The Visual Computer

Synthetic motion capture: Implementing an interactive virtual marine world

Qinxin Yu,
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

Biomechanical simulation enables the realistic animation of animals in virtual worlds, but at significant computational cost. Synthetic motion capture is a low-cost technique that prescribes (i) the capture of motion data through the systematic simulation of biomechanical animal models and (ii) compilation of the captured data into kinematic action repertoires rich enough 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 offline biomechanical/behavioral animation of artificial fishes into a real-time, interactive, stereoscopic, virtual undersea experience.

Key words: Motion capture and processing – Virtual reality – Artificial life – Level of detail – Biomechanics

The Visual Computer (1999) 15:377–394

1 Introduction

Artificial life modeling and animation is an exciting new trend in computer graphics (Terzopoulos 1997). It has yielded impressive prerecorded action animations, as epitomized by the artificial fishes animations of Tu and Terzopoulos (1994). Lifelike virtual animals naturally beckon active involvement, and one feels compelled to interact with artificial fishes in their virtual marine environment as scuba divers would interact with the marine life inhabiting a coral reef. For realistic motion synthesis and control, however, the artificial life modeling of animals typically relies on biomechanical simulation, which unfortunately requires 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 (Miller 1988), the virtual humans of Hodgins et al. (1995), and other realistically self-animating characters.

This paper proposes an approach that brings us closer to developing engaging virtual environments or interactive games populated by lifelike characters. Our goal is to develop fast derivatives of biomechanics-based animation models capable of supporting interactive 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 (Yan 1985; Pausch and Crea 1992) or CAVE-like installations (Cruz-Neira et al. 1993). Unfortunately, the dynamic simulation and photorealistic rendering of numerous biomechanical animal models of any complexity is usually computationally too 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 simulation of locomotion in the original biomechanical 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-real-time world of artificial fishes presented in (Tu and Terzopoulos 1994) into an interactive virtual undersea experience. The user pilots a virtual submarine, navigating in a 3D virtual world populated by lifelike marine fauna (see Figs. 9–11). 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. 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. Accelerated by synthetic motion capture, level-of-detail geometric modeling, and efficient rendering, 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 reviews Tu's artificial fishes which inspired our work. Section 4 explains the application of synthetic motion capture to artificial fishes and the compilation of the collected motion data into action repertoires for our kinematic fishes. Section 5 presents techniques for using action repertoires in functional motor control systems. Section 6 discusses our modification of the artificial fish behavioral model so that it can cope with the new, kinematic motor system. Section 7 reports on how we accelerate rendering by culling objects situated outside the view frustum, fast B-spline rendering, and geometrically modeling visible objects with a suitable level of detail based on their distance from the viewpoint. Section 8 presents sample results and discusses the performance of our virtual reality demonstrations. Section 9 presents conclusions and possible directions for future research.

2 Related work

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

Carlson and Hodgins (1997) 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. Nougaret et al. (1997) developed a coarse-to-fine modeling method to develop controllers for the dynamic locomotion of fishes. We too exploit multiple levels of detail in animation, but we propose synthetic motion capture as a means of animating numerous fishes at interactive frame rates while retaining as much as possible the realistic movements of Tu's artificial fish biomechanical model (see Sect. 3.1).

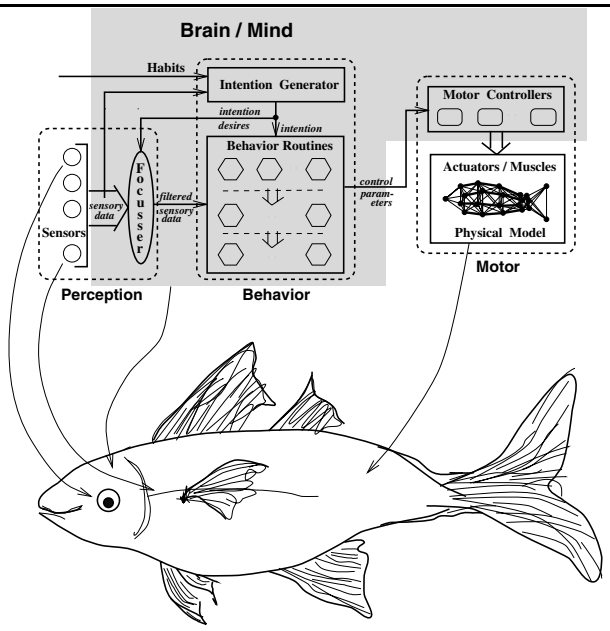
Granieri et al. (1995) used an offline process to record posture graphs for a human model. The recorded posture graphs were played back to animate human figures in a distributed simulation. They also used motion levels of detail, but concentrated more on procedurally generated motion. Van de Panne (1997) 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 (Bruderlin and Williams 1995; Rose et al. 1996; Maiocchi 1996). Wiley and Hahn (1997) applied an interpolation synthesis process to captured motion data to generate new motions for articulated figures. Lamouret and van de Panne (1996) 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 Artificial fishes

This paper develops a real-time version of the artificial fishes simulation created by Tu and Terzopoulos (1994). In this section, we will review aspects of the artificial fish model, which is described in detail in (Tu 1996), that are relevant to our work. Figure 1 shows a system overview of the artificial fish. The artificial fish model comprises three submodels: a graphical display model, a biomechanical model, and a brain model.

The graphical display model is a conventional texture mapped geometric model that captures the form and appearance of any specific real fish from its image. Two B-spline surfaces are juxtaposed, one for the left half and the other for the right half of the fish body. B-spline surfaces are also used to represent the dorsal and ventral fins, while the pectoral fins are modeled as polygonal surfaces. The body surfaces are mapped with textures extracted from the original fish image. The control points of the geometric surface model are coupled to an underlying biome-



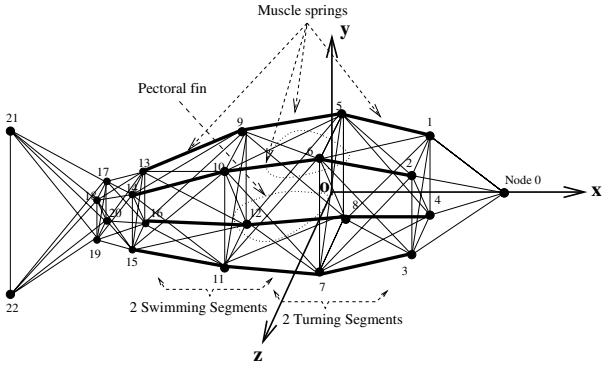


Fig. 1. System overview of the artificial fish (adapted from (Tu 1996))

Fig. 2. The artificial fish biomechanical model (adapted from (Tu 1996))

chanical model which deforms the surfaces and sets them in motion.

3.1 Biomechanical model

The biomechanical model represents the primary physical structures of the fish's body and is responsible for locomotion. Each fish is approximated by a mass-spring-damper model consisting of 23 nodal point masses and 91 uniaxial viscoelastic units as illustrated in Fig. 2. Each unit is a (Voigt) spring-damper pair which approximates the viscoelasticity of biological tissue. Twelve viscoelastic units, shown in bold in the figure, represent contractile muscles whose natural lengths can decrease as a function of time. The artificial fish achieves hydrodynamic locomotion through coordinated muscle contractions induced by controllers in the motor center of the fish's brain (see below). The muscle contractions deform the fish's body in the virtual water, thus producing reaction forces that propel the animal forward. The pectoral fins induce additional reaction forces that control the pitching, yawing, and rolling of the fish's body.

To simulate the dynamics of the fish model, a system of 69 second-order ordinary differential equations of motion, which governs the biomechanical model, is integrated in time using a numerically stable, semi-implicit Euler method. At each time step t, the integrator applies the 12 muscle contractions, computes the external hydrodynamic forces at time t, then solves a sparse 69×69 system of linear algebraic equations for the 23 nodal velocities at time $t + \Delta t$, for a suitable time step Δt, and finally integrates explicitly in time to obtain the 23 nodal positions $\boldsymbol{n}_i$, for $i = 0, \ldots, 22$, at $t + \Delta t$.

3.2 Brain model

As illustrated in Fig. 1, the brain model of the artificial fish is responsible for motor control, perception control and behavior control. It consists of three control centers: the motor center, the perception center, and the behavior center.

The motor center includes nine motor controllers (MCs) which are responsible for synthesizing coordinated muscle contractions and pectoral fin motions to produce locomotion:

1. swim MC – produces forward caudal locomotion
2. left-turn MC – executes a left turn
3. right-turn MC – executes a right turn

4. glide MC – provides a smooth transition from forward swimming to turning and vice versa
5. ascend MC – ascends towards the surface
6. descend MC – dives towards the seabed
7. balance MC – maintains the balance of the body
8. brake MC – slows the forward velocity
9. backward MC – retreats

The MCs offer to the behavior model a set of natural locomotion control parameters such as swim speeds and turn angles.

The perception system of a fish comprises a set of virtual sensors and a perceptual focus of attention mechanism. A vision sensor enables the artificial fish to detect objects in the environment that are relevant to its survival, such as food, predators and mates. The vision sensor is limited to a 300° spherical angle extending to a radius consistent with the visibility of the translucent water. This defines a view volume or field of view within which objects in the world model can be seen by the fish.

The behavior system, which controls action selection, comprises the mental state of the fish, an intention generator, and a set of behavior routines. At each time step, the intention generator issues an intention based on the fish's mental state and incoming sensory information, and one of nine behavior routines is selected and executed: avoiding-static-obstacle, avoiding-fish, chasing-target, eating-food, mating, leaving, wandering, escaping, and schooling. Each behavior routine uses the perceptual data to select appropriate motor controllers and provide them with the proper parameters.

3.3 Virtual marine world

Three types of artificial fishes were implemented in the original system: predators, prey, and pacifists. Physics-based animate models of seaweeds and plankton were created to enhance visual realism. The virtual water is translucent and it is rendered with a bluish fog effect. Water currents are simulated as simple fluid flow fields, and they appropriately affect the fishes, seaweeds, and plankton.

4 Compiling action repertoires

In this section, we explain the application of our synthetic motion capture technique to artificial fishes. Our goal is to replace the motor system – i.e., the biomechanical model and motor center – of the original artificial fish model with a kinematic action repertoire compiled using synthetic motion capture.

4.1 Motion data capture and processing

To eliminate the computationally intensive numerical simulation that was described in Sect. 3.1, we capture and compile into action repertoires the nodal positions computed over sequences of time frames.

The numerical simulator computes nodal positions $\boldsymbol{n}_i$ 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 $\boldsymbol{B}$, 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 $\boldsymbol{B}$, as well as the "body deformation", or the nodal positions with respect to this body coordinate system.

Referring to Fig. 2, the origin $\boldsymbol{o} = [o_1\ o_2\ o_3]^{\mathrm{T}}$ (center point of the fish) and the three unit vectors that define the body coordinate system $\boldsymbol{B}$ are computed as follows:

$$\boldsymbol{o} = \frac{1}{2}(\boldsymbol{n}_5 + \boldsymbol{n}_7) \tag{1}$$

$$\boldsymbol{x} = \frac{\boldsymbol{n}_0 - \boldsymbol{o}}{\|\boldsymbol{n}_0 - \boldsymbol{o}\|} \tag{2}$$

$$\boldsymbol{y} = \boldsymbol{x} \times \frac{\boldsymbol{n}_5 - \boldsymbol{n}_6}{\|\boldsymbol{n}_5 - \boldsymbol{n}_6\|} \tag{3}$$

$$\boldsymbol{z} = \boldsymbol{x} \times \boldsymbol{y}\,. \tag{4}$$

The $\boldsymbol{x} = [x_1\ x_2\ x_3]^{\mathrm{T}}$ axis points to the anterior of the fish, the $\boldsymbol{y} = [y_1\ y_2\ y_3]^{\mathrm{T}}$ axis points in the dorsal direction, and the axis $\boldsymbol{z} = [z_1\ z_2\ z_3]^{\mathrm{T}}$ points in the right lateral direction.[1] This body coordinate system can be represented by the homogeneous matrix

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

within which the upper-left 3×3 submatrix $\boldsymbol{R} = \begin{bmatrix} \boldsymbol{x}\ \boldsymbol{y}\ \boldsymbol{z} \end{bmatrix}$ indicates the rotation required to transform

[1] The superscript T denotes matrix transposition

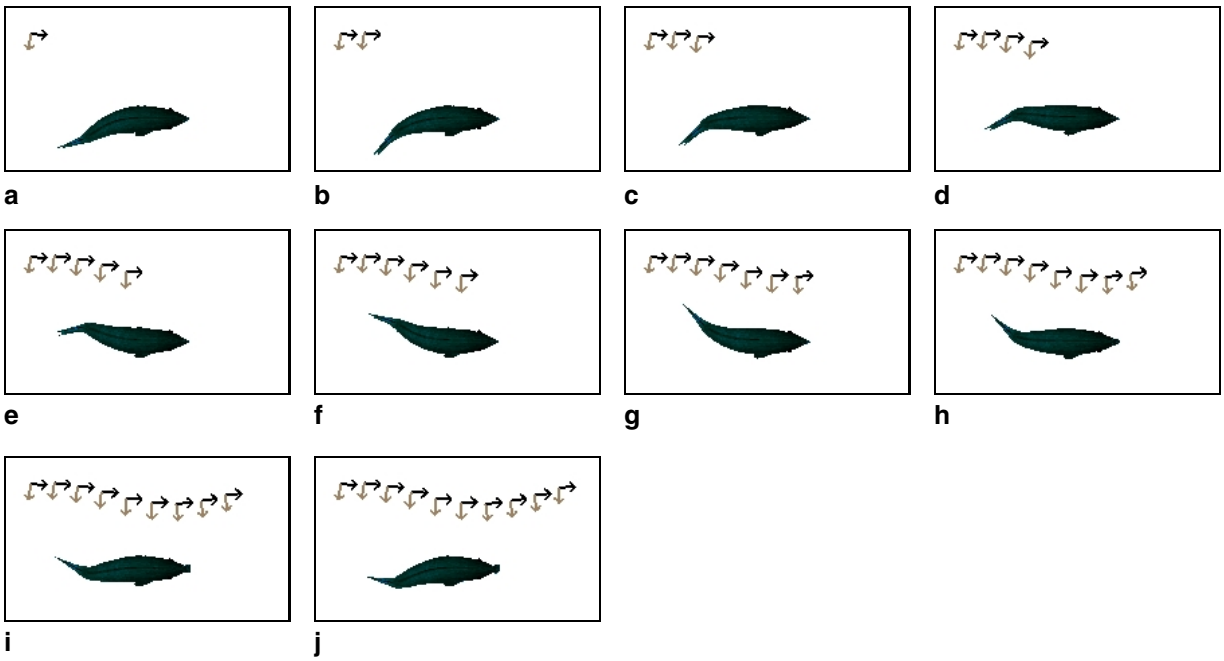


Fig. 3a–j. A forward swim action segment. In each frame, the trajectory of the x-axis (*darker arrow*) and z-axis (*lighter arrow*) of the body-centered coordinate system shows the sequence of position and orientation changes. The fish body shows the body deformation

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 $\boldsymbol{M}^t$ that transforms $\boldsymbol{R}^t$ into $\boldsymbol{R}^{t+\Delta t}$:[2]

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

where $(\boldsymbol{R}^t)^{-1} = (\boldsymbol{R}^t)^{\mathrm{T}}$, since it is an orthonormal matrix. This rotation matrix $\boldsymbol{M}^t$ 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:

$$\boldsymbol{t}^t = (\boldsymbol{R}^t)^{-1}(\boldsymbol{o}^{t+\Delta t} - \boldsymbol{o}^t)\,. \tag{7}$$

Let $\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{22}$ denote the deformation data, where $\boldsymbol{d}_i$ 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:

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

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 $\boldsymbol{x}$ and $\boldsymbol{z}$ 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 B-spline surfaces. To display the original artificial fish, the B-spline 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

[2] The superscripts t and $t + \Delta t$ indicate time frames

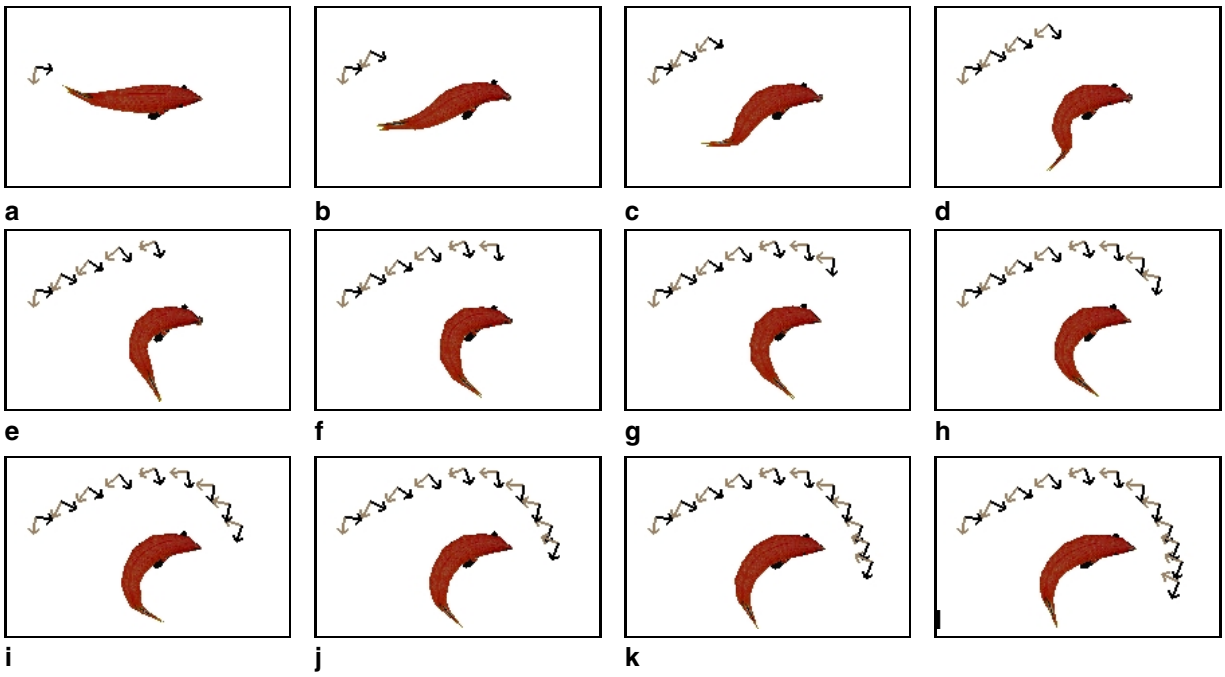


Fig. 4a–l. A right turn action segment. Refer to the caption of Fig. 3

exacerbated, tending to slow down the animation. Thus, the recording of control points for a particular fish species can be justified only when its population in the animation is substantial. In our marine world, we do this only for the schooling fish, whose population numbers 51. We record the control points $\boldsymbol{S}$ in body system coordinates for the schooling fish as

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

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

4.2 Pattern abstraction

Within the physical simulation, each fish acts as an intelligent agent, reacting to its environment and interacting with other fishes. The possible actions generate a rich set of movements from which we must select to form an action repertoire. In particular, the dynamic simulation of locomotion uses 9 motor controllers to generate coordinated muscle actions as described in Sect. 3.2. The 9 basic swimming patterns – forward swimming, left turning, right turning, gliding, ascending, descending, balancing, braking, and retreating – are 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.

In our application, it turns out that effective action repertoires can be compiled from three fundamental motion patterns – 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 can scale the stored data to accommodate the variability in size. Consequently, the action repertoires contain data for three motion patterns for each species.

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

left, or from turning left to turning right, etc. Significant space would be required for this action to be stored in the repertoire, since it assumes different forms in different transitions (for n action segments, there are $n(n-1)$ possible transitions to be recorded). Fortunately, motion warping (Witkin and Popović 1995) 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 5.2 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, crucial in animating a life-like swimming fish, continue to be computed kinematically as in the original system.

4.3 Recording action segments

Our goal is to select the minimal set of action segments that can best represent each motion 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. Figure 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 segment of each category is recorded for each species. Figure 4 shows a sample right turn segment.

To perform the actual recording, we pick a fish from each species, which has a medium size as a representative of that species. We then monitor these representatives and select the segments that satisfy the conditions outlined above. Once this step is completed, the physical simulation is put into action and data are recorded for these selected segments as was explained in Sect. 4.1.

5 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.

5.1 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:

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

and

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

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

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

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:

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

5.2 Gliding and balancing

We replace the gliding action with motion warping (Witkin and Popović 1995) 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. 2) 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

Table 1. Simulation levels of detail

Computation Process	Visible	Invisible
Orientation and Position Changes Recovery (update the Body Coordinate System)	Yes	Yes
Deformation Data Recovery (update nodal positions)	Yes	No
Control Points Computation/Restoration	Yes	No
Gliding	Yes	No
Balancing	Yes	Yes

a seamless transition, by setting a maximum distance that a correspondence point can move in a simulation time step. Appendix A provides the details.
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. A more detailed description is provided in Appendix B.

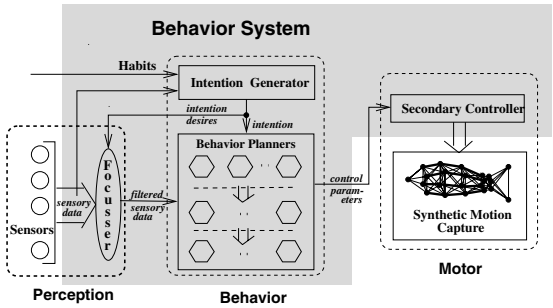


Fig. 5. The behavior system of the artificial fish

5.3 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 B-spline 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). Here, we still perform the balance adjustment to $\boldsymbol{y}$, because disabling this process may cause the fish to roll away from its upright posture and a sudden re-enabling of the balance mechanism as the fish re-enters the view volume would cause an awkward motion which, albeit transient, would be plainly evident, as the fish takes several frames to regain its balance. Table 1 summarizes the two levels of detail that are employed in our simulations.

6 Behavior system

The behavior center (Fig. 5) of the artificial fish's brain is responsible for higher level behavior, such as dynamic goal setting, obstacle avoidance, foraging, schooling, mating, etc. (see (Tu and Terzopoulos 1994)). 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 a synthetic motion capture action repertoire, we must introduce a secondary controller to mediate between the intention generator and our new motor system.

6.1 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.

6.2 Modified behavior system

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, thus 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. Referring to Fig. 6, let $\boldsymbol{g} = (\boldsymbol{p} - \boldsymbol{o})/\|\boldsymbol{p} - \boldsymbol{o}\|$ denote the direction of a fish's target, where $\boldsymbol{p}$ is the position of the target in the world-coordinate system. As the fish gets closer to its goal point, it tries to decrease the angle θ between its body orientation and the direction of the target, provided that the steering angles are not unnaturally drastic. Hence, it performs a fine tuning to align $\boldsymbol{x}$ with $\boldsymbol{g}$ when $\theta < 45°$. This fine tuning starts when the fish is far enough away from the target in order to allow sufficient time for small directional adjustments at each time step to bring the fish to its goal. This strategy has proved successful, as evidenced by the fish's ability to navigate towards and ingest food.

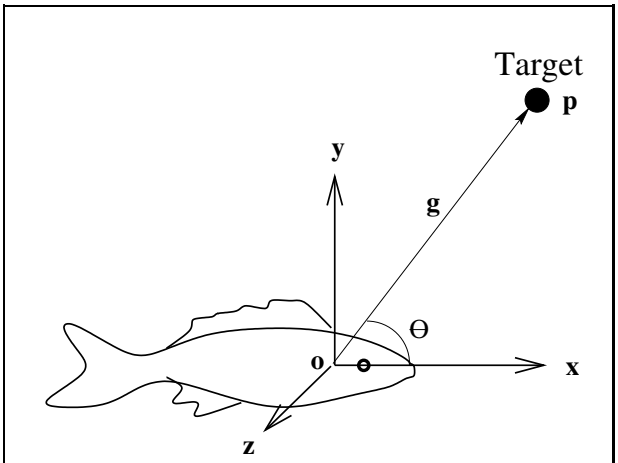


Fig. 6. Target pursuit

The fishes need to have a (territorial) sense of the boundaries of their virtual marine world so that they will stay within it. The world is bounded by the seabed below (a polygonal terrain mesh representing a height field), by the surface of the water above, and is confined to a certain horizontal extent (bounded by a cylinder, for simplicity). The task of keeping the fish inside their territory is similar to collision avoidance; however, the fishes are constrained in which directions they can turn to avoid a "collision". World constraints are enforced by a controller that performs world boundary tests at each time step against the centroid of the fish's body. If the centroid approaches within a specified distance from a boundary, the controller will initiate avoidance action, steering the fish inward such that it will have enough space and time to avoid penetrating the boundary.

7 Efficient rendering

Although synthetic motion capture saves significant computation time, the complexity of the highly textured B-spline based graphical display hampers the synthesis of real-time animation. We applied view frustum culling and level-of-detail (LOD) techniques (Funkhouser and Séquin 1993; Heckbert and Garland 1994) to reduce the rendering time.

7.1 View frustum culling

Because only a limited number of objects are typically visible at any time in a virtual world such as

View	Geometric Representation
Close	Control point mesh for body, B-spline surface for fins
Distant	Control point mesh for body and fins
Invisible	None

Table 2. Multiresolution geometric representations of fishes

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 B-spline surface control points, as was described in Sect. 5.3. To reduce the cost, a single point visibility check is used for the fishes. Each fish is approximated by its centroid. 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.

7.2 Fast B-spline surface rendering

Artificial fish display models are created using texture mapped B-spline surfaces. Each fish body consists of two juxtaposed B-spline surfaces, each of which has $u \times v = 9 \times 21$ control points and is of order 3 along both the u and v parametric axes. For fishes with dorsal and ventral fins, the fins are also B-spline surfaces, each having $u = 2$ and $v = 12$ control points and is of order 1 along the u axis and order 3 along the v axis. The left and right pectoral fins are B-spline surfaces, each having $u = 2$ and $v = 4$ control points and of order 1 along the u axis and order 2 along the v axis.

Instead of relying on the OpenGL (Neider et al. 1993) B-spline surface renderer which adaptively tessellates the surface, we implemented a faster drawing routine for the purposes of our application which takes advantage of the properties of B-spline surfaces (Ng-Thow-Hing and Fiume 1997; Rogers and Adlum 1990). Since we know the knot vectors and the order of the basis functions prior to rendering, we pre-evaluate the basis functions at given tessellation points and store these values in a table. Whenever a control point is changed during rendering, we can quickly update the surface shape without re-evaluating the basis functions at each time step. For multiresolution B-spline rendering where coarser levels are subsets of finer levels, only the basis functions for the finest level need be computed and the coarser levels can be accessed by appropriately decimating the stored values.

7.3 Level-of-detail surface rendering

We classify the fishes into several viewing categories according to how distant they are from the viewpoint. Among visible fishes, those that are close to the viewpoint are rendered using B-spline surfaces, while those sufficiently distant are displayed as control point meshes (surface meshes formed by connecting the $u \times v$ control points of each B-spline surface) at significantly lower cost. The fish body surfaces have considerably more control points than the fins and they consume most of the rendering time. Fortunately, we can replace the body surfaces with control point meshes without serious loss of quality. This effectively reduces the number of polygons sent through the graphics pipeline and eliminates the tessellation time.

Table 2 summarizes the geometric representation levels of detail. Figure 7 compares the two levels of detail against the original B-spline surface display model with tessellated bodies and fins. Through experimentation, we have determined a threshold distance beyond which a visible fish is far enough away that the switch from tessellated B-spline to control point mesh rendering will be subtle, thus guaranteeing a smooth transition between the two levels of detail. In Figs. 9 and 10, for instance, most fishes in the school are far away and they are displayed using control point meshes, while the rest are displayed as B-spline surfaces.

7.4 The marine environment

The virtual water in the marine environment is translucent and it is rendered using a bluish fog

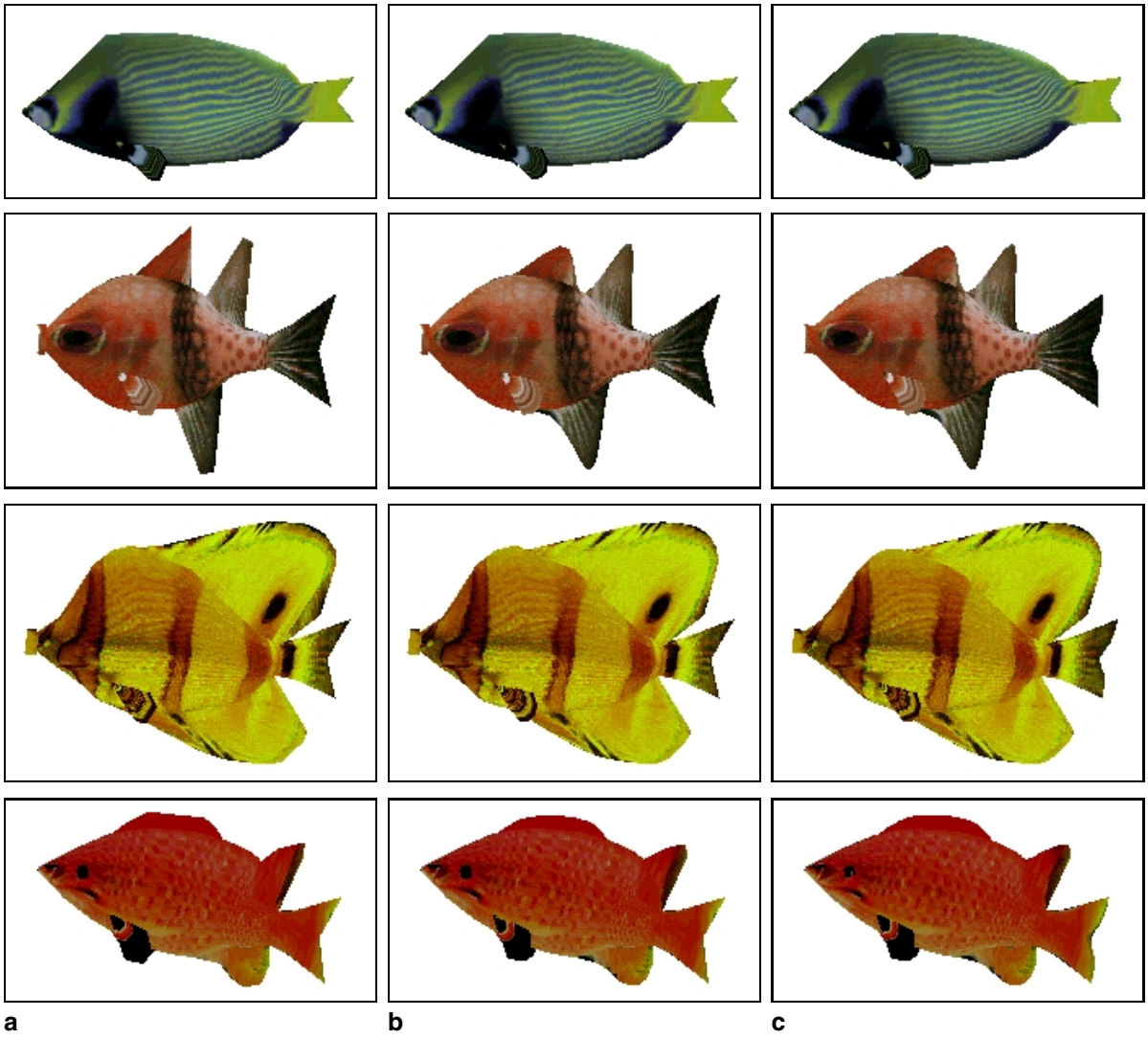


Fig. 7. a Distant view; **b** Close view; **c** Original B-spline surface rendering

effect. In addition to the artificial fishes, the marine environment also includes plankton and seaweeds rooted in the seabed. In our peaceful marine world, the fishes swim among the aquatic plants and feed on the white plankton as they please.

We continue to model the seaweeds using physics-based modeling as in the original system (Tu 1996), since the seaweed simulation is inexpensive. As a fish swims through or passes by a seaweed, the leaves are deflected in response to simulated hydrodynamic forces generated by the fish's body.

8 Results

We will now describe our virtual reality demonstrations, discuss their performance, and show selected results.

8.1 Performance

Running on an SGI workstation with a single 194 MHz MIPS R10000 CPU and an InfiniteReality graphics pipeline, the sustainable update rate of the original (biomechanics-based) artificial fishes animation

Table 3. The processing time for each fish with dynamic simulation and with synthetic motion capture on an SGI R10000 InfiniteReality workstation

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

Table 4. The rendering time for a similarly complex scene without and with the use of culling and level-of-detail rendering

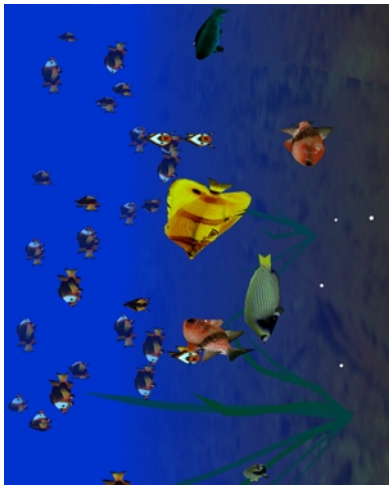
Graphics	CPU Time (ms)
Without Culling/LOD	2340
With Culling/LOD	85

system is approximately 0.25 frames per second, making it impossible for the user to perceive continuous motion, let alone interactively navigate the virtual world. Using synthetic motion capture for artificial fish animation along with culling and level-of-detail techniques, as described in this paper, we achieve an interactive frame rate on the same workstation. Our current implementation achieves a 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 interactivity.

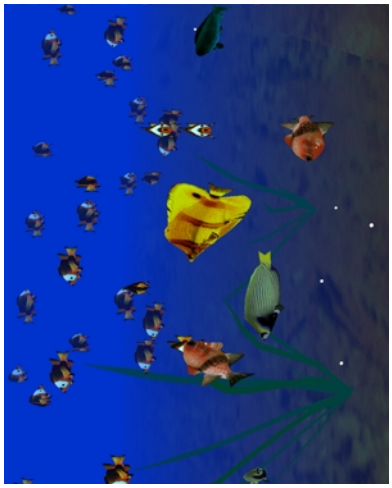
The speed up over the original biomechanical animation is due to the accelerated motor system and the efficient graphical display model. Table 3 compares for a single fish the computation times required for the original biomechanical model and our synthetic motion capture model. The indicated times are 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 4 shows the rendering time for a complex scene (about 40 fishes can be seen) with and without culling and multiresolution rendering. Our current technique cuts the rendering time by a factor of about 27.5.

8.2 Virtual submarine user interface

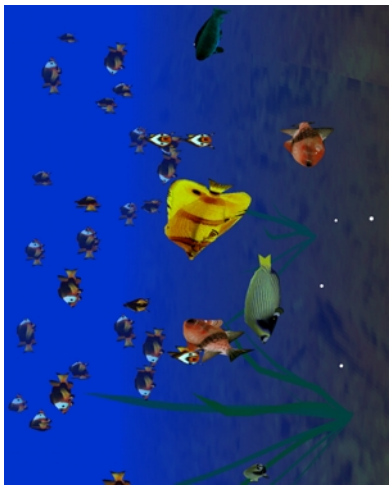
The dramatically improved frame rate enables interaction. Users can pilot a virtual submarine through the marine world. They navigate the submarine using a user interface, which consists of a submarine



(left)



(right)



(left)

Fig. 8. Stereoscopic view

control mechanism and a menu for selecting options. The possible maneuvers of the submarine include accelerating forward, decelerating, reversing, turning left, turning right, pitching up, pitching down, rising up, diving down, and full stop. These steering commands are controlled via the mouse.
We also implemented an option for users to visualize the world stereoscopically using a CrystalEyes viewing system. Figure 8 shows left and right eye views of a sample scene for stereoscopic free fusing. Figures 9–11 are still images captured on one of our virtual submarine dives.

8.3 Large-scale VR demonstration

We have furthermore developed a large scale version of our virtual undersea world in a "Reality Theater" (Fig. 12a) marketed by Trimension, Inc. (1998), which combines an SGI Onyx2 system with eight MIPS R10000 CPUs and a multichannel PRODAS projection system (Fig. 12b) from SEOS Displays, Ltd. The system incorporates three InfiniteReality graphics pipelines, each feeding video to an overhead projector. This system animates and renders our virtual world at a sustainable rate greater than 30 frames per second. It renders through the three projectors a seamless image of approximately 4000×1000 pixel resolution across a 5.5×2.5 m curved screen, producing a large panoramic display that fills the peripheral view (Fig. 13).
The results on the large display are spectacular and the audience enjoys a compelling submarine ride. When the stereoscopic view option is enabled, users can see the fishes moving around them and they enjoy the visual sensation of diving.

9 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.

9.1 Discussion and future work

It would enhance the realism of our virtual reality system if we increased the complexity of the marine habitat. We could create a much more realistic and interesting world by introducing a larger variety of marine plants and including more complex terrain data, or covering the seabed not just with texture-mapped sand but also with 3D rocks, coral reefs, etc.
It would heighten the excitement of our virtual marine world if the fishes responded to the submarine, say, as if it were another large fish. Pacifist fishes would be intrigued and attracted by the submarine, while prey fish would consider it a large predator and either form schools or scatter and flee depending on the proximity of the intruder. Another simple enhancement would be to enable the user to attach him/herself to any particular fish, control its actions, and see the marine world through the fish's eyes.
At present, we have a peaceful marine world where there are no predators. The addition of predator species will bring some interesting scenarios into the otherwise quiet marine environment. A greater variety of pelagic creatures also helps to increase the realism and stimulates the viewer's curiosity. Physical models of other marine animals like Grzeszczuk's

Fig. 9. 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

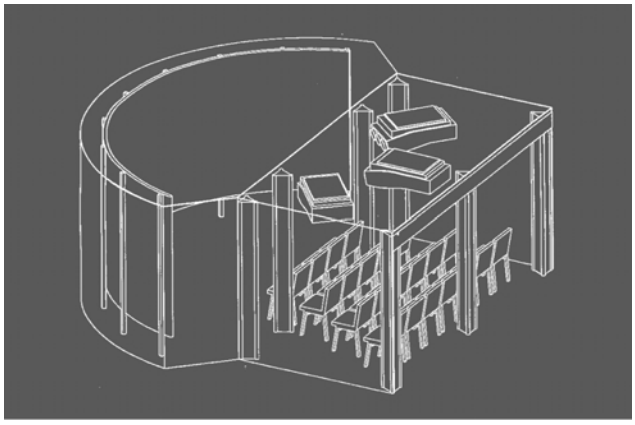
9

Fig. 10. We approach the school to take a closer look

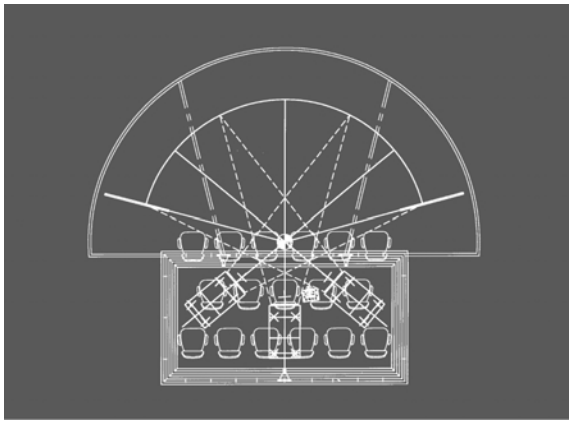
10

Fig. 11. While rising towards the surface, we see a variety of fishes beneath us

11



12a



12b

13

Fig. 12a,b. Reality Theater (Reproduced from (Trimension, Inc. 1998))

Fig. 13. The virtual undersea world experienced on the panoramic display in a Trimension Reality Theater

shark and dolphin (Grzeszczuk and Terzopoulos 1995) already exist and are readily introduced into our virtual marine world.

At present, the action repertoires that we compile comprise small libraries of locomotion data that produce realistic swimming motions. Elaborate behaviors such as mating are affected more seriously than simple behaviors by these minimalist action repertoires which reduce the maneuverability of the kinematic fishes relative to their biomechanical counterparts. We can improve the motor control by expanding the action repertoire with a greater number of recorded segments. However, there is a trade-off between the memory requirement and the quality of motion. We have chosen the current collection of data to achieve a good compromise (our current implementation requires about 40 MB of swap space). As the virtual memory requirement increases, the possibility of paging also increases, which may adversely affect the frame rate.

It may not be necessary to expand the action repertoire, because the ever increasing computing power of graphics systems will in due course make it possible for dynamic simulation to play a role in our multiple level-of-detail simulation models without excessively sacrificing the frame rate. When that day

comes, we can use dynamics for foreground fishes and our synthetic motion capture model for background fishes. This will improve the quality of the animation. A smooth blending between the kinematic model and the dynamic model can be achieved by adding some state variables from the biomechanical model to the action repertoires.

In exploiting physics-based modeling as the source of motion data for our action repertoire, we eliminate the need for expensive motion capture equipment and the noise that is normally present in natural motion data collected using conventional motion capture techniques. Because the dynamical simulation can only approximate natural movements, however, it limits the realism of the captured data. It would be useful to expand the source of our motion data to include data captured from real animals. Unfortunately, attaching sensors and systematically capturing the motions of small, highly deformable animals such as fishes is a challenging task. Video based tracking may be useful for this purpose (Masaki et al. 1998) demonstrate an unusual application which tracks the motion of a mechanical toy fish in a fishtank).

In the current implementation, we targeted our technique towards generating swimming motions for fishes. We believe it can be generalized to other biomechanical animal models. For example, we may be able to use synthetic motion capture to generate swimming motions for humans, or to produce other types of motions such as walking, and jumping for physics-based articulated figures. The physics-based modeling system for the athletes created by Hodgins et al. (1995) may be a good test bed for this. However, when applied to articulated figures, we run into the problem of path planning. The feet of articulated figures should maintain contact with the ground. Therefore, terrain variations pose a problem in adapting captured motion data. Although some research has been done on the subject (Gleicher 1998), it remains an open problem.

An alternative to synthetic motion capture is the NeuroAnimator recently proposed by Grzeszczuk et al. (1998), which uses neural networks to emulate physical dynamics. Both techniques aim to synthesize motions consistent with physics-based models, but their approaches differ dramatically, so it is difficult to draw conclusive comparisons at this time. The NeuroAnimator is typically one or two orders of magnitude faster than conventional numerical simulation, whereas synthetic motion capture yielded three orders of magnitude speedup in the particular application to artificial fishes, albeit with significantly greater online memory requirements.

Acknowledgements. 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, and Radek Grzeszczuk 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" in Toronto. Gary Schissler from SGI Canada was among the helpful people who provided technical support in the Reality Theater. The research reported herein was funded by the Natural Sciences and Engineering Research Council of Canada. QY acknowledges the financial support of an NSERC Postgraduate Scholarship. DT acknowledges the support of a Killam Research Fellowship from the Canada Council for the Arts.

Appendix A. Gliding

The algorithm (Fig. 14) is presented as follows:

Step 1: Determine the position for node 22 for the next time frame (i.e., the first frame of the next swim segment), $\boldsymbol{n}_{22}^{t+\Delta t}$ as shown in equation (12).

Step 2: Let $M_{max} = 2.0$ units in the world-coordinate system be the maximum movement that can be taken by a correspondence point during the linear interpolation, the number of frames required to complete the transition is given by:

$$N_{mw} = \frac{\| \boldsymbol{n}_{22}^{t+\Delta t} - \boldsymbol{n}_{22}^{t} \|}{M_{max}} .$$

If $N_{mw} > 1$, then do

Step 3: Determine the nodal position changes $\boldsymbol{w}$ for each frame during motion warping as:

$$\boldsymbol{w}_i = \frac{\| \boldsymbol{n}_i^{t+\Delta t} - \boldsymbol{n}_i^{t} \|}{N_{mw}}, \quad i = 0, 1, \ldots , 22 .$$

Since the orientation change occurring in any two consecutive frames is small, a linear interpolation of the nodal positions is sufficient to complete the transition. In a more general case when a large change in orientation is possible, an interpolation of the orientation is also required.

Step 4: Stretch the animation by N_{mw} frames, and for each of the N_{mw} intermediate frames, the nodal positions are determined by:

$$\boldsymbol{n}_i^{t+\Delta t} = \boldsymbol{n}_i^{t} + \boldsymbol{w}_i ,$$

and the body-coordinate system is established using equations (1)–(4).
Steps 1 to 3 are executed once when joining two segments belonging to different swim patterns. Step 4 is performed for each motion warping frame. We are essentially inserting N_{mw} gliding frames between the last and the first frame of the two adjoining segments.

Appendix B. Balancing

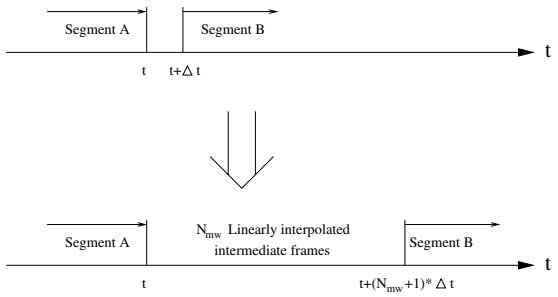
Let $\boldsymbol{y}'$ and $\boldsymbol{z}'$ denote the y and z axes for the body-coordinate system corresponding to the upright posture, the roll angle θ (Fig. 15) is given by:

$$\theta = \cos^{-1}(\boldsymbol{y}' \cdot \boldsymbol{y}) ,$$

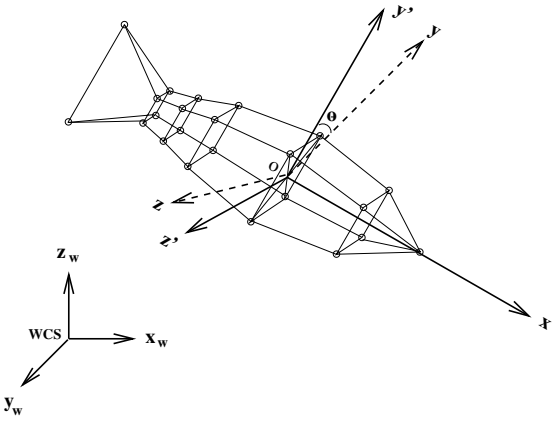
and $\boldsymbol{y}'$ is determined as:

$$\boldsymbol{z}' = \boldsymbol{x} \times \boldsymbol{z}_w ,$$
$$\boldsymbol{y}' = \boldsymbol{z}' \times \boldsymbol{x} ,$$

where $\boldsymbol{z}_w$ is the basis vector for the z-axis in the world-coordinate system, which always point vertically up.
This adjustment is done as part of the orientation update; therefore, the nodal position reconstruction will be done with respect to this modified body-coordinate system.



14



15

Fig. 14. Gliding
Fig. 15. The roll angle

References

1. Bruderlin A, Williams L (1995) Motion signal processing. In: SIGGRAPH 95 Conference Proceedings, 97–104
2. Carlson DA, Hodgins JK (1997) Simulation levels of detail for real-time animation. In: Proceedings of the Graphics Interface, 1–8
3. Cruz-Neira C, Sandin DJ, DeFanti TA (1993) Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In: Computer Graphics (SIGGRAPH '93 Proceedings) 27:135–142
4. Funkhouser TA, Séquin CH (1993) Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In: Computer Graphics (SIGGRAPH '93 Proceedings) 27:247–254
5. Gleicher M (1998) Retargeting motion to new characters. In: SIGGRAPH 98 Conference Proceedings, Annual Conference Series 33–42. ACM SIGGRAPH.
6. Granieri JP, Crabtree J, Badler NI (1995) Production and playback of human figure motion for visual simulation. ACM Transactions on Modeling and Computer Simulation 5(3):222–241
7. Grzeszczuk R, Terzopoulos D (1995) Automated learning of Muscle-Actuated locomotion through control abstraction. In: SIGGRAPH 95 Conference Proceedings, 63–70
8. Grzeszczuk R, Terzopoulos D, Hinton G (1998) NeuroAnimator: Fast neural network emulation and control of physics-based models. In: SIGGRAPH 98 Conference Proceedings, 9–20
9. Heckbert PS, Garland M (1994) Multiresolution modeling for fast rendering. In: Proc. Graphics Interface '94, 43–50
10. Hodgins JK, Wooten WL, Brogan DC, O'Brien JF (1995) Animating human athletics. In: SIGGRAPH 95 Conference Proceedings, 71–78
11. Lamouret A, van de Panne M (1996) Motion synthesis by example. In: Proceedings of the 7th Eurographics Workshop on Simulation and Animation, 199–212
12. Maiocchi R (1996) 3-D character animation using motion capture. Interactive Computer Animation, 10–39
13. Masaki T, Yamaguchi T, Kitamuza Y (1998) An interactive digital fishtank based on live video images. In: AMCP 98 Conference Proceedings, 392–402

14. Miller GSP (1988) The motion dynamics of snakes and worms. In: SIGGRAPH 95 Conference Proceedings, 169–178
15. Neider J, Davis T, Woo M (1993) OpenGL Programming Guide. Reading MA: Addison-Wesley
16. Ng-Thow-Hing V, Fiume E (1997) Interactive display and animation of B-spline solids as muscle shape primitives. In: Computer Animation and Simulation '97, 81–97 Springer Wien
17. Nougaret JL, Arnaldi B, Multon F (1997) Coarse-to-fine design of feedback controllers for dynamic locomotion. The Visual Computer 13(9–10):435–455
18. Pausch R, Crea T (1992) 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, USA
19. Rogers DF, Adlum LA (1990) Dynamic rational B-spline surfaces. Computer-aided Design, 609–616
20. Rose CF, Guenter B, Bodenheimer B, Cohen MF (1996) Efficient generation of motion transitions using spacetime constraints. In: SIGGRAPH 96 Conference Proceedings, 147–154
21. Terzopoulos D, ed. (1997) Artificial Life for Graphics, Animation, Multimedia, and Virtual Reality 22 ACM SIGGRAPH'98 Course Notes. ACM
22. Trimension, Inc. (1998) http://www.trimension-inc.com
23. Tu X, Terzopoulos D (1994) Artificial fishes: Physics, locomotion, perception, behavior. In: Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994) 43–50
24. Tu X (1996) Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior. PhD Dissertation, University of Toronto, Canada
25. van de Panne M (1997) From footprints to animation. Computer Graphics Forum 16(4):211–223
26. Wiley DJ, Hahn JK (1997) Interpolation synthesis of articulated figure motion. IEEE Computer Graphics and Applications 17(6):39–45
27. Witkin A, Popović Z (1995) Motion warping. In: SIGGRAPH 95 Conference Proceedings, 105–108
28. Yan JK (1985) Advances in computer-generated imagery for flight simulation. IEEE Computer Graphics and Applications 5(8):37–51

QINXIN YU received her BSc (Honors) degree in Computer Science from the University of New Brunswick in 1996, and her MSc in Computer Science from the University of Toronto in 1998. She is currently a PhD candidate in the Department of Computer Science at the University of Toronto. Her interests include computer animation, artificial life, and virtual reality.

DEMETRI TERZOPOULOS is Professor of Computer Science and Professor of Electrical & Computer Engineering at the University of Toronto, where he directs the Visual Modeling Group and is a Canada Council Killam Fellow. He received his PhD from MIT. He has been a research scientist at the MIT Artificial Intelligence Lab, a program leader at Schlumberger corporate research centers in California and Texas, and a visiting professor at Schlumberger, Digital, and Intel. His published work includes over 200 technical papers in computer vision and graphics, medical imaging, computer-aided design, and artificial intelligence/life. He has held three of Canada's most distinguished research fellowships. His research contributions have received awards from the IEEE, AAAI, NICOGRAPH, Ars Electronica, International Digital Media Foundation, Canadian Image Processing and Pattern Recognition Society, and the University of Toronto.