University of California
Los Angeles

The Modeling and Animation of Myriapoda

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Jingyi Fang

2015

Abstract of the Dissertation

The Modeling and Animation of Myriapoda

by

Jingyi Fang

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2015

Professor Demetri Terzopoulos, Chair

Unlike two, four, six, and eight legged animals, Myriapoda—i.e., centipedes and millipedes—have been largely overlooked in the computer graphics literature. This thesis presents an artificial life (A-Life) framework for animating the locomotive behavior of Myriapoda with compelling physical and biological realism. Our system produces real-time animation and the creatures are autonomous, requiring minimal animator intervention. Creatures of various body morphologies can be animated simultaneously, and they are capable of complex multi-legged locomotion as well as anguilliform swimming.

Taking an Artificial Life approach, we develop a hybrid animation system that combines kinematic and dynamic simulation to animate a novel biomechanics model specifically tailored to the unique body structure of myriapoda. In our simulated myriapoda, the characteristically vivid leg wave patterns of their biological counterparts result as an emergent behavior of our distributed, decentralized leg control system for terrestrial locomotion. A compelling anguilliform swimming pattern emerges as a result of hydrodynamic simulation and the coordinated actuation of elastically deformable body segments. Locomotive transitions from land to water and vice versa in the creature's simulated physical environment are achieved smoothly by our locomotion controllers. The adoption of robust and efficient elasticity simulation techniques give rise to natural body deformations

and support a novel approach to muscle actuation via rest-shape morphing. The virtual environment can be sensed by the simulated creature's antennae and the sensory information guides its adaptive behaviors, including obstacle avoidance and foraging.

The dissertation of Jingyi Fang is approved.

Stanley J. Osher

Song-Chun Zhu

Joseph M. Teran

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2015

To my beloved family

Table of Contents

List of Figures

Acknowledgments

First and foremost, I would like to express my deep gratitude to my advisor, Professor Demetri Terzopoulos. Needless to say, this thesis would never have existed without his guidance and support. Attracted by his fascinating work in computer graphics and computer vision, I made the decision to switch graduate programs from bioscience to computer science. With his generous assistance, I was able to start a new chapter in my education and professional career, which has been full of excitement and happiness. Demetri is one of the most approachable persons that I have ever met. He has been my mentor both in academics and in life. It has been my good fortune and honor to have met him and become his student.

I sincerely thank Professor Joseph Teran for teaching and advising me on the FEM simulation of elastic materials. It was a precious experience to attend his courses and group meetings. I also would like to thank the other members of my thesis committee, Professors Stanley Osher and Song-Chun Zhu, for their time and interest in reviewing my work and attending my PhD exams.

I am appreciative of my friendly colleagues. I would like to express my thankfulness to Eduardo Poyart, Garett Ridge, Xiaowei Ding, Weiguang Si, Wenjia Huang, Xiaolong Jiang, Craig Yu, Kresimir Petrinec, Masaki Nakada, Konstantinos Sideris, Matthew Wang, Gergely Klar, Sharath Gopal, Alexey Stomakhin, Andre Pradhaha, and Chenfanfu Jiang for sharing their knowledge and ideas with me and for making our lab a joyful and productive place in which to work. I would especially like to thank my friend and undergraduate classmate back in USTC, Chenfanfu. Together, we overcome many challenges during our PhD studies.

I thank the UCLA computer science department for supporting my PHD study by offering me teaching assistant appointments. Not only did these jobs alleviate my financial burdens, but they also sharpened my skills in computer graphics as

well as in teaching. My thanks also go to Lisa Snyder and Scott Friedman at the Institute for Digital Research and Education for contributing to the support of my PhD study via the VSIM project.

I am grateful for the intern mentorship and career development advice that I received from Yi Gong at Nvidia and from Pushkar Joshi and Eric Mueller at Google.

During my PhD research, I was privileged to mentor and work with MS students Andre Abrahamian and Jing Li and with undergraduate student Shin Hong. I would like to acknowledge Andre and Jing's efforts on shallow water simulation and rendering, and Shin's experimentation with Central Pattern Generators.

Finally, I would like to thank my family, especially my wife Xinli, for their persistent and unconditional love.

Vita

2006–2010	B.S., Applied Physics Special Class for the Gifted Young University of Science and Technology of China Hefei, China.
2010–2011	Research Assistant Electronic Imaging Center for Nano-machines University of California, Los Angeles Los Angeles, California
2011–2015	Teaching Assistant Computer Science Department University of California, Los Angeles Los Angeles, California
2011–2014	Research Programmer Institute for Digital Research and Education University of California, Los Angeles Los Angeles, California
2012 Summer	Software Engineering Intern Nvidia, Inc., GPU Infrastructure Team Santa Clara, CA
2013 & 2014 Summer	Software Engineering Intern Google, Inc., Google Web Designer Team Mountain View, CA

Publications

J. Fang, C. Jiang, D. Terzopoulos (2013) Modeling and Animating Myriapoda: A Real-Time Kinematic/Dynamic Approach. *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, Anaheim, USA, July, 2013, pp. 203–212.

Nomenclature

A-Life Artificial Life

AEP Anterior Extreme Pose

CPG Central Pattern Generator

DOF Degree of Freedom

DPIV Digital Particle Image Velocimetry

ECR Energy Consumption Rate

F-PS Frequency and Phase Shift

FEM Finite Element Method

MAC Marker-And-Cell Method

MDP Middle Down Pose

MUP Middle Up Pose

PDE Partial Differential Equations

PEP Posterior Extreme Pose

POV Persistence of Vision

SWE Shallow Water Equations

CHAPTER 1

Introduction

Many movies and games can make use of loathsome creatures such as snakes, spiders, and centipedes to rouse the viewer.[1] Entomophobia therapy can also take advantage of lifelike animations of such creatures.

Motion capture techniques are difficult to apply to multi-legged arthropods such as centipedes due to their small size and high frequency motion pattern (Gibson et al., 2007). It is a daunting task for animators to manually rig a realistic 3D centipede model such that it can locomote over an irregular surface, since at least two requirements must be met: (i) the physical realism of the deformable body and the leg contacts and (ii) the natural appearance of the distinctive leg wave pattern. Advanced simulation techniques are therefore desirable for the realistic synthesis of the locomotion patterns of such creatures.

In biology, the centipede and millipede are but two of the 13,000 identified species that form the Myriapoda subphylum of arthropods. Myriapods range from having over 750 legs to having fewer than ten legs (Minelli, 2011). Fig. 1.1 shows representative of four myriapod classes: Chilopoda, Diplopoda, Symphyla, and Pauropoda. The primary characteristic of myriapods is that they have their tens to hundreds of legs spread over an elongated, segmented body structure, plus a pair of sensory antennae on the frontmost segments (Ruppert et al., 2003). Most of the species have a rigid exoskeleton to protect the body, while the internal

[1] For example, the *Indiana Jones*, *Harry Potter*, *The Mummy*, *Metro 2033*, and *Diablo III* series of motion pictures and games. In the recent Pixar movie *Monsters University*, the Dean Hardscrabble character is featured with centipede legs.

Figure 1.1: Representative of four myriapod classes. Centipedes make up the class Chilopoda (A), they are fast, predatory and venomous, and there are approximately 3,300 species. Millipedes form the class Diplopoda (B), they are slow-moving and detritivorous, and there are approximately 8,000 species. Pauropoda (C) are small (0.5-2.0mm), live in soil, and there are about 700 species. Symphyla (D) are close to centipedes, but transparent, and there are about 200 identified species. Centipedes and millipedes make up nearly 90 percent of the entrire Myriapoda subphylum. *Image Source: Wikipedia.*

soft tissues that connect the segments are deformable and allow the body to flex via muscular contraction and relaxation for turning and twisting. Although myriapoda are terrestrial, some centipedes, such as *Scolopendra subspinipes*, are reported to be capable of swimmming (Minelli, 2011). Video footage can be found online (badboythuglife, 2009; Bifrost1107, 2009) of giant centipedes swimming on the surface of water with legs held against the side of their bodies, exibiting eel-like (anguilliform) swimming patterns. Minelli (2011) suggests that this ability can explain the creature's existence on some remote islands.

During its locomotion, the wave pattern of a myriapod's legs can steadily move

the entire body over highly irregular surfaces. If in the legged animal kingdom we compare four-legged animals to cars, then myriapods are like trains. In the 3D world, legged locomotion is much more adaptive and is thus an evolutionary outcome for almost all terrestrial animals. Because of their unusual structure and unique locomotion systems, myriapods are an interesting subject both for robotics (Hoffman and Wood, 2011) and computer graphics research. Unfortunately, despite the abundant works in bipedal (human) (Van Welbergen et al., 2010; Wang et al., 2012), as well as the interest in quarupedal animals (dogs) (Coros et al., 2011; Ijspeert et al., 2007) and hexapodal animals (spiders) (Cruse et al., 1998; Cenydd and Teahan, 2013), work on animating myriapods is absent from the graphics literature.

To tackle the task of synthesizing realistic animations of myriapod as well as to gain a better understanding of how to coordinate hundreds of legs, we take a bottom-up, Artificial Life (A-Life) approach (Terzopoulos, 1999; Tu and Terzopoulos, 1994) to synthesizing virtual myriapoda. The A-Life approach starts the modeling of virtual animals with a biomechanical body situated in a physical environment, then builds motor controllers to produce locomotion, and equips the body with perceptual sensors such that higher-level behavioral animation can be achieved.

In this context, this dissertation introduces an A-Life framework for myriapod animation, which combines state-of-the-art physics-based simulation, an innovative biomechanical structure, and novel legged locomotion and swimming controllers. Our implemented system can synthesize physically and biologically plausible animations that could be adopted for motion picture CGI special effects and real-time game applications. Moreover, it can help us gain insights into these fascinating creatures' terrestrial and aquatic locomotion abilities.

Fig. 1.2 shows an overview of our framework, emphasizing the interplay between the locomotion control system and the physical simulation. The former

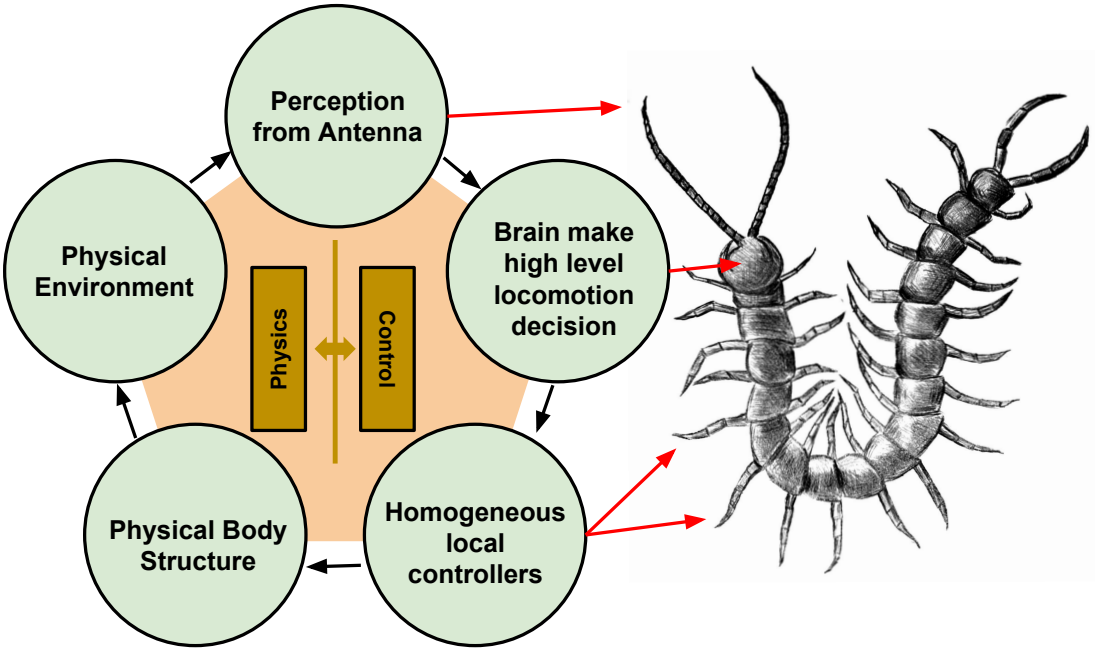


Figure 1.2: Overview of our A-Life animation framework. The physical environment and controller influence each other through antenna and contact sensors.

is composed of identical, modular, local controllers plus a motor center in the creature's head segment that governs higher-level locomotion variables such as ambulatory speed and turning angle. The decentralized leg control system is easy to understand and computationally inexpensive, yet robust enough to allow the simulated creature to walk over irregular surfaces. A pair of mobile antennas enhance the realism of our animated myriapods, but also function as environmental sensors that support their adaptive behaviors. A simple Braitenberg-vehicle-like (Braitenberg, 1986) behavioral mechanism proves to work well for obstacle avoidance and foraging. By positioning obstacles and food sources, animators can easily induce our simulated myriapod creatures to follow desired paths over irregular terrain.

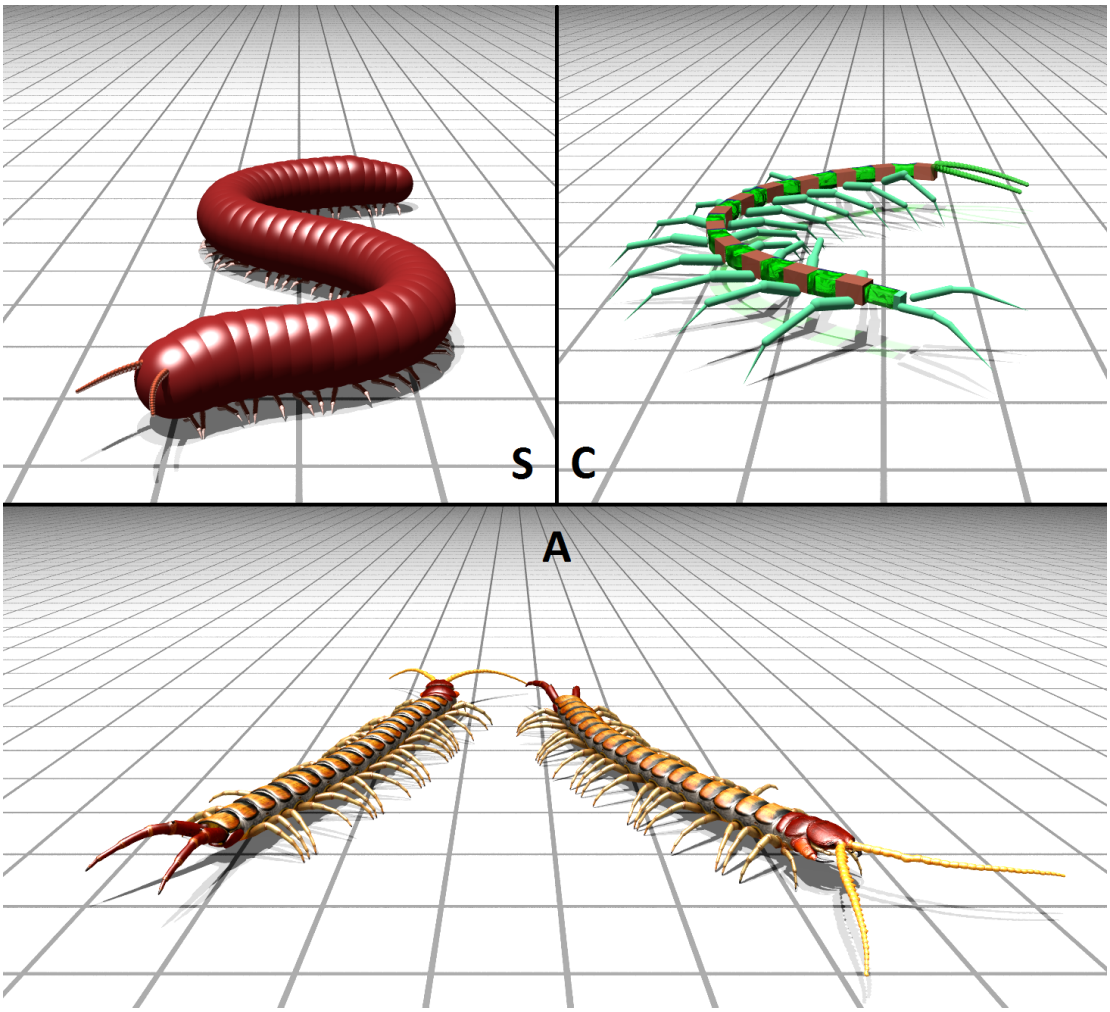


Figure 1.3: Forming the letters 'SCA' from the bodies of myriapods of different morphologies simulated within our framework, including (S) a millipede of 58 body segments, (C) a myriapod virtual robot with 12 body segments, and (A) two giant centipedes (Scolopendra) with 19 segments.

1.1 Contributions

This is the first dissertation in the field of Computer Graphics to address the modeling and animation of myriapoda. The primary technical contributions include the myriapod biomechanical model, a real-time hybrid simulation framework, a decentralized terrestrial locomotion controller for myriapoda, and an optimized anguilliform swimming controller, as described below:

Myriapod Biomechanical Model:

The novel biomechanical body model developed in this dissertation (Fig. 3.1) is specifically tailored to myriapod creatures. Insects with different body sizes, leg sizes, and number of segments can be simulated (Fig. 1.3). The Model comprises both rigid and deformable segments, mimicking the exoskeleton and internal soft tissues of a myriapod's biological body. The hybrid structure ensures ease of control through the rigid components while emulating the deformable nature of myriapod bodies by applying a fast, robust elasticity simulation method (Stomakhin et al., 2012). The legs are connected to rigid segments and, when in contact with the ground, can poise those segments directly. This allows the use of inverse kinematics to infer the desired rotational angles of the leg joints given the 3D configuration of the rigid segment. Via rest-shape morphing during elasticity simulation, the deformable segments that connect the rigid segments can actuate like muscles, allowing the creature to flex its body proactively. With adequate fluid coupling and coordinated muscle actuations, the biomechanical model can locomote in aquatic environments. A pair of antennae extend from the head of the creature to collect sensory information about food, obstacles, and pheromonal signals.

Real-time Hybrid Simulation Framework:

We would like our myriapod animations to exhibit as much physical realism

as possible and ultimately to run in real time. At the lowest level of our A-Life framework, the aforementioned biomechanical model is situated in its virtual environment, and its locomotion and ancillary movements are computed by numerical methods, specifically by a novel real-time hybrid simulation system. Our hybrid, kinematic/dynamic approach in conjunction with the rigid/deformable body structure enables efficient (forward and inverse) kinematic control of the relatively light legs of the myriapod, which greatly eases the design of locomotion controllers, while achieving realistic, physics-based motion and deformation of their relatively heavier body segments. Our system incorporates three aspects of physics-based simulations—rigid-body simulation, elastic deformable-body simulation, and fluid simulation. For the deformable segments, our elasticity simulation method replaces conventional mass-spring-damper systems, allowing rest-shape morphing such that muscular contraction and relaxation can be emulated. Also, an easily implementable two-way coupling method is employed to connect the deformable and rigid segments. To achieve aquatic locomotion, we introduce one-way coupling of the rigid and deformable bodies with a simple fluid model. Finally, we couple the biomechanical body model to a shallow-water simulation to synthesize, in real time, wakes along its locomotion path (see Figs. 5.7 and 5.5).

Decentralized Terrestrial Locomotion Controller:

We develop a biologically plausible decentralized and distributed leg control system that is suitable for different body morphologies and enables our simulated creatures to locomote over arbitrary surfaces with an emergent, naturally wave-like ambulatory gait (see Figs. 1.3, 6.2, and 6.1). The terrestrial locomotion system for myriapods is comprised of homogeneous segmental controllers that essentially represent a state machine for the rigid segments and legs. Our decentralized system is biomimetic and produces periodic leg

patterns as an emergent phenomenon rather than via pre-programmed global coordination. The creature's head takes sensory input from its antennae and make high-level adjustments in speed and turning direction. We enable our simulated creature to locomote over arbitrary 3D-mesh surfaces, such as a rock or skull.

Optimized Anguilliform Swimming Controller:

Four things are done to make our myriapoda swim quickly and efficiently in water: First, the deformable segments are turned into muscles by morphing their rest shapes during simulation, similar to the variation of spring rest length in conventional mass-spring-damper systems. Second, we implemented the physics of rigid and deformable segments in simple water, which provides propulsion forces to the creature's body. Third, a simple control model generates an anguilliform swimming pattern for myriapoda in water. Finally, we propose that instead of finding just one set of global optimal parameters for the swimming controller, it makes more sense to find optimal parameters for each speed. To this end, we perform heuristic search in the two-dimensional control-parameter space and find optimal parameters for each speed, optimal in that they are the most energy efficient to maintain that speed. Our result corresponds well with neurological studies of lamprey swimming. Like our decentralized leg locomotion controller, our swimming controller in conjunction with the dynamics simulation gives rise to anguilliform swimming as an emergent phenomenon.

1.2 Overview

The remainder of the thesis is organized as follows: Chapter 2 surveys relevant prior work on autonomous virtual creatures, biomechanical modeling, legged and swimming locomotion control. Chapter 3 presents our hybrid, kinematic/dynamic

model of myriapod bodies. In Chapter 4, we introduce our decentralized legged locomotion control system for our virtual Myriapoda. Chapter 5 considers the physics-based animation of anguilliform swimming and the optimization of swimming controller parameters. Chapter 6 showcases our implementation results and provides additional details regarding environmental interaction. Chapter 7 concludes the dissertation with a discussion and ideas for future work.

Appendix A details the simulation of deformable segments in the biomechanics model using the finite element method (FEM). Appendix B presents the derivation of shallow-water equations and the MAC grid method for numerically solving those equations. Appendix C presents the details of our optimization method. Appendix D provides the details of the state updates in each simulation frame. Appendix E presents the details of the methods used to render the images appearing in the dissertation and the supplementary demonstration videos.

CHAPTER 2

Related Work

Despite the absence of prior work directly addressing the modeling and animation of myriapods, in broader sense our efforts are well situated in the computer graphics and A-Life literature concerning the biomechanical modeling of animals. In particular, we were inspired by prior works on legged and aquatic locomotion. Such works often cut across various fields, such as physics-based simulation, biomechanics, robotics, etc.

2.1 Autonomous Virtual Creatures

Graphics researchers have done interesting and important work on simulating autonomous creatures in physics-based virtual environments. These creatures can walk on terrain (Guo et al., 2013; Holmes et al., 2006), swim in water (Tu and Terzopoulos, 1994; Tan et al., 2011), and fly (Wu and Popović, 2003). Reynolds (1987) simulated the group locomotion of all three types by applying a distributed behavioral model. In (Miller, 1988), mass-spring-damper systems are used to generate animation of snakes and worms. Tu and Terzopoulos (1994) introduced an artificial life framework for simulating autonomous creatures, including mass-spring-damper systems for the biomechanical simulation of deformable piscine bodies capable of producing muscle-based locomotion. Such biomechanical models are suitable for animals lacking legs. Subsequent work on salamander simulation (Ijspeert et al., 2007) used rigid-body dynamics to simulate the 4 legs with 8 kinematic degrees of freedom (DOF) and a global artificial neural network to work

as a central pattern generator (CPG) for controlling them. Beer (1990) built a 2D cockroach equipped with simple sensory feedback and artificial neural networks, demonstrating six-legged locomotion and adaptive behaviors in a complex environment and they also built a hexapod robot (Beer et al., 1992). Automated learning algorithms have been developed to train optimal locomotion controllers for simulated creatures. Sims (1994) applied genetic algorithm to naturally select both morphology and locomotion method of its simulated creatures. Grzeszczuk and Terzopoulos (1995) presented a bottom-up multilevel strategy for learning muscle controllers.

2.2 Biomechanical Modeling

Because of their simplicity and computational efficiency, mass-spring-damper systems were a popular method for biomechanical modeling of virtual creatures (Tu and Terzopoulos, 1994; Miller, 1988); however, their uniaxial elements do not model 3D material properties, and the biomechanics of the simulated animal body depends on how the spring-damper elements are assembled and how their parameters are tuned, which can be tricky. Alternatively, robust and efficient simulation of continuum mechanics based deformable objects is now possible (Stomakhin et al., 2012; McAdams et al., 2011). Shinar et al. (2008) create realistic creatures by combining rigid and elastic simulation, where the elastic components are passive and the creature is actuated by internal rigid bones. In (Coros et al., 2012), deformable objects are animated by changing the rest shape of the deformable mesh over time. State-of-the-art muscle models (Chen and Zeltzer, 1992; Teran et al., 2003) also employ continuum mechanics based elasticity. Our work on Myriapoda incorporates elastic and rigid simulation to achieve hybrid, kinematic/dynamic simulation.

2.3 Legged Locomotion Control

Graphics and robotics researchers have devoted considerable effort to legged locomotion (Raibert and Hodgins, 1991; Golubitsky et al., 1998; Wang et al., 2012; Guo et al., 2014). Holmes et al. (2006) survey the modeling, analysis, and challenges associated with insect locomotion dynamics. There has been much robotics research on insect-inspired four, six, and eight legged robots (Kimura et al., 2007; Raibert, 2008; Saranli et al., 2001), predominantly based on etiologic and neurophysiologic knowledge about cockroaches and stick insects (Full and Tu, 1991). Although there exists no prior computer graphics work specifically on Myriapoda animation, robotics researchers have started building centipede robots (Odashima et al., 1998; Inagaki et al., 2003, 2011). However, there exists a substantial body of graphics literature on bipedal (Hodgins et al., 1995; Faloutsos et al., 2001; Van Welbergen et al., 2010), quadrupedal (Skrba et al., 2008; Coros et al., 2011), and hexapodal (Cenydd and Teahan, 2013; McKenna and Zeltzer, 1990) figure animation. McKenna and Zeltzer (1990) proposed a forward dynamics algorithm for locomotion coordination of a simulated cockroach, capable of navigating irregular terrain. Attention has been paid to legged locomotion control from entomological studies by building accurate biomechanics models of leg muscles (Wang et al., 2012) with the purpose of accurately simulating ambulation for medical purposes. Specialized neural CPG structures have been the subject of experiments (Mellen et al., 1995) and simulations (Ijspeert et al., 2007; Ijspeert, 2008). Coupling simplified dynamics such as the Inverted Pendulum Model with a learnt CPG has been a popular approach for multi-legged locomotion (Coros et al., 2011; Tsai et al., 2010). The Walknet of Cruse et al. (2000, 1998) successfully reproduced the behavioral properties of hexapod locomotion via a decentralized organization of the control system—neither the movement of any single leg nor gait coordination is centrally preprogrammed, yet adaptivity and flexibility emerge from each leg controller applying only a few simple, localized rules involving the

states of neighboring legs. This decentralized controller paradigm also forms the basis of our system design.

2.4 Aquatic Swimming Animation

Aquatic creature locomotion and human swimming has attracted interest from researchers in multiple fields. Biologists have done extraordinary research (Bone, 1975; Bergmann and Iollo, 2011) to comprehend the different swimming patterns and the underlying mechanisms via observation, experimentation, and simulation. Vortex and wake geometry as well as locomotor forces can be analyzed with the help of digital particle image velocimetry (DPIV) (Drucker and Lauder, 2002). Anguilliform swimming is the primary type of locomotion pattern for many aquatic species such as eels and snakes. Those creatures have a slender body to propel themselves in water by generating periodic tail undulations while keeping the anterior of the body straight. Anguilliform swimming was first studied by Gray (1933), and has gained a substantial amount of attention both in terms of the neurological pattern of controllers (Ekeberg, 1993) and the hydrodynamics of the motion (Kern et al., 2008). Roboticists have built snake-like robots that can perform both serpentine terrestrial and aquatic locomotion (Nor and Ma, 2014). Transeth et al. (2009) present a good survey of snake robots.

Graphics researchers, with the purpose of generating realistic animations, have built both kinematic (Gates, 2002) and dynamic models (Tu and Terzopoulos, 1994) for snakes and fish. Simple mass-spring-damper biomechanical models were used in various important works on swimming creatures (Bowtell and Williams, 1991; Tu and Terzopoulos, 1994; Ijspeert et al., 2007). Recent developments on human swimming (Kwatra et al., 2010; Si et al., 2014) and animal swimming (Tan et al., 2011) have applied advanced simulation techniques to model body-fluid interaction (Arash et al., 2003; Carlson et al., 2004), with the hope of generating

more accurate results. However, gains in simulation precision come with compromises in real-time performance. Also, the computation time required to solve fluid dynamics equations even at moderate resolution curbs the optimization of controllers through forward simulation.

Given these constraints, we take a different approach that specifically deals with slender-bodied creatures such as myriapoda. By referring to previous works on anguilliform swimming (Ekeberg, 1993; Grzeszczuk and Terzopoulos, 1995), we narrowed our control space to include only two parameters—undulation frequency and segmental phase shift. By coupling our myriapoda model with simple fluid, our simulation is fast enough to run experiments that allow the examination of how the change of the two parameters would affect the simulated model's stable speed and energy consumption rate (ECR) during swimming. We also perform an exhaustive search to find the optimal pair of parameters for a continuous range of speeds at which the model would naturally swim.

CHAPTER 3

Biomechanical Model of Myriapoda

Prior work on animating arthropods has been limited to insects with only one major abdominal segment, such as a stick insect or a spider. The unique, elongated and deformable body structures of myriapoda require special attention to the biomechanics. The numerous slim legs of typical myriapoda have much lower mass than does the remainder of their body, which enables us to take an efficient hybrid kinematic/dynamic approach to modeling them in order to achieve real-time animation performance.

3.1 Overview of Body Structure

Fig. 3.1 illustrates the physical body structure that we developed for our artificial myriapoda. First, we simplified our model by neglecting the mass of the legs, thus making them kinematic. Initially, we used only deformable segments to model the body. However, coupling the slim legs to the deformable segments significantly increased the control challenge, since the leg attachments have a very local effect on the deformable body. By contrast, connecting a leg to a rigid segment emulates the natural exoskeleton of millipedes and enables the leg to easily control the position and orientation of the entire segment.

Thus, in our model, a pair of rigid, kinematic legs actively control each of the rigid dynamic exoskeletal segments, which are then connected in series using dynamic elastically deformable segments. When hanging in gravity, the body

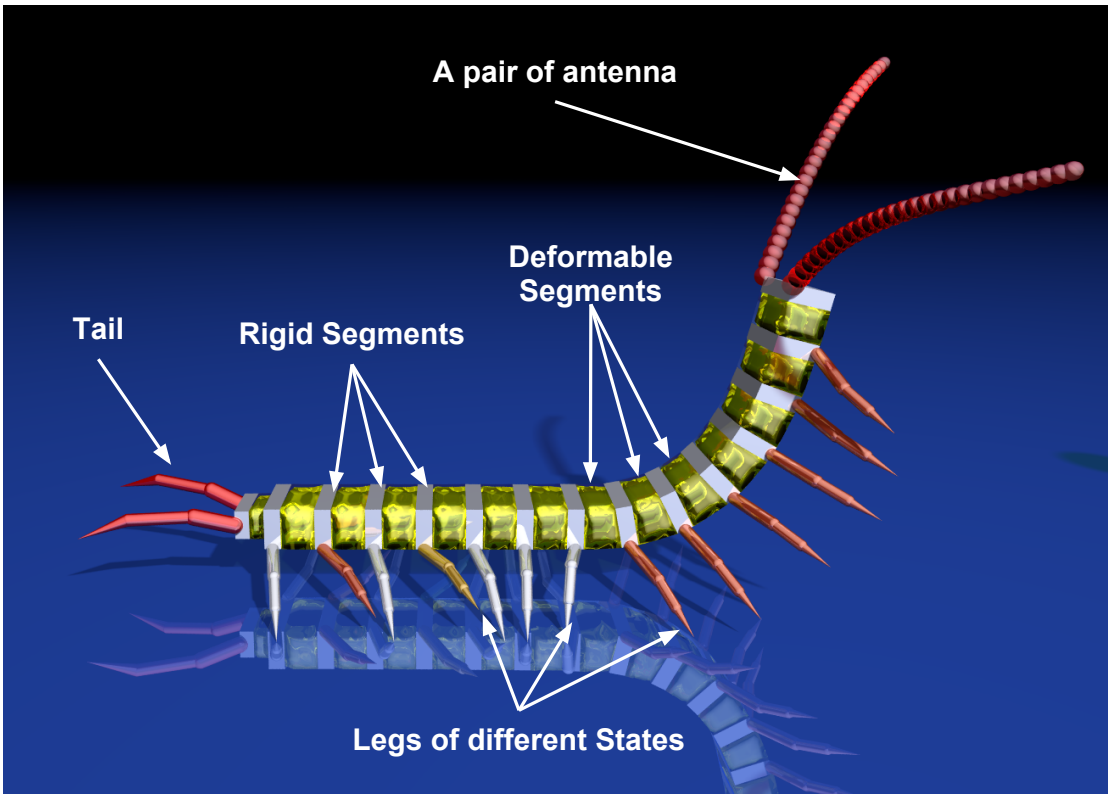


Figure 3.1: Myriapod body structure, comprising rigid segments (grey) with rigid legs, deformable segments (yellow), a pair of deformable antennas, and tail. The color of each leg indicates its current state (see Fig. 4.4).

model will naturally elongate (Fig. 4.4). The motion of the rigid segments is governed by rigid-body dynamics with collision and friction (Baraff, 1997). The deformable segments are governed by energetically consistent invertible elasto-dynamics with a fixed corotational constitutive model (Stomakhin et al., 2012). Beyond their passive dynamics, the deformable segments can also actuate like contractile muscles to drive swimming in water.

3.2 Deformable Segments

For our application, which requires real-time elastic simulations with nearly unconditional stability, we employ the fixed corotated constitutive model ((Stomakhin

et al., 2012)), which is given by

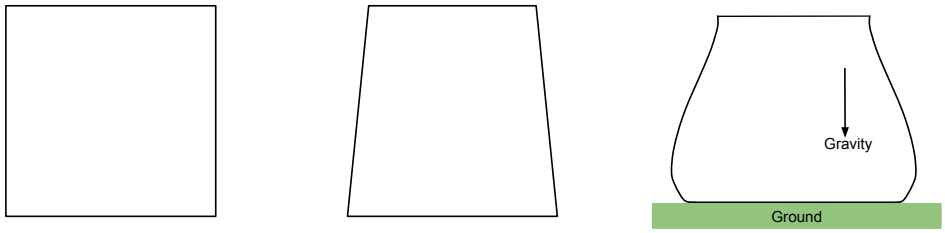
$$\Psi = \mu \sum_i (\sigma_i - 1)^2 + \frac{\lambda}{2}(J-1)^2, \tag{3.1}$$

where μ and λ are Lamé parameters, J is the determinant of the deformation gradient $\mathbf{F}$, and σ_i are the singular values of $\mathbf{F}$. This energy model is a smooth extension of the corotated model, and is extremely robust to large deformations.

Appendix A presents more details of the deformable segments and their simulation using the finite element method (FEM). The simulation domain is discretized into tetrahedral elements. The deformation gradient $\mathbf{F}$ is derived from the current instant shape (Fig. 3.2c) of an element in reference to its natural shape in the rest state (Fig. 3.2b). The fixed corotational energy as a functional of $\mathbf{F}$ generates the hyperelastic force to minimize the shape distortion. If the rest shape remains unchanged, unbalanced external forces will cause passive dynamics of the object. Natural animal motions are initiated by the active actuation of internal muscles. As an integral part of our biomechanics model, muscle actuation for the myriapod is achived by morphing the rest shape of the deformable segments from its initial shape (Fig. 3.2a) to the desired configuration. Internal forces will be generated automatically via simulation to drive the body. However, the actuations from many muscles need precise coordination to achieve the locomotion goal of entire body in the simulated physical environment. Such challenging tasks are usually handled by learning methods (Grzeszczuk and Terzopoulos, 1995).

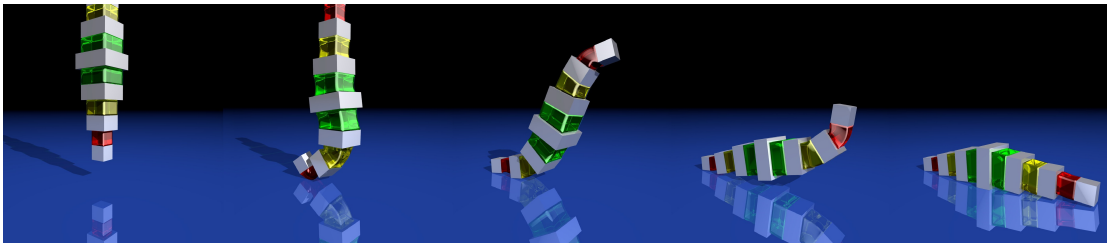
3.3 Coupled Rigid and Deformable Segments

Several authors have demonstrated two-way coupling between rigid and deformable objects (Shinar et al., 2008; Sifakis et al., 2007; Baraff and Witkin, 1997). We couple the rigid and deformable parts of our alternating rigid/deformable myriapod

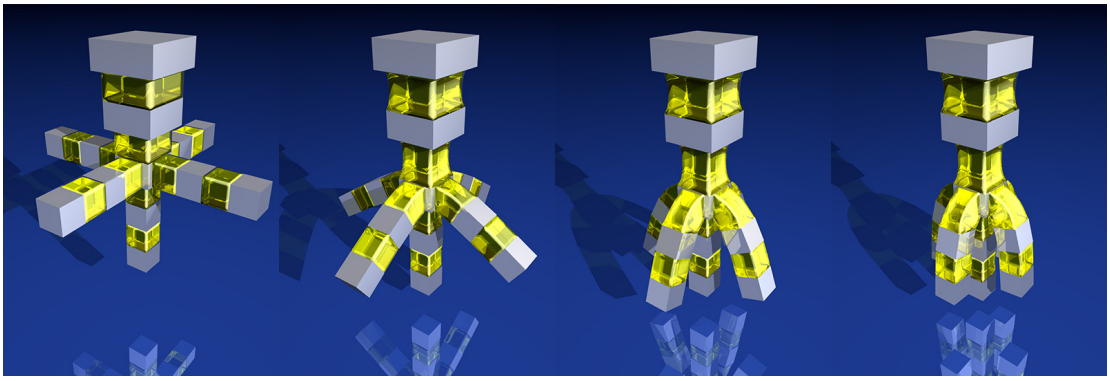


(a) Initial shape (b) Rest shape (c) Instant shape

Figure 3.2: (a) Initial shape of a 2D segment, representing the shape of the segment in equilibrium when there is no external force and internal actuation. (b) Rest shape of the segment at a certain instant, morphed from the initial shape. The shape of the segment in equilibrium in the absence of external forces will be the same as the rest shape; (c): Instantaneous shape of the segment under gravity, supporting force from the ground, internal actuation from rest shape in (b) and inertia.



(a) A linear structure, dropping onto a slippery ground plane



(b) A structure similar to a ceiling light with top fixed

Figure 3.3: The dynamics of coupled rigid and deformable segments.

body structure using the following interleaved simulation method:

1. Boundary nodes on deformable tetrahedral meshes that make contact with rigid segments are fixed to those rigid segments.

2. At each simulation time step, the elastic forces from the mesh deformation computation evaluated at the contact nodes are applied as external forces to their respective rigid bodies.

3. Finally, the positions of the contact nodes are updated in accordance with the rigid body dynamics simulation.

Since the deformable and rigid parts time-stepped independently, our interleaved method is not fully coupled as in (Shinar et al., 2008); however, by virtue of the robust, invertible elasticity method of (Stomakhin et al., 2012), stability is not an obvious issue. Our approach achieves two-way coupling yet is very straightforward to implement. Fig. 3.3 demonstrates the simulation of coupled rigid-deformable objects.

3.4 Kinematic Legs

The legs of the virtual myriapod are simulated as rigid links that rotate around joints. The leg tip and root positions are inputs and outputs for our locomotion control system and the leg joint angles are determined by an inverse kinematics (IK) solver.

Referring to Fig. 3.4, the leg inverse kinematics algorithm takes as input the position of the leg tip P and leg root O and computes the joint angles θ, α, and β. Fixing γ for simplicity, the steps to solve for θ, α, and β are as follows:

1. By restricting all the leg segments to lie in a plane, θ can be computed independently from α and β using the positions of P and O, as Fig. 3.4(2)

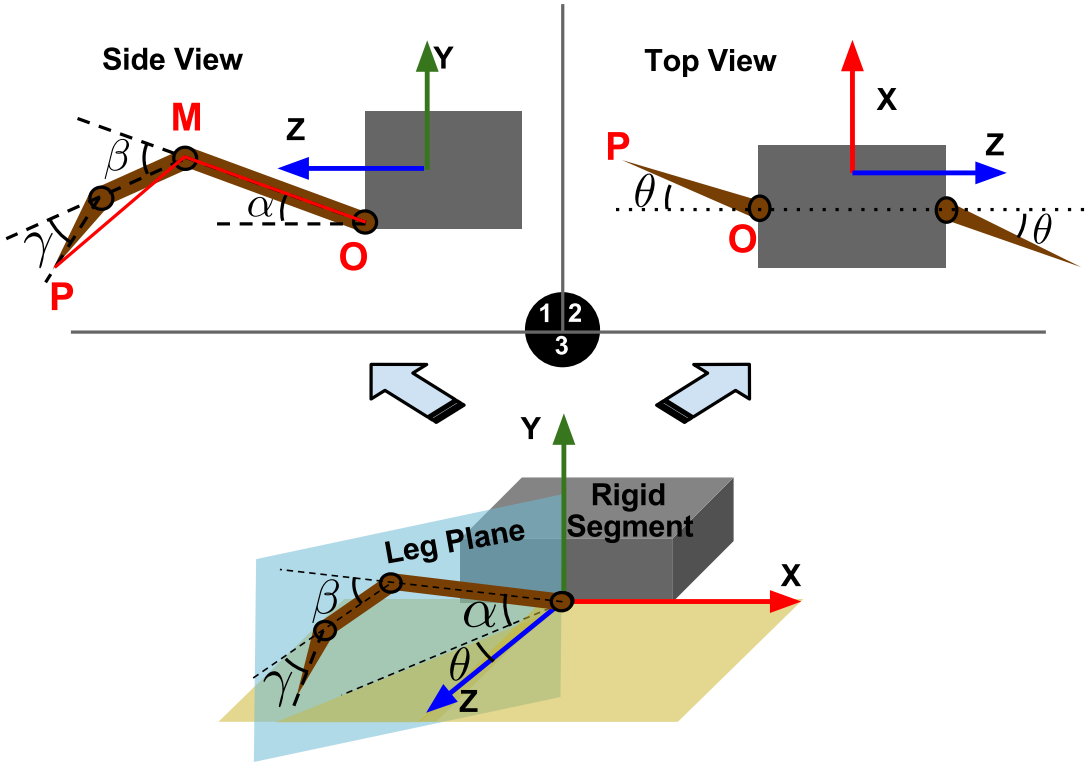


Figure 3.4: Myriapoda leg structure. The leg rotates around the y axis (θ) and can articulate in the leg plane around joints O and M (α, β). P is the tip of the leg.

shows.

2. Angles α and β can be computed by solving for the position of point M (Fig. 3.4(1)), which is determined by rotating segment OM around point O and segments PM (with γ fixed) around point P. The two circles will either not intersect (no solution), intersect at one point (tangent), or intersect at two points. If a convex solution with $\beta > 0$ exists for point M in the leg plane, then α and β are computed using simple geometric calculations.

Although it falls short of full biomechanical simulation and control, our efficient method meets the visual realism requirements. Furthermore, given the numerous legs involved, the animation of myriapoda does not present anywhere near as severe an "Uncanny Valley" to observers as does human bipedal locomotion.

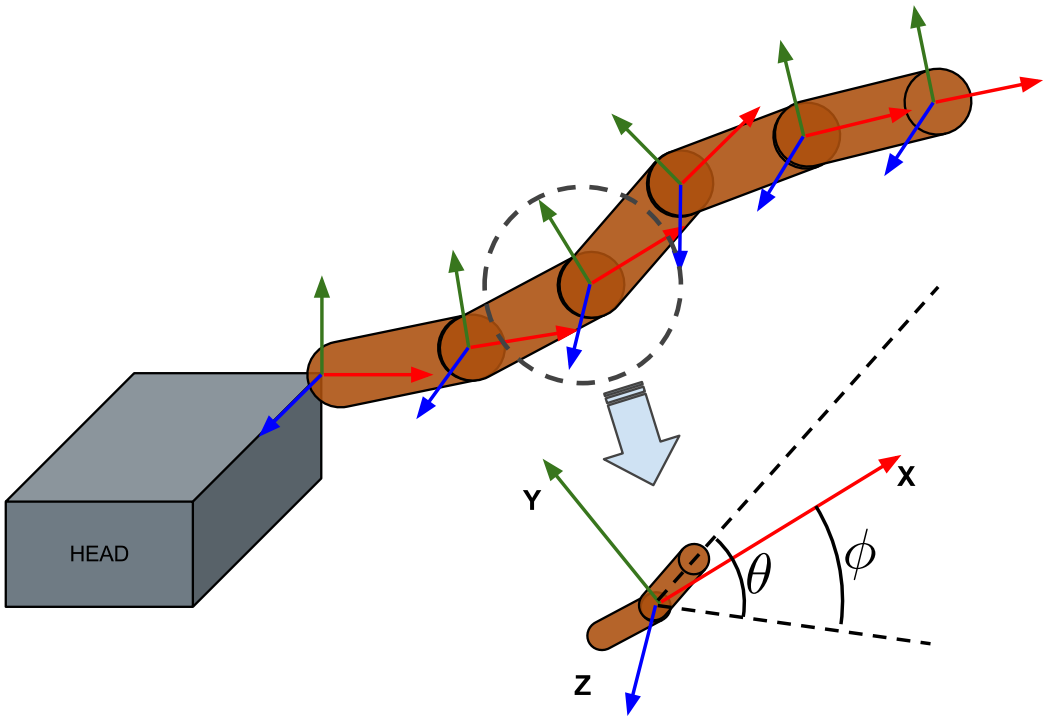


Figure 3.5: Antenna modeling and animation.

Hence, rather than trying to improve the realism of individual leg movement by modeling complex leg biomechanics, and pay the associated computational costs, we have focused on the development of the locomotion control system described in Chapter 4, which can realistically synthesize the natural wave motion pattern of the numerous legs.

3.5 Antennae Modeling and Animation

The antennae of myriapod function as sensors and is essential to its adaptive behavior. To increase visual authenticity of our animations, we wrote a procedure to generate their movement. We model the antennae as a chain of N short links, each of which has two rotational degree-of-freedom (θ, ϕ) joints relative to the previous link (Fig. 3.5). We observed from video footage real centipedes that during locomotion the antennae will articulate randomly for exploration.

Meanwhile, there is an outward traveling wave from the root of the antenna to the tip. To synthesize lifelike antenna movements, we devised the following two-step method: First, to achieve randomness in antenna rotation, we set random target rotations θ_T, ϕ_T for the root link ($i = 0$). The root link will rotate toward this target rotation at a constant speed (ω_0) and will generate new target rotations when the current ones are achieved. The formula is simply $[\theta_0, \phi_0]^{t+\Delta t} = [\theta_0, \phi_0]^t + \omega_0 \Delta t$. Second, for the links ($i = 1, 2, \ldots, N-1$), their relative rotation angles (θ_i, ϕ_i) are sampled from a wave function $[\theta_i, \phi_i] = A_{[\theta,\phi]} \sin(\omega t - \lambda i) + B_{[\theta,\phi]}$, where A_θ, A_ϕ, B_θ, and B_ϕ are constant amplitudes and base values, and ω and λ determine the wave frequency and wavelength.

3.6 Antennae-Based Sensory Perception

In our real-time simulator, food and obstacles can be placed in the scene to elicit the emergent behavior of an autonomous virtual myriapod exploring its environment. In fact, an animator can place food particles and vertical obstacles on the terrain surfaces to guide the locomotion path of the creature. We have developed two simple biologically rooted mechanisms to enable the sensing of food and obstacles by our simulated myriapoda.

The antennae detect obstacles through physical contact, when an antenna-obstacle collision occurs, the θ_T, ϕ_T, and ω_0 will be adjusted to retract the antenna in the direction of the obstacle's surface normal. A physical contact signal will be passed into the brain for locomotion adjustment. Our simple steering mechanism is similar to that in Braitenberg vehicles (Braitenberg, 1986)—whenever there is a contact signal from the left antenna, the head will enter avoidance mode and turn right, and vice versa. In order to avoid the myriapod becoming trapped by walking perfectly straight into a wall, one antenna has higher priority when both antennas sense a collision.

Food is modeled as point sources with intensity gradients that decrease with squared radial distance. The antenna can sense food intensity in the environment. Once the stimulus exceeds a certain threshold, it will send an intensity signal to the brain. The myriapod will turn in the direction of maximal food gradient. Our simulations shows that this simple foraging method works very well. Furthermore food sources can be used to plot the path of an artificial myriapod.

CHAPTER 4

Decentralized Locomotion Controller

In this chapter, we present a decentralized locomotion controller for simulated myriapoda that synthesizes realistic wave motion global legged locomotion patterns.

4.1 Overview

The Dragon Dance, a tradition that is performed during Spring Festivals in Chinese culture, requires approximately 9 to 15 dancers to control a long dragon whose segments are connected by joints. Three key observations can be made about this dance: First, the person controlling the head of the dragon makes the locomotion decisions and implements these decisions using his/her own two legs. Second, each of the remaining performers have one major locomotion goal—to follow the performer ahead of them. Third, applying the previous two simple local rules results in a global emergent behavior—the global wave pattern of the dragon movement.

The leg control model that we have developed for our myriapod creatures applies the idea of the head segment leading and the subsequent segments following. At any moment, the head segment synthesizes information sensed by the antennas to make high-level locomotion decisions, such as turning and changing speed. These decisions are processed by the head segment to determine the desired future configuration of the head; specifically, the position and orientation of the

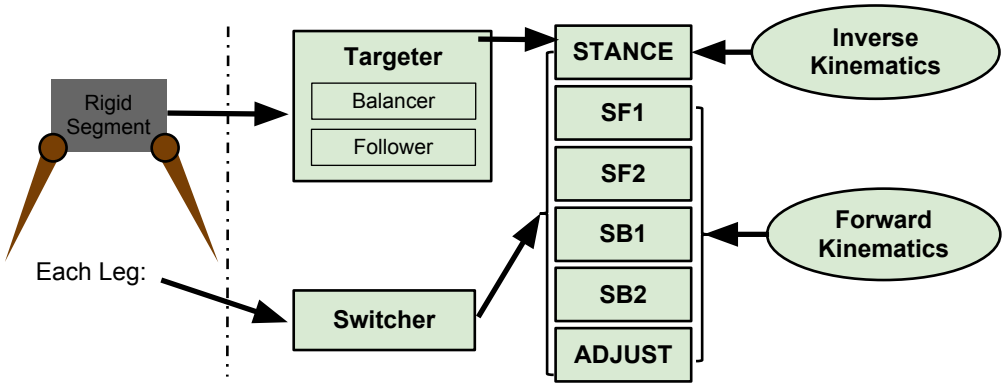


Figure 4.1: The local locomotion controller is comprised of a Targeter for the segment and two Switchers for left and right leg. Targeter is responsible of balancing the segment as well as following the previous segment; The Switcher's duty is to monitor and apply transition of six different leg states (Fig. 4.3) during locomotion.

rigid head segment in the next time step. The desired configuration is provided to the leg controller for execution. The subsequent segments have identical local leg controllers, which comprise two components shown in Fig. 4.1—(i) a *Targeter* that determines the desired target configurations of the segment and (ii) left and right leg *Switchers* that switch the leg states (Fig. 4.3).

A rigid segment can have two states: *supported* and *unsupported*. When the segment is supported, it is kinematically transported to the desired position and orientation by the legs. In the unsupported state, its motion is governed by the rigid-deformable coupling dynamics. The major benefit of this hybrid, kinematic/dynamic approach is that it circumvents complex leg dynamics to drive the body dynamics without sacrificing appreciable dynamical realism in the body. The legs are rotated via forward kinematics in most states and they control the segment through inverse kinematics during the stance state. The details of the six leg states and the switching mechanism are discussed in Section 4.3.

4.2 Targeter

A Targeter in each rigid segment i continually outputs its configuration (position $\mathbf{c}_i$ and orientation $\mathbf{R}_i$) to the legs. When the segment is unsupported, it outputs the updated configuration computed by dynamic simulation, whereas when the segment is supported, it outputs the desired configuration of the rigid segment for the segment's legs to achieve kinematically. As is described later, the left and right leg can be synchronized in order to maximize the time that a rigid segment is in the supported state. The Targeter has two objectives (Fig. 4.2)—to follow the previous segment, which is accomplished by a *Follower*, and to balance the body, which is accomplished by a *Balancer*. The Follower generates a target position $\mathbf{c}_i^T$ for the current rigid segment by modifying its current position in the direction of the link vector $\mathbf{l}_i = \mathbf{c}_{i+1} - \mathbf{c}_i$ such that the length of the link vector remains constant and it generates a target rotation $\mathbf{R}_i^T$ of the segment around its y axis so as to orient the x axis toward the link vector direction. The Balancer balances the y axis of the rigid segment toward ground surface normal vector $\mathbf{n}$ and adjusts the elevation of the rigid segment by further modifying the target position in the direction of $\mathbf{n}$. The combined adjustments of the current position and orientation by the Follower and Balancer result in a new target position and orientation that are outputted to the leg controllers as goals to achieve during the stance state.

4.3 Leg State Machine

4.3.1 Leg States

Typically, a leg will periodically cycle between 6 different states (Fig. 4.3) under the control of a Switcher. Referring to Fig. 3.4, a pose is uniquely defined by three rotational angles: θ, α, and β (γ is fixed). Angle θ is the rotation of the leg plane around the y axis of the rigid segment, while α and β are the 2 degrees of freedom

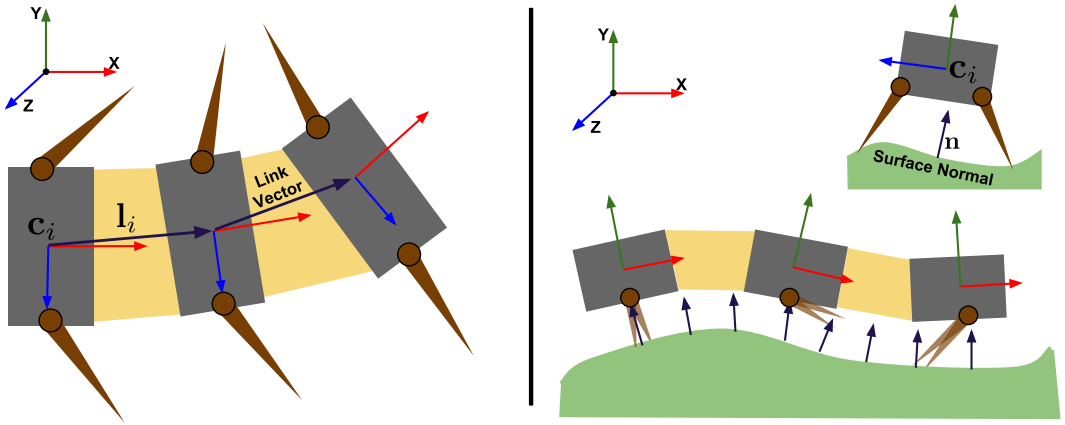


Figure 4.2: The Targeter of a rigid segment, comprising a Follower (left) and a Balancer (right). The Targeter computes desired configuration of the rigid segments and ask the legs to achieve it during STANCE state. With the target position $\mathbf{c}_i^T$ and orientation $\mathbf{R}_i^T$ of segment i initialized to its current position and orientation, the update equations for the Follower are $\mathbf{c}_i^T \stackrel{+}{=} k_1(|\mathbf{l}_i| - l_i^0)\mathbf{l}_i\Delta t$ and $\mathbf{R}_i^T \stackrel{*}{=} \mathbf{R}([0, \pm 1, 0], \omega\Delta t)$, with the $\pm$ sign determined by the sign of the projection of $\mathbf{l}_i$ onto the z axis, and the update equations for the Balancer are $\mathbf{c}_i^T \stackrel{+}{=} k_2(h_i - h_i^T)\mathbf{n}\Delta t$, where h_i and h_i^T are the segment's current height and target height, and $\mathbf{R}_i^T \stackrel{*}{=} \mathbf{R}([0, 0, \pm 1], \omega\Delta t)\mathbf{R}([\pm 1, 0, 0], \omega\Delta t)$, with the $\pm$ signs determined by the projection of $\mathbf{n}$ onto the x-z plane.

of the leg within the leg plane. Each leg state starts in one pose and ends at a target pose.

Sway Forward 1 (SF1)

Ideally, the leg sways forward up from the posterior extreme pose (PEP) ($\theta = \theta_m$, $\alpha = \alpha_S$, $\beta = \beta_S$) to the middle up pose (MUP) ($\theta = 0$, $\alpha = \alpha_M$, $\beta = 0$). This state allows the leg to leave the STANCE state and elevate off the ground.

Sway Forward 2 (SF2)

Ideally, the leg sways forward down from the MUP to the anterior extreme pose (AEP) ($\theta = \theta_M$, $\alpha = \alpha_S$, $\beta = \beta_S$). This state sends the leg to the STANCE state;

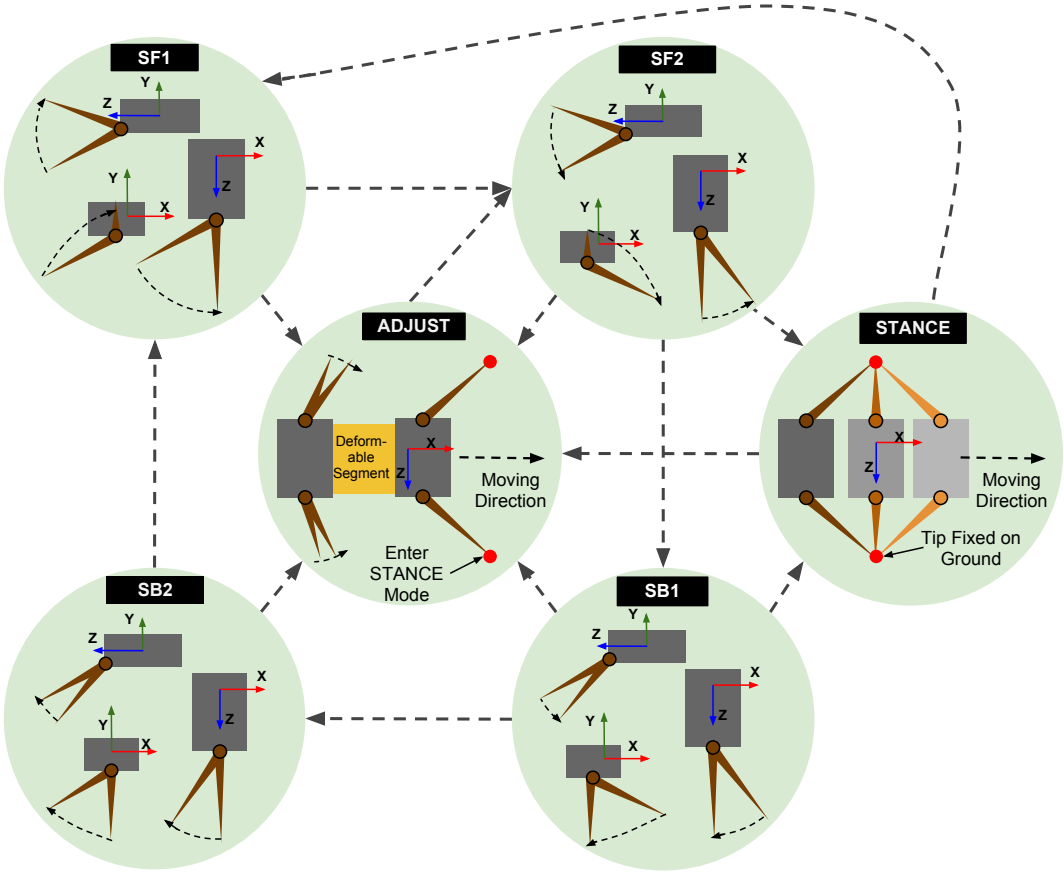


Figure 4.3: The six leg states and transitions.

Sway Backward 1 (SB1)

Ideally, the leg sways backward down from the AEP to the middle down pose (MDP) ($\theta = 0$, $\alpha = \alpha_m$, $\beta = 0$). This is to help the leg touch the ground.

Sway Backward 2 (SB2)

Ideally, the leg sways backward up from the MDP to the PEP. This is the last state of a cycle comprising SF1, SF2, SB1 and SB2.

STANCE

In the stance state, the leg tip is fixed on the ground and the root of the legs push the segment forward. Ideally, it starts at the PEP and ends at

the AEP. Inverse kinematics is used to compute the rotational angles (θ, α, and β) from the tip and root positions, as is detailed in Section 3.4. The root position of the leg is calculated from the output of the Targeter. The leg will leave the STANCE state to enter SF1 when the IK solver cannot resolve the current outputted root position from the Targeter.

ADJUST

This state generates the collective wave pattern in the leg motions. A leg will enter the ADJUST state immediately after the previous leg enters the STANCE state. During the ADJUST state, the leg will move to a target pose (θ_T, α_T, β_T). The differences $\theta_T - \theta_M$, $\alpha_T - \alpha_S$, and $\beta_T - \beta_S$ determine the desired rotational phase difference between the legs. Changing their values will change the frequency of the leg wave patterns.

Note that the word "ideally" above indicates that during locomotion the legs do not always start or end in precise poses, but attempt to reach target start and end poses; e.g., when the ground is irregular, the SF1 state can end prematurely and enter the STANCE state, as will be detailed below. Only during the STANCE state will output from the Targeter be used, whereas in other states, the leg rotates by forward kinematics toward the target rotations.

4.3.2 Switcher

The following rules are used to update leg states by the Switcher associated with each leg:

1. A normal loop is from SF1 to SF2 to SB1 to SB2 and back to SF1 (Fig. 4.4(A)), this normal loop can be interrupted by STANCE and ADJUST states;
2. Whenever the leg tip touches the ground surface, enter the STANCE state;
3. When the IK solver cannot solve for the STANCE state, enter the SF1 state;

4. Whenever the previous leg enters the STANCE state, enter the ADJUST state;

5. At the end of the ADJUST state, enter the SF2 state;

For example, when the simulated myriapod is suspended off the ground , the legs will periodically go from the SF1 to the SB2 states without entering the STANCE and ADJUST states(Fig. 4.4(A)). If it is dropped to the ground, the legs will start to enter the STANCE and ADJUST states (Fig. 4.4(B),(C)). When the creature starts walking, its leg motions quickly converge to a wave pattern in which each leg undergoes the state cycle SF1→SF2→STANCE with short appearances of the ADJUST and SB1 states (Fig. 4.4(D)).

In particular, Rule 4 enforces a stable phase difference locally between legs, finally resulting in a wave-like leg formation during locomotion. The wave-like locomotion pattern has several merits: First it ensures that at any moment the body will be supported by a fixed ratio of legs such that stability is guaranteed. Second it allows each leg to stretch to extreme poses (AEP and PEP) with maximal energy efficiency, since the work done to raise and lower each leg can be reduced for a given locomotion distance.

4.3.3 Synchronization of the Left and Right Legs

Synchronizing the leg waves on both sides maximizes the duration that a rigid segment stays in the efficient IK-driven supported state. The synchronization of left and right leg rotations is achieved by adding an extra rule to the first segment's Switcher—when the left leg enters the SF1 state, the right leg will also immediately enter SF1.

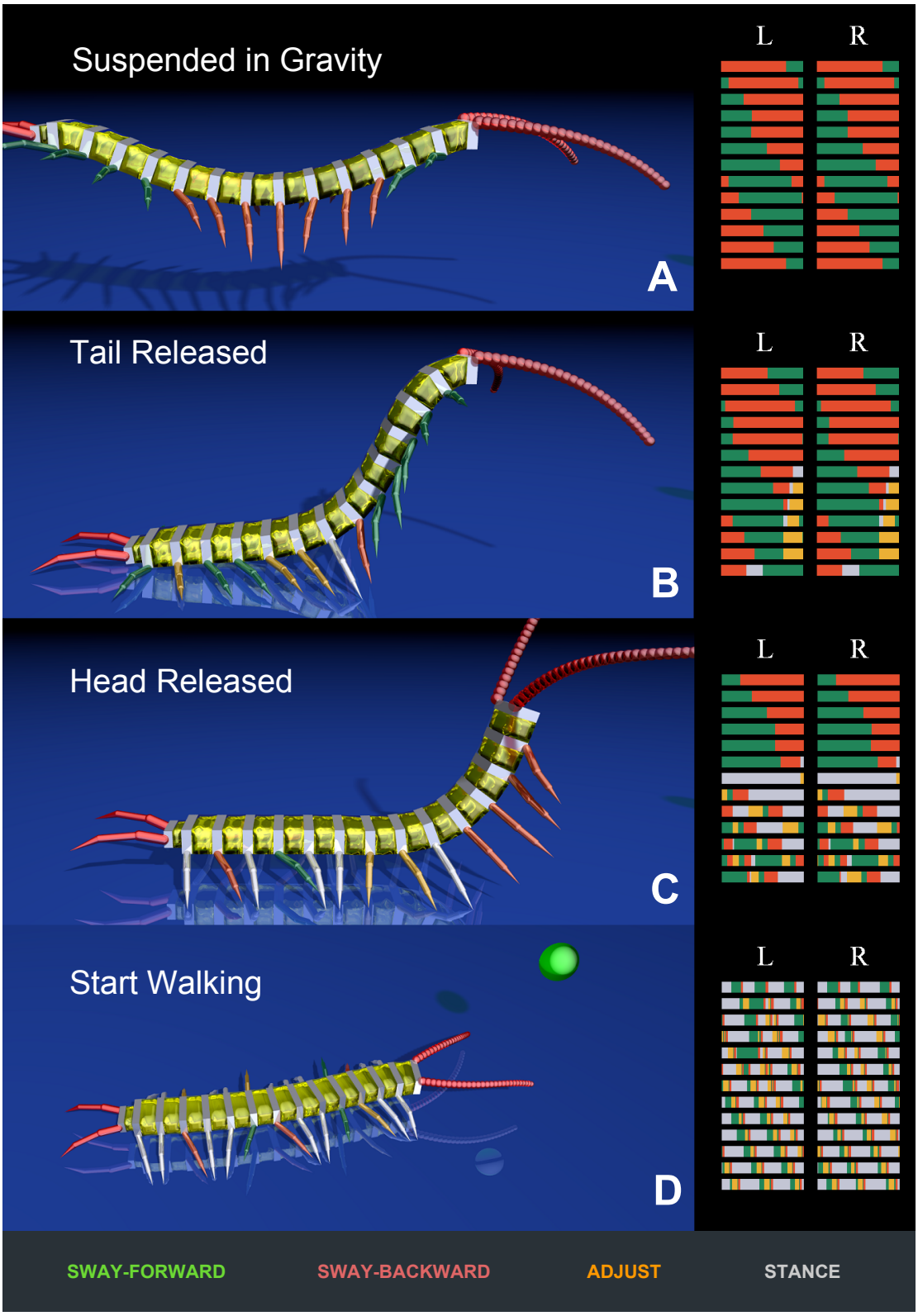


Figure 4.4: A centipede dropped to the ground. The right panel shows the states of each leg during the past second. The color scheme is indicated at the bottom.

4.4 The Head

The head segment is almost always kinematically updated by its own Targeter, which will employ the Balancer for balancing, but not have any anterior segments to follow. The Targeter updates the configuration of the head segment based on three variables: (i) a moving speed (the head will always move forward along its x axis), (ii) a turning speed around the y axis, and (iii) a binary turning direction. There are various brain modes that hold different values of the three variables. A special Switcher updates those brain modes as well as the variables based on internal states and signals from the antennae. The brain modes and associated switching mechanisms are as follows:

ADJUSTMENT: The head polls each segment to check if the majority of the legs are on the ground. If not, it will stay in the ADJUSTMENT mode and the head segment is controlled by the dynamics. This brain mode enables the animation of a creature dropped onto the ground to adjust its legs automatically before it starts walking (Fig. 4.4).

RANDOM-WALK: When not in ADJUSTMENT mode, the brain automatically enters random-walk mode. Turning direction and target turning angles are set randomly and renewed when they are attained.

AVOIDANCE: When the left/right antenna sends an obstacle contact signal, the brain enters the obstacle AVOIDANCE mode. The turning direction and speed will be adjusted to avoid the obstacle. After the obstacle is cleared, the head returns to RANDOM-WALK mode.

TROPHOTROPISM: similar to the obstacle AVOIDANCE mode, antenna sensed signals of food in the environment modify the turning speed and direction such that the creature orients itself toward the food. After the food is reached, the head returns to RANDOM-WALK mode.

CONTROL: The head is externally controlled by locomote and turn commands issued by a user.

For obvious reasons, the AVOIDANCE mode has a higher priority than the TROPHOTROPISM mode.

CHAPTER 5

Optimized Anguilliform Swimming

In this chapter, we introduce our approach to synthesizing optimized anguilliform swimming animations of our segmented biomechanical model as an emergent property of our physics simulation and a Frequency and Phase Shift (F-PS) controller.

5.1 Actuation of Deformable Segments

First, we transform the passive deformable segments into active muscles. Within the deformable segments, which are simulated as elastic bodies using the FEM method (Appendix A), internal stresses arise from the difference between the instantaneous shape and the rest shape (Fig. 3.2). In the absence of external forces, the stresses will restore the deformable segment towards its rest shape. Our deformable segments actuate themselves by morphing their rest shapes. At each time step, the rest shapes of the segments are updated, then the simulation generates the internal muscular force. For our purpose of generating horizontal swimming locomotion on the surface of water, we constrained the morphing of the rest shape to x-z plane (Fig. 5.1). Specifically, we use two parameters to control the morphing, l_{left} and l_{right}. We store the initial shape as a 3D mesh of size $l_x \times l_y \times l_z$ centered around the origin. For each node of the mesh, its position (x_0, y_0, z_0) in the initial shape will be updated to generate the morphed rest shape,

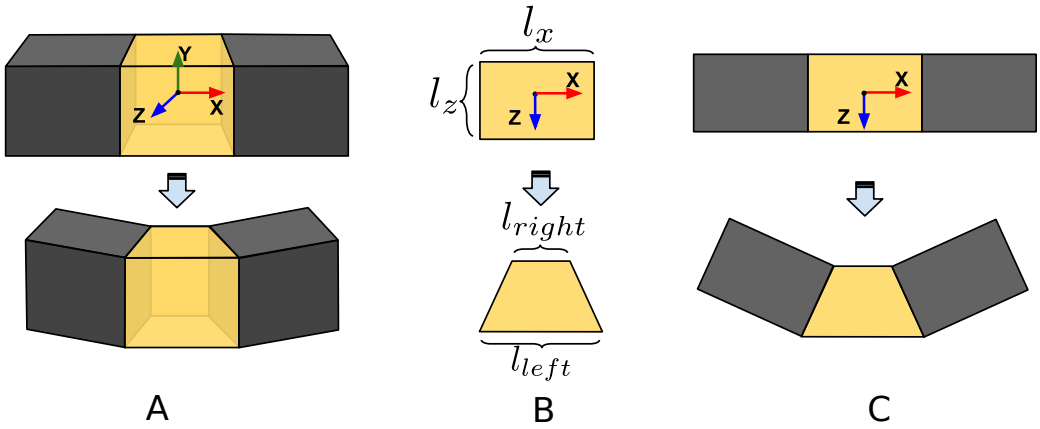


Figure 5.1: The rest shape of deformable segments morphed along the x axis, controlled by l_{left} and l_{right}. (A), (C) 3D and top-down views of two rigid segments (gray) connected by a deformable muscle segment (yellow), actuated to curve to the right. (B) Top-down view of a deformable segment's initial shape morphed to a new rest shape, which drives the flexing of the 3-segment body in (A) and (C).

as follows:

$$
\begin{aligned}
z_{ratio} &= 0.5 + \frac{z_0}{l_z} \\
x(t) &= \frac{x_0}{l_x} \times (l_{left}(t) \times z_{ratio} + l_{right}(t) \times (1 - z_{ratio})) \\
y(t) &= y_0; \quad z(t) = z_0;
\end{aligned}
\tag{5.1}
$$

Fig. 5.2 and Fig. 5.3 demonstrates the results of muscle actuation. Especially, Fig. 5.3 shows the body of our myriapod model, for which we ignore the drag effects in water of the (tucked) legs.

Our experiments show that in order to enable the elasticity simulation to generate sufficient muscle force without serious oscillation and instability, a large Young's modulus and damping coefficient must be used. For our simulation, the Young's modulus is 4000, the Poisson's ratio is 0.4, and damping coefficient is 50. Note that a biological muscle's material properties are changing during actuation and relaxation (Brozovich et al., 1988), which would require additional modeling and simulation fidelity.

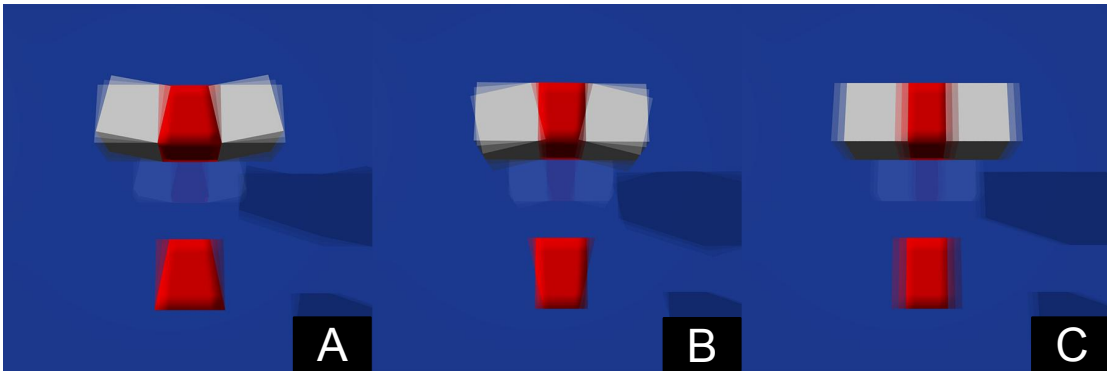


Figure 5.2: By morphing the rest shape of the deformable segments, we can actuate the three segments for different kinds of movements. (A), (B): Periodic flapping of the rigid segments (white). (C): Periodic shrinking and elongation of the body, like a worm.

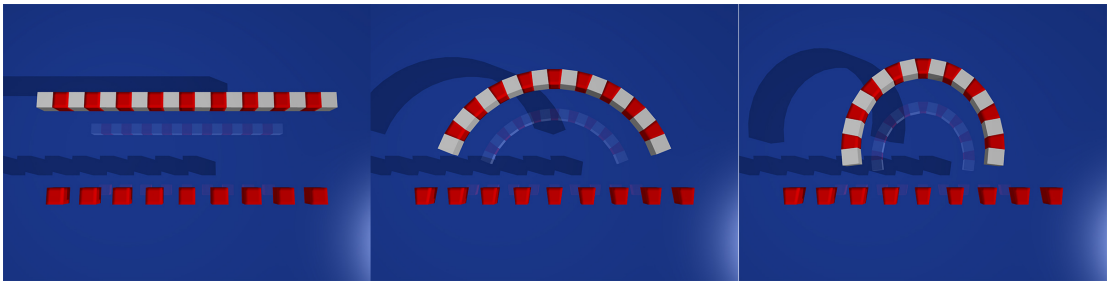


Figure 5.3: Curving the body to a 'C' shape. By simultaneously shrinking l_{left} of the deformable segments' rest shape.

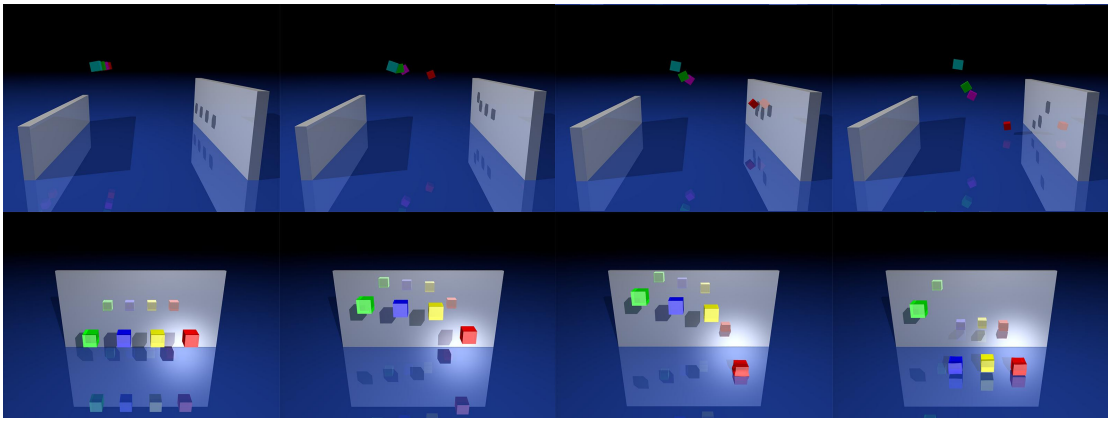


Figure 5.4: Top: Simulation of four rigid cubes of different density under water; Bottom: Simulation of four deformable cubes of different density under water.

Note that our dynamic approach does not merely morph the geometry of the deformable segments kinematically. As shown later, with appropriate external forces from water, our F-PS controller can generate the natural lateral undulation of the tail.

5.2 Simple Aquatic Simulation

To compute the hydrodynamic forces on the segmented body, we employ a force model similar to those found in (Tu and Terzopoulos, 1994; Lentine et al., 2011). The force on an infinitesimal surface element of area ds is computed as follows:

$$\mathbf{f}(s)\,ds = -\rho_w \max[0, \mathbf{n}\cdot\mathbf{v}](\mathbf{n}\cdot\mathbf{v})\mathbf{n}\,ds, \tag{5.2}$$

where ρ_w is the density of water, $\mathbf{n}(s)$ is the outward normal of the surface element, and $\mathbf{v}(s)$ is its velocity relative to the water. The total force on any object is the integral over all water-exposed surfaces: $\mathbf{f}_w = \int_S \mathbf{f}(s)\,ds$.

For the rigid segments, we add a high-resolution grid mesh over their surfaces and compute the water force on each quad patch exposed to water according to (5.2). Then we sum the forces and torques on those patches and feed them as external forces to the rigid-body simulator. For the deformable segments, we compute the force on each surface triangle exposed to water and distribute them to the three nodes of the triangle. Fig. 5.4 shows the result of our simulated rigid and deformable segments under water. For the segmented model, we sum water forces over the surfaces immersed in water. Buoyancy forces may be added by simply modifying the gravitational force $\mathbf{g}$. Depending on the density of the body ρ_b, the new gravity constant is $\frac{\rho_b-\rho_w}{\rho_w}\mathbf{g}$. We set our segmented body material density to be the same as water, such that the buoyancy counteracts gravity.

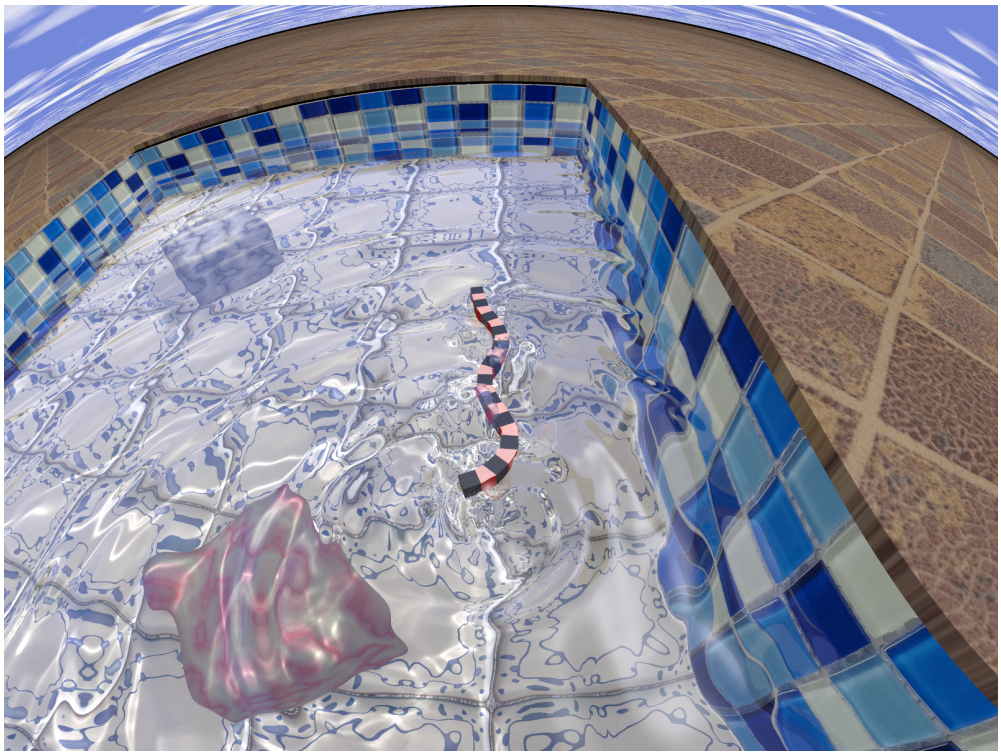


Figure 5.5: Anguilliform swimmer in a pool of water.

5.3 One-way Coupling with a Shallow-Water Simulation

To enhance the realism of our swimming animations, we couple the body of the anguilliform swimmer with a real-time simulation of shallow water in a one-way manner to generate vortices and wakes (Fig. 5.5). The main advantage of shallow-water simulation is real time performance compared to time consuming 3D fluid simulations via the full Navier-Stokes equations. Since it represents the water surface as a height field, no expensive tracking algorithms (Yu and Turk, 2013) are required to generate the water surface for the purposes of rendering. Our program can update a 200 by 200 resolution mesh in real time on a single 3.2 GHz CPU core. The central region where the creature swims requires proper coupling and poses difficulties for parallelization. However, for the outer regions, with a better numerical grid for far field (Zhu et al., 2013) or parallelization on GPU (Brodtkorb et al., 2012), the current mesh can be extended greatly while

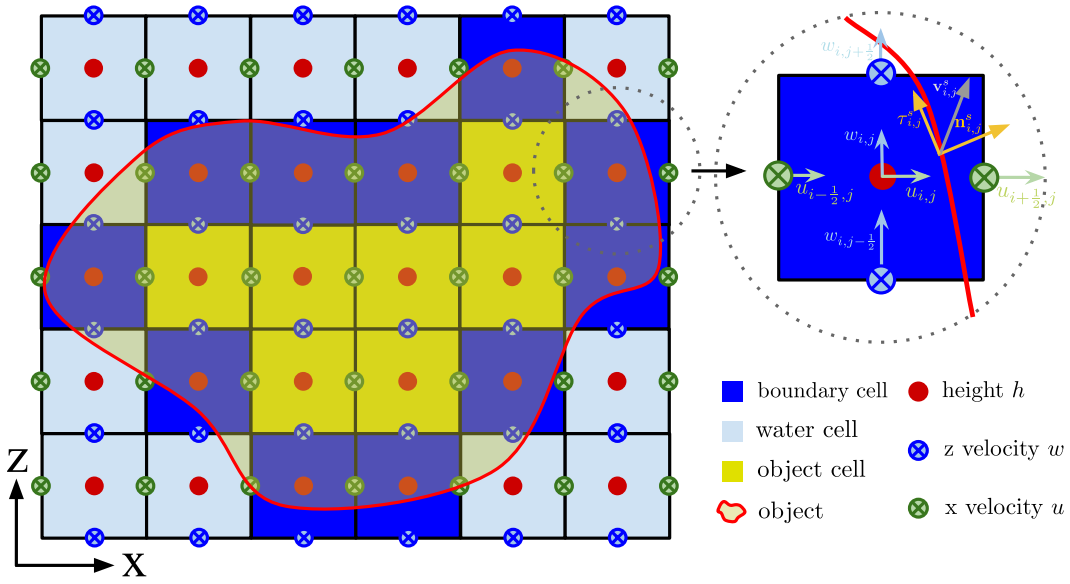


Figure 5.6: Illustration of a moving object in water on MAC grid

maintaining the real time performance.

The computational efficiency comes from the geometric simplification of 3D fluid to a 2.5D height map, which excludes interesting effects such as splashes and breaking waves.[1] Appendix B provides the derivation of the shallow-water equations from the incompressible Navier-Stokes equations and illustrates our MAC-grid-based numerical method for solving the shallow-water equations with different static boundary conditions.

Next, we will explain the coupling with the moving virtual animal. There are all together five steps:

1. Identify the cells that the anguilliform body occupies (Fig. 5.6 and Fig. 5.7), especially the boundary cells (blue). We test the center of each cell against the body shape contour to determine the type of cell. We accelerate the cell type determination by checking only cells in the bounding box that contains two rigid segments.

[1] There are computationally cost-effective ways to restore them with particle systems (O'Brien and Hodgins, 1995) (for splashes) and wave patches (Thurey et al., 2007) (for breaking waves).

2. Calculate the normal $\mathbf{n}^s_{i,j}$ and velocity $\mathbf{v}^s_{i,j}$ of the body at each boundary cell (i, j).

3. Update the height of internal cells (yellow) to be the height of the bottom of the body. In our simulation, we set the body's center height to be the same as the water level, so the height of yellow cells are set to be lower than the water level by half the body's size in the y dimension.

4. Update the height of the boundary cells (blue) to be the average height of surrounding water cells' heights.

5. Update (u, w) of the water on the boundary cells such that

$$(u, w) \cdot \mathbf{n}^s = \mathbf{v}^s \cdot \mathbf{n}^s. \tag{5.3}$$

Numerically, this is updated as follows:

$$\begin{aligned}(u, w)^{n+1}_{i,j} &= (u, w)^n_{i,j} - \big((u, w)^n_{i,j} \cdot \mathbf{n}^s_{i,j}\big)\mathbf{n}^s_{i,j} + \big(\mathbf{v}^s_{i,j} \cdot \mathbf{n}^s_{i,j}\big)\mathbf{n}^s_{i,j} \\ &= (u, w)^n_{i,j} + \big(\big(\mathbf{v}^s_{i,j} - (u, w)^n_{i,j}\big) \cdot \mathbf{n}^s_{i,j}\big)\mathbf{n}^s_{i,j}.\end{aligned} \tag{5.4}$$

Fig. 5.7 shows our coupling results, two rows of lateral vortex rings can be observed from the velocity field, which is the documented characteristic wake pattern for anguilliform swimming (Kern et al., 2008). Fig. 5.5 shows our rendered result of the anguilliform model swimming on the surface of a pool. Appendix E describes our method for realistically rendering water.

5.4 Swimming Locomotion Controller

Researchers have done extraordinary work on modeling and simulating fish swimming. In his pioneering work, Ekeberg (1993) built a neural network to control a 2D mass-spring-damper based lamprey mechanical body in a simulated water

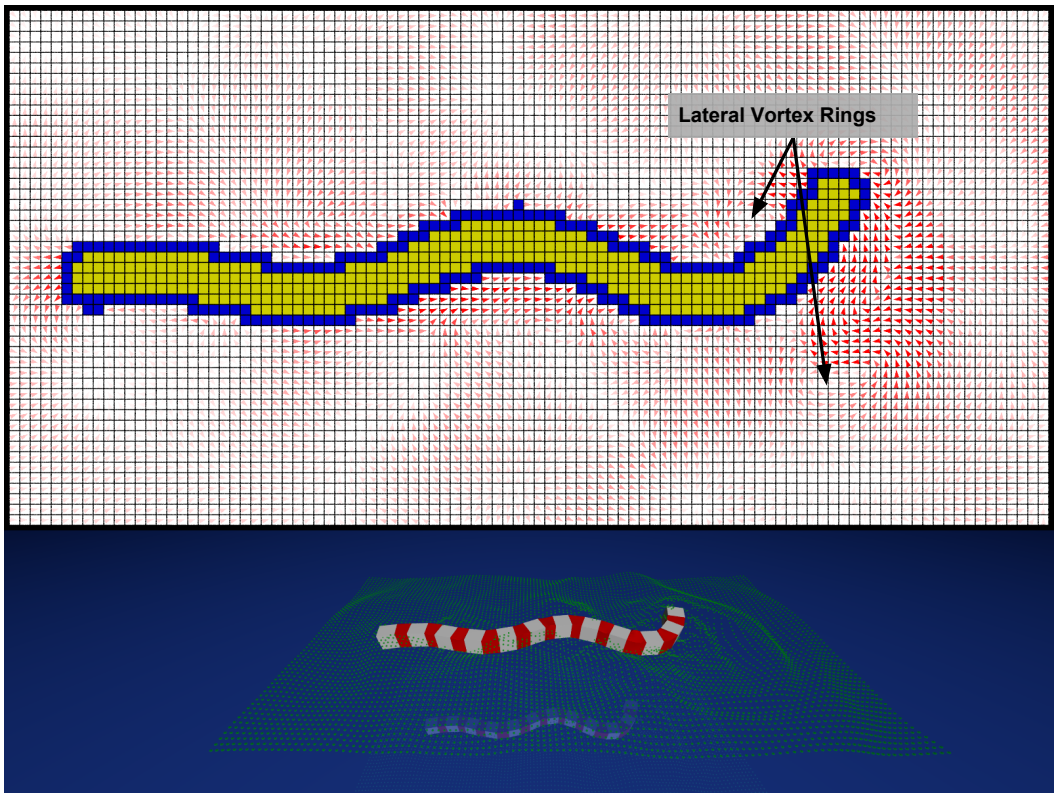


Figure 5.7: Top: The red arrows show the velocity field of water around the anguilliform swimmer. The arrow in the cells show direction and the intensity of color represent scale. The blue and yellow cells are occupied by the body. Bottom: Swimming in a moving box of shallow water. We use absorbing boundary conditions for the water simulation. More detail can be found in Appendix B.

environment. Ekeberg's neural network model is derived from neurological studies of lampreys. From experiments, Grillner (1974) recorded rhythmic motoneuronal activity alternating between the two sides of the spinal cord. Also, the alternations occur with a phase shift along the body proportional to the distance between the points. It is also pointed out that the phase shift between any pair of points along the cord remains constant even when the speed of swimming changes. Ekeberg's neural network can produce a rhythmic segmental pattern at various frequencies by changing the tonic brainstem input, as well as coordinated waves along the spinal cord with constant phase shift. His simulated model can achieve propulsion at various speeds and by making his tonic stimulation level asymmetric, turning

can be performed.

Although Ekeberg did not adopt optimization methods for his neural network, nor did he build an A-Life system for his simulated creature, his published work stands among important independent efforts by Tu and Terzopoulos (1994) on artificial fishes and by Ijspeert et al. (2007) on amphibious salamanders. In Ijspeert's work, a CPG is built in two stages to control the salamander swimming. The first stage is a segmental oscillator that outputs periodic signals; the second stage couples the segmental oscillators, to eventually deliver the constant phase shift. However, the optimization of the CPG networks is purely based on fitness functions that try to train for desired output signal shapes, without forward simulation to evaluate the true locomotive efficiency of the controller. In (Grzeszczuk and Terzopoulos, 1995), optimal controllers are learned via forward simulation to reward locomotive efficiency, which is evaluated as consuming the least energy for the distance traveled. However, the optimization find only one global optimum, while in reality, such creatures could locomote at various speeds all with optimal energy efficiency. Recent work by Kern et al. (2008) used accurate calculation of work exerted against water to optimize a anguilliform locomotion controller. However, the body shape of the model described in the paper is completely controlled by parametric curves, with no internal actuations, and is unaffected by water forces.

Our goal is to build a swimming controller based on prior efforts, which not only can generate realistic anguilliform swimming patterns, but can also be trained to compute optimally energy-efficient parameters to achieve any swimming speed. We would like correspondence of our simulation and training results with experimental observations. Lucid understanding can often result from a reduction of the complexity of a system to the point where only the most important components remain and are analyzed. We took this strategy to shrink our control space to include only two parameters, a frequency (f) and a segmental phase shift

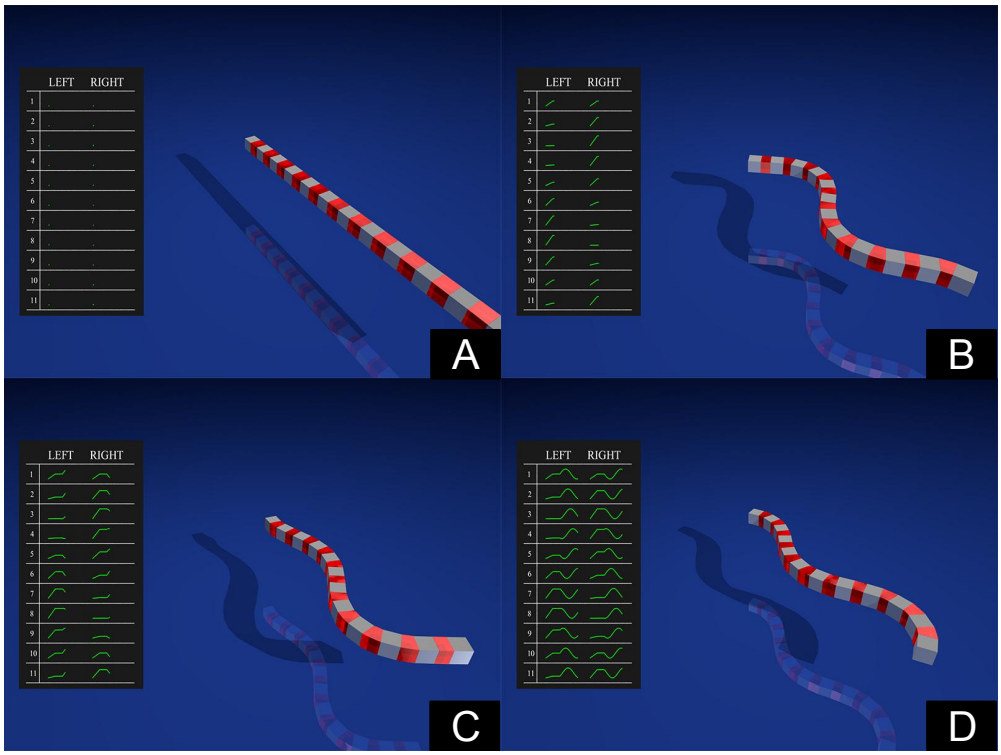


Figure 5.8: The anguilliform model morphs to a serpentine shape and starts swimming.

(ps). We name it the F-PS control model with F standing for frequency and PS standing for phase shift. Our controller outputs left s_l and right s_r sinusoidal wave signals with different frequency and phase shifts, and with amplitude in the range of $[0, 1]$, which represents the muscle activation level Signals s_l and s_r alternate with each other such that $s_l^2 + s_r^2 = 1$ and are linearly mapped to l_l and l_r to morph the deformable segments' rest shape, resulting in muscle contraction and relaxation. The mapping is as follows:

$$\begin{aligned} l_l &= l_{min} + s_l(l_{max} - l_{min}) \\ l_r &= l_{min} + s_r(l_{max} - l_{min}) \end{aligned} \tag{5.5}$$

The parameters l_{max} and l_{min} define the range in which l_l and l_r fluctuates. In our simulation, $l_{max} = 1.0$ and $l_{min} = 0.5$.

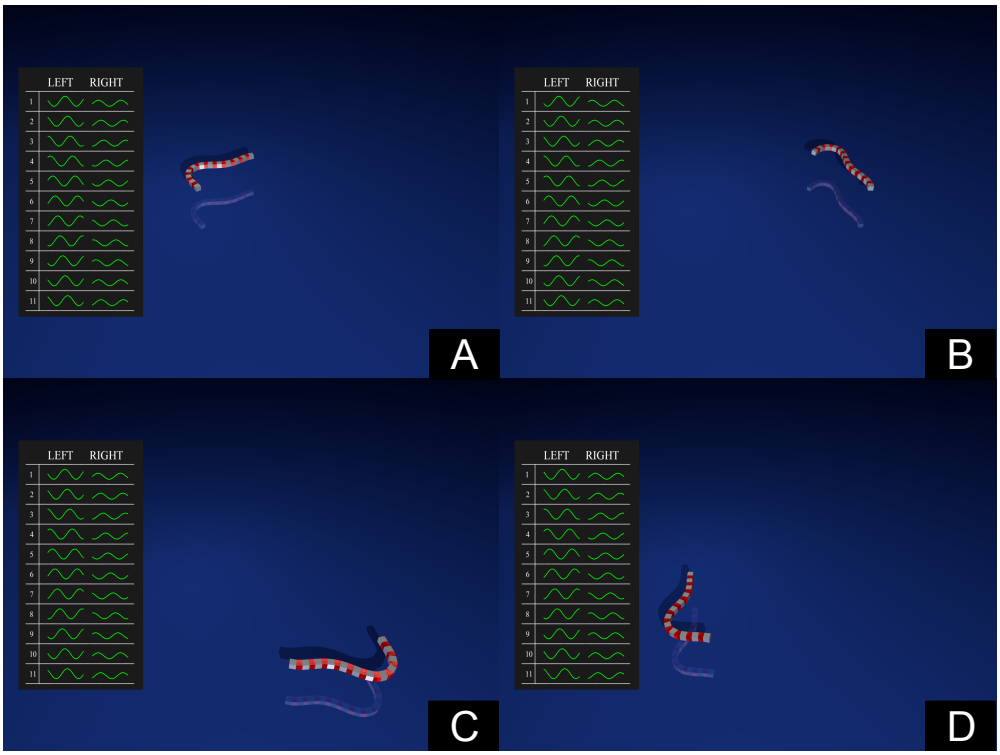


Figure 5.9: The anguilliform swimmer makes a clockwise circular turn.

5.4.1 The F-PS Controller

The F-PS controller preserves the key results from previous works, which all employ antisymmetric left/right motonueron excitations and constant phase shifts along the body. Fig. 5.8 shows our F-PS controller enabling the anguilliform swimmer to swim in a straight line. The controller will first flex the body to a serpentine shape for swimming. this initial stage is shown in the left control signal panel of (A) and (B) in Fig. 5.8. Note that the amplitude of the undulations increase from head to tail even though the amplitudes of muscle actuations are identical for all the deformable segments. This increase in undulation toward the tail is in agreement with natural anguilliform swimming (Gray, 1933) and it is purely an emergent property of the F-PS controller and physical simulation. Another observation is that the wave along the body propagates faster than the swimming speed, which is also a well-known property of most undulatory

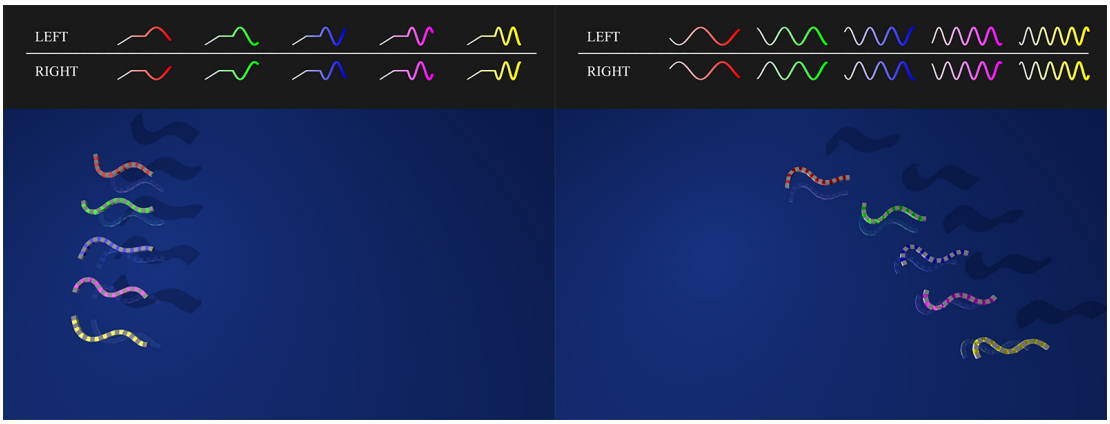


Figure 5.10: Comparison of linear speed among anguilliform swimmers with different muscle actuations. Obviously, the larger actuation frequency, the faster the swimming.

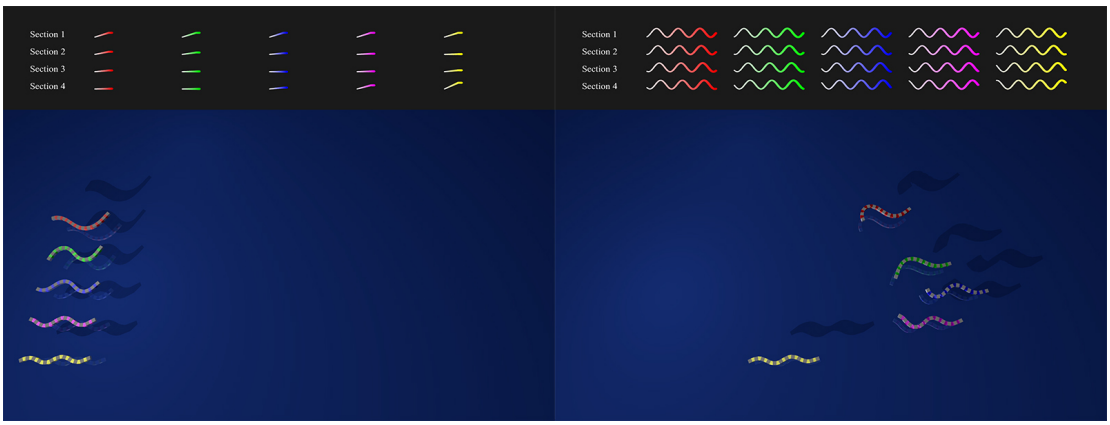


Figure 5.11: Comparison of linear speed among creatures of different segmental phase-shift. The blue anguilliform swimmer is fastest, indicating that there is an optimal phase shift.

modes of swimming (Lighthill, 1970). Turning is achieved, as usual, by making the amplitude of the actuation levels asymmetric. As shown in Fig. 5.9, our anguilliform swimmer will turn toward the side that has a weaker actuation level. A large curvature can be observed that bends toward the sides of turning and propagates backwards along the body, this phenomenon is consistent with observations of real lamprey turning (Gray, 1933).

5.4.2 Optimal Control at All Speeds

The drastic reduction in the dimensionality of the control space enables a careful inspection of the relationship between the controller parameters, f and ps, and locomotion performance through forward simulation. Fig. 5.10 illustrates a race experiment among five anguilliform swimmers of the same ps and increasing f. Not surprisingly, a higher undulation frequency yields faster locomotion. Fig. 5.11 illustrates a race experiment among five anguilliform swimmers with the same f and different ps. We discovered an optimal ps that results in the fastest locomotion. Appendix C presents the details.

5.5 Autonomous Anguilliform Creature

To create an autonomous anguilliform swimmer in its aquatic environment (Fig. 5.12), we use a vision-based sensory mechanism to guide its adaptive behavior. The green fan represents the field of view within which the swimmer can see food (green spheres). Whenever food falls inside the field of view (red sphere), the swimmer will turn toward the food. For obstacle avoidance, the swimmer also employs the Braitenberg vehicle steering mechanism. The yellow fan represents its alert field of view. When it reaches the wall, it will make a turn towards the side of the yellow fan that is not yet hitting the wall. If both sides of the yellow fan hit the wall, the swimmer will repeatedly turn toward a random selected side until the turn is completed. Our swimmer adjusts its speed depending on how far it is from the food and obstacle. For instance, it will accelerate when it initially spots food and slow down as it approaches the food; it will also slow down when a large turn is required in a short time. Our swimming locomotion controller can maintain optimal energy efficiency during the speed adaptations.

Fig. 5.13 shows snapshots from animations of multiple anguilliform swimmers in a pool on a rainy day.

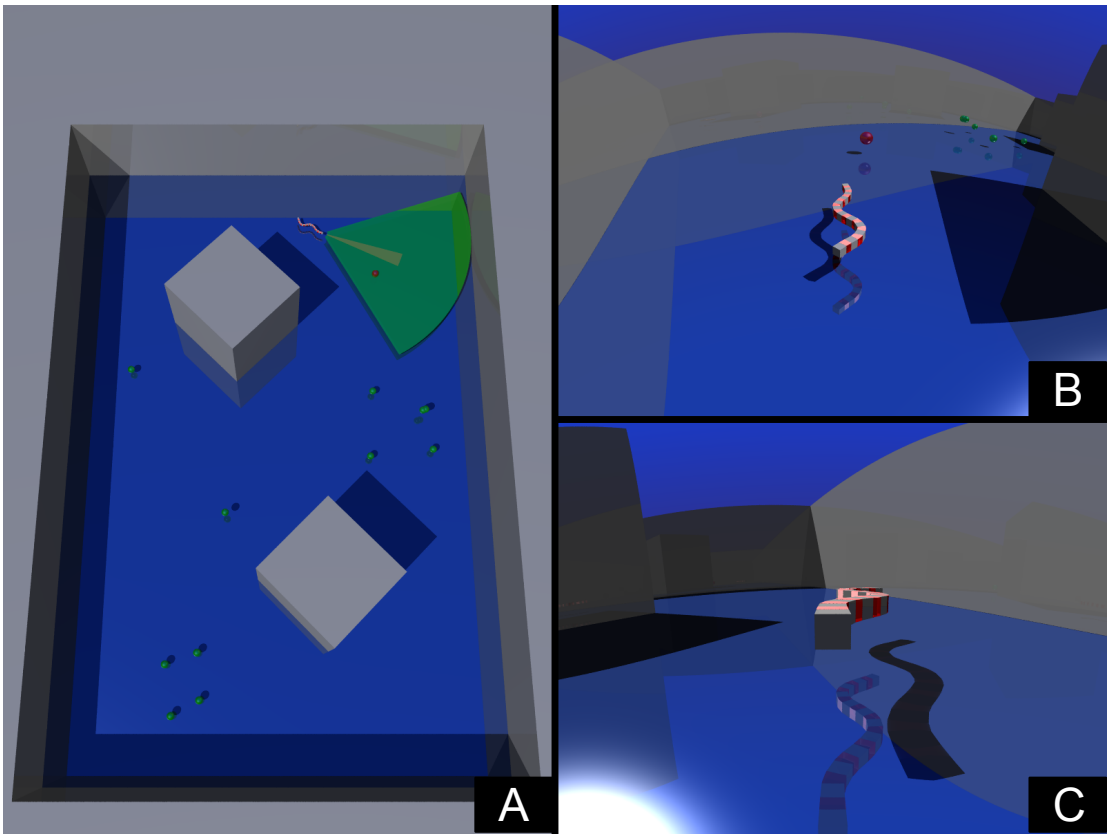


Figure 5.12: An autonomous anguilliform creature swimming hydrodynamically with its optimized F-PS controller, foraging while avoiding obstacles within its field of view.

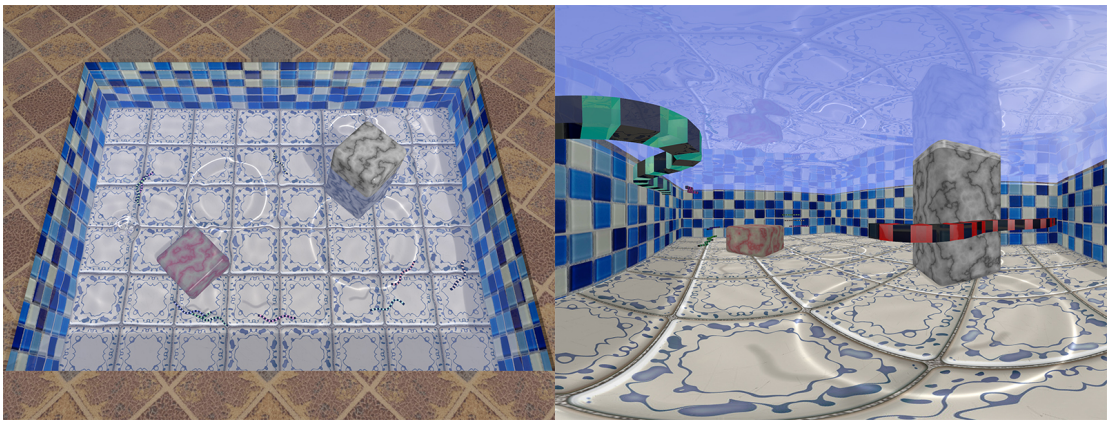


Figure 5.13: Multiple anguilliform swimmers simulated simultaneously in a pool of water, with rain drops generating waves and caustic light patterns on the bottom of the pool.

CHAPTER 6

Simulations and Results

We implemented the biomechanical body structure and leg controller and parallelized the code with OpenMP. Our Intel Core i7-3930k (6 core, 3.2 GHz) machine can update an 18 segment centipede model with $3 \times 3 \times 3$ deformable mesh resolution at a rate of 1.7×10^4 time-steps per second. Typically, a time-step of $1/3000s$ is used, resulting in approximately $6\times$ real-time speed. Most of the time is consumed by the semi-implicit FEM simulation of the deformable segments (fully implicit simulation should improve stability and speed). Different body parameters can be used to model different types of Myriapoda, as shown in Fig. 1.3. A detailed description of the simulation loop can be found in Appendix D. Next we will present the various animation results produced by our implemented animation system.

6.1 Irregular Terrain

The decentralized locomotion system works robustly in the presence of various types of terrains. In order for the simulated creature to locomote over it, the terrain model need to support three basic queries:

1. Determine if the tip P of a leg is touching the ground by doing an inside/outside test against the terrain geometry;

2. Obtain the height h_i of the center $\mathbf{c}_i$ of the rigid segment relative to the terrain surface.

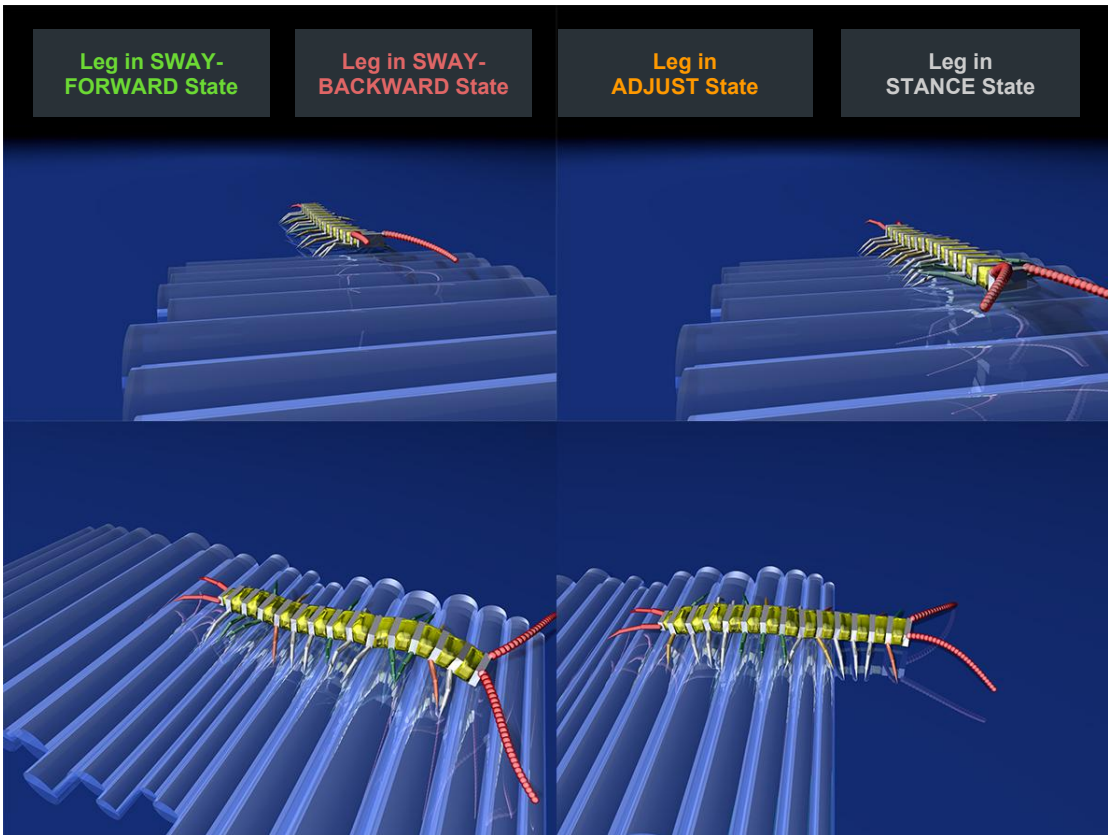


Figure 6.1: Centipede Walking over an irregular surface paved with random cylinders.

3. Obtain the normal vector of the terrain surface directly below $\mathbf{c}_i$.

The myriapod's locomotion controller depend on the first queried data to update its leg states and the later two to balance its rigid segments' height and orientation (Fig. 4.2).

6.1.1 Height-Field Terrain

We implemented a height-field-based terrain generator that can create arbitrary terrains from mixtures of Gaussian functions. All three queries are straightforward to perform for our height-field terrain. However, a height-field cannot represent a surface that inverts, such as a bulge on the ground that forms an "Ω" shape. In the next section, we will discuss our generalization to such terrains. Our myriapod

simulator can also introduce additional surface obstacles, such as cylinders and spheres for the creature to crawl over (e.g., Fig. 6.1). These cylinders and spheres are analytical geometric entities, so the aforementioned queries can be determined accurately.

6.1.2 Closed Surfaces

In addition to height-field-based terrain, we extended our simulator to support arbitrary closed triangulated meshes. OBJ files of such mesh geometries can be loaded into the simulator and serve as locomotion surfaces. Fig. 6.2 shows a centipede walking over the mesh of a crystal human skull. The interface between our simulated creature and the loaded mesh remain unchanged—the three queries listed in the previous section. To perform the queries efficiently, a spatial hash table is built as an acceleration structure for our triangle mesh storage. First, we partition the bounding box of the object into a 3D grid of a certain resolution (e.g., $50 \times 50 \times 50$ for the cranium in Fig. 6.2). Then, for each triangle, based on its position, we locate the index, idx, idy, and idz, of the cell that contains it. We use the equation

$$hash\_key = idx \times grid\_res\_y \times grid\_res\_z + idy \times grid\_res\_z + idz \tag{6.1}$$

as the key to store the triangle into our hash table. Each table entry is a list of triangles. To perform the query on whether a point is inside or outside the mesh, we cast a ray along the positive z axis and count the number of intersections n_i with the mesh. If n_i is even, this indicates that the point is inside the closed mesh. An intersection is determined by checking against all triangles contained in the grid cells along the ray; usually the internal cells are all empty so only n_i number of cells are needed. The second and third query both need to obtain the nearest triangle to $\mathbf{c}_i$. For that, we simply iterate over the triangles in the same cell (and

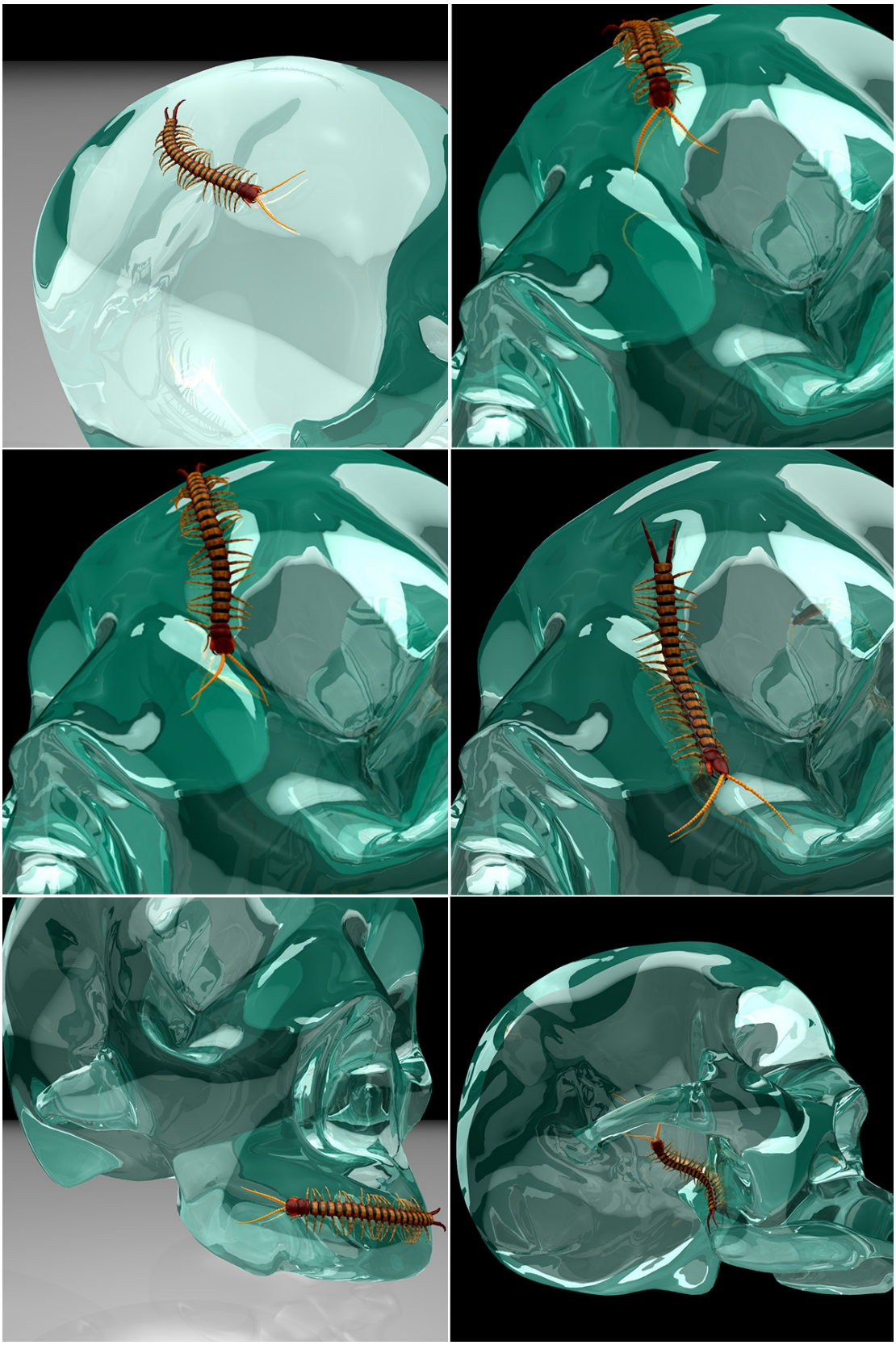


Figure 6.2: Centipede walking over a closed mesh of a human skull.

neighboring cells in extreme cases) and calculate the distance of point $\mathbf{c}_i$ to those triangles. After obtaining the closest triangle, we can easily obtain the height to the ground and its surface normal.

6.2 Walking in Shallow Water

Section 5.3 described how we couple the myriapod model with a shallow-water simulation (Fig. 5.7). Fig. 6.3 shows stills from an animation in which a centipede walks around obstacles toward food sources in shallow water. The water ripples realistically under the legs of the myriapod, leaving a wake along the path. One-way coupling of a myriapod' legs with sand (Narain et al., 2010) or snow (Stomakhin et al., 2013) would also generate realistic traces on such granular surfaces. This would be an interesting topic for future work.

6.3 An Amphibious Centipede

Chapter 4 and 5 discussed the controllers we developed to allow our centipede model for terrain locomotion and aquatic swimming. To make our simulated myriapods amphibious, we straightforwardly combine the two systems by defining new segment state, leg state, and body modes, as follows:

In Section 4.1, we defined two states for the rigid segments, that is *supported* and *unsupported*. For swimming, we introduce another state, *submerged*. This state is activated when the center of a rigid segment enters water and deactivated when it exits water to ground. In Section 4.3, we presented the leg state machine that is comprised of six leg states. We incorporate a switcher that monitors and updates leg states according to a list of five rules. Here we introduce a new leg state named TUCKED, it defines the set of joint angles for the legs, such that they will tuck against the body when the myriapod is swimming in water. The

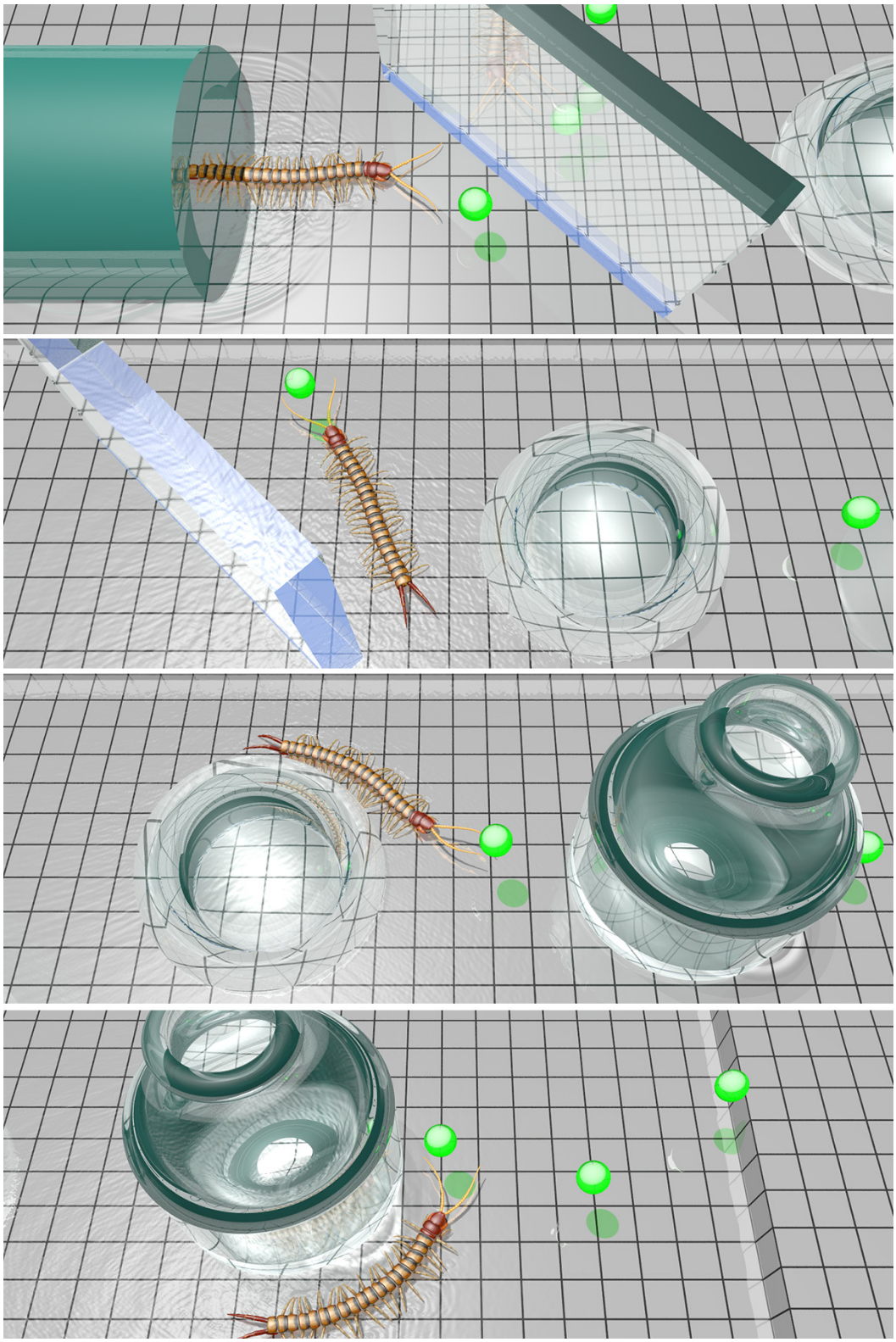


Figure 6.3: A centipede walking in shallow water, attracted by food (green spheres) and avoiding obstacles.

Enter Body Mode	When
IN-AIR	No legs ever enter STANCE state.
ON-GROUND	All segments' legs can all enter STANCE state.
IN-WATER	All segments are in *submerged* state.
AIR-TO-GROUND	The previous body mode is IN-AIR and some but not all segments' legs can enter STANCE state.
GROUND-TO-WATER	The previous body mode is ON-GROUND and some but not all segments are in *submerged* state.
WATER-TO-GROUND	The previous body mode is IN-WATER and some but not all segments are out of *submerged* state.

Figure 6.4: Amphibious body mode rules.

update rule for the new leg state is as follows:

- When the rigid segment enters *submerged* state, its legs will enter the TUCKED state;
- When the rigid segment exists *submerged* state, its legs will enter SF1 state.

To facilitate the transition of locomotion controllers, we define seven body modes that indicate the physical state of the myriapod, which are monitored by the brain in the head segment and updated accordingly by querying the states of the legs and rigid segment. Fig. 6.4 tabulates the rules for determining each mode. The body modes are used together with the brain modes, segment states, and leg states to adjust locomotion controllers in high level and guide autonomous behaviors.

Land to Water: Fig. 6.5 shows the process of a virtual centipede walking down the edge of a water pool and initiating swimming. First, lead by the head and with the following mechanism, each segment will enter the water and the legs will enter the TUCKED state. At the same time, the deformable segments will start to morph to give the body a serpentine shape in preparation for swimming. As the last segment enters the *submerged* state, the body mode switches from GROUND-TO-WATER to IN-WATER, and the swimming locomotion controller takes over.

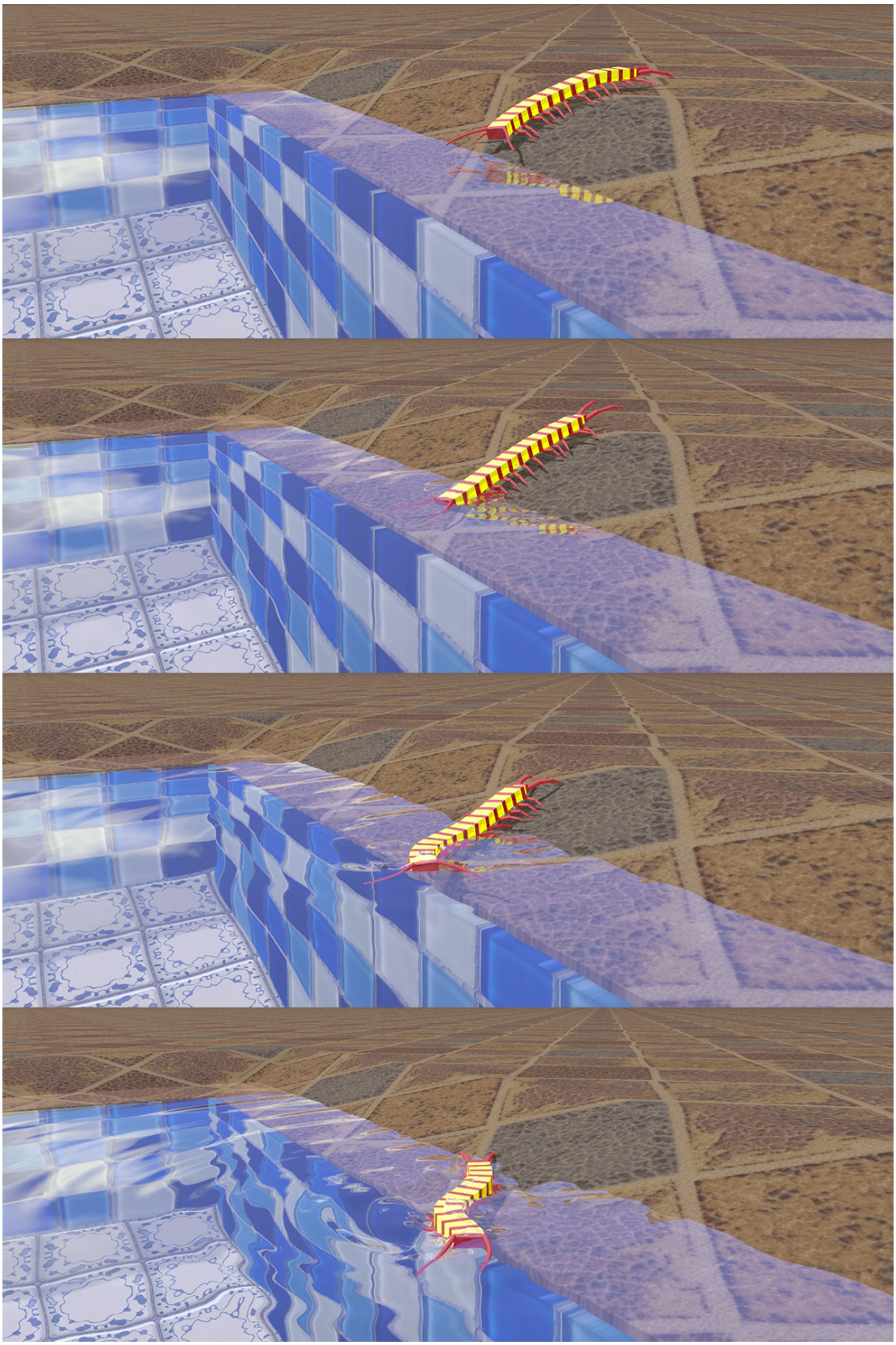


Figure 6.5: A centipede enters water from land.

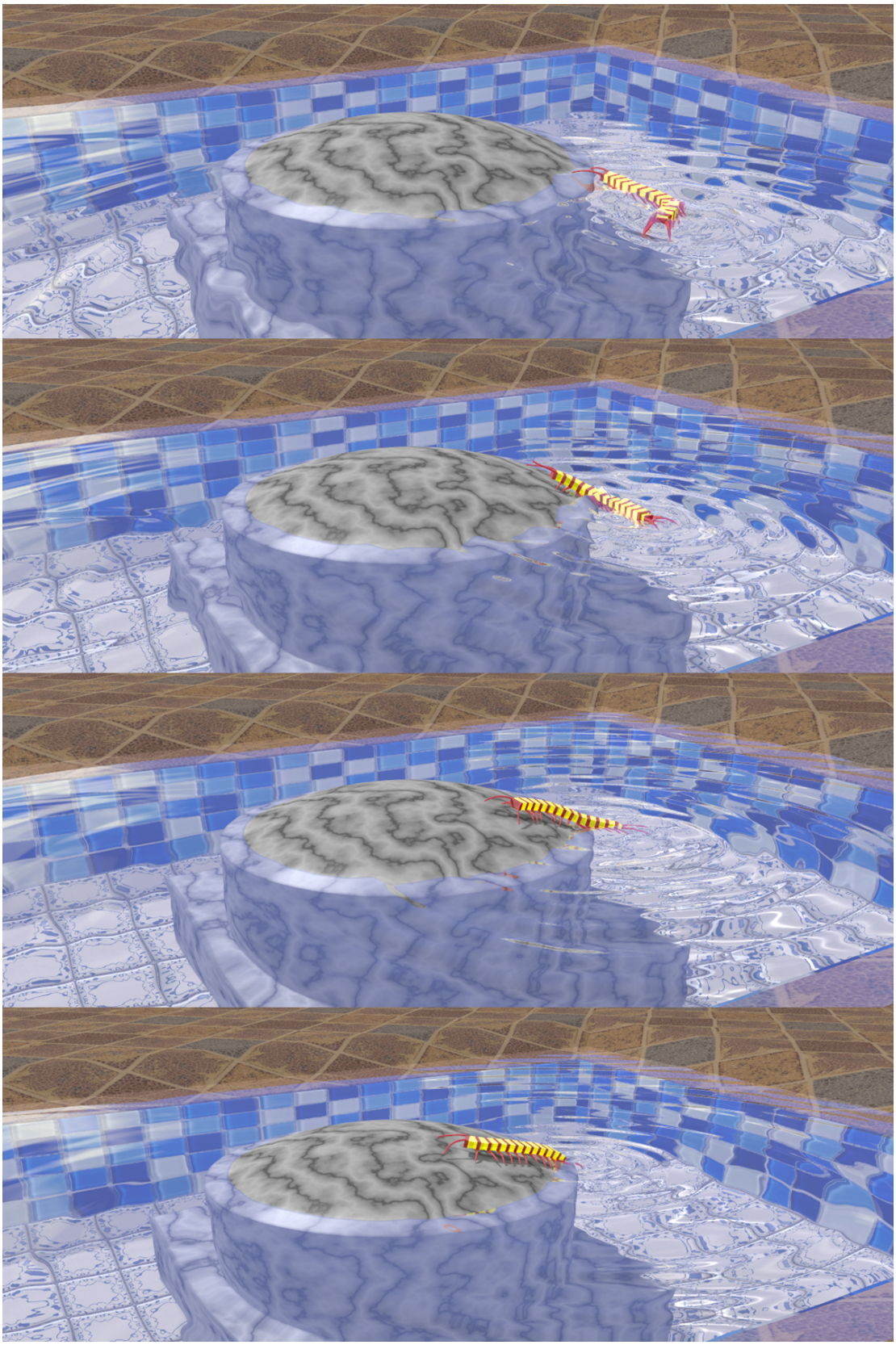


Figure 6.6: Centipede exits water to land.

The propulsion and turning of the centipede are then achieved through the F-PS controller, muscle actuation and hydrodynamics.

Water to Land: Fig. 6.6 shows the process of the virtual centipede exiting water and crawling to the top of the partially submerged pillar at the center of the pool. As the head segment exits water, the body mode switches to WATER-TO-GROUND and the hybrid locomotion controller takes over. Each rigid segment that exits water will inform its leg to enter the SF1 state and the rigid segment will be governed kinematically by its Targeter to follow and balance during STANCE states. At the same time, deformable segments that exit water will start to morph back to their initial shapes. When the tail segment exists water, the body mode is set back to ON-GROUND and the centipede will be controlled entirely by the decentralized locomotion controller.

Fig. 6.6 shows that our real-time simulation generates wakes in the water as the centipede swims.

CHAPTER 7

Conclusion

7.1 Summary

We have developed an A-Life framework for animating Myriapoda with lifelike ambulation and anguilliform swimming in real time. To deal with the unique body structure of Myriapoda relative to other arthropods, we devised a novel physical body structure that is composed of alternating rigid and deformable body segments. For legged locomotion, we devised a decentralized locomotion control system composed of identical local leg controllers. The locomotion controller demonstrates competence in various irregular terrains and over closed surface objects. To generate realistic swimming locomotion, we enabled the deformable body segments to be actively controlled and serve as muscles. We trained optimal F-PS controllers to sustain a range of swimming speeds. To enhance the realism of our real-time animations, we coupled our myriapod model with a shallow-water simulation that generates realistic wakes as the creatures locomote in water.

In addition to their motor control system, our Myriapoda are equipped with antennae for sensing food and obstacles, leading to interesting autonomous exploration behaviors in complex terrestrial and aquatic environments. Animators can take advantage of these high level behaviors to guide the locomotion path of our simulated creatures.

On the one hand, during terrestrial locomotion the rigid segments of our myriapod model with their attached legs serve as the active components that mo-

tivate the animal while the deformable body segments respond passively. On the other hand, once the animal starts swimming, the rigid segments become passive while the deformable segments begin to actuate as active muscles. In the latter mode, the simulated animal is amenable to the application of machine learning algorithms to obtain optimized locomotion controllers, whereas the former mode is more suitable for manually designed motor controllers, since the motor control of the rigid body parts has fewer degrees of freedom and is simpler than controlling deformable meshes.

Given its simplicity and physical realism, our use of efficient and highly robust elastic components has the potential to supplant traditional mass-spring-damper systems for simulating the deformable body structures of artificial animals (Miller, 1988; Tu and Terzopoulos, 1994; Ijspeert et al., 2007). There are various possible ways to actuate such structures.

7.2 Limitations and Future Work

Coiled Body Shape: In nature, millipedes can coil up their bodies when they feel threatened (Fig. 1.1 B). Centipedes can probe their head around and turn their body suddenly when they attack. Myriapods achieve those agile and irregular motions by actuating their segmental muscles in a highly collaborative manner. In Chapter 5, our biomechanical model successfully swam forward with emulated muscles. However, the actuation of the muscles are limited to only the left and right side of the body segments (Fig. 5.1). To coil the body up, the muscle actuation model must be extended to allow contraction and relaxation of a segment in the top and bottom sides. Such extension will also allow the virtual animal to steer vertically in the 3D aquatic environment. It will be interesting to combine the actuation of the muscle segments on both sides and train for complex swimming patterns under water.

Dynamic Lamé Parameters: In our simulation, we update the rest shapes of the deformable segments to generate internal forces. However, the material properties of the muscle remain unchanged. In real life, the Lamé parameter of a muscle will vary as it exerts force. For example, the stiffness will increase when a muscle bulges and decrease when it relaxes (Brozovich et al., 1988). Varying Lamé parameters in a dynamic manner could potentially lead to more accurate simulation results and stronger actuation forces. Also, the simulation time step could be adaptive with the Lamé parameters, and larger stiffness would usually require a smaller time step to guarantee numerical stability.

CPG for Swimming: The simplicity of the F-PS swimming controller allows us to obtain optimal swimming parameters for a range of speeds. However, a trained CPG that can perform optimal swimming for arbitrary speed is still the ideal choice. The inherent advantages of CPG controllers include robustness to external disturbances and smooth transition of the motion upon change of modulation input (e.g., speed) (Si et al., 2014; Ijspeert, 2008). Traditionally, CPGs are trained with an objective function that targets the global optimum. As a next step, we would like to optimize it for all the speeds that it generates.

More Adaptive Behaviors: We have developed only simple myriapoda agents. It would be a natural next step to enable communication between our currently independent creatures. With proper local communication protocols, animations of groups of millipedes could be procedurally generated. Predator/prey behavior would be another interesting behavior to model. A fascinating objective would be to animate a predator centipede and prey millipedes.

Other Body Shapes: The simulation framework that we developed for animating Myriapoda can be used as a substrate to build other artificial animals of arbitrary body shape. With proper training algorithms such as genetic algorithm

(Sims, 1994) and simulated annealing (Grzeszczuk and Terzopoulos, 1995), locomotion controllers can be obtained for those body shapes situated in the virtual environment. Furthermore, the morphology of the animal can also be evolved in order to obtain the most hydrodynamic body shapes.

APPENDIX A

Deformable Finite Element Modeling and Simulation

This appendix details our simulation of the deformable segments in our myriapod model using the finite element method (FEM).

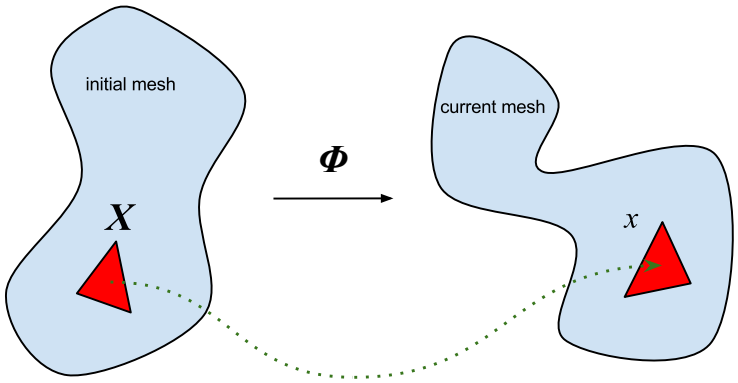


Figure A.1: Mapping of deformable object from initial shape to current shape. The red triangle shows a discretized triangular element.

Let $\mathbf{X}$ be the material coordinates of a deformable body, and $\mathbf{x}(t) = \phi(\mathbf{X}, t) = \mathbf{F}(t)\mathbf{X} + \mathbf{b}(t)$ be the world coordinates (Fig. A.1). We discretize the material space with a uniform tetrahedral mesh. In each tetrahedral element, the deformation gradient $\mathbf{F}$ can be computed as

$$\mathbf{F}_e = \mathbf{D}_s \mathbf{D}_m^{-1}, \tag{A.1}$$

where

$$\mathbf{D}_m = (\mathbf{X}_4 - \mathbf{X}_1, \mathbf{X}_3 - \mathbf{X}_1, \mathbf{X}_2 - \mathbf{X}_1) \tag{A.2}$$

are the edge vectors of the undeformed tetrahedron and

$$\mathbf{D}_s = (\mathbf{x}_4 - \mathbf{x}_1, \mathbf{x}_3 - \mathbf{x}_1, \mathbf{x}_2 - \mathbf{x}_1) \tag{A.3}$$

are the edge vectors of the deformed tetrahedron.

For hyperelastic material, we use the fixed corotational energy density function

$$\Psi = \mu \|\mathbf{F} - \mathbf{R}\|_F^2 + \frac{\lambda}{2}(J-1)^2 \tag{A.4}$$

$$= \mu \sum_i (\sigma_i - 1)^2 + \frac{\lambda}{2}(J-1)^2, \tag{A.5}$$

where λ and μ are Lamé parameters,[1] $J = \det \mathbf{F}$ is the determinant of $\mathbf{F}$, $\mathbf{R}$ comes from the polar decomposition $\mathbf{F} = \mathbf{RS}$, and σ_i are the singular values of $\mathbf{F}$, which we compute using the fast SVD method proposed by (McAdams et al., 2011). The elastic forces on nodes are computed as

$$\mathbf{f}_i = \sum_e V_e^0 \frac{\partial \Psi(\mathbf{F}_e)}{\partial \mathbf{x}_i} = \sum_e V_e^0 \mathbf{P} \frac{\partial \mathbf{F}_e}{\partial \mathbf{x}_i}, \tag{A.6}$$

where V_e^0 is the undeformed volume of tetrahedral element e, and $\mathbf{P}$ is the first Piola-Kirchhoff stress, given by

$$\mathbf{P} = \frac{\partial \Psi(\mathbf{F})}{\partial \mathbf{F}} = 2\mu(\mathbf{F} - \mathbf{R}) + \lambda J(J-1)\mathbf{F}^{-T}. \tag{A.7}$$

[1] In terms of Young's modulus E and Poisson's ratio ν, Lamé's first parameter $\lambda = E\nu/(1+\nu)(1-2\nu)$ and second parameter $\mu = E/2(1+\nu)$.

The derivation of (A.7) is as follows:

$$\begin{aligned}
\delta\Psi &= \delta\left(\mu\left(\sum_i \sigma_i^2 - 2\sum_i \sigma_i + 3\right) + \frac{\lambda}{2}(J-1)^2\right) && \text{(A.8)}\\
&= \mu\big(\delta\big(\mathrm{tr}(\mathbf{F}^T\mathbf{F})\big) - 2\delta(\mathrm{tr}(\mathbf{S}))\big) + \lambda(J-1)\delta J && \text{(A.9)}\\
&= 2\mu(\mathbf{F}:\delta\mathbf{F} - \mathrm{tr}(\delta\mathbf{S})) + \lambda(J-1)J\mathbf{F}^{-T}:\delta\mathbf{F}, && \text{(A.10)}
\end{aligned}$$

where the operators 'tr' and ':' denote the trace ($\mathrm{tr}\,\mathbf{A} = \sum_i A_{ii}$) and double contraction ($\mathbf{A}:\mathbf{B} = \sum_{i,j} A_{ij}B_{ij}$), respectively, and where

$$\delta\mathbf{F} = \delta\mathbf{R}\mathbf{S} + \mathbf{R}\delta\mathbf{S} \tag{A.11}$$

and

$$\delta\mathbf{S} = \mathbf{R}^T\delta\mathbf{F} - \mathbf{R}^T\delta\mathbf{R}\mathbf{S}. \tag{A.12}$$

Now,

$$\begin{aligned}
\mathrm{tr}(\delta\mathbf{S}) &= \mathrm{tr}(\mathbf{R}^T\delta\mathbf{F}) - \mathrm{tr}(\mathbf{R}^T\delta\mathbf{R}\mathbf{S}) && \text{(A.13)}\\
&= \mathbf{R}:\delta\mathbf{F} - \boxed{(\mathbf{R}^T\delta\mathbf{R}):\mathbf{S}} && \text{(A.14)}\\
&= \mathbf{R}:\delta\mathbf{F}. && \text{(A.15)}
\end{aligned}$$

For the boxed term, since $\mathbf{R}^T\mathbf{R} = \mathbf{I}$, we have

$$\delta(\mathbf{R}^T)\mathbf{R} + \mathbf{R}^T\delta\mathbf{R} = \big(\mathbf{R}^T\delta\mathbf{R}\big)^T + \mathbf{R}^T\delta\mathbf{R} = \mathbf{0}. \tag{A.16}$$

Obviously, $\mathbf{R}^T\delta\mathbf{R}$ is skew-symmetric, with $\mathbf{S}$ being symmetric, thus the boxed term is zero. Thus,

$$\delta\Psi = 2\mu\mathbf{F}:\delta\mathbf{F} - 2\mu\mathbf{R}:\delta\mathbf{F} + \lambda(J-1)J\mathbf{F}^{-T}:\delta\mathbf{F} \tag{A.17}$$

and

$$\frac{\partial \Psi}{\partial \mathbf{F}} : \delta\mathbf{F} = \left(2\mu\mathbf{F} - 2\mu\mathbf{R} + \lambda(J-1)J\mathbf{F}^{-T}\right) : \delta\mathbf{F}. \tag{A.18}$$

Since $\delta\mathbf{F}$ is arbitrary variation, (A.7) is obtained.

According to (A.6), the explicit force formula for each element can be written as

$$[\mathbf{f}_1^e, \mathbf{f}_2^e, \mathbf{f}_3^e] = V_e^0 \frac{\partial \Psi(\mathbf{F}_e)}{\partial [\mathbf{x}_1^e, \mathbf{x}_2^e, \mathbf{x}_3^e]} = V_e^0 \mathbf{P}\mathbf{D}_m^{-T}, \tag{A.19}$$

$$\mathbf{f}_0^e = \frac{\partial \Psi(\mathbf{F}_e)}{\partial \mathbf{x}_0^e} = -\mathbf{f}_1^e - \mathbf{f}_2^e - \mathbf{f}_3^e. \tag{A.20}$$

The calculation of elastic forces on each mesh node must take into account the contributions from each tetrahedral element; i.e.,

$$\mathbf{f}_i^E = \sum_{e, i \in e} \mathbf{f}_{i_e}^e, \tag{A.21}$$

where i_e is the local index of node i in element e. The mass of each node is calculated by averaging the mass over neighboring elements:

$$m_i = \frac{1}{4} \sum_{e, i \in e}^{e} \rho V_e^0, \tag{A.22}$$

where ρ is the density of the soft material.
We also introduce the damping force as

$$\mathbf{f}_i^D = -\gamma \sum_{e, i \in e}^{e} V_e^0 (\mathbf{v}_i - \frac{1}{4} \sum_{k=1}^{4} \mathbf{v}_{e_k}), \tag{A.23}$$

where e_k is the index of node k of element e, and γ is the damping coefficient.

A semi-implicit time integration scheme is applied at each time step:

$$(\mathbf{I} + \Delta t \mathbf{M}^{-1}\mathbf{D})\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \mathbf{M}^{-1}(\mathbf{f}^E + \mathbf{g} + \mathbf{f}), \tag{A.24}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1}, \tag{A.25}$$

where $\mathbf{v}$ is the nodal velocity vector, $\mathbf{x}$ is the nodal position vector, $\mathbf{M}$ is the diagonal mass matrix assembled from (A.22), $\mathbf{D}$ is the damping matrix assembled from (A.23), $\mathbf{f}^E$ is the internal, elastic force vector, $\mathbf{g}$ denotes gravity, and $\mathbf{f}$ are externally applied forces. In our simulations, we use a time step of $\Delta t = 1/3000$ sec.

For the deformable segments in our myriapod body model, we set the density of the hyperelastic material to $\rho = 1.0$, its Young's modulus $E = 4000$, its Poisson's ratio $\nu = 0.4$, and the damping coefficient $\gamma = 50$.

APPENDIX B

Water Simulation

In this Appendix, we first derive the Shallow Water Equations (SWE) from the *Navier-Stokes* equations. Then, we present the MAC-grid-based numerical scheme for solving the SWE.

The Navier-Stokes Equations

The incompressible Navier-Stokes equations are a set of partial differential equations (PDEs) that govern the mechanics of incompressible fluid flow. They can be drived from the laws of mass conservation and linear momentum conservation. For mass conservation,

$$\frac{d}{dt}\int_{\Omega}\rho\,dV = -\int_{\partial\Omega}\rho(\mathbf{v}\cdot\mathbf{n})\,dA, \tag{B.1}$$

where ρ is the density of fluid, Ω is an arbitrary element volume, $\mathbf{v} = (u, v, w)$ is fluid velocity and $\mathbf{n}$ is the outward unit normal vector on $\partial\Omega$, and the conservation of linear momentum (B.5). To derive the strong form, we assume continuity and apply Gauss' Theorem to the right hand side of (B.1), obtaining

$$\frac{d}{dt}\int_{\Omega}\rho\,dV = -\int_{\Omega}\nabla\cdot(\rho\mathbf{v})\,dV. \tag{B.2}$$

Since Ω is arbitrary,

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) = 0. \tag{B.3}$$

In most applications, the fluid density can be assumed constant, thus yielding the first Navier-Stokes equation from (B.3), which states that $\mathbf{v}$ is divergence free:

$$\nabla \cdot \mathbf{v} = 0. \tag{B.4}$$

The weak form of linear momentum conservation states that

$$\frac{d}{dt}\int_\Omega \rho\mathbf{v}\, dV + \int_{\partial\Omega} (\rho\mathbf{v})\mathbf{v}\cdot\mathbf{n}\, dA = \int_\Omega \rho\mathbf{g}\, dV + \int_{\partial\Omega} \mathbf{T}\mathbf{n}\, dA, \tag{B.5}$$

where $\mathbf{g}$ is gravity and $\mathbf{T}$ is the Cauchy stress tensor. Applying Gauss' Theorem again gives

$$\frac{d}{dt}\int_\Omega \rho\mathbf{v}\, dV + \int_\Omega \nabla\cdot(\rho\mathbf{v}\mathbf{v})\, dV + \int_\Omega \rho\mathbf{g}\, dV + \int_\Omega \nabla\cdot\mathbf{T}\, dV. \tag{B.6}$$

Assuming smoothness of $\rho\mathbf{v}$ and since Ω is arbitrary, we obtain

$$\frac{\partial}{\partial t}(\rho\mathbf{v}) + \nabla\cdot(\rho\mathbf{v}\mathbf{v}) = \rho\mathbf{g} + \nabla\cdot\mathbf{T}. \tag{B.7}$$

For incompressible isotropic Newtonian fluids, $\mathbf{T} = -p\mathbf{I} + \mu\nabla\mathbf{v}$, where μ is the viscosity tensor. Inserting this into (B.7) yields:

$$\frac{\partial\mathbf{v}}{\partial t} + \nabla\cdot(\mathbf{v}\mathbf{v}) + \frac{1}{\rho}\nabla p = \rho\mathbf{g} + \frac{\mu}{\rho}\nabla\cdot\nabla\mathbf{v}. \tag{B.8}$$

Expanding the second term into $(\nabla\cdot\mathbf{v})\mathbf{v} + \mathbf{v}\cdot(\nabla\mathbf{v})$ and applying (B.4), we obtain the second Navier-Stokes equation:

$$\frac{\partial\mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v} + \frac{1}{\rho}\nabla p = \rho\mathbf{g} + \frac{\mu}{\rho}\nabla\cdot\nabla\mathbf{v}. \tag{B.9}$$

For water, viscosity $\mu = 0$, and (B.9) simplifies to

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} + \frac{1}{\rho} \nabla p = \rho \mathbf{g}. \tag{B.10}$$

Note that $\frac{D\mathbf{v}}{Dt} = \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}$ is the material derivative of the velocity field.

The Navier-Stokes equations are therefore written as follows:

$$\begin{aligned} \nabla \cdot \mathbf{v} &= 0; & \text{(B.11)} \\ \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} + \frac{1}{\rho} \nabla p &= \rho \mathbf{g}. & \text{(B.12)} \end{aligned}$$

The Shallow Water Equations

The Shallow Water Model assumes that the fluid depth is very small compared to the horizontal wave length. For shallow water, the vertical velocity is ignored and the Navier-Stokes equations are reduced to 2D, solving for the horizontal components of the water flow velocity field and a height field that represents the surface of the fluid. To obtain the SWE, the first assumption is hydrostatic pressure

$$p(x, z) = \rho g h(x, z), \tag{B.13}$$

where h is the height field of the fluid. Substituting (B.13) into (B.12), we obtain

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + w\frac{\partial u}{\partial z} = -g\frac{\partial h}{\partial x} \quad \text{and} \quad \frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + w\frac{\partial w}{\partial z} = -g\frac{\partial h}{\partial z}, \tag{B.14}$$

which governs the update of the horizontal velocity field (u, w). Equation (B.11) must also be satisfied:

$$\frac{\partial v}{\partial y} = -\frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}. \tag{B.15}$$

Because u and w do not depend on y, $\partial v/\partial y$ is constant vertically. With Dirichlet boundary condition $\mathbf{v} \cdot \mathbf{n} = 0$ at the static bottom surface,[1]

$$v(x, b, z) = u\frac{\partial b}{\partial x} + w\frac{\partial b}{\partial z}. \tag{B.16}$$

From (B.15) and (B.16), we can derive that

$$v(x, y, z) = u\frac{\partial b}{\partial x} + w\frac{\partial b}{\partial z} - \left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z}\right)(y - b). \tag{B.17}$$

Such a linear v satisfies both (B.4) and the Dirichlet boundary condition at the bottom.

To derive the update equation for the height field, note that $\phi(x, y, z) = y - h(x, z)$ is the implicit surface function for the free surface. We know that the free surface will be advected by the fluid velocity:

$$\frac{D\phi}{Dt} = 0 \rightarrow \frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x} + w\frac{\partial h}{\partial z} = v. \tag{B.18}$$

Substituting (B.17) into (B.18), we obtain

$$\frac{\partial h}{\partial t} + u\frac{\partial (h - b)}{\partial x} + w\frac{\partial (h - b)}{\partial z} = -(h - b)\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z}\right). \tag{B.19}$$

Since $\partial b/\partial t = 0$, we can solve for $d = h - b$ and reconstruct h from $b + d$. For a flat ground $b = 0$, and (B.19) can be further simplified to

$$\frac{\partial h}{\partial t} + \frac{\partial (uh)}{\partial x} + \frac{\partial (wh)}{\partial z} = 0. \tag{B.20}$$

The physical interpretation of (B.20) is the conservation of the total fluid volume.

[1] The bottom terrain surface can be represented as $b(x, z)$, and its normal is $\mathbf{n}(x, z) = (-\partial b/\partial x, 1, -\partial b/\partial z)$.

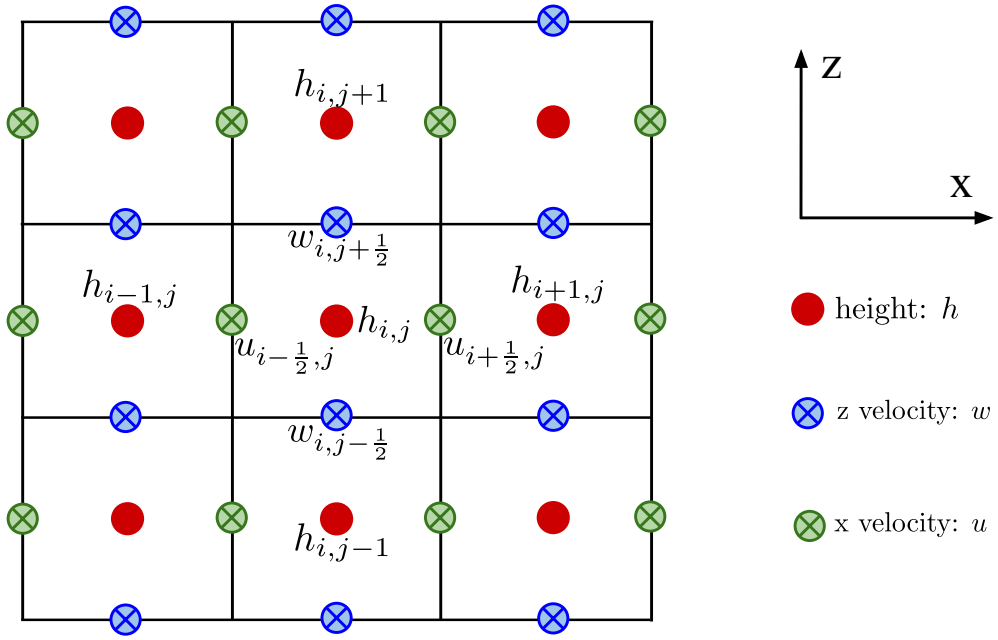


Figure B.1: The two-dimensional MAC grid for solving the SWE. The height data are stored at the centers of the cells, while the velocity data, u and w, are stored at the cell faces.

Solving the SWE on a MAC Grid

There are many possible methods for discretizing the SWE, each with its own pros and cons. We adopt the method described by Bridson (2008), which uses a two-dimensional staggered Marker-And-Cell (MAC) grid and a time-splitting approach. The MAC grid (Layton and van de Panne, 2002) is a staggered grid where the different variables are stored at different locations. For our shallow water simulation, we store the discrete height field $h_{i,j}$ at the centers of cells and split the two components of $\mathbf{v}$ on the faces of cells. The x axis component u is sampled at the centers of the vertical cell faces, such as $u_{i+\frac{1}{2},j}$ between cell (i,j) and cell $(i+1,j)$. The z axis component v is sampled at the centers of the horizontal cell faces, such as $w_{i,j+\frac{1}{2}}$ between cell (i,j) and cell $(i,j+1)$.

The advantage of the MAC grid over a conventional grid is that the central difference does not suffer from the non-trivial null-space problem. Also, compared

to either forward or backward differences on normal grids, the central difference on the staggered grid is unbiased and second-order accurate.

Rewriting (B.14) and (B.19) using the material derivative $\frac{D}{Dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + w\frac{\partial}{\partial z}$, and assume that the bottom $b(x, z)$ is constant, we have

$$\begin{aligned} \frac{Du}{Dt} &= -g\frac{\partial h}{\partial x}, \\ \frac{Dw}{Dt} &= -g\frac{\partial h}{\partial z}, \\ \frac{Dh}{Dt} &= -h\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z}\right). \end{aligned} \tag{B.21}$$

We use the time-splitting approach to handle advection first, generating intermediate quantities u^*, w^*, and h^*:

$$\begin{aligned} u^* &= \text{advect}(\mathbf{v}^n, \Delta t, u^n), \\ w^* &= \text{advect}(\mathbf{v}^n, \Delta t, w^n), \\ h^* &= \text{advect}(\mathbf{v}^n, \Delta t, h^n). \end{aligned} \tag{B.22}$$

Then, we update the velocities further with the pressure on the right hand side. On the MAC grid (Fig. B.1), this is written as

$$\begin{aligned} u^{n+1}_{i+\frac{1}{2},j} &= u^*_{i+\frac{1}{2},j} - g\frac{h^*_{i+1,j} - h^*_{i,j}}{\Delta x}\Delta t, \\ w^{n+1}_{i,j+\frac{1}{2}} &= w^*_{i,j+\frac{1}{2}} - g\frac{h^*_{i,j+1} - h^*_{i,j}}{\Delta z}\Delta t. \end{aligned} \tag{B.23}$$

Finally, we modify the height using the divergence of the new u and w fields such that total water volume is preserved:

$$h^{n+1}_{i,j} = h^*_{i,j}\left(1 - \left(\frac{u^{n+1}_{i+\frac{1}{2},j} - u^{n+1}_{i-\frac{1}{2},j}}{\Delta x} + \frac{w^{n+1}_{i,j+\frac{1}{2}} - u^{n+1}_{i,j-\frac{1}{2}}}{\Delta z}\right)\Delta t\right). \tag{B.24}$$

For advection, we use the unconditionally stable semi-Lagrangian method (Stam, 1999) together with bilinear interpolation over the MAC grid. The semi-Lagrangian

method solves the advection equation $Dq/Dt = 0$ by tracing imaginary particles that carry the physical quantity q (i.e., u, w, or h) along its trajectory in time:

$$q^*(x, z) = q^n(x_p, z_p), \tag{B.25}$$

where (x, z) is the position of the particle in the current time step while (x_p, z_p) is the position of the particle in the previous time step. A simple approximation of (x_p, z_p) would be one forward Euler step:

$$(x_p, z_p) = (x, z) - (u^n(x, z), w^n(x, z))\Delta t. \tag{B.26}$$

A second-order Runge-Kutta method can improve accuracy by using velocity at middle point (x_m, z_m):

$$\begin{aligned}(x_m, z_m) &= (x, z) - \frac{1}{2}(u^n(x, z), w^n(x, z))\Delta t, \\ (x_p, z_p) &= (x_m, z_m) - \frac{1}{2}(u^n(x_m, z_m), w^n(x_m, z_m))\Delta t.\end{aligned} \tag{B.27}$$

Although we solve only for (x, z) at the cell and face centers, (x_m, z_m) and (x_p, z_p) can be arbitrary and they are approximated by bilinear interpolation from the discretized grid values. Finally, we must handle different boundary conditions for those (x, z) on the boundary of the water domain.

Static Boundary Conditions

In this section we discuss the handling of static boundaries. Section 5.3 discusses the coupling of the shallow water to the body of the swimming creature. In our simulation, we indicate boundaries by setting a SOLID flag in the cells of the MAC grid. We support both the surrounding boundaries and immersed objects as static boundaries. Two types of boundary conditions are implemented for our simulation—reflecting boundaries can generate natural reflection of incoming

waves, whereas absorbing boundaries give the illusion of open water of infinite extent. We use the absorbing boundary condition for the box of water that moves together with the creature during simulation.

Reflecting boundary conditions are achieved by setting water velocities at the boundary to have a zero normal component:

$$(u, w)_b \cdot \mathbf{n}_b = 0. \tag{B.28}$$

For the walls of a rectangle pool, this is as simple as setting $u = 0$ for the cells forming the left and right wall and $w = 0$ for the cells forming the front and back wall. The other boundary condition for the height field is traditionally set by mirroring the height of the water along the boundary line:

$$\left(\frac{\partial h}{\partial x}, \frac{\partial h}{\partial z}\right) \cdot \mathbf{n}_b = 0. \tag{B.29}$$

We modified this by setting it to a constant height $h_b = h_t$ representing our target water level. This achieves the same reflection result as mirroring and it keeps the total water surface level around the target value, eliminating the possibility of overflow or depletion of the pool water level due to the accumulation of numerical errors.

Absorbing boundaries are more challenging. For absorbing boundary conditions, we implemented the first-order Higdon (1994) boundary condition

$$\left(\frac{\partial}{\partial t} + \sqrt{gh}\frac{\partial}{\partial x}\right) h = 0. \tag{B.30}$$

We apply this absorbing boundary condition over a box centered around our swimming creature. Without loss of generality, for the boundary cells at the left

side of the box ($i = 0$), the height and velocity are set to

$$\begin{aligned} h_{0,j}^{n+1} &= \frac{\Delta x h_{0,j}^{n} + \Delta t \sqrt{g h_{1,j}^{n+1}}\, h_{1,j}^{n+1}}{\Delta x + \Delta t \sqrt{g h_{1,j}^{n+1}}}, \\ u_{\frac{1}{2},j}^{n+1} &= u_{\frac{1}{2},j}^{n} - g \frac{h_{1,j}^{n+1} - h_{0,j}^{n+1}}{\Delta x} \Delta t, \\ w_{0,j+\frac{1}{2}}^{n+1} &= 0. \end{aligned} \tag{B.31}$$

The update of u is the same as (B.23) to let waves flow past the boundary in the x direction.

With absorbing boundaries, we can translate the fluid simulation grid together with the creature during simulation. To this end, we monitor the difference of the grid center and the creature center. When the difference along the x or z axes exceeds the cell dimension Δx or Δz, we shift the grid by one cell in the respective direction. We shift at most one cell at a time at the end of the simulation loop (see Appendix D). In practice, if we shift, e.g., in x, we copy data from column $i + 1$ to column i for all columns but the last. We simply retain the data of the last column and, in next iteration, it serves as the right boundary cell and will be updated accordingly by (B.31).

In our system, the update for the shallow water simulation iterates at 500 fps, for a grid of 200×200 resolution that contains the anguilliform swimmer in a 80×80 area, and it can move as much as 200 units of distance per second, which suffices to keep up with the swimmer's fastest swimming speed.

APPENDIX C

Optimization of Locomotion Control

To gain more insight into the mapping of the control space to locomotion performance, we defined measures of locomotion performance and did a quantitative analysis. We measured two values in our locomotion experiments—the first is the linear speed v_s and the second value is the energy consumption rate r_e, calculated as the total amount of work the body exerted against water in unit time to sustain its speed. The total work against water over time Δt is

$$W = \Delta t \int_S -f(s) \cdot \mathbf{v}(s)\, ds. \tag{C.1}$$

With $f(s)$ as defined in (5.2) and $r_e = W/\Delta t$, we obtain

$$r_e = \int_S \rho_w \max[0, \mathbf{n} \cdot \mathbf{v}](\mathbf{n} \cdot \mathbf{v})^2\, ds. \tag{C.2}$$

We approximate r_e in our simulator by calculating r_e of all exposed triangle surfaces and summing them. Fig. C.1 shows our experiment over six anguilliform swimmers with a combination of two different f and three different ps. Several observations can be made from the chart of v_s and r_e over time. First, as the anguilliform swimmer oscillates its body during locomotion, v_s and r_e also oscillate at the same frequency. Second, there are two stages of locomotion—a ramp-up stage and a stable-speed stage. For our subsequent experiments, we measure only during the stable-speed stage to obtain stabilized v_s and r_e. Finally, we can see that although the green swimmer locomotes fastest, the red one is the real

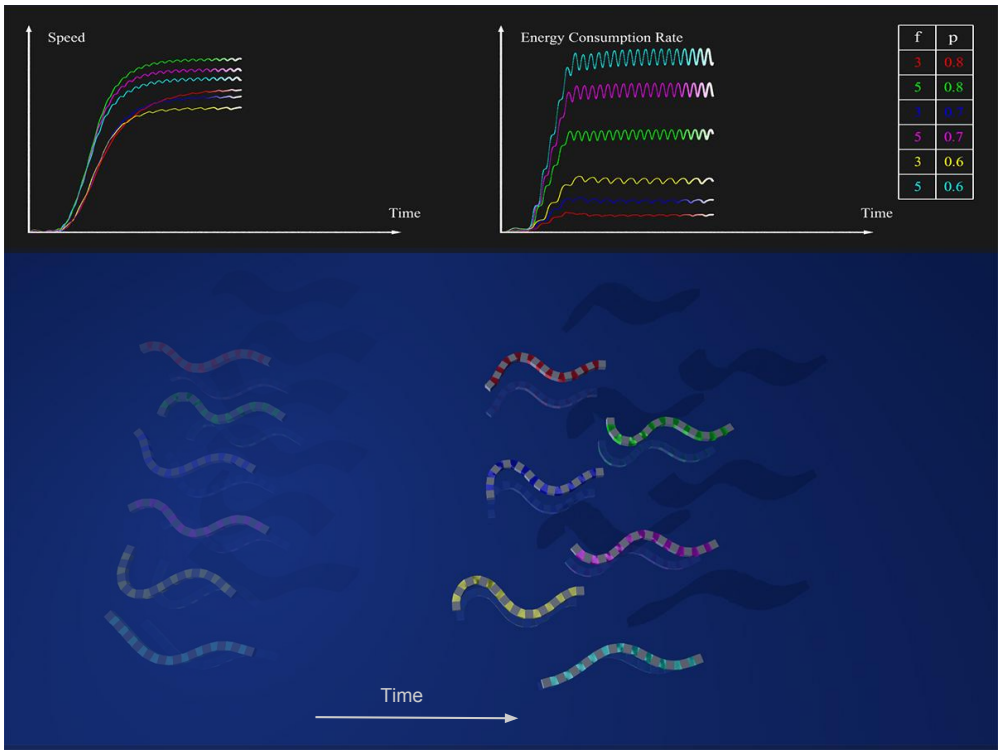


Figure C.1: Measuring the speed and r_{ec} of different anguilliform swimmers during ramp-up.

champion in terms of converting consumed energy into kinetic energy. Kern et al. (2008) used this conversion rate of consumed energy toward kinetic energy as the optimization goal. While this makes perfect sense if our goal is to find the one best set of parameters for our anguilliform swimmer, it is not really useful for a real eel. A real eel would locomote at different speeds when performing different tasks: It could be swimming at a moderate speed when foraging and suddenly speed up to chase a spotted prey; after catching and eating its prey, it might then swim slowly while digesting. Since our simulated animals are autonomous and will perform the aforementioned tasks with various speeds, we would like to obtain the set of parameters that yield optimal energy efficiency for a range of speeds, instead of just one global optimum.

Fig. C.2 shows the optimization we performed over a coarse grid of *f-ps* parameter pairs. We run 10 anguilliform swimmers in parallel. For each, its

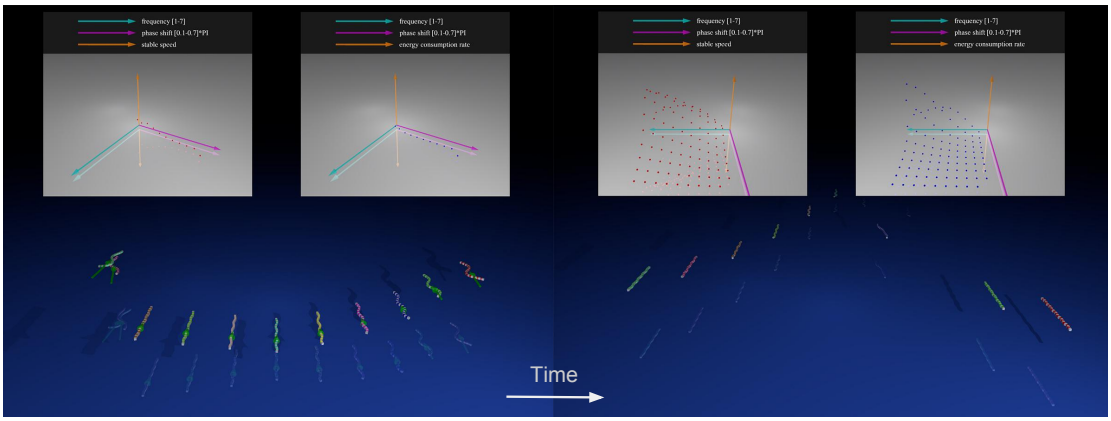


Figure C.2: Heuristic training over 2D parameter space of frequency and phase-shift, the result shown here is a coarse sampling of the parameter space.

swimming controller will be initialized with an (f, ps) parameter pair from the task pool. After reaching stable the stage, it will be evaluated for its v_s and r_e, then assigned a new parameter pair. For our high resolution (200 by 200) results shown in Fig. C.3 and Fig. C.4, the experiment took less than 2 hours to run. There are two interesting observations from Figs. C.3 and C.4. First, both profiles share the same hill shape, with the energy profile's shape having a larger slope. As shown in Fig. C.5(C), r_e and v_s share a quadratic relationship. Second, for the majority of the range of ps, the speed increases as frequency increase, however, when ps become really small(0.10π–0.18π), the speed decreases as frequency continues to increase. This extreme case happens probably due to the fact that when ps is smaller than 0.18π, the body cannot even form a complete wave, but only a large curvature that is hydrodynamically inefficient. Notice that the body is comprised of 11 deformable segments, so 0.18π is just sufficient to form a complete wave of 2π.

After obtaining the profile of stable speed and r_e over the f-ps control space, we can get the most energy efficient control parameters that sustain each speed. In Fig. C.5(A), we first obtain the set of f-ps parameter pairs (green dots) that achieve certain speed, then we locate their energy curve in (B) and find the lowest energy point (red) in (D). (C) shows the quadratic relationship between the r_e

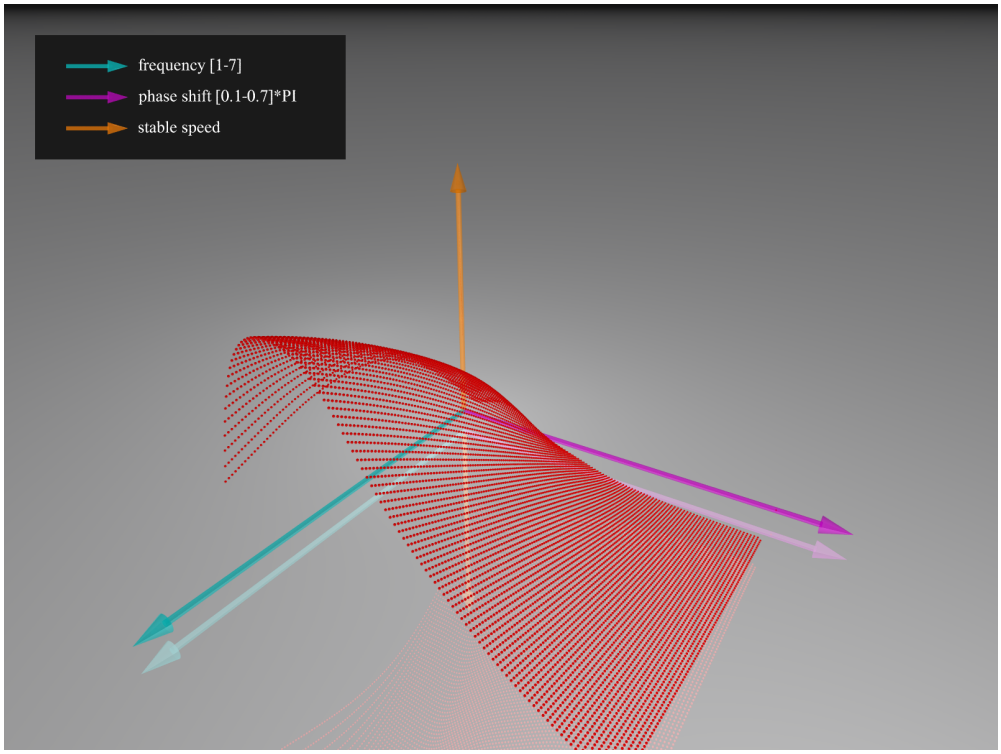


Figure C.3: Stable speed profile over *f-ps* parameter space.

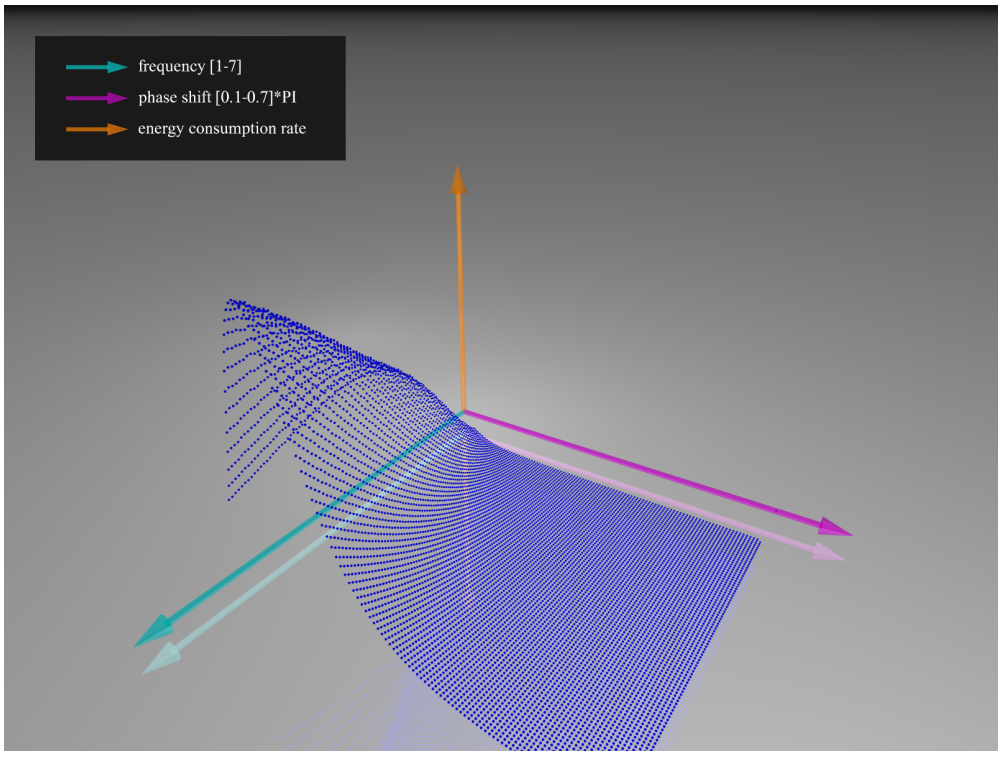


Figure C.4: Energy consumption rate profile over f-ps parameter space.

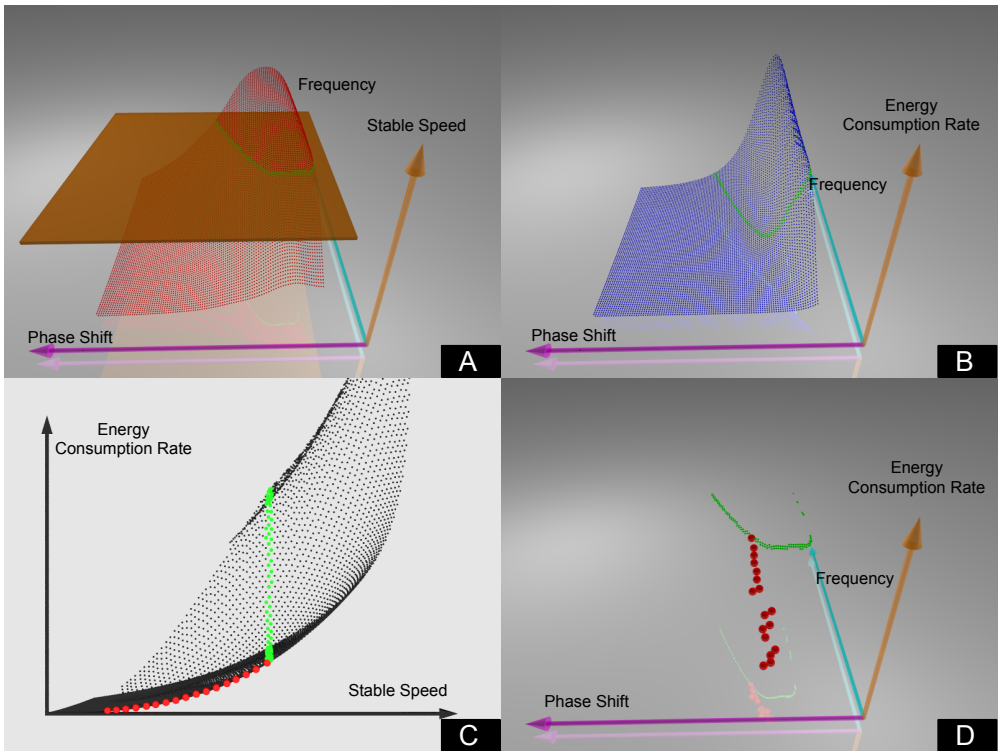


Figure C.5: Optimizing f-ps parameters for each speed.

and the stable speed; essentially, we are selecting the red points that represent the optimal controllers. To our surprise, the range of optimal phase shift is very small (0.28π–0.33π). This result matches and explains the observations Grillner (1974) made from his neurological experiments on lampreys. A nearly constant phase shift for the segmental oscillators actually achieves optimal energy efficiency toward all locomotion speeds. The optimal set of (f, ps) form a line and through regression we can obtain its analytical expression to sample for any speed in the range.

We use this result when setting the speed for our autonomous anguilliform swimmer in Section 5.5.

APPENDIX D

The Simulation Loop

In this Appendix, we detail the update procedures of our simulation. For our object oriented implementation, each class has its own update procedure and they are organized in a hierarchy. For example, the update function of the *world* class will trigger the update of all registered objects, including the virtual myriapod and the terrain with shallow water. Similarly, the update of the myriapod will trigger the updates of its body parts, such as the antennae, head, and body segments. The update procedures for the body parts usually contain two components, a control update and a physics update. The updates of different objects such as the shallow water and individual myriapods, can be parallelized using OpenMP within the loop of *world*'s update procedure.

Update of the Antenna

1. The rotation angles for the root link are updated with a constant rotation velocity toward the target rotations;
2. The rotation angles for the remaining links are updated by sampling from a wave function;
3. If a collision with an obstacle occurs, a retraction phase is triggered.

Update of the Head

1. The sensory information and internal states are processed and, if necessary, the brain and body modes are switched. Sections 4.4 and 6.3 list the different

brain and body modes;

2. If the body mode is In-air or Air-to-ground, the segment's position and orientation are updated via rigid-body dynamics simulation in air;
3. If the body mode is In-water, the segment's position and orientation are updated via rigid-body dynamics in water;
4. If the body mode is Water-to-ground or On-ground, update the segment kinematically as follows:
   (a) The Balancer adjusts the orientation and elevation of the segment to keep it balanced;
   (b) The head segment's position and orientation are updated according to the brain's mode. The head always moves in the direction in which it is pointing and its orientation is updated to execute a turn.
5. If the body mode is Ground-to-water, the segment is updated both kinematically to lead the body into water and dynamically to begin forming the serpentine body shape.

Update of the Rigid Segments

1. If the segment is in the *unsupported* or *submerged* state, it will be updated dynamically with or without water forces depending on its location.
2. If the segment is in the *supported* state, it will be updated kinematically, as follows:
   (a) The Balancer will adjust the orientation and elevation of the segment to keep it balanced;
   (b) The Follower will adjust the position and orientation of the segment. The leg root positions are updated accordingly.

Update of the Legs

1. The Switcher checks the leg states and updates them accordingly;
2. If a leg is in the STANCE state, the IK solver (Section 3.4) computes the leg rotation angles from the tip and root position of the leg;
3. Otherwise, forward kinematics will update the leg rotations and tip positions towards the target poses of that state.

Update of the Deformable Segments

1. The deformable segments are always dynamically updated via the elasticity simulation (Appendix A) and the coupling method described in Section 3.3;
2. When the entire body is out of water, it deforms according to the simulation passively;
3. When the entire body is in water, the rest shapes of the deformable segments will be updated by the muscle controller to actuate the body for swimming;
4. When a segment is in water but the body mode is still GROUND-TO-WATER, the rest shape of the deformable segments will be morphed to prepare for swimming control.

Update of the Shallow Water

1. The velocity and height are advected using the semi-Lagrangian method.
2. The velocity field is updated according to the hydrostatic pressure.
3. The height field is updated by according to the velocity field to preserve volume.
4. If rain simulation is on, rain drops are added to the water height map.
5. The heights and velocities at cells that comprise the static boundary are updated depending on the selected boundary condition, either reflective or absorbing.

6. Cells occupied by the swimming creature are identified and updated to achieve coupling.
7. If enabled, the entire simulation grid is translated to keep the moving creature centered.

APPENDIX E

Rendering

We use the open-source POV-Ray software to render our simulations. The rigid segments and legs of the physical model are rendered as rigid primitives (cubes, spheres, cylinders). The deformable segments are rendered as translucent jellies (Fig. 1.3(C) and Fig. 3.1). We output the position and orientation of those primitives into a *.pov* file that can be included by a POV-Ray scene script to render. The scene script contains definitions of the camera, illumination, background, and material properties of all the geometric entities. We output each deformable segment as a *mesh2* object that contains the surface triangle mesh of the segment, along with vertex positions and normals.

For the textured myriapod mesh models shown in Fig. 1.3(A) and Fig. 6.2, some extra work is needed to connect the physics output to the surface mesh. We purchased the models from *TurboSquid* and rigged a skeleton that represents the biomechanical model. After attaching the rig to the surface mesh, we can apply basic affine transformations, such as translation and rotation, to nodes on the rig. The geometric parameters of the mesh model, such as segment size and leg length, are calibrated and set in the simulation. Per-frame data, including rigid segment position, orientation, leg rotation angles, etc., are exported at 30 frames per-second as *.mel* Maya script files to set keyframes of the rigged model. Maya can then output the correctly rigged model as *.obj* files which can be used by Pov-ray to render together with other objects.

The graphs in Fig. C.1 and Fig. C.5 are also rendered by POV-Ray, by

representing the data points as spheres. The vector graphics in this thesis are rendered using Inkscape (Fig. B.1) and Google Drawings (Fig. 3.4), both of which are free software.

We rendered our shallow water model with caustics by casting millions of photons into the scene and having the water reflecting and refracting those photons. It is a nontrivial trial-and-error process to set the correct rendering parameters such that the water surface will look natural. Photon mapping tremendously increases the rendering time required to achieve a reasonable output speed for our rendering. We therefore compromised on the total number of photons cast and the level of tracing that we perform. Our final animations are rendered using about 500 million photons with a max trace level of 4. It takes around 5–12 minutes to render one frame of 1600×1200 pixels on a single core of our 3.2 GHz Intel Core i7-3930k computer. The rendering speed varies greatly depending on the projected area of water pool in the image.

For the animation shown in Fig. 5.12(B), we implemented a stable camera-following mechanism in our simulator to mitigate high-frequency shaking of the view. We store a position $\mathbf{P}_{cam}$ and heading $\mathbf{H}_{cam}$ for our camera. Position $\mathbf{P}_{cam}$ is initialized to be distance d behind the center $\mathbf{P}$ of the creature's body, and $\mathbf{H}_{cam}$ is initialized to be the heading $\mathbf{H}$ of the creature. As the creature swims autonomously, we monitor the difference between $\mathbf{H}_{cam}$ and $\mathbf{H}$. If the angular difference exceeds a threshold α_τ, we turn the camera swiftly toward $\mathbf{H}$, which mitigates the transfer to the camera of high-frequency oscillations in $\mathbf{H}$. The camera position $\mathbf{P}_{cam}$ is constantly updated by $\mathbf{P}_{cam} \mathrel{+}= \Delta t \mathbf{H}_{cam} k(d - |\mathbf{P} - \mathbf{P}_{cam}|)$, which updates $\mathbf{P}_{cam}$ as if a spring with constant k (we use $k = 10$ in our simulation) drags it in the direction of $\mathbf{H}_{cam}$. Finally, we raise the camera by adding a positive displacement to $\mathbf{P}_{cam}$ along the vertical axis and adding a negative vertical vector to $\mathbf{H}_{cam}$.

Bibliography

Arash, O. E., Gnevaux, O., Habibi, A., and Dischler, J.-M. (2003). Simulating fluid-solid interaction. In *In Graphics Interface*, pages 31–38.

badboythuglife (2009). Centipede Swimming. https://www.youtube.com/watch?v=4EfqAD85Js0. [Online; published 24-May-2009].

Baraff, D. (1997). An introduction to physically based modeling: Rigid body simulation I—Unconstrained rigid body dynamics. In *An Introduction to Physically Based Modeling, SIGGRAPH '97 Course Notes*, page 97.

Baraff, D. and Witkin, A. P. (1997). Partitioned dynamics. Technical Report CMU-RI-TR-97-33, The Robotics Institute, Canegie-Mellon University, Pittsburgh, PA.

Beer, R. D. (1990). *Intelligence as adaptive behavior: An experiment in computational neuroethology*. Academic Press, San Diego, CA.

Beer, R. D., Chiel, H., Quinn, R., and Espenschied, K. (1992). A distributed neural network architecture for hexapod robot locomotion. *Neural Computation*, 4(6):356–365.

Bergmann, M. and Iollo, A. (2011). Modeling and simulation of fish-like swimming. *J. Comput. Phys.*, 230(2):329–348.

Bifrost1107 (2009). Centipede. https://www.youtube.com/watch?v=JP_c1Gzbcro. [Online; published 23-November-2009; from 2:07 to 2:27].

Bone, Q. (1975). Muscular and energetic aspects of fish swimming. In Wu, T.-T., Brokaw, C., and Brennen, C., editors, *Swimming and Flying in Nature*, pages 493–528. Springer US.

Bowtell, G. and Williams, T. L. (1991). Anguilliform body dynamics: Modelling the interaction between muscle activation and body curvature. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 334(1271):385–390.

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. MIT Press, Cambridge, MA.

Bridson, R. (2008). *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press.

Brodtkorb, A. R., Stra, M. L., and Altinakar, M. (2012). Efficient shallow water simulations on gpus: Implementation, visualization, verification, and validation. *Computers & Fluids*, 55(0):1 – 12.

Brozovich, F. V., Yates, L. D., and Gordon, A. M. (1988). Muscle force and stiffness during activation and relaxation: Implications for the actomyosin ATPase. *The Journal of General Physiology*, 91(3):399–420.

Carlson, M., Mucha, P. J., and Turk, G. (2004). Rigid fluid: Animating the interplay between rigid bodies and fluid. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 377–384, New York, NY, USA. ACM.

Cenydd, L. a. and Teahan, B. (2013). An embodied approach to arthropod animation. *Computer Animation and Virtual Worlds*, 24:65–83.

Chen, D. T. and Zeltzer, D. (1992). Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. *Computer Graphics (Proc. ACM SIGGRAPH 92)*, 26:89–98.

Coros, S., Karpathy, A., Jones, B., Reveret, L., and van de Panne, M. (2011). Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics*, 30:59:1–12.

Coros, S., Martin, S., Thomaszewski, B., Schumacher, C., Sumner, R., and Gross, M. (2012). Deformable objects alive! *ACM Transactions on Graphics*, 31:69:1–9.

Cruse, H., Dean, J., Dürr, V., Kindermann, T., Schmitz, J., and Schumm, M. (2000). Control of hexapod walking: A decentralized solution based on biological data. In *Proc. 4th Int. Conf. on Climbing and Walking Robots*, 13, pages 1–10.

Cruse, H., Kindermann, T., Schumm, M., Dean, J., and Schmitz, J. (1998). Walknet: A biologically inspired network to control six-legged walking. *Neural Networks*, 11:1435–1447.

Drucker, E. G. and Lauder, G. V. (2002). Experimental hydrodynamics of fish locomotion: Functional insights from wake visualization. *Integrative and Comparative Biology*, 42(2):243–257.

Ekeberg, . (1993). A combined neuronal and mechanical model of fish swimming. *Biological Cybernetics*, 69(5-6):363–374.

Faloutsos, P., van de Panne, M., and Terzopoulos, D. (2001). Composable controllers for physics-based character animation. In *Proc. ACM SIGGRAPH 01*, pages 251–260.

Full, R. J. and Tu, M. S. (1991). Mechanics of a rapid running insect: Two-, four- and six-legged locomotion. *Journal of Experimental Biology*, 156:215–231.

Gates, W. F. (2002). Animation of fish swimming. Technical report, University of British Columbia, Vancouver, BC, Canada, Canada.

Gibson, D. P., Oziem, D. J., Dalton, C. J., and Campbell, N. W. (2007). A system for the capture and synthesis of insect motion. *Graphical Models*, 69:231–245.

Golubitsky, M., Stewart, I., Buono, P.-L., and Collins, J. (1998). A modular network for legged locomotion. *Physica D: Nonlinear Phenomena*, 115:56–72.

Gray, J. (1933). Studies in animal locomotion. i. the movement of fish with special reference to the eel. *Journal of Experimental Biology*, 10:88–104.

Grillner, S. (1974). On the generation of locomotion in the spinal dogfish. *Experimental Brain Research*, 20(5):459–470.

Grzeszczuk, R. and Terzopoulos, D. (1995). Automated learning of muscle-actuated locomotion through control abstraction. In *Proc. ACM SIGGRAPH 95*, pages 63–70.

Guo, S., Chang, J., Cao, Y., and Zhang, J. (2014). Special section on cad/graphics 2013: A novel locomotion synthesis and optimisation framework for insects. *Comput. Graph.*, 38:78–85.

Guo, S., Tharib, S., Chang, J., and Zhang, J. (2013). Biologically inspired motion pattern design of multi-legged creatures. In *Evolutionary and biologically inspired music, sound, art and design*, volume 7834, pages 145–156. Springer.

Higdon, R. (1994). Radiation boundary conditions for dispersive waves. *SIAM Journal on Numerical Analysis*, 31(1):64–100.

Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. (1995). Animating human athletics. In *Proc. ACM SIGGRAPH 95*, pages 71–78.

Hoffman, K. and Wood, R. (2011). Myriapod-like ambulation of a segmented microrobot. *Autonomous Robots*, 31:103–114.

Holmes, P., Full, R. J., Koditschek, D., and Guckenheimer, J. (2006). The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM Review*, 48:207–304.

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653.

Ijspeert, A. J., Crespi, A., Ryczko, D., and Cabelguen, J.-M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315:1416–1420.

Inagaki, S., Niwa, T., and Suzuki, T. (2011). Decentralized control of centipede-like multi-legged robots with passive intersegment joints based on follow-the-contact-point gait control. *Transactions of the Society of Instrument and Control Engineers*, 47:282–290.

Inagaki, S., Yuasa, H., and Arai, T. (2003). CPG model for autonomous decentralized multi-legged robot system generation and transition of oscillation patterns and dynamics of oscillators. *Robotics and Autonomous Systems*, 44:171–179.

Kern, S., Chatelain, P., and Koumoutsakos, P. (2008). Modeling, simulation and optimization of anguilliform swimmers. In Kato, N. and Kamimura, S., editors, *Bio-mechanisms of Swimming and Flying*, pages 167–178. Springer Japan.

Kimura, H., Fukuoka, Y., and Cohen, A. H. (2007). Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts. *International Journal of Robotics Research*, 26:475–490.

Kwatra, N., Wojtan, C., Carlson, M., Essa, I. A., Mucha, P. J., and Turk, G. (2010). Fluid simulation with articulated bodies. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):70–80.

Layton, A. T. and van de Panne, M. (2002). A numerically efficient and stable algorithm for animating water waves. *The Visual Computer*, 18(1):41–53.

Lentine, M., Gretarsson, J. T., Schroeder, C., Robinson-Mosher, A., and Fedkiw, R. (2011). Creature control in a fluid environment. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):682–693.

Lighthill, M. J. (1970). Aquatic animal propulsion of high hydromechanical efficiency. *Journal of Fluid Mechanics*, 44:265–301.

McAdams, A., Zhu, Y., Selle, A., Empey, M., Tamstorf, R., Teran, J., and Sifakis, E. (2011). Efficient elasticity for character skinning with contact and collisions. *ACM Transactions on Graphics*, 30(4):37:1–12.

McKenna, M. and Zeltzer, D. (1990). Dynamic simulation of autonomous legged locomotion. *Computer Graphics (Proc. ACM SIGGRAPH 90)*, 24:29–38.

Mellen, N., Kiemel, T., and Cohen, A. H. (1995). Correlational analysis of fictive swimming in the lamprey reveals strong functional intersegmental coupling. *Journal of Neurophysiology*, 73:100–1030.

Miller, G. S. P. (1988). The motion dynamics of snakes and worms. *Computer Graphics (Proc. ACM SIGGRAPH 88)*, 22:169–173.

Minelli, A. (2011). *Treatise on Zoology - Anatomy, Taxonomy, Biology. The Myriapoda, Volume 1.* Brill.

Narain, R., Golas, A., and Lin, M. C. (2010). Free-flowing granular materials with two-way solid coupling. *ACM Trans. Graph.*, 29(6):173:1–173:10.

Nor, N. M. and Ma, S. (2014). A simplified cpgs network with phase oscillator model for locomotion control of a snake-like robot. *J. Intell. Robotics Syst.*, 75(1):71–86.

O'Brien, J. F. and Hodgins, J. K. (1995). Dynamic simulation of splashing fluids. In *CA*, pages 198–.

Odashima, T., Yuasa, H., and Ito., M. (1998). The autonomous decentralized myriapod locomotion robot which is consist of homogeneous subsystems. *Journal of the Robotic Society of Japan*, 16:81–88.

Raibert, M. (2008). Bigdog, the rough-terrain quadruped robot. In *Proc. 17th International Federation of Automatic Control World Congress*, pages 10822–10825.

Raibert, M. H. and Hodgins, J. K. (1991). Animation of dynamic legged locomotion. *Computer Graphics (Proc. ACM SIGGRAPH 91)*, 25:349–358.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34.

Ruppert, E. E., Fox, R. S., and Barnes, R. D. (2003). *Invertebrate Zoology, 7th Ed.* Cengage Learning.

Saranli, U., Buehler, M., and Koditschek, D. E. (2001). RHex: A simple and highly mobile hexapod robot. *International Journal of Robotics Research*, pages 616–631.

Shinar, T., Schroeder, C., and Fedkiw, R. (2008). Two-way coupling of rigid and deformable bodies. In *Symposium on Computer Animation*, pages 95–103.

Si, W., Lee, S.-H., Sifakis, E., and Terzopoulos, D. (2014). Realistic biomechanical simulation and control of human swimming. *ACM Trans. Graph.*, 34(1):10:1–10:15.

Sifakis, E., Shinar, T., Irving, G., and Fedkiw, R. (2007). Hybrid simulation of deformable solids. In *Symposium on Computer Animation*, pages 81–90.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artificial Life*, 1(4):353–372.

Skrba, L., Reveret, L., Hétroy, F., Cani, M.-P., and O'Sullivan, C. (2008). Quadruped animation. In *Eurographics State-of-the-Art Report*.

Stam, J. (1999). Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Stomakhin, A., Howes, R., Schroeder, C., and Teran, J. M. (2012). Energetically consistent invertible elasticity. In *Symposium on Computer Animation*, pages 25–32.

Stomakhin, A., Schroeder, C., Chai, L., Teran, J., and Selle, A. (2013). A material point method for snow simulation. *ACM Trans. Graph.*, 32(4):102:1–102:10.

Tan, J., Gu, Y., Turk, G., and Liu, C. K. (2011). Articulated swimming creatures. *ACM Trans. Graph.*, 30(4):58:1–58:12.

Teran, J., Blemker, S., Hing, V. N. T., and Fedkiw, R. (2003). Finite volume methods for the simulation of skeletal muscle. In *Symposium on Computer Animation*, pages 68–74.

Terzopoulos, D. (1999). Artificial life for computer graphics. *Communications of the ACM*, 42(8):32–42.

Thurey, N., Müller-Fischer, M., Schirm, S., and Gross, M. (2007). Real-time breakingwaves for shallow water simulations. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, PG '07, pages 39–46, Washington, DC, USA. IEEE Computer Society.

Transeth, A. a., Pettersen, K. y., and Liljebäck, P. (2009). A survey on snake robot modeling and locomotion. *Robotica*, 27(7):999–1015.

Tsai, Y.-Y., Lin, W.-C., Cheng, K. B., Lee, J., and Lee, T.-Y. (2010). Real-time physics-based 3D biped character animation using an inverted pendulum model. *IEEE Transactions on Visualization and Computer Graphics*, 16:325–337.

Tu, X. and Terzopoulos, D. (1994). Artificial fishes: Physics, locomotion, perception, behavior. In *Proc. ACM SIGGRAPH 94*, pages 43–50.

Van Welbergen, H., Van Basten, B. J. H., Egges, A., Ruttkay, Z. M., and Overmars, M. H. (2010). Real time animation of virtual humans: A trade-off between naturalness and control. *Computer Graphics Forum*, 29:2530–2554.

Wang, J. M., Hamner, S. R., Delp, S. L., and Koltun, V. (2012). Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics*, 31:25:1–11.

Wu, J.-c. and Popović, Z. (2003). Realistic modeling of bird flight animations. *ACM Trans. Graph.*, 22(3):888–895.

Yu, J. and Turk, G. (2013). Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph.*, 32(1):5:1–5:12.

Zhu, B., Lu, W., Cong, M., Kim, B., and Fedkiw, R. (2013). A new grid structure for domain extension. *ACM Trans. Graph.*, 32(4):63:1–63:12.