required to transform a point in the body coordinate system to a point in world coordinate system.

At each time frame t, we record the change in orientation and position. The orientation change is recorded in the form of a 3×3 rotation matrix $\mathbf{M}^t$ that transforms $\mathbf{R}^t$ into $\mathbf{R}^{t+\Delta t}$:

$$\mathbf{M}^t = \mathbf{R}^{t+\Delta t}(\mathbf{R}^t)^{-1}, \qquad (6)$$

where $(\mathbf{R}^t)^{-1} = (\mathbf{R}^t)^T$, since it is an orthonormal matrix. This rotation matrix $\mathbf{M}^t$ captures the three orientation degrees of freedom. The change of position is recorded as the translation of the center point with respect to the orientation of the body coordinate system:

$$\mathbf{t}^t = (\mathbf{R}^t)^{-1}(\mathbf{o}^{t+\Delta t} - \mathbf{o}^t). \qquad (7)$$

Let $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{22}$ denote the deformation data, where $\mathbf{d}_i$ is a vector indicating the position for the ith node in the fish model. The deformation data are recorded with respect to the body coordinate system as follows:

$$\mathbf{d}_i^t = (\mathbf{R}^{t+\Delta t})^{-1}(\mathbf{n}_i^{t+\Delta t} - \mathbf{o}^{t+\Delta t}), \quad i = 0, 1, \ldots, 22. \quad (8)$$

Figures 3 and 4 show examples of recorded locomotion segments, a forward swim segment and a right turn segment, respectively. For each frame, the trajectory of the local $\mathbf{x}$ and $\mathbf{z}$ axis shows (in top view) the evolving sequence of position and orientation changes up to the current frame, while the fish body shows the deformation relative to the body coordinate system in the current frame.

The artificial fish geometric display model uses NURBS surfaces. To display the original artificial fish, the NURBS control points are computed relatively inexpensively from the nodal positions of the biomechanical model. We have the option of explicitly storing control points as part of the synthetic motion capture process. Since there are many more control points (426) than there are nodes (23), there is a tradeoff between the memory required to store control points explicitly versus the time required to compute them on the fly from the nodal points. As the storage requirements grow, memory paging is exacerbated, tending to slow down the animation. Thus, the recording of control points for a particular species of fish can be justified only when its population in the animation is substantial. In our marine world, only the schooling fish, whose population totals 51, satisfies this condition. We record the control points $\mathbf{S}$ in body system coordinates for the schooling fish as

$$\mathbf{s}_i^t = (\mathbf{R}^{t+\Delta t})^{-1}(\mathbf{c}_i^{t+\Delta t} - \mathbf{o}^{t+\Delta t}), \quad i = 0, 1, \ldots, 425, \quad (9)$$

where the $\mathbf{c}_i$ denote the control point positions in global world coordinates.

3.2. Pattern abstraction

The dynamic simulation of the physics-based artificial fishes uses 9 motor controllers to generate coordinated muscle actions [15, 14]. The 9 basic swimming patterns—forward swimming, left turning, right turning, gliding, ascending, descending, balancing, braking, and retreating—can be combined to synthesize continuous locomotion. To reduce the size of the action repertoire, the following considerations lead us to abstract from these a smaller set of fundamental motion types.

Gliding serves as a transition action between forward swimming and turning. For example, it is used to switch

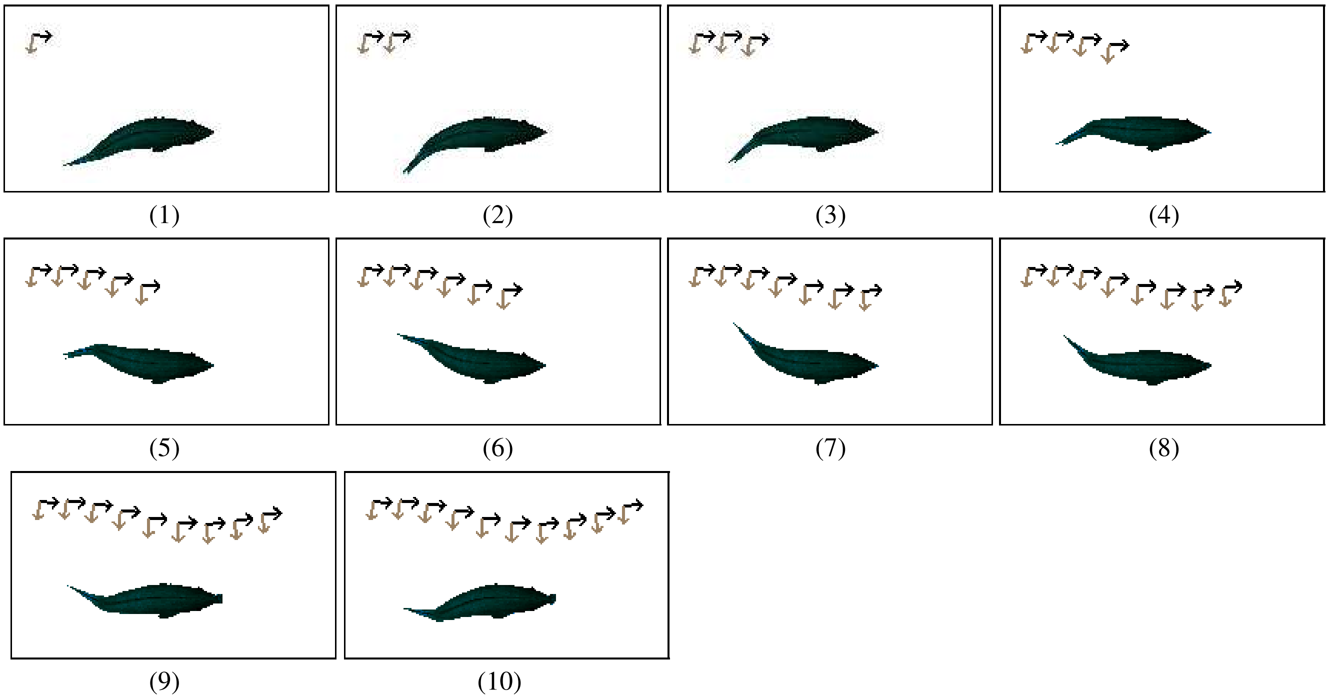

Figure 3. A forward swim action segment. In each frame, the trajectory of the x-axis (darker arrow) and z-axis (lighter arrow) of the body-centered coordinate system shows the sequence of position and orientation changes. The fish body shows the body deformation.

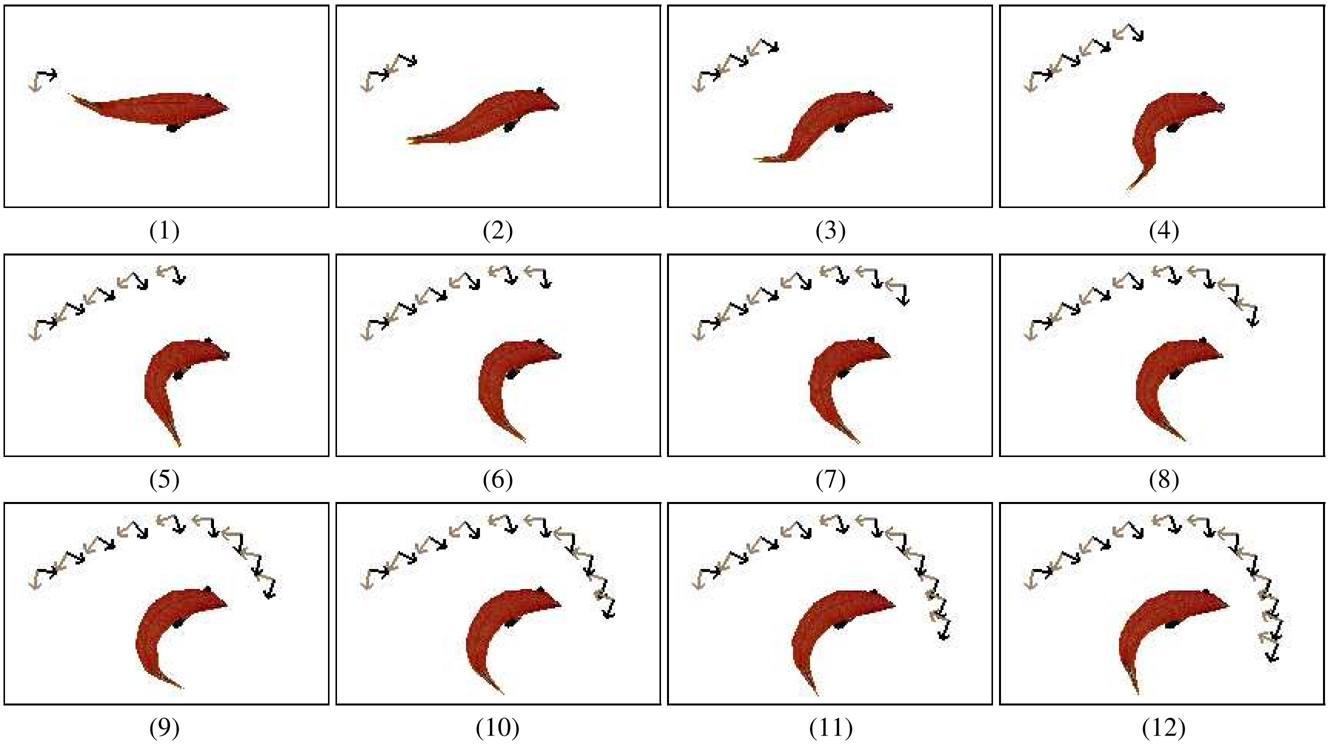

Figure 4. A right turn action segment. Refer to the caption of Fig. 3.

smoothly from swimming forward to turning left, or from turning left to turning right, etc. Significant storage space is required for this action to be stored in the repertoire, since it assumes different forms in different transitions. Fortunately, motion warping [18] serves as a good replacement for gliding, and it requires almost no storage. Ascending and descending can be easily represented by a forward swim heading upwards and downwards respectively. The balancing action helps a fish to maintain its balance, so it does not go belly-up, and it must be done with care when we generate animations. Section 4.2 will describe gliding and balancing in more detail. Braking slows the forward velocity and it can be approximated by a forward swim with a negative acceleration. Similarly, retreating can be accomplished by a forward swim with a negative velocity. The pectoral fin movements that play a vital role in animating a life-like swimming fish continue to be computed using a kinematic model as in the original system.

Hence, the action repertoires consist of three fundamental motion types—forward swimming, left turning, and right turning. Furthermore, fishes from different species exhibit different muscle actions for any given swim pattern because of differences in body shapes and mass distributions. As a result, we must compile a different action repertoire for each species. Fish in the same species may vary in size, but they display similar muscle actions for the same swim pattern. In this case, we scale the stored data to accommodate the variability in size. Consequently, the action repertoires contain data for three swim patterns for each species.

3.3. Action segments

Our goal is to select the minimal set of action segments that can best represent each swim pattern. Since extended swimming motion is cyclic in nature, we record one cycle for each selected segment. For a forward swim, the locomotion speed is approximately proportional to the contraction amplitudes and frequencies of the muscles, and a fish can swim no faster than a certain top speed. Hence, we select three segments with three speeds—slow, medium and fast—which serve adequately to approximate the full range of possible speeds. Fig. 3 shows a sample forward swim segment.

The turn angle of a fish is also proportional to the contraction amplitudes and frequencies of the muscles. We categorize turns into sharp and gradual turns. Hence, a single segment of each category is recorded for each species. Figure 4 shows a sample right turn segment.

4. The motor system

Each fish navigates around the virtual world autonomously and its motor system is responsible for locomotion. In this section, we describe how we use the action repertoire to synthesize fast, kinematic locomotion.

4.1. Action reconstruction

When we generate a new animation, the recorded data must be adapted to the current position and orientation of a fish. At any time step, we need to determine the nodal positions based on the data stored in the repertoire. First, we apply the orientation and position changes to establish where the fish should be by updating the body coordinate system as follows:

$$\mathbf{R}^t = \mathbf{M}^{t-\Delta t}\mathbf{R}^{t-\Delta t}, \tag{10}$$

and

$$\mathbf{o}^t = \mathbf{o}^{t-\Delta t} + \mathbf{R}^{t-\Delta t}\mathbf{t}^{t-\Delta t}. \tag{11}$$

The nodal positions can then be computed according to the deformation data:

$$\mathbf{n}_i^t = \mathbf{o}^t + \mathbf{R}^t(\alpha \mathbf{d}_i^{t-\Delta t}), \quad i = 0, 1, \ldots, 22, \tag{12}$$

where α is a scaling factor for applying the motion data to fish of the same species, but of varying sizes, and it is computed as the ratio of the size of the animated fish to the size of the recorded fish. For the schooling fish, the control points can be restored similarly as:

$$\mathbf{c}_i^t = \mathbf{o}^t + \mathbf{R}^t(\alpha \mathbf{s}_i^{t-\Delta t}), \quad i = 0, 1, \ldots, 425. \tag{13}$$

4.2. Gliding and balancing

We replace the gliding action with motion warping [18] where the nodal points serve as correspondence points. It turns out that a linear interpolation is sufficient to provide a smooth transition between swimming patterns. Since the fish's tail usually undergoes the largest deformation, the distance that node 22 (refer to Fig. 1) travels between the last frame of a data segment to the first frame of the subsequent segment is used to determine how many linearly interpolated intermediate frames are necessary to produce a seamless transition, by setting a maximum distance that a correspondence point can move in a simulation time step.

The continuous application of rotation to the body coordinate system employed by motion synthesis may introduce artifacts that occasionally cause the fish to swim on its side. A compensatory rolling motion is needed to maintain the fish's balance. We determine the necessary roll angle, defined as the angle of rotation about the fish's body-centered x-axis such that the projection of the y-axis in the vertically up direction is maximal, and slowly roll the fish to an upright posture.

4.3. Level-of-detail animation

The animation may be simplified significantly when the fish is not in view. In this case, it is unnecessary to render the graphical display model, hence we suppress the reconstruction of the nodal positions (12) and the NURBS surface control points (13). Motion warping is also disabled. We only update the body-centered coordinate system of an invisible fish using equations (10) and (11).

5. Behavior system

The behavior system of the artificial fish is responsible for higher level behavior, such as dynamic goal setting, obstacle avoidance, foraging, schooling, mating, etc. (see [15]). At each time step, an intention generator examines sensory information acquired by the perception system and selects appropriate action. Because we replace the original biomechanical locomotion controllers of the artificial fish with our action repertoire, we must introduce a secondary controller to mediate between the intention generator and our new motor system.

5.1. Secondary controller

The intention generator makes a decision about the swim pattern and sets the appropriate motor controller parameters for the dynamic model. We rely on these parameter values to select among the action segments in the action repertoire to produce the desired swim pattern. For a forward swim, the parameters determine the swimming speed—slow, medium, or fast—and a suitable swim segment is selected. Similarly for a turn, the parameters determine the turn angles—gradual or sharp—and a suitable turn segment is selected.

The limited number of action segments in the repertoire reduces the locomotion abilities of the fishes. As a result, the probability of multiple fishes swimming in synchrony increases. To address this problem, the secondary controller monitors how long a fish has been pursuing any particular segment. When the duration exceeds a threshold, the controller will post a recommendation to the intention generator to switch randomly to a different swim pattern. The intention generator decides the feasibility of the recommendation. This approach succeeds in synthesizing the diversity of motion essential to a natural looking marine world.

5.2. Behavior planner adaptation

Replacing the dynamic model by a kinematic action repertoire reduces the precision in the maneuverability of the fishes, and they may experience problems accomplishing certain goals. For example, the obstacle avoidance mechanism may fail more frequently. We adjust the relevant behavior planners to compensate.

For obstacle avoidance, we enlarge the sensitivity region for detecting collision threats, giving the fishes enough time and space to maneuver around each other despite their somewhat weakened motor abilities. Another affected behavior is the pursuit of targets. The original approach had the fish attending increasingly carefully to the location of the target as it approaches. For instance, the fish swims merely in the general direction of a distant target. As it approaches the target, it tries harder to steer to its exact location. To offset the weakened motor system, we increase the fish's alertness. The motion planner begins at a further distance its careful steering towards the exact location of the target. This strategy has proved successful, as evidenced by the fish's ability to navigate towards and ingest food.

6. Graphical display model

Although synthetic motion capture saves significant computation time, the complexity of the highly textured NURBS-based graphical display hampers the synthesis of real-time animation. We applied view frustum culling and level-of-detail (LOD) techniques [4, 7] to reduce the rendering time.

Because only a limited number of objects are typically visible at any time in a virtual world such as ours, culling the display of fishes outside the view frustum helps to maintain a relatively fast frame rate, not only by avoiding the graphical display, but also by avoiding nodal position reconstruction and the processing of the NURBS surface control points, as was described in Section 4.3. To reduce the cost, a single point visibility check is used for the fishes. Each fish is approximated by its center point. This requires a slight enlargement of the view frustum so that a fish with its center point just off-screen will still be correctly considered as visible. This may cause a few fishes that are marginally outside the view volume occasionally to be regarded as visible, but the reduction in computation time for visibility checking more than offsets. Culling is also applied to the seaweeds.

Among visible fishes, those that are close to the viewpoint are rendered using NURBS surfaces, while those sufficiently distant are displayed as control point meshes at significantly lower cost. In Figs. 5 and 6, for instance, most fishes in the school are far away and they are displayed using control point meshes, while the rest are displayed as NURBS surfaces.

7. Results and discussion

The sustainable update rate of the original (biomechanics-based) artificial fishes animation system running

Figure 5. We stop the submarine to watch a group of tropical fishes swimming across the viewport, with a school passing by in the distance. Some fishes are feeding on white plankton floating among the aquatic plants.

Simulation Method	CPU Time (ms)
Dynamics	40
Synthetic Motion Capture	0.01

Table 1. The processing time for each fish with dynamic simulation and with synthetic motion capture on a Silicon Graphics R10000 InfiniteReality workstation.

on a Silicon Graphics 1 $\times$ 194 MHz R10000 InfiniteReality workstation is about 0.25 frames per second, making it impossible for the user to perceive continuous motion, let alone navigate the virtual world. Using the techniques presented in this paper, we achieve an interactive frame rate on the same workstation. With the improved speed, users can explore the virtual marine world as if they are piloting a submarine. The user can navigate the vessel using the mouse. Fig. 5, 6 and 7 are still images captured on one of our virtual submarine dives. We also implemented the option to see the world stereoscopically, providing a compelling depth perception and a quasi-immersive 3D experience. Fig. 8 shows a user enjoying the ride, wearing a pair of CrystalEyes stereo glasses.

With the help of culling and level-of-detail in the graphical display, our current implementation has a fluctuating frame rate that depends on the number of visible fishes and the percentage of fishes that are in close view. We observe frame rates in the range of 10 to 50 frames per second. This is fast enough to provide the user a sense of action and interactiveness.

Figure 6. Let's approach the school to take a closer look.

Figure 7. While on our way up towards the surface, we see a variety of fishes beneath us.

The speed up over the original biomechanical animation is attained by accelerating the motor system and the graphical display model. Table 1 compares the computation times required for the dynamic model and our motion capture model, for a single fish. The indicated times are for the case when the fish is fully visible, which includes the reconstruction of the body coordinate system and positions of all the nodal points. The computation time is reduced by a factor of 4000. Table 2 shows the rendering time for a complex scene (about 40 fishes can be seen) with and without culling and multi-resolution. Our current technique cuts the rendering time by a factor of about 27.5.

We have furthermore developed a large scale version of our virtual undersea world in a "Relocatable Reality Theater" marketed by Trimension, Inc. (http://www.trimension-

Figure 8. A user is enjoying the ride stereoscopically, wearing a pair of CrystalEyes stereo glasses.

Figure 9. The virtual undersea world experienced on the panoramic display in a Trimension Reality Theater.

Graphics	CPU Time (ms)
Without Culling/LOD	2340
With Culling/LOD	85

Table 2. The rendering time for a similarly complex scene without and with the use of culling and level-of-detail rendering.

inc.com), which combines a Silicon Graphics 8 $\times$ R10000 CPU Onyx2 system and multichannel PRODAS projection technology from SEOS Displays, Ltd. The system features three InfiniteReality graphics pipelines, each feeding video to an overhead projector. This system animates and renders our virtual world at a sustainable rate of at least 30 frames per second. It renders through the three projectors a seamless image of approximately 4000 $\times$ 1000 pixel resolution across a 18 $\times$ 8-foot curved screen, producing a large panoramic display that fills the peripheral view. Fig. 9 shows the theater.

8. Conclusion

We have introduced the idea of replacing biomechanical models of animals with ultra-fast kinematic replicas that capture with reasonable fidelity the locomotion abilities of the original models. In applying synthetic motion capture, we collect segments of motion data generated through the systematic numerical simulation of the biomechanical model, select a minimal set of action segments that parsimoniously represents the various locomotion patterns, and compile these segments into an action repertoire for the artificial animal. The motor system retrieves action segments from the action repertoire to synthesize continuous kinematic locomotion, using motion warping to smooth transitions between different locomotion patterns. The artificial animal's behavior system combines locomotion patterns into meaningful higher-level behavior.

To demonstrate the power of our approach, we have developed an interactive system that provides users a virtual undersea experience. The user pilots a virtual submarine to explore a marine environment populated by lifelike fauna. The virtual marine world may be easily extended so that the artificial fishes will interact with users, further enhancing interactivity and enjoyment. Excluding rendering, our synthetic motion capture approach is three orders of magnitude faster than the original biomechanical simulation of the artificial fishes. By eliminating the significant burden of numerical simulation, the frame rate of our virtual world becomes bounded by graphics rendering performance. We accelerated rendering by culling objects relative to the view frustum and displaying visible objects with a suitable geometric level of detail based on their distance from the viewpoint.

Acknowledgments

We would like to thank Xiaoyuan Tu for making this research possible by providing her artificial fishes software upon which we have developed our system. We are grateful to Michiel van de Panne, Victor Ng-Thow-Hing, Joe Laszlo, Radek Grzeszczuk, and Neil Enns for their invaluable suggestions and assistance. Special thanks to Mark Deacon and the SMART Toronto organization for providing access to the Trimension/SGI Reality Theater as a development and test facility during Reality 98. Gary Schissler from SGI Canada was among the helpful people who provided technical support in the Reality Theater. QY acknowledges the financial support of an NSERC Postgraduate Scholarship. The research reported herein was funded by the Natural Sciences and Engineering Research Council of Canada through a E.W.R. Steacie grant.

References

[1] A. Bruderlin and L. Williams. Motion signal processing. In *SIGGRAPH 95 Conference Proceedings*, pages 97–104, Aug. 1995.

[2] D. A. Carlson and J. K. Hodgins. Simulation levels of detail for real-time animation. In *Proceedings of the Graphics Interface*, pages 1–8, May21–23 1997.

[3] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, Aug. 1993.

[4] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 247–254, Aug. 1993.

[5] J. P. Granieri, J. Crabtree, and N. I. Badler. Production and playback of human figure motion for visual simulation. *ACM Transactions on Modeling and Computer Simulation*, 5(3):222–241, July 1995.

[6] B. Guenter, C. F. Rose, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using space-time constraints. In *SIGGRAPH 96 Conference Proceedings*, pages 147–154, Aug. 1996.

[7] P. S. Heckbert and M. Garland. Multiresolution modeling for fast rendering. In *Proc. Graphics Interface '94*, pages 43–50, May 1994.

[8] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *SIGGRAPH 95 Conference Proceedings*, pages 71–78, Aug. 1995.

[9] A. Lamouret and M. van de Panne. Motion synthesis by example. In *Proceedings of the 7th Eurographics Workshop on Simulation and Animation*, pages 199–212, Poiters, France, 1996.

[10] R. Maiocchi. 3-D character animation using motion capture. In N. Magnetat-Thalmann and D. Thalmann, editors, *Interactive Computer Animation*, pages 10–39. Prentice-Hall, London, 1996.

[11] G. S. P. Miller. The motion dynamics of snakes and worms. In *SIGGRAPH 95 Conference Proceedings*, pages 169–178, Aug. 1988.

[12] R. Pausch and T. Crea. A literature survey for virtual environments: Military flight simulator visual systems and simulator sickness. Technical Report CS-92-25, Department of Computer Science, University of Virginia, Aug. 19 1992.

[13] D. Terzopoulos, editor. *Artificial Life for Graphics, Animation, Multimedia, and Virtual Reality*, volume 23 of *ACM SIGGRAPH'97 Course Notes*. 1997.

[14] X. Tu. *Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior*. PhD thesis, University of Toronto, 1996.

[15] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH '94*, pages 43–50, July 1994.

[16] M. van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–223, 1997.

[17] D. J. Wiley and J. K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6):39–45, Nov./Dec. 1997.

[18] A. Witkin and Z. Popović. Motion warping. In *SIGGRAPH 95 Conference Proceedings*, pages 105–108, Aug. 1995.

[19] J. K. Yan. Advances in computer-generated imagery for flight simulation. *IEEE Computer Graphics and Applications*, 5(8):37–51, Aug. 1985.