Title
Bio-Inspired Simulation With Learning-Based Automatic Motion Control

Author
Zeng, Xiao

Publication Date
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Bio-Inspired Simulation With Learning-Based Automatic Motion Control

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

by

Xiao Zeng

2022

ABSTRACT OF THE DISSERTATION

Bio-Inspired Simulation With Learning-Based Automatic Motion Control

by

Xiao Zeng
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2022
Professor Demetri Terzopoulos, Chair

Machine learning-based motion control of 3D characters is a classic computer graphics problem with applications spanning from the entertainment industry to robotics. In this thesis, we present novel approaches to achieving automatic and adaptive control of the simulated character motions that are anatomically consistent. More specifically, our main contribution is twofold. First, we develop a neuromuscular controller for a muscle-driven face-head-neck biomechanical model, which comprises 72 neck muscles and 52 facial muscles. Our pipeline transfers facial expressions and head poses from reference images and videos onto the biomechanical model in real-time. Second, we propose a deep reinforcement learning-driven controller for fish behavior in order to simulate ecologically valid underwater scenes. Our simulation framework, named DeepFoids, autonomously generates fish schooling patterns that accommodate environmental changes in factors such as neighbor interaction, light intensity, and underwater temperature. Our approaches are advantageous relative to previous methods as they (1) eliminate human labor in data collection, (2) require minimal manual calibration under different simulation settings, (3) retain a high level of biological accuracy, and (4) leverage the power of deep learning to capture the non-linear relations between the controlling variables and desired motions.

The dissertation of Xiao Zeng is approved.

Chenfanfu Jiang

Bolei Zhou

Achuta Kadambi

Song-Chun Zhu

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2022

To my family, friends, and mentors
who supported me all the way to my PhD degree
among so many other things

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I cannot adequately express my gratitude to Professor Demetri Terzopoulos, who has been a superb and thoughtful mentor throughout my MS and PhD programs at UCLA. When I first entered UCLA five years ago, I had no research experience nor had I taken any class related to Computer Graphics. It was Demetri's graduate courses that brought me into the fascinating world of biomechanical human modeling and artificial life. I was inspired by this experience to join his prestigious UCLA Computer Graphics & Vision Lab and, more importantly, became determined to pursue a PhD degree. Demetri has been so supportive of not only my research, but also my life. He is an insightful, kind, and hard-working mentor and role model for me. Our conversation about study plans, research directions, and even personal relationships are fresh in my memory, and have guided me to overcome many challenges. Without his help, I would never have accomplished the results in this thesis. I am greatly indebted to him for his unconditional support.

I would like to thank Professors Song-Chun Zhu, Chenfanfu Jiang, Achuta Kadambi, and Bolei Zhou for serving on my thesis committee and offering me invaluable advice on how to improve my dissertation.

I would like to express my particular appreciation to Dr. Masaki Nakada. Masaki has been a former labmate, a reliable friend, and a helpful research mentor for the past few years. I worked under his supervision on the bio-inspired fish simulation projects as a member of the NeuralX team. I also received much advice from him about the biomechanical face model. It has been a great pleasure working with him, and his guidance and encouragement were inspiring to many aspects of my research.

I would also like to thank my wonderful co-author labmates, including Surya Dwarakanath who contributed significantly to the development of our facial expression transfer pipeline and Wuyue Lu who helped me build the head-neck complex.

I am thankful to Dr. Alex Vasilescu for supervising me prior to my PhD study on a project of semi-supervised face and object classification in the tensor framework. Her high standards and passion helped me to quickly grow as an academic researcher. I am

grateful to Professor Sung-Hee Lee and to Wilson Yan. The musculoskeletal head-neck and hand-forearm systems that I used in my research are largely based on the models they developed for their theses. I am also grateful to Arjun Lakshmipathy and Sarah Gross for sharing useful information and datasets that were valuable inputs to my research.

Furthermore, my sincere appreciation goes to my industry collaborators. Dr. Yuko Ishiwaka at SoftBank Corporation has been extremely helpful and supportive in the fish simulation project. I also want to thank her colleagues Michael Lee Eastman, Sho Kakazu, Dr. Shun Ogawa, and Dr. Tadayuki Tone, as well as Ryosuke Mizutani from Nosan Corporation for providing enormous help in the collection of field data and the incorporation of fish animation. I appreciate Donovan Michael Westwater, my colleague at NeuralX, Inc., for his assistance in synthetic data generation and physics-based environment simulation.

I was very fortunate to be guided by Dr. Jianyuan Min and Rajat Vikram Singh during my internship at NVIDIA Corporation, where I had an amazing opportunity to work on face semantic editing using Generative Adversarial Networks.

My appreciation also goes to Professor Ming-Chun Huang, my undergraduate research mentor, and my excellent fellows in the Sensing and Interaction Lab at Case Western Reserve University, Dr. Haotian Jiang and Dr. Yi Cai. I thank them immensely for their precious inputs to my career.

My special thanks goes to Joseph Brown and Juliana Alvarez in the UCLA Computer Science Department's Graduate Student Affairs Office, who offered me tremendous support with genuine compassion over the last five years.

Finally, I thank my family for their unlimited love—my parents, Dr. Yu Zeng and Qirong Fan, who gave me life and continue to nurture me with unreserved support and constant care; my grandmother Yuqiang Liu and grandfather Zhongyao Fan, who passed away last year, both of whom accompanied me throughout my childhood and raised me to become the person I am today; and my beloved, Bijun Li, who has been my source of motivation and has had unconditional faith in me.

VITA

2017	B.A. (Computer Science) and B.S. (Marketing) Case Western Reserve University Cleveland, Ohio
2019	M.S. (Computer Science) University of California, Los Angeles Los Angeles, California
2020–2022	Research Engineer NeuralX, Inc Los Angeles, California
2020–2022	Teaching Associate Computer Science Department University of California, Los Angeles Los Angeles, California
2022	Deep Learning Computer Vision and Graphics Software Intern NVIDIA Corporation Santa Clara, California

PUBLICATIONS

Ishiwaka, Y., Zeng, X. S., Ogawa, S., Westwater, D. M., Tone, T., and Nakada, M. (2022). DeepFoids: Adaptive bio-inspired fish simulation with deep reinforcement learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Ishiwaka, Y., Zeng, X. S., Eastman, M. L., Kakazu, S., Gross, S., Mizutani, R., and Nakada, M. (2021). Foids: Bio-inspired fish simulation for generating synthetic datasets. *ACM Transactions on Graphics (TOG)*, 40(6), 1–15.

Zeng, X. S., Dwarakanath, S., Lu, W., Nakada, M., and Terzopoulos, D. (2021). Neuromuscular control of the face-head-neck biomechanical complex with learning-based expression transfer from images and videos. In *16th International Symposium on Visual Computing (ISVC 2021)*, pages 116–127, Springer.

Zeng, X. S., Dwarakanath, S., Lu, W., Nakada, M., and Terzopoulos, D. (2021). Facial expression transfer from video via deep learning. In *The 20th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2021)*, pages 1–2.

Vasilescu, M. A. O., Kim, E., and Zeng, X. S. (2021). CausalX: Causal eXplanations and block multilinear factor analysis. In *25th International Conference on Pattern Recognition (ICPR 2020)*, pages 10736–10743. IEEE.

Jiang, H., Cai, Y., Zeng, X., and Huang, M. C. (2018). Does background really matter? Worker activity recognition in unconstrained construction environment. In *IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks (BSN 2018)*, pages 50–53. IEEE.

Cui, J., Chen, J., Qu, G., Starkman, J., Zeng, X., Madigan, E., Pekarek, M., Xu, W. and Huang, M. C. (2017). Wearable gait lab system providing quantitative statistical support for human balance tests. *Smart Health*, 3, 27–38.

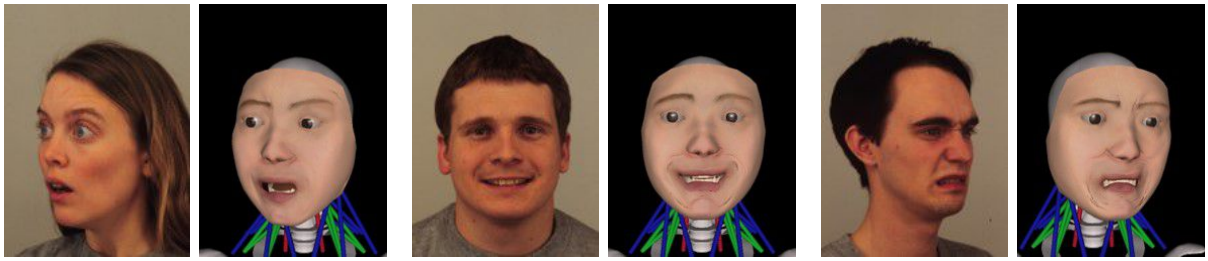
CHAPTER 1

Introduction

Computer simulation has been employed to generate realistic animation and evaluate model performance in the fields of computer graphics, biomechanics, and artificial life for years. Generally speaking, given a well-designed model and well-chosen parameters, simulation should yield the desired behavior and accurately reveal the dynamics of the system, thus proving useful to solving problems involving complex decision-making processes (Olariu and Zomaya, 2005). However, many simulation methods require manual setting of the controlling variables and extensive adjustment if any part of the model or simulation scene is changed. This limits the robustness of the model and, often, its applicability.

This dissertation introduces methodologies that automatically produce realistic and highly adaptive biomechanical and behavioral simulations with minimal manual effort. Specifically, we tackle the problems of learning-based neuromuscular control of a face-head-neck biomechanical complex and reinforcement learning control of a behavioral model of fish schooling. Both of our novel simulation frameworks leverage the power of deep learning to train the respective controllers that generate motions autonomously, thus achieving the goal of adaptive simulation at micro (muscle control) and macro (flocking control) levels of abstraction.

The resulting control schemes are capable of synthesizing natural motions of the human face-head-neck complex and fish species in a physically accurate and highly interactive manner, and are potentially applicable to the creation of training datasets for down-stream computer vision tasks. In the remainder of this chapter, we will discuss these topics in greater detail and preview our approaches to building the automatic simulation



(a) Subject 1 (b) Subject 2 (c) Subject 3

Figure 1.1: Our deep learning framework enables the transfer of facial expressions and head poses from images or videos to a biomechanical face-head-neck model.

frameworks that are the major technical contributions of this dissertation.

1.1 Human Simulation and Control

The face, actuated by the muscles of facial expression, and head movements, actuated by the cervical muscles, are a powerful mode of nonverbal communication between humans. The simulation of the face-head-neck musculoskeletal complex is of importance in understanding how we convey thinking and feeling in fields from affective science to 3D computer animation. Biomechanical human musculoskeletal models realistically capture the anatomy and physics underlying human motion generation. In particular, those pertaining to the simulation of the human face have succeeded in convincingly emulating facial expressiveness; however, they require significant effort in parameter tuning to produce realistic results.

We will show how to endow a biomechanical musculoskeletal model of the human face with the ability to produce facial expressions via machine learning from real-world reference images and videos (Figure 1.1). To this end, we introduce a deep neural-network-based method for learning the representation of human facial expressions through Ekman's Facial Action Coding System (FACS) (Cohn et al., 2007) in the context of the muscle actuators that drive the musculoskeletal face model augmented with a musculoskeletal cervicocephalic (neck-head) system to animate head movement during facial expression synthesis.

1.2 Fish Simulation and Control

The sustainability of the food supply and the protection of marine resources are two of the most important global issues, and are part of the agenda set by the United Nations to be achieved by 2030.[1] Fish farming is one of the keys to a sustainable seafood supply to support the global population, as seafood consumption is growing rapidly. One of the current major problems in the fish farming industry is managing feeding. Under-feeding slows down the growth of the fish due to malnutrition. Over-feeding is even worse as it can kill the fish and the residual food contaminates the marine environment. To optimize the amount and timing of feeding, it is important to automate fish counting using inexpensive commercially available RGB cameras and computer vision. Deep learning has achieved great success in the field of computer vision. The quality of the dataset determines the accuracy of deep learning models, but it is difficult to obtain data to train a fish counting model.

Ishiwaka et al. (2021) proposed a fish schooling simulation called Foids, and demonstrated the efficacy of using a CG synthetic dataset for training. Foids adds rules based on fish biology to the Boids (Reynolds, 1987) algorithm. However, both Boids and Foids require manually setting a number of parameters for a given fish species, which must be adjusted should any conditions change. To solve this problem, we propose a method to autonomously generate schooling behavior in fish using multi-agent deep reinforcement learning. Fish behavior in a fish farming cage depends not only on natural factors like temperature and light intensity, but also the size and shape of the fish cage, and even the species, number, and size of the fish themselves.

By taking into account biological data such as the preferred light intensity, temperature, and inter-fish distance of each fish, our method achieves several distinct patterns of collective behavior depending on population density through autonomous learning. Furthermore, we develop a physically-based underwater environment simulation. This simulation is capable of accurately reproducing the conditions of oceanic scenes of arbitrary

[1] https://sdgs.un.org/2030agenda

(a) (b) (c)

Figure 1.2: (a) 3D fish model. Note that salmonids is used as an example in this figure, but our framework also works on other fish species. (b) 3D synthetic scene with DeepFoids. (c) Result of fish counting system trained using generated synthetic data.

locations and seasons. Figure 1.2 illustrates the three main components of our DeepFoids pipeline. The bio-inspired fish simulation and physically-based environment simulation allow for the creation of a high-quality synthetic dataset, with which we successfully train a deep learning model to count fish of various species in fish cages.

1.3 Contributions

The specific contributions of this dissertation are as follows:

1. **Facial Expression Transfer from Video Via Deep Learning (Zeng et al., 2021a):** The transfer of facial expressions from people to 3D face models is a challenging problem in computer graphics. We present a novel, learning-based approach to transferring facial expressions and head movements from images and videos to a biomechanical model of the face-head-neck musculoskeletal complex. Specifically, leveraging the Facial Action Coding System as an intermediate representation of the expression space, we train a deep neural network to take in FACS Action Units (AUs) and output suitable facial muscle and jaw activations for the biomechanical model. Through biomechanical simulation, the activations deform the face, thereby transferring the expression to the model. The success of our approach is demonstrated through experiments involving the transfer of a range of expressive

facial images and videos onto our biomechanical face-head-neck complex.

2. **Neuromuscular Control of the Face-Head-Neck Biomechanical Complex (Zeng et al., 2021b):** We propose the first biomechanical face-head-neck animation system that is capable of learning to reproduce expressions and head orientations through neuromuscular control. Our novel deep neuromuscular motor controller learns to map between FACS Action Units extracted from human facial images and videos and the activations of the muscle actuators that drive the biomechanical system. As a proof of concept, we demonstrate an automated processing pipeline for animating expressions and head poses using an improved version of the physics-based neck-head-face animation system developed by Lee and Terzopoulos (2006), but which can potentially be applied to any muscle-driven model.

3. **Bio-Inspired Fish Simulation for Generating Synthetic Datasets (Ishiwaka et al., 2021):** We present a bio-inspired fish simulation platform, which we call "Foids", that generates realistic synthetic datasets for use in computer vision algorithm training. This is a first-of-its-kind synthetic dataset platform for fish, which generates all the 3D scenes through simulation. One of the major challenges in deep learning based computer vision is the preparation of the annotated dataset. It is already hard to collect a good quality video dataset with enough variations; moreover, it is a painful process to annotate a sufficiently large video dataset frame by frame. This is especially true when it comes to a fish dataset because it is difficult to set up a camera underwater and the number of fish (target objects) in the scene can range up to 30,000 in a fish cage on a fish farm. All of these fish need to be annotated with labels such as a bounding box or silhouette, which can take hours to complete manually, even for only a few minutes of video. We solve this challenge by introducing a realistic synthetic dataset generation platform that incorporates details of biology and ecology studied in the aquaculture field. Because it is a simulated scene, it is easy to generate the scene data with annotation labels from the 3D mesh geometry data and transformation matrix. To this end, we develop an

automated fish counting system utilizing the part of synthetic dataset that shows comparable counting accuracy to human eyes, which reduces the time compared to the manual process, and reduces physical injuries sustained by the fish.

4. **Adaptive Bio-Inspired Fish Simulation with Deep Reinforcement Learning (Ishiwaka et al., 2022):** Our goal is to synthesize realistic underwater scenes with various fish species in different fish cages, which can be utilized to train computer vision models to automate the fish counting task. It is a challenging problem to prepare a sufficiently diverse labeled dataset of images from aquatic environments. We solve this challenge by introducing an adaptive bio-inspired fish simulation. The behavior of caged fish changes based on the species, size, and number of fish, and the size and shape of the cage, among other variables. We propose a method for achieving schooling behavior for any given combination of variables, using multi-agent deep reinforcement learning (DRL) in various fish cages in diverse environments. Furthermore, to visually reproduce the underwater scene in different locations and seasons, we incorporate a physically-based underwater simulation.

1.4 Thesis Overview

The remainder of the dissertation is organized as follows:

In Chapter 2, we review the existing literature relevant to biomechanical human models, expression transfer, and facial animation. We also introduce the prior work in the area of synthetic data for underwater computer vision task, flocking control, DRL based swimming simulation, as well as important biological and environmental domain knowledge that inspires our design.

In Chapter 3, we present our muscle-actuated human face-head-neck simulation system with neuromuscular control that transfers facial expression and head poses from images and videos onto the model using a learning-based approach.

In Chapter 4, we present DeepFoids, an adaptive bio-inspired fish simulation system

that employs DRL techniques to achieve reasonable schooling behaviors given different environmental settings.

Chapter 5 presents our conclusions, discusses the limitations of our work, and suggests future research directions.

CHAPTER 2

Related Work

2.1 Biomechanical Musculoskeletal Models

Biomechanical musculoskeletal models of the human body synthesize body movements by simulating the physics of hard and soft tissues. They are gradually gaining popularity in computer graphics since pioneering efforts in modeling the dynamics of bones (Hodgins et al., 1995; Faloutsos et al., 2001) and muscles (Chen and Zeltzer, 1992). Sophisticated full-body musculoskeletal models have now emerged that can perform complex motor tasks such as swimming, walking, and sensorimotor control (Lee et al., 2009; Van den Bogert et al., 2013; Si et al., 2014; Nakada et al., 2018; Lee et al., 2019). Although these systems achieve realistic movements with high anatomical accuracy, they often require manual parameter tuning and lots of domain-specific knowledge due to their remarkable complexity. Other efforts have focused on developing biomechanical models of specific body parts, such as the upper extremities (Sueda et al., 2008; Sachdeva et al., 2015), the lower extremities (Rajagopal et al., 2016; Cardona and Cena, 2019), and the head-neck complex (Lee and Terzopoulos, 2006). The latter, which is most relevant to our work in this dissertation, is driven by Hill-type muscle actuators. Its neuromuscular motor controller was trained, using data synthesized by the biomechanical model itself, to synthesize natural head movements. Moreover, it incorporated the biomechanical face model of Lee et al. (1995) for facial expression synthesis.

2.2 Expression Representation and Transfer

Many facial animation researchers have utilized the well-known Facial Action Coding System (FACS) (Ekman and Friesen, 1978), a quantitative phenomenological abstraction that decomposes expressions into the intensity levels of Action Units (AUs), each of which corresponds to the actions of one or more facial muscles. An advantage of using the FACS is that it encodes anatomical and psychological semantics (Cohn et al., 2007).

Expression transfer or retargeting is a trending topic and recent approaches often use FACS-based blendshapes as the basic parametric representation (Weise et al., 2011; Thies et al., 2016; Zhang et al., 2020). Others have used techniques such as interactive mesh deformation control (Xu et al., 2014) and neural-network-based perceptual models (Aneja et al., 2016) to represent expressions in blendshapes. However, transferring expressions using a musculoskeletal system is more natural since facial actions are the result of coordinated muscle contractions inducing soft tissue deformation. Despite the existing literature, there remains a deficiency of work in transferring expressions to musculoskeletal facial systems.

2.3 Muscle-Based Facial Animation

Muscle-based facial animation has been studied for decades (Waters, 1987). A series of studies have endeavored to build increasingly anatomically accurate musculoskeletal face models. Using such models, (Terzopoulos and Waters, 1993) and (Sifakis et al., 2005), among others, have addressed the automatic determination of facial muscle activations from videos, but these methods require nonrigid facial motion trackers or motion capture markers. In this context, the work by Ishikawa et al. (1998, 2000) is of particular interest in that it employed three trained neural networks to transduce from the positions of facial markers to the activations of facial muscles. We present a markerless, real-time biomechanical face-head-neck simulation system that can automatically learn muscle activation parameters for a variety of expressions from reference images and videos. Most

relevant to our present work is an existing biomechanical model of the cervicocephalic complex (Lee and Terzopoulos, 2006), which incorporates an improved version of the face model developed by Lee et al. (1995).

2.4 Fish Schooling Simulations

The Boids algorithm (Reynolds, 1987) successfully simulated flocks of birds and schools of fish with 3 simple rules for spatial coordination among members of the same species. Tu and Terzopoulos (1994) applied these rules to demonstrate schooling with their biomechanical fish model. The Boids algorithm was further extended by Podila and Zhu (2017b), Stephens et al. (2003), and Podila and Zhu (2017a), who introduced predator-prey relationships to diversify the resulting animation. However, these methods are not suitable for a caged environment because the space is protected by a net; hence, predator-prey interaction cannot be expected. Reynolds (1999) proposed a method for steering behaviors for autonomous characters in computer animation and games, and succeeded in generating various motions autonomously, such as achieving a goal while avoiding obstacles in a Boids simulation. Satoi et al. (2016) proposed a trajectory planning method with a tube for Boids simulation to allow artists to control the animation; however, this was highly artificial, and moreover, it does not scale well when generating many variations. Furthermore, these proposed methods based on the Boids model were missing key ecological behavioral responses which come from temperature, light intensity, and boundary effects from the net and the water surface, which greatly contribute to the fish behavior in caged underwater environments.

2.5 DRL-Driven Swimming Simulation

It is well known that mammals learn via reinforcement (Skinner, 1956). Experiments in neuroscience have demonstrated that dopamine neurons play a role in reinforcement learning (Schultz et al., 1997; Glimcher, 2011; Montague and Berns, 2002). These dopamine

neurons are present not only in mammals but in flies and fish as well (Sun et al., 2018). It has been shown that dopamine neurons are used in cooperative behavior (Messias et al., 2016a) and learning (Messias et al., 2016b) in fish. Therefore, we adopted reinforcement learning for training cooperative schooling behavior.

In the field of fish swimming simulations, DRL has been used to produce efficient swimming and verify adaptation behavior (Verma et al., 2018; Zhu et al., 2021; Ahrens, 2019; Min et al., 2019). These studies focused on simulations of a few fish or soft-bodied animals, but they did not cover large numbers of fish or differences between species. Tu and Terzopoulos (1994) proposed a fish simulation with perception, cognition and muscle based locomotion. Lindsey (1978) defined 12 types of locomotion in fish, and Satoi et al. (2016) incorporated this definition and proposed a controller which automatically selects a type of locomotion based on a desired target behavior. Although those methods worked well at simulating individual fish behaviors, they were not directly useful for our objective of simulating tens of thousands of fish, while forming schooling behavior organically through interactions among the fish.

DRL-based approaches have also been studied for flocking control tasks (Sunehag et al., 2019; Zhu et al., 2020; Yu et al., 2022) and crowd simulation (Lee et al., 2018; Wang et al., 2019). These models do not take biological specifics and their influences on flocking behaviors of simulated objects into consideration, while the adaptation of these species-dependent characteristics constitute a crucial part of our fish control system.

The biological properties we use in our simulation are aggregation, cohesion and separation rules (Breder and Halpern, 1946; Reynolds and Casterlin, 1979); fish's preferred temperature and light intensity (Jensen et al., 1989; Huse and Holm, 1993; Fernö et al., 1995); personal space (Juell, 1995; Føre et al., 2009); sense of depth due to hydrostatic pressure (Macaulay et al., 2020; Davis et al., 2021); response to obstacles (Odling-Smee and Braithwaite, 2003; Brown et al., 2007); and decision making interval (Miller et al., 2017), as well as the social ranking of fish. Fish often form dominance hierarchies, where the dominant fish display aggressive behavior by chasing the subordinate individuals from time to time (Ejike and Schreck, 1980; Sakakura and Tsukamoto, 1998b). In this

dissertation, we set rewards and penalties for DRL based on these properties.

2.6 Fish Biology and Aquatic Environments

Fish species prefer different temperatures and will change their swimming patterns or vertical alignment based on the surrounding water temperature (Reynolds and Casterlin, 1979). Generally, adult salmonids prefer water temperature within the 8-9 degrees Celsius range (Jensen et al., 1989), whereas yellowtail amberjack and red seabream like higher temperatures (Kohbara et al., 2003; Honryo et al., 2018). Caged fish also show distinct swimming patterns at daytime and night. They move very little and stay closer to water surface after sunset, but gradually become more active and swim deeper after sunrise as the amount of sunlight increases (Huse and Holm, 1993; Fernö et al., 1995). Additionally, when fish in the cage reach a certain crowding density threshold, they attempt to maintain their optimal personal space while still being able to move around (Juell, 1995) by forming a circular schooling pattern, consistent with the Boids model (Oppedal et al., 2011). Fish normally react to their neighbors within a distance of 2 to 3 body lengths (BLs) (Føre et al., 2009; Herbert-Read et al., 2011) and can learn to avoid cage walls (Odling-Smee and Braithwaite, 2003; Brown et al., 2007). The reaction appears regulated by visual stimuli (Juell, 1995), as well as the lateral line system, which allows the fish to feel the vibrations of their neighbors (Bleckmann, 2004). Fish align their orientation primarily to the fish in front of them, and adjust speed as necessary to avoid collision with neighbors (Herbert-Read et al., 2011). Caged fish like salmonids often settle into two or three vertical layers (Cubitt et al., 2003) since they typically group with similarly sized fish (Krause et al., 1996), where larger fish tend to swim at lower depths and smaller fish stay at higher depths (Hvas et al., 2021; Folkedal et al., 2012). Moreover, the decision making interval of 95% of fish species is controlled by the Mauthner cells (M-cells) that trigger a rapid escaping movement (C-start) for risk avoidance and command neurons active in decision-making (Wiersma and Ikeda, 1964; Korn and Faber, 2005). Miller et al. (2017) studied zebrafish social groups and found that action potentials for swimming, which

is generated by the neural circuit, lasted for a duration of 50-200 ms. Since salmonids have lower body temperature whereas yellowtail amberjack and red seabream have similar body temperatures as zebrafish, we set 200ms as the the decision-making interval for coho salmon and 100ms as that for yellowtail amberjack and red seabream.

Appendix A.1 presents a more in-depth description of the important biological factors governing the swimming patterns of fish.

Colors underwater are determined by the intrinsic colors of objects and the scattering and absorption of light. The main factors that determine the absorption and scattering are the properties of the water itself and the particulates in the water, usually chlorophyll and sediments (Gallegos and Moore, 2000a). Water absorbs light of different wavelengths at different rates, characterized by attenuation coefficients, which can be changed by the presence of particulates (Bricaud et al., 1998a; Solonenko and Mobley, 2015a). The exact relationship between transmitted light and the light absorbed is expressed by the Beer Lambert law (Gallegos and Moore, 2000a). Chlorophyll absorbs blue light the most, followed by red, then green. Therefore, a higher concentration of chlorophyll shifts the coloring in a reddish green direction. Sediments in the water scatter the light, altering the amount of light absorbed as well as changing the attenuation coefficients. Sediments and suspended solids also create turbidity and occlusion underwater. In our model, sediment concentration is the main driving force behind turbidity.

2.7 Computer Vision Datasets of Underwater Scenes

Only a few datasets are available for computer vision when it comes to underwater scenes. Australian Institute of Marine Science (AIMS) (2019) developed a dataset of 80k images with 45K 2D bounding box annotations, which contained 507 fish species. Deepfish (Saleh et al., 2020) collected a dataset of 40k images with associated classification labels, 3,200 labels for counting, and 310 segmentation masks. Ditria et al. (2021) collected 9,429 images of luderick and annotated them with classification labels. Unfortunately, we could not use these datasets to train our fish counting algorithm for several reasons. Firstly,

they are limited to Australian fish species. They had to be annotated manually; hence the size and the variation of the datasets is limited. Additionally, they all have few fish per image and backgrounds that are very dissimilar to fish farms.

Ditria et al. (2021) and Tarling et al. (2021) applied data augmentation techniques to enhance a fish dataset, and succeeded in increasing the size and the variation of the dataset; however, the annotated data only contain photos of fish taken from the side. To detect and count fish in a fish farm, images from a bottom perspective are also needed, but additional perspectives cannot be created through data augmentation. Furthermore, data augmentation of this sort, where 2D images are overlaid, is not physically accurate and lacks components such as shadows, occlusion, reflections, and depth of field, especially when many overlapping fish are added.

2.8 Synthetic Data Generation

Varol et al. (2017) proposed a synthetic dataset approach for humans (SURREAL), which enabled them to generate more than 6 million images with ground truth labels of pose, depth maps and segmentation masks by superimposing an animated character with different poses taken from motion capture data onto a various 2D scene background images. They proved that a synthetic data approach can be used to increase the size of the dataset which can be utilized for computer vision algorithm training. Hwang et al. (2021) proposed ElderSim which generates synthetic data platform for action recognition. PHAV (de Souza et al., 2017) is a synthetic data generation platform for gaming. Dosovitskiy et al. (2017) proposed a platform named CARLA for autonomous driving systems. CARLA provides 3D CG assets of various actors, vehicles, sensor suites, environmental conditions, and so on. Matthews et al. (2020) generated a synthetic dataset based on the dataset by Ghorbani et al. (2020). Although the effectiveness of using synthetic data in training deep learning models is now well established, such a synthetic dataset does not yet exist for the problem of fish counting.

CHAPTER 3

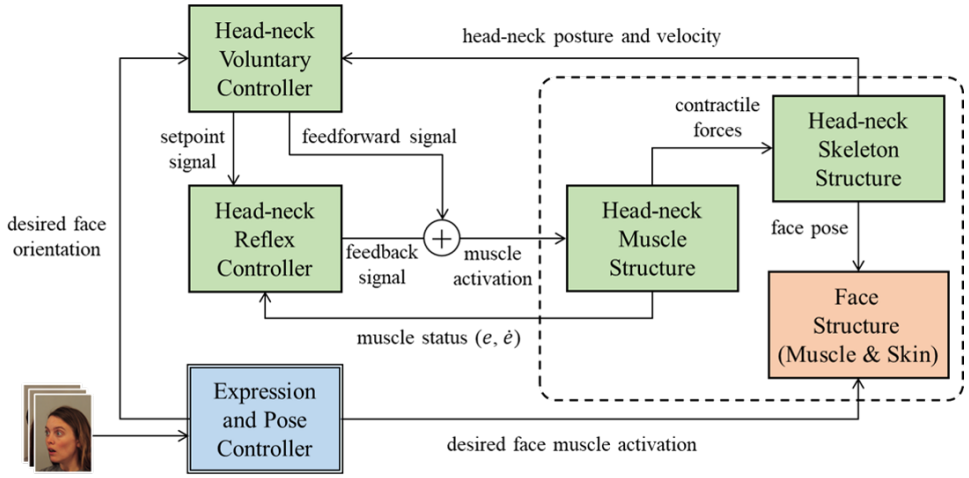
The Face-Head-Neck Model

In this chapter, we present the architecture of our physics-based facial expression and head motion simulation system as well as the two-step pipeline of our neuromuscular controller for the automatic expression and pose transfer task, all of which are illustrated in Figure 3.1. Our controller uses a deep learning approach to learn motions from reference images and videos, and synthesizes them on a face-head-neck biomechanical complex in an anatomically consistent manner. We showcase the experimental result of applying the proposed framework on three face datasets.

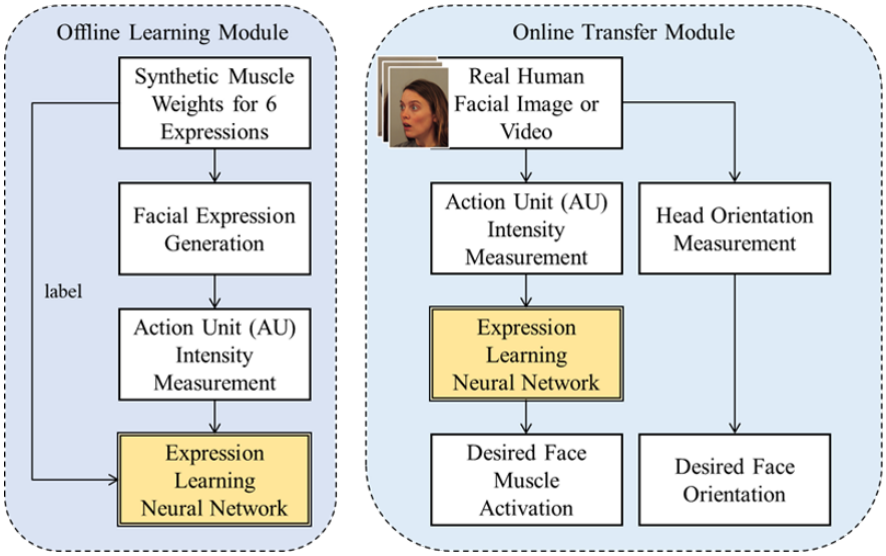
3.1 Musculoskeletal Model

Our real-time musculoskeletal model is based on the one of Lee and Terzopoulos (2006), but both the underlying face-head-neck control system and the facial expression system are significantly improved. Its overall architecture is controlled in a hierarchical manner, as illustrated in Figure 3.1a. Specifically, the skeletal structure is an articulated multibody dynamics system, with bones and joints consistent with human anatomy. The skeletal system is driven by a Hill-type muscle actuator model.

The biomechanical face component consists of a facial soft tissue model comprising epidermis, dermal-fatty, fascia, and muscle layers supported by an underlying skull, which is constructed based on the work of Lee et al. (1995). The soft tissue is a deformable model assembled from discrete uniaxial finite elements, which simulates dynamic facial deformation in an anatomically consistent yet simplified way compared to the models described in (Sifakis et al., 2005), (Wu et al., 2014), and (Cong et al., 2015), thus



(a) Framework



(b) Controller

Figure 3.1: The structure of our expression and head pose transfer framework (a) and the components of the facial expression and head pose controller (b). The expression learning neural network (yellow) is first trained offline. In a transfer task, an image or a video sequence of a real face is fed into the online transfer module of the expression and pose controller to output the desired facial muscle activations and head orientation information. The muscle activations are input to the biomechanical face model (orange) to perform the corresponding expression and the head orientation is provided to the head-neck biomechanical neuromuscular system (green) to produce the desired head pose.

maintaining a low computational cost that affords real-time simulation performance on readily available hardware.

There are 26 pairs of primary facial muscles embedded in the biomechanical face, including the frontalis, corrugator, levator labii, orbicularis oculi, mentalis, and orbicularis oris muscle groups. The contractions of these muscles apply forces to the facial soft tissue layers, inducing deformations that produce meaningful facial expressions. We have augmented the expressive details, such as wrinkles on the face model, by applying multiple levels of subdivision to increase significantly the number of surface nodes that can be activated by muscle forces, and applied a high resolution texture map to the surface mesh for a natural appearance.

The following subsections explain the biomechanical components as well as the controller of our model in greater detail.

3.1.1 Skeleton

The skeletal structure of our model contains 9 links and is modeled as an articulated rigid-body system. The bone links are built on the base link, which is set to be immobile. Among the 8 movable bones, there are seven cervical bones, named C1 to C7, and the skull. The human neck skeleton system contains soft tissue between vertebrae and the cervical spine, which enables motion with 6 degrees of freedom (3 for translation and 3 for rotation) for the bones. We simplify each bone joint to have just 3 rotational degrees of freedom.

The equation of motion of the rigid-body system can be written as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{K}_s\mathbf{q} + \mathbf{K}_d\dot{\mathbf{q}} = \mathbf{P}(\mathbf{q})\mathbf{f}_m + \mathbf{J}(\mathbf{q})^{\mathsf{T}}\mathbf{f}_{\text{ext}}, \tag{3.1}$$

where $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ are 24-dimensional vectors representing the joint angles, the angular velocities, and the angular accelerations, respectively. $\mathbf{M}$ is the mass matrix, and $\mathbf{C}$ represents the internal forces among the system, including Coriolis forces, gravity, and

forces from connecting tissues. The moment arm matrix $\mathbf{P}$ maps the muscle force $\mathbf{f}_m$ (contractile muscle force $\mathbf{f}_C$ and passive muscle force $\mathbf{f}_P$) to the related joint torques, while the Jacobian matrix $\mathbf{J}$ transforms the applied external force $\mathbf{f}_{\text{ext}}$ to torques. The $\mathbf{K}_s\mathbf{q} + \mathbf{K}_d\dot{\mathbf{q}}$ term represents the rotational damping springs that we attach to the joints in order to simulate the stiffness of the inter-vertebral discs. We can alternatively write the torque from the spring as:

$$\tau_s = -k_s(q - q_0) - k_d\dot{q}, \tag{3.2}$$

where q is the current joint angle in the generalized coordinates, and q_0 is the joint angle in the natural pose (resting angle). The k_s and k_d are the stiffness and damping coefficients of the spring, respectively.

3.1.2 Neck Muscles

We use a modified version of the Hill-type muscle model (Ng-Thow-Hing, 2001), which is a good balance of biomechanical accuracy and computational efficiency, to simulate a total of 72 neck muscles. The muscle force $f_m = f_P + f_C$ has two sources. The passive element f_P generates a restoring force due to the muscle elasticity, which constrains the muscle deformation passively. The passive muscle force is represented as

$$f_P = \max(0, k_s(\exp(k_c e) - 1) + k_d\dot{e}), \tag{3.3}$$

where k_s and k_d are the stiffness and damping coefficients of the above uni-axial exponential spring model. e is the strain of the muscle and $\dot{e}$ is the strain rate. We can calculate them using $e = (l - l_0)/l_0$ and $\dot{e} = \dot{l}/l_0$, respectively, where l and l_0 are the muscle length and muscle resting length.

The contractile element f_C generates the proactive contractile force of the muscle,

which is proportionate to the *activation level* of the muscle:

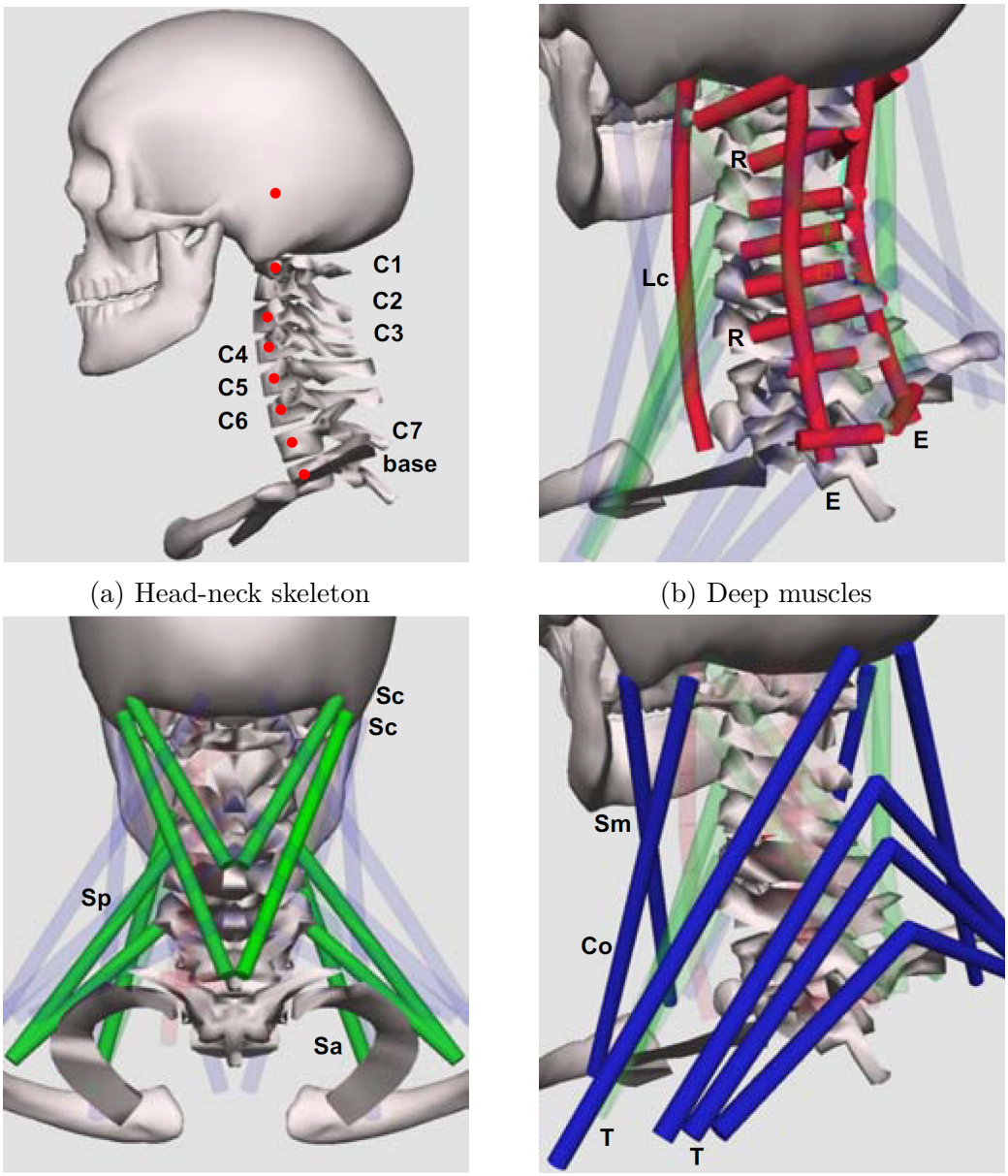
$$\begin{aligned} f_C &= aF_l(l)F_v(\dot{l}), \\ F_l(l) &= \max(0, k_{max}(l - l_m)), \\ F_v(\dot{l}) &= \max(0, 1 + \min(\dot{l}, 0)/v_m), \end{aligned} \tag{3.4}$$

where $a \in [0, 1]$ is the muscle activation level, F_l is the force-length relation, and F_v is the force-velocity relation. We present the plots of the two relations in Figure 3.3. During their calculation, we need the maximum muscle stiffness k_{max}, the maximum muscle length l_m, and the maximum contractile velocity v_m. l_m is valued at $0.5l_0$, v_m is valued at $8l_0$ sec^{-1}, and k_c is set to 7 for all neck muscles. The remaining coefficients k_s, k_d and k_{max} are set to be proportional to the strength weight factor of each muscle, which is computed as roughly proportional to the cross sectional area of the muscle. A list of strength weight factors as well as additional details about the setting can be found in (Lee and Terzopoulos, 2006).

Figure 3.2 illustrates the skeleton and the 72 neck muscles, which are arranged in three layers—deep, intermediate, and superficial. The 48 muscles in the deep layer improve the controllability and ensure the freedom to model the 12 muscles in the intermediate layer and another 12 muscles in the superficial layers. Although the number of simulated neck muscles greatly outnumbers the total degrees of freedom of the head-neck system, such muscular redundancy is a necessity of the natural and controllable head movement.

3.1.3 Facial Tissue

Our soft tissue model is automatically generated using the technique described in (Lee et al., 1995). After laser-scanning the individual's facial data, a range image and a reflectance image in cylindrical coordinates are adapted to a well-structured face mesh. Then the algorithm creates a dynamic skin and muscle model for the face mesh, which contains automatically generated facial soft tissues, on top of an estimated skull surface. Also, major muscles responsible for facial expression are inserted to the model. Each of



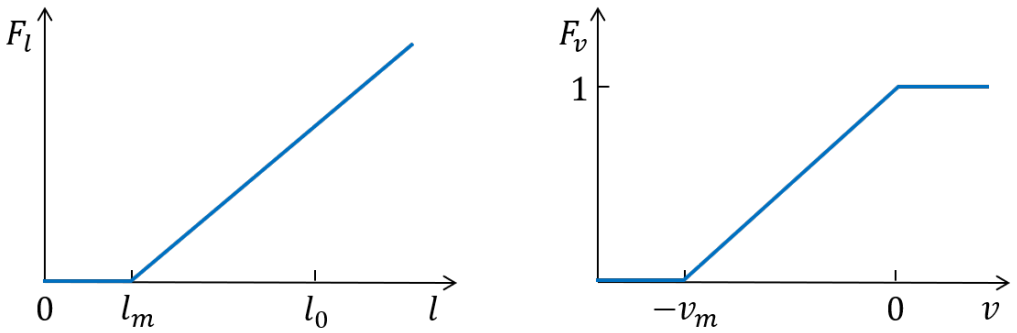
(a) Head-neck skeleton

(b) Deep muscles

(c) Intermediate muscles

(d) Superficial muscles

Figure 3.2: Neck musculoskeletal model (from (Lee and Terzopoulos, 2006)). (a) The red dots are the pivots of the eight joints of the cervical spine and the skull. (b) 48 deep muscles with six muscles attached across each cervical joint. (c) 12 intermediate muscles. (d) 12 superficial muscles.



(a) Force-length relationship (b) Force-velocity relationship

Figure 3.3: The force relationships of Hill type muscle model.

these muscles is attached to the skin on the one end through elastic tendons at many points in the fascia layer, but is fixed to the skeleton surface on the other end.

The physical simulation of the muscle-actuated facial skin model is implemented as a discrete deformable model (DDM), where a network of fascia nodes are connected using uni-axial springs. The force exerts from spring j on node i can be written as

$$\mathbf{g}_i^j = c_j(l_j - l_j^r)\mathbf{s}_j, \tag{3.5}$$

where l_j and l_j^r are the current and resting length of spring j, and $\mathbf{s}_j = (\mathbf{x}_j - \mathbf{x}_i)/l_j$ is the spring direction vector.

The facial skin model is also actuated by the underlying muscles. We calculate the force exerts from muscle j on node i according to the length scaling function Θ_1 and the muscle-width scaling function Θ_2 as follows:

$$f_i^j = \Theta_1(\varepsilon_{j,i})\Theta_2(\omega_{j,i})\mathbf{m}_j. \tag{3.6}$$

where $\mathbf{m}_j$ is the normalized muscle vector for muscle j. In (Lee et al., 1995), the plots of the two scaling functions, as well as the definition of the length ratio ε and muscle width ω are explained in detail. Moreover, there are other aspects of the generated discrete deformable model; e.g., the volume preservation forces and skull penetration constraint forces. The Explicit Euler method is used to compute the nodal acceleration, velocity

and position at each time step.

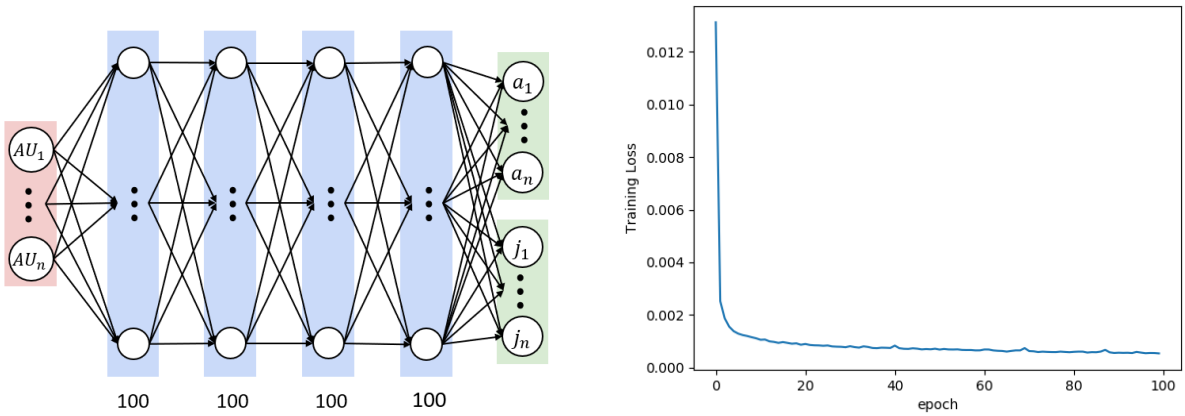
3.1.4 Control

To control the face-head-neck system, our novel neural network-based expression and pose controller (Figure 3.1) generates facial muscle activations that produce recognizable expressions. It concurrently outputs head pose estimates to a pretrained controller of the head-neck musculoskeletal complex, where voluntary and reflex neuromuscular control layers generate muscle activation signals to achieve the desired head orientations.

The higher-level voluntary controller receives the current head-neck posture and velocity information, as well as the desired adjustment of the posture, and generates a feedforward muscle activation signal and a setpoint control signal. The latter, which encodes the desired muscle strains and strain rates, is input to the reflex controller that then generates a feedback muscle activation signal and adds it to the feedforward signal generated by the voluntary controller. As a result, each cervical muscle receives an activation signal a and, through simulation, generates a contractile muscle force accordingly. Together with the external environmental forces and gravity, the whole system is simulated through time and rendered as a physics-based animation. The voluntary controller runs at 25Hz (in simulation time), while the reflex controller runs at 4KHz along with the physical simulation time steps. Refer to (Lee and Terzopoulos, 2006) for more details of the head-neck neuromuscular controller.

3.2 Expression Learning

We next explain our machine learning approach of using a deep neural network to transfer facial expressions to our biomechanical face model. To achieve automatic synthesis of muscle activations and resulting facial expressions using the trained deep neural network, instead of using motion capture data, which can be expensive to collect, we leverage the Facial Action Coding System (FACS). The following subsections describe the architecture of the network, the pipeline for using the biomechanical face model to generate training



(a) Neural network architecture

(b) Convergence of the neural network

Figure 3.4: The expression learning neural network.

data, and the process of training the neural network.

3.2.1 Network Architecture

The function of the neural network is to generate activation signals for the muscles of the biomechanical face model such that they generate contractile forces that deform the synthetic facial tissues to produce facial expressions. We employ a fully connected deep neural network architecture for this purpose (Figure 3.4a). The input layer consists of a total of 17 neurons representing the important AUs that are involved in the majority of facial movements, each neuron corresponding to a single normalized AU. We include 4 hidden layers with 100 neurons in each layer and ReLU activation, a choice based on the Mean Squared Errors (MSE) on the test data of networks with different numbers of hidden layers and different numbers of neurons per hidden layer The output layer consists of 56 neurons, 52 of which encode the activations a_i, with $1 \leq i \leq 52$, for each of the 26 pairs of facial muscles, and the remaining 4 encode the jaw rotation, jaw slide, jaw twist, and an auxiliary value. Given its architecture, the network has a total of 37,300 weights. It is implemented in Keras with a TensorFlow backend.

3.2.2 Training Data Generation

The training data generation process is divided into two steps:

1. generation of muscle activations, and
2. generation of AUs for the corresponding muscle activations.

In the first step, each basic expression requires a combination of muscles to be activated. Given n muscles, we define W_e as a set of weights w_i, with $1 \leq i \leq n$, which determine the effect each muscle will have on an expression e. The activation for each muscle a_i for an expression is then defined as $a_i = w_i s$, where $w_i \in W_e$ and s is a scale term. For a single expression, we determine the weights in a set W_e manually by visually analyzing the facial expressions formed by the face model. We repeat the process for all the basic expressions, namely (1) joy, (2) sadness, (3) anger, (4) fear, (5) disgust, and (6) surprise to generate the sets W_{Joy}, W_{Sadness}, W_{Anger}, W_{Fear}, W_{Disgust} and W_{Surprise}, respectively.

We then sample the value of s randomly in the range $[0.0, 1.0]$ to generate all the muscle activations a_i. We also assign a random value between 0 and 1 for the jaw rotation. For the purpose of our experiments, we maintain the jaw twist and jaw slide at a value of 0.5 to keep the jaw position centered. We further form a set A consisting of all the muscle activations a_i, where $1 \leq i \leq n$, and the jaw activations including jaw rotation, jaw twist and jaw slide. We iterate the above process, generating the set A by repeatedly sampling the value of s for a single expression. We finally extend this to all basic expressions and obtain multiple sets A_{Joy}, A_{Sadness}, A_{Anger}, A_{Fear}, A_{Disgust} and A_{Surprise} for each expression.

In the second step, we leverage the functionality of OpenFace 2.0 (Baltrusaitis et al., 2018) for facial expression recognition and head pose estimation. OpenFace is a computer vision tool capable of facial landmark detection, head pose estimation, facial AU recognition, and eye-gaze estimation. OpenFace employs Convolutional Experts Constrained Local Models (CE-CLMs) for facial landmark detection and tracking. CE-CLMs use a 3D representation of the detected facial landmarks by which OpenFace estimates the head pose. Eye gaze estimation is done using a Constrained Local Neural Fields (CLNF)

landmark detector. The final task, facial expression recognition, is performed using Support Vector Machines and Support Vector Regression.

We use a single set A formed by the above described procedure to activate the muscles and jaw of the biomechanical face model. We then render the model to form an image. The image is input to OpenFace, which performs facial expression recognition and head pose estimation, outputting the estimated AUs and head orientation associated with the input image. We repeat this process for each set A formed (as described previously) to obtain the corresponding AUs and head orientations.

Thus, we synthesize a large quantity of training data pairs each consisting of (i) muscle and jaw activations A and (ii) the associated AUs and head orientations.

3.2.3 Network Training

We use the aforementioned data to train our deep neural network to input AUs and output corresponding muscle and jaw activations. This training process takes place offline before the actual expression transfer.

The AUs from each training pair, generated as described in the previous section, are passed to the network as input. We then compare the corresponding muscle and jaw activations; i.e., the ground truth compared to the predictions of muscle and jaw activations given by the neural network. We use a Mean Square Error (MSE) training loss between the predictions and the ground truth, which is backpropagated to update the weights, thus training the neural network.

We normalize the intensity values of each AU class across all pairs to remove the disparity of intensity values between synthetic faces and real faces. We use a total of 6,000 pairs, with about 1,000 pairs for each basic expression.

To train the neural network, we use the ADAM stochastic gradient descent optimizer with an MSE loss, a batch size of 32, and a learning rate of 0.01. We train the network in a total of 100 epochs, running on an NVIDIA GeForce GTX 1650 GPU installed on a Windows 10 machine with a 2.6GHz Intel Core i7-9750H CPU. Figure 3.4b shows the

convergence of the training error.

3.2.4 Expression Transfer Pipeline

To transfer real facial expressions on the fly, we use a pipeline similar to the offline training module, again leveraging OpenFace for facial expression recognition and head pose estimation. We input an image of an expressive face into OpenFace to obtain all of the corresponding AUs and head orientations. The AUs are then passed into the trained neural network which outputs predictions of the muscle and jaw activations, driving the biomechanical face to deform the muscles and transfer the expressions onto it.

We transfer both image and video inputs. Each frame in a video is processed independently and a resulting video is created using the transferred frames. The intensity values for each AU class are normalized across all the images or frames as in the case of the training pipeline.

3.3 Experiments and Results

We next present the results of transferring facial expressions from the wild using our trained neural network. We evaluate our expression transfer pipeline on different expressions while using a variation of AUs and muscles in the biomechanical face model.

3.3.1 Facial Expression Datasets

Several facial expression datasets are available online. The datasets that we used in our experiments are as follows:

Karolinska Directed Emotional Faces (KDEF) (Calvo and Lundqvist, 2008). The KDEF dataset consists of 4,900 pictures of human facial expressions. It covers 70 subjects (35 female and 35 male) enacting all 6 basic facial expressions, namely neutral, joy, sadness, anger, fear, disgust, and surprise. Each expression performed by the subject is imaged from multiple directions. We use the dataset to transfer facial expressions

onto the biomechanical face model and both qualitatively and quantitatively analyze the performance of our trained neural network in this dissertation.

Cohn Kanade Dataset (CK) and *Extended Cohn Kanade Dataset (CK+)* (Kanade et al., 2000; Lucey et al., 2010). The CK and the CK+ dataset combined consist of 593 video sequences of 123 subjects. Each sequence consists of images from a neutral expression (first frame) to a peak expression (last frame). The peak expressions are FACS coded for AUs and cover 7 emotions, anger, contempt, disgust, fear, happiness, sadness, and surprise. We use the sequences in the CK+ dataset to transfer videos of expression transitions onto the biomechanical face.

Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) (Livingstone and Russo, 2018). The original RAVDESS dataset consists of 24 actors vocalizing two lexically-matched statements. An extension of the dataset, named *RAVDESS Facial Landmark Tracking*, contains tracked facial landmark movements from the original RAVDESS datasets (Swanson et al., 2019). Each actor performs 60 speech trials and about 44 song trials with 8 emotions (neutral, calm, happy, sad, angry, fear, disgust and surprise) and 2 intensities (normal and strong). This yields a total of 2,452 video files for the complete dataset. We leverage this dataset to test the transfer of actor faces in speech videos onto the biomechanical face.

3.3.2 Action Units and Muscle Activations

There exist a total of 30 Action Units (AUs) corresponding to facial expressions. OpenFace provides occurrence predictions for 18 out the 30 AUs and measures intensity values for 17 out of the 30 AUs. We consider the 17 AUs for which the intensity values are present as the super-set of the AUs for our use case.

Due to the correlation between the AUs and muscles in the face, there also exists a correlation between a basic facial expression and the AUs activated by it. Our initial experiments focused on training the neural network for each expression in an isolated manner. The neural network was trained to output muscle activation for muscles corre-

Table 3.1: Comparison between the Mean Squared Errors of training individual neural networks for each expression and of training a single neural network for all expressions.

MSE_{Joy}	$MSE_{Sadness}$	MSE_{Fear}	MSE_{Anger}	$MSE_{Surprise}$	$MSE_{Disgust}$	MSE_{All}
0.000591	0.002925	0.001611	0.009689	0.000266	0.002516	0.000729

sponding to a single expression using AUs which pertained to the same expression. In further trials, we observed that usage of all 17 AUs and all facial muscles improved the performance and the scalability of the expression transfer pipeline.

Table. 3.1 provides a comparison between training a single neural network for all expressions and training individual neural networks for each expression, where each network is trained only from the AUs and muscles relevant to its expression (using the mappings of expressions to AUs and AUs to muscles presented in (Du et al., 2014; Clark et al., 2020), (Clark et al., 2020) and (Scherer et al., 2019)). We observe better performance when training a single neural network for all the expressions, suggesting that AUs not directly relevant to an expression also play a role in expression transfer.

Due to limitations in the biomechanical model, the range of each AU class differs from that of the real faces. Hence, we use normalization to overcome the bias and better transfer real facial expressions onto the biomechanical model. Figure 3.5 shows the expression intensities in transferred expressions with normalization better represent the real faces than those without normalization.

We choose to activate only jaw rotation so as to maintain the symmetry of the expression for our use case. We observe, that without jaw activations, expressions such as surprise are not well synthesized by the biomechanical face model (Figure 3.6).

3.3.3 Head Orientation

Leveraging OpenFace for head pose estimation, we pass the estimated orientation of the head into the trained neck controller to activate the neck muscles. This in turn actuates the neck to adjust the head orientation in accordance with the input image. Figure 3.7 presents sample transferred results including head orientations from the KDEF dataset.

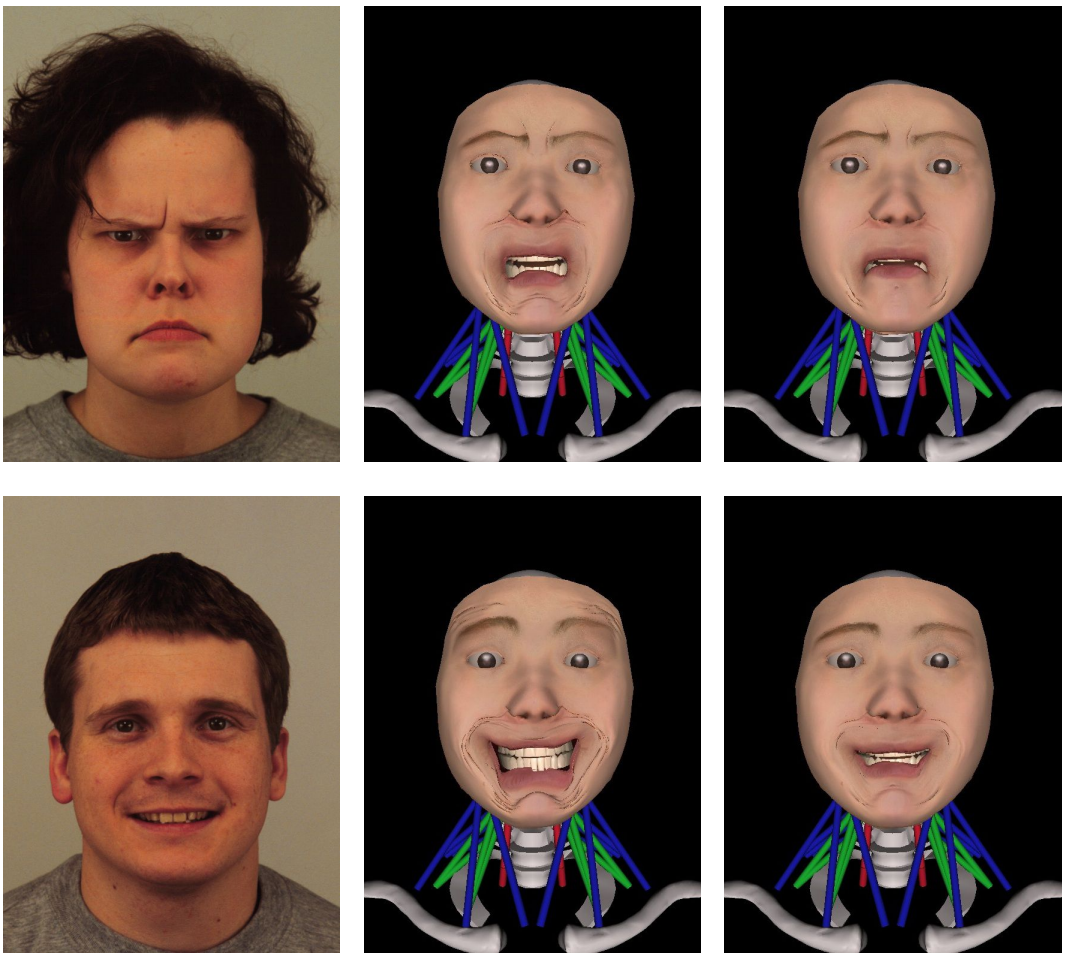


Figure 3.5: Comparison of transfer results with and without AU normalization. The three columns correspond to (i) the original image from the KDEF dataset, (ii) result without normalization of AUs, and (iii) result with normalization of AUs.

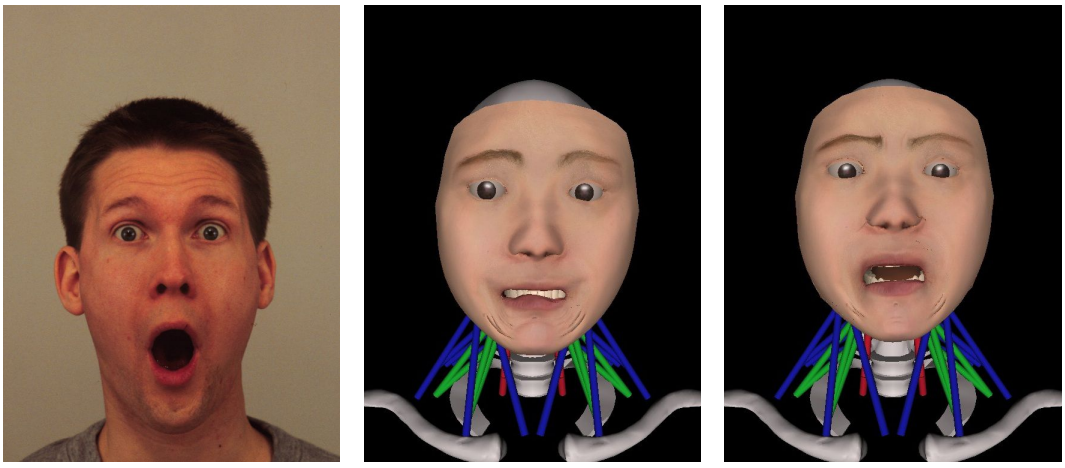
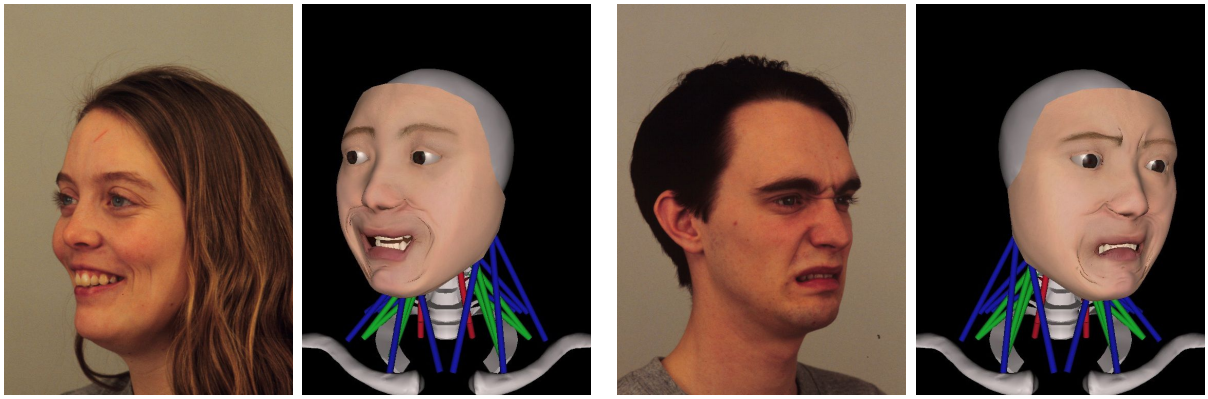


Figure 3.6: omparison of transfer results with and without jaw activations. The three images are (i) the original image from the KDEF dataset, (ii) the transferred face without any jaw activations, and (iii) the transferred face with jaw activations.



(a) Subject 1 (b) Subject 3

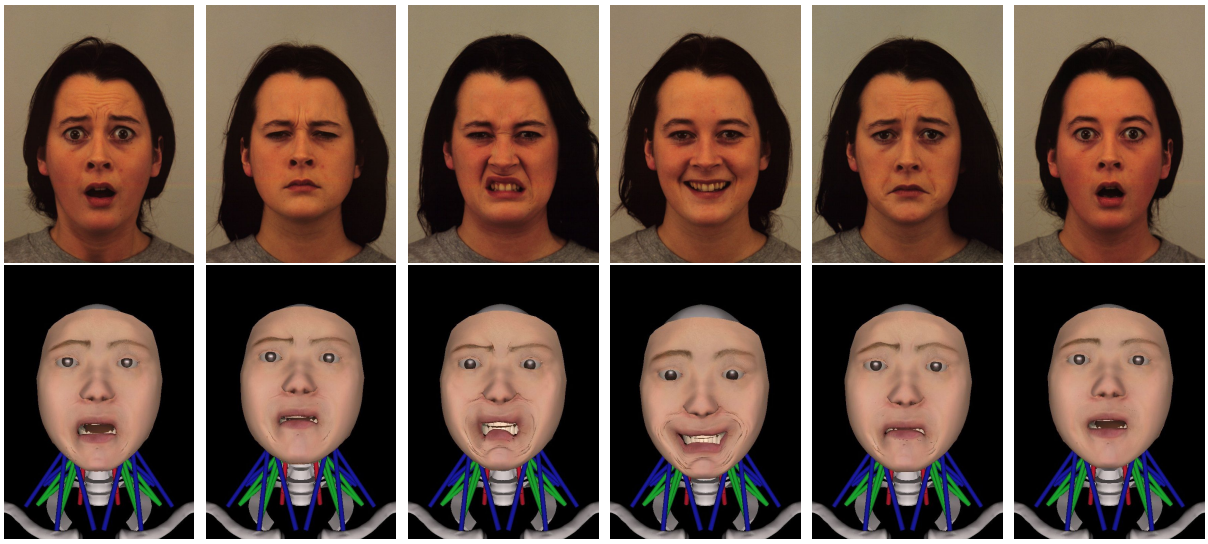
Figure 3.7: Transfer of head orientation along with facial expression.

3.3.4 Facial Expression Transfer

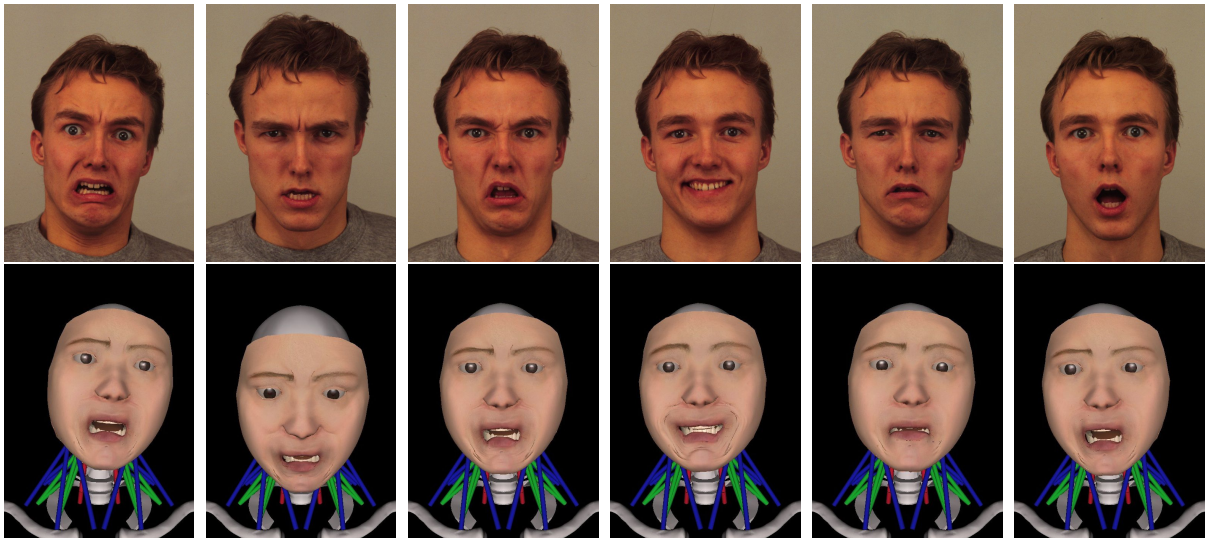
The offline training of the expression and head pose controller took less than a day to complete. The generation of 6,000 muscle activations was instant once the basic weights for each expression (i.e. W_{Joy}, W_{Sadness}, W_{Anger}, W_{Fear}, W_{Disgust} and W_{Surprise}) were chosen. The generation of AUs took the majority of time, which was roughly 17 hours, since we waited 10 seconds for the biomechanical face to stabilize the expression given each of the 6,000 activation sets before capturing the rendered scene to feed into OpenFace. The training of the network took less than 5 minutes to complete for 100 epochs.

The results for the facial expression transfer for each of the expressions is shown in Figure 3.8. We present transfer results of a small sample of the KDEF dataset, selecting two subjects (a female and a male) enacting all the basic expressions. It can be seen that our framework correctly transfers both the expression type and the head orientation. For the same expression performed by different subjects, the results reproduce the difference in head pose and emotion intensity. Additionally, for two similar expressions such as fear and surprise enacted by the same subject, the results can be differentiated by the movements of facial features like eyebrows and wrinkles.

We also evaluate the transfer results by comparing the MSE of selected AUs in Table 3.2. We calculate the average MSE by AUs over all transferred expressions and their corresponding reference data, then compute the mean value of the average MSEs



(a) Subject 4



(b) Subject 5

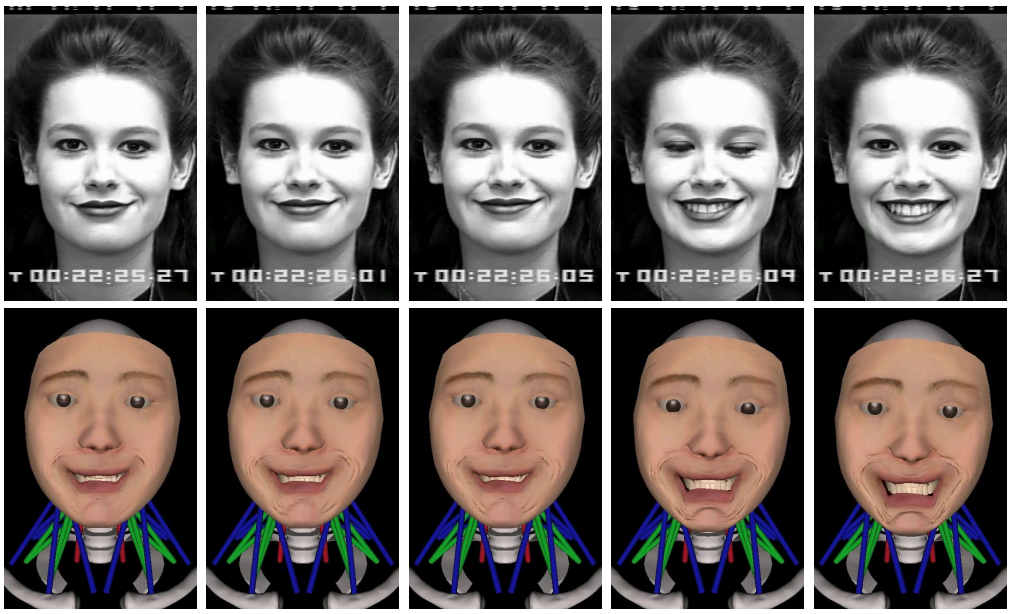
Figure 3.8: Transfer of facial expressions (fear, anger, disgust, joy, sadness, surprise) and head poses from KDEF images to the biomechanical face-head-neck model.

Table 3.2: Average MSE of selected AUs in the original data and transfer results using different settings. Note that we randomly select 8 out of 17 AUs output by OpenFace. The last column is the average MSE over all the AUs in each setting.

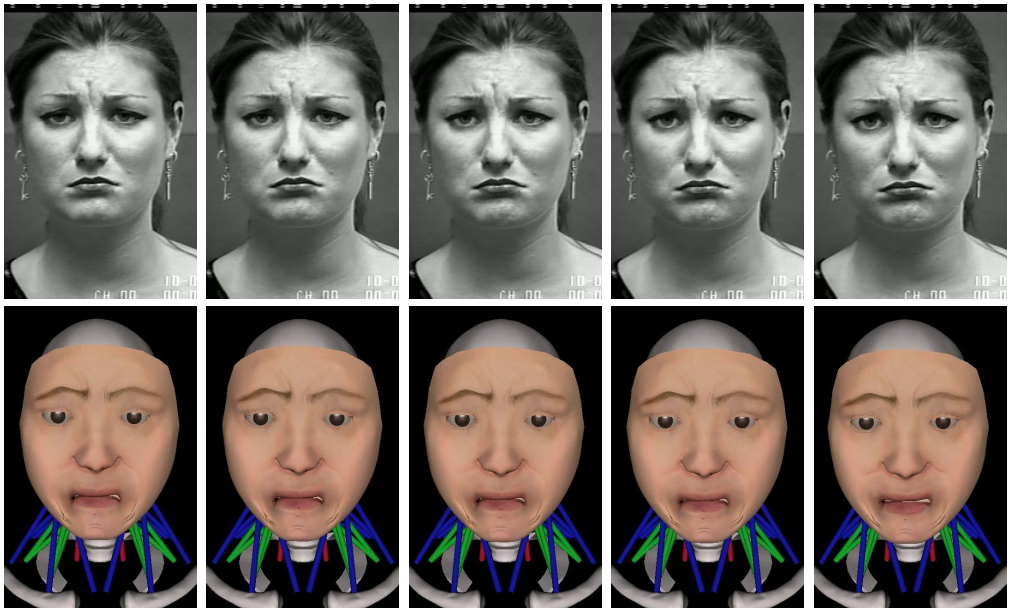
Dataset	MSE_{AU1}	MSE_{AU4}	MSE_{AU6}	MSE_{AU9}	MSE_{AU12}	MSE_{AU15}	MSE_{AU20}	MSE_{AU25}	Average
KDEF (unnormalized)	0.191	0.088	0.186	0.133	0.011	0.155	0.129	0.092	0.157
KDEF (normalized)	0.118	0.064	0.186	0.068	0.028	0.109	0.146	0.052	0.129
RAVDESS (video)	0.012	0.222	0.107	0.044	0.083	0.016	0.077	0.036	0.086
CK (video)	0.038	0.054	0.059	0.015	0.008	0.023	0.057	0.034	0.048

over all AUs in each experimental setting. We report such mean values together with 8 randomly chosen AUs. The normalization step decreases the MSEs of most selected AUs and yields results with higher AU similarity.

Figure 3.9 and Figure 3.10 show selected frames of the transfer of expressions from videos in the CK+ dataset and RAVDESS dataset, respectively. The transfer results successfully reproduce the changes in emotion intensity in addition to the type of emotions. The head orientation and jaw movement are well synthesized and match those in reference frames. Furthermore, our simulation captures the subtle expressions performed by subjects. The larger number of subtle expressions enacted by subjects in the RAVDESS and CK+ videos also results in lower average MSEs compared with those for the KDEF image dataset as shown in Table 3.2.

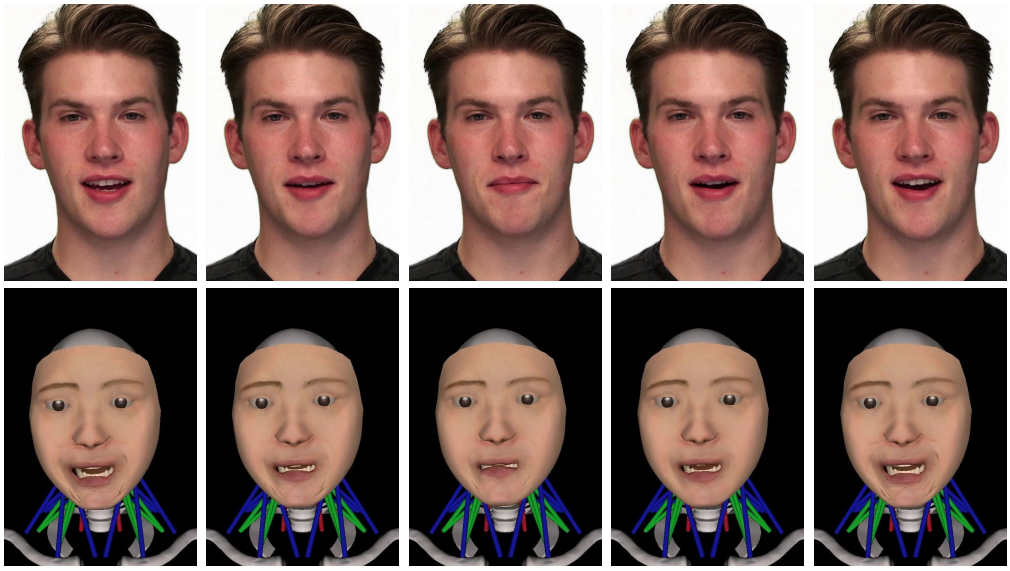


(a) Joy

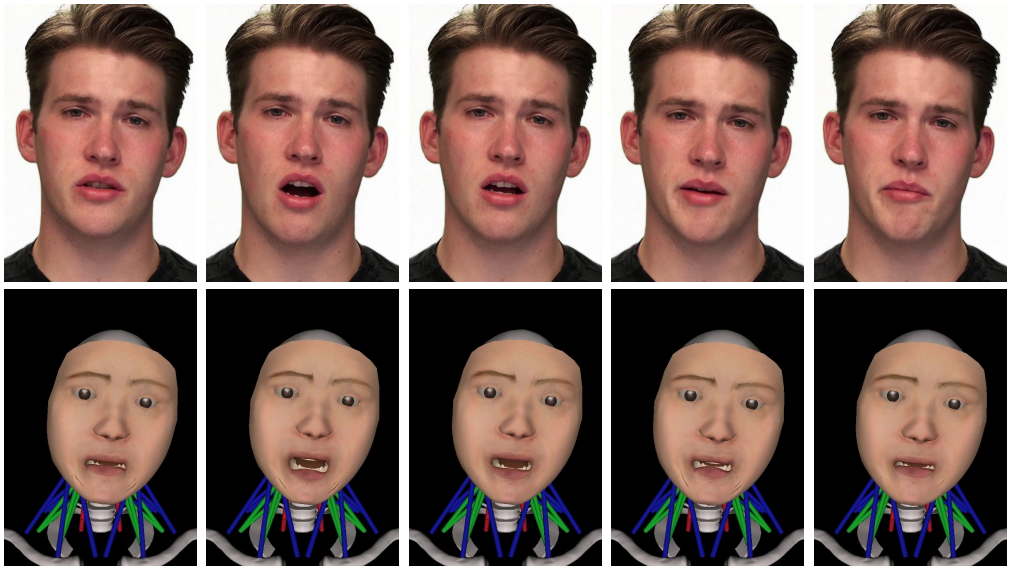


(b) Sadness

Figure 3.9: Facial expression and head pose transfer from video image sequences from the CK+ dataset of two subjects expressing joy (a) and sadness (b), respectively.



(a) Joy



(b) Sadness

Figure 3.10: Facial expression and head pose transfer from a video image sequence from the RAVDESS dataset of a talking subject performing joy (a) and sad (b) speech.

CHAPTER 4

The DeepFoids Model

In this chapter, we explain our Deep Reinforcement Learning (DRL) driven adaptive bio-inspired fish simulation system. The following sections start with an in-depth description of the components in the fish simulation pipeline and the design of the DRL environment. This is followed by the details of the DRL control module for fish behavior and the synthetic dataset generation. Then we introduce a computer vision system that, using the generated data, is trained for the fish counting task. Finally, the implementation and results of a variety of experiments are presented.

4.1 Fish Simulation

In this section, we introduce a bio-inspired fish simulation trained with DRL. We structure the behavioral model of fish as a multi-agent reinforcement learning problem, where fish agents need to navigate around a constrained 3D space that resembles a fish farm environment while reacting to multiple factors that simultaneously influence their behavior. During the course of interacting with the environment and other fish, each agent acts based on its own observations and is responsible for learning a general policy that maximizes a reward. We model the policies using neural networks and optimize them using Proximal Policy Optimization (PPO) (Schulman et al., 2017) with Generalized Advantage Estimation (GAE) (Schulman et al., 2015b), which will be described in the next section.

A schematic illustration of our fish simulation pipeline is provided in Figure 4.1. The parameters controlling biological and environmental factors in the fish simulation are set

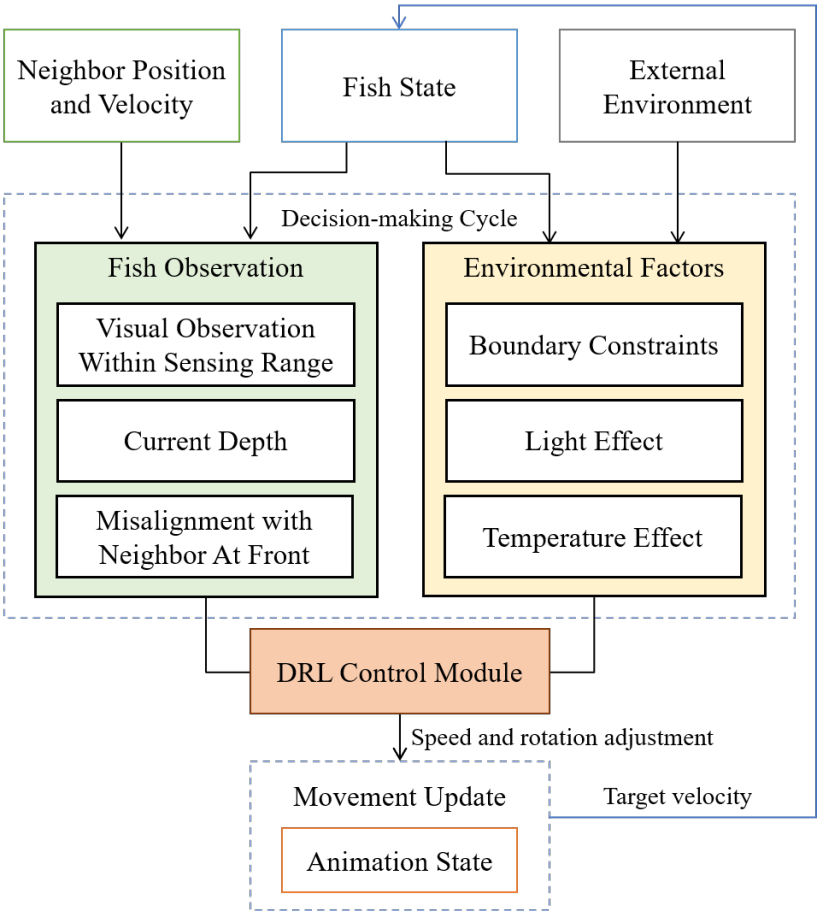


Figure 4.1: Structure of the DeepFoids DRL-based fish simulation framework.

based on the literature presented in Section 2.6 and field data collected from fish farms at Onmaehama and Nishiki in Japan. Fish animation is generated by the finite state machine based on the velocity of the locomotion and the fish's behavior state.

4.1.1 Biological and Environmental Factors

We incorporate important biological and environmental drivers of fish swimming patterns into the simulation framework. A fish stays within a comfortable distance of its neighboring fish and aligns its direction with that of its nearest neighbors in the forward direction (Reynolds, 1987; Herbert-Read et al., 2011). Its activity space is constrained by the cage boundaries and water surface, which reproduce the commercial cage environment and facilitate the formation of schooling. Additionally, caged fish are divided into dominant

and subordinate groups, where the dominant individuals may initiate antagonistic acts by aggressively approaching the subordinate members. These factors are carefully integrated into the process of learning control policies that generate the delta velocity $\Delta \mathbf{v}_t^f$ of fish at each time step t through the use of the PPO algorithm.

There are effects of underwater light intensity and temperature on the fish's vertical distribution, which we represent as $\Delta \mathbf{v}_{\text{light}}$ and $\Delta \mathbf{v}_{\text{temp}}$. A fish prefers to stay within a certain area of the tank where its light intensity preference matches the local light intensity computed by the Beer-Lambert law:

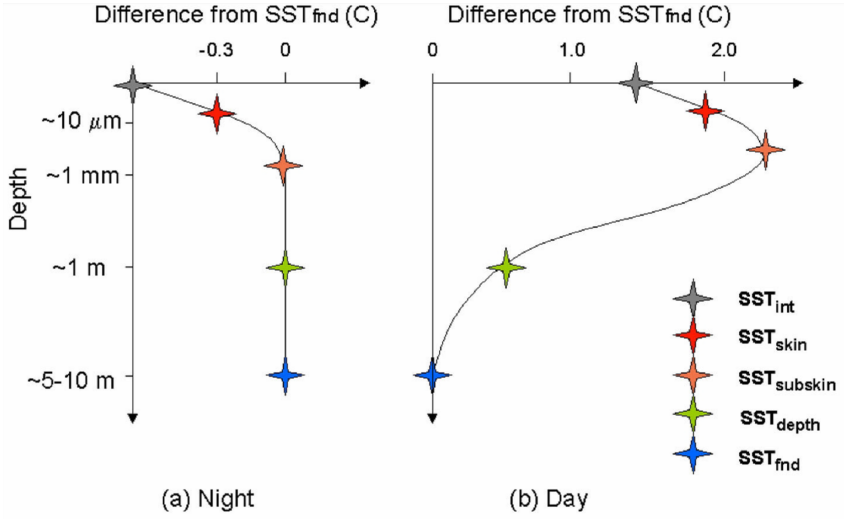
$$I = I_0 \times e^{-ad}, \tag{4.1}$$

where I and I_0 denote the light intensity at the local depth and on the water surface, a is an attenuation coefficient and d is the depth of the fish. I_0 and α are valued based on the field data collected from aquaculture sites as described in Appendix A.4. Any deviation of I from the light intensity preference range results in a vertical delta velocity

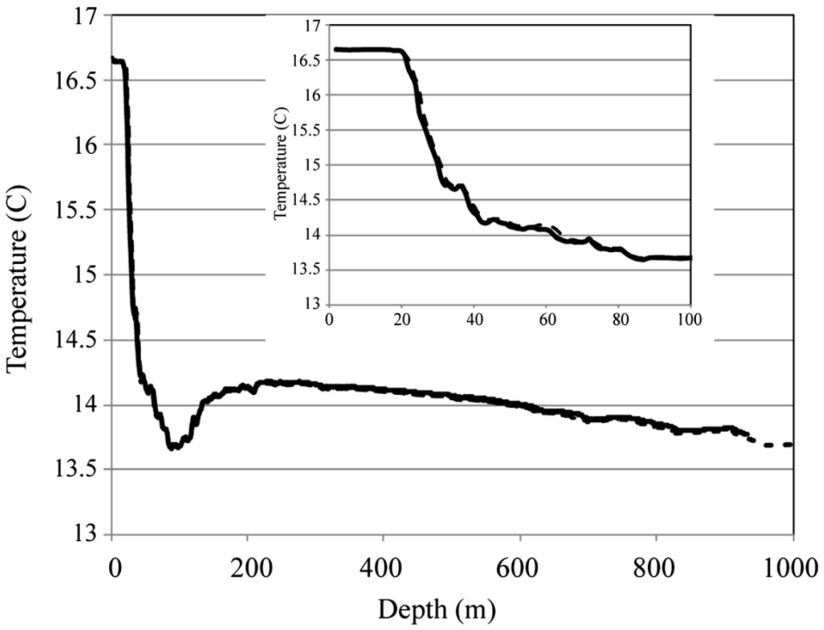
$$\Delta \mathbf{v}_{\text{light}} = \begin{cases} \frac{I_{\text{lpref}} - I}{I_{\text{lpref}} - I_{\text{lsteep}}} \, [0, 1, 0] & \text{if} \quad I \leq I_{\text{lpref}}, \\ \frac{I - I_{\text{hpref}}}{I_{\text{hsteep}} - I_{\text{hpref}}} \, [0, -1, 0] & \text{if} \quad I \geq I_{\text{hpref}}, \end{cases} \tag{4.2}$$

where parameters I_{lsteep} and I_{hsteep} control the steepness of the reaction to light and are set in an identical manner to (Føre et al., 2009). I_{lpref} and I_{hpref} denote the lower and upper bounds of the light preference interval and are valued based on the biological studies of each simulated species (Stuart and Drawbridge, 2011; Kohbara et al., 2003; Honryo et al., 2018; Huse and Holm, 1993).

Similarly, a fish likes to stay in the area where the local temperature satisfies its temperature preference. We use a two-layer method to obtain T based on the studies by Abraham et al. (2013) and Kawai and Wada (2007). Figure 4.2 illustrates the temperature changes in the near-surface and deeper layers of the ocean. Note that in the near-surface layer (Figure 4.2a), the water temperature drops approximately 1° C from the water



(a) Near-surface ocean temperature



(b) Deeper ocean temperature

Figure 4.2: The ocean temperature model in the DeepFoids framework consists of a near-surface layer and a deeper water layer. (a) Temperature differences between near-surface sub-layers (grey, red, orange, and green star symbols) and deeper ocean (blue star symbol) at daytime and night are reproduced from (Kawai and Wada, 2007). (b) Deeper ocean temperature are measured by expendable bathythermograph (dashed line) and conductivity-temperature-depth (solid line) instruments are reproduced from (Abraham et al., 2013).

surface (the grey star symbol) to the sub-layer at a depth of 1 meter (the green star symbol). In the deeper water layer (Figure 4.2b), the water temperature remains constant at depths between 1 to 20 meters. Therefore, given a water surface temperature T_{surf}, we linearly decrease T_{surf} by 1° C from depths of 0 to 1 meters to compute T, and keep T constant below 1 meter as the sea cage is only 6 meters deep in our simulation. We then use the obtained T to calculate the temperature induced vertical delta velocity

$$\Delta \mathbf{v}_{\text{temp}} = \begin{cases} \frac{T_{\text{lpref}} - T}{T_{\text{lpref}} - T_{\text{lsteep}}} [0, 1, 0] & \text{if} \quad T \leq T_{\text{lpref}}, \\ \frac{T - T_{\text{hpref}}}{T_{\text{hsteep}} - T_{\text{hpref}}} [0, -1, 0] & \text{if} \quad T \geq T_{\text{hpref}}, \end{cases} \tag{4.3}$$

where the temperature response steepness parameters T_{lsteep} and T_{hsteep} are set to be the same as those in (Føre et al., 2009), and the bounds of the temperature preference interval T_{lpref} and T_{hpref} are determined according to Kohbara et al. (2003), Honryo et al. (2018) and Jensen et al. (1989).

In addition to the above-mentioned components, we integrate the fish decision making interval into the framework to emulate the latency of fish's responses to environmental changes. The duration (in units of simulation steps) of the decision making interval Δt_{res} for simulated species is predefined in accordance with the literature studies described in Section 2.6. At the time of simulation, given the time interval of simulation steps, Δt_{sim}, a fish updates its observation of the environment every $\lfloor \Delta t_{\text{res}} / \Delta t_{\text{sim}} \rfloor$ steps and takes actions between updates. The delta accumulated velocity to apply at each simulation step can then be derived as

$$\Delta \mathbf{v}_t^a = \Delta \mathbf{v}_t^f + \frac{\Delta \mathbf{v}_{\text{light}} + \Delta \mathbf{v}_{\text{temp}}}{\lfloor \Delta t_{\text{res}} / \Delta t_{\text{sim}} \rfloor}. \tag{4.4}$$

4.1.2 Fish Animation

Fish swimming is animated using a finite state machine approach to further enhance the realism of the fish simulation. Eleven animation clips are divided into three states: hover, c-start, and swim. The state transition and blending of animation are controlled by the

state of the fish agent, and the animation playback speed is mapped to the current fish speed in simulation. The direction of fish's turn animation depends on the rotation of fish.

4.1.3 States and Actions

Each state s_t in the state space $\mathcal{S}$ encodes the information a fish agent observes in the environment at time step t. It can be represented by a tuple $(\mathbf{u}_t, d_t, \mathcal{I}_t)$, where $\mathbf{u}_t$ is the difference between the forward directions of the agent and its nearest neighboring fish in front of it at current step t, the depth of the agent with respect to the water surface is d_t, and $\mathcal{I}_t$ is a visual observation tensor. During the time of exploration, a fish agent collects visual observations with a spatial grid sensor that imitates the sensing area of real fish (Huth and Wissel, 1992). The visual observation is stored as a third-order tensor, whose dimensions are grid width, grid height, and number of channels. The width and height are defined by the grid resolution, which is set to be 34×20. There are 6 channels encoding a scalar value of the normalized distance from the closest detected object within the fish's sensing range d_{sense} to the agent and a one-hot encoding of the object type (i.e., fish, boundary, or obstacle). d_{sense} is valued at 2 body lengths (BLs) for yellowtail amberjack (yellowtail) (Sakakura and Tsukamoto, 1998a) and 3 BLs for coho salmon and red seabream (Huth and Wissel, 1992). We stack three visual observations together to infer movement before passing them to the networks. All the state components are computed in the local coordinate system of the agent, with the origin located at the body center and the z-axis parallel to the fish's facing direction. Note that a simulated fish is not capable of precisely observing its speed and rotation since a real fish can only sense its relative rotation through the use of the lateral line system (Jiang et al., 2019; Ristroph et al., 2015).

The action a in the action space $\mathcal{A}$ determined by the policy specifies the delta speed Δv_t, as well as delta rotation $\Delta\theta_t^x$ about the x-axis, and $\Delta\theta_t^y$ about the y-axis, in degrees. The rotation angle about the z-axis is clamped to a small angle θ^{zt} to avoid unnatural

rolling behavior. Δv_t is also clamped by the maximum delta speed $\Delta v_{\max}$ allowed in the cage environment. The three action components are then used to compute the delta velocity $\Delta \mathbf{v}_t^f$ of the fish at the current time step and thus drive the motion of the fish agent according to (4.4).

4.1.4 Reward

The reward r_t at each time step is defined to encourage schooling behavior while avoiding boundary collisions and to be consistent with the biological studies described in Section 2.6, as follows:

$$r_t = r_t^{BC} + r_t^{NC} + r_t^{BD} + r_t^{ND} + r_t^{E} + r_t^{M} + r_t^{C}. \tag{4.5}$$

The reward r_t^{BC} represents the penalty of colliding with the spatial boundaries, which include the cage walls and water surface. It has a fixed value of -300 if a boundary collision occurs or 0 otherwise. r_t^{NC} penalizes the collision with neighboring fish detected by box colliders using an associated weight w^{NC} and accumulates with the number of colliding agents N_{hit}, as follows:

$$r_t^{NC} = -w^{NC}\, N_{\text{hit}}.$$

The boundary avoidance reward r_t^{BD} encourages the fish to keep a distance from a detected spatial boundary. Its value depends on the agent's sensing range d_{sense}, the number of detected boundaries N_{bnd}, the distance d_{i} to the boundary i, and the boundary avoidance weight w^{BD}, as follows:

$$r_t^{BD} = -w^{BD} \sum_{i=1}^{N_{\text{bnd}}} \left(\frac{d_{\text{sense}} - d_{\text{i}}}{d_{\text{sense}}} \right).$$

The neighbor interaction reward r_t^{ND} encourages the fish to stay close to its neighbors within the sensing range and to align its direction with those of its neighbors. The angle $\Delta\theta_i^{\text{mov}}$ in degrees between the directions of the agent and each of its N_{nei} neighbors is

calculated and a neighbor interaction weight w^{ND} is used for the computation:

$$r_t^{ND} = w^{ND} \sum_{i=1}^{N_{\text{nei}}} \left(\frac{90 - \Delta\theta_i^{\text{mov}}}{90} \right).$$

On the other hand, r_t^E penalizes energy consumption of the fish while rotating its body or adjusting speed. It is computed from a rotation penalty weight w^r, a speed penalty weight w^s together with the delta angle $\Delta\theta_t$ of the body rotation, and the delta accumulated speed Δv_t^a at the current time step:

$$r_t^E = -w^r \ \Delta\theta_t - w^s \ |\Delta v_t^a| \, .$$

The movement reward r_t^M encourages the fish to swim faster than a minimum speed and penalizes sudden changes in depth caused by aggressive pitch motion (around the local x-axis). In the following expression, the variable θ^{rt} denotes the pitch angle threshold, v^{st} is the speed threshold, θ_t^x is the current pitch angle, and v_t^a represents the current accumulated speed:

$$r_t^M = \begin{cases} -10 & \text{if} \quad \theta^{rt} \leq \theta_t^x \leq (360 - \theta^{rt}). \\ 2 & \text{if} \quad -\theta^{rt} < \theta_t^x \leq \theta^{rt} \text{ and } v_t^a \geq v^{st}. \\ 0 & \text{otherwise.} \end{cases}$$

Lastly, r_t^C denotes the chase reward, which encourages aggression or escape behavior based on the social rank of the fish. Specifically, a dominant fish (aggressor) randomly starts a chase mode with a small probability p_a and initiates an attack on its nearest subordinate neighbor (target). This triggers the subordinate being chased to start its escape mode and swim away from the aggressor. We reward the aggressor by a fixed large value if it collides with its target. This process can be expressed using the aggressor's accumulated velocity $\mathbf{v}_t^a$, a normalized vector $\mathbf{d}$ spanning from the aggressor to the target, a chase reward weight w^{agg} for the aggressor, and an escape penalty weight w^{tar} for the

target, as follows:

$$r_t^C = \begin{cases} w^{\text{agg}} \left(\mathbf{d} \cdot \mathbf{v}_t^a\right) & \text{if fish is an aggressor in chase mode and chases after its target,} \\ 100 & \text{if fish is an aggressor in chase mode and collides with its target,} \\ -w^{\text{tar}} \left(\mathbf{d} \cdot \mathbf{v}_t^a\right) & \text{if fish is a target in escape mode,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

The chase and escape modes end for the two fish either when they collide, the target escapes to the back of the aggressor, or after a short period of time as the dominants prefer not wasting energy on a prolonged chase.

The values of parameters used in action definition and reward function for PPO are listed in Table 4.1. The weight of each reward term is set according to the objective of each training. For example, we set the boundary avoidance weight to be the highest when the DRL controller was trained for the very first time because the primary goal of the training for the fish was to avoid the collision to the fish cage. When it is learnt, we increased the weights for the neighbor collision penalty and energy consumption penalty so that fish learn the smooth schooling in an environment as well as energy saving, which is an important benefit of being in a school (Marras et al., 2015).

4.2 DRL Control Module

4.2.1 Proximal Policy Optimization

The Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017) improves upon the vanilla policy gradient method by providing more stability and reliability during learning. It is a variant of the Trust Region Policy Optimization (TRPO) algorithm (Schulman et al., 2015a) with significantly simpler implementation by optimizing a surrogate loss instead of KL divergence constraints. In our work, the clipped surrogate

Table 4.1: Parameters in action definition and the reward function.

Description	Parameter	Unit	Value
Sensing range (coho salmon and red seabream)	d_{sense}	BL	3
Sensing range (yellowtail amberjack)	d_{sense}	BL	2
Clamping angle of rotation about z-axis	θ^{zt}	Degree	10
Pitch angle threshold	θ^{rt}	Degree	53
Speed threshold	v^{st}	BL/second	0.3
Probability to start chase mode at each time step	p_a	N/A	0.005
Neighbor collision weight (pretraining)	w^{NC}	N/A	0.5
Neighbor collision weight (transfer learning)	w^{NC}	N/A	4
Boundary avoidance weight	w^{BD}	N/A	2
Neighbor interaction weight	w^{ND}	N/A	1.5
Rotation penalty weight (pretraining)	w^{r}	N/A	0.001
Rotation penalty weight (transfer learning)	w^{r}	N/A	0.5
Speed penalty weight (pretraining)	w^{s}	N/A	5
Speed penalty weight (transfer learning)	w^{s}	N/A	10
Chase reward weight for aggressors	w^{agg}	N/A	8
Escape penalty weight for targets	w^{tar}	N/A	1

loss $L^{CLIP}(\theta)$ with respect to the current policy parameters θ is defined as

$$L^{CLIP}(\theta) = \mathbb{E}_{s_t,a_t}\left[\min(l_t(\theta)\hat{A}_t, \text{clip}(l_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right]$$
$$l_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)},$$

where π_θ and $\pi_{\theta_{old}}$ are current and old policies, and $l_t(\theta)$ is the likelihood ratio of the action probability under the current policy over that of the old policy. As the new policy deviates from the old policy, $l_t(\theta)$ will deviate from 1 and be constrained by the interval $[1-\epsilon, 1+\epsilon]$. This means that whenever excessive policy update happens the gradient will be set to zero to prevent the new policy from being too different from the previous one. $\hat{A}_t$ represents the generalized advantage estimation (Schulman et al., 2015b). A minimum value is then chosen to obtain a lower bound of loss.

Since the parameters between the policy and value function are shared in networks, we further integrate the squared-error loss of value function $L_t^{\text{VF}}(\theta)$ and an entropy term

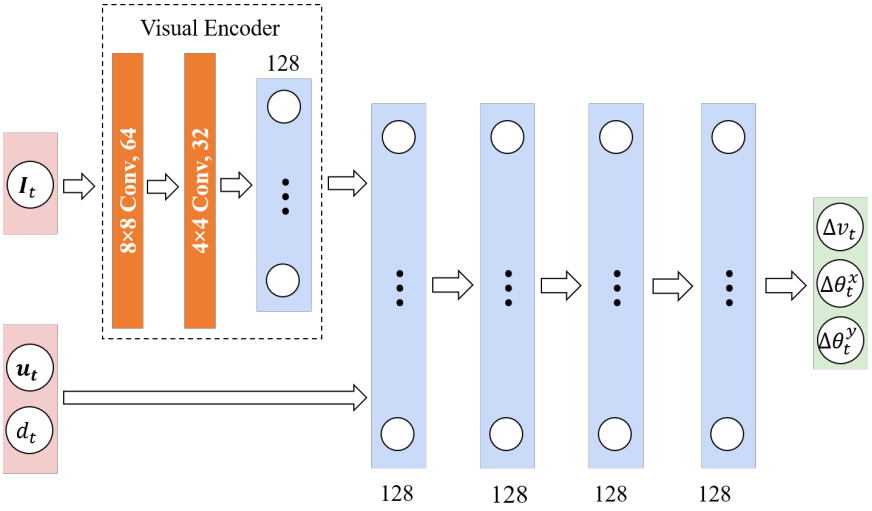


Figure 4.3: Architecture of the policy network.

$S[\pi_\theta](s_t)$, ensuring sufficient exploration to obtain the final objective loss

$$L_t^{CLIP+VF+S}(\theta) = \mathbb{E}_{s_t,a_t}\left[L_t^{CLIP}(\theta) - cL_t^{VF}(\theta) + \beta S[\pi_\theta](s_t)\right],$$

where the value function loss coefficient c is 0.5 and the entropy coefficient β is 0.0005.

4.2.2 Network

The simulation of realistic fish schooling behaviors can be broken down into a series of tasks where fish agents need to choose optimal actions within a continuous action space. In the Actor-Critic framework used by PPO, a policy (actor) π is modeled as a neural network (Figure 4.3) that maps a state s_t to a Gaussian distribution over action $\pi(a_t|s_t)$. The agent's visual observation $\mathcal{I}_t$, which is formed as a 3D tensor, is processed by a visual encoder comprising two convolutional layers and a dense layer before being passed to a fully-connected network with other observation components. The first and the second convolutional layers contain 16 8×8 filters and 32 4×4 filters respectively and use Leaky ReLU activation. The fully-connected layer in the visual encoder contains 128 units and also uses Leaky ReLU as the activation function. Each of the other four fully-connected layers consists of 128 units and uses the Swish activation function. The output layer

is just a linear layer with output size equal to the size of the action space. We train the policies using PPO with a clipped surrogate objective and Generalized Advantage Estimation (Schulman et al., 2015b) as explained above. The value function (critic) $V(s_t)$ is another deep neural network with similar architecture except the output layer is a single unit. The networks are built on the PyTorch open source library.

4.3 Synthetic Dataset Generation

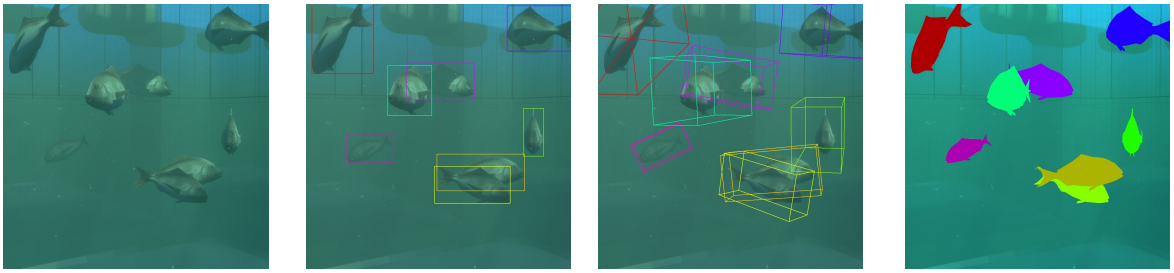
Our simulation is developed on Unity Engine [2] with the ML-Agents toolkit (Juliani et al., 2018), and the Crest Ocean System HDRP asset [3] is utilized to create the sun and ocean wave effects. The simulation is controlled by parameters including, but not limited to, fish biological details such as fish species, count, size, and speed; temperature and light preferences; latitude, longitude, date and time; wave size, sediment and chlorophyll concentrations; sunlight color and intensity; and light attenuation and scattering. A major drawback of prior fish simulation systems (Føre et al., 2009; Ishiwaka et al., 2021) is the need for manual parameter tuning when applying the simulation to new species or new environments. However, in this work we leverage the power of DRL to train the fish to adapt to changing environmental conditions given their biological data, and apply the species-specific trained models to emulate the corresponding swimming patterns.

We also use the physically-based environmental simulation described in Appendix A.2 to control the scene with a small number of hyperparameters instead of extensive adjustment with programmable shaders. These improvements significantly automate the simulation process and allow us to generate an arbitrarily large and varied dataset by randomly setting the above mentioned parameters within a physically valid range. Figure 4.4 displays an example of a synthesized dataset.

The input data is a sequence of rendered images from the simulation scene. The

[2]Available at https://unity3d.com/

[3]Available at https://assetstore.unity.com/packages/tools/particles-effects/crest-ocean-system-hdrp-164158 under the Standard Unity Asset Store EULA license.



(a) Scene (b) 2D and 3D bounding boxes (c) Silhouette

Figure 4.4: Synthetic dataset example. (a) Scene rendering. (b) 2D and 3D bounding box annotations. (c) Silhouette annotation.

output data contains 2D bounding box, 3D bounding box, silhouette, and visibility data for each fish, which can be used to train various computer vision algorithms for tasks such as detection, recognition, tracking, localization and region proposal.

Since the data are generated using computer graphics, it is trivial to compute the fish coordinates in both pixel (screen) space and 3D world space. Therefore, the 2D and 3D bounding box annotations can be automatically obtained for the dataset. Silhouettes of the fish in 2D pixel coordiantes can also be easily generated by projecting the fish geometry onto screen space, and can be used as annotated data to train segmentation algorithms such as Mask R-CNN (He et al., 2017).

4.4 Fish Counting System

The fish counting system comprises three modules: an image pre-processing module, which converts the input video to a sequence of images and applies denoising; a fish detection module, which is a trained network based on YOLOv4 (Bochkovskiy et al., 2020) with the synthetic dataset; and a fish counting module as shown in Figure 4.5. The system takes as input a fish cage video, and outputs an estimate of the total number of fish shown in the video.

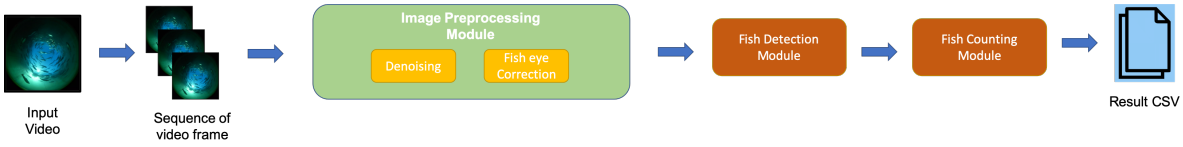


Figure 4.5: The fish counting pipeline, consisting of image pre-processing, detection, and fish counting modules. The image pre-processing module denoises and corrects fish-eye distortion, and feeds the processed data to the detection module. The detection module inputs the processed image frames and outputs the number of fish in each frame. The final fish counting module computes the maximum value of all the estimated values from detection module, which we take as the final output of the estimation.

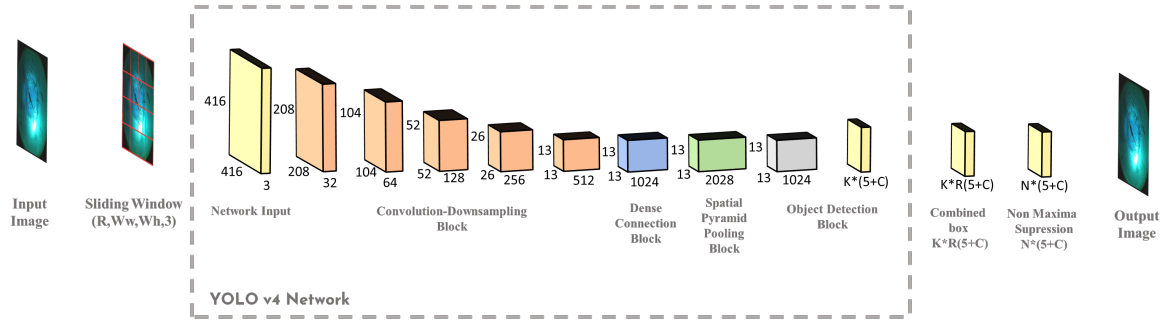


Figure 4.6: The fish detection model network architecture. This module is a region proposal network based on the YOLOv4 model. A sliding window is added as a first layer to crop sub-windows to magnify smaller fish in original images, which effectively avoids false negatives during smaller fish detection.

4.4.1 Image Pre-Processing

This first module transforms the input video to an image sequence and applies denoising and fish-eye lens distortion correction to convert the original video data to the desired format for the detection module. The denoising process applies median blur to the images to reduce the effect of noise, which can be introduced by various factors such as caustics, sediment, and other particles, often seen in underwater photography. The fish-eye correction module is applied to repair distorted images due to the fish-eye lens, which we utilize to capture scenes with a wider field of view (FOV). Our fish-eye lens has a 235 degree FOV. The output of this module is a sequence of repaired-perspective images with suppressed noise, which are fed to the detection module.

4.4.2 Fish Detection

Figure 4.6 shows the network architecture of the fish detection model, based on YOLOv4 (Bochkovskiy et al., 2020). The input to the model is a sequence of images, and the output is a list of proposed bounding boxes (x, y, w, h), where (x, y) is the coordinate of the top left corner of the bounding box, and w and h are the width and height of the bounding box, respectively, with an associated class label and confidence score. We set the confidence score threshold to be 0.5 and any proposed regions below the threshold are discarded. In order to solve the issue of false negatives in the case of smaller fish, a sliding window is added as the first layer of the network. Each window crops a sub-image SI_i with a size of 416×416 from the original image I. The sliding window is shifted repeatedly horizontally and vertically by an amount equal to half the window size, producing a sub-image SI_i at each position that is fed to the YOLOv4 network. The produced number of sub-images, R, is computed as

$$R = \left(\left| \frac{I_{\text{width}} - SI_{\text{width}}}{s} \right| + 1 \right) \times \left(\left| \frac{I_{\text{height}} - SI_{\text{height}}}{s} \right| + 1 \right), \tag{4.7}$$

where SI_{width} and SI_{height} are the width and height of the sliding window and I_{width} and I_{height} are the width and height of the original image, and s is a step-size term and is set to be half of the sliding window size. This technique magnifies small fish in the original images. The computation time increases by a factor of R when processed serially; however, it helps to reduce the number of false negatives, which is a major problem with the original YOLOv4. Note that we concatenate all the results from the cropped images into a single array and apply Non-Maximum Suppression (NMS) to produce a single output, unlike the standard YOLO where the NMS is applied just after a single detection block.

4.4.3 Fish Counting

After the completion of the detection module, the estimation of the fish count is tabulated for each frame and the maximum value is taken as the final estimate of the number of

Table 4.2: Simulation environment configurations.

Species	Fish Number	Body Scale	Cage Size	Cage Shape
Coho salmon	1000	[0.9, 1.1]	Edge of 3, Height of 4.6	Octagon
Yellowtail amberjack	45	[0.9, 1.1]	3×3×3	Cube
Red seabream	10	[0.9, 1.1]	3×3×3	Cube
Red seabream	10	[0.9, 1.1]	5×5×5	Cube
Red seabream	50	[0.9, 1.1]	3×3×3	Cube
Coho salmon	300	0.5	Edge of 3, Height of 4.6	Octagon
Coho salmon	300	1.0	Edge of 3, Height of 4.6	Octagon
Coho salmon	300	1.5	Edge of 3, Height of 4.6	Octagon
Coho salmon	1000	0.5, 1.0 or 1.5	Edge of 1.8, Height of 3	Octagon
Coho salmon	1000	0.5, 1.0 or 1.5	Edge of 3, Height of 4.6	Octagon
All three species	300	[0.9, 1.1]	Edge of 3, Height of 4.6	Octagon

fish in the cage. Despite its simplicity, we found the MAX method to give us the highest counting accuracy.

4.5 Experiments and Results

We next describe the simulation setting, training process, experimental results of the fish behavioral model with diverse biological and cage settings, underwater simulation, and fish counting.

4.5.1 Simulation Setting

We created eleven simulation environments to assess the performance of our fish simulation framework and to generate synthetic datasets for computer vision tasks. The configuration of each environment is available in Table 4.2. The first three environments reproduce the underwater scenes in the fish farms from which we collected field data and are used as default scenes to pretrain each of the three species. The other eight environments are used to examine the adaptiveness of the framework and transfer learning is applied. Values in the cage size column are in units of meters. The biological and environmental parameters used in these scenes are summarized in Table 4.3.

Table 4.3: Biological and environmental parameters in fish simulation.

Description	Parameters	Unit	Value
Body length	BL	m	[0.34, 0.52]
Speed while schooling (coho salmon)	v_t^a	BL/second	[0.2, 1.9]
Speed while schooling (yellowtail amberjack)	v_t^a	BL/second	[0.2, 2.1]
Speed while schooling (red seabream)	v_t^a	BL/second	[0.2, 2.7]
Initial speed	v_0	BL/second	0.2
Maximum delta speed in cage environment	$\Delta v_{\max}$	BL/second	4.1
Light intensity at water surface (Onmaehama)	I_0	PAR	[0, 519.69]
Light intensity at water surface (Nishiki)	I_0	PAR	[0, 15.13]
Light attenuation coefficient (Onmaehama)	a	N/A	1.5
Light attenuation coefficient (Nishiki)	a	N/A	0.26
Lower bound of preferred light intensity (coho salmon, regular size)	I_{lpref}	PAR	[1.1, 4.5]
Lower bound of preferred light intensity (coho salmon, small size)	I_{lpref}	PAR	[1.1, 4.5]
Lower bound of preferred light intensity (coho salmon, large size)	I_{lpref}	PAR	[0.1, 0.5]
Lower bound of preferred light intensity (yellowtail amberjack)	I_{lpref}	PAR	[0.01, 0.02]
Lower bound of preferred light intensity (red seabream)	I_{lpref}	PAR	5.75
Upper bound of preferred light intensity (coho salmon, regular size)	I_{hpref}	PAR	[3.5, 7]
Upper bound of preferred light intensity (coho salmon, small size)	I_{hpref}	PAR	[17.5, 35]
Upper bound of preferred light intensity (coho salmon, large size)	I_{hpref}	PAR	[2.5, 4]
Upper bound of preferred light intensity (yellowtail amberjack)	I_{hpref}	PAR	[341.55, 805]
Upper bound of preferred light intensity (red seabream)	I_{hpref}	PAR	368
Steepness parameter of light reaction	I_{lsteep}	PAR	-20
Steepness parameter of light reaction	I_{hsteep}	PAR	1000
Lower bound of preferred temperature (coho salmon)	T_{lpref}	°C	8
Lower bound of preferred temperature (yellowtail amberjack)	T_{lpref}	°C	18
Lower bound of preferred temperature (red seabream)	T_{lpref}	°C	20
Upper bound of preferred temperature (coho salmon)	T_{hpref}	°C	9
Upper bound of preferred temperature (yellowtail amberjack)	T_{hpref}	°C	28
Upper bound of preferred temperature (red seabream)	T_{hpref}	°C	28
Steepness parameter of temperature reaction	T_{lsteep}	°C	-60
Steepness parameter of temperature reaction	T_{hsteep}	°C	80
Temperature at water surface at noon	T_{surf}	°C	8.3

4.5.2 Training

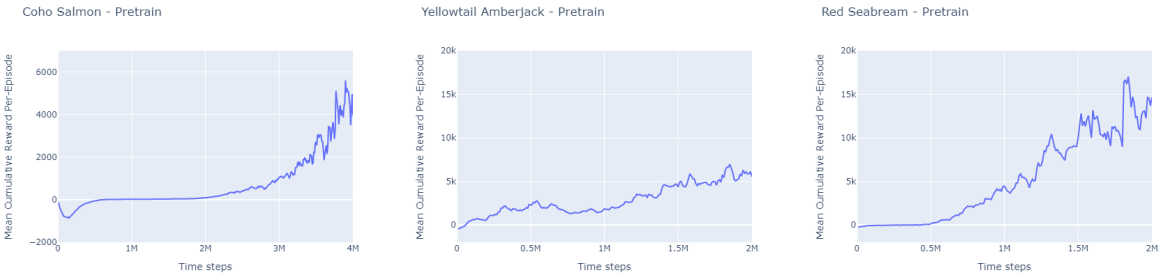
We trained the behavioral models of coho salmon, yellowtail and red seabream. A two-step training scheme was employed to facilitate the adaption for agents in eleven environments with the same action space, observation space, and types of detectable objects, but different fish species, fish sizes, fish numbers, cage sizes, and cage shapes. We pretrained the three species in their corresponding default scenes, which contain 1,000 coho salmon, 45 yellowtail and 10 red seabream, respectively, with the cage configurations set to be the same as the fish farm environments from which we collected field data. The pretraining results were usually sufficient to enable the agents to navigate and avoid cage boundaries to some extent, but collisions between fish and with boundaries still occur and unnatural

movements persist. We then performed transfer learning based on the pretrained policies for each species in its corresponding environment-specific experiments, including the default scene that is used to reproduce the real scene. We found that the use of such a two-step training scheme led to faster convergence and overall performance improvements compared to training from scratch for each environment.

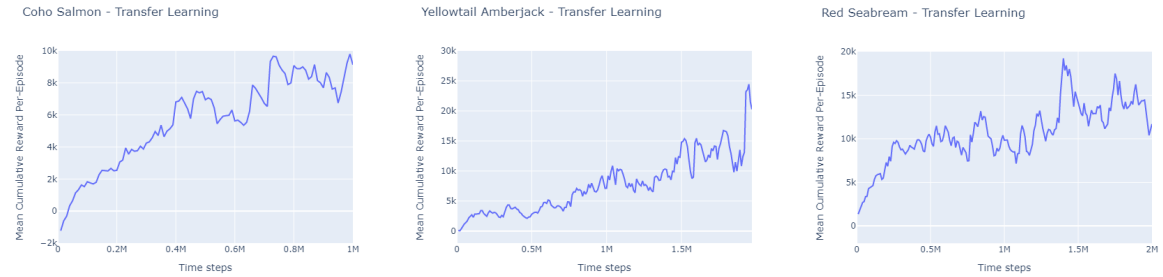
We enabled early termination of an episode if the agent collides with either a cage wall or the water surface. The agent would subsequently restart a new episode from a random position within the cage with a random valid rotation and an initial speed v_0. This strategy discourages boundary collision and prevents the agents from collecting data that is irrelevant to boundary avoidance. The values of hyperparameters in the pretraining and transfer learning phases are similar if not identical. For yellowtail and red seabream, we ran a total of two million time steps and used a linearly decaying learning rate of 0.001 in both phases. For coho salmon, we adopted the same settings, but increased the pretraining and transfer learning steps to four million and one million time steps respectively, because the 1,000 coho salmon in the default scene are much more than the other two species.

We present the training hyperparameters of PPO in Table 4.4. Similar configurations were used for the pretraining and transfer learning phases except for the total number of steps when training coho salmon. We also include an illustration of the learning curves of the three species in Figure 4.7 to showcase the performance improvements over the entire training process.

The total wall-clock training times for four million and five million simulation steps were around 5,500 seconds and 6,800 seconds, respectively. The training ran on an NVIDIA GeForce RTX 3080 Laptop GPU installed on a Windows 10 machine with a 3.3 GHz AMD Ryzen 9 5900HX CPU.



(a) Mean cumulative reward over all agents at each episode in the pretraining phase. Left-to-right: coho salmon, yellowtail amberjack, and red seabream



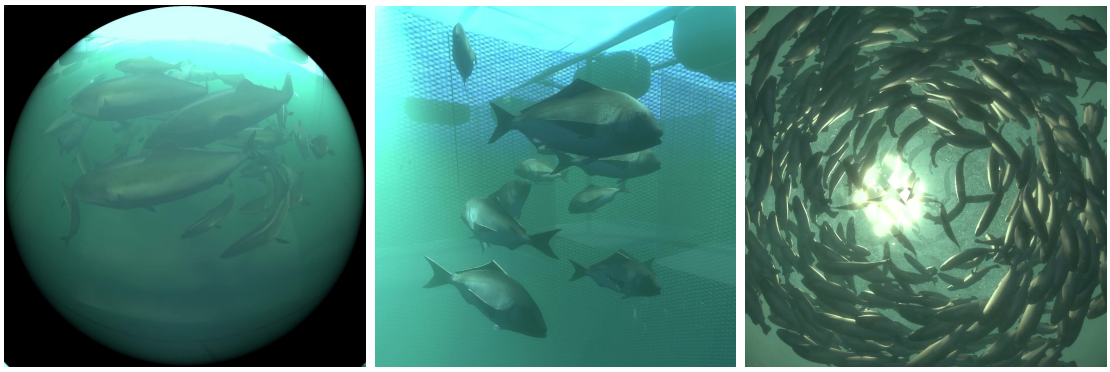
(b) Mean cumulative reward over all agents at each episode in transfer learning phase. Left-to-right: coho salmon, yellowtail amberjack, and red seabream

Figure 4.7: Learning curves from PPO of the three fish species in the pretraining and transfer learning phases in their corresponding default environments. Performance is measured by mean cumulative reward per episode smoothed by an exponential moving average. Notably, coho salmon took four million time steps during pretraining followed by one million during transfer learning because of the larger fish quantity. Nevertheless the other two species were trained for two million steps in both phases.

4.5.3 Fish Behavior

We simulated swimming scenes of the above-mentioned three species in order to provide a variety of synthetic data to train the computer vision system that detects and tracks fish. Fish agents were initialized at random positions in the cage with random valid rotations, and would start to form circling patterns by following the trained policies. A qualitative comparison between the image sequences from the simulation and the videos captured underwater is displayed in Figure 4.8. Note that the swimming patterns of the three fish species are distinct in the captured video. Our simulation successfully reproduced such behavioral differences as well as the environmental variation such as lighting, water color, and turbidity.

(a) Frames from real videos



(b) Frames from simulation

Figure 4.8: Comparison between the scenes captured from different views at a fish farm and in the simulation. Left-to-right: yellowtail, red seabream, and coho salmon. Note that a fish-eye lens is used for yellowtail while a regular lens is used for coho salmon and red seabream.

We also showcase the simulation results of five example environments with different species, fish size, fish number, cage size, and shape settings in Figure 4.9. A body scale of 1.0 refers to the default body lengths (BLs) of coho salmon, yellowtail and red seabream, which were set to be 0.49 meters, 0.52 meters, and 0.34 meters, respectively, according to the collected field data. We additionally applied a mild repulsive force acting as an object-to-object colliding force to push any penetrating fish away from each other to further facilitate collision avoidance among agents. The fish swimming patterns vary with the scene configuration since they result from a combination of factors discussed in Section 4.1.1. Notably, the 10 red seabream in Figure 4.9(a: left 3) slowly circle around the cage center given the small cage size, whereas the same 10 fish in Figure 4.9(a: right 3) learn to swim rapidly and away from the center since the cage size is much larger.

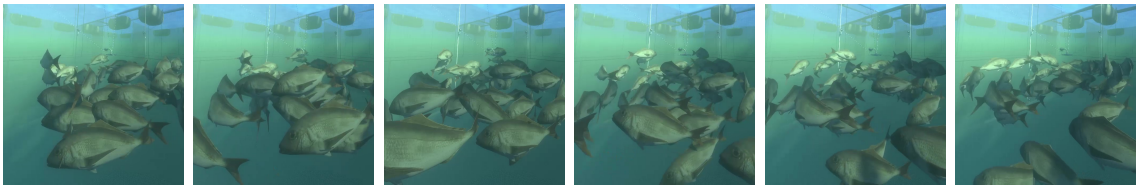
On the other hand, the red seabream in Figure 4.9(b) form a more compact school and slowly circle around the entire small cage because of their larger quantity. In addition, we simulated two fictional scenes where fish with large body size disparity (Figure 4.9(c)) and multiple species (Figure 4.9(d)) are mixed together. With associated biological measurements and trained policies, fish in each of the size and species groups exhibit distinct swimming patterns while influencing those of other groups in the same fish cage.

4.5.4 Simulation vs Reality

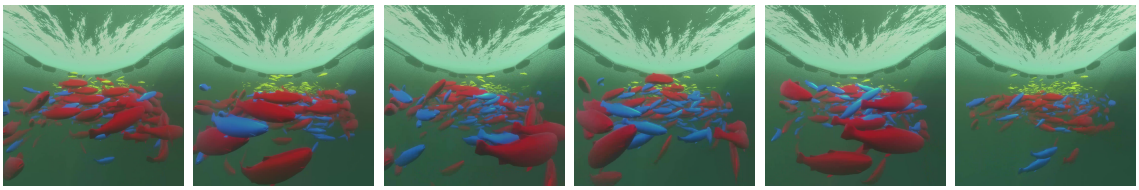
We focus on the relationship between fish density in a cage and the resulting fish school states, and compare simulation results with data obtained in the field. We use two kinds of schools in our simulations, sparse (0.59 fish/m^3) and dense (14.7 fish/m^3), where the densities are set to be consistent with the field work. In the real fish farms, the sparse cage is 6.5 m $\times$ 6.5 m square with a depth of 6m and contains 272 fish. The dense cage is an octagon with 6.5 m edges and a depth of 10 m and contains about 30,000 fish. We quantify states of schools, and focus on milling, characterized by regular circulation around an axis, and swarming, in which fish aggregate with a weak polarization (Delcourt and Poncin, 2012; Calovi et al., 2014). Figure 4.10 shows sampled trajectories, a polar order parameter P given by the norm of the average of direction vectors for each fish in a school, and a normalized angular momentum M representing how regular the circulation is (Calovi et al., 2014) for two schools with different densities. Details of the computation of P and M can be found in Appendix A.3. The simulation result demonstrates that the collective behavior of the school transitions from swarming to milling as density increases, and this tendency is in good agreement with the real data. Several trajectories of individual fish in real video in both sparse and dense schools are exhibited in Figure 4.11, and they resemble the ones in simulation results drawn by the bold trajectories in the right panels.

We further compare the images obtained from the bottom camera to show how the simulation results are consistent with the real data obtained from the field work. These images are fish-eye lens view, and black and white images are made with some threshold

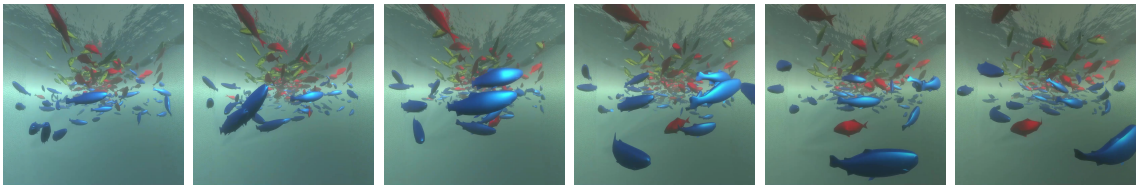
(a) 10 red seabream in a cubical cage with an edge length of 3 meters (left three) and 5 meters (right three).



(b) 45 red seabream in a cubical cage with an edge length of 3 meters.

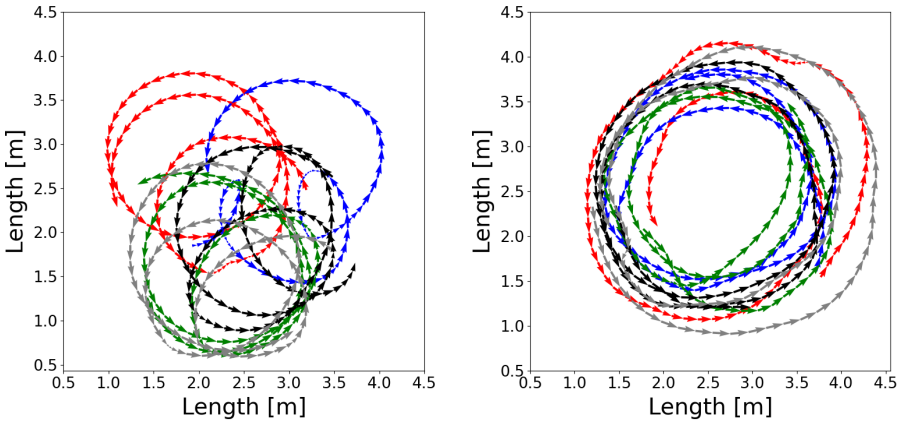


(c) 1,000 coho salmon with body scales of 0.5 (yellow), 1.0 (blue) and 1.5 (red) in an octagon cage with an edge length of 3 meters. The simulated fish form two vertical layers based on their body sizes.

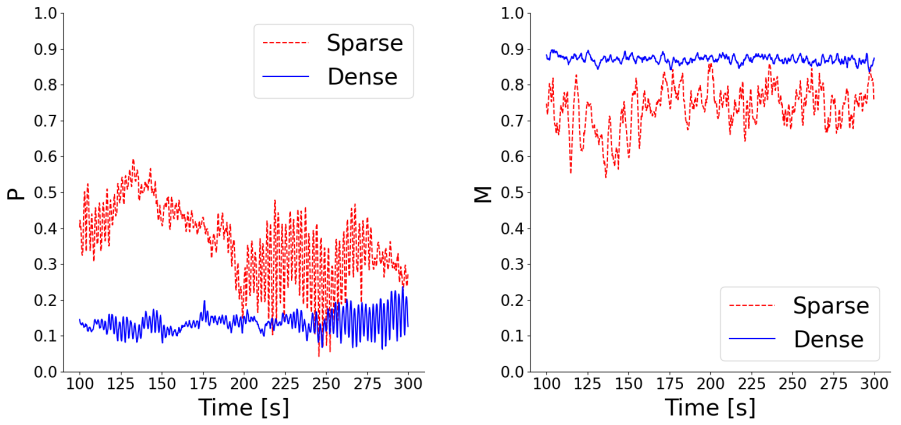


(d) 100 coho salmon (blue), 100 yellowtail (yellow) and 100 red seabream (red) in an octagon cage with an edge length of 3 meters. The swimming patterns vary with their distinct biological factors.

Figure 4.9: Sequence of frames from example environments with diverse fish simulation configurations. All fish have random body scales within the interval [0.9, 1.1] by default unless specified.

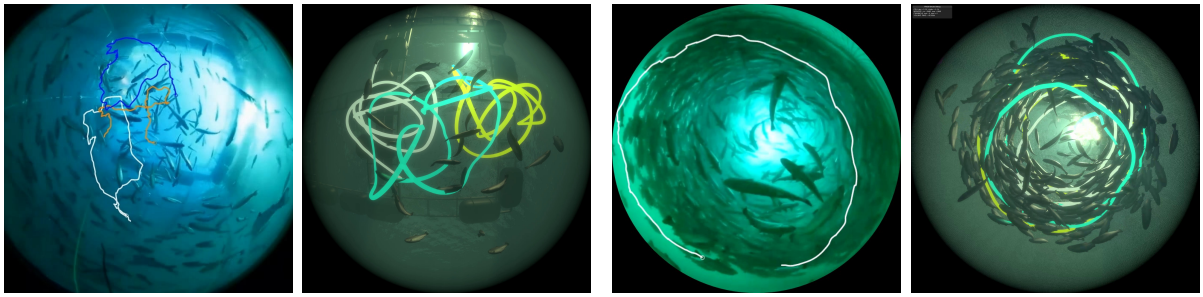


(a) Sampled trajectories



(b) Characteristics of the school

Figure 4.10: Relation between density and states (swarming or milling) of fish schools and their characteristics: (a) 5 trajectories in sparse (0.59 fish/m^3, left) and dense (14.7 fish/m^3, right) schools. The left and right panels show the swarming and milling states, respectively. The fish cages are square with 4.0 m edges in the left panel and octagon with 1.66 m edges in the right panel. (b) Time series of P (left) and M (right) for dense (blue line) and sparse (red dashed line) schools.



(a) Sparse school (b) Dense school

Figure 4.11: Fish trajectories in schools in real video (left panels) of coho salmons and simulated ones (right panels). The number of fish and the size and shape of the fish cage in each simulation are described in the caption of Figure 4.10. The parameters of real ones are as follows : (a) a sparse school (the fish count is 615) in a rectangular cage with 6.5 m edge and 6 m depth. Some fluctuations are observed on the orbits in real videos because the camera cannot be fixated completely in the ocean. (b) a dense school (the fish count is approximately 30,000) in a octagon cage with 6.5 m edge and 10 m depth.

from them. We compare the density of black pixels for each bin obtained by dividing each image into 16 annuli. As a result exhibited in Figure 4.12, the simulation image seems to be consistent with the real one.

Moreover, we show the influence of the dominance hierarchy on fish behavior in Figure 4.13. We initialized the scene with 20% of the fish being dominant and the rest being subordinate based on the reported distribution of social hierarchies among fish in (Sakakura and Tsukamoto, 1998b). The dominant individuals would occasionally chase their subordinate neighbor until colliding with or losing sight of the latter, which matches our observation of the fish behavior in underwater videos captured in aquaculture sites. A demonstration of the simulation scenes is also included in our supplementary video.

4.5.5 Boids vs DeepFoids vs Reality

Figure 4.15 shows a comparison between Boids and DeepFoids-based schooling simulations. We added the boundary avoidance rule to the Boids model in order to conduct a fair comparison with DeepFoids. Boids works well in the open air for birds; however, there are some discrepancies when compared to the behavior of real fish in a caged environment, which are caused by missing details in the implementation from a physics and biology

(a) Real data

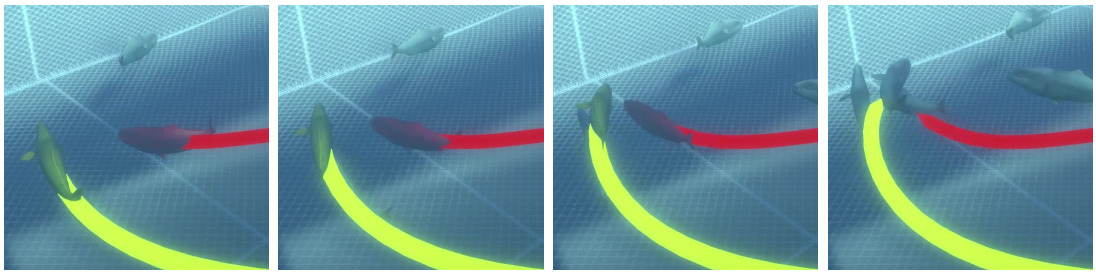
(b) Simulation

Figure 4.12: Comparison between schooling patterns in (a) real and (b) simulation-obtained images. For both (a) and (b), the left, middle, and right panels are, respectively, the fish-eye view picture, black and white images made from the left ones with some threshold, and density of black pixels for each bin obtained from the fish-eye images by dividing them into 16 annuli whose centers are the center of each picture.

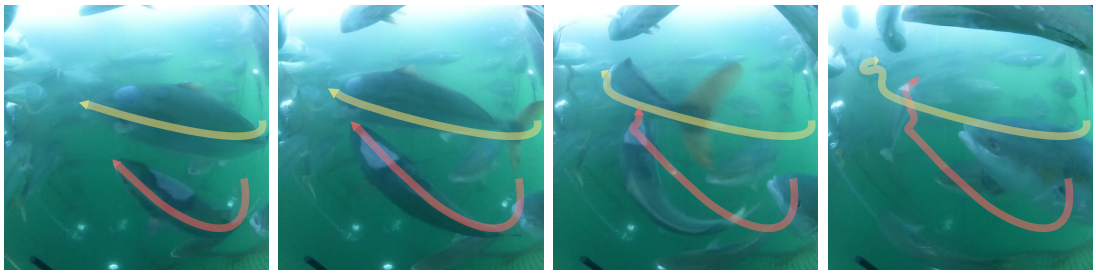
standpoint explained in Section 2.6. Specifically, Boids does not consider the effects of temperature, light intensity and decision interval, which makes the resulting animation quite different from the real behavior. Furthermore, the resulting animation is quite monotone as there are no predators or obstacles in the fish cage, which is usually the way to introduce variability in Boids-based animations. With DeepFoids, aside from the adaptive nature of the approach, we implemented the aforementioned additional factors, which resulted in schooling behavior that more closely resembled the real behavior.

4.5.6 PPO vs SAC

We compare the results of simulations trained by PPO and Soft Actor-Critic (SAC) (Haarnoja et al., 2018), in the default environment of coho salmon, in which 1,000 coho salmon swim in a large octagonal cage with edges of 3 m and a height of 4.6 m. We



(a) Simulation results



(b) Real recordings

Figure 4.13: Sequence of frames from an aggressive behavior instance where a dominant fish (with red trajectory) chases after a subordinate neighbor (with yellow trajectory) until hitting the latter.

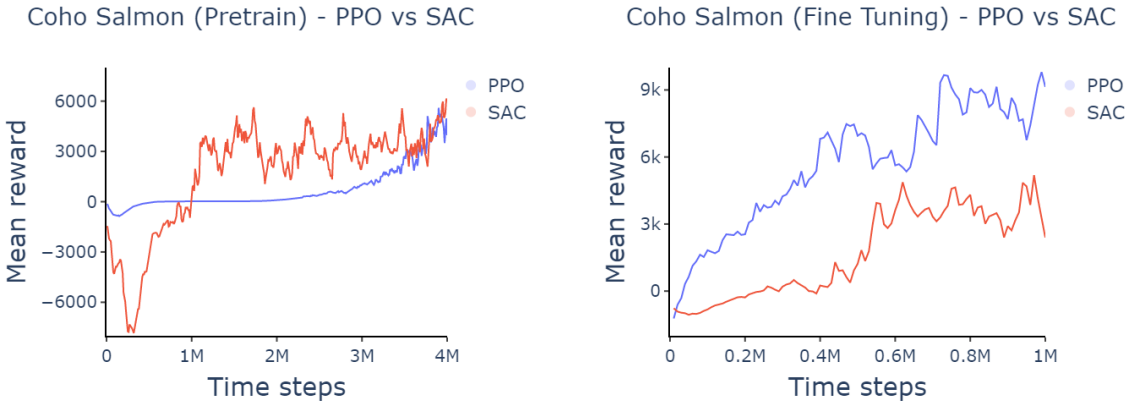
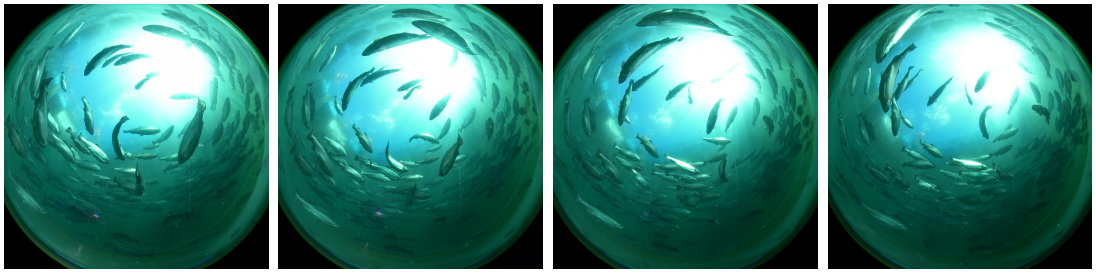
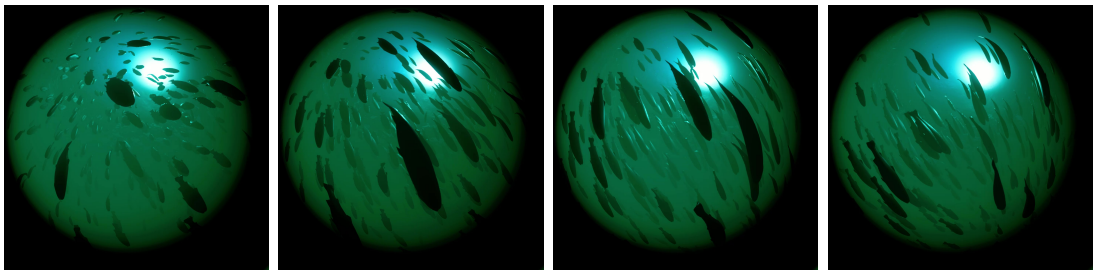


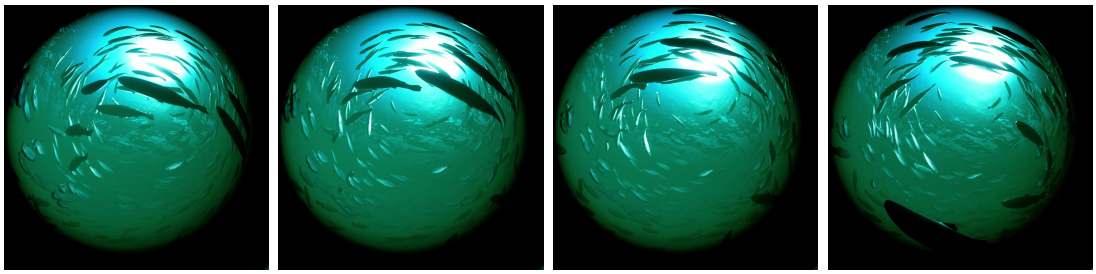
Figure 4.14: Comparison between per-episode mean cumulative rewards over all coho salmon agents trained using PPO (blue) and SAC (red) algorithms in pretraining (left) and fine tuning (right) phases.



(a) Sequence of frames from a bottom view around 10 AM in a real scene.



(b) Sequence of frames from an animation generated by Boids simulation from a bottom view at 10 AM.



(c) Sequence of frames from an animation generated by our simulation from a bottom view at 10 AM.

Figure 4.15: Comparison between the scenes captured from a bottom view around 10 AM: (a) real scene, (b) Boids simulation and (c) our simulation.

report the training hyperparameters of the SAC algorithm (Haarnoja et al., 2018) that we employed in Table 4.5. Note that since we only compare the results of SAC and PPO in the default environment of coho salmon, the total numbers of time steps for SAC agents in pretraining and fine tuning phases are also four and one million.

Both PPO and SAC eventually produced agents forming swimming patterns that resembled those of real coho salmon in reference videos, while the former yielded more stable outcomes. The same reward parameter setting used for PPO was initially applied to SAC in the pretraining stage. Although the SAC agents collected a higher average reward over episodes from an early stage of training than their PPO counterparts, they

became over-trained in the later phase of training and exhibited a circling pattern where all salmon tended to cruise in the same direction and align their body rotations, which would be unnatural in a real-world farming cage setting. This result was jointly caused by a higher sample efficiency and entropy maximization strategy of the SAC algorithm that encourages exploration. After adjusting the reward parameters accordingly, SAC agents were able to swim in equally natural patterns as PPO agents after the pretraining and fine tuning stages. However, no significant improvement in the cumulative average reward was observed in the fine tuning stage of SAC and its resulting value was lower than that of PPO as shown in Figure 4.14. A possible explanation is that the aggregated penalties from constraining factors such as boundary and neighbor collisions overwhelmed SAC agents as their off-policy nature required them to collect experience from previous episodes into the replay buffer for future updates.

4.5.7 Environment Simulation

The results of the experiments with each component of the environment simulation are illustrated in Figure 4.17. Figure 4.17(a) shows how the oceanic scene becomes more reddish brown with higher concentrations of chlorophyll resulting in an increasing attenuation of blue. A higher value of sediment concentration makes the water cloudier as shown in Figure 4.17(b). Figure 4.17(c) depicts a timelapse of the scene. Figure 4.17(d) demonstrates the change in brightness with increasing depth, since more light is absorbed at greater depths. Figure 4.17(e) shows the influence of changing small particle concentration on the brightness of the scene. A higher concentration of small particles increases the amount of light scattered, both towards and away from the camera, causing the light around the sun to brighten and everything away from the light source to darken. Figure 4.17(f) shows the influence of large particle concentration, which works the same way as small particle concentration but has a smaller effect on light scattering.

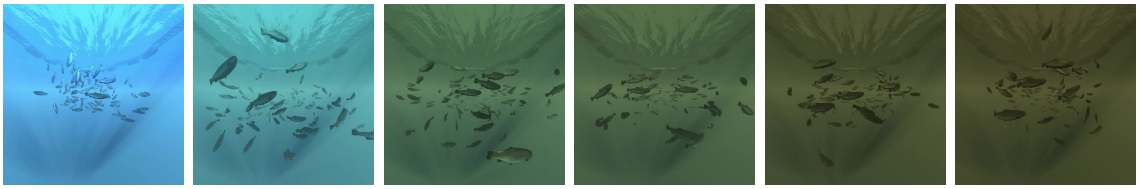
Figure 4.16: Example images of a fish counting system using the YOLOv4 trained model. The captured video data with (left) coho salmon (middle) yellowtail (right) red seabream were fed to our fish counting pipeline, and the total number of fish in each cage was estimated automatically.

4.5.8 Fish Counting

We applied our modified YOLOv4 model (Bochkovskiy et al., 2020)[4] trained with the synthetic dataset to real video footage. The training took 2,000 epochs on a single Nvidia GeForce GTX 1080 Ti with an initial learning rate of 0.001. The training process completed in 5 hours. We then manually annotated 5 frames from 5 videos recorded at a real fish farm—a total of 25 images—and used them to fine-tune the model.

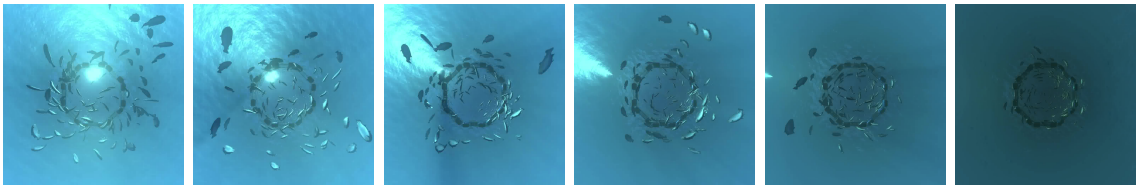
Figure 4.16 shows some results with the videos of coho salmon, yellowtail, and red seabream. The trained fish counting algorithm estimated the number of fish in each frame, and the total number of fish was computed by taking the max of the estimated values while the network processes all the video frames. For coho salmon, we conducted two counting experiments in different farming cages. In the first experiment, the estimated fish count by our fish counting system was 264, while the fish count estimated by human vision was 272. In the second experiment, the count estimation yielded by our fish counting system was 69, and the count estimation by human vision was 61. A single experiment was conducted for yellowtail and for red seabream. The system counting result versus the manual counting result for the two species are 22:32 and 7:7, respectively. We observed that when the density of the fish is high and many occlusions are observed, the accuracy of the counting system drops.

[4]Under Apache License 2.0.

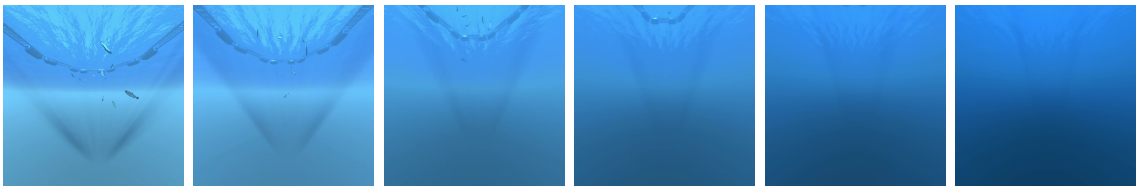


(a) Chlorophyll concentration from 0 to 9 mg/m^3.

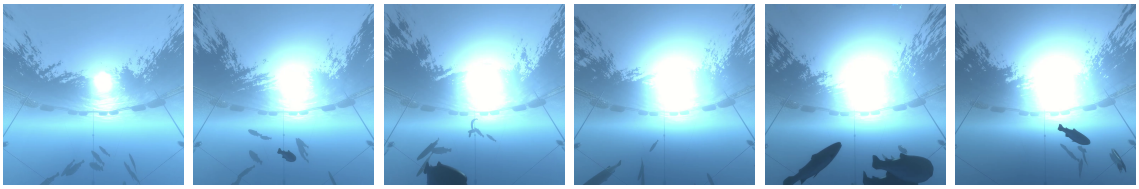
(b) Sediment concentration from 0 to 5 g/m^3.



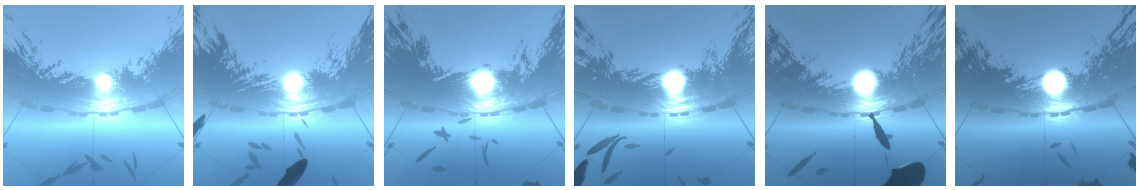
(c) Time-lapse from noon to 6 pm.



(d) Light attenuation from 0 to 10 meters in depth.



(e) Light scattering with small particles (<1 micron in diameter) concentrations from 0 ppm to 4 ppm.



(f) Light scattering with large particles (>1 micron in diameter) concentrations from 0 ppm to 4 ppm.

Figure 4.17: Experiments with each component of the environment simulation.

Table 4.4: Hyperparameters used in training fish agents with PPO.

Parameter	Value
Value function loss coefficient for PPO c	0.5
Entropy coefficient for PPO β	0.0005
PPO clip threshold ϵ	0.2
Regularization parameter for GAE λ	0.95
Batch size	1024
Buffer size	10240
Learning rate	0.001
Number of epochs	3
Discount factor γ	0.99
Time horizon	64
Total number of steps (pretraining, coho salmon)	4 million
Total number of steps (transfer learning, coho salmon)	1 million
Total number of steps (pretraining, yellowtail amberjack and red seabream)	2 million
Total number of steps (transfer learning, yellowtail amberjack and red seabream)	2 million

Table 4.5: Hyperparameters used in training coho salmon agents with SAC.

Parameter	Value
Target smoothing coefficient τ	0.005
Initial entropy coefficient α (pretraining)	0.3
Initial entropy coefficient α (fine tuning)	0
Average steps per update of policy	1000
Batch size	1024
Replay buffer size	102400
Learning rate (pretraining)	0.001
Learning rate (fine tuning)	0.0005
Discount factor γ	0.99
Time horizon	64
Total number of steps (pretraining)	4 million
Total number of steps (fine tuning)	1 million

CHAPTER 5

Conclusions and Future Work

5.1 Conclusions

This dissertation has introduced two novel and highly adaptive simulation pipelines that yield realistic behavior in a biologically valid manner. Our frameworks have achieved the autonomous generation of natural motions for the human facial and cervical muscles as well as fish through the use of deep learning.

First, we proposed a face-head-neck biomechanical complex with a neuromuscular controller for the expression and head pose transfer task. This simulation system is uniquely advantageous. It is anatomically consistent as the biomechanical model emulates the human cervicocephalic musculoskeletal system. Furthermore, our approach is based on the Facial Action Coding System, which is a widely adopted representation of facial expressions, including in the computer animation field. Additionally, our approach is based on deep learning, which can capture the complex, non-linear relation between the Action Units of the FACS and associated facial muscle activations.

Our neural-network-based approach does not require any manual data collection as the training data is generated directly using the biomechanical model itself. This only needs to take place in advance. Once trained offline using the synthesized training data, the neural network can quickly transfer a large number of facial expressions. Unlike popular face tracking software such as Faceware, which requires manual calibration steps to transfer expressions of different subjects, our proposed approach needs no additional parameter adjustments to perform the same task.

Second, we shifted focus to bio-inspired simulation at a macro level. We presented a

fish simulation system that is built based on findings in biology and ecology. With the use of DRL techniques, the system adapts to various conditions autonomously by applying transfer learning, and which enabled us to simulate a broad range of fish behaviors in many scenarios. In addition, we introduced a physically-based underwater simulation which enabled us to simulate various undersea conditions. Using these, we were able to synthesize a large dataset with accurate annotations that we could use for computer vision tasks while avoiding the human effort of data capturing or labeling.

We proved the effectiveness of our fish simulation approach by visualizing the result and comparing it to real video footage, and further proved that the dataset is useful and effective as a training dataset for computer vision by developing a fish counting system to support fish farms. The result of the computer vision estimation was also compared against the count conducted by human vision for different fish species, and the resulting accuracy is encouraging. It is worth mentioning that our model automated the fish counting process and can reduce the burden of fish farm workers. Furthermore, it reduces the chance of bodily injury to the fish, which can result from fish being exposed to air and fish rubbing against the nets.

5.2 Limitations and Future Work

5.2.1 Face-Head-Neck Biomechanical Model

Although we have achieved satisfactory transfer results for all the basic facial expressions, the simulation and transfer of complex mouth and eye movements lies outside the scope of our current work. To improve the fidelity of the results, our musculoskeletal model can be extended by adding more muscles to help activate facial AUs in a more anatomically accurate manner.

Our system still requires manual effort in deciding basic muscle weights for each expression prior to the training data generation. We would like to replace this subjective step by setting the weights based on studies of the correlation between expressions and

muscle activation. Since there seems to be some discrepancy on such correlation in the literature, we may use a union or intersection of the active muscle set for each expression reported in different publications.

We would also like to improve the associated control system so as to explicitly control the lips and eyes, as well as to transfer mixtures of expressions, subtle expressions, and microexpressions, which the human face is capable of producing. Furthermore, it would be interesting to employ the proposed control and transfer pipeline to synthesizing motions with volumetric muscles and tetrahedral flesh model, which is a more widely adopted approach by the community for physically-based facial animation (Ichim et al., 2017).

A methodology that is similar to our neuromuscular control approach for the face-head-neck motion generation can potentially be applied to control a muscle-actuated hand-forearm animation system as described in Appendix B. Although Ip et al. (1998) proposed a Hand Action Coding System that can be considered as a hand motion counterpart to the FACS, to our knowledge no computer vision system exists that can recognize or track hand action units. Hence, we will likely need to estimate hand joint angles with a markerless hand motion trakcing model (Sinha et al., 2016; Isaac et al., 2022), and find the mapping between the muscle activations and joint torques instead of using a more high-level abstraction like the AUs. Another research direction for automatic simulation of the hand-forearm complex is building a DRL-based controller similar to the MyoSuite (Caggiano et al., 2022) developed in MuJoCo. Incorporating motion-imitation objectives into the DRL pipeline and applying modular reinforcement learning techniques to control groups of actuators (Dong et al., 2022) can be two feasible ways to reduce the large search space of the hand-forearm neuromuscular control task.

5.2.2 DeepFoids Model

We presented one application, fish counting, but the potential of our synthetic dataset created with our novel simulations goes beyond fish counting. We can extend the application to 3D detection, pixel-wise segmentation, size estimation, and tracking tasks

in order to make more useful tools for fish farms.

A limitation of our current pipeline is the manual setting of weights for reward functions. We can explore techniques that can dynamically adjust weights based on the performance of the DRL controller in the training process. Moreover, we chose not to apply any behavior cloning method because it was difficult to collect and annotate enough samples of fish trajectory data from real video, which was the main challenge we aimed to solve with the proposed method for the preparation of the dataset for computer vision models. However, the fine-tuning of the trained DRL model using behavior cloning with a small sample of the real fish behavioral data is an interesting method to enhance the reality of the simulation, which can be worth working on.

Our fish simulation with biological and ecological details has further potential to improve an animation pipeline. Although Boids has been the standard approach, the resulting animation misses several details in fish behavior, including the reactions to environmental changes such as water temperature and light intensity, the swimming speed proportional to body lengths, and occasional aggressive behavior as a result of the social hierarchy formed in many schooling fish species. The incorporation of these factors in our model made the resulting animation more realistic. Even so, we still excluded some influential factors like the effect of tidal currents and the influence of fish body temperature on movement dynamics from our simulations, and the proposed simulation method is not applicable to special occasions such as fish behavior in feeding and disease conditions yet. These factors can be incorporated into future versions of the DeepFoids model to make our simulation more biologically accurate.

Finally, we have focused on coho salmon, yellowtail, and red seabream in our work, but our system can be applied to other fish species, including tuna and carp, which are often farmed for the purpose of food sustainability. The same simulation system of the environment can be directly applied to this extension, and the basic simulation pipeline for the fish behavioral model can be reused. The biological and ecological characteristic of different fishes varies among species, and those variables have to be adjusted; however, this can be achieved fairly easily by referring to papers in marine biology.

APPENDIX A

DeepFoids Model

A.1 Biological Background

A.1.1 Schooling in Cages

While schooling in a farmed environment appears similar to swimming in the wild, the reasons behind the need for schooling differs greatly and therefore, the intentions and decisions in the schooling behavior are unique to the situation. Schooling in cages occurs as the result of creating optimal personal space for each individual fish, while still being able to move around (Juell, 1995). When the salmonids in the cage reach a certain crowding density threshold, they begin to form what appears to be a circular schooling pattern in order to best avoid collisions with one another, consistent with the Boids model (Oppedal et al., 2011). In contrast, schooling in the wild is not constrained to a certain area, and fish form schools that swim far and wide for reasons such as energy efficiency and protection from predators (Pavlov and Kasumyan, 2000; Marras et al., 2015). In the caged environment, schooling appears primarily regulated by visual stimuli (Juell, 1995), and the fish's lateral line system also helps to regulate schooling behavior by allowing the fish to feel the vibrations of the swimming patterns of the fish surrounding them (Bleckmann, 2004). The fish orient themselves primarily to the fish in front of them, and accelerate according to the fish surrounding them in the front and back, speeding up if they are too close to the fish behind and slowing down if they are too close to the fish in front (Herbert-Read et al., 2011).

A.1.2 Vertical Distribution of Fish

Often fish in sea cages appear to settle into two or three distinct layers of salmonids grouped vertically in the cage (Cubitt et al., 2003). This group layering phenomenon is likely a product of combined environmental influences such as crowding density, light intensity, and temperature preferences. Fish have individual preferences that vary slightly and are distributed throughout the vertical space accordingly (Føre et al., 2009). Additionally, the fish eye is slower to adjust to changes in light; this is likely a significant factor in the unique vertical grouping of the caged salmonids. The size of the salmonids is also a large factor in the vertical grouping of the schooling fish, and is likely a big component in influencing some of the differences in preferences between the fish. Fish typically group in shoals with similarly sized fish (Krause et al., 1996) and bigger fish tend to go to depths of greater water density, which is found lower in the cage at lower temperatures, while smaller fish swim faster higher up above (Hvas et al., 2021; Folkedal et al., 2012). Implementing these specific parameters of small preferential differences among the fish and the size of the fish allow the model to accurately reproduce the detailed aspects of sea-caged schooling behavior.

A.1.3 Temperature

Salmonids prefer cooler temperatures overall and will exhibit thermoregulatory behavior–changing their swimming patterns or vertical alignment–based on the surrounding water temperature (Reynolds and Casterlin, 1979). Generally, younger salmonids prefer slightly warmer temperatures compared to older salmonids. Salmonids are typically raised for 6-15 months until they reach the adult stage. But just under a month after hatching, salmonids will regulate closer to common adult temperature ranges in the 8–9 degrees Celsius range (Jensen et al., 1989).

A.1.4 Crowding

The density of the caged population is one of the biggest influencing factors affecting the behavior of salmonids in captivity. The circular schooling pattern characteristic to caged salmonids is not a case of classic schooling, but rather is a product of the fish avoiding colliding with one another at high stocking densities. Schooling begins at a density of approximately 243 individuals/2000m^3 (0.1215 fish/m^3), with consistent, uniform schooling patterns observed at a threshold value of 729 individuals/2000 m^3, or at approximately 0.3645 fish/m^3 (Oppedal et al., 2011). This circular swimming pattern allows for salmonids to maintain some movement while avoiding collisions with one another and maintaining a preferred distance from the sides of the cage (Sutterlin et al., 1979). According to literature values, fish maintain a distance of at least 0.66 body lengths (BL) away from their neighbors (Føre et al., 2009). There is also much research that shows the intelligence of these fish, and that they will learn to not swim into cage walls (Odling-Smee and Braithwaite, 2003; Brown et al., 2007). Salmonids are known to be a higher intelligence fish species who have the capacity to remember their pathway home across thousands of miles of ocean (Nevitt et al., 1994; Dittman and Quinn, 1996). This memory serves them well in a caged environment where salmonids prefer to stay at least 0.25 meters from the very top or very bottom of the cage during normal daylight hours, with the exception of feeding times, and on average stay about 1.0 meters away from the sides of the cage if possible (Føre et al., 2009). The inclusion of the memory component adds depth to the previous Boids model in which trajectory decisions were made based only on current information.

A.1.5 Light Intensity

Daylight patterns have arguably the largest influence on swimming patterns in caged salmonids. Salmonids have little movement at night and start to increase movement as the sunlight increases at the start of the day, eventually swimming in a circular, school-like pattern around the cage (Huse and Holm, 1993). Light intensity also affects the vertical

distribution of salmonids. Fish swim closer to the surface of the water during the nighttime and travel lower down in the cage during the day (Fernö et al., 1995). This stark change in swimming behavior from day to night is an important factor in the simulation and will likely be highly beneficial to salmonid farming industry workers who will be able to better predict the behavior of their fish without disturbing the cage environment when they are unable to physically see them in the dark.

A.1.6 Decision Making Interval

The Mauthner cells (M-cells), which exist in 95 % of fish species, are a pair of large and easily identifiable neurons located in the hindbrain. M-cells handle risk avoidance , which triggers a C-start as a rapid escaping movement when a risk event happens. The role of M-cells is not limited to risk avoidance. It is also a command neuron which acts as a neural decision-making cell (Wiersma and Ikeda, 1964), (Korn and Faber, 2005). (Miller et al., 2017) showed that the neural circuit including M-cells is involved in decision making in social groups using zebrafish (Danio rerio). They found that action potentials for swimming lasted for a duration of 50-200 ms and are repeated after a C-start movement is triggered. Although zebrafish and salmonids are different fish species, both have a similar M-cell size (Zottoli, 1978). It is known that the response of the action potential has similar characteristics in the same cell type (Hodgkin and Huxley, 1952) even in different species, but varies with body temperature. To be precise, slower responses occur at lower temperatures (Reyes et al., 2008). Salmonids have a lower body temperature than zebrafish. Therefore, we set the decision-making interval for our salmonids simulation to 200ms, the lower end of the processing speed range.

A.2 Physically Based Environment Simulation

A.2.1 Light Attenuation

The simulation colors the water by first marching rays in fixed distance steps for every pixel on screen. The ray's direction is determined based on the pixel's location when transformed into world space. The current color of the pixel is then treated as incoming light, and becomes the input for the attenuation calculation using the Beer-Lambert Law shown in Eq. (4.1). I_0 represents the incoming light (or in this case the current associated pixel color) of the ray, a (or in this case $a_{\mathrm{r/g/b}}$) is the attenuation coefficient of red, green or blue color in the vector **ac** which is determined by three components that we explain next, and d is the distance the ray of light has traveled through the water (Gallegos and Moore, 2000b). Then the simulation calculates the ambient light at the current point the ray is at. This is calculated by first summing up all the light that is heading towards the current point, and multiplying it by the dot product of the sunlight direction and ray direction. Next, the output of the ambient calculation is multiplied by the output of the attenuation calculation and is then added to a running total. Then the ray marches for a step in its set direction and the process repeats until it reaches its maximum distance. Once that is done, the running total is returned as the output color for the pixel associated with the given ray.

The attenuation coefficient vector consists of three parts: the attenuation of the water itself, the attenuation of chlorophyll, and the attenuation of the suspended sediments. Each part is a 3 dimensional vector representing the attenuation coefficients for red, green, and blue. The values of these coefficient vectors are determined based on data found in (Gallegos and Moore, 2000b; Bricaud et al., 1998b; Akkaynak et al., 2017). The coefficients of each are combined together, with the chlorophyll and sediments both multiplied by the concentration of their respective parameters using the model described in (Gallegos and Moore, 2000b) and (Solonenko and Mobley, 2015b). The formula is as follows:

$$\mathbf{ac} = \mathbf{ac}_{\mathrm{bw}} + co_{\mathrm{chl}}\mathbf{ac}_{\mathrm{chl}} + co_{\mathrm{sed}}\mathbf{ac}_{\mathrm{sed}} \tag{A.1}$$

where $\mathbf{ac}_{\text{bw}}$ is the attenuation coefficient vector of the base water, co_{chl} is the concentration of chlorophyll, $\mathbf{ac}_{\text{chl}}$ is the attenuation coefficient vector of chlorophyll, co_{sed} is the concentration of the sediments, and $\mathbf{ac}_{\text{sed}}$ is the attenuation coefficient vector of sediments.

The factors such as the concentrations of sediments and chlorophyll are set by the simulation to random values from within user defined ranges. These ranges are based off values found in real world, like those laid out in (Bricaud et al., 1998b) and (Solonenko and Mobley, 2015b). The scene uses these factors to determine everything else about the environment. For example, the turbidity of the water is determined by the concentration of suspended sediments.

A.2.2 Light Scattering

A light scattering system computes the light refraction based on the medium's properties such as the particles inside a fluid (Hoobler, 2016). It uses a phase function, which can be obtained from a volume scattering function (VSF), to calculate the amount of light traveling through a medium and being scattered towards the viewer. We use the Kopelevich model (Kopelevich, 1983) as the VSF to determine how different wavelengths of light are scattered underwater using large and small suspended particles. It can be formulated as the following equation (Mobley, 1994):

$$\text{VSF} = \beta_w + v_s\beta_s\psi\left(\frac{500}{\lambda}\right)^{1.7} + v_l\beta_l\psi\left(\frac{500}{\lambda}\right)^{0.3} \tag{A.2}$$

where β_w, β_s and β_f denote the VSF of pure sea water, small and large particles respectively and their values can be selected from tabular values in (Kopelevich, 1983) by performing visual inspection. v_s and v_l are the concentrations of small and large particles (in units of parts per million). ψ is the angle between the light and view directions, and λ is the wavelengths of red, green and blue. The resulting VSF is stored in a vector that is used to calculate the scattering coefficients for the three colors via numerical integration using the trapezoidal rule. Finally, the VSF vector is divided by the scattering coefficients to obtain the phase function for each of the colors, which is then used to multiply the output

of the ray marching system to scatter the color based on the RGB value.

A.3 Characteristics of fish school states

Data provided by the simulation are body length l_{BL}, position $\mathbf{x} = (x, y, z)$ of fish body center, speed v and its direction vector $\mathbf{d} = (d_x, d_y, d_z)$ ($|\mathbf{d}| = 1$) for each individual fish and each time slice. The velocity vector is $\mathbf{v} = v\mathbf{d}$, and the deviation from the center of the school $\mathbf{x}_c = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$ is $\mathbf{r}_i = \mathbf{x}_i - \mathbf{x}_c$, where N denotes the number of fish in the school.

A.3.1 Order parameters

According to Delcourt and Poncin (2012); Calovi et al. (2014) two parameters, a polar order parameter P and a normalized angular momentum M of the school, are used to characterize state of school. The parameters P and M are given respectively as

$$P = \left| \frac{1}{N} \sum_{i=1}^{N} \mathbf{d}_i \right|, \quad M = \left| \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbf{r}_i \times \mathbf{v}_i}{|\mathbf{r}_i||\mathbf{v}_i|} \right| = \left| \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbf{r}_i \times \mathbf{d}_i}{|\mathbf{r}_i|} \right|, \tag{A.3}$$

in the same manner as in (Calovi et al., 2014). All fish move in the same direction in parallel when $P \simeq 1$ (Delcourt and Poncin, 2012; Calovi et al., 2014), which cannot be observed in a fish cage. A large $M \simeq 1$ and small $P \simeq 0$ brings about the milling state, in which fish in a school rotate around an axis regularly (Delcourt and Poncin, 2012; Calovi et al., 2014). The smaller M and P indicate that the state is swarming state rather than milling state (Delcourt and Poncin, 2012).

A.4 Field Data collection

We collected data on four species (two salmonids, yellowtail, red seabream) at marine fish cages. The salmonids cages are in the north, and the yellowtail and red seabream are in the south. Our goal was to measure the physical properties and dimensions of

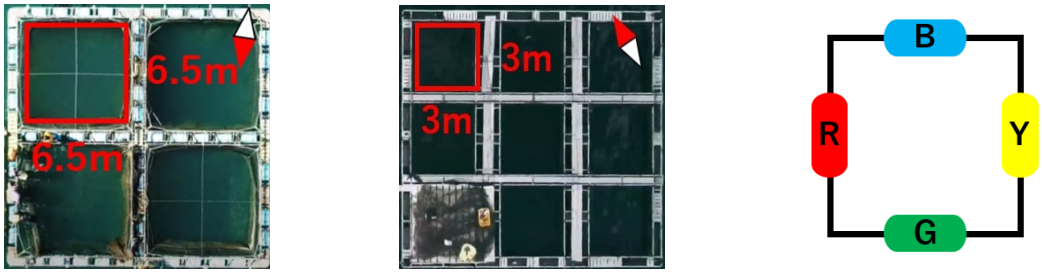
the environment and fish, to set the parameters of our simulation, and to record video of caged fish schooling behavior. In this appendix, we will discuss that data collection process and show our results.

A.4.1 Location

In March, May and July of 2021, we collected data from coho salmon and trout salmon cages on a northern fish farm (latitude 38.465, longitude 141.480). Both fish cages were cubes with a side length of 6.5m (Figure A.1(a)). Then, in June and November of 2021, we collected data from yellowtail and red seabream cages on a southern fish farm (latitude 34.215 and longitude 136.386). Both cages were cubes with side lengths of 3m (Figure A.1(b)).

A.4.2 Equipment

Figure A.2 shows the equipment used. Figure A.2a is circuit board used to measure light intensity. A Raspberry Pi is used to collect data from the TSL2571 light intensity sensor. To help it withstand the water pressure, the circuit board was placed in a clear smartphone case with the sensor positioned to avoid being obstructed. Additionally, the case was placed in a waterproof housing. After waterproofing, the sensor was attached alongside cameras to PVC pipes and used to measure light intensity at various depths underwater. Figure A.2b shows the device used to measure water temperature, the SK SATO SK1260. The sensor must be in direct contact with sea water, so it was lowered into the ocean. To film video of the fish schooling behaviors, we used KODAK 4KVR360 cameras. A camera placed at the center of the floor of the cage pointing up (Figure A.2c) was fixed in place with four ropes. The cameras placed at the sides of the cage facing the center (Figure A.2d) were fixed to PVC pipes and lowered to the target depth. When filming from multiple depths simultaneously, the cameras were fixed to long PVC pipes at 1m intervals, as in Figure A.2f. The PVC pipe fixtures are shown in Figure A.2e. By fixing the T-shaped part to the iron pipes of the cage, long video shoots are possible. The



(a) In north (b) In south (c) Camera locations

Figure A.1: Fish firming cages and camera locations. (a) Salmonoids fish cages in north (coho salmon and trout salmon). (b) Yellowtail and red sea bream fish cages in south. (c) Camera locations in a cage.

cameras are placed at R, G, B, and Y as shown in Figure A.1c.

A.4.3 Environment Data

Figure A.3a shows an example of the results of our light intensity measurements. The measurements were taken at the southern fish farm in November 2021, starting at 11:10 AM. The x-axis shows the number of minutes since the start, while the y-axis is the light intensity (Lux). Light blue shows the readings at a depth of 5m (center bottom), red shows readings at 3m (side), yellow shows readings at 2m (side), and black shows readings at 1m (side). It can be seen that light has lower intensity at greater depths. Sudden noisy drops in brightness are likely the result of fish blocking light from reaching the sensor. Figure A.3b shows the temperature at each depth. The top green line is data from the southern farm in June. The rest are data from the north farm, recorded in (from bottom to top) March, May, and July. Although there are seasons without any temperature variation, at other times the temperature can differ by 3-4 degrees. During the period from May to July, when water temperature exceeds 10 degrees, the chlorophyll content of the water increases, lowering the clarity of the water.

A.4.4 Body Measurements and Fish Count

The number of fish in a fish farm is unclear. Fish are purchased by weight as juveniles. This weight is divided by an average weight to give an approximate count, but this can

(a) (b) (c)

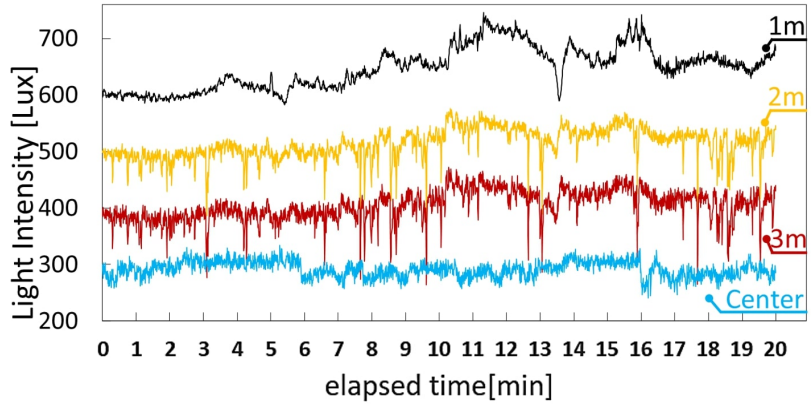
(d) (e) (f)

Figure A.2: Equipment used at the fish cage. (a) Light intensity sensor. (b) Water temperature sensor. (c) Center bottom camera. (d) Side camera. A light intensity sensor is set behind the camera. (e) Fixing PVC pipe, with cameras attached to the cage. (f) Cameras attached to a PVC pipe.

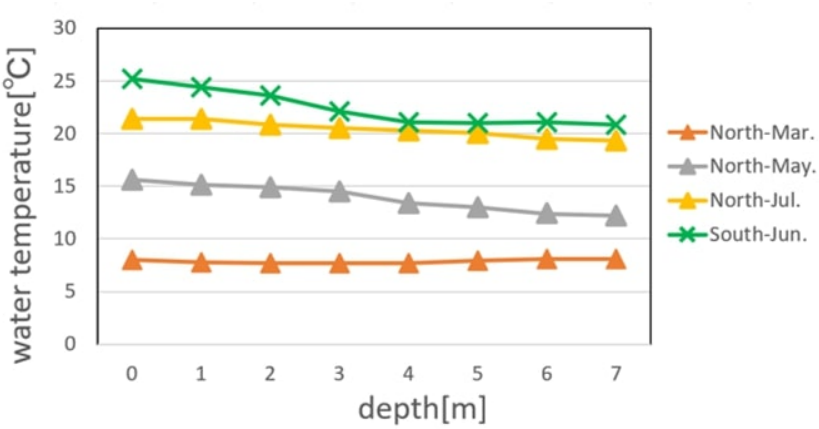
result in up to 50% error. A fish's value on the market is determined by weight. Therefore, fish farmers try to monitor the state of their fish (length, height, width, and weight) in order to optimize their feeding, but the process is dependent on farmers' intuition. The only way to accurately measure a fish is by hand after removing it from the water[5]. To do so, fish are scooped up by a net a few fish at a time; when the fish are released again, they are counted by eye. When populations are small this process can be performed for all fish, but it is not realistic for more than 1000 fish. For such cases, the process is performed for approximately 10% of the fish and the result is multiplied by 10, giving an inexact value. Figure A.4c shows the current fish counting process.

Fish body measurements are even more difficult. To measure a fish, the fish is removed from the water and anesthetized to prevent injury, and measured by hand with a tape measure (see Figure A.4). Figure A.5 shows the body lengths of 10 randomly selected

[5]These measurement methods, while designed to avoid injuring the fish, are neither completely safe nor particularly accurate. There is a demand for an alternative method to obtain these data from video data



(a) Light intensity



(b) Water temperature

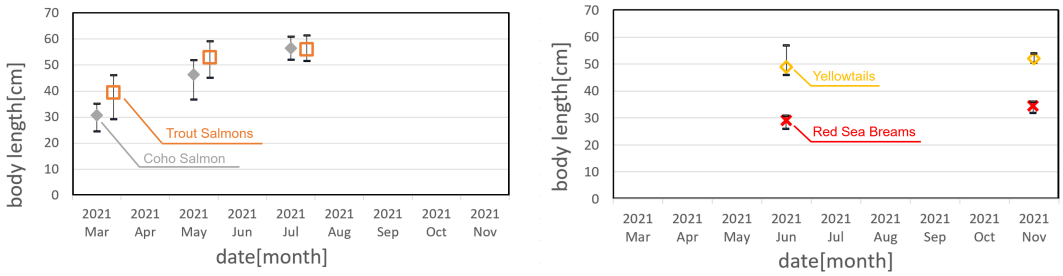
Figure A.3: Environment data: (a) November 2021, Northern farm light intensity. From bottom to top: center bottom (5m), 3m, 2m, 1m. (b) Water temperature for each depth.

(a) Scooping fish

(b) Fish body measurement

(c) Fish counting

Figure A.4: Fish body measurement and counting process.

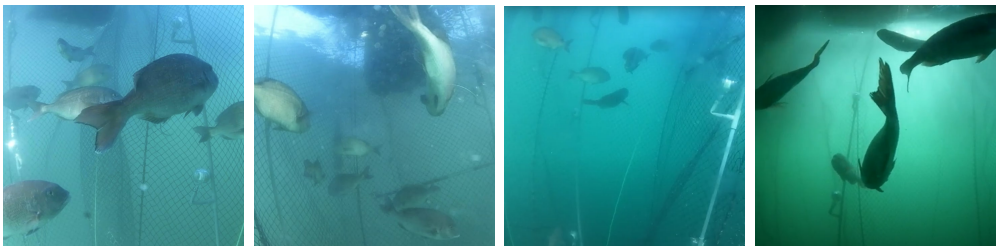


(a) (b)

Figure A.5: Body length: (a) Salmon body length. Orange: trout salmon, gray: coho salmon. (b) Yellowtail and red seabream body length. Red marks: Red seabream, yellow marks: yellowtails.

(a) Coho salmons at a depth of 3m in April, 2022 at north fish cage.

(b) Yellowtail amberjacks at a depth of 3m in November, 2021 at south fish cage.



(c) Red seabreams at a depth of 2m in November, 2021 at south fish cage.

Figure A.6: Examples of recorded video shots of (a) coho salmons, (b) yellowtail amberjacks, and (c) red seabreams. For each fish species, from left to right panels show camera views located at R, G, B, and Y respectively.

fish. Figure A.5a shows the results for salmon (coho salmon and trout salmon), and Figure A.5b shows the results for yellowtail and red seabream. The x-axis shows the month the measurement was performed. The lines show the range of body lengths, and the symbol is the average length (coho salmon: gray, trout salmon: orange, yellowtail: yellow, red seabream: red). Salmon farming begins in December, and they are shipped to market in July. The graph shows that there exists large variation in salmon body length, which gets smaller as July approaches. The shipping date for yellowtail and red seabream is not fixed; they are shipped once they reach a set size.

A.4.5 Video Recording

Videos at the fish farm were filmed with KODAK 4KVR360 cameras. One camera was placed at the bottom center of the cage, facing the ocean surface. The others were attached to PVC pipes placed at the center of each side at 1m intervals. The footage captured in March, May, and July 2021 on the northern farm and June 2021 on the southern farm was filmed in the KODAK 4KVR360 235 degree dome mode. The November 2021 footage on the southern farm was filmed in the KODAK 4KVR360 155 degree front mode. The coho salmon, yellowtail, and red seabream footage is show in Figure A.6. The cameras are placed at R, G, B, Y in Figure A.1c. The salmon and yellowtail footage was taken at a depth of 3m, while the red seabream footage was taken at a depth of 2m. The footage at these depths contained the largest number of fish for each species, showing that the preferred depth differs per species.

APPENDIX B

Hand-Forearm Biomechanical Model

B.1 Introduction

The hand and forearm constitute a inseparable part of the human body and, similar to the face, is a vital manifestation of the ego (Van Nierop et al., 2008). Therefore, musculoskeletal modeling of the hand and forearm is a natural and essential step to enhance the realism of anthropomimetic animation. Due to the complex coupling and synergy of musculotendons and bones, there remains a deficiency of literature that studies anatomically accurate muscle-driven biomechanical hand simulation.

In this Appendix, we introduce a prototype of a musculoskeletal hand-forearm complex that is capable of generating a variety of basic hand motions from muscle activation inputs. Although we only simulate 25 hand and forearm muscles out of around 50 in total, the muscles are still redundant, thereby producing complementary and antagonistic effects on joints, and allowing the system to work in a physiologically consistent manner.

B.2 Methodology

The hand-forearm model is numerically simulated as a force-driven articulated multi-body system. Our musculoskeletal system is based on the one by (Yan, 2012), but with muscles modeled as 1D vectors connecting two ends with a selection of control points in between. There are 16 passive rigid bodies representing 30 bones of a right hand-forearm complex, some of which (e.g., metacarpal bones at the palm) are grouped together for simplicity. The system contains a total of 16 degrees of freedom. The thumb has 2 bones (the

proximal phalanx and distal phalanx) and 2 hinge joints comprising 2 degrees of freedom. Each of the other four fingers comprises 3 bones (the proximal phalanx, intermediate phalanx, and distal phalanx) with 3 hinge joints that constitute 3 degrees of freedom. The remaining 2 degrees of freedom come from the wrist, which is modeled as a universal joint connecting the palm and the forearm.

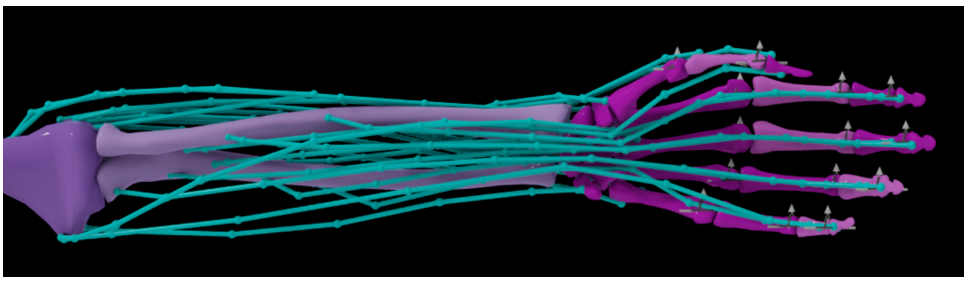
A total of 25 muscles are simulated in the hand and forearm. The thumb is actuated by 3 muscles (one flexor and two extensors), each of the other four fingers by 4 muscles (two flexors and two extensors), and the wrist by 6 muscles. Each muscle actuator is a modified version of the Hill-type muscle model (Ng-Thow-Hing, 2001), as discussed in Section 3.1.2. An activation level between 0 and 1 is associated with each muscle and is proportional to the proactive contractile force in addition to a passive muscle force that constrains muscle deformation.

Figure B.1 shows the muscle routing in our simulation and visualizations of the muscle geometries. At each simulation step, muscle forces are applied to each of the attachment points to the bones. We adopt the attachment points from (Yan, 2012) and uniformly sample a subset of them with a target density for higher computational efficiency while ensuring the capability of the model.

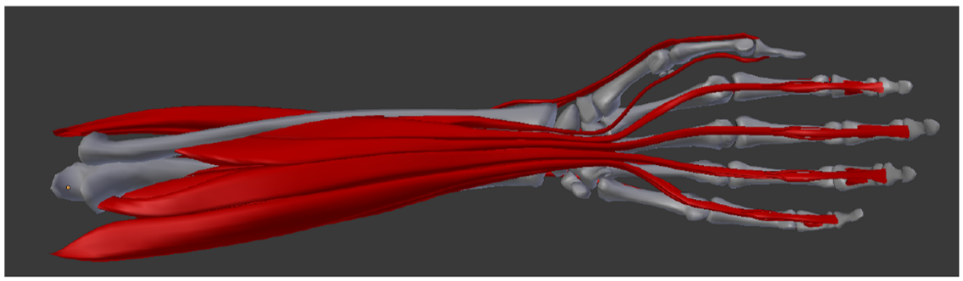
B.3 Experiments

We have tested the musculoskeletal behaviors of a biomechanical complex consisting of a right hand and a right forearm. The system was built on the NVIDIA Omniverse Create platform[6] and runs on a Windows 10 machine with a 3.3 GHz AMD Ryzen 9 5900HX CPU and an NVIDIA GeForce RTX 3080 Laptop GPU. The physics simulation runs at 600 time steps per second. We employed Absolute Character Tools and Autodesk 3ds Max to obtain muscle properties required for force computation such as physiological cross-sectional area (PCSA) for each muscle.

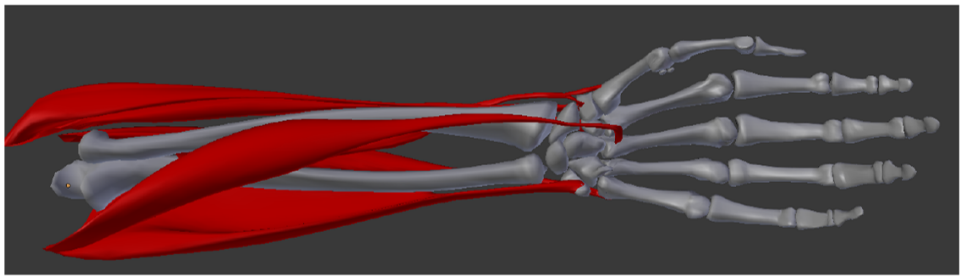
[6]See https://developer.nvidia.com/nvidia-omniverse-platform



(a) Simulated muscles and bones



(b) Visualized hand muscles



(c) Visualized wrist muscles

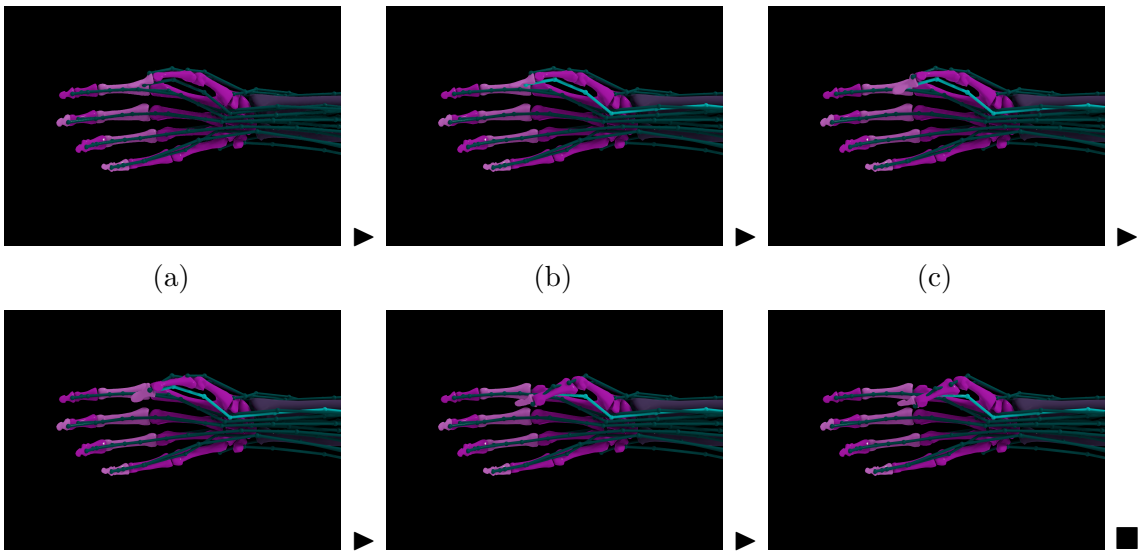
Figure B.1: Frontal view of the hand and forearm muscles. (a) Our simulated muscles (blue) with a subset of control points (small spheres along the muscles) are shown along with the skeleton (purple), implemented in Omniverse. (b) Visualization of the hand muscles. (c) Visualization of the wrist muscles. Both the muscle visualization figures were reproduced from (Yan, 2012).

Fingers: Figure B.2 and Figure B.3 show the sequences of frames from the simulation of the thumb flexion and extension, respectively. We only activated the relevant muscles for each of the tasks, which are shown in bright blue while the deactivated muscles have a dark blue color. The activated muscles have a muscle activation set to 1.0 and the deactivated ones have the activation set to 0.0. Although the model works in gravity, we disabled the gravity in the demonstration to better showcase the finger movement. The flexion and extension of the index finger are illustrated in Figure B.4 and Figure B.5. When the finger flexor is activated, the finger bones rotate so that the finger is adducted towards the palm, which is a basic movement for making a fist or grasping objects. In the case of the finger extensor being activated, the finger extends from its natural pose. Additionally, when all the muscles that actuate a finger are deactivated, the finger slowly returns to its natural pose.

Wrist: Figure B.6 and Figure B.7 show sequences of frames from the simulation of the wrist flexion and extension, both of which can be achieved by only activating the corresponding muscles. The joints of the fingers were frozen in this case. Additionally, Figure B.8 and Figure B.9 show the abduction and adduction of the wrist controlled by the simulated muscles. When abduction happens, the entire hand moves towards the side of the thumb, while in adduction the hand moves towards the side of the little finger.

B.4 Limitations and Future Work

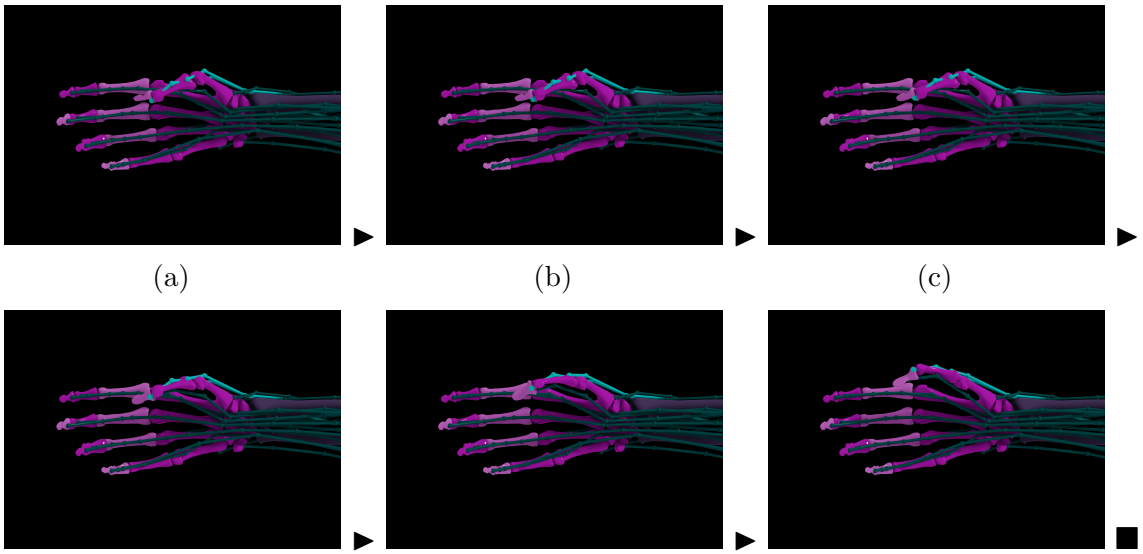
We have focused on building the muscle routes and incorporating the muscle actuators in this preliminary work. However, the system still requires a high-level neuromuscular controller to achieve automatic generation of proper hand motions to perform different tasks such as reaching to targets and object manipulation. This is a promising direction for our musculoskeletal hand-forearm model and we plan to develop such controllers using learning-based methods such as fully-connected deep neuromuscular controllers (e.g., (Nakada et al., 2018) and (Nakada, 2017)) or Deep Reinforcement Learning methods



(a) (b) (c)

(d) (e) (f)

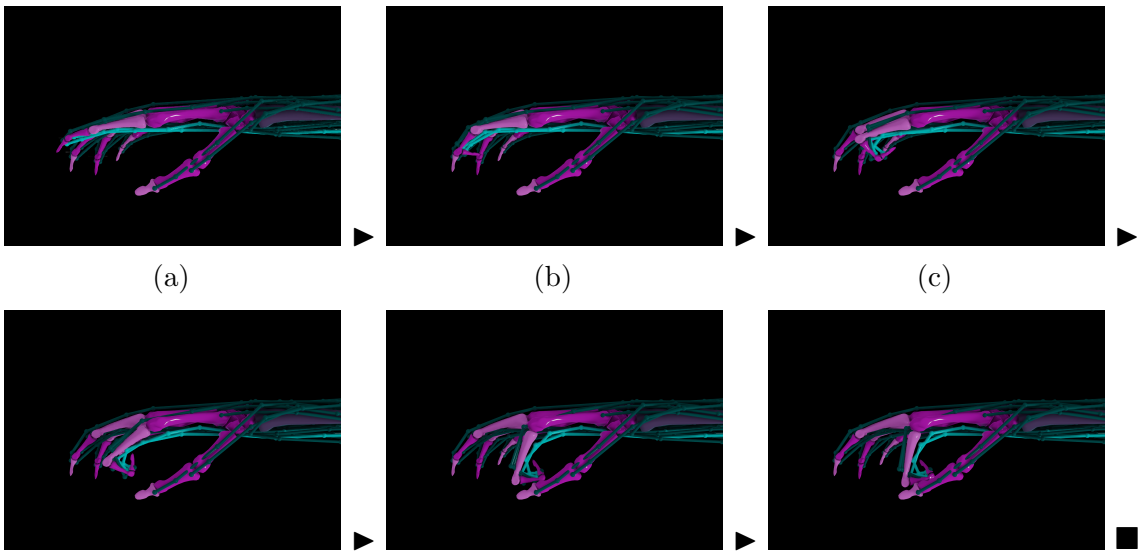
Figure B.2: Sequence of frames from the simulation of the flexion of the thumb. Only the muscles that are responsible for the thumb flexion (light blue) is activated.



(a) (b) (c)

(d) (e) (f)

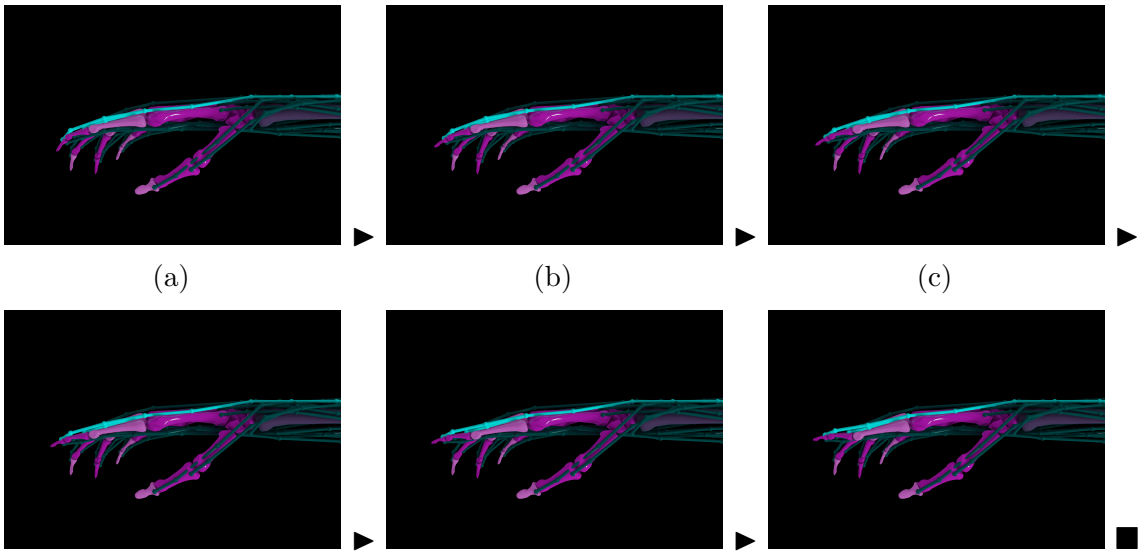
Figure B.3: Sequence of frames from the simulation of the extension of the thumb. Only the muscles that are responsible for the thumb extension (light blue) is activated.



(a) (b) (c)

(d) (e) (f)

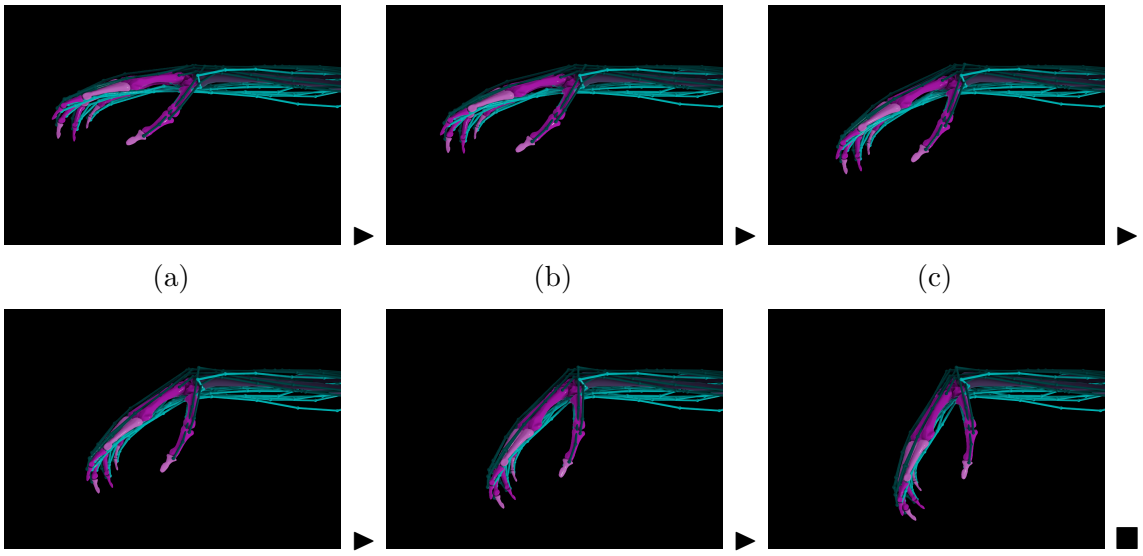
Figure B.4: Sequence of frames from the simulation of the flexion of the index finger. Only the muscles that are responsible for the finger flexion (light blue) is activated.



(a) (b) (c)

(d) (e) (f)

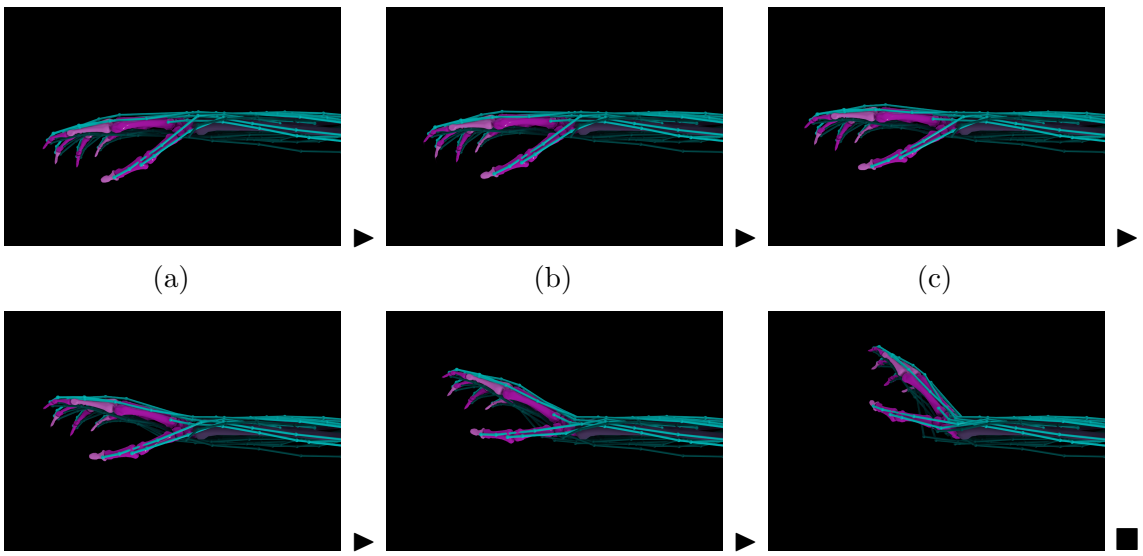
Figure B.5: Sequence of frames from the simulation of the extension of the index finger. Only the muscles that are responsible for the finger extension (light blue) is activated.



(a) (b) (c)

(d) (e) (f)

Figure B.6: Sequence of frames from the simulation of the flexion of the wrist.



(a) (b) (c)

(d) (e) (f)

Figure B.7: Sequence of frames from the simulation of the extension of the wrist.

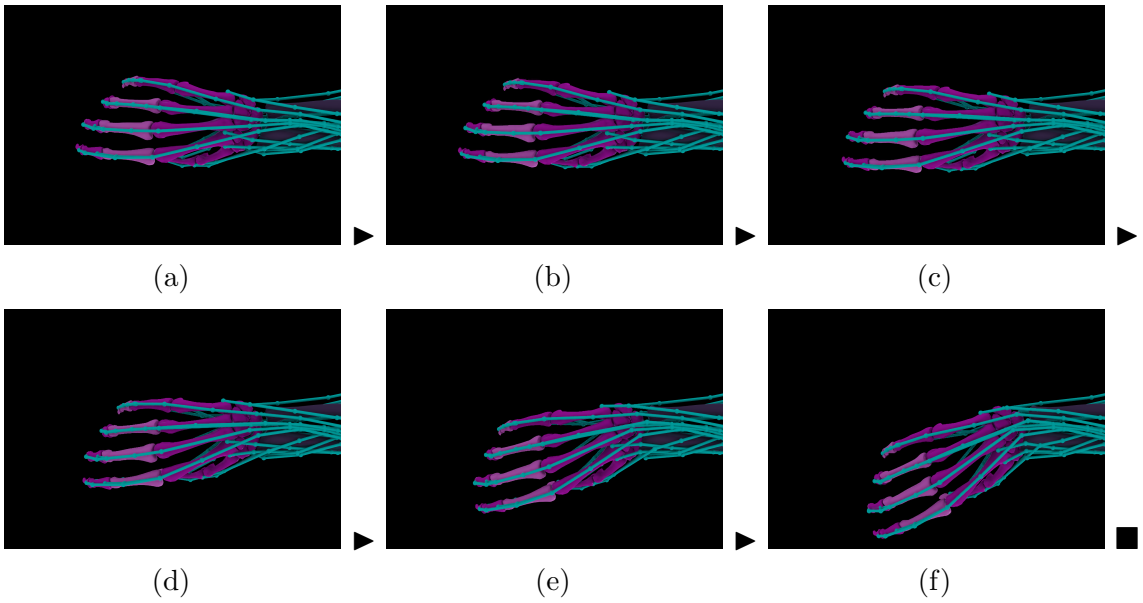


Figure B.8: Sequence of frames from the simulation of the abduction of the wrist.

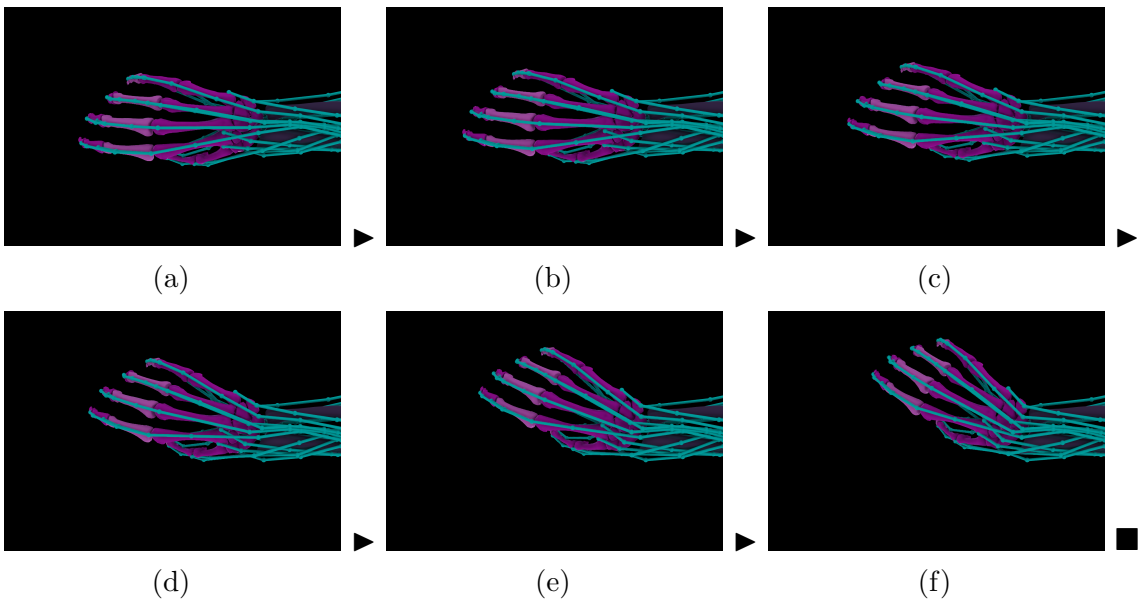


Figure B.9: Sequence of frames from the simulation of the adduction of the wrist.

(e.g., (Caggiano et al., 2022) and (Dong et al., 2022)).

Another limitation of the current system is that we ignore the penetration of muscles into the hand bones. This issue not only causes visual deficiency, but more importantly, sometimes creates incorrect short paths for muscles to contract through the bones and may result in improper finger bending directions, which seems to be particularly problematic for finger extension. Therefore, approaches that add bone penetration constraints will need to be developed for the hand model in the future.

Abraham, J. P., Baringer, M., Bindoff, N., Boyer, T., Cheng, L., Church, J., Conroy, J., Domingues, C., Fasullo, J., Gilson, J., et al. (2013). A review of global ocean temperature observations: Implications for ocean heat content estimates and climate change. *Reviews of Geophysics*, 51(3):450–483. 37, 38

Ahrens, M. B. (2019). Zebrafish neuroscience: Using artificial neural networks to help understand brains. *Current Biology*, 29(21):R1138–R1140. 11

Akkaynak, D., Treibitz, T., Shlesinger, T., Loya, Y., Tamir, R., and Iluz, D. (2017). What is the space of attenuation coefficients in underwater computer vision? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4931–4940. 74

Aneja, D., Colburn, A., Faigin, G., Shapiro, L., and Mones, B. (2016). Modeling stylized character expressions via deep learning. In *Proc. Asian Conference on Computer Vision*, pages 136–153. 9

Australian Institute of Marine Science (AIMS) (2019). OzFish dataset — machine learning dataset for baited remote underwater video stations. 13

Baltrusaitis, T., Zadeh, A., Lim, Y. C., and Morency, L.-P. (2018). OpenFace 2.0: Facial behavior analysis toolkit. In *Proc. IEEE Conference on Automatic Face and Gesture Recognition*, pages 59–66. 24

Bleckmann, H. (2004). 3-d-orientation with the octavolateralis system. *Journal of Physiology-Paris*, 98(1-3):53–65. 12, 70

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*. 47, 49, 63

Breder, C. M. and Halpern, F. (1946). Innate and acquired behavior affecting the aggregation of fishes. *Physiological Zoology*, 19(2):154–190. 11

Bricaud, A., Morel, A., Babin, M., Allali, K., and Claustre, H. (1998a). Variations of light absorption by suspended particles with chlorophyll a concentration in oceanic (case 1) waters: Analysis and implications for bio-optical models. *Journal of Geophysical Research*. 13

Bricaud, A., Morel, A., Babin, M., Allali, K., and Claustre, H. (1998b). Variations of light absorption by suspended particles with chlorophyll a concentration in oceanic (case 1) waters: Analysis and implications for bio-optical models. *Journal of Geophysical Research: Oceans*, 103(C13):31033–31044. 74, 75

Brown, A. A., Spetch, M. L., and Hurd, P. L. (2007). Growing in circles: Rearing environment alters spatial navigation in fish. *Psychological Science*, 18(7):569–573. 11, 12, 72

Caggiano, V., Wang, H., Durandau, G., Sartori, M., and Kumar, V. (2022). Myosuite–a contact-rich simulation suite for musculoskeletal motor control. *arXiv preprint arXiv:2205.13600*. 68, 91

Calovi, D. S., Lopez, U., Ngo, S., Sire, C., Chaté, H., and Theraulaz, G. (2014). Swarming, schooling, milling: phase diagram of a data-driven fish school model. *New Journal of Physics*, 16(1):015026. 55, 76

Calvo, M. G. and Lundqvist, D. (2008). Facial expressions of emotion (kdef): Identification under different display-duration conditions. *Behavior Research Methods*, 40(1):109–115. 26

Cardona, M. and Cena, C. E. G. (2019). Biomechanical analysis of the lower limb: A full-body musculoskeletal model for muscle-driven simulation. *IEEE Access*, 7:92709–92723. 8

Chen, D. T. and Zeltzer, D. (1992). Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 89–98. 8

Clark, E. A., Kessinger, J., Duncan, S. E., Bell, M. A., Lahne, J., Gallagher, D. L., and O'Keefe, S. F. (2020). The facial action coding system for characterization of human affective response to consumer product-based stimuli: A systematic review. *Frontiers in Psychology*, 11:920. 28

Cohn, J. F., Ambadar, Z., and Ekman, P. (2007). Observer-based measurement of facial expression with the facial action coding system. *The Handbook of Emotion Elicitation and Assessment*, 1(3):203–221. 2, 9

Cong, M., Bao, M., E, J. L., Bhat, K. S., and Fedkiw, R. (2015). Fully automatic generation of anatomical face simulation models. In *Proc. ACM SIGGRAPH/EG Symposium on Computer Animation*, pages 175–183. 15

Cubitt, K., Churchill, S., Rowsell, D., Scruton, D., McKinley, R., et al. (2003). 3-dimensional positioning of salmon in commercial sea cages: Assessment of a tool for monitoring behaviour. In *Aquatic telemetry. Advances and applications. Proceedings of the fifth Conference on Fish Telemetry held in Europe, Ustica, Italy*, pages 25–33. 12, 71

Davis, V. A., Holbrook, R. I., and de Perera, T. B. (2021). Fish can use hydrostatic pressure to determine their absolute depth. *Communications biology*, 4(1):1–5. 11

de Souza, C. R., Gaidon, A., Cabon, Y., and Peña, A. M. L. (2017). Procedural generation of videos to train deep action recognition networks. *arXiv:1612.00881*. 14

Delcourt, J. and Poncin, P. (2012). Shoals and schools: back to the heuristic definitions and quantitative references. *Reviews in Fish Biology and Fisheries*, 22(3):595–619. 55, 76

Ditria, E. M., Connolly, R. M., Jinks, E. L., and Lopez-Marcano, S. (2021). Annotated video footage for automated identification and counting of fish in unconstrained marine environments. 13, 14

Dittman, A. and Quinn, T. (1996). Homing in pacific salmon: mechanisms and ecological basis. *Journal of Experimental Biology*, 199(1):83–91. 72

Dong, H., Wang, T., Liu, J., and Zhang, C. (2022). Low-rank modular reinforcement learning via muscle synergy. *arXiv preprint arXiv:2210.15479*. 68, 91

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. *arXiv:1711.03938*. 14

Du, S., Tao, Y., and Martinez, A. M. (2014). Compound facial expressions of emotion. *Proceedings of the National Academy of Sciences*, 111(15):E1454–E1462. 28

Ejike, C. and Schreck, C. B. (1980). Stress and social hierarchy rank in coho salmon. *Transactions of the American Fisheries Society*, 109(4):423–426. 11

Ekman, P. and Friesen, W. V. (1978). *Manual for the Facial Action Coding System*. Consulting Psychologists Press. 9

Faloutsos, P., Van de Panne, M., and Terzopoulos, D. (2001). Composable controllers for physics-based character animation. In *Proceedings of the 28th Annual Conference on Computer Craphics and Interactive Techniques*, pages 251–260. 8

Fernö, A., Huse, I., Juell, J.-E., and Bjordal, Å. (1995). Vertical distribution of atlantic salmon (salmo solar l.) in net pens: trade-off between surface light avoidance and food attraction. *Aquaculture*, 132(3-4):285–296. 11, 12, 73

Folkedal, O., Stien, L. H., Nilsson, J., Torgersen, T., Fosseidengen, J. E., and Oppedal, F. (2012). Sea caged atlantic salmon display size-dependent swimming depth. *Aquatic Living Resources*, 25(2):143–149. 12, 71

Føre, M., Dempster, T., Alfredsen, J. A., Johansen, V., and Johansson, D. (2009). Modelling of atlantic salmon (salmo salar l.) behaviour in sea-cages: A lagrangian approach. *Aquaculture*, 288(3-4):196–204. 11, 12, 37, 39, 46, 71, 72

Gallegos, C. L. and Moore (2000a). Factors contributing to water-column light attenuation. *Chesapeake Bay submerged aquatic vegetation water quality and habitat-based requirements and restoration targets: a second technical synthesis*, pages 35 – 55. 13

Gallegos, C. L. and Moore, K. (2000b). Factors contributing to water-column light attenuation. *Chesapeake bay submerged aquatic vegetation water quality and habitat-based requirements and restoration targets: A second technical synthesis*. 74

Ghorbani, S., Mahdaviani, K., Thaler, A., Kording, K., Cook, D. J., Blohm, G., and Troje, N. F. (2020). Movi: A large multipurpose motion and video dataset. *arXiv:2003.01888*. 14

Glimcher, P. W. (2011). Understanding dopamine and reinforcement learning: the dopamine reward prediction error hypothesis. *Proceedings of the National Academy of Sciences*, 108(Supplement 3):15647–15654. 10

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR. 59, 61

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969. 47

Herbert-Read, J. E., Perna, A., Mann, R. P., Schaerf, T. M., Sumpter, D. J., and Ward, A. J. (2011). Inferring the rules of interaction of shoaling fish. *Proceedings of the National Academy of Sciences*, 108(46):18726–18731. 12, 36, 70

Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. (1995). Animating human athletics. In *Proceedings of the 22nd Annual Conference on Computer graphics and Interactive Techniques*, pages 71–78. 8

Hodgkin, A. and Huxley, A. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544. 73

Honryo, T., Kurata, M., Sandval, D., Yamao, S., Cano, A., and Sawada, Y. (2018). Effect of water temperature and light intensity on swim bladder inflation and growth of red sea bream pagrus major larvae. *Fisheries science*, 84(3):553–562. 12, 37, 39

Hoobler, N. (2016). Fast, flexible, physically-based volumetric light scattering. In *Game Developers Conference*, page 608. 75

Huse, I. and Holm, J. (1993). Vertical distribution of atlantic salmon (salmo salar) as a function of illumination. *Journal of Fish Biology*, 43:147–156. 11, 12, 37, 72

Huth, A. and Wissel, C. (1992). The simulation of the movement of fish schools. *Journal of theoretical biology*, 156(3):365–385. 40

Hvas, M., Folkedal, O., and Oppedal, F. (2021). Fish welfare in offshore salmon aquaculture. *Reviews in Aquaculture*, 13(2):836–852. 12, 71

Hwang, H., Jang, C., Park, G., Cho, J., and Kim, I.-J. (2021). Eldersim: A synthetic data generation platform for human action recognition in eldercare applications. *IEEE Access*, PP:1–1. 14

Ichim, A.-E., Kadleček, P., Kavan, L., and Pauly, M. (2017). Phace: Physics-based face modeling and animation. *ACM Transactions on Graphics (TOG)*, 36(4):1–14. 68

Ip, H. H.-S., Chan, S. C., and Lam, M. S. (1998). Hacs: Hand action coding system for anatomy-based synthesis of hand gestures. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 2, pages 1207–1212. IEEE. 68

Isaac, J. H., Manivannan, M., and Ravindran, B. (2022). Single shot corrective cnn for anatomically correct 3d hand pose estimation. *Frontiers in Artificial Intelligence*, 5. 68

Ishikawa, T., Morishima, S., and Terzopoulos, D. (1998). 3D estimation of facial muscle parameter from the 2D marker movement using neural network. In *Proc. Third Asian Conference on Computer Vision*, Lecture Notes in Computer Science, Vol 1352., pages 671–678. Springer. 9

Ishikawa, T., Morishima, S., and Terzopoulos, D. (2000). 3D face expression estimation and generation from 2D image based on a physically constraint model. *IEICE Transactions on Information and Systems*, 83(2):251–258. 9

Ishiwaka, Y., Zeng, X. S., Eastman, M. L., Kakazu, S., Gross, S., Mizutani, R., and Nakada, M. (2021). Foids: Bio-inspired fish simulation for generating synthetic datasets. *ACM Transactions on Graphics (TOG)*, 40(6):1–15. 3, 5, 46

Ishiwaka, Y., Zeng, X. S., Ogawa, S., Westwater, D. M., Tone, T., and Nakada, M. (2022). Deepfoids: Adaptive bio-inspired fish simulation with deep reinforcement learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*. 6

Jensen, A. J., Johnsen, B. O., and Saksgård, L. (1989). Temperature requirements in atlantic salmon (salmo salar), brown trout (salmo trutta), and arctic char (salvelinus alpinus) from hatching to initial feeding compared with geographic distribution. *Canadian Journal of Fisheries and Aquatic Sciences*, 46(5):786–789. 11, 12, 39, 71

Jiang, Y., Ma, Z., and Zhang, D. (2019). Flow field perception based on the fish lateral line system. *Bioinspiration & biomimetics*, 14(4):041001. 40

Juell, J.-E. (1995). The behaviour of atlantic salmon in relation to efficient cage-rearing. *Reviews in fish biology and fisheries*, 5(3):320–335. 11, 12, 70

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., et al. (2018). Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*. 46

Kanade, T., Cohn, J., and Tian, Y. (2000). Comprehensive database for facial expression analysis. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 46–53. 27

Kawai, Y. and Wada, A. (2007). Diurnal sea surface temperature variation and its impact on the atmosphere and ocean: A review. *Journal of oceanography*, 63(5):721–744. 37, 38

Kohbara, J., Hidaka, I., Matsuoka, F., Osada, T., Furukawa, K., Yamashita, M., and Tabata, M. (2003). Self-feeding behavior of yellowtail, seriola quinqueradiata, in net cages: diel and seasonal patterns and influences of environmental factors. *Aquaculture*, 220(1-4):581–594. 12, 37, 39

Kopelevich, O. (1983). Small-parameter model of optical properties of sea water. *Ocean optics*, 1:208–234. 75

Korn, H. and Faber, D. S. (2005). The mauthner cell half a century later: a neurobiological model for decision-making? *Neuron*, 47(1):13—28. 12, 73

Krause, J., Godin, J.-G. J., and Brown, D. (1996). Phenotypic variability within and between fish shoals. *Ecology*, 77(5):1586–1591. 12, 71

Lee, J., Won, J., and Lee, J. (2018). Crowd simulation by deep reinforcement learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, MIG '18, New York, NY, USA. Association for Computing Machinery. 11

Lee, S., Park, M., Lee, K., and Lee, J. (2019). Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG)*, 38(4):1–13. 8

Lee, S.-H., Sifakis, E., and Terzopoulos, D. (2009). Comprehensive biomechanical modeling and simulation of the upper body. *ACM Transactions on Graphics (TOG)*, 28(4):1–17. 8

Lee, S.-H. and Terzopoulos, D. (2006). Heads up! biomechanical modeling and neuromuscular control of the neck. *ACM Transactions on Graphics*, 25(3):1188–1198. 5, 8, 10, 15, 19, 20, 22

Lee, Y., Terzopoulos, D., and Waters, K. (1995). Realistic modeling for facial animation. In *Proc. ACM SIGGRAPH 95 Conference*, pages 55–62. 8, 10, 15, 19, 21

Lindsey, C. C. (1978). 1 - form, function, and locomotory habits in fish. *Fish Physiology*, 7:1–100. 11

Livingstone, S. R. and Russo, F. A. (2018). The Ryerson audio-visual database of emotional speech and song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PLOS ONE*, 13(5):1–35. 27

Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., and Matthews, I. (2010). The extended Cohn-Kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *Proc. IEEE CVPR Workshops*, pages 94–101. 27

Macaulay, G., Wright, D., Oppedal, F., and Dempster, T. (2020). Buoyancy matters: Establishing the maximum neutral buoyancy depth of atlantic salmon. *Aquaculture*, 519:734925. 11

Marras, S., Killen, S. S., Lindström, J., McKenzie, D. J., Steffensen, J. F., and Domenici, P. (2015). Fish swimming in schools save energy regardless of their spatial position. *Behavioral ecology and sociobiology*, 69(2):219–226. 43, 70

Matthews, O., Ryu, K., and Srivastava, T. (2020). Creating a large-scale synthetic dataset for human activity recognition. *arXiv:2007.11118*. 14

Messias, J. P., Paula, J. R., Grutter, A. S., Bshary, R., and Soares, M. C. (2016a). Dopamine disruption increases negotiation for cooperative interactions in a fish. *Scientific reports*, 6(1):1–8. 11

Messias, J. P., Santos, T. P., Pinto, M., and Soares, M. C. (2016b). Stimulation of dopamine d1 receptor improves learning capacity in cooperating cleaner fish. *Proceedings of the Royal Society B: Biological Sciences*, 283(1823):20152272. 11

Miller, T. H., Clements, K., Ahn, S., Park, C., Hye Ji, E., and Issa, F. A. (2017). Social status–dependent shift in neural circuit activation affects decision making. *Journal of Neuroscience*, 37(8):2137–2148. 11, 12, 73

Min, S., Won, J., Lee, S., Park, J., and Lee, J. (2019). Softcon: Simulation and control of soft-bodied animals with biomimetic actuators. *ACM Transactions on Graphics (TOG)*, 38(6):1–12. 11

Mobley, C. D. (1994). *Light and Water: Radiative Transfer in Natural Waters*. Academic. 75

Montague, P. R. and Berns, G. S. (2002). Neural economics and the biological substrates of valuation. *Neuron*, 36(2):265–284. 10

Nakada, M. (2017). *Deep Learning of Neuromuscular and Sensorimotor Control with Biomimetic Perception for Realistic Biomechanical Human Animation*. University of California, Los Angeles. 86

Nakada, M., Zhou, T., Chen, H., Weiss, T., and Terzopoulos, D. (2018). Deep learning of biomimetic sensorimotor control for biomechanical human animation. *ACM Transactions on Graphics*, 37(4):1–15. 8, 86

Nevitt, G. A., Dittman, A. H., Quinn, T. P., and Moody, W. J. (1994). Evidence for a peripheral olfactory memory in imprinted salmon. *Proceedings of the National Academy of Sciences*, 91(10):4288–4292. 72

Ng-Thow-Hing, V. (2001). *Anatomically-based models for physical and geometric reconstruction of humans and other animals*. PhD thesis, Department of Computer Science, University of Toronto. 18, 84

Odling-Smee, L. and Braithwaite, V. A. (2003). The role of learning in fish orientation. *Fish and Fisheries*, 4(3):235–246. 11, 12, 72

Olariu, S. and Zomaya, A. Y. (2005). *Handbook of bioinspired algorithms and applications*. CRC Press. 1

Oppedal, F., Dempster, T., and Stien, L. H. (2011). Environmental drivers of atlantic salmon behaviour in sea-cages: a review. *Aquaculture*, 311(1-4):1–18. 12, 70, 72

Pavlov, D. and Kasumyan, A. (2000). Patterns and mechanisms of schooling behavior in fish: a review. *Journal of Ichthyology*, 40(2):S163. 70

Podila, S. and Zhu, Y. (2017a). Animating escape maneuvers for a school of fish. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '17, New York, NY, USA. Association for Computing Machinery. 10

Podila, S. and Zhu, Y. (2017b). Animating multiple escape maneuvers for a school of fish. In *Proceedings of Graphics Interface 2017*, GI 2017, pages 140 – 147. Canadian Human-Computer Communications Society / Société canadienne du dialogue humain-machine. 10

Rajagopal, A., Dembia, C. L., DeMers, M. S., Delp, D. D., Hicks, J. L., and Delp, S. L. (2016). Full-body musculoskeletal model for muscle-driven simulation of human gait. *IEEE transactions on biomedical engineering*, 63(10):2068–2079. 8

Reyes, B. A., Pendergast, J. S., and Yamazaki, S. (2008). Mammalian peripheral circadian oscillators are temperature compensated. *Journal of Biological Rhythms*, 23(1):95–98. PMID: 18258762. 73

Reynolds, C. (1999). Steering behaviors for autonomous characters. 10

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34. 3, 10, 36

Reynolds, W. W. and Casterlin, M. E. (1979). Behavioral thermoregulation and the "final preferendum" paradigm. *American zoologist*, 19(1):211–224. 11, 12, 71

Ristroph, L., Liao, J. C., and Zhang, J. (2015). Lateral line layout correlates with the differential hydrodynamic pressure on swimming fish. *Physical Review Letters*, 114(1):018102. 40

Sachdeva, P., Sueda, S., Bradley, S., Fain, M., and Pai, D. K. (2015). Biomechanical simulation and control of hands and tendinous systems. *ACM Transactions on Graphics (TOG)*, 34(4):1–10. 8

Sakakura, Y. and Tsukamoto, K. (1998a). Effects of density, starvation and size difference on aggressive behaviour in juvenile yellowtails (seriola quinquevadiata). *Journal of Applied Ichthyology*, 14(1-2):9–13. 40

Sakakura, Y. and Tsukamoto, K. (1998b). Social rank in schools of juvenile yellowtail, seriola quinqueradiata. *Journal of Applied Ichthyology*, 14(1-2):69–73. 11, 58

Saleh, A., Laradji, I. H., Konovalov, D. A., Bradley, M., Vazquez, D., and Sheaves, M. (2020). A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis. *Scientific Reports*, 10(1). 13

Satoi, D., Hagiwara, M., Uemoto, A., Nakadai, H., and Hoshino, J. (2016). Unified motion planner for fishes with various swimming styles. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 35(4). 10, 11

Scherer, K. R., Ellgring, H., Dieckmann, A., Unfried, M., and Mortillaro, M. (2019). Dynamic facial expression of emotion and observer inference. *Frontiers in Psychology*, 10:508. 28

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015a). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR. 43

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015b). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*. 35, 44, 46

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. 35, 43

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599. 10

Si, W., Lee, S.-H., Sifakis, E., and Terzopoulos, D. (2014). Realistic biomechanical simulation and control of human swimming. *ACM Transactions on Graphics (TOG)*, 34(1):1–15. 8

Sifakis, E., Neverov, I., and Fedkiw, R. (2005). Automatic determination of facial muscle activations from sparse motion capture marker data. In *Proceedings of ACM SIGGRAPH 2005*, pages 417–425. 9, 15

Sinha, A., Choi, C., and Ramani, K. (2016). Deephand: Robust hand pose estimation by completing a matrix imputed with deep features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4150–4158. 68

Skinner, B. F. (1956). A case history in scientific method. *American psychologist*, 11(5):221. 10

Solonenko, M. and Mobley, C. D. (2015a). Inherent optical properties of jerlov water types. *Applied optics*, 54 17:5392–401. 13

Solonenko, M. G. and Mobley, C. D. (2015b). Inherent optical properties of jerlov water types. *Applied optics*, 54(17):5392–5401. 74, 75

Stephens, K., Pham, B., and Wardhani, A. (2003). Modelling fish behaviour. In *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, GRAPHITE '03, page 71–78, New York, NY, USA. Association for Computing Machinery. 10

Stuart, K. R. and Drawbridge, M. (2011). The effect of light intensity and green water on survival and growth of cultured larval california yellowtail (seriola lalandi). *Aquaculture*, 321(1-2):152–156. 37

Sueda, S., Kaufman, A., and Pai, D. K. (2008). Musculotendon simulation for hand animation. In *ACM SIGGRAPH 2008 papers*, pages 1–8. 8

Sun, F., Zeng, J., Jing, M., Zhou, J., Feng, J., Owen, S. F., Luo, Y., Li, F., Wang, H., Yamaguchi, T., et al. (2018). A genetically encoded fluorescent sensor enables rapid and specific detection of dopamine in flies, fish, and mice. *Cell*, 174(2):481–496. 11

Sunehag, P., Lever, G., Liu, S., Merel, J., Heess, N., Leibo, J. Z., Hughes, E., Eccles, T., and Graepel, T. (2019). Reinforcement learning agents acquire flocking and symbiotic behaviour in simulated ecosystems. In *ALIFE 2019: The 2019 Conference on Artificial Life*, pages 103–110. MIT Press. 11

Sutterlin, A., Jokola, K., and Holte, B. (1979). Swimming behavior of salmonid fish in ocean pens. *Journal of the Fisheries Board of Canada*, 36(8):948–954. 72

Swanson, R., Livingstone, S., and Russo, F. (2019). RAVDESS facial landmark tracking (version 1.0.0) [dataset]. 27

Tarling, P., Cantor, M., Clapés, A., and Escalera, S. (2021). Deep learning with self-supervision and uncertainty regularization to count fish in underwater images. *arXiv:2104.14964*. 14

Terzopoulos, D. and Waters, K. (1993). Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):569–579. 9

Thies, J., Zollhofer, M., Stamminger, M., Theobalt, C., and Nießner, M. (2016). Face2face: Real-time face capture and reenactment of RGB videos. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2387–2395. 9

Tu, X. and Terzopoulos, D. (1994). Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, page 43–50, New York, NY, USA. Association for Computing Machinery. 10, 11

Van den Bogert, A. J., Geijtenbeek, T., Even-Zohar, O., Steenbrink, F., and Hardin, E. C. (2013). A real-time system for biomechanical analysis of human movement and muscle function. *Medical and Biological Engineering and Computing*, 51(10):1069–1077. 8

Van Nierop, O. A., van der Helm, A., Overbeeke, K. J., and Djajadiningrat, T. J. (2008). A natural human hand model. *The Visual Computer*, 24(1):31–44. 83

Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. (2017). Learning from synthetic humans. In *Proceedings of the IEEE CVPR Conference*. 14

Verma, S., Novati, G., and Koumoutsakos, P. (2018). Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 115(23):5849–5854. 11

Wang, Q., Liu, H., Gao, K., and Zhang, L. (2019). Improved multi-agent reinforcement learning for path planning-based crowd simulation. *IEEE Access*, 7:73841–73855. 11

Waters, K. (1987). A muscle model for animation three-dimensional facial expression. *Proc. ACM SIGGRAPH 87 Conference*, 21(4):17–24. 9

Weise, T., Bouaziz, S., Li, H., and Pauly, M. (2011). Realtime performance-based facial animation. *ACM Transactions on Graphics*, 30(4):1–10. 9

Wiersma, C. and Ikeda, K. (1964). Interneurons commanding swimmeret movements in the crayfish, procambarus clarki (girard). *Comparative Biochemistry and Physiology*, 12(4):509–525. 12, 73

Wu, T., Hung, A., and Mithraratne, K. (2014). Generating facial expressions using an anatomically accurate biomechanical model. *IEEE Transactions on Visualization and Computer Graphics*, 20(11):1519–1529. 15

Xu, F., Chai, J., Liu, Y., and Tong, X. (2014). Controllable high-fidelity facial performance transfer. *ACM Transactions on Graphics*, 33(4):1–11. 9

Yan, W. (2012). *Biomechanical Simulation of the Human Hand and Forearm*. PhD thesis, Citeseer. 83, 84, 85

Yu, H., Liu, B., Wang, C., Liu, X., Lu, X.-Y., and Huang, H. (2022). Deep-reinforcement-learning-based self-organization of freely undulatory swimmers. *Phys. Rev. E*, 105:045105. 11

Zeng, X. S., Dwarakanath, S., Lu, W., Nakada, M., and Terzopoulos, D. (2021a). Facial expression transfer from video via deep learning. In *The ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–2. 4

Zeng, X. S., Dwarakanath, S., Lu, W., Nakada, M., and Terzopoulos, D. (2021b). Neuromuscular control of the face-head-neck biomechanical complex with learning-based expression transfer from images and videos. In *International Symposium on Visual Computing*, pages 116–127. Springer. 5

Zhang, J., Chen, K., and Zheng, J. (2020). Facial expression retargeting from human to avatar made easy. *IEEE Transactions on Visualization and Computer Graphics*. 9

Zhu, P., Dai, W., Yao, W., Ma, J., Zeng, Z., and Lu, H. (2020). Multi-robot flocking control based on deep reinforcement learning. *IEEE Access*, 8:150397–150406. 11

Zhu, Y., Tian, F.-B., Young, J., Liao, J. C., and Lai, J. C. (2021). A numerical study of fish adaption behaviors in complex environments with a deep reinforcement learning and immersed boundary–lattice boltzmann method. *Scientific Reports*, 11(1):1–20. 11

Zottoli, S. J. (1978). Comparison of mauthner cell size in teleosts. *Journal of Comparative Neurology*, 178(4):741–757. 73