LEARNING SWIMMING GAITS AND NAVIGATION ON A BIOMIMETIC MARINE ROBOT

HAMEED IMRAN

PhD

The Hong Kong Polytechnic University

2023

The Hong Kong Polytechnic University
Department of Mechanical Engineering

Learning Swimming Gaits and Navigation on a Biomimetic Marine Robot

Hameed Imran

A thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

April 2023

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

___HAMEED Imran_____ (Name)

ABSTRACT

Marine ecosystem contains enormous resources for mankind in the form of food, fuel, oxygen, minerals etc. It also contains answers to a lot of questions related to the inception of life and subsequent evolution. Ocean exploration is thus very beneficial to mankind. Marine robotics is one way to explore aquatic environments.

Biologists have been studying marine life for centuries. They have classified organisms into several categories. One ubiquitous category is Body and Caudal Fin (BCF) swimmers. They are abundantly found in all regions (oceans, rivers, ponds etc.). They have further subtypes which swim with different swimming gaits. All the gaits have their own features and benefits.

In bio-inspired robotics, inspiration is drawn from nature for design, and locomotion strategies. We find a lot of studies to develop fish-like robots in which a certain organism is mimicked in terms of design and with respect to swimming gait. Most studies revolve around a particular organism to develop and study its gait and characteristics. However, multiple gait patterns can be utilized together on a single biomimetic design to employ collective benefits. In literature, we find that there is a lack of a unified control scheme that can be used to optimize and mimic undulatory patterns observed among different organisms in the Body and/or Caudal Fin (BCF) category.

Therefore, in this study, a generic biomimetic marine robot BCFbot (Body and Caudal Fin Robot) is conceived to study BCF locomotion. The robot, owing to having a generic design, can adopt multiple motion patterns related to different kinds of natural organisms in the BCF category. Moreover, a bio-inspired control scheme is proposed to develop swimming gaits for the robot. The schematics incorporate central pattern generators in a deep reinforcement learning (DRL) framework which allows the framework to develop/optimize distinct motion patterns and switch between them seamlessly and instantly.

The first part of the study focuses on developing three gaits (namely anguilliform, sub-carangiform, and carangiform). After development of these gaits, a thorough comparison is made between them by testing the gaits through simulation and on the hardware. In depth testing and comparison reveal different benefits associated with different gaits. One gait is found to be the fastest, another to be the most energy economical and one to be multi-modal. Hence, the benefits of three benchmark motion patterns can be well realized with the developed robot and can be freely switched and optimized with the developed DRL mechanism. This should be the first attempt for achieving a multi-modal pattern optimization and switching within a single BCFbot and demonstrating a successful motion generation regime similar to real animals. Oscillators integrated into a learning paradigm provide a bioinspired framework to systematically develop a variety of swimming gaits.

The second part of the study focuses on on-surface navigation of the robot along desired trajectories. To realize this, dynamic motion primitives, which can represent a large range of motion behaviors, are combined with a decoupled reinforcement learning framework. The proposed architecture optimizes the motion primitives first to develop a travelling wave undulation pattern in the tail and then to navigate the robot along different predefined paths. Through this framework, effective swimming gait emerges, and the robot is able to navigate well on the surface of water. This framework combines the optimization potential of deep reinforcement learning with stability and generalization properties of dynamic motion primitives. The method is trained and tested on a simulated model of the robot to demonstrate the effectiveness of the method and conduct experimental testing on the real robot to verify the results.

In the third part, the design and schematics are modified to perform 3D underwater locomotion and navigation. In literature, for depth control, mostly pectoral fins are employed which do not fall into the BCF category. In this part, the same caudal fin used in previous parts for planar locomotion is used to perform underwater locomotion by adding extra pitch joints to the robot. The robot's ability to perform underwater locomotion is tested thoroughly by a series of experiments performed via teleoperation. Testing results reveal that the modified design and the architecture can well realize three-dimensional underwater locomotion. Finally, 3D path-following along desired trajectories is also realized in simulation.

LIST OF PUBLICATIONS

As first author:

1. **Imran Hameed**, Xu Chao, David Navarro-Alarcon, and Xingjian Jing. Deep Reinforcement Learning Enabling a BCFbot to Learn Various Undulatory Patterns. Submitted to *IEEE Transactions on Systems, Man & Cybernetics Systems* (IEEE SMC) (Under Review).
2. **Imran Hameed**, Xu Chao, David Navarro-Alarcon, and Xingjian Jing. Autonomous Underwater Navigation of BCFbot using Deep Reinforcement Learning. To be submitted to *IEEE Transactions on Robotics* (IEEE TRO).
3. **Imran Hameed**, Xu Chao, David Navarro-Alarcon, and Xingjian Jing. Training Dynamic Motion Primitives using Deep Reinforcement Learning to Control a Robotic Tadpole. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS 2022), Kyoto, Japan, 2022, pp. 6881-6887

As a co-author:

4. Xu Chao, **Imran Hameed**, David Navarro-Alarcon, and Xingjian Jing. Performance Oriented Understanding & Design of a Robotic Tadpole: Lower Energy Cost, Higher Speed. Submitted to Soft Robotics (*SoRo*) (Under Review).
5. Zhengchao Li, Xu Chao, **Imran Hameed**, Jianan Li, Wen Zhao, and Xingjian Jing. Biomimetic omnidirectional multi-tail underwater robot. *Mechanical Systems and Signal Processing* (MSSP), vol. 173, p. 109056, 2022.

ACKNOWLEDGMENTS

This thesis is being submitted in partial fulfilment of my PhD studies at the Hong Kong Polytechnic University (PolyU). I would like to thank the University for the funding it rendered throughout my studies and the facilities it provided that enabled the research work carried out throughout the course of PhD.

I would like to thank for the support from the Laboratory of Nonlinear Dynamics, Vibration, and Control, Department of Mechanical Engineering, City University of Hong Kong, for most research and development tasks of this thesis.

I would like to express my gratitude to my supervisor Prof. JING Xingjian for his continuous support and guidance during the research work. I appreciate his profound knowledge and deep insight into the research topics and his assistance on completion of this thesis. I would like to thank Dr. David Navarro-Alarcon for his support and help through the course of my PhD.

I would like to thank my group members in general, and Mr. Xu Chao in particular, for his help with the hardware and the countless hours he devoted to help perform experiments in the laboratory.

Finally, I am grateful to my family members, especially my parents for their endless support, affection, and prayers.

List of Contents

List of Figures

List of Tables and Algorithms

List of Abbreviations

Abbreviation	Expansion
AUV	Autonomous Underwater Vehicles
ROV	Remotely Operated Vehicles
BCF	Body and/or Caudal Fin
MPF	Median and/or Pectoral Fin
BCFbot	Body and Caudal Fin Robot
DNN	Deep Neural Network
MLP	Multi-Layer Perceptron
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
DDPG	Deep Deterministic Policy Gradient
TRPO	Trust Region Policy Optimization
PPO	Proximal Policy Optimization
KL	Kullback Liebler
CPG(s)	Central Pattern Generator(s)
DMP(s)	Dynamic Motion Primitive(s)
COT	Cost of Transport

COTu	Cost of Turning
LWR	Linear Weighted Regression
MuJoCo	Multi-Joint dynamics with Contact
ACM	Active Chord Mechanism
RPi	Raspberry Pi
FPS	Frames Per Second
3D	Three Dimensional
DoF(s)	Degree(s) of Freedom

CHAPTER 1

INTRODUCTION

1.1 Marine Robotics – Need and Challenges

Human beings have been exploring oceans for centuries. The efforts were initially thwarted because of lack of specialized instruments such as oxygen containers and hence, humans were able to investigate only up to a certain depth. With the inventions of special vessels (like submarines), it became easy to survive underwater for longer time. Since then, thousands of underwater creatures and organisms have been discovered and are still being discovered [18-20].

More than 70% of the earth's surface is covered by water. Oceans contain resources which are directly or indirectly beneficial to human beings. Only a small portion of those resources has been tapped until now. Oil rigs work round the clock to extract crude oil eventually used as fuel for commercial vehicles and in industries. Similarly, it contains different minerals. For human consumption, it provides a huge contribution in the form of sea food (fishes, plants, weeds etc.). With advancements in water purification techniques, sea water can also be used for human consumption and irrigation after due treatment [21].

Oceans provide an excellent environment for aquaculture, also known as aquafarming [22]. In aquaculture, different species of fish are raised and bred in very big facilities built in rivers, dams, and oceans. Mostly, it is used for farming purposes, but it can also be used to provide suitable habitat for endangered species for conservation. Aquaculture provides a promising avenue towards food security [22].

Other than the above benefits, the ocean contains answers to a lot of questions regarding life, evolution, and history of the planet. In research and exploration, underwater archaeology [23] plays an important role. Other than fuel, the ocean provides energy solutions in the form of wind energy, and waves energy [24]. It, therefore, has a lot of potential in energy harvesting.

For each application, different exploration tools and techniques are required. Some applications like large-scale fishing, surface ships or vessels are used. Oil extraction requires a complete rig to be built once substantial presence of crude is detected. For underwater applications like exploration and archaeology, researchers need to observe certain phenomena (e.g., hydrothermal vents) or deep-sea remains and creatures. For this purpose, small-scale (manned) submarines or (unmanned) autonomous underwater vehicles (AUVs) or remotely operated vehicles (ROVs) fitted with surveillance equipment are used. Such unmanned platforms come in the class of marine robotics. AUVs or ROVs provide promising solution to ocean exploration. These “mobile marine robots” can carry a range of sensors such as temperature, chemical, salinity, vision etc. to great depths. Moreover, they can carry sampling apparatus [25]. Using such apparatus, samples are collected using syringe like modules. These samples are later analyzed in laboratories for different investigations such as presence of microbial life, discovery of new species etc.

Marine robots are mainly divided into two categories – propeller-based robots relying on thrusters [26], and bioinspired designs mimicking marine creatures [27] (Fig. 1.0).

Bioinspired marine robots have undulating and/or oscillating surfaces inspired by marine counterparts they mimic. They make use of tentacles [28, 29], fins [8, 15, 30-32], membranes [33, 34], flappers [35-38] etc., to produce propulsion.

1.1.1 Challenges in Marine Environment

The marine environment poses more challenges as compared to terrestrial exploration. A few of them are presented below,

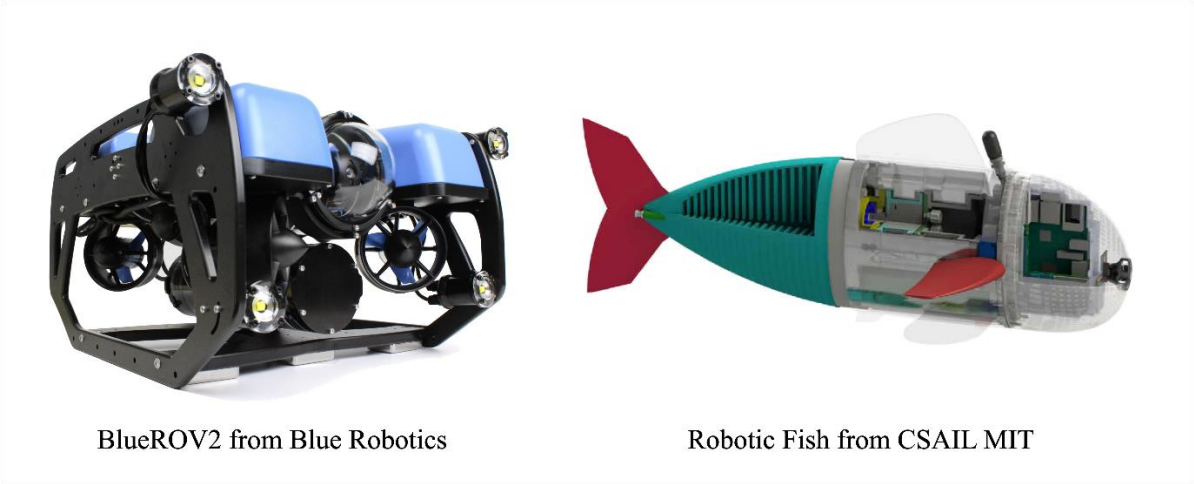
1) The first and the main challenge is related to design. In marine environment, water proofing needs utmost attention. All robots (marine or not) require power for operation – motion, perception, and communication. All these operations require power (usually from batteries), electronic circuits, microcontrollers, and actuators. These electronic devices including batteries need to be safely waterproofed before they can be put into water.
2) Next big challenge is communication. The common wireless communication methods such as *Wi-Fi* and *Bluetooth* do not work underwater. Many commercially available ROVs are tethered. A wire is used for communication (and sometimes even for power for long duration testing).
3) Satellite positioning systems such as GPS (Global Positioning System) etc. also fail to deliver underwater since they work on radio waves. Some alternatives [39] that combine such positioning systems with acoustic modems provide good but expensive solution for underwater positioning.
4) The next challenge is sensing. Although vision-based cameras can be used, but since these cameras depend on visible light, they can only work up to a certain depth after which a dedicated light source is required. Traditional laser-based range sensors such as 2D or 3D laser scanners [40] cannot work underwater. Some depth sensors come coupled with infrared sensing [41] whereas some of them use

triangulation on stereo images [42]. Only the latter can be used in water although there are not many applications.

5) In the underwater environment, acoustic signals travel for long distances. Hence, acoustics is used for communication as well as sensing. Acoustic modems can communicate wirelessly underwater. Similarly, underwater acoustic sonars can sense the surroundings for obstacles. These modems send and receive sound signals of varying frequency to communicate but since the speed of sound is very slow as compared to radio waves. The bandwidth of acoustic underwater communication and feedback rate of acoustic sonars is very low.
6) Ocean environment is highly inconsistent and dynamic. This makes the overall control of marine robots much more challenging than terrestrial robots. Complexity of fluid dynamics compared to rigid body dynamics makes modelling and control difficult. Underwater ocean currents are extremely strong and pose a great challenge to both design and control.
7) Using cameras (underwater) adds a peculiar tint to the video recordings. Such tint is different in different areas according to the color of underwater environment. To use the video stream for feedback or other purposes, sometimes it is required to post-process them heavily.
8) For the terrestrial/aerial case, the air pressure decreases with increase in altitude. Underwater, the pressure increases with the increase in depth. This makes the design, especially in biomimetic cases, very challenging. The range of materials that can be used becomes very narrow.
9) Underwater acoustic modems and sonars are big in size. It is difficult to accommodate them on small scale marine robots.

1.2 Screw-based Propellers or Biomimetics

Marine Robotics always accompanies a question. Why do we need biomimetic designs when propellers are cheap, easy to control, and can provide large propulsive force?



BlueROV2 from Blue Robotics Robotic Fish from CSAIL MIT

Fig. 1.0: (*Left*) Propeller based ROV – *BlueROV2* from *BlueRobotics* [3]. (*Right*) A bio-inspired robotic fish – *SoFi* developed by *CSAIL MIT* [8].

Most AUVs and ROVs are of (screw-based/rotary) thruster or propeller type. A thruster is a rotary propeller operated using a motor. Rotary propellers, although very useful, have some disadvantages as well. Some of them are mentioned below,

1) Propellers create high frequency vibrations in water. These vibrations cause significant noise in marine environment that may repel marine life. This can make exploration of marine life difficult. This is common to all types of propellers ranging from the ones on small AUVs to the ones on big ships.
2) Many marine creatures communicate using acoustics. Noise caused by propellers may cause disruption which may affect behavioral studies related to marine life.
3) Propellers contain blades that rotate at high angular speed due to which there is an unsafe region in the vicinity of a propeller. The region around the rover/propeller may be dangerous and may cause harm to marine life. We see many cases of such dangerous interactions between marine fauna and ships.

4) For small scale propellers made for marine robots, the blades are made of plastic. If the water the robot is moving in is not clean, hard foreign waste (e.g., plastic or any other material) may come in contact with the blades of a fast-rotating propeller. The blades may break thereby causing damage to the propeller. Soft waste (e.g., polythene bags) can get stuck in the blades thereby blocking the passage of water, essentially rendering the propeller useless. This may also happen with soft and long aquatic weeds and plants.
5) For rovers operating very deep in the ocean, they should be able to withstand very high pressure. Biomimetic designs can make use of soft materials which are light weight, can withstand high pressure and can go to virtually any depth. For example, a recent study uses soft material to design a soft robot capable of diving into the Mariana Trench [34]. Traditional screw propellers, however, cannot be manufactured out of soft materials.
6) In case of propellers, either a separate maneuvering surface (e.g., a rudder in case of ships) is required or more than one propulsors are required to turn the rover/platform. Whereas in the case of biomimetic designs, a single propulsor can propel as well as maneuver.

Robots inspired by marine ecosystem, i.e., biomimetic marine robots, provide answers to many of the above problems. These robots do not render noise in the ecosystem since the frequency of undulation or oscillation is usually very low. These robots do not have very fast-moving parts and there is no strong cavitation or suction region around them. Hence, they render no threat to marine flora and fauna. Whether these robots are soft or hard, any contact with an external agent, be it some hard/soft waste or aquatic flora, renders no damage to the robot. Similarly, the robot does no damage to marine creatures including plants. A big advantage of biomimetic designs is in that they can be made from soft materials. At great depths

where the water pressure is very high, plastics no longer remain a good choice. Soft materials are ideal for missions that need to be operated very deep in the ocean as they can bear high pressure.

Moreover, the propelling surfaces (i.e., the body of the robots) can house all the circuitry and power resources [43] required for the robot's operation. Thrusters are usually separate attachments to the main body of the robot. When multiple thrusters are used for separate DoFs, they make the robot heavy and bulky.

1.3 Classification

Like terrestrial organisms, there is a great diversity in the marine ecosystem. Similarly, as one could expect, designs of biomimetic marine robots have a great spread. To better grasp bioinspired designs in underwater domain, one should first develop some basic understanding of the classification of marine life. A brief overview of classification with respect to fin types will be presented below. The idea is not to present a biological study, but to give an abstract insight into distinctive categories marine organisms are divided into.

Marine biologists have classified the propulsion phenomena in marine creatures into two broad and distinctive categories – body and/or caudal fin (BCF) propulsion and median and/or paired Fin (MPF) propulsion. Both categories contain several further sub-classes. Fig. 1.1 gives a good overview of both types and leaves less to explain. It is taken in principle from '*Form, function, and locomotory habits*' by *C. C. Lindsey* [1], and is edited for clarity and simplicity. It is a very common reference used in marine robotics literature.

Fig. 1.1a, adapted from [2], shows types and names of fins. 'Caudal fin', as the name suggests, is at the end of the body. Fins on the top and bottom side of the organism are called 'dorsal' and 'anal' fins respectively. Dorsal and anal fins are collectively called median fins.

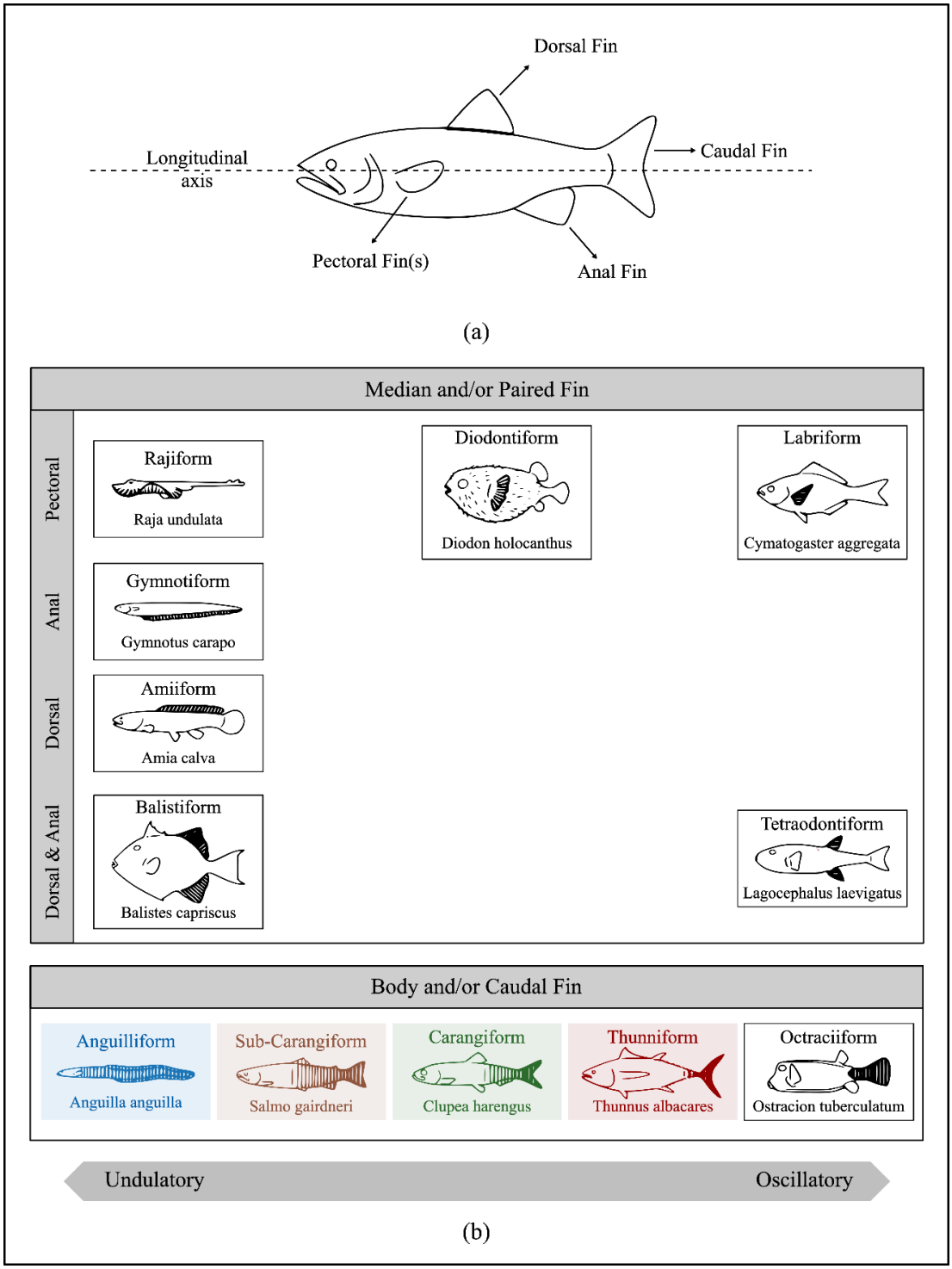


Fig. 1.1: (a) Different fin types possessed by different swimmers adapted from [2] (edited for brevity). (b) Classification of swimmers originally proposed by [1] (edited for brevity) based on fin type (top to bottom) and motion type (undulation/oscillation) (left to right). In each box, a sketch of a sample organism from the respective class is shown with its class name written above it and its biological name shown on bottom. Part of body or fin taking part in undulation or oscillation is shaded by lines.

Fins on the sides are called pectoral fins. Sometimes ‘paired’ is also used instead of ‘pectoral’ since these fins are on both sides of the body. Not all types of swimmers possess all fin-types. Swimmers using **median** (dorsal and/or anal fins) **and pectoral fins** to swim and maneuver are included in the **MPF** category (top box in Fig. 1.1b). Swimmers who mainly use their **body and caudal fin** to generate propulsion are included in the **BCF** category (bottom box in Fig. 1.1b). There exists, however, a great diversity among swimmers in nature. Some swimmers (e.g., shiner perch) possess both caudal and pectoral fins. It can switch between labriform and carangiform modes [1]. Pectoral fins help in precise maneuvering and caudal fins give better forward speed. BCF swimmers are therefore generally faster than MPF swimmers.

To better understand the classes inside the BCF and MPF categories, concepts of undulation and oscillation should first be made clear. Later, these terms will be frequently used in the study since the study revolves around development of different undulatory and oscillatory swimming patterns. There is an undulation-oscillation scale at the bottom of Fig. 1.1b. This scale pertains to both the (MPF and BCF) classes. It shows that the categories mentioned on the left side of the chart perform undulatory motion of their propelling surfaces whereas the ones mentioned on the right side move their propelling surfaces in oscillatory way. Therefore, within both categories (BCF and MPF), the trend changes from undulation (left) to oscillation (right). These terms can also be understood from Fig. 1.2.

Organisms towards the ‘undulation’ side (left of Fig. 1.1b and Fig. 1.2) have very soft and flexible bodies and propelling surfaces. They can wriggle those soft propelling surfaces to generate short wavelength waves into them. These waves are also referred to as ‘travelling waves’ in marine robotics literature. In Fig. 1.1b, the parts of bodies taking part in travelling waves generation are shaded with lines. In the undulation side of the BCF category, these waves

start from the first half of the body and terminate at the caudal fin (also see Fig. 1.2). MPF swimmers render undulatory travelling waves into dorsal, anal, or pectoral fins.

'Oscillatory' pattern means pendulum-like 'to and fro' motion of propelling surface whereby the wavelength of travelling wave is very large. For organisms showing oscillatory motion, the fins are usually too hard such that there is no apparent travelling wave. Organisms towards the oscillation side of the spectrum have stiffer bodies and fins and therefore, small wavelength travelling waves are not possible.

This work is related to the BCF propulsion system. BCF type is further elaborated in Fig. 1.2 (also adapted from [1]). BCF category (bottom of Fig. 1.1a and Fig. 1.2), as mentioned before, generates propulsion with combined use of body and caudal fin. The contribution of body and fin varies within this category. In the anguilliform sub-type (left in Fig. 1.2), the contribution of the body is large compared to the thunniform sub-type (right in Fig. 1.2). For example, Anguilla anguilla (American eel) possesses very soft and flexible body. The end of the body becomes flat to take the shape of a caudal fin [44]. Hence, it uses both the body and

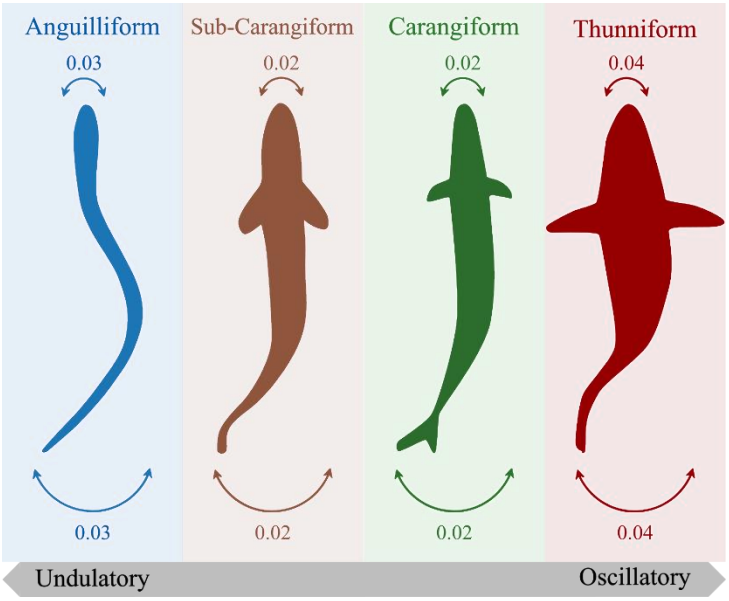


Fig. 1.2: Canonical swimming patterns included in the BCF category. Snapshots are originally from [1] and amplitude information is taken from [6].

the fin to generate propulsion whereas Thunnun albacares (Yellowfin tuna), simply oscillates its fin to generate propulsion [30]. Hence, the contribution of the body varies from the anguilliform pattern (higher) to the carangiform (and thunniform) pattern (lower).

BCF swimmers exhibit different swimming modes (namely anguilliform, sub-carangiform, carangiform, and thunniform) (Fig. 1.2). Classically, they were categorized based on oscillation amplitude of head and tail [1]. But amplitude investigation from a recent study [6] (indicated in Fig. 1.2) shows that the amplitudes of the constituent members are either the same or very slightly different. Although the swimming gaits of BCF swimmers look visibly different, it is difficult to quantitatively differentiate them, and wavelength is one yardstick that can be used for this purpose.

The average wavelength of the anguilliform mode is least among all. An example is Anguilla rostrata (American eel). Swimmers with long and slender bodies like snakes and eels have this motion pattern. Then come swimmers with fusiform shaped bodies. Depending upon the flexibility in their bodies, they show sub-carangiform to carangiform motion pattern. For example, Salvalinus fontinalis (*trout*) show sub-carangiform and Scomber scombrus (*mackerel*) exhibit carangiform pattern. The trend for wavelength increases as we move from anguilliform to sub-carangiform and carangiform swimmers. Hence, the wavelength expands, and the wave number shrinks as we move from the anguilliform to carangiform mode.

The first three patterns (anguilliform, sub-carangiform, and carangiform) are considered in this work. The thunniform and ostraciform patterns not taken into consideration as they require specific design requirements. It should also be noted that the above taxonomy proposes a continuous spectrum rather than discrete classes. The difference between the two neighbors may be small but the extreme ends have pronounced differences. In short, the

spectrum of BCF swimmers range from purely undulatory to purely oscillatory motion with a mixture thereof in between.

There are certain benefits associated with all types of swimming gaits mentioned above. Anguilliform swimmers, for instance, are generally slow but some of them can move forward as well as backward without needing to turn. (e.g., eel). A kinematic comparison between forward and backward swimming modes in eel can be found in [44]. Swimmers from other categories cannot exhibit such multi-modal behavior. As we move to the right hand of the spectrum, we see other features. For example, carangiform swimmers locomote much faster with average linear speed of forward locomotion much higher than anguilliform swimmers e.g., scombroids have been reported to swim at speeds more than 1 m/s [1]. In the bioinspired side of marine robotics, certain motion patterns are adopted on robotic platforms to exploit benefits related to those patterns. For example, the platform in [45] can locomote forward and backward using the anguilliform locomotion pattern and the robotic fish in [15] can swim at high speed using the carangiform locomotion pattern.

1.4 Examples from Robotics

In marine robotics, we can find a large body of work in both (MPF and BCF) categories. The reviews in [27, 46-52] specifically cover the biomimetic aspects of marine robotics. They refer to a large body work done in this field. In this study, it is difficult to cover them all. However, it is tried to cover all the important literature related to this work.

Reproducing motion behaviors related to a particular organism on robotic platforms let researchers study the dynamics related to that organism and compare it with other species. It is difficult to conduct exhaustive studies on live animals. Robotic platforms provide repeatability and ease of experimentation [53]. Practice initially started in the form of swimming machines which were not really mobile robots [54]. Certain propulsion mechanisms like oscillating foils

actuated using large actuators were fixed on a static structure to study propulsion performance and fluid mechanics around oscillating fins and sometimes to perform trajectory stabilization [55] and motion planning [56]. With miniaturization in actuation and energy storage, these machines have started to take the form of mobile marine robots.

This work is pertinent to BCF mode of locomotion. Hence, the MPF category will not be covered in this part. However, some MPF based designs will be discussed later (Sec. 1.8 and Sec. 5.1) when 3D underwater locomotion and navigation will be discussed. Here, the focus will be kept on platforms mimicking the BCF type.

BCF, being easier to mimic, is frequently adopted since the design remains simple. In most cases, a single actuated fin (hard or flexible) at the end of robot's body is enough to realize certain BCF modes (e.g., thunniform in [30]). The actual design, however, depends on which organism or category is being mimicked. [50] reviews robots inspired from BCF swimmers only.

The designs of some efforts regarding reproduction of different swimming patterns on robotic platforms are reviewed in Fig. 1.3. A thorough treatment on mechanics of anguilliform motion pattern [57] and development of locomotion controllers and trajectory tracking of an eel-like robot is shared in [7] (Fig. 1.3b). Anguilliform locomotion controllers are developed for an amphibious snake-like robot in [5] (Fig. 1.3a) . Similarly, anguilliform motion pattern is optimized on a multi-joint snake like robot in [43]. A robot, also multi-link eel-like, showing both forward and backward locomotion is shared in [10] (Fig. 1.3c) exploiting the anguilliform motion pattern.

In terms of design, anguilliform pattern is difficult to mimic since it requires large number of degrees of freedom and actuation for each degree [27]. Whereas sub-carangiform or carangiform patterns are relatively easy to mimic using as few as a single actuator and flexible

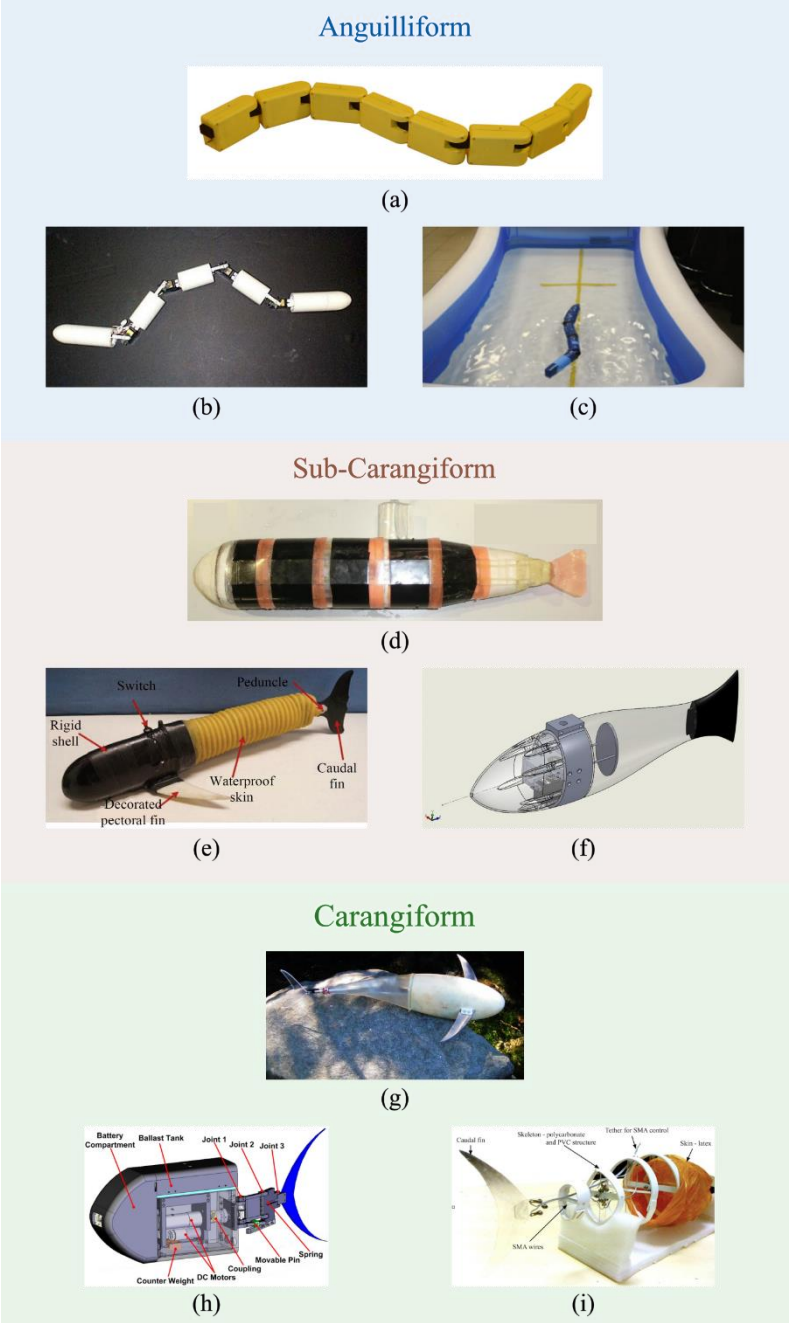


Fig. 1.3: *Top*: Robots inspired from the anguilliform motion pattern. Sub-figures (a), (b), and (c) are from [5], [7], and [10] respectively. *Mid*: Platforms mimicking the sub-carangiform motion pattern. Sub-figures (d), (e), and (f) are from [12], [13], and [14] respectively. *Bottom*: Platforms mimicking the carangiform motion pattern. Sub-figures (g), (h), and (i) are from [15], [16], and [17] respectively.

materials for the body and fin. Hence, in the sub-carangiform and carangiform category we find quite a few efforts.

For example, [12] mimics sub-carangiform and carangiform by using different materials for the body of the robot (Fig. 1.3d) and using one actuator for both cases. Similarly,

the robotic fish (Fig. 1.3f) in [58] and [14] also exhibit sub-carangiform pattern with single actuator and complaint tail. [13] presents similar looking motion pattern on a multi-joint robotic fish with 4 actuators (Fig. 1.3e). List of carangiform-mimicking designs is very exhaustive [27]. [15, 16, 59] are just a few examples of robots that mimic the carangiform motion pattern. Three examples are shown in Fig. 1.3g-i.

Although ostraciform and thunniform categories are not considered in this study but in literature we find examples for these swimming gaits as well (e.g., [60] for the ostraciform and [30] for the thunniform category).

All the efforts described above focus on one specific motion-pattern/swimming gait. In terms of realization of multiple motion patterns on the same platform, very few works can be found. In [12] both sub-carangiform and carangiform patterns are realized but by making changes in design (adjusting compliance). In [61], three patterns are realized by selecting corresponding kinematics from literature. In this work, our focus will be to realize different motion patterns on the same platform not through any change in design or rendering pre-determined kinematics but via learning different swimming gaits. The platform we use is all rigid with no complaint part. The design of the robot (hardware) and the learning methodology proposed in this study to develop/learn different swimming gaits will be explained in the following two sections.

1.4.1 Modelling Related Studies

Before moving to explain design and methodology in the next section, modelling related studies are very briefly mentioned in this sub-section. A large body of work in the area of marine robotics is related to modelling of dynamics of the robot by considering fluid forces. Since this study explores the potential of model-free (learning based) methods, those studies are not discussed in depth and only their designs and aspects related to this study are discussed.

Readers can take lead from the following notable studies to explore more about modelling specific literature.

Two key theories have been used to mathematically explain swimming phenomena. Wu [62] makes use of a waving plate theory to model dynamics of a swimming phenomenon. Similarly, Lighthill comes up with Elongated Body theory [63] to explain an underwater locomotion mode in which the deviation of tail or caudal fin is not very large and Large Amplitude Elongated Body theory [64] to explain motion of a broad class of underwater locomotion formats.

In [4], a thorough study of general undulatory locomotion considering internal shape changes and non-holonomic constraints is provided. Similarly, in [5], a specific case of undulation (anguilliform) is considered. In [65], *pisciform* type motion is modelled using Euler-Lagrange formulation and then the simulation results are verified by hardware experiments on a laboratory-developed swimming machine. Similarly in [56], carangiform mode is modelled and feedback based control and motion planning is done on an aquatic platform.

[2, 66] are good review studies on fish swimming modes and on mechanics and control issues related to swimming. In [67], 3D geometric modeling is performed using Lagrange-Poincare model to model added mass, lift, and drag effects for an all rigid multi-link robotic fish. Similarly, there are efforts related to modelling of rigid body and soft fin [68-70]. Traditional modelling approach can be referred to [71] and the referred studies in it.

1.5 BCFbot

The hardware considered in this work is briefly introduced in this sub-section.

The robot is named ‘BCFbot’ - Body and Caudal Fin robot. BCFbot is custom designed keeping a general anatomy of BCF swimmers in view. There is a body part which contains a head and a tail. There is a caudal fin at the posterior end. The hardware does not mimic any

organism in particular and a generic design is opted so that it can exhibit more than one motion profile to exploit their benefits. The hardware can be seen in Fig. 1.4.

The robot is modular in design. Joints can be easily added or removed. A reduced version with four joints is also considered in some parts of this work (Sec. 4). Fins of different sizes, shapes and materials can be attached and replaced easily. Two types of heads have been used. One of them is ellipsoidal in shape (Fig. 1.4a and 1.4b). A different head with a small form factor (Fig. 1.4c) is also used for reduced drag. This head makes the robot look like a snake (Fig. 1.4c). Generally, the hardware is described here briefly. However, it is used with some variations in different parts of this study and specific details will be mentioned in respective chapters (also see Table 1.1 in Sec. 1.9 for more details).

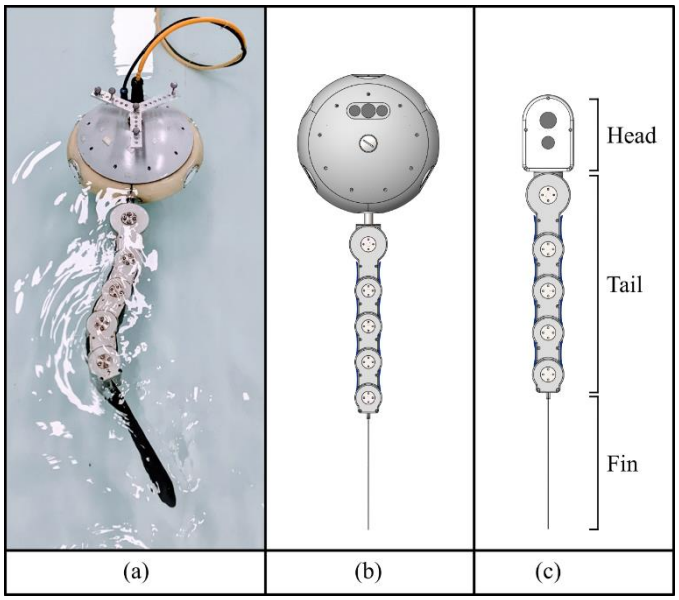


Fig. 1.4: (a) Hardware BCFbot (b) CAD model with ellipsoidal head. (c) CAD model with smaller snake-like head. Parts of the robot are labeled in (c).

In above design (Fig. 1.4), the axes of rotation of all the joints are parallel to each other. Therefore, the robot can only perform planar locomotion. A modified version (able to locomote spatially underwater) with axes of rotation of alternate joints perpendicular to each other will be introduced later (Sec. 1.8) (Refer to Table 1.1 in Sec. 1.9 for comparison among all versions).

Testing setups will be introduced in detail in respective chapters. Briefly the setup includes a swimming pool (4m x 2.5m) with tracking cameras installed around it. The position and velocity of the robot can be measured using the tracking setup. The setup also includes a thrust measurement apparatus which can measure the propulsive forces developed by the robot. It is a useful tool to evaluate robot's performance.

1.6 Developing motion modes

In this section, the first contribution of this study will be described. In Sec. 1.4, different designs mimicking different motion patterns are reviewed and it is shown that there have been efforts to mimic the anguilliform [10, 43], sub-carangiform [14, 58], and carangiform [15, 31] gaits on BCF-type robots. However, most existing works develop one particular motion pattern and study features related to one kind of swimmer. In this study, we plan to develop multiple motion patterns on the same platform to capture the benefits associated with different patterns. In this sub-section, we will first briefly see the methods to develop motion patterns in literature and then introduce the proposed method in this study.

Regarding methods to develop swimming gaits in literature, travelling-wave kinematics based on Lighthill's work [63] have been extensively studied and applied. For example, carangiform motion pattern is studied on robotic fishes [15, 72-74] using travelling wave kinematics. Different patterns can also be established by capturing the kinematic motion parameters from live organisms and then rendering the corresponding swimming patterns on robotic counterparts. For example, data from live lampreys is used to reproduce anguilliform swimming gait on a robotic lamprey [75].

Instead of using pre-defined kinematics or focusing on a particular swimming pattern, we try to come up with a unified scheme to create such patterns whereby the same scheme can be optimized for various patterns exhibited by different organisms.

Central Pattern Generators (CPGs) have long been studied as circuits in vertebrates' spinal cords for rhythmogenesis. They are responsible for rhythmic motion generation without requiring periodic inputs [76]. They have been modelled and used on robots inspired from nature [43, 45, 77, 78] for motion generation. They receive commands from midbrain for gait modulation and transition [9]. Taking inspiration from this two-level gait generation and modulation process in nature, a similar bioinspired scheme is conceived to explore the prospect of developing different motion patterns exhibited by different swimmers of BCF category. CPGs, modelled as network of coupled oscillators, are used for low-level gait generation. This network will still require an efficient high-level learning and optimization technique that can exploit and modulate CPGs to acquire diverse behaviors. For this purpose, we choose Deep Reinforcement Learning (DRL).

Fig. 1.5: (a) Representation of sub-carangiform swimming gait (from Fig. 1.2). (b) Similar swimming gait learned on a simulated model of the robot using the proposed method. (c) Similar swimming gait rendered on the physical robot.

DRL combines reward-based optimization using multi-layer perceptron models [79]. It has demonstrated its potential on various biomimetic platforms [77, 78, 80]. It has multiple variants and, in this study, Trust Region Policy Optimization (TRPO) [81] is used. Using DRL as high-level controller lets the robot explore ways to modulate the low-level controller by training without any supervised setup. In this way, the proposed schematics offers a learning

framework to develop not one but a range of gaits. Hence, it has a wide applicability on different platforms resembling the BCF anatomy.

Like natural swimmers, when the robot wriggles in a yielding medium (such as water in this case), the medium returns the forces back to the robot. These forces are responsible to propel the robot. We use these forces as key yardsticks used by the DRL module to tune the generators to develop different motion modes. When different targets for desired force generated by the robot are set, the framework explores different gaits to deliver those targets. Those gaits resemble the gaits observed in natural BCF swimmers. This helps the robot to learn and explore swimming patterns in simulation. To evaluate and compare the swimming patterns developed by the framework in simulation, we render them on the hardware of the robot – BCFbot (Fig. 1.4).

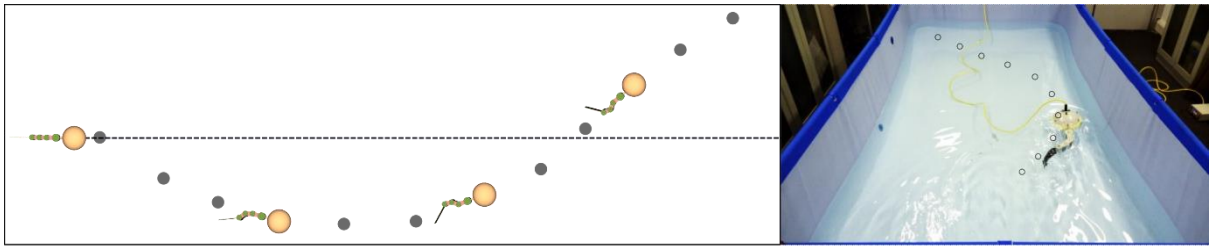
Development of different swimming gaits will be explained in detail in Sec. 3 where the proposed architecture and the training methodology will be explained in detail along with simulation and hardware experimentation results. Some benefits of the proposed schematics will also be shared. We explore the first three motion patterns i.e., the anguilliform, sub-carangiform, and carangiform patterns (Fig. 1.2) but it is believed that the framework can be further extended to include more swimming gaits as well.

Fig. 1.5 shares one of the gaits (sub-carangiform) developed in simulation and rendered on the physical robot just to introduce the idea of this part.

1.7 On-Surface Navigation

In this part, the prospect of using learning techniques for navigation in marine environment is explored. Fig. 1.6 conveys the abstract idea of this part. A reference trajectory is generated as a set of path points. The robot follows the path points one by one and finishes the whole trajectory in simulation. Later the results are verified on the hardware.

Navigation in marine environment has always been a challenging area of research due to the inclusion of hydrodynamics and limited sensing options. Thrusters are commonly used in marine robotics and are easy to control and perform navigation on. Biomimetic robots, on the other hand, employ tails and fins for movement and are difficult to control. Different techniques have been proposed to control and navigate such bioinspired robots. Control based on mathematical models for such robots is complicated and model-free methods provide a convenient alternative.



(a) (b)

Fig. 1.6: (a) Simulated model of BCFbot following a pre-defined trajectory in simulation. (b) Hardware BCFbot following a trajectory in laboratory pool.

In this part of study, it is aimed to develop swimming controllers for the BCFbot (Fig. 1.4). For effective swimming, the motion of joints should be properly timed such that a wave travelling from the anterior part of the tail to its posterior part is induced [82]. Hydrodynamic interaction of the tail in the presence of travelling waves produces a thrust force that enables the robot to swim forward. Some tail-bias is added to the segments of the tails to adjust the direction of thrust vector and hence the heading of the robot.

CPGs (introduced briefly in previous Sec. 1.6) are commonly used to create rhythmic motion patterns [83] and have been demonstrated on many different biomimetic robots mimicking fishes [84] and snakes [85] etc. They are used to produce cyclic or rhythmic motion profiles without using periodic input. CPGs can be considered a special case of a much broader class of motion representation paradigm called the Dynamic Motion Primitives (DMPs).

DMPs have an edge over CPGs that they can represent virtually any motion pattern and also provide a systematic framework to modulate output behavior [86]. To encode different behaviors into the DMPs, some supervised setup or demonstrations are required [87]. For example, a recorded trajectory, when a human teacher plays table tennis with a tennis bat attached to an end effector of a robotic arm, can be used to fit motion primitives [88]. Later these motion primitives can be used to perform similar actions. In an improved form, such primitives, pre-trained using demonstrations first, can be improved upon using some learning methods [89] to perform even better than the original demonstrations. In either case, supervision is required and those platforms where supervision is difficult to obtain, such as ours, it is difficult to use this method. We also use DMPs due to their inherent properties of stability, temporal and spatial invariance [86] but, at the same time, devise a way to train DMPs without using demonstrations or supervision.

To this end, again the same method – DRL is used. It combines state-of-the-art optimization using deep neural networks (DNN) with reinforcement learning strategy [79]. DRL has been very popular and successfully implemented for navigation of mobile robots for both terrestrial [90, 91] and marine domains [80, 92].

A navigation training framework is designed. Learning is done without supervision using simple reward functions. During training, random paths are generated for the robot to follow. The robot is rewarded for following the path correctly and is penalized upon deviating from it. After the training is done, the robot can track previously known and unknown trajectories. The results are also verified on the physical robot. Hence, DRL and DMPs are combined to develop a control scheme for a marine robot to navigate on the surface of water. The combination adds the benefits of both DRL and DMPs. DRL allows to optimize motion primitives in simulation without demonstrations and DMPs, in addition to stability, allow a feasible framework for the modulation of motion behavior such as frequency adaptation which

is difficult to achieve using DRL only. This usefulness of the combined framework will be demonstrated in Sec. 4.4. The problem is treated as two separate reinforcement learning problems that employ two policies and one set of motion primitives. First, the motion primitives are trained using DRL to establish travelling wave in the tail to produce thrust. Once the wave is established, another policy is trained to modulate the primitives, trained in the first step, to perform navigation along any desired trajectory.

1.8 Underwater Locomotion and Navigation

Up to this section, locomotion on the surface of water is considered. The robots in the last two sections have a series of 1-DoF joints (as introduced in Sec. 1.5 and Fig. 1.4). In this section underwater locomotion and navigation (with depth variation) is considered.

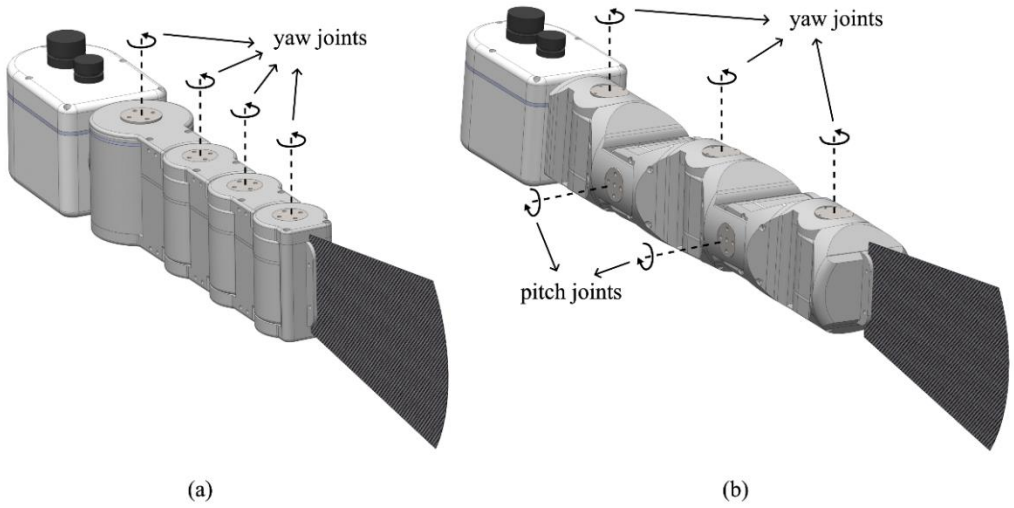


Fig. 1.7: (a) 1-DoF BCFbot with four yaw joints. (b) 2-DoF BCFbot with two pitch and three yaw joints.

To perform underwater locomotion on biomimetic platforms, pectoral fins are mostly used [32, 67, 93]. This fin-type is briefly discussed in the start of introduction (Sec. 1.3) under the MPF category. These fins are paired on either side of the body. But in this work pectoral fins are not considered and only the caudal fin is used for depth variation as well. For this purpose, the design is updated (Fig. 1.7).

In the final part of this thesis, the design of the robot is modified to add another degree of freedom to the fin. Two pitch joints are added alternately with three yaw joints (Fig. 1.7). In this way, the fin can perform motion in multiple planes. Before this modification, the robot was restricted to planar locomotion and could swim only on the surface of water. Owing to the above-mentioned design amendment, the propulsion space of the robot is no longer restricted to planar domain. It is now able to generate propulsion in a spatial region. The modified hardware is shown alongside the original version in Fig. 1.7. The modified version is referred to as "2-DoF BCFbot". By 2-DoF, it does not mean that the robot has only two actuated joints, but it means that it has two different joint types (pitch and yaw) unlike the original 1-DoF BCFbot which only has one type (yaw) of 5 joints.

Before explaining methods used in this part, we name the methods used for previous parts since the new method is evolved from them. Pattern generators are used in the swimming gaits generation part (Sec. 1.6). After that, motion primitives are used to perform on-surface navigation along pre-defined trajectories (Sec. 1.7). And on the top of both, RL is used on the high-level to learn desired behaviors with the above two methods on the low-level.

For the modified hardware, the framework of Sec. 1.6 is extended. Two networks of CPGs are used. The first network has three oscillators coupled together for the three yaw joints. This is for continuous rhythm generation to produce propulsion. To achieve pitch motion, another (second) network is used in point attractor fashion [88]. Fixed targets are set as goals for this network to generate pitch motion. With pitch motion, the overall thrust vector can be in the spatial region (rather than planar region) which essentially means that the robot can now move in all dimensions underwater.

For experimentation, the thrust measurement setup (briefly mentioned in Sec. 1.5) is modified to include measurement of the third (z) component F_z, in addition to F_x and F_y.

Experiments are done for several cases. First, constant pitch bias is added in the pitch joints while the yaw joints undulate. Upon adding positive biases, the tail bends upward causing the thrust vector to point downwards. Similarly, upon adding negative biases, the tail moves downward causing the thrust vector to orient upwards. Free-swimming trials are then conducted to observe robot's underwater locomotion performance. A snapshot from free-swimming experiments is shown in Fig. 1.8. The results establish that singular caudal fin, when supplemented with yaw and pitch joints, can orient the thrust vector in a wider domain to make the robot perform underwater navigation.

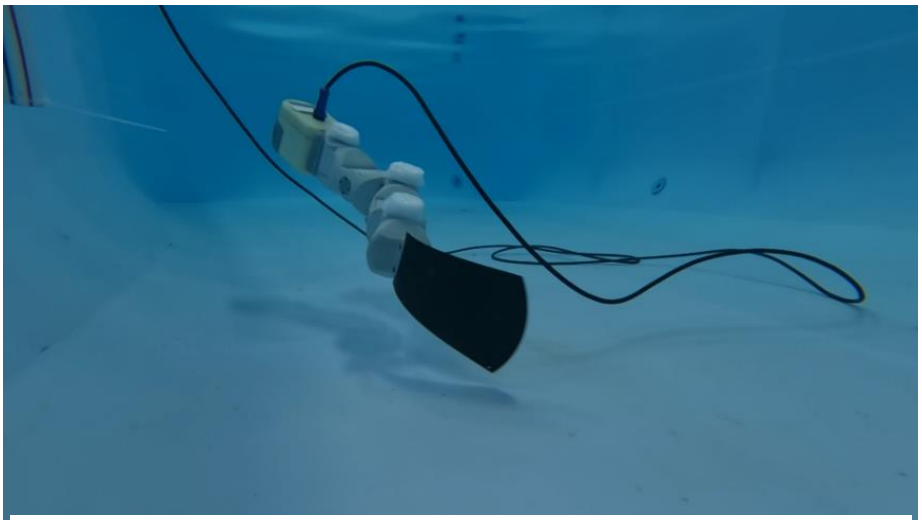


Fig. 1.8: 2-DoF BCFbot performing 3D underwater locomotion.

In the end, navigation/path-following is done by extending the navigation architecture introduced in the previous section. For this case, random trajectories are generated in 3D region (for on-surface path-following, planar paths were generated). The robot is trained to track them in simulation. The robot is rewarded for following the path correctly and is penalized upon deviating from it. Once the training is done, the robot can track 3D paths well.

The three directions (Gaits generation – Sec. 1.5, on-surface navigation – Sec. 1.6, and underwater locomotion and navigation – Sec. 1.7) introduced in this section conclude the introductory part. They will be treated thoroughly in the respective chapters (Sec. 3, Sec. 4, and Sec. 5 respectively).

1.9 Contributions

The contributions of this study are summarized in the following points:

1. A bioinspired learning framework which combines DRL and CPGs, is proposed from which a variety of swimming gaits resembling different natural BCF swimmers' gaits emerge. These include the anguilliform, sub-carangiform, and carangiform gaits. This should be the first time to develop and optimize multiple swimming gaits using one same architecture without using any predefined kinematics.

2. The swimming gaits are visualized and compared on physical BCFbot. The multi-segment tail of the robot can imitate more than one gaits in the same design which makes it easier to visualize and contrast different gaits. This visualization is recorded in the form of waveforms as well as in the form of snapshots of the experiments.

3. A thorough comparison is made between the swimming gaits which reveals certain benefits associated with each gait type making them appropriate for different scenarios and applications. This comparison includes linear and angular speed performance, and power consumption of the robot while swimming in laboratory pool.

4. Features of the combined DRL-CPG approach are demonstrated which are not possible using DRL-only approach. Notable features include easy frequency adaptation, amplitude modulation, and flexibility to train DRL networks for high level tasks such as navigation. These features will be highlighted in the respective chapters.

5. Control architecture is extended to include DMPs. This extension helps to expand the scope of work from swimming gait optimization to on-surface navigation of the robot along desired trajectories. This extension does not require any supervised setup, or any library based on demonstrations. It also allows the use of generalization properties of DMPs such as spatial and temporal invariance which are of particular use for biomimetic robots.

6. Finally, the design of the robot is modified to include pitch joints to perform 3D underwater locomotion by using a single caudal fin and without pectoral fins. Using only the caudal fin keeps the design simple and provides a simple solution for 3D locomotion. Rhythmic and point oscillators are combined to control yaw and pitch joints for underwater locomotion. Finally, the robot is also trained for 3D path-following using DRL.

In this way, it is targeted that similar architectures with small variations are used in all the study i.e., gaits generation, on-surface navigation, and underwater locomotion and path-following. A unified scheme is presented for gait generation, modulation, and navigation with the hope that it can later be extended to include exteroceptive perception as well. In this work, we imitate the first three motion patterns i.e., anguilliform, sub-carangiform and carangiform but we believe that, in future, the framework can be further extended to include more swimming gaits as well.

For this thesis, terms ‘navigation’ and ‘path-following’ (are/will be used frequently and) mean the same thing (i.e., following a path that constitutes discrete way points). Similarly, ‘trajectories’ and ‘paths’ also mean the same (i.e., a way consisting of discrete way points).

To remove any later confusion two tables are provided below to serve as a reference for different variations of BCFbots used in the entire study. Table 1.1 shows four different

variations of BCFbots. Sr. 1-3 have yaw-only joints; hence they can only swim in one plane (surface of water). They will be used in Sec. 3 and Sec. 4. Sr. 4 has both yaw and pitch joints owing to which it can swim spatially. It will be used in Sec. 5 in which underwater locomotion and navigation will be discussed. Dimensions of individual parts are mentioned in Table 1.2.

Table 1.1 – Types of BCFbots used in this study

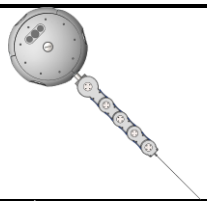
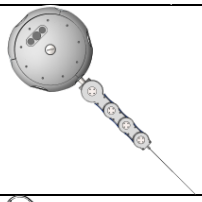
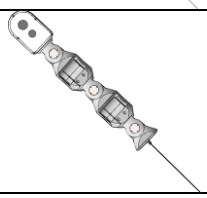
Sr. No.	Usage Chapter in thesis	Application	Type of joints	No. of joints	Type of head	Total Length (cm)	Top CAD view
1	3	Learning Swimming Gaits	Yaw only	5	Big (Ellipsoidal)	80	
2	3	Learning Swimming Gaits	Yaw only	5	Small (Snake like)	65	
3	4	On-surface Navigation	Yaw only	4	Big (Ellipsoidal)	73	
4	5	Underwater Locomotion & Navigation	Yaw and Pitch	5	Small (Snake like)	68	

Table 1.2 – Dimensions of sub-parts of BCFbots in Table 1.1

Sr.	Part	Dimension (l × w × h) (cm)
1	Individual Joint in Sr. 1,2,3	6.6 × 5 × 8
2	Individual Joint in Sr. 4	8 × 8 × 8
3	Small head (Snake like)	12 × 8 × 8
4	Big head (Ellipsoidal)	26 × 26 × 15
5	Caudal Fin	20 × 0.7 × 10

1.10 Organization of thesis

In each chapter there is a brief introduction at the start followed by a review on important literature related to the work presented in that chapter. The rest of the thesis is organized in the following order.

In Sec. 2, we review some literature on DRL (Sec. 2.1) and CPGs (Sec. 2.2) with application to robotics in general and to marine robotics in particular. We give a brief background of CPGs and DRL and introduce the combined framework along with algorithm.

In Sec. 3, we describe the training procedure according to control schematics introduced in the previous section. In this section different swimming gaits are developed. Then the swimming gaits, developed in the previous section, are visualized in simulation and on the hardware. Also, a thorough comparison is made between different gaits in terms of different metrics. Based on the comparison some conclusions are shared.

Sec. 4 extends the architecture further to navigation of robot along desired trajectories. Some related work, background of methods used in this part, training methodology, simulation and hardware experimentation results are shared.

Sec. 5 deals with underwater locomotion. Both the hardware and the methodology are extended to include underwater locomotion. Modifications in the design of the robot (Sec. 5.2.1), testing setup (Sec. 5.2.2), control framework (Sec. 5.3) are followed by thrust testing results for different cases. Some free-swimming trials (Sec. 5.5) are shared. In the end, 3D path-following is shared in simulation (Sec. 5.6).

Sec. 6 concludes the study. It summarizes the work shared in previous sections. Some conclusions related to work shared in the study are shared. Contributions of the study are highlighted. And some possible directions and ideas for future work are also discussed.

Equations, figures, tables, and algorithms are numbered chapter wise. To refer to a figure, ‘Fig.’ is used along with the figure number (e.g., Fig. 1). To refer to an equation, just the equation number is used in round parentheses. To refer to any section or chapter, ‘Sec.’ is used along with the section number (e.g., Sec. 1). For tables and algorithms, full terms (e.g., Table 1.1 and Algorithm 3.1) are used.

CHAPTER 2

METHODS AND BACKGROUND

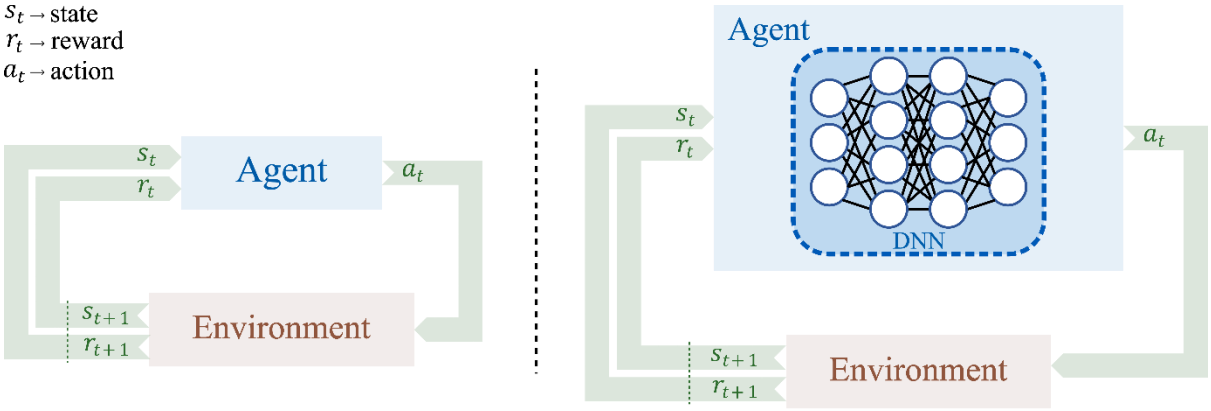
In this part, individual components of control schematics will be introduced. The overall schematics contains two components – Deep Reinforcement Learning (DRL) combined with Central Pattern Generator (CPG). Both parts will be introduced first. Both have a significant amount of literature in the applied domain. Some notable studies will be reviewed which are relevant to this work. It is aimed to keep the literature review focused on bio-inspired designs although some work related to other pertinent cases is also introduced.

2.1 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) is a model-free method which combines state-of-the-art optimization using deep neural networks (DNN) with reinforcement learning (RL) strategy [79]. Learning is done without supervision using simple reward functions.

In short, reinforcement learning can be defined as a type of learning mechanism in which an agent learns by experience. With each instance or iteration of experience it receives a reward or penalty. The definition of reward should be carefully crafted to appreciate desired

behavior and penalize undesired behavior. The agent tries to maximize the reward or minimize the penalty [94]. In doing so, it learns the desired behavior. This scheme is explained briefly in Fig. 2.1a. The agent's current state $s(t)$ or s_t is updated to $s(t+1)$ or s_{t+1} upon taking an action $a(t)$ or a_t and receives a reward $r(t)$ or r_t. This transition is shown in Fig. 2.2.



Reinforcement Learning
(a)

Deep Reinforcement Learning
(b)

Fig. 2.1: (a) Illustration of Reinforcement Learning. (b) Illustration of Deep Reinforcement Learning. DNN stands for Deep Neural Network.

In RL, the agent takes action according to any scheme (commonly referred to as a 'policy' in reinforcement learning literature). It tries to optimize that policy so as to maximize its reward. The policy can be as simple as a linear policy, or it can be a highly non-linear function depending upon the task [95].

In deep RL (or DRL), the policy is a deep neural network (DNN) [79]. It is illustrated in Fig. 2.1b. It can contain many layers with hundreds of neurons (or nodes) in each layer. Number of layers or neurons in each layer should be selected according to the application. So, a reinforcement learning scheme together with deep neural networks is called deep reinforcement learning (DRL).

A brief background with mathematical formulation will be given in the next section (Sec. 2.1.1). After which, related work in the domain of robotics will be shared (Sec. 2.1.2).

2.1.1 Background

Some theoretical background of DRL used in this work is explained in this part.

In DRL, Markov Decision Process framework is established in which an agent (which can be a simulated version of robot) is trained in an episodic fashion. The agent upon taking an action $a(t)$ according to policy π i.e., $a(t) \sim \pi\big(s(t)\big)$ switches from state $s(t)$ to $s(t+1)$ and is awarded a reward r based on a preset reward function definition. This transition takes place according to state transition dynamics $D = p\big(s(t+1)|s(t), a(t)\big)$. The scheme repeats itself over the course of many timesteps called an episode.

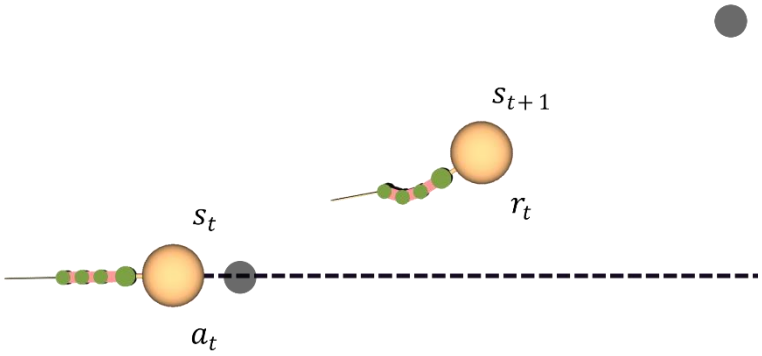


Fig. 2.2: Agent (robot in simulation) transitioning from one state to next.

Collective reward of an entire episode is called a Return. Q_π is expected return upon taking $a(t)$ in $s(t)$ following π,

$$Q_\pi(s,a) = \mathbb{E}_{s(t+1),a(t+1)\ldots}\left[\sum_{t=0}^{T} \gamma^t r\big(s(t),a(t)\big)\right] \tag{2.1}$$

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s), \tag{2.2}$$

γ being a limiting factor for future rewards, T being length of the episode, and V_π is expected return. A_π is called the advantage function.

An on-policy method called Trust Region Policy Optimization (TRPO) [81] is used in this work. In the following, trust region methods are described.

Many optimization methods are used in reinforcement learning [96]. Gradient based techniques can be roughly divided into two groups. In one group, gradient is calculated to determine the desired direction and then step length is calculated based on some preset stationary or sometimes dynamic parameter. In the other group, a region is defined, usually called a trust region, which is in the close vicinity of the point of optimization. The region is carefully chosen so that the optimization step may not stray far away from the point of interest.

In the policy gradient methods, the gradient update generally follows the following update rule [97],

$$\theta \leftarrow \theta + \alpha \gamma^n A(s_n, a_n) \nabla \log \pi_\theta(a_n|s_n), \tag{2.3}$$

here α is the learning rate and θ represents policy parameters. A high learning rate gives a fast convergence rate of optimization but is usually unreliable. And a slower learning rate is usually reliable to converge but is very slow. Hence, α is difficult to tune.

If the actual objective that needs to be optimized is very complex. For example, the objective function is highly non-linear. A surrogate objective function can be assumed. The surrogate can be a lower order function, a quadratic approximation for instance. Such lower order surrogate function can be optimized easily and efficiently. One problem here can be the divergence of approximation function during optimization. The surrogate can deviate or stray a lot from the original objective function. To overcome this problem, a trust region is defined to limit the search.

So, a trust region can be defined as a vicinity around the point of optimization where the surrogate function remains a good approximation of the original function. An improvement estimation based on the simplified surrogate function should be an improvement for the original objective function. Hence, trust region can guarantee monotonic improvement in optimization.

Considering a policy π_θ with parameters θ, a trust region around parameters θ can be defined with respect to some distance metric D as,

$$\theta' \mid D(\theta, \theta') < \delta \tag{2.4}$$

Similarly, a trust region of policy π_θ will be,

$$\theta' \mid D(\pi_\theta, \pi_{\theta'}) < \delta, \tag{2.5}$$

where D in (2.4) is distance between old and new parameter and D in (2.5) is distance between old and new policy as a whole.

Value function varies slowly with a slow change in policy π_θ since a small change in policy will lead to a small change in the probability of policy output. On the other hand, value function can vary rapidly with a small change in parameters θ if the trust regions are considered around parameters. Hence, the policy trust region (2.5) is a better choice compared to parameter trust region.

The distance metrices used in the above trust region definitions (2.4 and 2.5) can be different. In Trust Region Policy Optimization (TRPO), Kullback-Leibler (KL) divergence is used. KL divergence between two distributions $p(x)$ and $q(x)$ can be defined as,

$$D_{KL}(p(x), q(x)) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{2.6}$$

which states that divergence between two distributions $p(x)$ and $q(x)$ is the expectation of logarithmic difference between $p(x)$ and $q(x)$ using the law of logarithm.

In the same way KL divergence between two policies π_θ and $\pi_{\bar{\theta}}$ can be calculated and given as under,

$$D_{KL}(\pi_\theta(\cdot\,|s), \pi_{\bar{\theta}}(\cdot\,|s)) \;=\; \sum_a \pi_\theta(a|s) \log \frac{\pi_\theta(a|s)}{\pi_{\bar{\theta}}(a|s)} \tag{2.7}$$

where π_θ and $\pi_{\bar{\theta}}$ are the old and current policies.

The objective function in TRPO can now be formally stated as,

$$\theta \leftarrow \underset{\bar{\theta}}{\text{argmax}} \;\; \mathbb{E}_{s_0 \sim p} \left[V_{\pi_{\bar{\theta}}}(s_0) - V_{\pi_\theta}(s_0) \right], \tag{2.8}$$

subject to the following KL divergence condition,

$$\max_s \;\; D_{KL}(\pi_\theta(\cdot\,|s), \pi_{\bar{\theta}}(\cdot\,|s)) \leq \delta \tag{2.9}$$

The objective in (2.8) is to make sure that the raise in value function is as much as possible. The constraint in (2.9) makes sure that the new and the current policies diverge no more than δ. So, the constraint statement (2.9) is actually the trust region.

The problem with the above optimization problem (2.8) is that for most tasks, the function itself is usually very complex and sometimes it happens that it is not directly computable. A surrogate objective is therefore defined, which serves as a simpler objective. (2.8) can therefore be approximated as [98],

$$\begin{aligned} &\underset{\bar{\theta}}{\text{argmax}} \;\; \mathbb{E}_{s_0 \sim p} \left[V_{\pi_{\bar{\theta}}}(s_0) - V_{\pi_\theta}(s_0) \right] \\ &\approx \underset{\bar{\theta}}{\text{argmax}} \;\; \mathbb{E}_{s \sim \mu_\theta,\; a \sim \pi_\theta} \left[\frac{\pi_{\bar{\theta}}(a|s)}{\pi_\theta(a|s)} A_{\pi_\theta}\big(s(t), a(t)\big) \right] \end{aligned} \tag{2.10}$$

where μ_θ is stationary state distribution for π_θ. Above approximation is key to trust region methods. Similar to the above approximation, the constraint is also simplified as,

$$\begin{aligned} &\max_s \;\; D_{KL}\big(\pi_\theta(\cdot\,|s), \pi_{\bar{\theta}}(\cdot\,|s)\big) \leq \delta \\ &\qquad\qquad \approx \mathbb{E}_{s \sim \mu_\theta} \left[D_{KL}\big(\pi_\theta(\cdot\,|s), \pi_{\bar{\theta}}(\cdot\,|s)\big) \right] \leq \delta \end{aligned} \tag{2.11}$$

Simplified objective is in the right-hand side of (2.10) and simplified constraint in the right-hand side of (2.11). Left-hand side of (2.11) states that the KL divergence between the current and the desired policy should remain under δ. This is difficult to optimize, so an expectation is taken regarding states.

But how to reach at the simplification in (2.10)? To reach at the surrogate objective in (2.10), consider the surrogate objective rewritten below,

$$\underset{\bar{\theta}}{\text{argmax}} \;\; \mathbb{E}_{s \sim \mu_\theta,\; a \sim \pi_\theta} \left[\frac{\pi_{\bar{\theta}}(a|s)}{\pi_\theta(a|s)} A_{\pi_\theta}\big(s(t), a(t)\big) \right] \tag{2.12}$$

The state action pairs are taken from stationary distribution μ_θ and policy π_θ. (2.12) says that the improvement that is originally required in (2.8) and L.H.S of (2.10) is roughly equal to the ratio of the new policy and the existing policy times the advantage function.

We open the expectation using definition of expectations,

$$\underset{\bar{\theta}}{\text{argmax}} \; \sum\nolimits_s \mu_\theta(s) \sum\nolimits_a \pi_\theta(a|s) \left[\frac{\pi_{\bar{\theta}}(a|s)}{\pi_\theta(a|s)} A_{\pi_\theta}\big(s(t), a(t)\big) \right] \tag{2.13}$$

Cancelling like terms gives us simplified expression,

$$= \underset{\bar{\theta}}{\text{argmax}} \; \sum\nolimits_s \mu_\theta(s) \sum\nolimits_a \left[\pi_{\bar{\theta}}(a|s) A_{\pi_\theta}\big(s(t), a(t)\big) \right] \tag{2.14}$$

Replacing μ_θ with $\mu_{\bar{\theta}}$,

$$\approx \underset{\bar{\theta}}{\text{argmax}} \; \sum\nolimits_s \mu_{\bar{\theta}}(s) \sum\nolimits_a \left[\pi_{\bar{\theta}}(a|s) A_{\pi_\theta}\big(s(t), a(t)\big) \right] \tag{2.15}$$

'$\approx$' is used since above replacement is an approximation. It is possible to take samples from the existing set, but it is desired to measure the value of new policy. So μ_θ is replaced by $\mu_{\bar{\theta}}$.

The stationary distribution of states is defined as,

$$\mu_{\bar{\theta}}(s) \propto \sum_{n=0}^{\infty} \gamma^n P_{\bar{\theta}}(s_n = s) \tag{2.16}$$

which measures the proportion of landing in each state. Replacing the stationary distribution by its definition by inserting (2.16) in (2.15),

$$= \underset{\bar{\theta}}{\text{argmax}} \sum_s \sum_{n=0}^{\infty} \gamma^n P_{\bar{\theta}}(s_n = s) \sum_a \left[\pi_{\bar{\theta}}(a|s) A_{\pi_\theta}\big(s(t), a(t)\big)\right] \tag{2.17}$$

Rearranging (2.17),

$$= \underset{\bar{\theta}}{\text{argmax}} \sum_s P_{\bar{\theta}}(s_n = s) \sum_a \pi_{\bar{\theta}}(a|s) \left[\sum_{n=0}^{\infty} \gamma^n A_{\pi_\theta}\big(s(t), a(t)\big)\right] \tag{2.18}$$

Writing again in the form of expectations but now expanding for all expected states for all of the event horizon,

$$= \underset{\bar{\theta}}{\text{argmax}} \; \mathbb{E}_{s_0,\, s_1 \ldots \sim P_{\bar{\theta}},\;\; a_0,\, a_1 \ldots \sim \pi_{\bar{\theta}}} \left[\sum_{n=0}^{\infty} \gamma^n A_{\pi_\theta}\big(s(t), a(t)\big)\right] \tag{2.19}$$

s_0, s_1 ... $\sim P_{\bar{\theta}}$ are all the states that will be seen in the horizon and a_0, a_1 ... $\sim \pi_{\bar{\theta}}$ are all the actions respectively. So, the above expression is expectation of summation of the discounted advantage function.

Advantage function is defined as the difference in action value function and the value function [98],

$$A_{\pi_\theta} = \mathbb{E}_{s' \sim P(s'|s,a)}[r(s) + \gamma V_{\pi_\theta}(s') - V_{\pi_\theta}(s)] \tag{2.20}$$

where $r(s)$ is reward, $\gamma V_{\pi_\theta}(s')$ is the value of the next state and $V_{\pi_\theta}(s)$ is the value of the current state. Putting above definition into (2.19),

$$= \underset{\bar{\theta}}{\text{argmax}} \;\; \mathbb{E}_{s_0,\, s_1 \ldots \sim P_{\bar{\theta}},\;\; a_0,\, a_1 \ldots \sim \pi_{\bar{\theta}}} \left[\sum_{n=0}^{\infty} \gamma^n \left[r(s_n) + \gamma V_{\pi_\theta}(s_{n+1}) - V_{\pi_\theta}(s_n) \right] \right] \tag{2.21}$$

Opening the above summation will lead to cancelling of most terms. The only remaining term will be the one related to initial state.

$$= \underset{\bar{\theta}}{\text{argmax}} \;\; \mathbb{E}_{s_0,\, s_1 \ldots \sim P_{\bar{\theta}},\;\; a_0,\, a_1 \ldots \sim \pi_{\bar{\theta}}} \left[\sum_{n=0}^{\infty} \gamma^n \, r(s_n) - V_{\pi_\theta}(s_0) \right] \right] \tag{2.22}$$

In (2.22), $\sum_{n=0}^{\infty} \gamma^n \, r(s_n)$ is definition of value function. Using $V_{\pi_{\bar{\theta}}}(s_0)$ as its symbol to rewrite it,

$$= \underset{\bar{\theta}}{\text{argmax}} \;\; \mathbb{E}_{s_0,\, s_1 \ldots \sim P_{\bar{\theta}},\;\; a_0,\, a_1 \ldots \sim \pi_{\bar{\theta}}} \left[V_{\pi_{\bar{\theta}}}(s_0) - V_{\pi_\theta}(s_0) \right] \tag{2.23}$$

Since all other states have already been considered and only s_0 remains. Hence, simplifying expectation,

$$= \underset{\bar{\theta}}{\text{argmax}} \;\; \mathbb{E}_{s_0 \sim p} \left[V_{\pi_{\bar{\theta}}}(s_0) - V_{\pi_\theta}(s_0) \right] \tag{2.24}$$

which is the left-hand side of the original objective (2.10). Hence the surrogate objective assumed/approximated in (2.10), is actually a good approximation of the original objective that needs to be optimized.

Therefore, in TRPO, a surrogate objective can be optimized. Rewriting the objective again to formally devise out an algorithm with some change in notations for brevity,

$$\underset{\theta}{\text{maximize}} \qquad \mathbb{E} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\pi_{\theta_k}}(s(t), a(t)) \right] \tag{2.25}$$

TRPO takes a guaranteed step to improve the policy from the previous one. It keeps the updated policy in-check with the policy it improves upon, by measuring divergence between the two. This is done by using Kullback-Leibler divergence, D_{KL}, between the old and the new policy,

$$\mathbb{E}_{s,a \sim \pi_{\theta_k}} \left[D_{KL}\left(\pi_{\theta_k}(\cdot|s), \pi_{\theta}(\cdot|s)\right)\right] \leq \delta \tag{2.26}$$

which states that the divergence must be kept under a fixed value δ.

For most real-world cases, we will experience the original objective to be non-linear. Same will be true for the constraint. It will be really difficult to optimize them. Therefore, in most practical cases, the objective is represented by an analogous linear approximation and the constraint is represented by an approximation of second degree [99]. Doing these approximations will ease the computational process and speed up the optimization process.

For the main objective (2.25), linear approximation is done and for the constraint (2.26) quadratic approximation is done using Taylor expansion. Denoting the objective in (2.25) by $\mathcal{L}_{\theta_k}$, we can approximate as (2.25) and (2.26) as,

$$\mathcal{L}_{\theta_k}(\theta) \approx \mathcal{L}_{\theta_k}(\theta_k) + g^T(\theta - \theta_k) \tag{2.27}$$

$$D_{KL} \approx \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \tag{2.28}$$

g being the first derivative of $\mathcal{L}_{\theta_k}$ and H being the second derivative of D_{KL},

$$g = \nabla_\theta \mathcal{L}_{\theta_k} = \mathbb{E}_{s,a \sim \pi_{\theta_k}} \left[\nabla_\theta \log \pi_\theta(a|s)\, A_{\pi_{\theta_k}} \right] \tag{2.29}$$

$$H = \nabla_\theta^2 D_{KL} \tag{2.30}$$

Considering above approximations, the original objectives, (2.25) and (2.26), can now be written as,

$$\begin{cases} \theta_{k+1} = \underset{\theta}{\text{argmax}} \quad g^T(\theta - \theta_k) \\ \text{s.t.} \quad \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leq \delta \end{cases} \tag{2.31}$$

which can be analytically solved as [100],

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g \tag{2.32}$$

Above weight update equation (2.32) is the main update rule that will be followed to update the weights of neural networks involved in policies that will be used in this work. Using above procedure different policies will be optimized in the 'learning swimming gaits' chapter (Sec. 3) to optimize swimming patterns, in the 'on-surface navigation' chapter (Sec. 4) to learn navigation along pre-defined trajectories and in the 'underwater locomotion and navigation' chapter to learn navigation in 3D (Sec. 5). Regardless of the size of the policies or the task itself, the optimization process will remain the same.

2.1.2 Pseudocode

After deriving the above optimization scheme, it is better to construct a pseudocode that will later be used throughout the study.

The algorithm (Algorithm 2.1) starts by initializing the policy π_θ, a DNN in our case (line 1 in Algorithm 2.1). The exact specifications of the policy will be mentioned later with each case in the respective chapters. TRPO requires a number of state action pairs to construct a batch (lines 3 to 6 in Algorithm 2.1) according to which the objective and the constraint are evaluated. In other actor-critic methods, updates can be done at every iteration. TRPO is different in this regard. It operates on a batch not on each iteration [101]. The idea behind this is that after the surrogate has reliably captured the dynamics, the optimization will take place in the vicinity of current policy. This will make sure that the update step will definitely improve upon the current policy.

After collection of experience (lines 3 to 6), returns and advantages are calculated (line 7). Gradient over the policy is calculated in the next step (line 8) using (2.29). After which

Hessian is prepared for update rule (line 9) followed by step size (line 10). In the end, policy parameters are updated according to the update rule (line 11) derived in background (2.32).

Algorithm 2.1: Pseudocode – Trust Region Policy Optimization (Vanilla)

1. Initialize policy π_θ with random parameter set θ.
2. **for** $1 \leq k \leq K$ **do**
3. **for** (pre-defined batch size) **do**
4. Execute the desired MDP according to π_θ in an episodic format and note/save the (s_t, a_t, r_t, s_{t+1}) transitions of each episode.
5. Store the transition in a set of trajectories $\mathcal{B}$.
6. **end for**
7. Calculate returns G_t and advantage estimates A_t.
8. Compute policy gradient g_k according to (2.29)

$$g_k = \frac{1}{L} \sum_{\mathcal{B}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta (a|s) A_t$$

9. Compute the following relation according to a part of (2.32),

$$x_k \approx H_k^{-1} g_k$$

H_k being the Hessian of the Kullback Liebler divergence (2.30) between the old and new policies.

10. Compute step size,

$$\beta \approx \sqrt{\frac{2\delta}{x_k^T H_k x_k}} x_k$$

11. Update the policy parameters according to (2.32),

$$\theta_{k+1} = \theta_k + \eta\beta$$

η being the smallest value that satisfies the KL divergence constraint.

12. **end for**

2.1.3 Related Work

Model-free RL methods have shown a great potential in the field of robotics [95]. Mnih et al. [79] use a combined architecture of RL and DNN and present promising results on ATARI games that surpass human-level performance on a lot of games. Lillicrap et al. [102] extend the idea for continuous action spaces in an algorithm called Deep Deterministic Policy Gradient (DDPG) and share some results of simulated robotic tasks of continuous nature. [103] provide a parallel framework in which many RL algorithms perform much better and take significantly

less time for training. Shulman et al. [81] present another model-less optimization technique that also uses DNN as function approximator and demonstrate success on ATARI games and some simulated robotic environments with continuous action spaces. Similarly, Proximal Policy Optimization (PPO) [104] is another variation to TRPO that offers better sample complexity.

Now some studies from applications point of view on robotic platforms are reviewed.

For navigation of terrestrial robots, the following efforts are notable. Tai et al. [105] control a differential-drive robot using an asynchronous version of DDPG [102]. Using same platform, [106] perform navigation across similar environments using successor features. [107] use Soft actor-critic [108] to navigate in a crowded environment on a 4-wheeled mobile platform. Kahn et al. [109] use generalized computation graph to perform navigation on a platform similar to [107]. Among complex platforms, we find out that bipedal robots, e.g., humanoids, are famous platforms to benchmark different DRL algorithms in order to judge the performance of algorithms on complex structures. Bipeds have higher degrees of freedom (DoF) and are inherently unstable which enhances the complexity of the problem. [110] use a 7-DoF structure to perform straight walking behavior on both simple and uneven terrains at different speeds using PPO. Similarly, [111] use another biped – ATRIAS [112] to train neural network policies for walking on different terrains also using PPO [104].

Recently there have been some studies exploring the potential of DRL in marine domain as well. In marine domain, [92] navigate a small propeller operated boat using DRL. [80] utilize DRL to optimize a baseline behavior obtained by CPG on a biomimetic fish. In [113], DRL is used on a robotic fish for target tracking using visual feedback. Similarly in [77], it is used for attitude holding of a biomimetic fish in unknown flow fields.

2.2 Central Pattern Generators

Central Pattern Generators (CPGs) will be introduced in this part.

Periodic motion is a common phenomenon in living beings. Marine organisms are no exception. From internal organs such as heart that pumps blood by rhythmic beating, and lungs pumps in oxygen by continuous breathing. Movement of upper and lower limbs while walking and running, swiveling motion of wings in birds, and undulation and oscillation phenomena observed widely in marine life. All these observations include periodicity.

Parameters of rhythmic activity, like frequency, are modulated to obtain rapid response like heartbeat becomes faster when oxygen demand is high. Similarly, birds flap their wings faster when they need to fly faster and animals and humans move their limbs much faster when they need to run. Other parameters also change like profile of motion and amplitude of motion when required.

Researchers conducted investigations behind generation of periodic motions [114]. Different theories were proposed in the 20th century. One of them was proposed by Brown [115] states that periodicity is generated by a central system without requiring any necessary contribution from feedback from sensory organs.

CPGs are a network of circuits present in peripheral control system of vertebrates to provide rhythmic patterns for locomotion and other periodic activity [116]. They have also been found in invertebrates [117]. Mesencephalic Locomotor Region (MLR) is a region in brainstem that acts as a high-level control center for many activities in organisms [118]. Evidence suggests its role in controlling rhythmic activity [11, 119].

Fig. 2.2a illustrates the concept of CPG on a dorsal view of a salamander by indicating several CPG circuits in different parts of its body. Fig. 2.2b illustrates control system for a

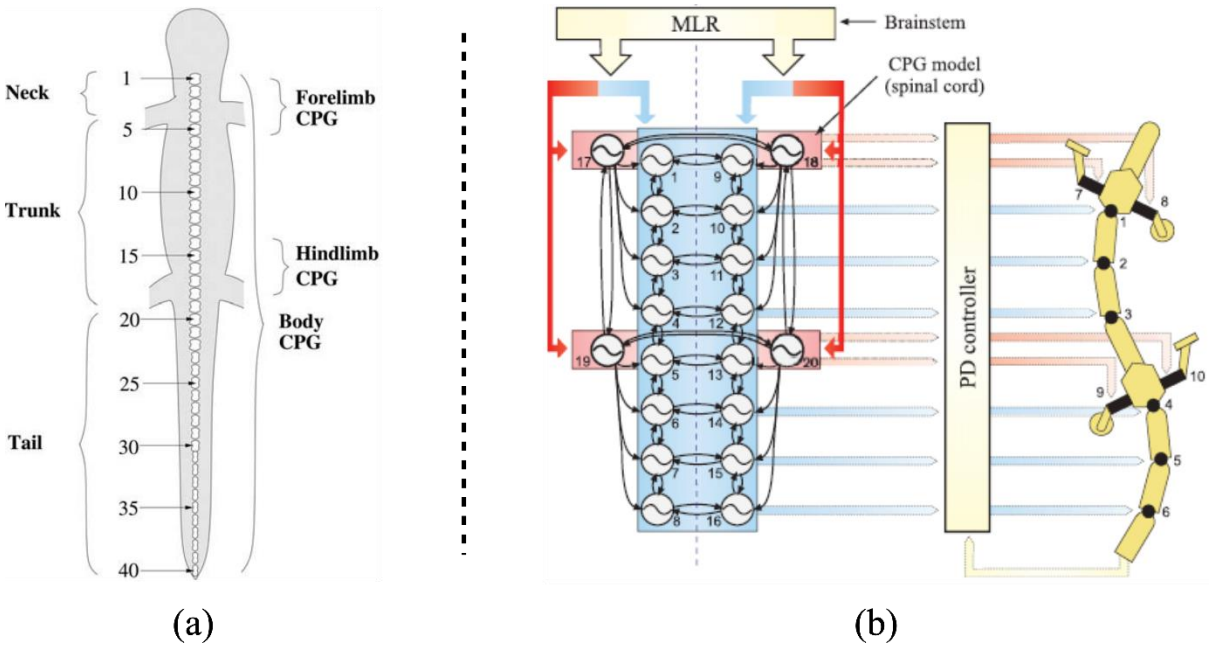


Fig. 2.3: (a) Illustration of Central Pattern Generators (CPGs) on a dorsal view of a salamander [4]. (b) Illustration of CPG implemented on a robotic salamander [9]. MLR stands for mesencephalic locomotor region – a region in brainstem that sends high-level commands to CPG circuits in the peripheral system to control locomotion [11].

robotic salamander inspired from CPGs and modulation signals from MLR. This system can control a robotic salamander to perform swimming and walking actions [9].

2.2.1 Mathematical Formulation

Different models have been proposed for generation of rhythmic signals [76]. One of the models based on oscillators is commonly used in robotics. Mathematical formulation proposed by [120] is based on a system of coupled oscillators. Similar schemes have been used by many works [9, 13, 43, 121] in robotics.

CPGs can be implemented as a network of coupled differential equations,

$$\dot{\phi}_i^t = 2\pi f + \zeta_i^t \tag{2.31a}$$

$$\zeta_i^t = \sum_j u_j^{t-1} c_{ij} \sin(\phi_j^{t-1} - \phi_i^{t-1} - \psi_{ij}) \tag{2.31b}$$

$$\ddot{u}_i^t = c_1 \left[\frac{c_1}{4} (U_i - u_i^{t-1}) - \dot{u}_i^{t-1} \right] \tag{2.31c}$$

$$\ddot{d}_i^t = c_2 \left[\frac{c_2}{4} (D_i - d_i^{t-1}) - \dot{d}_i^{t-1} \right] \tag{2.31d}$$

$$\alpha_i^t = u_i^t \sin(\phi_i^t) + d_i^t \tag{2.31e}$$

For each oscillator i in the network, u_i represents the amplitude while d_i and ϕ_i denote deviation and phase respectively. α_i is the actual output of the oscillator i. c_{ij} and ψ_{ij} specify couplings among the oscillators. The frequency f is rate by which rhythmic patterns are generated.

2.2.2 Related Work

Following are some examples about the use of CPGs in robotics.

In robotics, there have been many applications regarding the use of CPG for controlling bioinspired robots including quadrupeds, bipeds, and many other terrestrial and marine robots. We review some of the marine applications in the following.

In [43], gait optimization is performed for a multi-DOF snake like robot for both crawling and swimming locomotion. Optimization is performed on-the-go during simulation and not in an offline manner. Since two different locomotion formats are considered, the findings are interesting. It is shown that optimal gait parameters are no way close to each other when comparing both locomotion formats. The approach is shown to be better and faster than regular search methods. The CPG used is immune and is not affected by changes in parameters on-the-go and absorbs the sudden changes in parameters nicely.

In [122], muscular response for the development of robotic lamprey is studied and analogous motion is designed and studied using biomimetic CPG-like approach. The resultant force produced by the mechanism is quantified with respect to stroke size of the mechanism.

Similarly, in [123], a small fish capable of crawling and swimming is introduced. The fish is controlled by CPG. Different locomotion modes are performed for both crawling and swimming. And on-the-go switching is also performed which shows that the CPG is capable of absorbing sudden changes in the input parameters.

A 5-link robotic fish capable of 3D navigation in water is controlled using CPG [121]. Many locomotion modes are designed using CPG and on-the-go switching is performed between those modes. Depth control is also performed utilizing feedback from depth sensor. The work shows the possibility of developing a large repertoire of skills by using CPG-based approach.

An approach similar to [121] is adopted in [13], in which locomotion modes namely forward and backward are compared for efficiency and speed while varying the input parameters. A couple of interesting findings are shared. An improved CPG is used to control the robot in two locomotion modes (moving forward and backward). Motion profiles for the two modes are generated by incorporating feedback in the CPG. Hopf oscillator is used as CPG. Phase is altered to change the direction of swimming. A thorough comparison between the two swimming modes is given in terms of swimming speed at different input parameters of CPG. Turning performance is also quantified. Several useful results are deducted in the end such as forward mode performs faster, gets affected more by phase changes and encompasses broad range of frequencies.

2.3 Conclusion

In this chapter, individual components of the overall control schematics are introduced. DRL is introduced in Sec. 2.1. A comprehensive yet concise background for DRL is provided by introducing all the important definitions. There are many versions of DRL, and the focus is kept at TRPO since it is the method used in this study. A derivation is provided to show the

relevance between the original and the surrogate objective. And finally, approximations are done for the surrogate objective and the constraint (trust region) to come up with an update rule which is the primary objective of this section. After that, a pseudocode is formulated to be used in the rest of the study to update policy parameters.

CPGs are introduced in Sec. 2.2 as biological circuits. Analogous concepts are used in biomimetic robots where periodicity is predominant. After that a mathematical formulation is shared followed by some relevant studies from literature.

In the subsequent chapters, these two concepts will be used to formulate control schematics to first learn swimming gaits (Sec. 3) for biomimetic BCF type robot and then to learn to perform on-surface navigation (Sec. 4) and later to perform 3D navigation (Sec. 5). The formulations will be explained in detail in the corresponding sections.

CHAPTER 3

LEARNING SWIMMING GAITS

In this chapter, as described in Sec. 1.6 (*Developing motion modes*), control schematics and training methodology to develop different swimming gaits will be explained in detail. Training results of simulation will be shared, and those results will be verified on the hardware. Hardware setup, which was briefly introduced in the introduction chapter, will also be shared in detail.

Several studies related to reproduction of various swimming gaits on robotic platforms have been introduced in Sec. 1.4 (*Examples from Robotics*) where some robotic platforms from literature are shared and classified in separate categories w.r.t their gaits (Fig. 1.3).

In short, there have been efforts to mimic the anguilliform [10, 43], sub-carangiform [14, 58], and carangiform [15, 31] gaits on BCF-type robots. However, most existing works develop one particular motion pattern and study features related to one kind of swimmer. In this part, we plan to develop multiple motion patterns on the same platform to capture the benefits associated with different patterns. To achieve this, a control scheme is proposed which combines the two methods shared in the last chapter i.e., DRL and CPG. The schematics can successfully develop a range of swimming patterns.

3.1 Control Schematics and Algorithm

3.1.1 Motivation for schematics

As described in the background of CPGs (Sec. 2.2), oscillators or pattern generators are present in spinal cords of vertebrates [124]. The parameters of oscillators are controlled centrally by the midbrain. For example, stimulating midbrain electrically generates bimodal response in a salamander [11]. Therefore, the rhythmic patterns are modulated by a higher-level central system. This concept is demonstrated on a robotic salamander in [9].

Hence, gait generation is carried out at low-level and modulation commands come from high-level. Inspired by this multi-level process in nature, a similar bioinspired scheme is developed to explore the prospect of learning different motion patterns exhibited by a variety of swimmers in the BCF category. CPGs are used for low-level gait generation. For high-level learning and optimization, DRL is used to exploit and modulate CPGs to acquire diverse behavior.

3.1.2 Schematics

In the schematics (Fig. 3.1), DRL part is implemented as a multilayer perceptron (MLP) with 2 layers (64 neurons per hidden layer). The parameters of this network are updated during training using TRPO (Sec. 2.1). This network outputs parameters of CPG network namely the phase couplings – ψ_{ij}. Detailed training scheme will be explained in Sec. 3.3. Oscillators are realized by implementing the CPG network (Sec. 2.2).

During training, the schematic is implemented and executed sequentially. After the MLP outputs the parameters, the oscillators are integrated to generate commands for the robot. The robot executes those commands and updates the joint positions accordingly. The updated joint configurations are then saved in a queue along with a history of the last tail-beat cycle. This cycle is then analyzed to estimate phase difference between the joints. This information

is then concatenated with the desired force and fed to the MLP. The algorithm 3.1 is used for training.

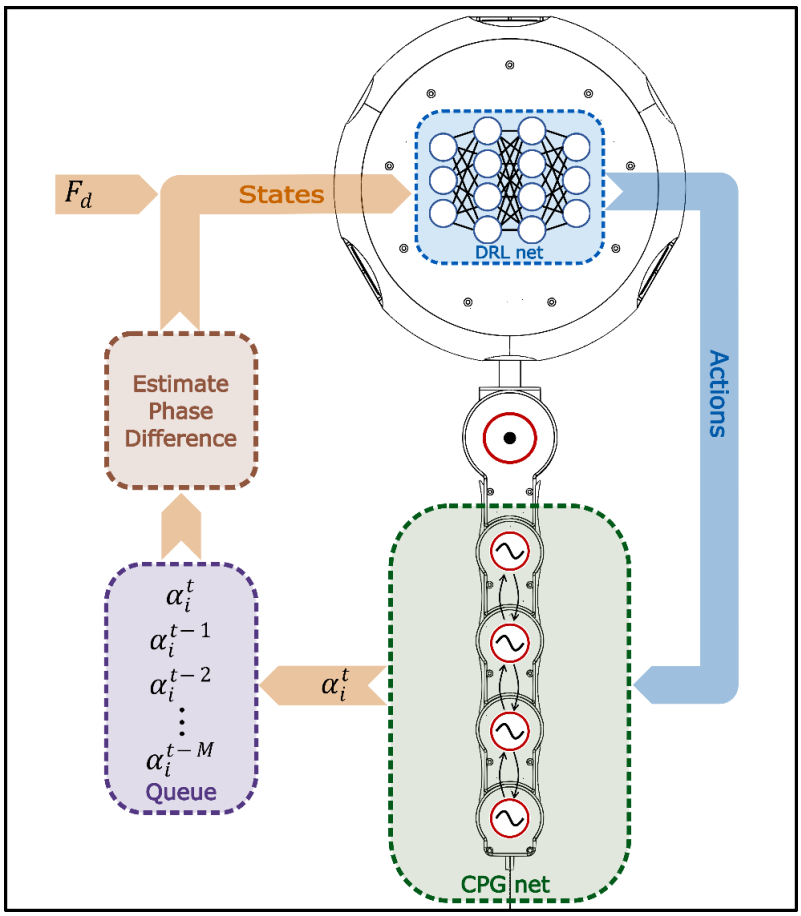


Fig. 3.1: Control Architecture overlayed on the sketch of the robot in the background. Empty circles in the DRL net indicate neurons. Circles with sinusoids inside in the CPG net indicate oscillators coupled by phase coupling indicated by small black arrows between them. α_i indicates the actual joint angles obtained via feedback. F_d represents desired force.

Some useful features or benefits of the framework are briefly mentioned here. Since the motion generation and modulation are handled separately, it is possible to fix the tail-beat frequency and let the method develop different swimming gaits based on different thrust targets at the same frequency. Once the gaits are developed, the frequency can easily be modified while maintaining any particular gait. This feature gives complete control on the speed of the robot and helps to compare the gaits with each other. Once the gaits are developed, the robot

can shift smoothly from one gait to another owing to smooth convergence properties of oscillators in short transient period. In the same way, other parameters of the oscillators can also be changed during runtime.

3.1.3 Algorithm

The algorithm is described in pseudocode form in Algorithm 3.1. The algorithm follows the same scheme as in control architecture (Fig. 3.1). It contains two parts. One is the DRL part or more specifically TRPO part from Sec 2.1 (Construction of batch (line 9) and policy updates lines 11 to 17). Second is the CPG part implemented in the form of oscillators from Sec. 2.2.

Algorithm 3.1: Algorithm to develop swimming gaits

1: Initialize policy π_θ with random parameter set θ.
2: **for** $1 \leq k \leq K$ **do**
3: **for** $1 \leq l \leq L$ **do**
4: **for** $1 \leq t \leq T$ **do**
5: Sample an action $a_t \sim \pi_\theta$
6: Input the action a_t to the CPG-net to obtain inputs α_i^t for the agent.
7: Render α_i^t to the agent and record the next state s_{t+1} and the reward r_t.
8: **end for**
9: Store the transition in a set of trajectories $\mathcal{B}$.
10: **end for**
11: Calculate returns G_t and advantage estimates A_t.
12: Compute policy gradient g_k,

$$g_k = \frac{1}{L} \sum_{\mathcal{B}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta (a|s) A_t$$

13: Compute the following relation,

$$x_k \approx H_k^{-1} g_k$$

14: H_k being the Hessian of the Kullback Liebler divergence (5b) between the old and new policies.
15: Compute step size,

$$\beta \approx \sqrt{\frac{2\delta}{x_k^T H_k x_k}} x_k$$

16: Update the policy parameters,

$$\theta_{k+1} = \theta_k + \eta\beta$$

17: η being the smallest value that satisfies the KL divergence constraint.
18: **end for**

3.2 Setup Description

We briefly explain our experimental setup in this part. There are three components of the overall setup – the robot hardware, the thrust measurement setup, and a pool for free swimming experiments.

3.2.1 Robot Hardware

Although the physical robot has been described in general in Sec. 1.5 (*BCFbot*). However, there are a few differences in the hardware used in Chapter 3 and 4. Hence, it is briefly explained in both parts.

The robot has a simple configuration. It exhibits the general anatomy of BCF swimmers. Fig. 3.2 shows the solid model of the robot. The head contains a Raspberry Pi (RPi 4B) microcontroller, and a battery to run RPi. Power for servos is supplied externally using a power supply. An external laptop communicates with RPi to send and receive high-level commands. The body part is composed of five 1-DoF joints in series. These segments (and the head) are made with machined plastic and have position servos inside to actuate them. At the end, there

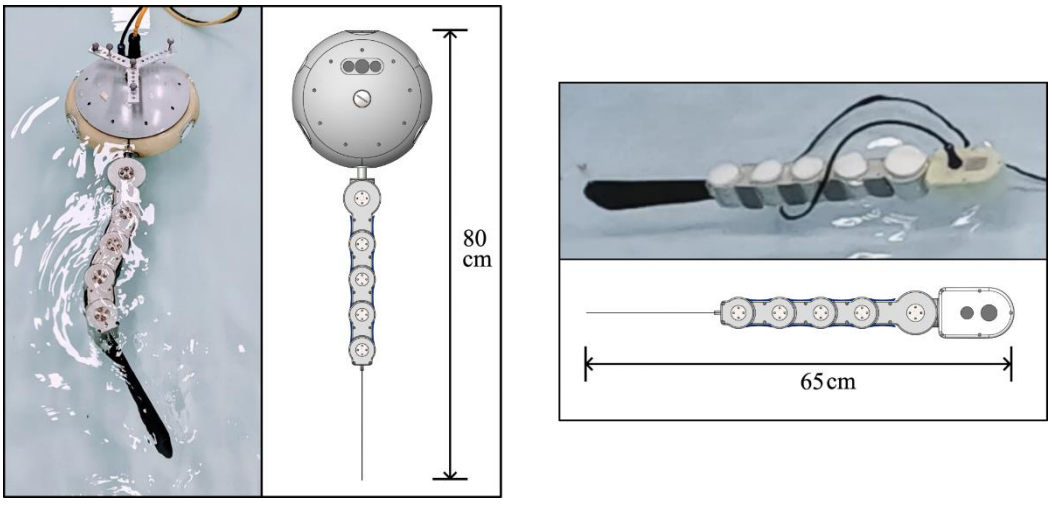


Fig. 3.2: Snapshots of physical BCFbot and its CAD model with big head (*left*) and small head (*right*).

is a caudal fin. The caudal fin is carved out of a carbon fiber sheet (0.7 mm thick). The fin is actuated by the actuator in the last joint.

3.2.2 Experimental Setup

Free experiments are conducted in a small laboratory swimming pool (2.5m x 4m). The water level is kept at around 1 ft. The buoyancy of the robot is set to keep the robot's body (tail and fin) just under the surface of water as can be noticed in Fig. 3.14. On the corners of the pool, four tracking cameras are installed at 3m height. Infrared reflective markers are installed on the head and the fin of the robot (can be seen in Fig. 3.14). The setup can track the position of the robot while it swims in the pool. The tracking data is then used to get the trajectory and velocity information of the robot.

For thrust measurement, a setup (Fig. 3.3) is designed for this work. A static structure is erected inside the pool. A rod is attached to the robot's head. An arrangement is made using bearings to let the rod rotate around x and y axes passing through the center of rod. The bearings are affixed to the static structure, due to which the rod and the robot itself cannot translate in any direction. The other end of the rod comes in contact with a load cell attached to the structure. This scheme transfers the forces developed by the robot to the load cell. Data from load cells

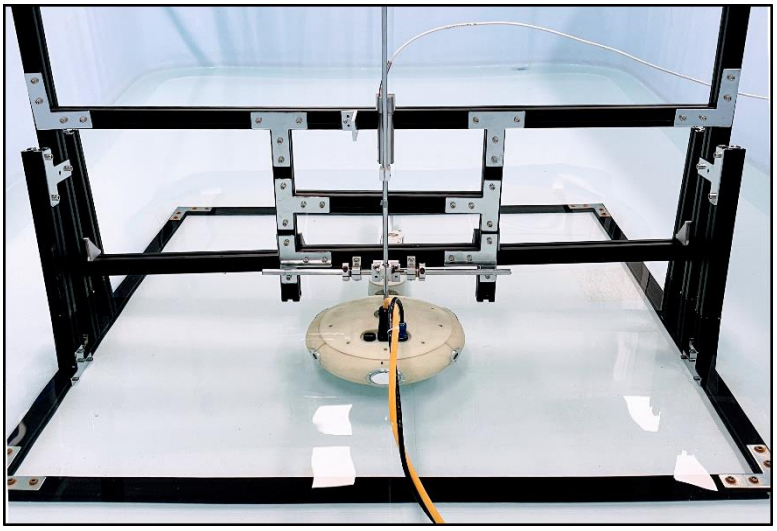


Fig. 3.3: Thrust measurement setup.

is first amplified and then recorded using a data recorder. A camera is installed on the top to capture waveforms induced in the robot's body.

3.3 Training Methodology

This section deals with the learning process adopted to train the agent in simulation and simulation results.

In DRL, a large number of trials are required for training. It is a usual practice to train the networks in simulation first. A physics emulator [125] is used to simulate the agent to perform training. A simulated version of the platform is modelled in the emulator. It is modelled in the same way as the hardware having four joints fitted with position servos. All joints are single degree of freedom as we consider planar swimming (x-y plane) on the surface of water in this work. In the simulation environment, the properties of the medium (e.g., density and viscosity) are set to that of water.

The rhythm generator renders a travelling wave through the body. When this wave travels back from the anterior part of the body to the fin, the robot should move forward in the direction opposite to the travel of wave. During training, however, the translational degrees of freedom i.e., along x and y axes are locked and therefore, the robot stays in a plane. A force

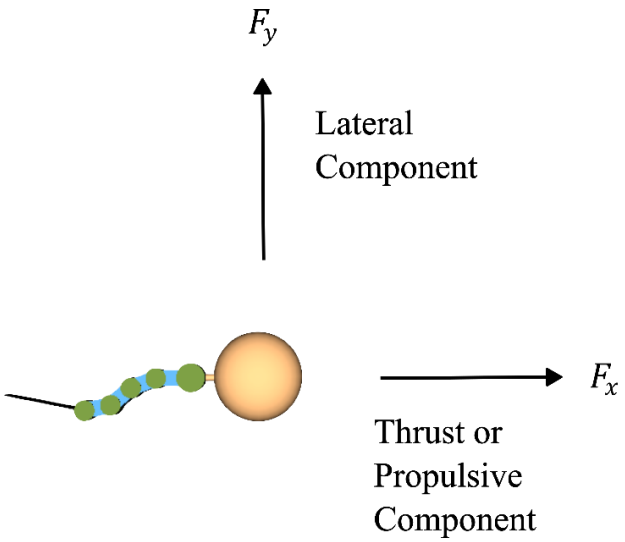


Fig. 3.4: Trust or propulsive and lateral component of forces.

sensor is installed in the global frame of reference at the location where the center of the head lies. Since the robot cannot translate (as mentioned above), the forces produced by the agent will be recorded by the force sensor. This force has two components. The component along the longitudinal axis – F_x is the thrust or propulsive component since it is primarily responsible to propel the robot if it is allowed to translate. F_y is the lateral component perpendicular to the thrust component (Fig. 3.4 and Fig. 3.5).

This force is used in the reward function to train the agent for the desired force targets. DRL will optimize the policy according to different conditions imposed on the force in the reward function. The architecture presented in control schematics is used to train the policy according to the pseudocode given in Algorithm 3.1.

In the reward function, the difference between the actual and the desired force is minimized. It can be put in the following way,

$$r = -\|F_d - F_a\|^2 \tag{3.1}$$

where

$$F_a = \sum_{i=n}^{m} \sqrt{{F_{x_i}}^2 + {F_{y_i}}^2} \tag{3.2}$$

Here F_d represents desired set point for the force whereas F_a is the actual force recorded by the sensor. When the tail undulates, the joints move according to travelling wave profile and return to the original configuration after one tail-beat cycle. The duration of the cycle depends on tail-beat frequency. A cycle will take 1s if the frequency is 1hz. The actual force is averaged over one complete tail-beat cycle, $(m - n)$ represents the length of the cycle. To generate a higher thrust force (F_x), at fixed frequency, swimming pattern has to change. This change will also alter F_y since both components are coupled together and one cannot change completely independent of the other. Same will be the case if a higher lateral force is desired. Using one component provides little room for exploration but using both components provide large room

for exploration in parameter space and therefore helps development of different patterns. It will be shown in the form of difference in combined force (Sec. 3.5).

It should be noted here that the average thrust can be controlled directly by controlling the frequency of oscillations. This method is commonly used in literature [30] to control the

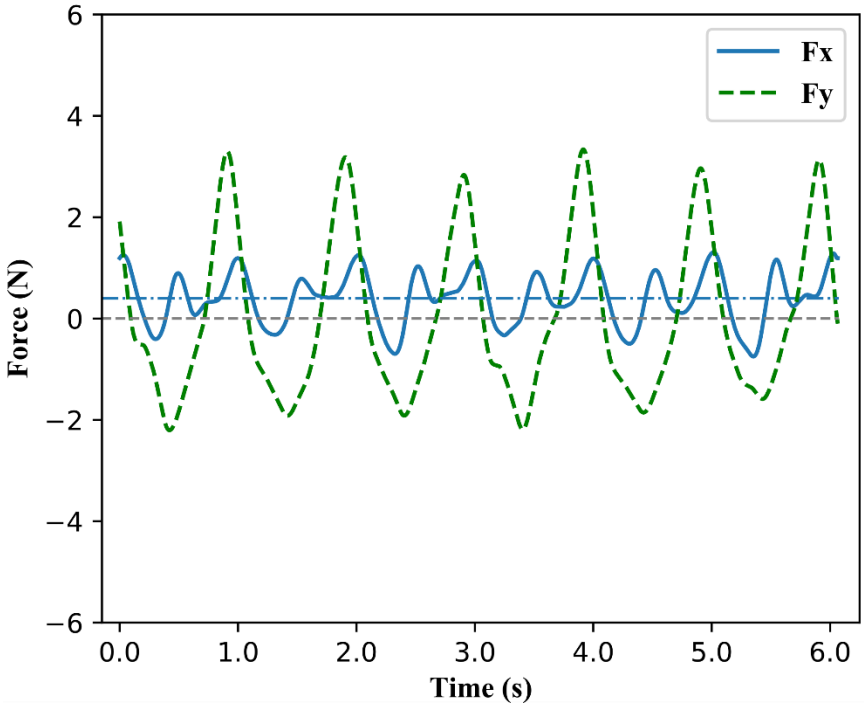


Fig. 3.5: Periodic force patterns developed by the robot while swimming with the anguilliform pattern at 1.0hz for 6 seconds. Blue (solid) line represents the thrust component – F_x (mean value ~0.8N) whereas green (dashed) line represents the y-component – F_y which oscillates around zero.

thrust and the speed of such robots. The goal here is to develop different motion patterns. To avoid the neural networks from learning frequency to thrust relationship, the frequency of oscillations is kept constant throughout the training process. In this way the control architecture has to explore other means – different swimming gaits to deliver different target presets for the force. This is the key idea that enables the robot to learn and adopt multiple motion patterns.

To develop varied patterns, we make use of F_y in addition to F_x. It can be observed from Fig. 3.5 that for each tail-beat cycle, the average of F_y remains zero for straight line motion

contrary to F_x, average of which has to be non-zero for forward motion. At low frequencies, the effect of F_y is visible in the form sway of body about the longitudinal axis. But at high frequencies, this effect is not very visible. In real fishes, this effect is also mitigated by the flat shape of the fishes posing high drag resistance perpendicular to the longitudinal axis but a strong F_y component does exist. This component, in addition to F_x, is also critical in determining the swimming gait, and due to this reason, it is included in the reward function. Later it will be shown that the combined average force of both components increases as we move from the anguilliform mode to the carangiform mode.

3.4 Visualization and Comparison of Performance among Swimming Gaits

3.5 Visualizing Swimming Gaits

To observe the motion patterns, we remove the restriction previously imposed during training and let the agent translate. Then, the robot can swim forward due to the propulsion it generates by the continuous wriggling of the body. To visualize the patterns, we will first have a look at static postures of the agent (Fig. 3.6a and 3.7a). We also record the trajectories traced by the fin of the agent while it swims shown by the periodic patterns in Fig. 3.6b and 3.7b. The fin trajectory provides a good way to visualize and compare swimming gaits. Following the same color scheme as in Fig. 1.2, the data in blue, brown, and green colors in Fig. 3.6 and 3.7 correspond to the anguilliform, sub-carangiform, and carangiform motion patterns respectively. Fig. 1.2 has been readded on the top of Fig. 3.6 for convenience.

3.5.1 Simulation Results

The three results from the left to right in Fig. 3.6 are corresponding to the force targets (F_d) from 1.8 N, 3.0 N, to 4.2 N respectively. For the lowest target (1.8 N), the motion pattern in Fig. 3.6a (left) looks similar to the anguilliform pattern shown in Fig. 1.2 or 3.6 (*top*). The wavenumber is the largest as almost half of the propulsive wave is contained in the agent's body. As the set point for the desired force is increased by 1.2 N, the resemblance of tail configuration shifts to the sub-carangiform and eventually to the carangiform pattern on further increase of 1.2 N. These poses should be observed along with Fig. 1. The agent now contains almost one quarter (Fig. 3.6a-center) and even less than one quarter (Fig. 3.6a-right) of the travelling wave within its body.

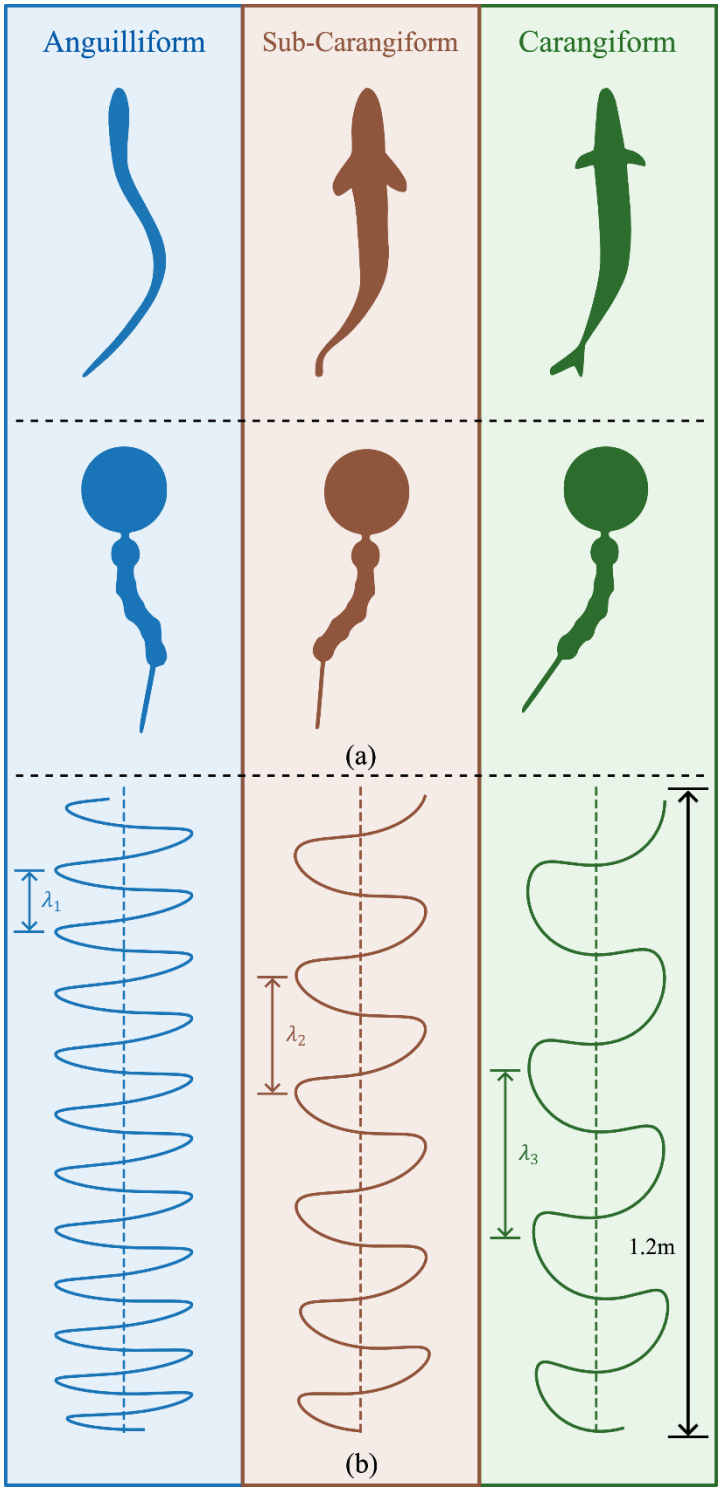


Fig. 3.6: *Top*: Canonical swimming modes from Fig. 1.2. (a) Snapshots of the robot in simulation when the framework renders the anguilliform (left), sub-carangiform (center), and carangiform (right) motion patterns after training. (b) Fin tracks traced by the fin of the robot in simulation while it swims with the patterns shown in (a).

Fig. 3.6b shows fin trajectories for the three cases. The wavelength (λ_1=12.2 cm) is the shortest in the anguilliform pattern. The waveform starts to expand, and the wavelength

increases to 24.5 cm in the sub-carangiform pattern and finally to 38cm in the carangiform case. Therefore, as the desired force targets increase, the wavelengths increase. The ratio of the wavelengths from the anguilliform to sub-carangiform pattern is 0.5 and from the sub-carangiform to carangiform pattern is 0.6. Later, these ratios will be compared with the hardware results.

3.5.2 Hardware Results

Hardware results are summarized in Fig. 3.7. The pose on the left (Fig. 3.7a) is the anguilliform pattern. It has similar body and fin configuration as the one in simulation, Fig. 3.6a (left) whereby the tail of the robot contains one half of the travelling wave. The pose in the center (Fig. 3.7a), corresponds to the sub-carangiform pattern and is similar to the body posture in Fig. 3.6a (center). Lastly, the tail configuration on the right (Fig. 3.7a) looks similar to the carangiform pattern and resembles Fig. 3.6a (right). The hardware poses in all three cases conform with the simulation results. One can observe the similarity of hardware poses (Fig. 3.7a) and simulation poses (Fig. 3.6a) with the classical BCF modes introduced in Sec. 1.3.

Next, fin trajectories are compared between the simulation (Fig. 3.6b) and hardware experiments (Fig. 3.7b). The one on the left is acquired when the robot swims with the anguilliform pattern whereby the wavelength is the shortest (13cm) among all. It starts to expand when it swims with the sub-carangiform pattern (24cm) and becomes the largest in the case of carangiform pattern (40cm) thereby giving the ratios of λ_1 and λ_2 to be 0.54 and from λ_2 to λ_3 to be 0.6. These ratios match closely with the ones in simulation (Sec. 3.5.1).

Next, force measurement results on the hardware are shared in Fig. 3.7c. These results are not individual component of forces, but the combined force of both components calculated by $F_c = ({F_x}^2 + {F_y}^2)^{1/2}$, same as the definition of F_a in (3.2). Combined force (F_c) is shown by solid and its average ($\overline{F_c}$) by dashed lines. For the anguilliform case, the average is around 1.7N.

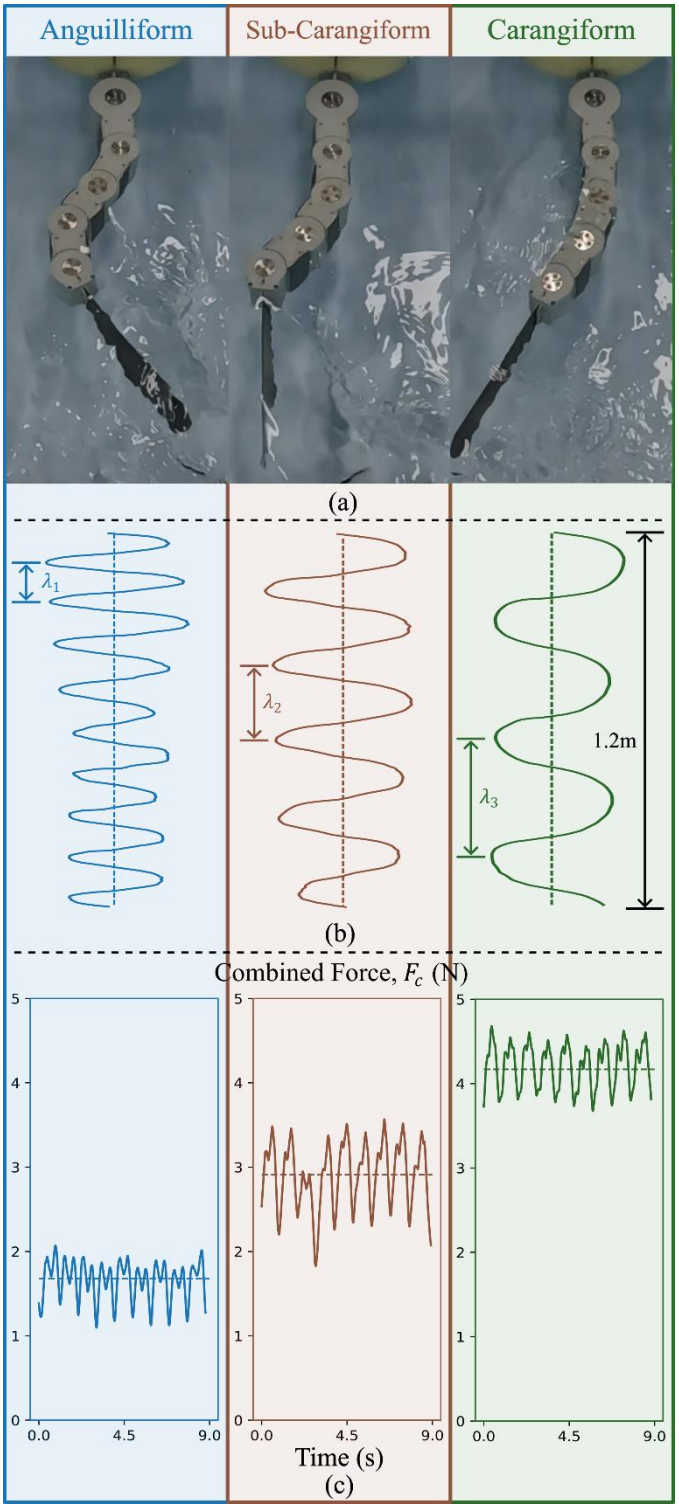


Fig. 3.7: (a) Snapshots of tail when the anguilliform (left), sub-carangiform (center) and carangiform (right) patterns are rendered on the robot hardware while it is attached to the thrust measurement setup. (b) Fin trajectories traced by the fin of the robot while swimming with the patterns shown in (a). (c) Combined force output F_c(N) of the robot while swimming with the three patterns in (a).

It rises to 2.9 N and 4.1 N for the sub-carangiform and carangiform cases. There exists a clear difference in the combined average forces in the three cases which makes it easier for the

control framework to explore multiple gaits in the parameter space.

It should be noted from Fig. 3.6b and 3.7b that the fin trajectories expand from left to right. It can also expand if the tail-beat frequency is increased. If the frequency is increased, the body will wriggle faster, and the agent will swim faster resulting in an expanded fin track compared to the one generated at low speed when observed for a fixed distance of travel. But the expansion phenomena shown in Fig. 3.6b and 3.7b is not due to any change in tail-beat frequency. The frequency is kept constant (at 1.0 hz) in all three cases. The presence of 9 periodic cycles in the time duration of 9 s (Fig. 3.7c) also confirms that the tail-beat frequency remains same in all the cases. Hence, the expansion and the increase in force is purely due to changes in swimming gaits. Since the frequency is kept constant in all the cases, the system cannot simply increase the tail-beat frequency to increase the output force. To cope up with increasing force targets, the DRL part must evolve the swimming gait to generate larger thrust and lateral forces. In doing so, it develops different swimming gaits.

Force components for the anguilliform case are shown in Fig. 3.5 and for the sub-carangiform and carangiform cases in Fig. 3.8. Change in force profiles can be observed. The nature of profiles is characteristic to each swimming gait. For the first two cases, the rise or fall in F_y is gradual whereas for the last case, it stays at maxima before falling rapidly to the next minima. The maximum amplitudes of both components increase when the motion pattern changes from the anguilliform to carangiform pattern. The average value of F_x increases from the anguilliform to carangiform case and remains zero for F_y.

Due to change in force components/profiles, the performance of the robot in terms of linear and angular speed varies a lot. In the next section, all the gaits will be compared by performing free-swimming experiments whereby the robot will swim with different gaits and power consumption and linear/angular speeds will be measured.

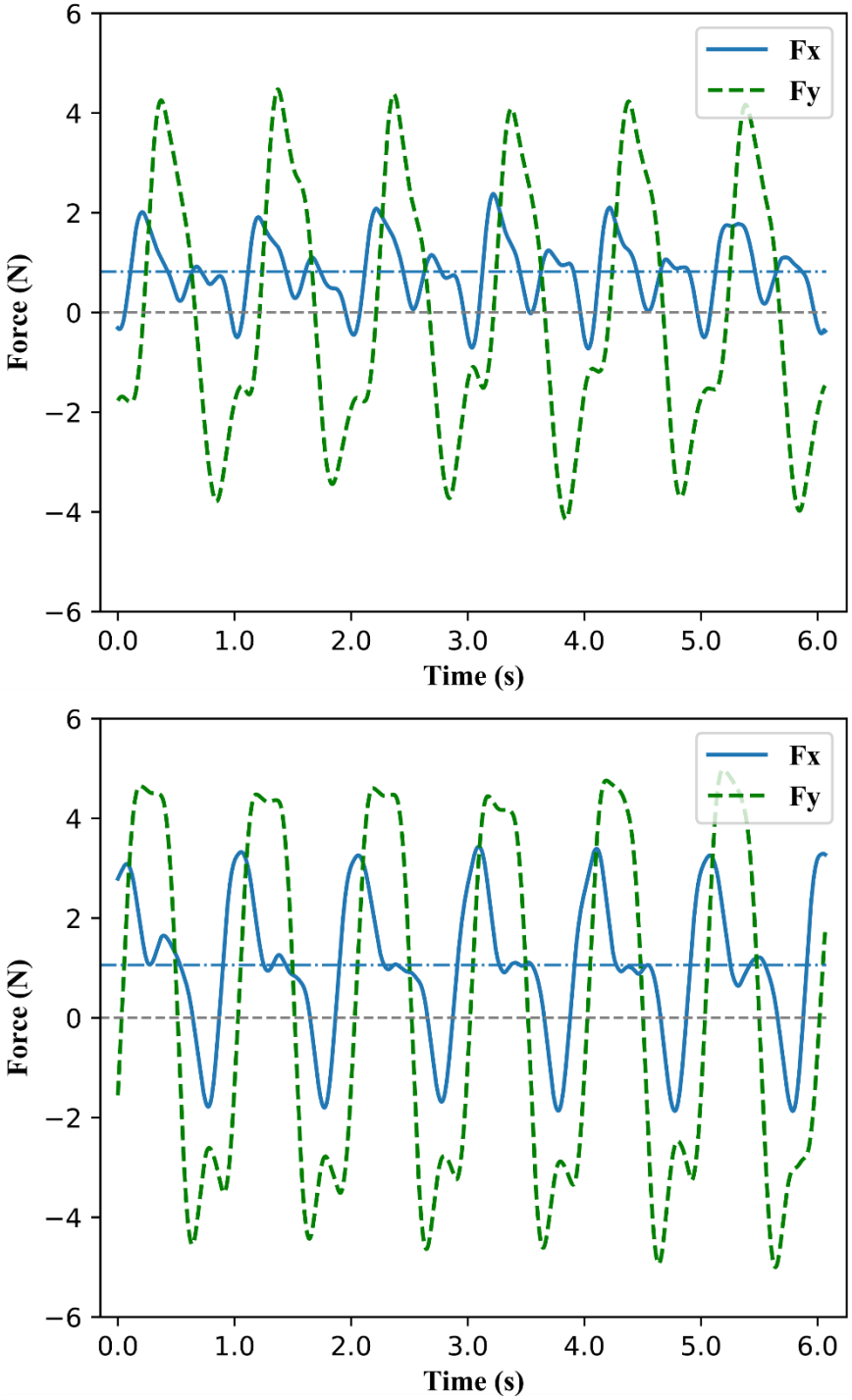


Fig. 3.8: Periodic force patterns developed by the robot while swimming with the sub-carangiform (*top*) and carangiform (*bottom*) pattern at 1.0hz for 6 seconds. Blue (solid) line represents the thrust component – F_x (mean value ~0.8 N) whereas green (dashed) line represents the y-component – F_y which oscillates around zero.

3.6 Comparison of Swimming Performance for different Swimming Gaits

Three swimming gaits have been developed and visualized in the previous section. In this section, these patterns will be compared in terms of their performance including speed and efficiency by performing free swimming experiments. For this part, markers are installed on the head of the robot to measure its linear and angular velocities. The power consumed by the robot is also recorded to analyze energy efficiency of the swimming gaits.

Frequency is a key parameter to control the speed of such biomimetic robots. Using DRL-only scheme, it is not straightforward to control the speed on biomimetic platforms. Since this method involves a network that has to be queried at every iteration. The frequency is usually limited by the frequency at which the microcontroller can query the policy. In the proposed framework, since motion generation is handled separately by a central pattern generator (Fig. 3.1), it is possible to control and vary the key parameters of undulatory/oscillatory motion e.g., the period and amplitude of oscillations can be varied independent of the query rate of policy in the learning part of the framework. When a different speed is required, the architecture helps to alter the speed of the robot by adjusting the period of generator while maintaining a certain motion pattern.

3.6.1 Straight-line Swimming

The speed of the robot at different frequencies while swimming with the three patterns is compared in Fig. 3.9 (left). The carangiform pattern has a sharp rising trend whereas the slope of the trend decreases as we move towards anguilliform. Overall, we observe the anguilliform pattern to be the worst performing with speeds ranging from 12 cm/s to 17 cm/s. The sub-carangiform case follows, giving a minimum of 22.5 cm/s at 0.7hz and a maximum of 36 cm/s at 1.6 hz. The carangiform pattern performs the best with outputs ranging from 30cm/s

to 45 cm/s. We restrict the maximum frequency to 1.6 hz due to hardware limitation of actuators. Using DC motors instead of servos, much higher frequencies may be possible [30].

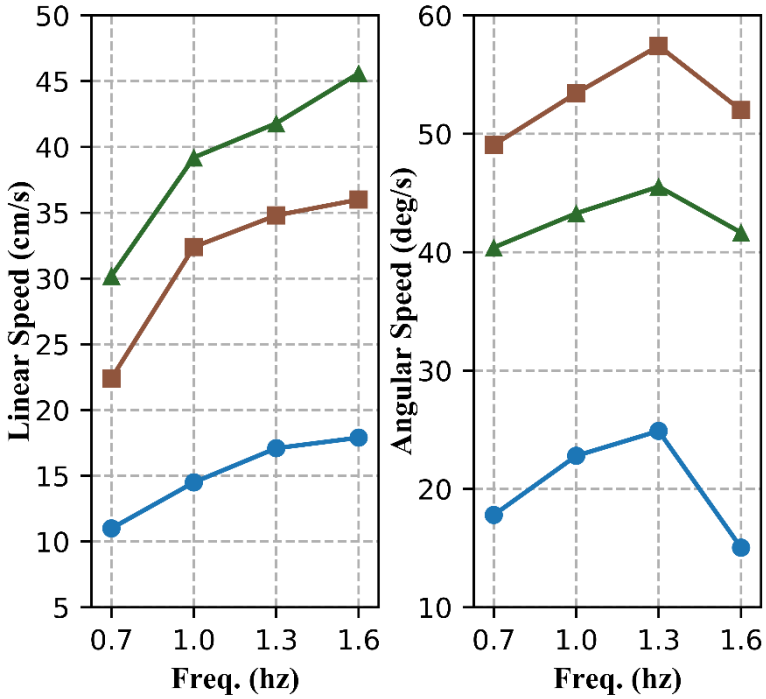


Fig. 3.9: Linear and angular speed comparison of the robot while swimming with different motion patterns at four different tail-beat frequencies. Blue (●), brown (■), and green (▲) markers represent the robot's linear (*left graph*) and angular (*right graph*) speeds while swimming with the anguilliform, sub-carangiform and carangiform modes respectively.

The values in Fig. 3.9 represent the average values when the robot attains a steady state value. While the average value at steady state condition is important, the velocity profile can also provide useful information about each swimming gait. Fig. 3.10 shares the measured velocities when the robot swims with different gaits. It can be observed from the profiles that the highest average is in carangiform mode but since the carangiform mode is close to the oscillation side of BCF spectrum, the variation around the average is more than the rest of two patterns. The linear speeds of the sub-carangiform and anguilliform modes are consistent with less variation about the average values.

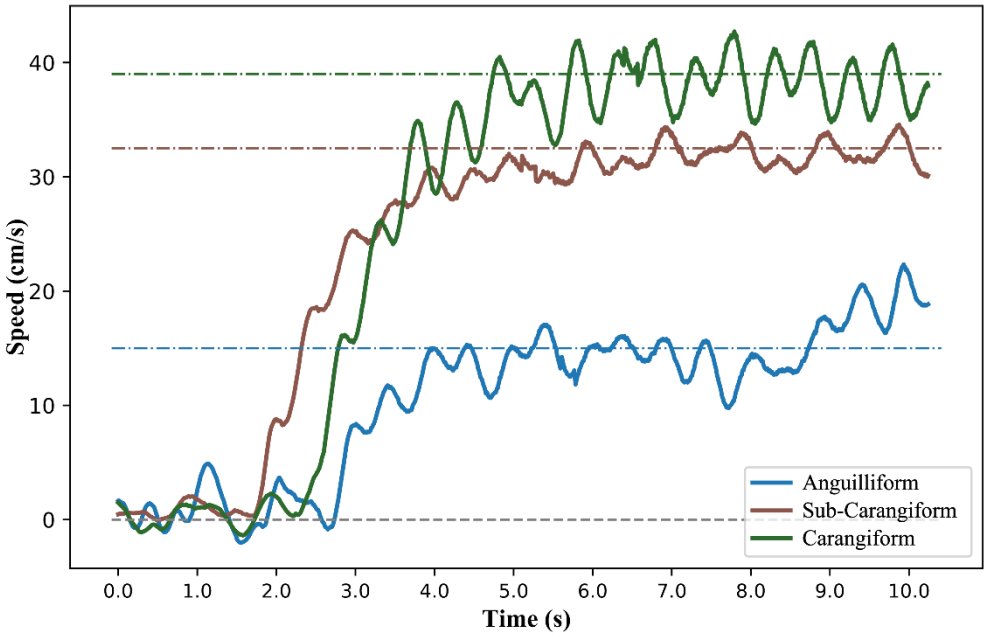


Fig. 3.10: Linear velocities of BCFbot when it swims with the anguilliform (*blue*), sub-carangiform (*brown*) and carangiform (*green*) swimming gaits at tail-beat frequency of 1.0hz. The steady state averages are 14, 32.5, and 39 cm/s respectively.

Cost of transport (CoT) [30] is a parameter that can be helpful to compare the efficiencies of the patterns,

$$CoT = P_{in}/v \tag{3.3}$$

where P_{in} is the power consumption of the robot while swimming with linear speed v. Power is calculated by logging voltage and current data from the external power supply. CoT is calculated for all the cases compared above for linear speed and the results are shared in Table 3.1.

CoT decreases with increasing tail-beat frequency. The anguilliform mode is the most expensive owing to its poor linear speed performance whereas the sub-carangiform pattern is the most economical among all giving the lowest value of 0.67 J/m at 1.6 hz.

Table 3.1

Frequency (hz)	$f = 0.7$	$f = 1.0$	$f = 1.3$	$f = 1.6$
Motion Pattern	**Cost of Transport-CoT ($\times 10^2$ J/m)**			
Anguilliform	1.29	1.16	1.17	1.23
Sub-Carangiform	**0.88**	**0.73**	**0.71**	**0.67**
Carangiform	1.15	0.85	0.82	0.77
Motion Pattern	**Cost of Turning-CoTu ($\times 10^2$ J/rad)**			
Anguilliform	0.61	0.58	0.53	0.96
Sub-Carangiform	**0.32**	**0.30**	**0.28**	**0.32**
Carangiform	0.51	0.47	0.44	0.46

3.6.2 Turning

To quantify turning performance we use similar architecture and input non-zero constant deviation (D_i = [0.0, 0.3, 0.3, 0.3] rad) (2.31e) to the rhythm generator. This deviation adds a bias to the signals generated by the CPG. Fig. 3.9 (right) summarizes the turning performance in terms of turning rates observed at 4 different frequencies while the agent employs the three gaits.

In all cases, the angular velocity increases with the increase in tail-beat frequency. The increase is almost linear in all cases up to 1.3hz. However, after 1.3 hz, there is a fall in angular velocities in all cases. This effect will be explained later. Unlike straight-line swimming speed results, sub-carangiform gait performs the best in the turning case with a maximum turning rate of 57 º/s at 1.3 hz. The anguilliform pattern yields very low turning rates giving a maximum of 25 º/s only. The carangiform pattern lies in the middle of two with steering rates ranging from 40 º/s to 45 º/s.

Unlike the linear motion case, turning at high frequency does not necessarily mean that it would be faster. In straight-line swimming, when the tail undulates, it generates propulsion

for the entire tail-beat cycle. This continuity breaks during turning. When the tail oscillates, there is a pushing phase in which the robot uses its fin to push water to turn itself. Then, there is a retraction phase in which the tail retracts back. This retraction creates an effect opposite to the desired turning direction. Lower frequency allows the robot enough time to make complete use of extension phase. At higher frequency the retraction phase breaks the angular momentum before the pushing phase is completed. This is the reason behind the fall of angular turning rates after 1.3 hz (Fig. 3.9 right). This effect is also observed and discussed in [126].

Like Fig. 3.10, Fig. 3.11 shares angular velocities measured for the three cases when the BCFbot takes turn at tail-beat frequency of 1.0 hz. Like the linear velocity case, the variation of angular speed around the mean value is minimum for the anguilliform gait and maximum for the carangiform gait. For turning, as mentioned above, the sub-carangiform gait performs the best giving the mean turning rate of around 55 deg/s.

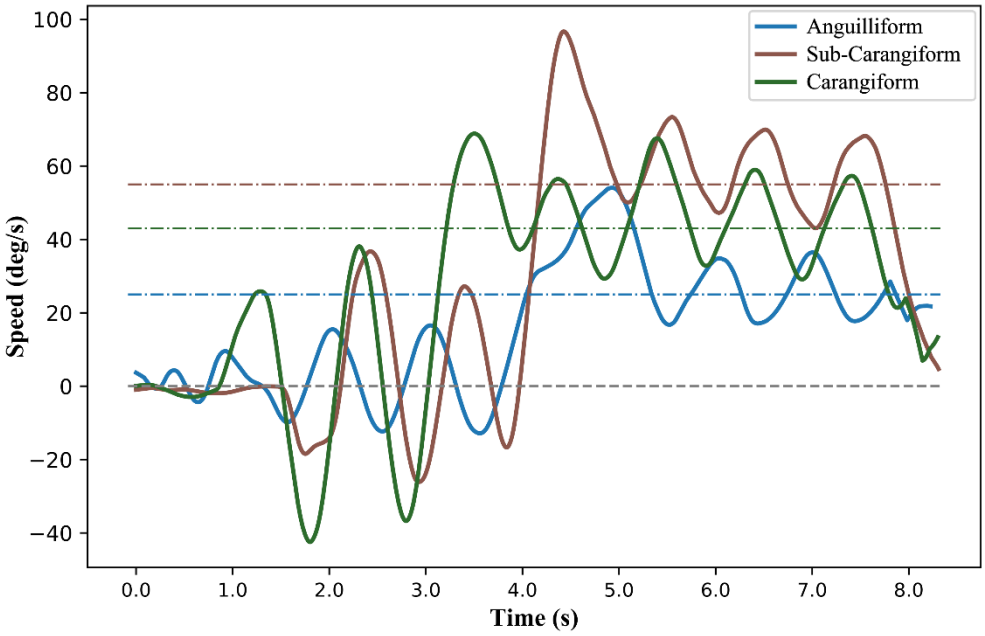


Fig. 3.11: Angular velocities of BCFbot when it swims with the anguilliform (*blue*), sub-carangiform (*brown*) and carangiform (*green*) swimming gaits at tail-beat frequency of 1.0 hz. The steady state averages are 23, 43, and 54 deg/s respectively.

In literature, we do not find any yardstick to rate turning performance on such platforms. Therefore, in the style of cost of transport (CoT), we calculate cost of turning ($CoTu$).

$$CoTu = P_{in}/\omega \tag{3.4}$$

where ω is the angular velocity of the robot. The results are shared in Table 3.1. The cost is minimum for the case of sub-carangiform mode and the anguilliform is the most expensive mode for turning as well.

3.7 Backward mode and Free-Swimming Snapshots

3.7.1 Backward Mode

As discussed in the introduction, one distinguishing feature of some anguilliform swimmers is their ability to travel backward. They can switch the direction of heading by switching the direction of travelling wave instantly. This gives them a unique feature that other swimmers from the BCF category do not possess. We explore the prospect of whether our

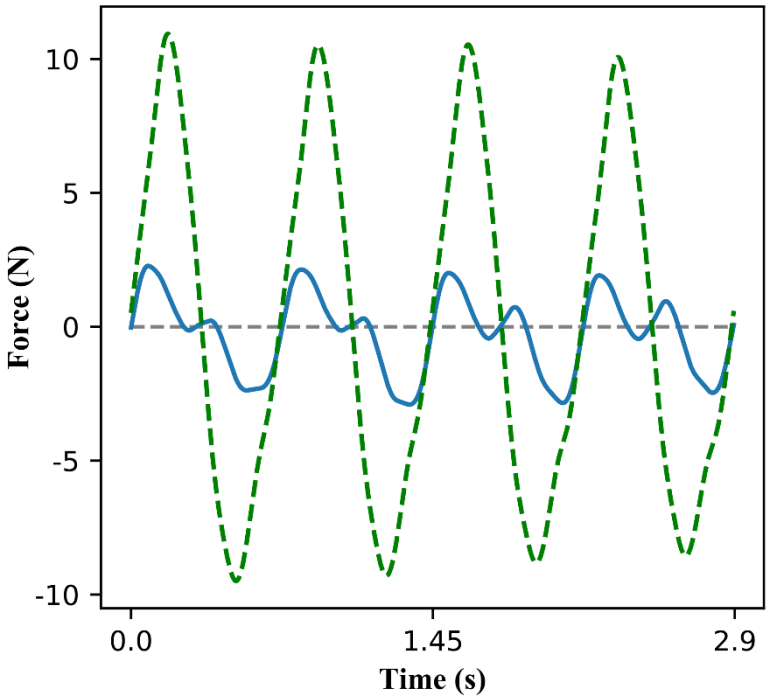


Fig. 3.12: Thrust F_x (solid blue) and lateral F_y (dashed green) forces developed by the robot undulating in backward mode while attached to the thrust measurement setup.

method can develop a wave travelling in opposite direction i.e., originating from the fin, travelling towards the anterior part of the body, and eventually terminating at the head. Upon rendering such wave in the body, the robot will move in opposite direction i.e., towards the direction of fin. For this purpose, we set the target for the target force to be negative.

We observe that the method successfully optimizes the pattern generator such that it generates a forward travelling wave thereby generating negative average thrust. The corresponding result can be observed in Fig. 3.12.

It is difficult to measure the forces during backward motion. The robot on the thrust measurement setup is held from the head due to which it cannot undulate thereby generating very low thrust output (~0.2 N in the negative direction). Then, a free-swimming experiment is conducted for the backward mode. The backward velocity observed is shared in Fig. 3.13. The average speed in the backward mode is less than 8 cm/s.

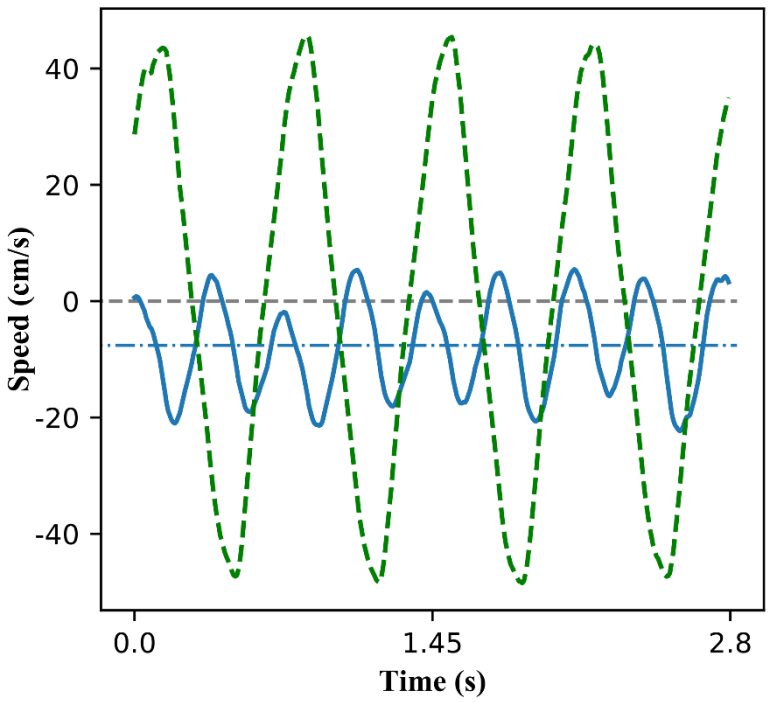


Fig. 3.13: Velocities V_x (solid blue) and V_y (dashed green) while the robot is swimming freely in backward mode.

3.7.2 Free-Swimming Experiments

Some stills from the swimming experiments are shown in Fig. 3.14 and Fig. 3.15. Fig 3.14a, 3.14b and 3.15a represent the cases for the anguilliform, sub-carangiform, and carangiform cases respectively. The robot starts from the left side of the pool (left snapshots) and swims towards the right side. Tail-beat frequency in all three cases is same (1.0hz) but the linear speeds for the three cases vary (as shown in Fig. 3.9) due to which the robot takes different time intervals to swim for the same distance. The speeds are about 15 cm/s, 32 cm/s, and 40 cm/s respectively.

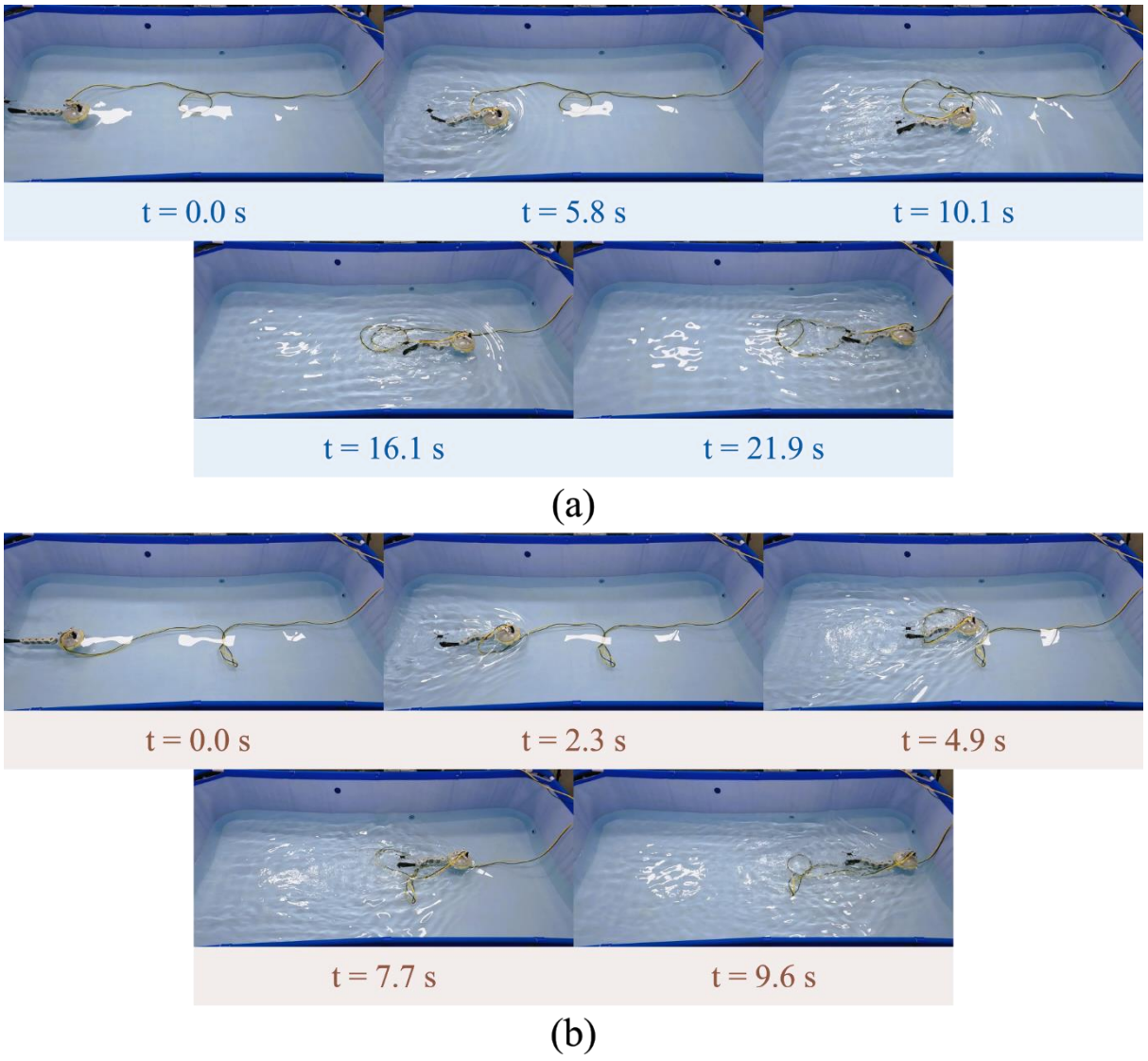


Fig. 3.14: Stills from free-swimming experiments for the anguilliform (a), sub-carangiform (b) motion modes. Time stamps are mentioned at the bottom of snapshots. (a) and (b) are from left to right.

The sub-figure (Fig. 3.15b) is for the backward swimming case. In this case, the robot starts from the right side, and swims in backward mode to the left side. The fin is in the front of the robot when it swims in backward mode. The motion pattern in this case is like the anguilliform pattern. The hardware that has been used so far has a big and heavy head and it is difficult for the robot to swim with a heavy posterior end. So, a smaller head (Fig. 3.2-right) is used in this case. The slow speed in this case is due to the fact that there is no fin at the point where the travelling wave terminates.

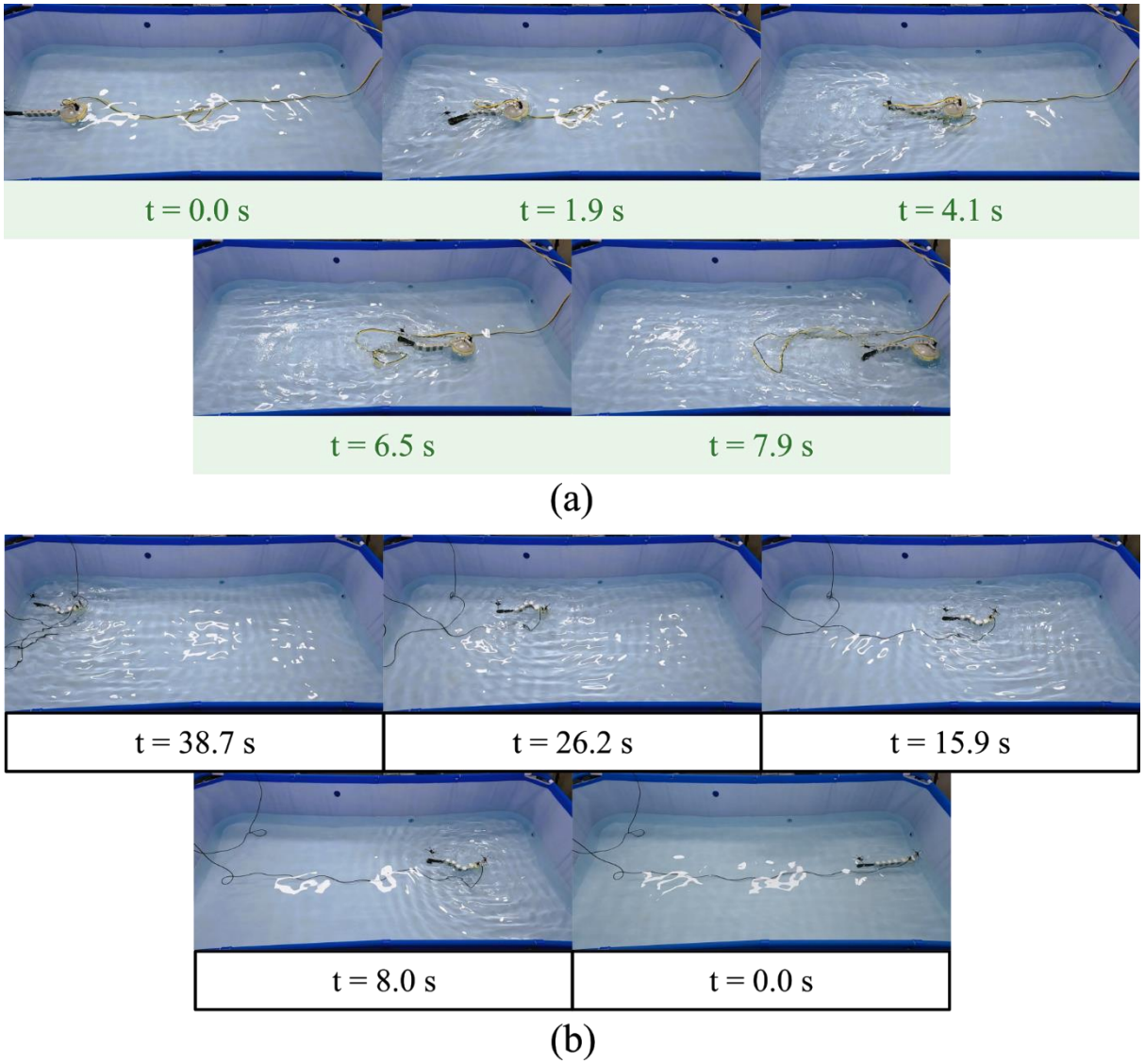


Fig. 3.15: Stills from free-swimming experiments for the carangiform (c) and backward (d) motion modes. (c) is from left to right whereas (d) is from right to left.

3.8 Conclusion and Discussion

The control schematic presented in this chapter trains the robot in simulation to learn different swimming modes. Those modes are then tested on the physical robot and compared with each other in terms of their performance. It is revealed that all the developed swimming gaits have certain benefits as follows. The following Fig. 3.16 summarizes this chapter. Table 3.2 shows results quantitatively and ranks the swimming gaits from worst to best for their linear and angular speed performance.

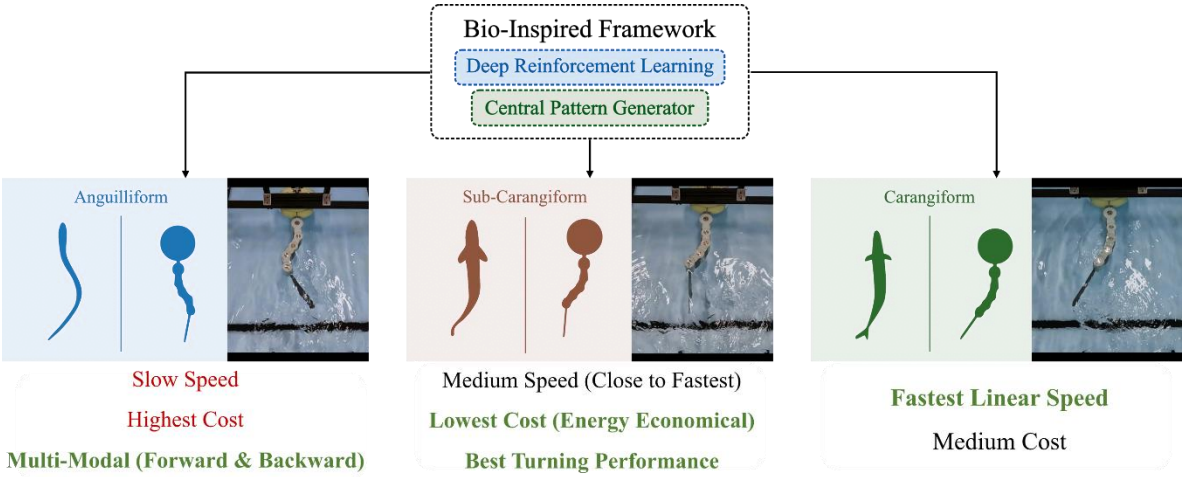


Fig. 3.16: Figure summarizing results of the three swimming gaits.

Table 3.2

Linear Speed Comparison			
Frequency	f = 0.7 hz	f = 1.6 hz	Performance
Motion Pattern	Linear Speed (cm/s)		
Anguilliform	12	17	Worst
Sub-carangiform	22.5	36	Intermediate
Carangiform	**30**	**45**	**Best**
Angular Speed Comparison			
Frequency	f = 0.7 hz	f = 1.6 hz	Performance
Motion Pattern	Angular Speed (deg/s)		
Anguilliform	17	25	Worst
Sub-carangiform	**49**	**57**	**Best**
Carangiform	40	45	Intermediate

Following results are concluded from the finding in this chapter,

1) The carangiform mode produces the fastest linear speed but the power consumption in this mode is also very high which inhibits it to be the most economical mode; Sub-carangiform swimming gait for its low wattage consumption stands out in terms of economy; In terms of speed, although it is not the leading mode, it follows the leading mode closely.

2) Similarly, predominance of undulatory feature in the sub-carangiform mode makes it an ideal candidate for turning as well. Carangiform pattern lies a step further towards the oscillatory end of the spectrum makes it an ideal candidate for straight-line swimming but not for turning. This means that the turning phenomena is better achieved with a mixture of undulatory and oscillatory modes rather than pure modes.

3) The anguilliform mode performs worse among all the three modes in both linear and angular cases. But the backward swimming in this mode gives the robot a unique ability. Hence, all three modes have benefits of their own.

After a thorough study on a large number of BCF swimmers, [6] suggests that the classical kinematic classification of BCF swimmers into four major modes should better be considered as a continuum. This suggests a unified BCF scheme for efficient propulsion and motivates development of a flexible and generic control architecture for robots mimicking BCF anatomy. A scheme that can tap the benefits of not an individual but a range of organisms. This study takes a step in this direction by proposing a scheme that generates a spectrum of motion patterns. DRL seems to be an ideal candidate to develop such a spectrum. It offers a framework that helps train artificial agents like their biological counterparts [96]. We have seen many studies in recent years in which bio-inspired robots are trained to act like their natural equals

using this method [91]. In DRL, joint exploration and exploitation allows it to keep on exploring new ways while exploiting the efficient ones already explored. In this study, it is coupled with CPG to develop multiple swimming patterns similar to ones observed among natural BCF swimmers for a robot that mimics BCF anatomy.

In this work, one platform is used so that only the difference between the motion patterns can be visualized without any change in hardware. The objective is not to find out the best swimming pattern with respect to speed or any other metric, but to develop a unified method that can generate a range of patterns. However, there is almost no work done for this purpose.

One of the most used methods for such biomimetic designs is called Lighthill's fish body wave equation [63]. It can be expressed as,

$$y(x,t) = (c_1 x + c_2 x^2) sin\,(kx + \omega t) \tag{3.5}$$

The parameters of the above system can be adjusted to fit some specific waveform but only to a limited extent.

The results in [61] and [127] render the anguilliform, carangiform, and thunniform motion patterns on the same robotic fish but demonstrate slight difference in performance with averaged linear speeds of 17, 21, and 22 cm/s for the three cases. The results are compared in Table 3.3.

Table 3.3

Speed Comparison (cm/s)					
Motion Pattern	[61][114]	%age rise	This study (0.8 Hz)	%age rise	This study (1.6 Hz)
Anguilliform	17.1	-	12	-	17
Sub-Carangiform	21.8	27.4%	22.5	**87%**	36
Carangiform/ Thunniform	22.1	1.4%	30	**33%**	45

The results in this study report obvious difference and better performance quantitatively. It can be observed that the percentage rise in Anguilliform to Sub-Carangiform is 27.4% whereas it is just 1.4% in case of sub-carangiform to thunniform. In this study, the rise is 87% and 33%. This high rise sufficiently demonstrates that the proposed method can better reproduce swimming gaits.

Rhythmogenic circuits or pattern generators have been studied in animals for a long time [76, 116]. Central control system plays a role in modulating these generators to acquire desired behavior [9]. It raises a question whether the one same central control system can generate patterns exhibited by different swimmers. To explore this question, in this work, a similar bioinspired scheme is realized whereby a central system commands pattern generators in peripheral system to act in a certain way to achieve a desired behavior. It turns out that the central network learns to optimize and modulate the pattern generator to extract multi-modal behavior upon training the central network for multiple objectives. It is observed that different patterns similar to the anguilliform, sub-carangiform and carangiform modes emerge upon setting propulsive targets from low, medium to high.

CHAPTER 4

PLANAR NAVIGATION

In this part, navigation of the robot on the surface of water is detailed out. The focus in this chapter is not development of motion pattern or swimming gait. One swimming gait is adopted in the entire chapter and the focus is kept at navigating the robot along desired trajectories. A desired trajectory is first generated in the form of a set of path points for the robot to follow. The robot, in simulation, learns to follow those points one by one to complete the trajectory. In doing so, whenever it reaches a path point, it receives a positive reward. Similarly, it receives a penalty when it fails to reach a path point. After several thousand iterations, the robot learns to track both previously known and unknown trajectories [128].

In this chapter, related works related to the methods used in this part will be reviewed (Sec. 4.1). The proposed method will then be discussed in detail along with training process, simulation results, and hardware experiments. In the end, some benefits of the proposed method and some future directions will be discussed.

4.1 Related Work

Navigation is challenging in marine domain due to a range of challenges discussed in Sec. 1.1. Some notable efforts related to biomimetic platforms are reviewed as below.

In [129], a turtle like platform with six flaps is used with two front facing cameras and a rear camera. Convolutional networks are used to process visual inputs followed by a deep network to learn states action relation. Sparse waypoints are given as input. The robot follows those waypoints while avoiding obstacles. The robot is able to navigation in close vicinity of coral reefs. In [80], a multi-joint robotic fish is used to perform navigation. There are no on-board cameras. External setup is used to track the robot and define path points for the robot to follow. A mixed framework with DDPG implemented by an MLP model followed by an oscillator network to generate periodic inputs for the joints is used. Target path points are defined in series for the robot to follow. The method is compared with traditional techniques such as proportional-integral-differential, sliding mode, and active disturbance rejection controllers and is shown to be efficient in terms of energy consumption. On a robotic dolphin in [130], a scheme similar to [80] is performed whereby visual navigation is performed using binocular images. Since binocular images are rich in information and contain distance information for the objects in frame. A stereo network is proposed to use the parallax to effectively use the parallax information.

For navigation in this chapter, a control schematic based on Dynamic Motion Primitives (DMP) and DRL is introduced. In the following some literature related to DMPs is reviewed.

DMPs are a feasible and generic way to represent and encode complex motion behaviors. Mostly learning from demonstrations strategy is used to encode different motion profiles and is mostly used where it is easy to obtain demonstrations e.g., from human supervisors [89] or from mathematical models [131].

In [88], a library of motion primitives is created first using a human supervisor, after that mixture of motion primitives strategy is proposed to perform in scenarios not covered during supervision. In this way, primitives are generalized to previously unseen scenarios.

Whereas, in [89], demonstrations are fitted into primitives using reinforcement learning on a robotic manipulator for trajectory traversing and pick and place task. Later DRL is further used to generalize the demonstrations to new goal positions and to optimize the motions where initial demonstrations are not efficient. [132] use convolutional neural network to optimize pre-trained primitives directly from images from a camera setup. For mobile robots, since demonstrations are difficult to obtain, we find a few efforts using DMPs. [133] use primitives but still baseline behavior is obtained using CPGs and is further optimized using actor-critic learning architecture. Whereas, [131] obtain primitives using a mathematical model of the robot.

CPGs can also be considered a special case of DMPs [134]. Literature related to CPGs has been reviewed in Sec. 2.2. The proposed framework in this part also uses DRL which has been detailed out in Sec. 2.1.

In the proposed method, it is aimed to use DRL to train DMPs from scratch without using any demonstrations or model for baseline behavior. This combined approach preserves the benefits of DMPs while adding the potentials of DRL and thus has multiple benefits. First, it does not require supervised setup and avoids the time-consuming step to collect demonstrations to build a library. Instead, the system learns from experience gained during the training process using simple reward functions. Second, it allows to use generalization properties attributed to DMPs such as spatial and temporal invariance which are of particular importance to biomimetic robots. Hence, the proposed approach will have wide applicability.

4.2 Schematics for Navigation

The schematic used for navigation is different from the one used for swimming gaits optimization introduced in Chapter 3. This scheme includes combination of DMPs with DRL. The schematics will be introduced in this section, and the training methodology and reward functions will be shared along with simulation (Sec. 4.4). First, a brief background of DMPs is given. Then, the combined architecture will be introduced. DRL has already been introduced in Chapter 2.

4.2.1 Dynamic Motion Primitives

A set of non-linear differential equations is employed to create and represent DMPs. Our construction is similar to the one presented in [135],

$$\tau \ddot{y}_i = a_1(a_2(g_i - y_i) - \dot{y}_i) + u_i \tag{4.1}$$

$$\tau \dot{\phi} = 1 \tag{4.2}$$

The system (4.1) can be integrated to obtain output trajectory y_i. The weights a_1 and a_2 are set to make (4.1) critically damped so that the system converges towards goal g_i. The construction can be extended or reduced feasibly according to the DoFs of the application involved. (4.2) is called canonical system with ϕ as the phase of the primitive. u_i is commonly referred to as a forcing function which can be used to encode desired behavior into the above system. This behavior can be discrete (point attractor) or rhythmic (limit cycle) or a mixture thereof. Rhythmic forcing functions are used since oscillatory patterns are required for the joints to develop undulations in the tail.

$$u_i(r_i, \phi) = \frac{\sum_{j=1}^{N} \psi_j w_j r_i}{\sum_{j=1}^{N} \psi_j} \tag{4.3}$$

$$\psi_j = \exp\left(h_j\left(\cos\left(\phi - c_j\right) - 1\right)\right), \tag{4.4}$$

r_i is the amplitude of the forcing function u_i and w_j are the weights associated with the basis function – ψ_j . Normally, linear weighted regression (LWR) [88] or some other technique is used to find appropriate weights to fit the forcing function according to some demonstration signal. DRL is used to train the weights without using any guiding trajectory but by using reward signal only. Values of the constants are mentioned in Table 4.1.

4.2.2 Control Schematic

Two policies and a set of three motion primitives are used as shown in Fig. 4.1. Thrust-policy π_1 is trained first. This policy is used to train the weights w_j (4.3) of the three motion primitives used for the three joints of the robot. After training of policy π_1 is done, the robot learns to develop travelling wave in its tail to swim in a straight line (forward direction). These motion primitives, whose weights have already been trained by π_1, are further trained by navigation-policy π_2 for their goals g_i (4.1) to train the robot for navigation along random trajectories. Training chronology will be explained in detail in the next section.

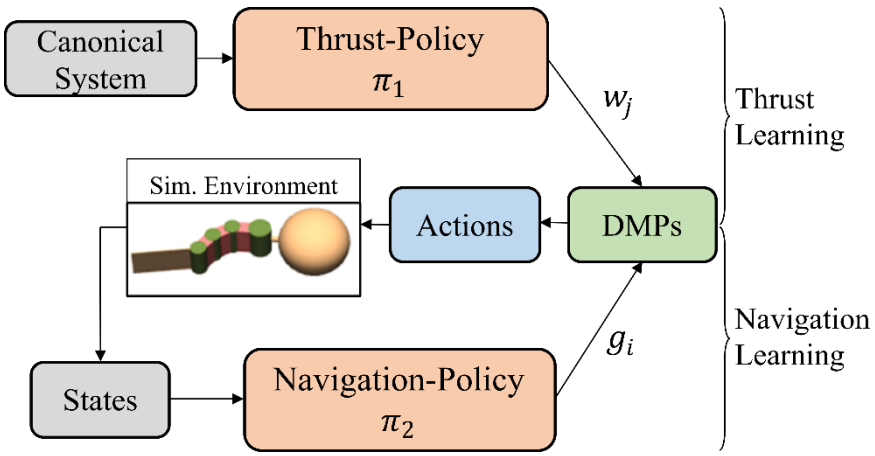


Fig. 4.1: Control schematic showing both thrust-learning and navigation-learning. The training is done sequentially (refer to Sec. 4.4).

Both policies, π_1 and π_2, are non-linear function approximators – DNNs with 2 fully connected hidden layers of 64 units. The sizes of input and output layers differ according to the state and action spaces of π_1 and π_2 and will be mentioned in the next section. They have parameters θ_1 and θ_2 which are trained using TRPO by optimizing the relevant functions (Sec. 2.1) given in the background section.

4.3 Setup Description

Although the robot hardware has already been introduced in Chapter 3.2.1. The hardware used in this part of the work is somewhat different from the one used in Chapter 3. The basic design remains the same. A few changes are highlighted below.

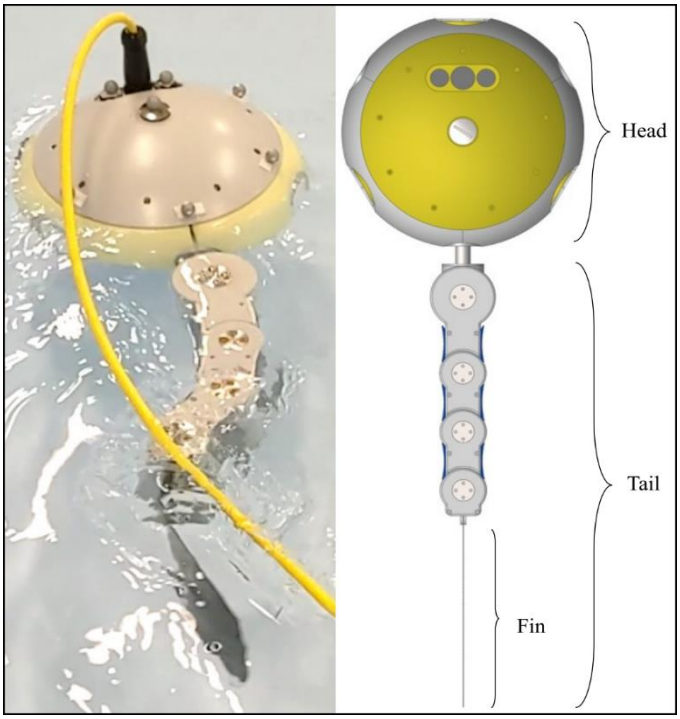


Fig. 4.2: *Left*: Physical robot hardware. *Right*: Top view of the 3D model of the robot.

The robot hardware, as shown in Fig. 4.2, consists of a head and a tail having **four** 1-DoF joints (instead of five). The joints are actuated with same position actuators (servo motors) inside them. Like before, there is a micro-controller (Raspberry Pi 4B) inside the head that communicates with an external computer (laptop) using **a wireless connection** and sends

commands to the motors. Therefore, it can be observed (in Fig. 4.9) that there is no network wire while performing hardware experiments. The robot is powered from an external power supply like in the previous case. On the physical robot, the tail can rotate about its own axis and the robot is capable of diving underwater. But for this part, we consider planar locomotion on the surface of water. Therefore, the rotational joint is locked in place. Additionally, the first joint in the tail is also locked at center position and only the last three joints are considered for the purpose of simplicity and to keep the focus on navigation.

For simulation, like Sec. 3, a model of the robot is created in MuJoCo – Multi-Joint dynamics with Contact [125]. MuJoCo is a very accurate physics engine to emulate contact forces and behavior dynamics. Using this engine, we can simulate swimming and flying phenomena by setting viscosity and density parameters corresponding to the medium involved. The simulated model is shown in Fig. 4.1 and Fig. 4.4.

4.4 Training and Simulated Experiments

We will now explain the training process, individual tasks handled by the policies and simulation results. Training is done sequentially as explained below.

4.4.1 Thrust-Learning

First, the upper part of the control schematic (indicated as thrust-learning in Fig. 4.1) is considered and policy π_1 is trained. It is called thrust-policy because it trains the agent to render travelling wave in the tail which is the source of thrust for the robot to swim.

To set the state space, we follow [89] in which the phase of the canonical system is chosen as state space. The action space is the weight vector w_j (4.3). These weights are used to obtain forcing functions which are then used to obtain motion primitives (4.1) to calculate the position commands for the three joints. Following reward function is used,

$$r = r_f - c_{rot} \tag{4.5}$$

where $r_f = c_f v_x$

and $c_{rot} = c_r \alpha_z$

r_f is forward-reward which encourages the robot to move forward in the x-direction and is primarily responsible to develop undulation pattern in the tail. It utilizes v_x, the component of velocity of the head along the x-direction multiplied by c_f (forward-reward weight). c_{rot} is a penalty that inhibits unnecessary rotation of the head. It penalizes the reward using α_z, the angle by which the head rotates about z-axis (perpendicular to the plane of motion) scaled by rotation-penalty weight c_r. The resultant motion is critical to c_{rot} term and varies significantly when c_r is tweaked. This network is trained for 40k episodes where one episode is of 1250 timesteps. The reward curve is shown in Fig. 4.3. The reward converges in almost half of the episodes.

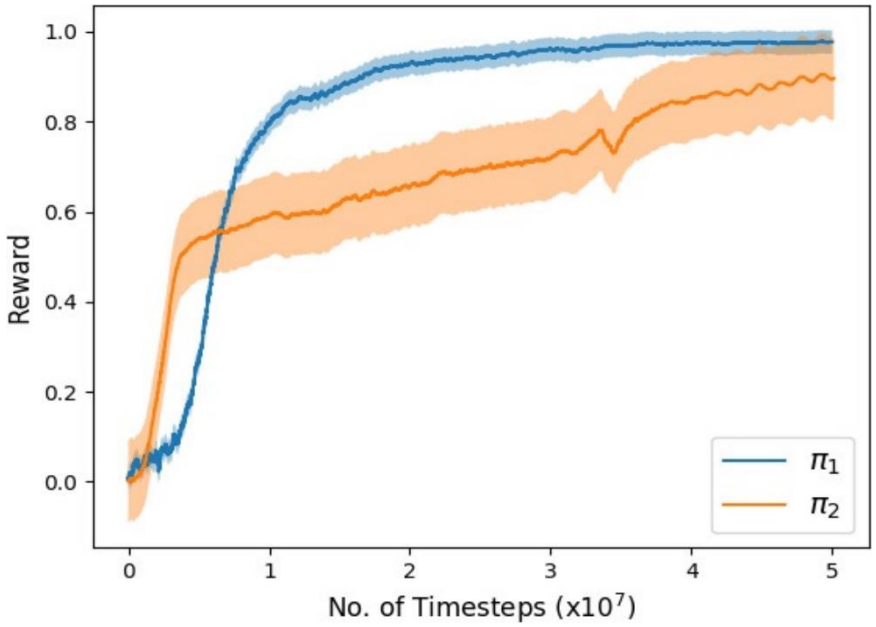


Fig. 4.3: Scaled reward curves for thrust-policy π_1 (Sec. 4.4.1) and navigation-policy π_2 (Sec. 4.4.2).

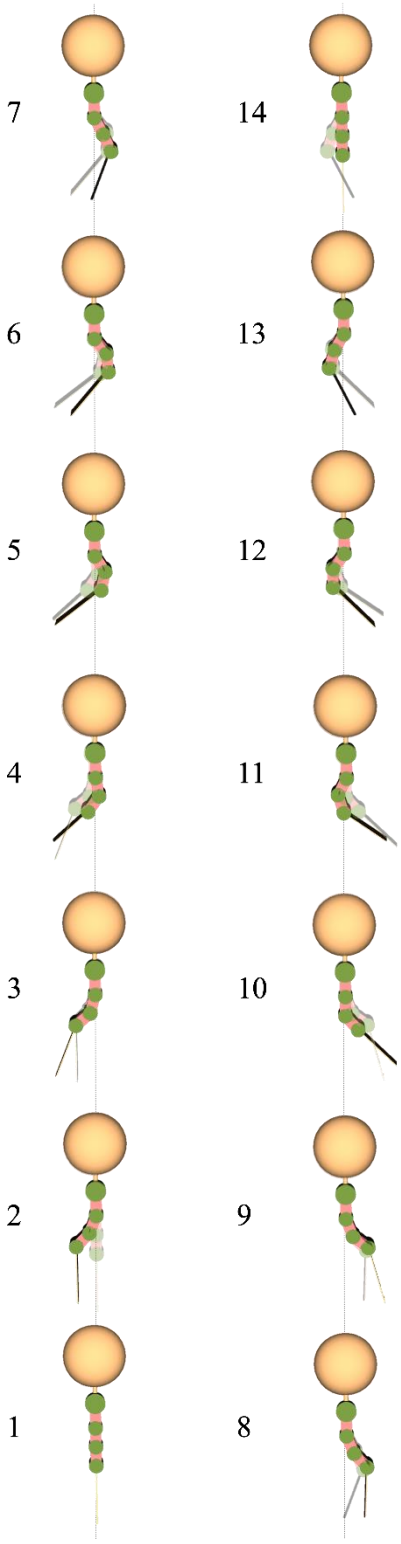


Fig. 4.4: Simulation results of trained thrust-policy π_1. Snapshots showing one complete cycle of travelling-wave generated by the motion primitives trained by the policy π_1 (Sec. 4.4.1).

Resultant policy is tested in simulation. Simulation result can be seen in Fig. 4.4. It shows the motion pattern of the tail when this policy is run on a fixed simulated model. 14 snapshots are added to show one cycle of undulation pattern in the tail. The undulation pattern can be considered as anguilliform [82] as it is distributed evenly across the entire tail. It can provide sufficient thrust to move the robot and works well both in simulation and on the actual robot (Sec. 4.5).

4.4.1.1 Algorithm for thrust-learning

The following algorithm is used in this part.

Algorithm 4.1: Thrust learning

Initialize policy π_1 with random parameter set θ_1.
Initialize parameters of DMPs i.e., goals g_i and weights w_j to null vectors of sizes $|g|$ and $|w|$.
for $1 \leq k \leq K$ **do**
    for $1 \leq l \leq L$ **do**
        for $1 \leq t \leq T$ **do**
        Sample a weight set $w_j \sim \pi_1$.
        Input w_j to primitives and integrate primitives to generate inputs for the agent a_i.
        Render a_i to the agent and record the next state s_{t+1} and the reward r_t.
        end for
    Store the transition in a set of trajectories $\mathcal{B}$.
    end for
Update the policy parameters θ_1 according to TRPO algorithm (from Sec. 2.1).
end for

In Fig. 4.6, T_1 shows the trajectory traversed by the center of head when the agent is run using the motion primitives trained by this policy. The motion is straight line since this policy trains the primitives only for straight swimming. Swimming along arbitrary paths will be done in the next part. The policy is also tested on the hardware details of which will be shared in the hardware testing section (Sec. 4.5).

4.4.2 Navigation Learning

After the thrust-policy is trained and the primitives that develop undulations in the tail have been learnt, we proceed to the lower part of the control schematic (i.e., navigation learning in Fig. 4.1). During this phase, the undulation pattern is provided by the motion primitives trained by π_1 and another network π_2 optimizes the goals g_i (4.1) of the motion primitives for the navigation task to allow the robot to swim along a number of randomly generated paths.

4.4.2.1 Algorithm for navigation-learning

The following algorithm is used for navigation learning.

Algorithm 4.2: Navigation learning

Initialize policy π_2 with random parameter set θ_2.
Initialize weights w_j from Algorithm 4.1.
Initialize goals g_i to null vector.
Generate a random path T_i according to path generator (described in Sec. 4.4.2) for the agent to follow.
for $1 \leq k \leq K$ **do**
    for $1 \leq l \leq L$ **do**
        for $1 \leq t \leq T$ **do**
        Sample a goal set $g_i \sim \pi_2$.
        Input g_i to primitives and integrate primitives to generate inputs for the agent a_i.
        Render a_i to the agent and record the next state s_{t+1} and the reward r_t.
        if T_i is fully tracked
            End the episode.
            Add r_{arr} to total reward.
            if T_i is fully tracked 25 times
                Generate a new random path T_{i+1}.
            end if
        end if
        end for
    Store the transition in a set of trajectories $\mathcal{B}$.
    end for
Update the policy parameters θ_2 according to TRPO algorithm (from Sec. 2.1).
end for

For navigation learning, different paths are generated during the training process (like shown by solid dots in Fig. 4.5 and 4.6). These paths are generated using discrete sinusoids

with random amplitudes A and wavelengths λ (4.5.1). In each path, there are discrete path points or targets. These points serve as target points for the agent. The agent is supposed to traverse the entire path by following these points. One episode is 6250 timesteps long (1 timestep = 0.01s). The environment is reset to the starting position if the agent completes the entire path or if the episode length expires, whichever happens earlier. After the agent completes one path 20 times, a new path is generated from the next episode and training continues like this for a total of 8000 episodes. This is also explained in Algorithm 4.2.

$$\boldsymbol{x} = 0{:}\,L{:}\,\Delta L \tag{4.5.1}$$

$$\boldsymbol{y} = A\,sin(\lambda \boldsymbol{x}) \tag{4.5.2}$$

$$A \sim \sigma_A(-1,1)\ \text{m} \tag{4.5.3}$$

The state space for this task is different from the previous one. It contains the rotation angle of the head about z-axis, distance of the robot from the next target, and heading of the next target with respect to the center of the head. The action space contains the goals of the motion primitives. The motion primitives are finally integrated to obtain position commands for the actuators.

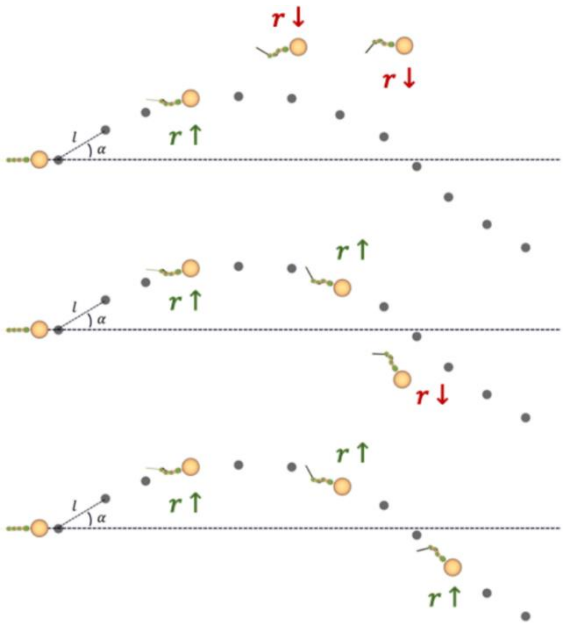


Fig. 4.5: A visualization of training and learning process. Each path contains a set of points. Range and bearing relative to next target point is considered.

The state space for the navigation part and an illustration of a training process is given in Fig. 4.5 whereby a path consisting of path points is generated and the robot follows the path points to finish the whole path. Range and bearing to the next target are used as state space.

Reward function for this part is,

$$r = r_{dist} + r_{arr} - c_t \tag{4.6}$$

where $$r_{dist} = c_{dist}(l_{t-1} - l_t)$$

and $$r_{arr} = 10 \text{ if } l_t < l_{thresh}$$

and $$c_t = 0.01$$

r_{dist} encourages the robot to go towards the next target point in the set of path points. This reward is dense and is calculated and accumulated at every timestep over the course of entire episode. l_{t-1} and l_t are the distances between the position of robot and the target position at previous and current time instants. When l_t becomes less than l_{thresh}, the target is set to the next point in the set of path points and the agent gets an arrival reward r_{arr} of 10. This reward is sparse which is added to the total reward only when the agent reaches a target position. c_t is a cost which encourages the agent to finish the task in the least possible time. The reward curve is shown in Fig. 4.3. Simulation results for this case are shown in Fig. 4.6. It shows the trajectories traversed by the head of the robot while following 7 different paths. The trajectories are shown by solid lines whereas the path points (for the corresponding trajectories) by solid dots of same color. Trajectories T_2 and T_3 are traversed when the robot follows paths generated by the same function that has been used during training. The agent is also tested with paths of shapes other than those generated during training. For instance, trajectories T_4, T_5, T_6, and T_7 are traversed when the robot follows some other paths. Only seven trajectories are shown to keep the figure uncluttered.

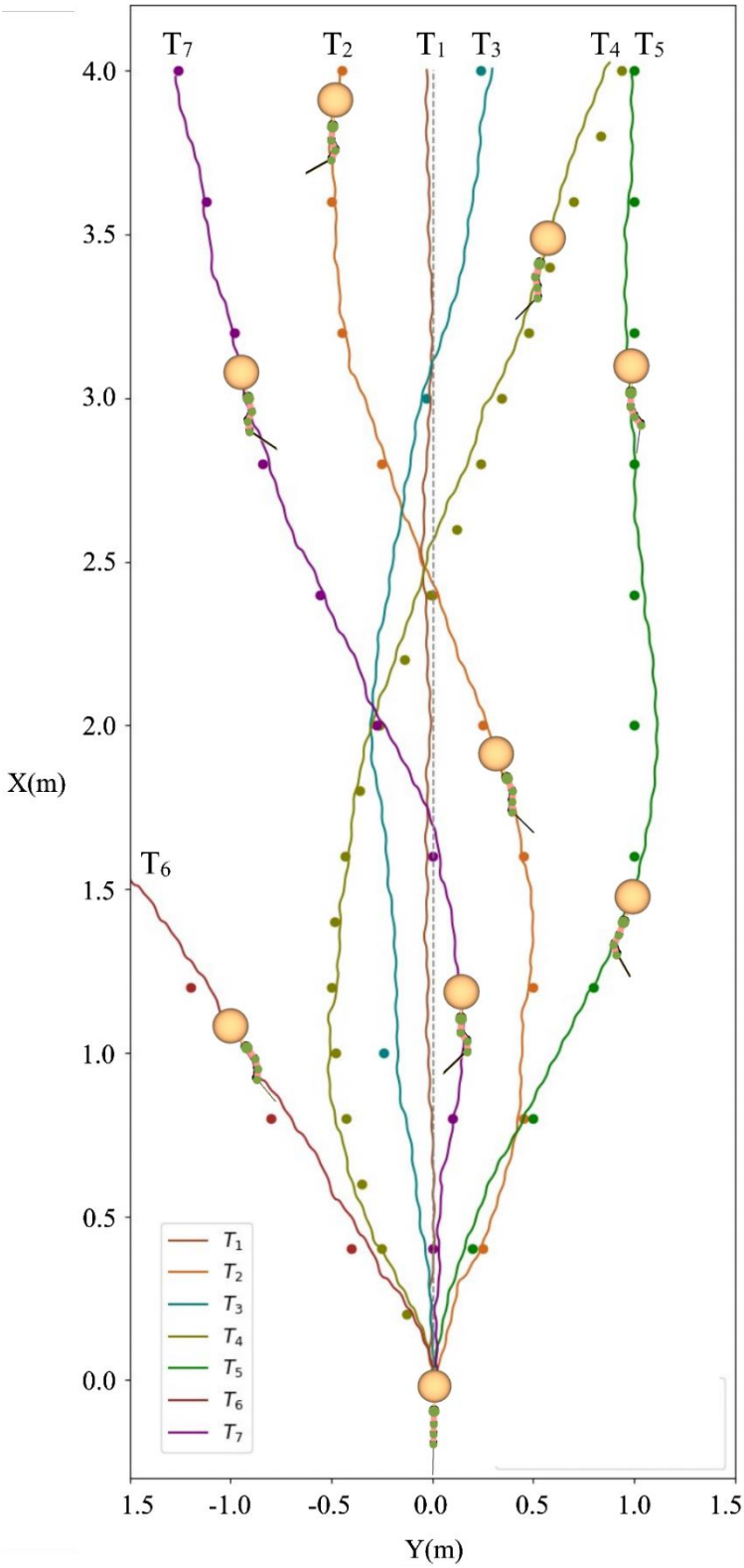


Fig. 4.6: Simulation results of combined architecture (Fig. 4.1) – Trajectories traversed (indicated by solid lines) by the head of the robot while following different paths (indicated by filled circles).

The agent has been tested for many different cases including very long random paths as well. The trained system is able to navigate the agent well. In some cases, the agent passes at some distance from the path point. This is due to relatively large l_{thresh} = 0.2 set during training. Smaller value of l_{thresh} is expected to improve the target tracking accuracy.

We also test the policy when the path points are generated with different resolution in timescale. For example, in T_3, the resolution for generating path points is doubled such that the number of samples is halved whereas in T_4, the resolution for generating path points is halved such that the number of samples is doubled. Difference in number of samples can be observed along T_3 and T_4. The agent traverses well in both cases. When the number of samples are very less, the agent moves to the next target in straight line which is expected (T_3). Higher number of samples can be generated when the agent is required to precisely track some path (T_4).

Then, we demonstrate some benefits of our method by trying to modulate the motion behavior on the already trained setup. One limitation usually encountered while using reinforcement learning policies alone in robots is lack of an explicit method for frequency

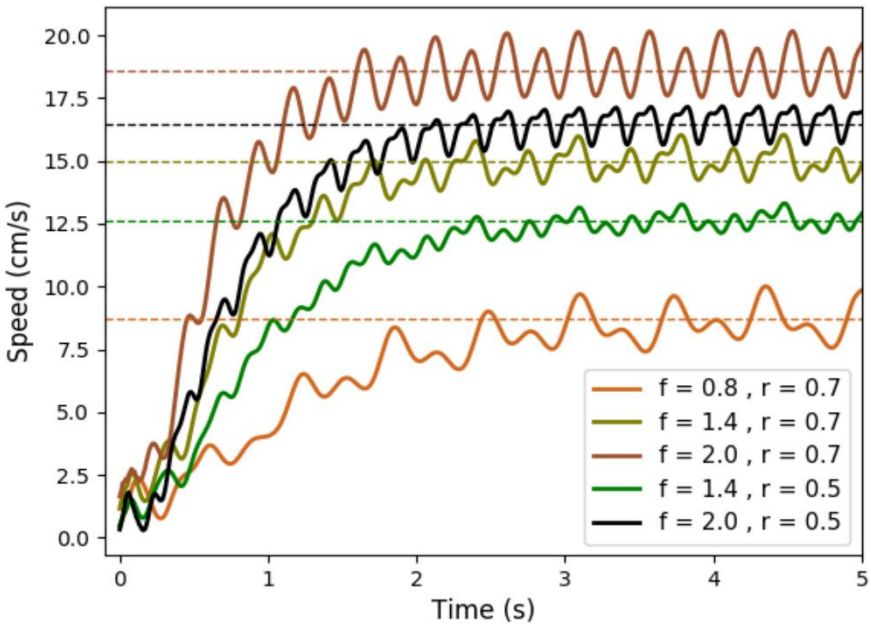


Fig. 4.7: Speed of the robot at different tail undulation frequencies – f (hz) and amplitudes – r (rad). Dashed lines show steady state average speed.

adjustment. In most cases, the network is tested at one frequency after training. Changing the frequency of the entire control loop (i.e., reading states, querying network, and writing actuators) is not an efficient way. Similarly, encoding this functionality into the reward function is also complex [91]. In the proposed framework, DMPs are used at the end of DRL policies (Fig. 4.1). They provide a feasible way to modulate the output motion behavior [135]. Properties like frequency and amplitude can be adjusted by varying the corresponding parameters and these properties are especially important for biomimetic robots. The frequency by which the motion primitives are generated (4.1) can be altered to change the frequency of the undulation pattern. Simulation is run using three different values of frequency.

The same policies, without retraining, effectively navigate the agent on altered frequencies. Using higher or lower frequency allows to control the speed of the agent. Speed of the agent traversing on a similar trajectory with three different undulatory frequencies (0.8, 1.4, and 2.0 hz) is shown in Fig. 4.7 with steady state average speed shown with dashed line. The oscillatory nature of speed is a natural phenomenon for such robots [91]. It should be noted that the control loop frequency is not altered. Only the frequency by which primitives are generated is changed. Similarly, the amplitude can be varied using the factor r_i (4.3). The robot is run at two different amplitudes (0.5 and 0.7 rad) (Fig. 4.7) to see the effect of change in amplitude on speed.

Table 4.1 – Values of Constants in Sec. 4

γ	0.99	r_j	1*
δ	0.001	c_j	$\pi/4$
a_2	$a_1/4$	c_f	1
τ	0.05*	c_r	0.1
h_j	1	c_{dist}	100

* Held constant during training. Can be changed as required during testing.

4.5 Hardware Experiments

Some details of the physical robot have been mentioned in the system description (Sec. 4.3). Here we discuss experimental setup (Fig. 4.8) and results (Fig. 4.9). To verify the simulation results, small-scale testing is done on the real robot. Hardware experiments are done in a limited laboratory environment. To obtain states on the actual robot, the robot is tracked

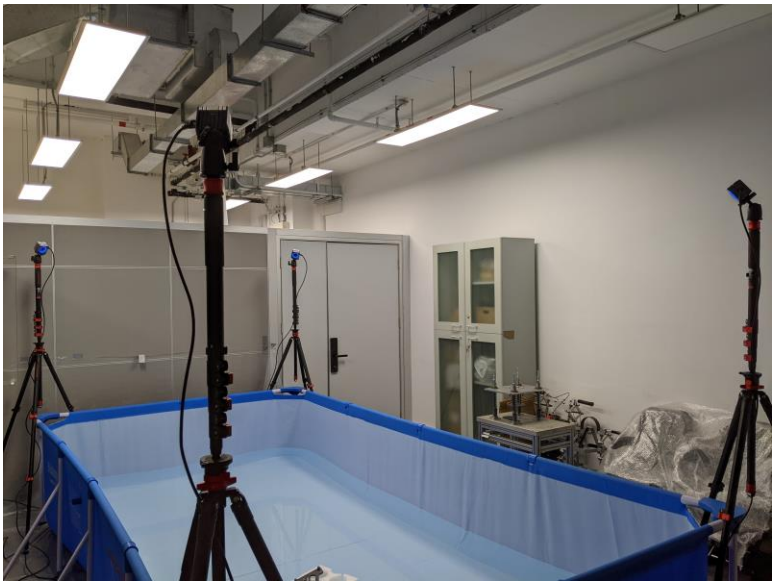


Fig. 4.8: Tracking setup to track robot's position.

using small spherical markers placed on the head of the robot as shown in Fig. 4.2 (left). The angular position of the head, distance, and heading of the path point w.r.t the head is calculated with the help of 4 tracking cameras installed on the sides of the pool.

First, the thrust learning part (Sec. 4.4.1) is verified, and the trained thrust-policy π_1 is tried on the hardware. The result is shown in Fig. 4.9 (left). Six snapshots of the robot are shown. The robot can be observed swimming in a straight line just like the trajectory T_1 traversed by the simulated agent in Fig. 4.6.

Then, the navigation learning part (Sec. 4.4.2) is tested to verify the navigation results obtained in the simulation. Only some trajectories are tried due to limited dimensions of the pool (4m x 2m). But the testing seems sufficient to demonstrate the effectiveness of the method

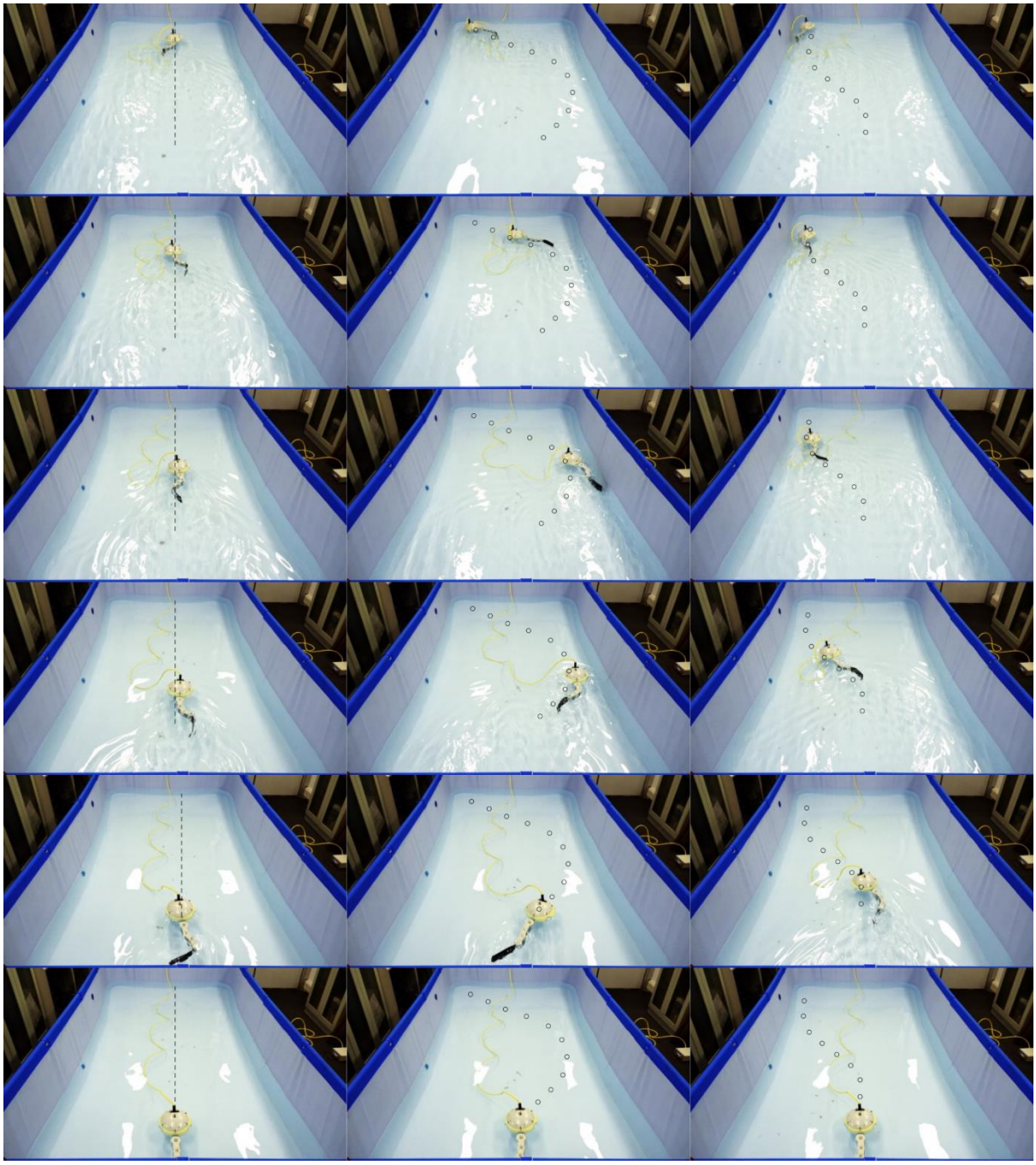


Fig. 4.9: Hardware Experiments. Left: Result for Thrust Learning (Sec. 4.4.1) (Straight line). Mid and Right: Results for Navigation Learning (Sec. 4.4.2) (Target tracking along two paths. The expected paths are overlayed on the snapshots for reference).

and the results show that the robot can perform well in large scale as well. The states, obtained in real time, are used to query the goals of the DMPs from the navigation policy π_2. DMPs are then integrated to get output position commands. The servo motors in the joints of the robot track their respective position commands using low-level PD controllers. The actions are

clipped in [-60 60] degrees range to avoid any damage to the robot. Fig. 4.9 (center and right) shows snapshots of the robot while it tracks two different paths. Hardware experiments conform with the results obtained in simulation.

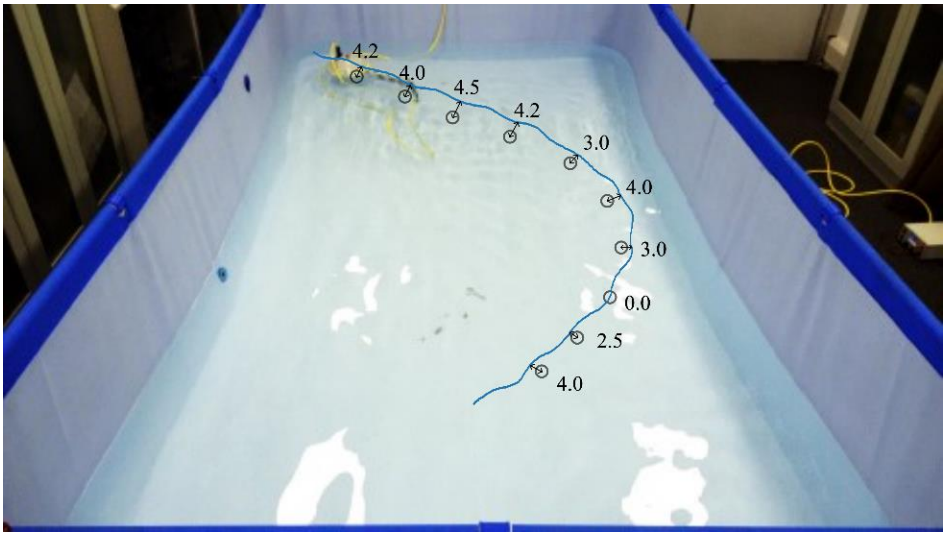


Fig. 4.10: Trajectory of the robot plotted via object tracking done on the video.

The trajectory of the robot was not collected in the experiments shown in Fig. 4.9. However, using some object tracking algorithm in *OpenCV*, the position of the robot has been tracked and the corresponding results can be seen in Fig. 4.10. The deviation is found to be in range of 9 cm ($\epsilon \sim [-4.0,\ 4.5]$ cm).

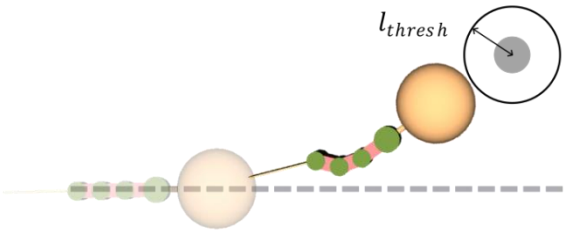


Fig. 4.11: Illustration of threshold distance.

It should also be noted that during training, as mentioned in the navigation learning section (Sec. 4.4.2), a threshold distance around the vicinity of target is used as shown in Fig 4.11. So, an error in the range of 10 cm is expected.

4.6 Discussion and Future Work

Unlike most works using DMPs [88, 89, 131-133], the method (Sec. 4.2.2) does not require any predefined baseline behavior to train motion primitives. Instead, simple reward functions are used to induce required behavior using reinforcement learning. Hence, it can cover a broad range of motion behaviors. This will allow this method to be used for other robots as well, such as quadrupeds.

$$r_1 = r_{dist} + r_{arr} - c_t \tag{4.7}$$

where $r_{dist} = c_{dist}(l_{t-1} - l_t)$

$$r_2 = c_{\theta_z} + r_{arr} - c_t \tag{4.8}$$

where $c_{\theta_z} = -|\theta_z|$

The reward function used to complete the navigation task attracts the agent towards the next target r_1 (4.7). There can be another reward function that will penalize the agent on maintain large deviation from the target r_2 (4.8).

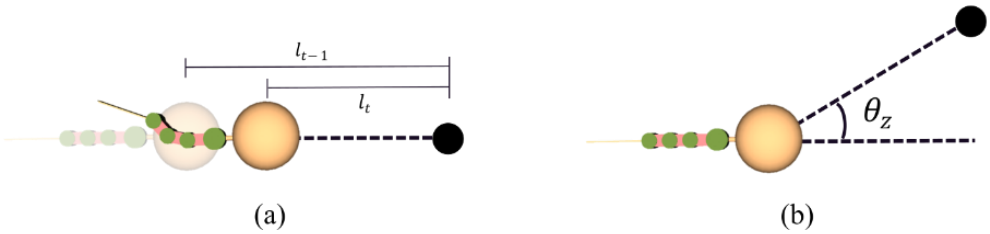


Fig. 4.12: (a) Illustration of original reward function (4.7). (b) Illustration of new reward function (4.8).

Upon using the reward formulation in r_2 (4.8), the cumulative reward does not compete the one with r_1 (4.7). Hence, r_1 is opted finally.

Sensors to observe the environment (e.g., cameras) are not considered at this point but using DRL will allow us to incorporate different sensing modalities directly without any significant change in architecture. Hence, using DRL with DMPs adds the potential to optimize without demonstrations as well as using high-level feedback.

In this work, swimming on the surface of water is considered. As a future work, we can extend the system architecture and the simulation framework to accommodate the rotational degree of freedom in the head. This way the robot will be able to dive underwater in addition to swimming on the surface. More dexterous motion primitives can then be trained for underwater navigation. Another direction will be to accommodate sensors like sonars and underwater cameras and try to manipulate motion primitives according to surrounding observations.

The proposed schematics may encounter some problems in real world domain. For example, large disturbances are expected in river or ocean environment. It is expected that the system will respond well to those disturbances since the overall system is decoupled into a high-level controller and a low-level rhythm generator.

If the system will face issues in above mentioned scenarios, there can be two possible solutions that can be tried. First, the weights of the networks can be tuned on the hardware. This solution is mostly opted in literature. Another solution can be domain randomization. In this technique, the parameters of the robot, the state and the action spaces are all randomized to a degree and the robot can still learn states to actions relation.

CHAPTER 5

UNDERWATER LOCOMOTION AND NAVIGATION (EXTENSION TO 2–DOF BCFBOT)

The platforms that have been used so far have four (Sec. 4) and five (Sec. 3) series single degree of freedom joints with axes of rotation parallel to each other. All the links of the tail move in one plane and hence, the robot performs planar locomotion on the surface of water. The buoyancy of the platform is adjusted such that the robot stays on the surface. Therefore, gait optimization in Sec. 3 and navigation in Sec. 4 were performed on the surface of water.

This raises questions about the applicability of methods proposed in this work to the platforms that may be able perform underwater locomotion. So, in this chapter, both the hardware and the methodology are extended to the scope of underwater locomotion.

5.1 Related Work

This section deals with related efforts found in literature about underwater locomotion. This section is limited to efforts in biomimetic domain and includes designs inspired from natural organisms in marine domain. It does not include efforts that use thrusters or rotary propellers for depth control. In bio-inspired domain, for depth control in underwater

locomotion, pectoral fins are mostly used. Pectoral fins are introduced in Sec. 1.3 in the MPF category. Although in this study caudal fin is considered for depth control but a brief review of related work on key studies on pectoral fins is also included for reference. After that, some literature on designs able to perform underwater locomotion (including depth control) without pectoral fins is reviewed.

a) With Pectoral fins:

In [136], a flexible robotic pectoral fin inspired from *Gomphosus varius* (bird wrasse) is designed. The design is studied and optimized by performing fluid dynamics studies. The physical fin has a bulky and exposed design which makes it difficult to incorporate it into a mobile robot. It is tested for its performance in a thrust measurement tank. The design in [67] is very compact. It incorporates a caudal fin and two pectoral fins on each side but the fins are rigid unlike [136]. 3D geometric modelling is performed to obtain equations of motion. Lagrange-Poincare model is used to model dynamic effects. Depth control is performed using pectoral fins and a close match is found between simulation and hardware experiments. A *labriform* mimicking fish is developed with two multi-DoF rigid pectoral fins [137]. Multiple motions such as in-plane swimming, turning, and hovering are developed owing to multi-DoF design. In [138], a control mechanism is studied to optimize and actively control the curvature of a flexible pectoral fin type mechanism. A model for rowing motion of pectoral fins is established in [93] whereby in-plane motion and turning is simulated and verified on the hardware.

In [139], pectoral fins are used to control depth of a fish-like robot. A similar design is opted in [32] to study leaping of a robotic fish using caudal fin. Before leaping, depth is adjusted using a set of rigid pectoral fins. A *rajiform* mimicking robot with wide undulating flexible pectoral fins is shared in [140] with three actuators per fin.

Similarly, a robotic manta-ray is presented in [141] with wide pectoral fins that can effectively perform spatial locomotion underwater by using a single actuator per fin.

b) Without Pectoral Fins:

In this part literature related to depth control without pectoral fins is reviewed. Also, some designs related to multi-degree terrestrial locomotion (e.g., terrestrial snake-like robots) which inspired later underwater robots are reviewed.

Snake like terrestrial and underwater robots from *Tokyo Institute of Technology, Japan* have been leading designs in this category. An early implementation, (Active Chord Mechanism) ACMR2 [142], presents a snake-like robot capable of 3D locomotion in terrestrial domain. It can perform three different types of motion patterns including traditional snake-like spiral motion, twist motion, and loop motion. ACMR3 [143] has a better design with large passive wheels and better mobility. It can perform additional gaits compared to its predecessor. ACMR4 [144] presents an improved dust-proof and water-proof design but underwater testing is still not done. ACMR5 [145] shares a unique bevel gear mechanism to perform pitch and yaw motion which has been used later by many other designs as well (e.g., [146, 147]). This mechanism allows easy and reliable waterproofing. Some versions of this robot have therefore been tested underwater as well.

Snake robot [148] from Robotics Institute at *Carnegie Mellon University* has modular design with per joint current and temperature sensing. An improved version [149] features custom electronics, improved sensing and memory alloy brakes. It can utilize different skins to adapt to different terrains. A later iteration, [150], features elastic actuators with options of torque, position and velocity control. But all three are terrestrial robots and cannot be used underwater. They are reviewed here since the design in this work is inspired from ideas from above studies.

Design shared in [151] also contain series of 2-DoF joints with tactile sensing and [152] also uses similar design. Some other particularly underwater designs are shared below. Series 1-DoF eel-like completely waterproof platforms are shared in [153] to investigate the effect of caudal fin geometry, in [154] to study the effect of position of dorsal fins and in [155] to examine the effects of flexible caudal fins on swimming performance. Similarly, robotic platforms in [9] and [156] used to study salamander-like gaits and undulatory swimming also have series single DoF designs.

5.2 Extension in design and setup

In this part, the modifications made to the design of the robot and to the thrust measurement setup previously made for the old design, are shared.

5.2.1 Design

On biomimetic platforms, for underwater locomotion (or depth control), usually pectoral fins are installed in addition to caudal fins [139]. In this way, propulsion for forward locomotion is achieved using the caudal fin in the end of the body and propulsion for depth control is achieved using the pectoral fins [32] in the frontal part of the robot on both sides.

In this study, pectoral fins are not considered, and the same single caudal fin is used for depth control. 1-DoF BCFbot is modified and extended to 2-DoF BCFbot. The single DoF yaw joints are alternately added with single DoF pitch joints. This time the axes of rotation of alternate joints are kept perpendicular to each other. In this way, the fin has 2-DoFs and can perform both yaw and pitch motions. The modified hardware is shown in Fig. 5.1. With both yaw and pitch joints, the reachability of such platform expands a lot [147] (Fig. 5.2).

By 2-DoF, it does not mean that the robot has only two actuated joints, but it means that it has two different joint types (pitch and yaw) unlike the original 1-DoF BCFbot which only has one type (yaw) of 5 joints.

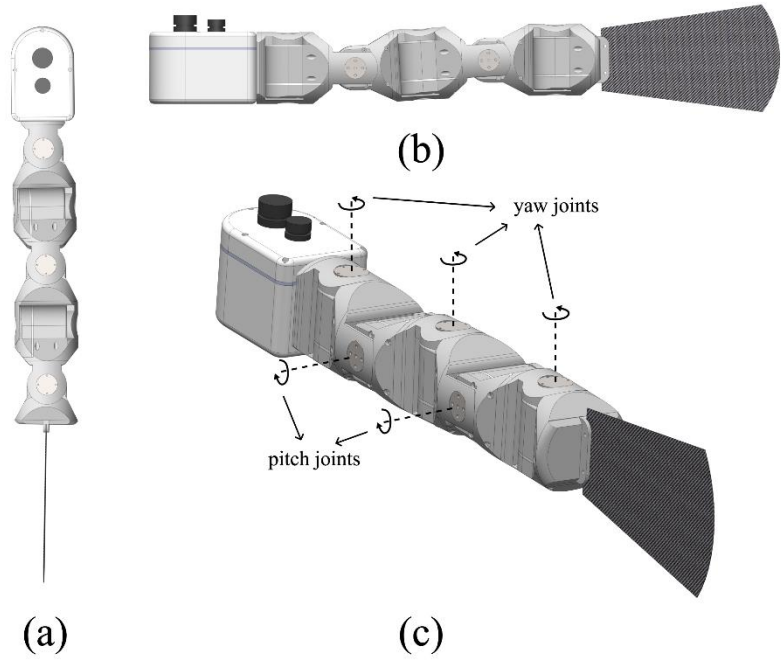


Fig. 5.1: Top view (a), side view (b), and perspective view (c) of CAD model of 2-DoF BCFbot.

Fig. 5.2 shows reachable motion space of 2-DoF BCFbot. It shows motion spaces when only the yaw joints are actuated (Fig. 5.2a) and when only the pitch joints are actuated (Fig. 5.2b) and when both the yaw and pitch joints are actuated together (Fig. 5.2c). The reachability of yaw joints tail is planar (Fig. 5.2a) and covers a circular sector of almost 150 degrees. Similarly, the reachability of pitch joints is also planar but in vertical plane (Fig. 5.2b). By combining motion space of both types of joints, the reachability of tail expands from planar to spatial domain (Fig. 5.2c). The reachability space is now a spherical section instead of planar circular sector. The view in Fig. 5.2c seems planar and therefore another view (Fig. 5.2d) is added which shows that the space is like a portion on the surface of a sphere.

The overall thrust vector space for the old configuration (yaw joints only) should form a circular sector. With the changes made, since the motion space is conical, the overall thrust space should also make a conical region. This will effectively enable spatial locomotion in addition to planar. By appropriately adjusting the overall thrust vector in the desired direction,

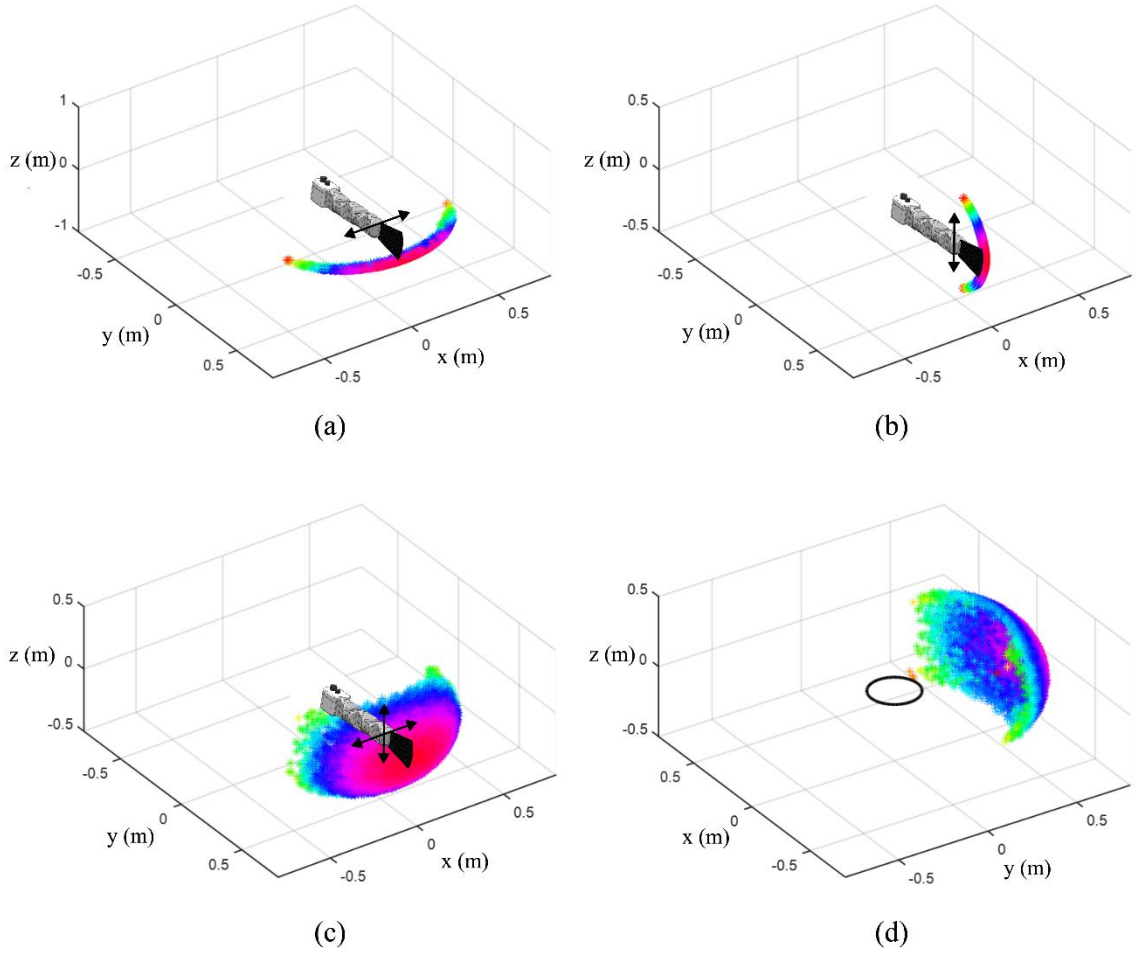


Fig. 5.2: (a) Reachable motion space of yaw-joints-only of 2-DoF BCFbot (circular section or arc in horizontal plane). (b) Reachable motion space of pitch-joints-only of 2-DoF BCFbot (circular section or arc in vertical plane). (c) Combined motion space of both yaw and pitch joints (spherical sector in x-y-z plane). (d) Combined motion space of both yaw and pitch joints from a different view.

depth control can be achieved. Later it will be shown by performing experiments with the updated hardware.

5.2.2 Setup

The thrust measurement setup described in Sec. 3.2.2 can measure F_x and F_y. The rod attached to the head can rotate around x and y axes passing through the center of the rod by means of two bearings holding the rod. These bearings are attached with the static structure using support brackets to avoid any translational movement in any direction.

To measure F_z, the translational motion of rod in z direction is released. The rod can now rotate about x and y axes as before but the lock on translation in z direction is released. The other end of the rod is attached to the sensing surface. The sensing surface can now receive impact from the rod in z direction (since the translation lock has been released) in addition to x and y directions. In this way all three components (F_x, F_y, and F_z) can be measured. The modified setup is shown in Fig. 5.3. When observed along with measurement setup figure in Sec. 3.2.2, it is easy to observe the changes made to the setup.

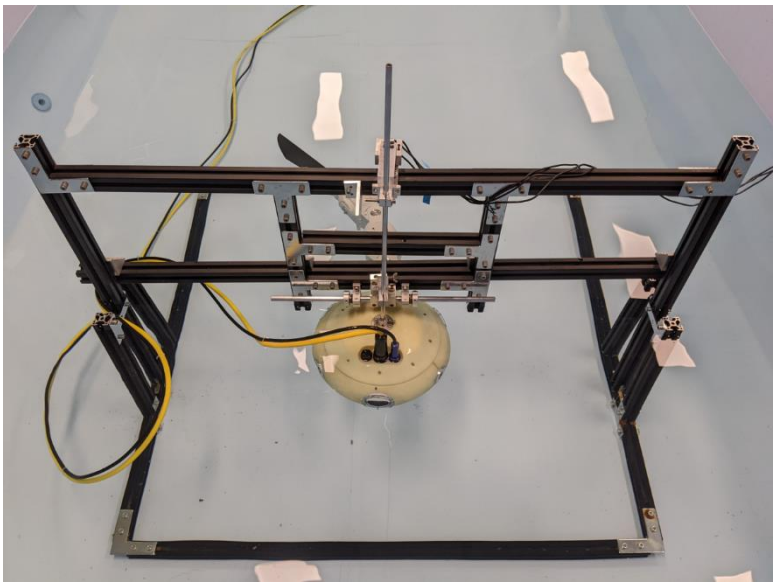


Fig. 5.3: Modified thrust measurement setup which can measure all three components (F_x, F_y, and F_z) of force.

Also, since depth variation will be involved in this part, it is necessary to record the robot's motion from underwater, not only from the top of the pool as in previous sections. Therefore, a water-proof camera (*GoPro Hero-10 [157]*) is placed inside the pool to record robot's motion. Pictures in the experimentation section (Sec. 5.5) are from the underwater camera.

5.3 Extension in framework

In the previous part, the extension in design and setup was discussed. In this part, the control schematic originally presented in Sec. 3 is extended to accommodate the modifications done to the platform to perform underwater locomotion.

The architecture in Sec. 3 contains a DRL network (represented in blue color in Fig. 3.1) and an oscillator network (represented in green color in Fig. 3.1). For this part, another oscillator network is added. The resultant scheme is portrayed in Fig. 5.4.

The network in green color will generate inputs for joints y_1, y_2, and y_3 whereas the second network in red will generate signals for joints p_1 and p_2.

Oscillator networks for yaw and pitch joints can be states as,

$$ {}_k\dot{\phi}_i^t = 2\pi f + {}_k\zeta_i^t \tag{5.1a}$$

$$ {}_k\zeta_i^t = \sum_j {}_k u_j^{t-1} {}_k c_{ij} \sin\left({}_k\phi_j^{t-1} - {}_k\dot{\phi}_i^{t-1} - {}_k\psi_{ij}\right) \tag{5.1b}$$

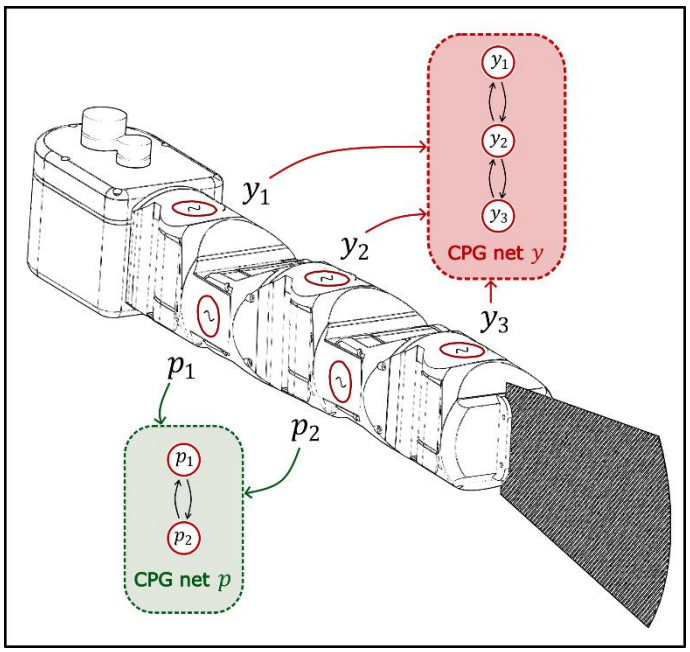


Fig. 5.4: Modified control architecture for 2-Dof BCFbot containing two CPG networks y and p for yaw and pitch joints.

$$ {}_k\ddot{u}_i^t = {}_kc_1\left[\frac{{}_kc_1}{4}({}_kU_i - {}_ku_i^{t-1}) - {}_k\dot{u}_i^{t-1}\right] \tag{5.1c} $$

$$ {}_k\dot{d}_i^t = {}_kc_2\left[\frac{{}_kc_2}{4}({}_kD_i - {}_kd_i^{t-1}) - {}_k\dot{d}_i^{t-1}\right] \tag{5.1d} $$

$$ {}_k\alpha_i^t = {}_ku_i^t \sin({}_k\phi_i^t) + {}_kd_i^t \tag{5.1e} $$

For the above system, the variables have similar definitions as in system (2.31a-e) in Sec. 2.2.1. The above depiction is verbose and is explained below.

Pre-subscript k defines the joint type which can either be yaw-y or pitch-p type (as described in hardware extension and in above Fig. 5.4). Expanding the above system for separate joint type will separate out networks. Post-subscript i has similar definition as before – individual oscillator number within the pattern generator. Post-superscript t is the time evolution variable.

CPG network y is used in limit cycle behavior such that it will keep on generating periodic patterns and at the same time it can receive high level commands about deviation. This way is similar to the one used in Sec. 4 where primitives with limit-cycle behavior are used. Hence, for this case the above system will be expanded for,

$$ i = 1, 2, \text{ and } 3 \quad \text{for} \quad k = y \tag{5.2} $$

CPG network p is used in point attractor behavior such that it will converge on biases from high-level commands. This way is similar to point attractor behavior of primitives used in many works where a non-rhythmic activity is performed [89] . For this case the above system will be expanded for,

$$ i = 1 \text{ and } 2 \quad \text{for} \quad k = p \tag{5.3} $$

In our case, if the robot is made to change depth while swimming from one level to another level. The overall job can be split into two sub tasks. One is swimming which is a

continuous and rhythmic activity. The second sub task is depth change from one level of depth to another. This is a non-rhythmic activity in which a bias will be generated in pitch direction causing the robot to change depth. Upon reaching the desired depth, the bias will terminate. This sub task is aperiodic as compared to the previous one. For the first case, i.e., continuous swimming, CPG network y is used in limit-cycle format. For the latter case, i.e., aperiodic deviation, CPG network p is used in point attractor format.

The rest of the behaviors follow from the above description. For example, to perform planar swimming at a certain depth only yaw joints are used for which yaw network is employed. To perform swimming, turning, and changing depth at the same time (or, in other words, performing spatial locomotion in three dimensions), it will require combination of periodic and aperiodic activity along yaw joints and only aperiodic activity along pitch joints. Hence, a combined signal will be generated accordingly.

5.4 Teleoperation

To perform experiments in this part, teleoperation is used. The general schematic of teleoperation is described in Fig. 5.5. The architecture is divided into two parts – a remote setup and the BCFbot itself. The remote setup contains a power supply (EA Elektro-Automatik Bench Power Supply, 320W [158]). As mentioned before (Sec. 4.3), sometimes a battery is put inside the head of the robot to power the actuators. So, this component is optional and is used during long duration testing. Other than power supply, a computer is used which is connected to the robot using a network. Ethernet (wired LAN) is used in this section. But a wireless network can also be used when feasible. A joystick (handheld controller) is attached to the computer. It is used by the operator to control the robot.

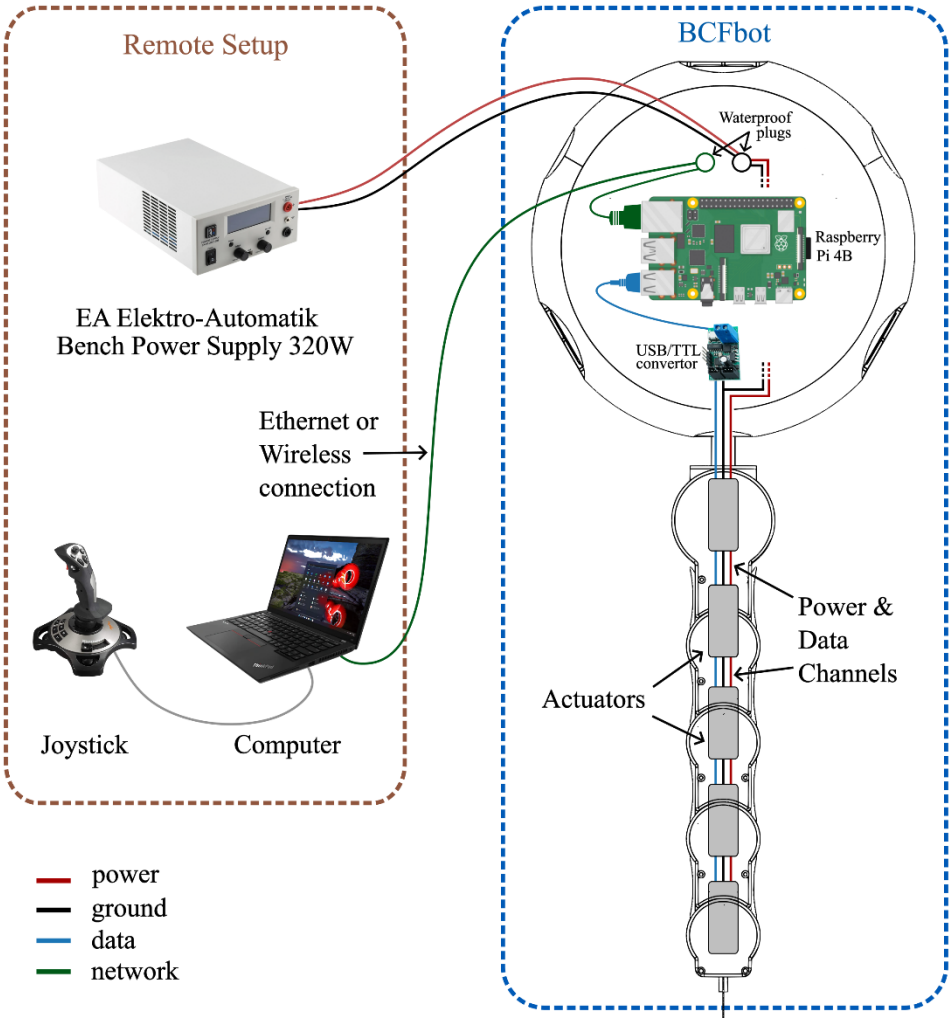


Fig. 5.5: Schematics for teleoperation. Remote setup contains power supply which is optional in case batteries are used inside the BCFbot. Joystick is used by the operator for teleoperation. A computer reads joystick commands and sends high-level commands to RPi through a network connection. RPi then translates those commands into signals for actuators. The computer and RPi are connected using Robot Operating System (ROS).

Horizontal and vertical deviation commands are given by the operator while observing the robot swimming in the pool at the same time. The commands are read by a computer program. Right after the commands are received, they are mapped to desired angles from the output scale of joystick. Afterwards, they are transferred to the microcontroller inside the robot's head (Raspberry Pi 4B – RPi) via network. Both the machines (computer in the remote setup and RPi in BCFbot) are connected using Robot Operating System (ROS).

To perform teleoperation efficiently, the RPi runs multiple tasks in parallel. The first task is to receive commands from the remote computer. The second task is to integrate oscillators networks (5.1), according to the updated deviations received from the computer. The oscillators should be integrated in real time. To avoid any interruption, both routines are handled in separate threads and parallel processing is used. Since a small interruption may cause undesired delay in the integration loop thereby causing the output motion pattern to distort. So, one thread's job is just to integrate oscillator network. The output of integration (5.1e) are angular position commands for the joints. Once the desired positions are calculated, they are sent to motors via a USB/TTL converter. This communication is done using a single channel. The transmission packet contains the motor's ID followed by desired control commands. After each integration loop, the program outputs transmission packets. All actuators are connected on the USB/TTL line serially. The motor with corresponding ID accepts the packet and executes command accordingly. The transmission packets are stored in a buffer. Old packets are replaced by the latest packets. In this way, the integration loop does not have to wait for the respective commands to be executed on the actuators. The time period for the integration loop is set to be exactly 10ms.

For such applications where the performance of the system is very sensitive to the output motion pattern and its absolute and relative phase, precise looping is crucial. Using parallel processing, it is ensured that commands are generated, communicated, and executed at the exact same time instants and no delay is introduced.

5.5 Experiments

This section is arranged in the same way as the experiments in the swimming gaits generation (Sec. 3) section. In Sec. 3, the planar case has been discussed. In this part, extension to third dimension will be discussed.

As explained above (in Sec. 5.2.1), the motion space of the modified robot is spatial (not planar). Therefore, the thrust-vector space is also spatial. To measure force components, the robot is attached with the thrust measurement setup shown in Fig. 5.3. It can measure all three components shown in Fig. 5.6.

Like the planar case, the rhythm generator generates input patterns for the robot's joints. These patterns are rendered on the corresponding joints. This generates a travelling-wave travelling from one end of the tail to the other. It generates propulsive forces opposite to the direction of travel of the wave. And when pitch deviations are added in the pitch joints while the yaw joints are undulating, these forces are developed in all three directions as shown in Fig. 5.6. This aspect will be verified in the experiments in the next section.

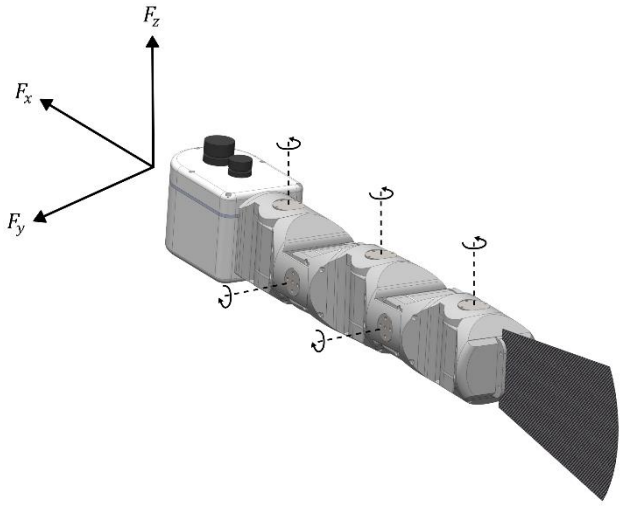


Fig. 5.6: Depiction of directions of force components – F_x, F_y, *and* F_z.

5.5.1 Visualizing Results

Results are visualized in this part. First, results from the thrust measurement setup are shared and then some free-swimming trials are shared.

a) Biasing downwards:

The first case is when the tail biases downwards. For this case, the robot tied to the thrust setup close to the surface of water. To quantify biasing downward and upward cases, z-component of the force - F_z is measured. In the start, the tail undulates in the horizontal plane and the measurement is started. This state is captured in the snapshot in Fig. 5.7a. Force generated during this state is shown in the 0-3s of the graph plotted in Fig. 5.7c which captures F_z during this experiment. F_z oscillates around zero since there is no resultant component in z-direction when the tail undulates purely in the horizontal plane. As the tail switches its state from being in horizontal plane to biasing downwards, the force in z-direction increases to a non -

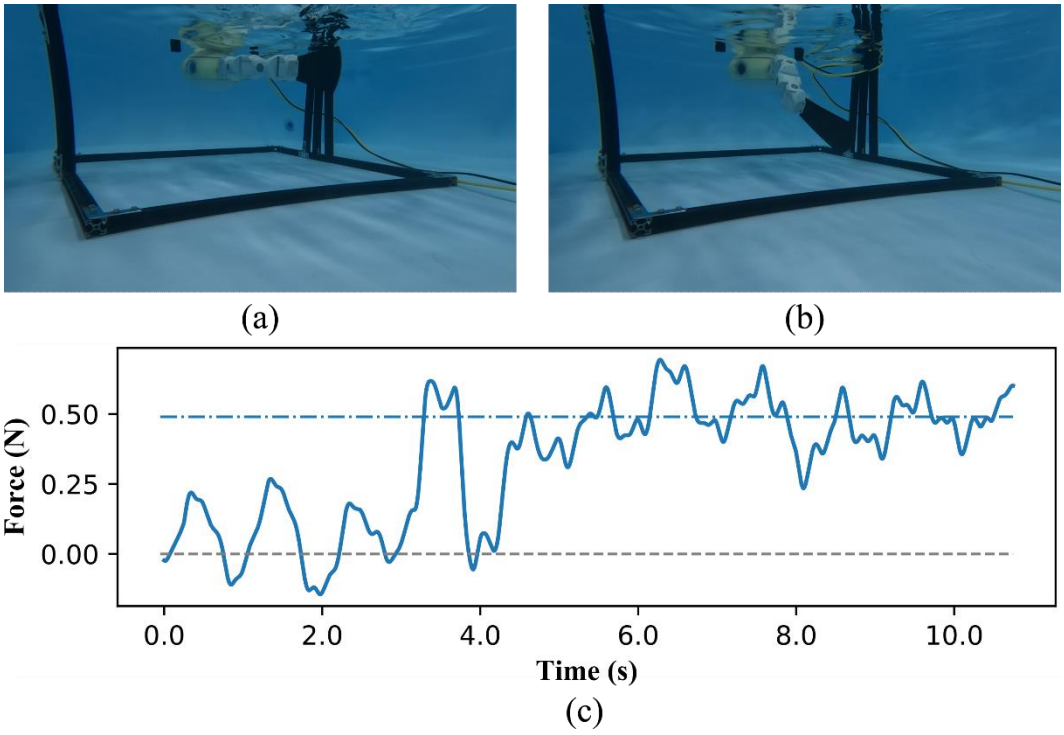


Fig. 5.7: (a) Posture of robot before downward biasing. (b) Posture of tail during downward biasing. (c) z component of the force-F_z before and after downward biasing.

zero average as shown in Fig. 5.7c. After the juncture at 3.5s, F_z rises to about 0.5 N. The later state of the robot is shown in Fig. 5.7b.

b) Biasing upwards:

The second case is when the tail biases upwards. For this case, the robot is initially tied close to the bottom of the pool so that it can have enough room to bias upwards. Similar to the previous case, the tail first undulates in the horizontal plane. The corresponding state of the robot is shared in Fig. 5.8a. Measurement of F_z is started at this state. The tail undulates for 5s and generates zero average force in z-direction. After 5s, it shifts its state to biasing upwards. This causes the trust vector to point downwards since the z-component should have non-zero negative average in this state. It can be seen in the F_z plot (Fig. 5.8c) after 6s point. The robot with tail biasing upwards is shown in Fig. 5.8b.

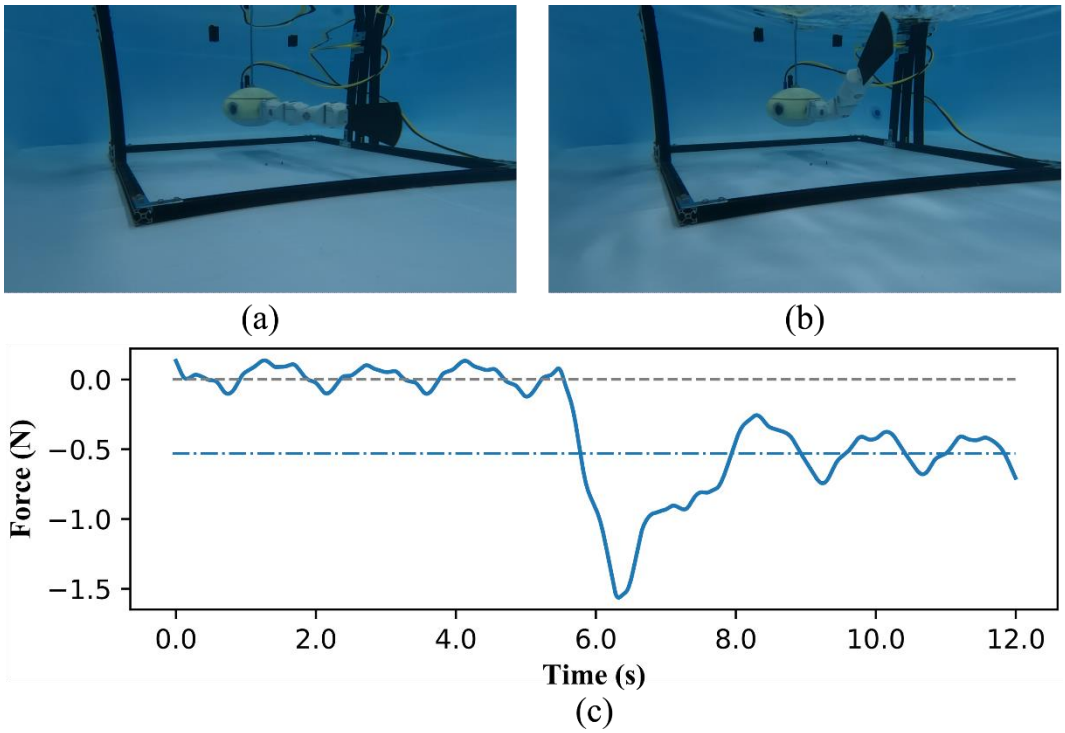


Fig. 5.8: (a) Posture of robot before upward biasing. (b) Posture of tail during upward biasing. (c) z component of the force-F_z before and after upward biasing.

c) Biasing leftwards:

To quantify robot's turning performance during underwater locomotion, y-component of the force developed by the robot is measured when the tail biases sidewards. In this part, F_y is observed when the tail biases leftwards. The direction (leftwards) is when the robot is seen in perspective view from behind (as in Fig. 5.9a and 5.9b). If viewed from the top, the tail moves in clockwise direction. Like previous experiments, the robot starts undulating in vertical plane (Fig. 5.9a), biases leftwards (Fig. 5.9b) and returns to the original state (Fig. 5.9c). The force pattern for F_y during above motion is shown in Fig. 5.9d. When the tail oscillates around centerline, F_y oscillates around zero just like the thrust measurement results shared for single DoF BCFbot in Sec. 3.5. In the time bracket of 5-13s, the tails biases leftwards due to which the F_y shifts upwards and oscillates around a non-zero average value of 1 N before returning to zero average when the tail returns to the original position.

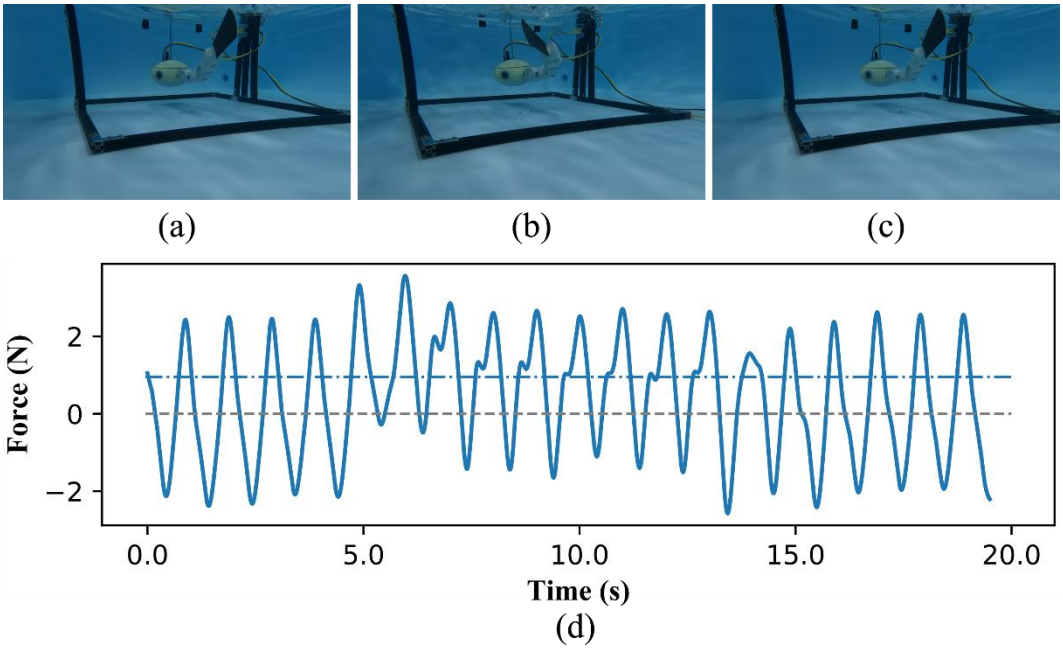


Fig. 5.9: Posture of robot before (a), during (b), and after (c) biasing leftwards. (d) y component of the force-F_y before and after biasing leftwards.

d) Biasing rightwards:

In the last case, vice versa of the previous case (c) is tried. F_y is observed when the tail steers rightwards (if viewed from back as in Fig. 5.10) or counter-clockwise (if viewed from top). The tail first undulates about the centerline for 5s (Fig. 5.10a) and then biases rightwards for the next 7s (Fig. 5.10b) and returns to the previous state (Fig. 5.10c). The force pattern during this transition is shown in Fig. 5.10d. For this case, the average of F_y shifts to a negative non-zero value of -1 N during biasing (5-13s) and remains zero otherwise.

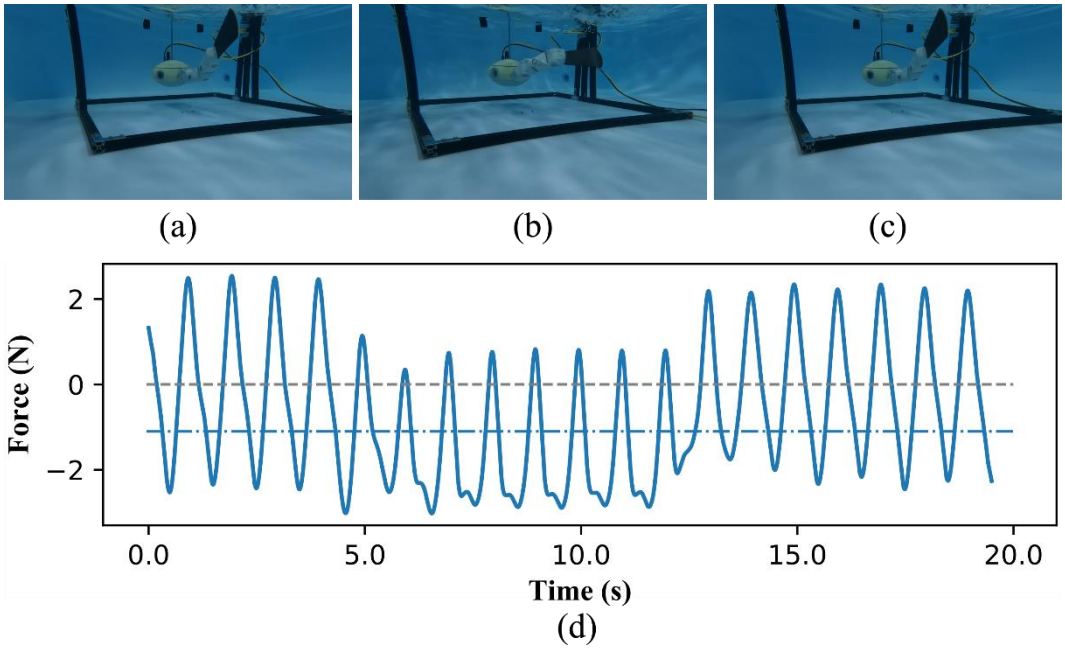


Fig. 5.10: Posture of robot before (a), during (b), and after (c) biasing rightwards. (d) y component of the force-F_y before and after biasing rightwards.

In the above experiments, performance of 2-DoF BCFbot on thrust measurement setup is observed. The two pitch joints, when biased upwards and downwards, can steer the overall thrust vector of the robot in vertical direction. This is confirmed by non-zero z-component observed in the first two cases. From this result, it can be implied that if the robot is allowed to swim freely it should be able to change its depth by using pitch joints.

Similarly, from cases (c) and (d), it can be inferred that yaw and pitch joints together can help to orient the thrust vector in a much wider region. And by appropriately controlling the two degrees of freedom, the robot can perform underwater locomotion effectively.

5.5.2 Free-Swimming Experiments

Some free-swimming trials are shown in this part. This is to verify that the modifications made in design can help the robot to undergo underwater locomotion and also to further validate the results from the thrust measurement setup in the previous section.

For thrust measurement, the robot is tied to the setup from the center of head. The tail is the only part that undulates. Therefore, fluid forces acting on the tail joints are reflected to the measured thrust output. The head is just used to hold the robot. The shape and size of the head does not impact the measured thrust output. Therefore, the same head as used with the previous setup (Sec. 4) is used.

For dynamic (free-swimming) experiments the head plays a substantial role. It faces a significant amount of drag force and hence, has a great impact on the performance of the robot. In the previous sections (Sec. 3 and Sec.4), the buoyancy of the robot is adjusted so that the robot always stays just under the surface for which the head was kept light. For underwater locomotion, the robot should be completely neutrally buoyant. To make it neutrally buoyant and to minimize the drag posed by water, a small snake-like head is used for free-swimming trials. The head used in the thrust measurement setup and the one used in free-swimming setup can be seen in Fig. 5.3 and Fig. 5.1 respectively. A small weight is attached to the head externally and small floats are attached to the joints to make the robot completely neutral or very close to neutral in buoyancy. With neutral buoyancy, when pitch joints are provided with some bias while the yaw joints are undulating, the development of force in z-direction - F_z changes the absolute orientation of the robot instantly. Hence, a bias (upward or downward) in

pitch joints, while the yaw joints are undulating, helps to adjust robot's vertical posture to facing upwards or downwards. In order to change from one level of depth to another while the robot is initially in horizontal plane, a bias in pitch joints to change robot's vertical posture only need to be momentary. The robot maintains the altered (tilted) posture (since it is neutrally

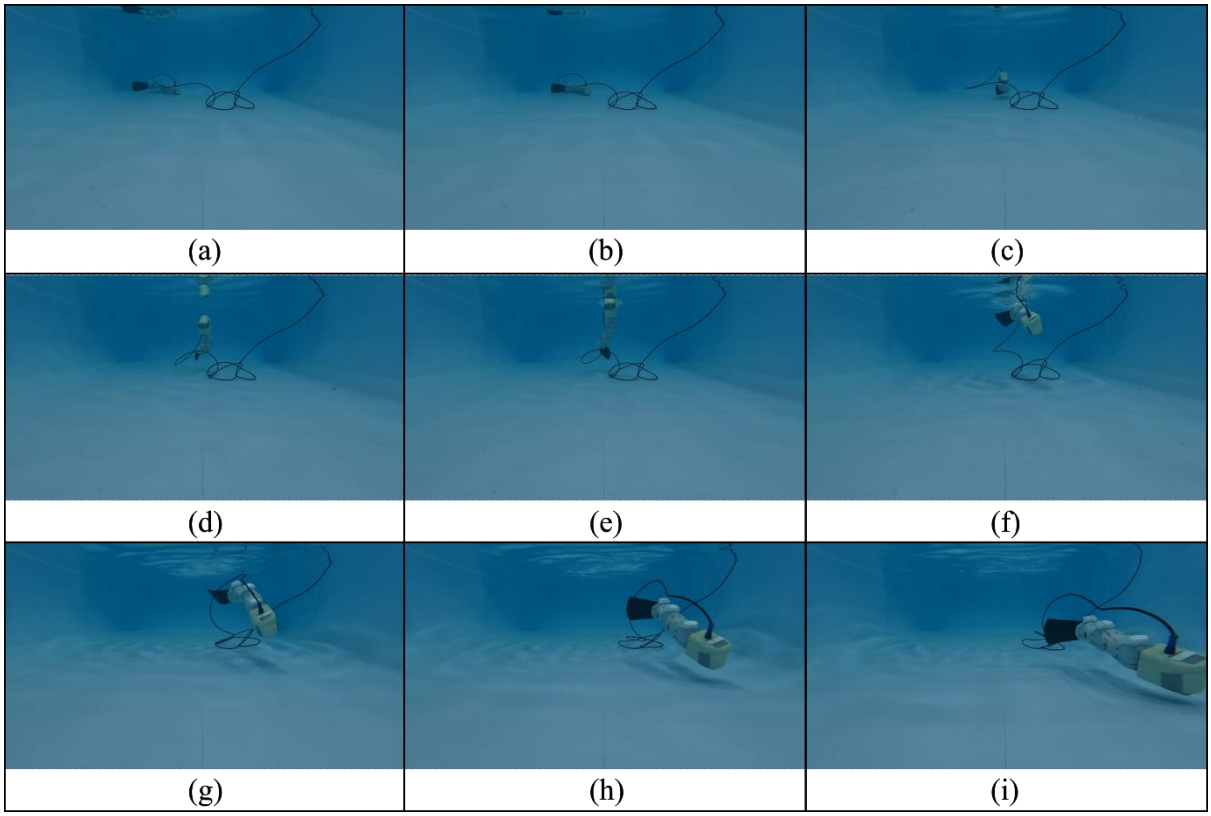


Fig. 5.11: Snapshots from free-swimming experiments of 2-DoF BCFbot. The robot starts from the far end of the pool (a) and swims to the near end (i). In between it raises its altitude from pool-bottom to around 1ft from the bottom and then returns to the bottom in snapshots (a) to (i).

buoyant) while the propulsion from undulation of yaw joints takes the robot to a different depth. Once the desired new depth level is achieved, removal of bias will bring the robot back into a horizontal plane which will be at a different depth than the plane robot started with.

The above phenomena can be observed from the snapshots of two trials (Fig. 5.11 and Fig. 5.12) of the robot swimming in a laboratory pool.

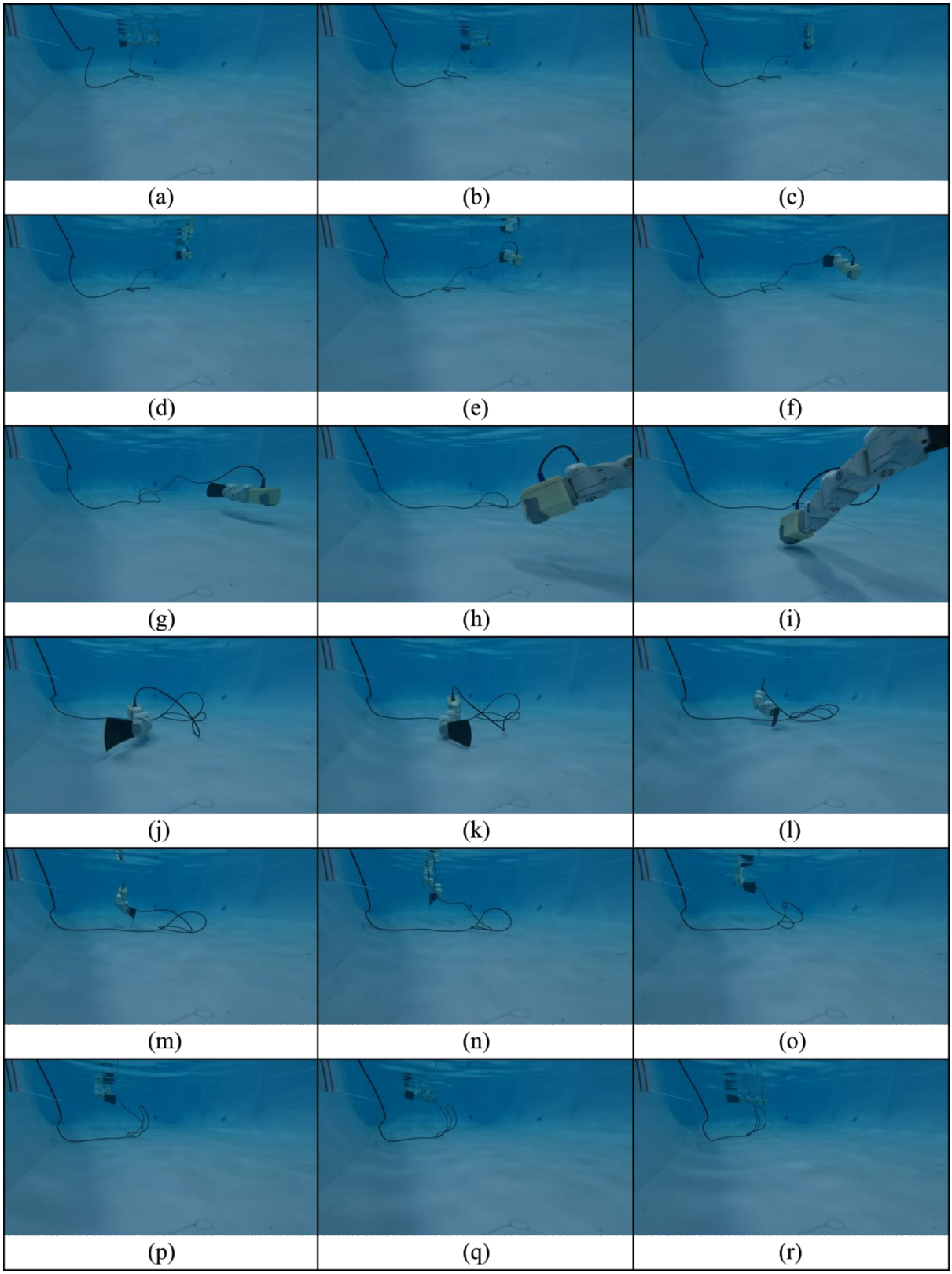


Fig. 5.12: Snapshots from free-swimming experiments of 2-DoF BCFbot. The robot starts from the far end of the pool and near the surface of water (a), swims to the near end close to the bottom of pool (j) and swims back to the far end raising its level back to the surface of water (r). (Water depth ~ 1ft.)

5.6 Underwater Navigation

After underwater locomotion using teleoperation, underwater navigation is explored in simulation. The same simulator used for on-surface navigation (Sec. 4) is utilized in this part. A model of BCFbot with yaw and pitch joints is modelled (Fig. 5.14). A number of small spheres serve as target points (Fig. 5.13). Random paths are generated which are composed of a number of target points. When the robot comes in close vicinity of one target point, the next point becomes its target, and the robot is trained to complete the entire path in this way. Next, trajectory generation and navigation training are discussed followed by navigation results

5.6.1 Trajectory Generation

To generate three dimensional trajectories, the trajectory generator of Sec. 4 is extended to have rotation. The following system (5.4) is used,

$$\boldsymbol{x} = 0{:}\, L{:}\, \Delta L \tag{5.4a}$$

$$\boldsymbol{y} = A\ sin(\lambda \boldsymbol{x}) \tag{5.4b}$$

$$A \sim \sigma_A(-1{,}1)\ \text{m} \tag{5.4c}$$

A sample trajectory created by the above generator is shown in Fig. 5.13a. The trajectory lies in the horizontal $(x - y)$ plane.

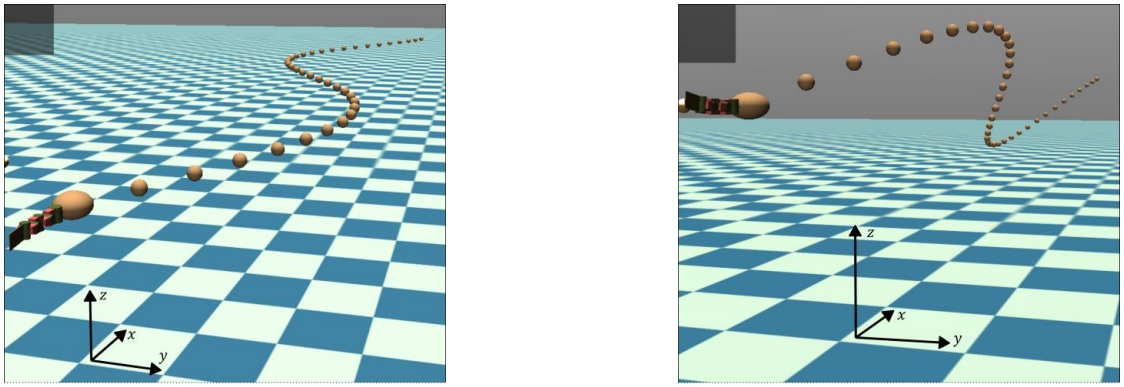
The above trajectory can be rotated using the following relation (5.5a) where $\mathbf{R}_{x,\theta}$ is defined by the rotation matrix in 5.5b.

$$\boldsymbol{y}_R = \mathbf{R}_{x,\theta}\, \boldsymbol{y} \tag{5.5a}$$

$$\mathbf{R}_{x,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.5b}$$

$$\theta \sim \sigma_\theta(-\pi, \pi)\ \text{rad} \tag{5.5c}$$

When the trajectory (Fig. 5.13a) generated by generator (5.4) is rotated about x-axis by 45 degrees. Using (5.5). It will appear as in the following Fig. 513b.



(a) (b)

Fig. 5.13: (a) 2D path in the horizontal $x-y$ plane in front of the robot in simulation generated using (5.7). (b) The same path rotated using (5.8) to 3D path in $x-y-z$ space.

If the amplitude A, the wavelength λ, and the rotation angle θ are all sampled from random distributions during training, then it is safe to say that after training, the agent will have diverse experience to track most trajectories it will face during testing.

5.6.2 Navigation Task

The situation can be understood using the following Fig. 5.14. The simulated model of the robot is located at O_R. The target point is shown in dark black at distance l from the center of the head of the robot. The vicinity of the target is shown as a relatively less opaque region around the target. The target makes an angle θ_z in the horizontal $x-y$ plane and θ_y in the vertical $x-z$ plane from the center of the head. These angles can be calculated as,

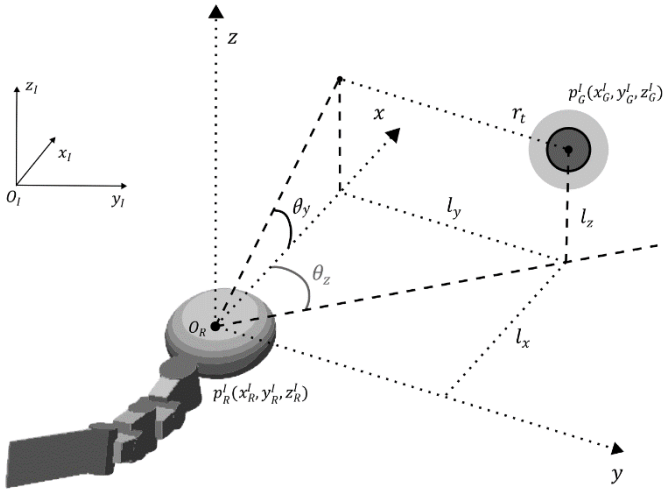


Fig. 5.14: Visualization of robot model and target.

$$\theta_z = \tan^{-1}\frac{y_G - y_R}{x_G - x_R} \tag{5.6a}$$

$$\theta_y = \tan^{-1}\frac{z_G - z_R}{x_G - x_R} \tag{5.6b}$$

Based on above scenario, the following reward function is used,

$$r = r_{arr} + c_{\theta_z} + c_{\theta_y} + c_t \tag{5.7a}$$

$$c_{\theta_z} = -|\theta_z| \tag{5.7b}$$

$$c_{\theta_y} = -|\theta_y| \tag{5.7c}$$

$$r_{arr} = 10 \tag{5.7d}$$

$$c_t = -0.01 \tag{5.7e}$$

So, the agent is penalized for maintaining large deviation angles with the target point in the horizontal and vertical planes via the dense terms c_{θ_z} and c_{θ_y}. c_t is timing penalty to complete the task as soon as possible. r_{arr} is a sparse reward assigned when the agent reaches within the threshold distance of the target point.

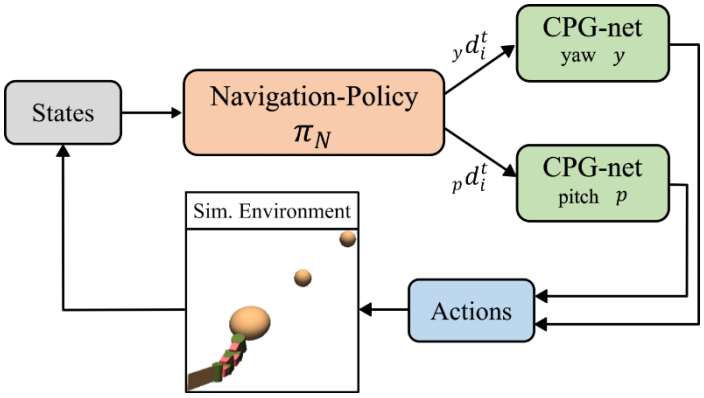


Fig. 5.15: Control schematic for navigation training.

Control schematic for 3D navigation training is presented in Fig. 5.15. A navigation policy π_N is trained according to reward function (5.7). The policy outputs deviations or biases of the two CPG networks (5.1) (yaw for yaw joints and pitch for pitch joints). The oscillators are integrated according to latest policy outputs and the actions are fed to the agent.

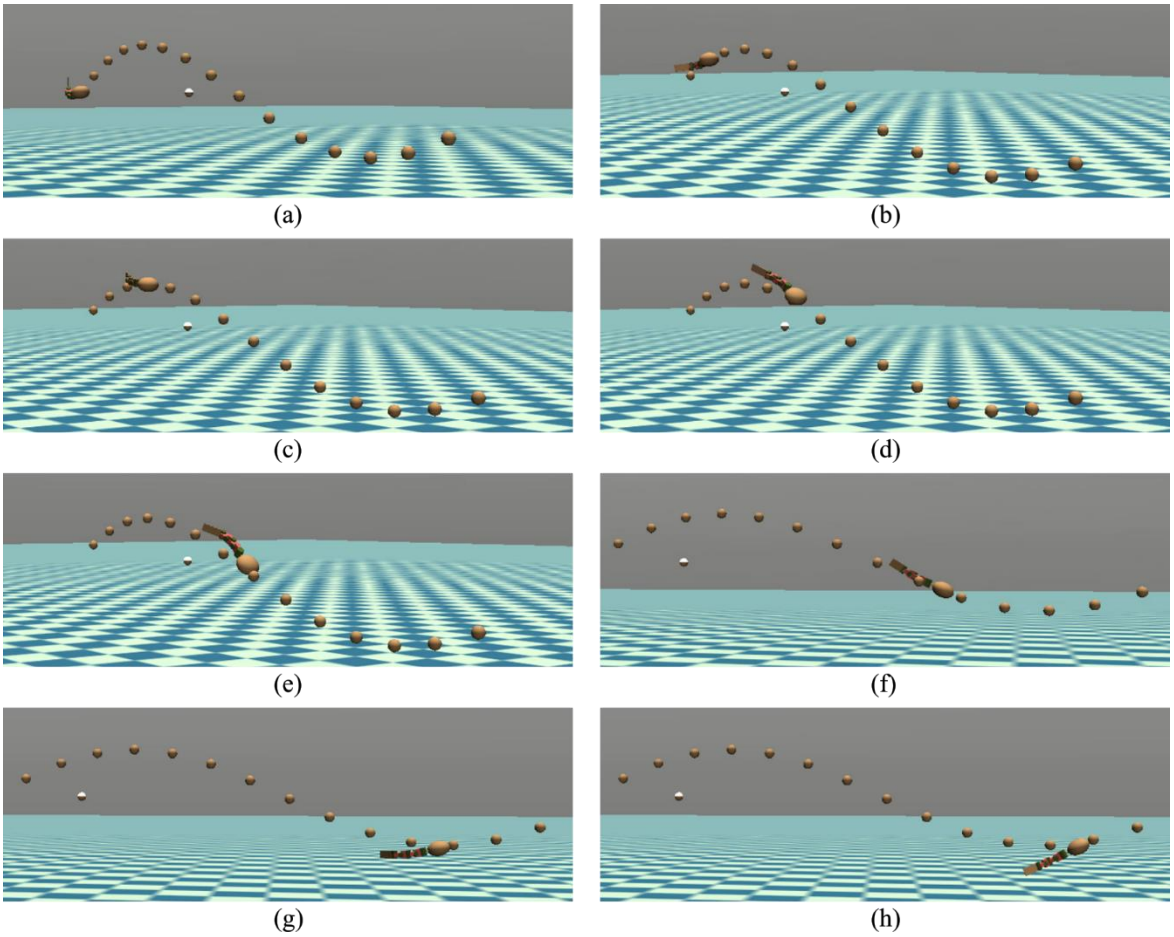


Fig. 5.16: Snapshots from path-following of 2-DoF BCFbot in simulation. The robot starts from the left side (a) and swims to the right side (h). It utilizes both the pitch and yaw joints. Pitch joints adjust the pitch (depth) of the robot to first increase it and then decrease it according to the path while the yaw joints undulate to produce propulsion to complete the path.

5.6.3 Navigation results

The agent has pitch and yaw joints. The yaw joints are used to produce propulsion and the pitch joints are used to adjust the attitude of the robot. The angle the robot makes with respect to the target point in the horizontal and vertical plane, θ_z and θ_y, are used as state space. And the biases of yaw ${}_yd_i^t$ and pitch ${}_pd_i^t$ joints are set as action space.

For navigation learning, different paths are generated during the training process (like shown by small spheres in Fig. 5.13) according to Sec. 5.6.1. In each path, there are discrete path points or targets. These points serve as target points for the agent. The agent is supposed to traverse the entire path by following these points. The environment is reset to the starting

position if the agent completes the entire path or if the episode length expires, whichever happens earlier. After the agent completes one path 20 times, a new path is generated from the next episode and training continues like this.

When the agent is trained using the above-mentioned training procedure, it can track the paths generated using the generator (5.4) and rotator (5.5). Some snapshots of the agent tracking a path are shown in Fig. 5.16.

Fig. 5.16 contains some snapshots of the agent traversing a single trajectory. The agent is tested for several other trajectories; the results can be seen in Fig. 5.17. It can be observed that the agent is able to navigate the trajectories with good accuracy.

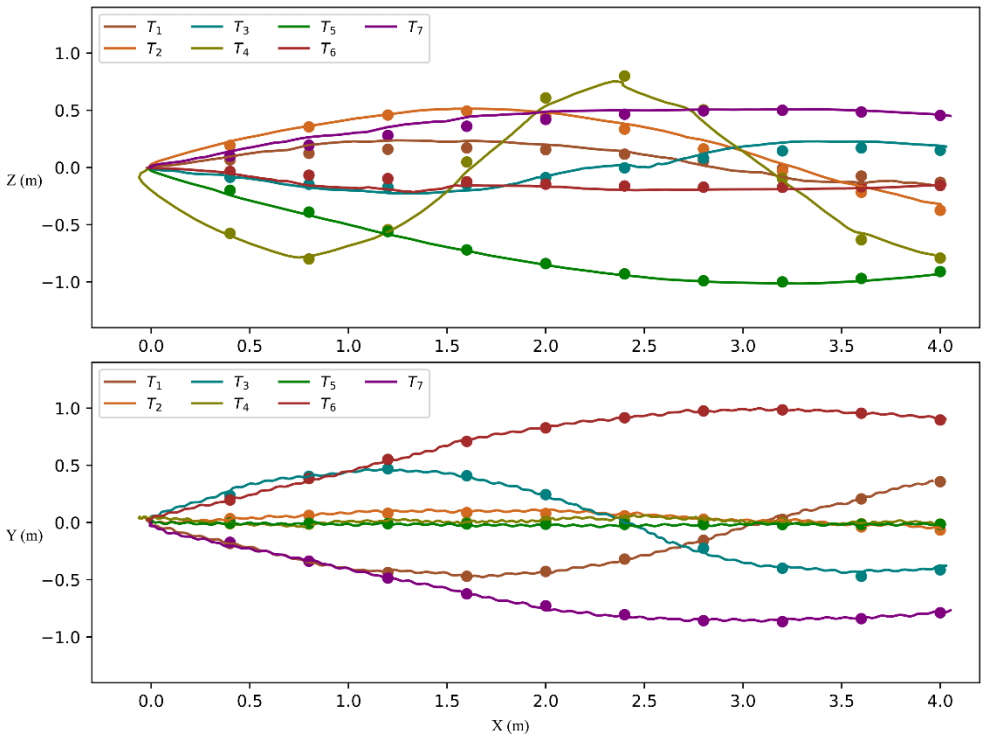


Fig. 5.17: (Top:) $x - z$ and (bottom:) $y - z$ view of navigation along trajectories. Dots represent reference paths and solid lines represent path traced by the centre of robot's head.

5.7 Discussion

For the yaw motion, caudal fin poses high drag resistance to the water and hence contributes predominantly to the propulsion. For the pitch direction, since the caudal fin poses very less (almost zero) drag resistance, therefore it contributes minimally to the propulsion, but

it helps to adjust the pitch/attitude of the overall thrust vector generated by the rhythmic motion of yaw joints. In some organisms, such as dolphins, the caudal fin is oriented laterally. In such cases, rhythmic pitch motion will render prime contribution to propulsion.

In this section, the tail, and hence the fin, are shown to perform both yaw and pitch motions. As mentioned above, the main contribution to propulsion is from the periodicity induced in the yaw direction. Pitch direction is only used for aperiodic adjustments of attitude towards desired direction. Any rhythmic of periodic undulations in pitch directions will lead to very low value of thrust. This low contribution is due to the drag posed by the top and bottom surfaces of pitch joints and not due to the fin.

Based on the above observation, one improvement that can be suggested with respect to design is using cross fin on similar type of platforms. Cross-fin will be a combination of axially oriented BCF-like caudal fin and a laterally oriented dolphin-like caudal fin. Such a fin will have better thrust generation. With cross-fin, rhythmic motion can be rendered on both type of joints since both yaw and pitch motion will pose drag resistance. We cannot find such usage in literature to refer to, but such motion will look roughly similar to flagellum like motion realized on a robotic flagellum [146], although the generation method and design are very different.

If the design is realized, the next step will be to modify the control schematic. The schematic should develop rhythmic patterns for both types of joints. Using either one mode, i.e., either classical BCF-type mode or dolphin-type mode will be straightforward and will follow a similar strategy used in this chapter (Sec. 5.3). Using both modes together will be complex. The second network should be used in similar configuration to the first one. Both networks should operate in limit-cycle as well as point-attractor mode. Comparison of

swimming performance of such design with traditional MPF type depth swimmers will be an interesting direction to explore.

Fig. 5.18: *Left*: Setup for on-surface 2D tracking. *Right*: Setup for underwater tracking.

Data Collection for 3D Navigation

Data results like the position of the robot are not available for Fig. 5.11 and Fig. 5.12. The robot for these two cases is teleoperated using a handheld joystick. The existing tracking setup can track the robot while swimming on the surface of water using markers attached to the part of the head outside water. As soon as the entire robot dives in, the infrared light cannot penetrate inside water and can no longer be reflected by the markers and the cameras cannot track the markers (as explained in 5.18-*left*).

Underwater navigation is performed in simulation in Sec. 5.6. Hardware testing on physical pitch and yaw BCFbot will be performed later. Another setup is being prepared to get position feedback for underwater navigation (Fig. 5.18-*right*). This setup includes a single video camera on the top of the pool. This camera will track a color marker on the robot to get its planar trajectory. A depth sensor will be used to collect depth information. The planar information from the camera will be fused with depth information from a highly sensitive pressure sensor installed on the head of the robot to reproduce and compile 3D trajectory. This way we will be able to collect data for hardware testing.

CHAPTER 6

CONCLUSION AND DISCUSSION

Findings of this study are concluded in this section. Some points are discussed afterwards which can serve as potential future directions.

Overall, three streams are followed which are summarized one by one,

1. The first stream is the development of different swimming gaits inspired by natural organisms in marine life. Classically, BCF swimmers are divided into five categories. Three of them (i.e., the anguilliform, sub-carangiform, and carangiform) are targeted in this part. To mimic the remaining two (i.e., the thunniform and ostraciform), certain design considerations must be followed which are not considered in this study since the focus is kept on developing swimming controllers instead of design. An architecture is proposed which combines oscillators and learning methods. These methods are first explained in detail individually (Sec. 2). And then a combined architecture is introduced (Sec. 3). Training methodology is explained along with details of emulation to explain how different motion patterns are explored and learned. After training, simulation results are shared which show the robot performing the three types of swimming gaits. Other than stationary poses

of the robot, fin trajectories are also plotted to provide a better yardstick to compare with hardware results. After simulation, the results are verified on the hardware. The hardware (BCFbot) is specially designed to mimic the general anatomy of BCF swimmers. A close conformity of results is found between simulation and hardware results in terms of stationary poses and trajectories traced by the fin. The three swimming gaits are then compared for the linear and angular speeds they can move the robot with and costs of transport and turning are also calculated for the linear and angular cases. From the comparison, conclusive results are deduced about the three patterns.

a) The carangiform mode is found to be the fastest among all modes. Its cost of transport is high. But this mode can be used when agile and fast response is necessarily required.
b) The sub-carangiform gait is found to be the cheapest in terms of energy consumption. The speed is not the fastest but does not fall too much behind the carangiform gait. Moreover, the cost of transport is the lowest, making it the most economical gait with respect to energy consumption. Hence, this mode can be used for long duration cruising to save energy.
c) Anguilliform gait, being the slowest and the most expensive can elicit multimodal locomotion (i.e., forward and backward). Hence, this mode is useable in congested places where it is difficult to turn, and the robot must reverse by maintaining the same heading.

Hence, in the first part, different swimming gaits are learned, and some benefits related to them are deduced.

2. The second direction is related to navigation. After the motion patterns are developed, the next step is to navigate the robot along desired trajectories. In this

part, an extended version of the architecture is proposed to perform navigation of robot along pre-defined trajectories. It combines motion primitives and learning methods. Motion primitives are learnt to first optimize motion pattern and then the same set of primitives are trained for navigation. Training methodology is explained in detail (Sec. 4.4). A path is generated which is a set of multiple waypoints. In the training process, the robot in simulation tries to follow the way-points. It is rewarded based on how well it tracks the trajectory. After it successfully completes one trajectory several times, a new random trajectory is generated, and the training is continued like this for many thousand episodes. The method is then tested in simulation first by letting the robot navigate previously known and unknown trajectories. Finally, small-scale hardware testing is carried out to verify the simulation results. Some benefits of the proposed architecture are shown in the form of amplitude and frequency modulation. The architecture successfully demonstrates its ability to perform the navigation task.

Hence, in the second part, a swimming controller is designed and trained to navigate the robot along desired trajectories.

3. The third stream is related to underwater locomotion on a platform with pitch and yaw joints. In this part, an extension to the previous design is considered. The caudal fin of the robot can now move horizontal as well as vertical directions owing to which the robot can move vertically and go at different depths underwater. The architecture is extended to include both the yaw and pitch joints. The new architecture includes two oscillator networks. To verify that the modifications can help the robot to perform underwater locomotion, the testing setup is extended to include measurement of force component in z-direction. Force components are measured for different cases to judge the robot's ability to dive into water. Finally,

hardware testing is performed using teleoperation. Free-swimming testing results reveal that the robot with modified design and control architecture can easily dive in, maneuver, and locomote underwater. In the end, the path-following scheme for the planar case (Sec. 4) is extended for 3D path-following. Trajectory generation is extended to generate 3D paths. Training is done in simulation in which the agent is trained to complete randomly generated paths. Later, the agent is tested for several cases, and it is found that the extended design and schematics can well realize 3D path-following.

Hence, in the third and the last part, the robot's design and the architecture is extended to perform underwater locomotion and navigation.

The following discussion is related to the applicability of the architectures proposed in this study to different areas,

1) Extension to visual and acoustic navigation:

In the presented work, exteroceptive perception is not considered. The most common external sensing includes visual feedback from cameras. Other types include 2D and 3D laser range finders [105] which provide distance information of surroundings in planar and spatial forms. Conventional laser range finders cannot be used underwater; only special laser sensors can operate in marine environment. But cameras can be used for marine navigation. In the underwater environment ultrasonic sensing is also used for range sensing and obstacle detection [159]. There are many studies to directly accommodate visual sensing with DRL to perform navigation in terrestrial domain sometimes by using convolutional layers to extract certain features [106, 107, 109]. Similarly, vision-based underwater target tracking using DRL [113] and vision-based navigation on robotic dolphin using deep stereo networks and RL [130]

are examples from marine domain. Underwater navigation on a robotic vehicle with flappers [129], and on a robotic manta ray [160] is also vision-based.

Since our proposed methods use DRL as high-level policies, similar architecture without significant changes can be used to include external sensing like including visual and acoustic feedback etc. Deep learning based on multi-layer perceptron models have a benefit that high-dimensional feedbacks can directly be accommodated. For example, convolutional layers can be added before deep layers to accommodate feedback from cameras. This way swimming gaits can be optimized for different scenarios. For instance, for target following, a faster gait can be developed when the target is moving fast, and an energy efficient gait can be developed when the target is cruising slowly. Similarly, a gait with more propulsive force can be optimized when swimming against the flow and an efficient one while swimming in the direction of flow by using pressure sensing. These can be interesting future directions. Also, this will be a step towards developing a unified framework for gait generation, modulation, and navigation.

2) The frequency with which different types of feedback is used can be different. Visual feedback usually requires some preprocessing whereas joint positions are almost immediately available. Since the framework used in this study uses separate oscillators at lower level, it can ensure continuous rhythm generation in parallel. There is no interruption in motion generation due to any problem or delay in feedback. The central network can handle feedback whenever it is available and can modulate oscillators according to requirement. In this way, it is easy to accommodate different modalities.

3) Applicability/extension to MPF locomotion:

Median and/or Pectoral Fin (MPF) locomotion alone and mixed with caudal fin are also researched, as discussed in detail in Sec. 1.3 and Sec. 5.1. MPF swimmers are generally slow, but they have good agility in low-speed range. They also have great maneuverability owing to two pectoral fins, one on each side. Within MPF swimmers, there is a great diversity in shapes and flexibility of fins (Fig. 1.1). Similar to BCF swimmers, they also have variation in undulation-oscillation spectrum. Hard fins usually oscillate, and softer ones undulate.

In this study, only BCF type of anatomy is considered to develop methods to produce a variety of swimming gaits and to perform navigation. Keeping in view the similarity of motion profiles in both fin types, the same methods can also be used to develop swimming patterns for robots mimicking MPF swimmers. Similarly, the navigation architecture can also be used to train navigation controllers on such platforms.

In case of mixed locomotion by using both the caudal and pectoral fins. Caudal fin is mainly used for propulsion. Pectoral fins are usually used for maneuvering the body or for depth control [67]. Role of pectoral fins depends on their orientation and their attachment axis with the body. Oscillation about horizontal axis will generate greater vertical force and will facilitate depth control. On the other hand, oscillation about vertical axis will generate greater horizontal force and will facilitate propulsion and maneuvering. Very flexible fins will generate mixed force profiles irrespective of attachment axis with the body and will require an active control mechanism. The method studied in Sec. 3.1 uses the force generated by the caudal fin to regulate swimming patterns. The architecture can be extended to include pectoral fins to generate desired force profiles in both directions for precise control.

Using combined control for platforms with both fin types will enhance agility and maneuverability. In this way, the scope of its use and applications can be expanded. For the above case, a thrust-platform like the one used in this work (Fig. 3.2) can be developed for pectoral fins. This can be a possible future direction of this work.

Some points are discussed below which can also help to point towards possible future directions,

1) In literature, many directions are followed in marine robotics. Modelling of fluid forces and study of fluid dynamics around the robot is one of the commonly researched areas (as briefly discussed in Sec. 1.4.1). In this study, fluid mechanics are not studied. Several studies have been conducted to explore hydrodynamics of fish-like locomotion [161], study fluid interaction between dorsal and caudal fins [162], observe differences in wake patterns for different morphologies [163], perform flow relative control of fish-like robots [58], study flow around robotic counterparts of natural swimmers [75], investigate pectoral and caudal fin performance [164], research kinematic variations on fish-like robots [61], examine impacts of dorsal fins on undulatory platform [154] and caudal fin geometry on overall performance of the robot [153], observe vortices on high-frequency platforms [30], and harness vortices pattern in case of multi-agent swimming [165] etc.. In Sec. 3, different gaits are developed. These gaits can be analyzed in terms of fluid mechanics. To study hydrodynamics of different swimming patterns, respective patterns can first be generated and then studied to find out which pattern can benefit most from wake patterns.
2) Fin-shape [153] and fin-flexibility [166] also impacts the performance of different swimming gaits. Therefore, fin-geometry and fin-stiffness can be optimized to generate higher propulsion target.

3) Similarly, aspect ratio of body length to fin length will have a substantial impact on the performance of robot and on each swimming gait. Certain aspect ratios may be suitable with certain gaits.
4) Caudal fin is explored in this study, comparison to dorsal and anal fins or investigation of combined performance can also be done.
5) Extension to visual and acoustic navigation by installing on-board cameras and other sensors and including features like obstacle avoidance and target following can be very useful for aquatic life exploration.

Above are just a few of many possible directions for future work and research to explore the potential in BCF/MPF mimicking robots and other similar platforms. Some of them are already being explored in our research group and interesting results are expected soon.

BIBLIOGRAPHY

[1] C. C. Lindsey, "Form, Function, and Locomotory Habits in Fish," *Fish Physiology,* vol. 7, pp. 1-100, 1978.

[2] M. Sfakiotakis, D. M. Lane, and J. B. C. Davies, "Review of fish swimming modes for aquatic locomotion," *IEEE Journal of Oceanic Engineering,* vol. 24, no. 2, pp. 237-252, 1999, doi: 10.1109/48.757275.

[3] BlueRobotics. "BlueROV2: The World's Most Affordable High-Performance ROV." https://bluerobotics.com/store/rov/bluerov2/ (accessed 3 Mar 2023, 2023).

[4] A. J. Ijspeert, A. Crespi, and J.-M. Cabelguen, "Simulation and robotics studies of salamander locomotion," *Neuroinformatics,* vol. 3, no. 3, pp. 171-195, 2005/09/01 2005, doi: 10.1385/NI:3:3:171.

[5] A. J. Ijspeert and A. Crespi, "Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 10-14 April 2007 2007, pp. 262-268, doi: 10.1109/ROBOT.2007.363797.

[6] V. Di Santo *et al.*, "Convergence of undulatory swimming kinematics across a diversity of fishes," *Proceedings of the National Academy of Sciences,* vol. 118, no. 49, p. e2113206118, 2021/12/07 2021, doi: 10.1073/pnas.2113206118.

[7] K. A. McIsaac and J. P. Ostrowski, "A Framework for Steering Dynamic Robotic Locomotion Systems," *The International Journal of Robotics Research,* vol. 22, no. 2, pp. 83-97, 2003/02/01 2003, doi: 10.1177/0278364903022002001.

[8] R. K. Katzschmann, J. DelPreto, R. MacCurdy, and D. Rus, "Exploration of underwater life with an acoustically controlled soft robotic fish," *Science Robotics,* vol. 3, no. 16, p. eaar3449, 2018/03/21 2018, doi: 10.1126/scirobotics.aar3449.

[9] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, "From Swimming to Walking with a Salamander Robot Driven by a Spinal Cord Model," *Science,* vol. 315, no. 5817, pp. 1416-1420, 2007/03/09 2007, doi: 10.1126/science.1138353.

[10] X. Niu, J. Xu, Q. Ren, and Q. Wang, "Locomotion Generation and Motion Library Design for an Anguilliform Robotic Fish," *Journal of Bionic Engineering,* vol. 10, no. 3, pp. 251-264, 2013/09/01 2013, doi: 10.1016/S1672-6529(13)60221-8.

[11] J.-M. Cabelguen, C. Bourcier-Lucas, and R. Dubuc, "Bimodal Locomotion Elicited by Electrical Stimulation of the Midbrain in the Salamander Notophthalmus viridescens," *The Journal of Neuroscience,* vol. 23, no. 6, p. 2434, 2003, doi: 10.1523/JNEUROSCI.23-06-02434.2003.

[12] S. Fujiwara and S. Yamaguchi, "Development of Fishlike Robot that Imitates Carangiform and Subcarangiform Swimming Motions," *Journal of Aero Aqua Bio-mechanisms,* vol. 6, pp. 1-8, 03/17 2017, doi: 10.5226/jabmech.6.1.

[13] Z. Wu, J. Yu, M. Tan, and J. Zhang, "Kinematic Comparison of Forward and Backward Swimming and Maneuvering in a Self-Propelled Sub-Carangiform Robotic Fish," *Journal of Bionic Engineering,* vol. 11, no. 2, pp. 199-212, 2014/04/01/ 2014, doi: https://doi.org/10.1016/S1672-6529(14)60037-8.

[14] H. E. Daou, T. Salumäe, A. Ristolainen, G. Toming, M. Listak, and M. Kruusmaa, "A bio-mimetic design and control of a fish-like robot using compliant structures," in *2011 15th International Conference on Advanced Robotics (ICAR),* 20-23 June 2011 2011, pp. 563-568, doi: 10.1109/ICAR.2011.6088645.

[15] S. Farideddin Masoomi, S. Gutschmidt, X. Chen, and M. Sellier, "The Kinematics and Dynamics of Undulatory Motion of a Tuna-Mimetic Robot," *International Journal of Advanced Robotic Systems,* vol. 12, no. 7, p. 83, 2015/07/01 2015, doi: 10.5772/60059.

[16] K. H. Low, C. W. Chong, and C. Zhou, "Performance study of a fish robot propelled by a flexible caudal fin," in *2010 IEEE International Conference on Robotics and Automation,* 3-7 May 2010 2010, pp. 90-95, doi: 10.1109/ROBOT.2010.5509848.

[17] C. Rossi, W. Coral, J. Colorado, and A. Barrientos, "A motor-less and gear-less bio-mimetic robotic fish design," in *2011 IEEE International Conference on Robotics and Automation*, 9-13 May 2011 2011, pp. 3646-3651, doi: 10.1109/ICRA.2011.5979611.

[18] A. Alvaro, "CSIRO biodiversity and sea-floor mapping mission finds weird, wonderful fish species," *ABC News*, 28 Oct 2022. [Online]. Available: https://amp.abc.net.au/article/101571160

[19] T. Koumoundouros, "A Host of Bizarre Creatures Has Been Found At The Bottom of The Ocean," *Science Alert*, 04 November 2022. [Online]. Available:

https://www.sciencealert.com/a-host-of-bizarre-creatures-has-been-found-at-the-bottom-of-the-ocean

[20] N. University, "A new species of deep-sea fish discovered in the Atacama Trench," *Science Daily*, Oct 12 2022. [Online]. Available: https://www.sciencedaily.com/releases/2022/10/221012132546.htm

[21] Y. Cohen, *Advances in water desalination technologies* (Materials and energy ; vol. 17). Singapore ;: World Scientific Publishing Co. Pte. Ltd., 2021.

[22] Á. Einarsson and Á. D. Óladóttir, *Fisheries and Aquaculture: The Food Security of the Future*. San Diego: Elsevier Science & Technology, 2020.

[23] G. N. Bailey and N. C. Flemming, "Archaeology of the continental shelf: Marine resources, submerged landscapes and underwater archaeology," *Quaternary Science Reviews*, vol. 27, no. 23, pp. 2153-2165, 2008/11/01/ 2008, doi: https://doi.org/10.1016/j.quascirev.2008.08.012.

[24] A. Pecher and J. P. Kofoed, *Handbook of ocean wave energy* (Ocean engineering & oceanography ; volume 7.). Cham], Switzerland: Springer Open, 2017.

[25] J. Das *et al.*, "Data-driven robotic sampling for marine ecosystem monitoring," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1435-1452, 2015/10/01 2015, doi: 10.1177/0278364915587723.

[26] H. Singh, T. Maksym, J. Wilkinson, and G. Williams, "Inexpensive, small AUVs for studying ice-covered polar environments," *Science Robotics*, vol. 2, no. 7, p. eaan4809, 2017/06/28 2017, doi: 10.1126/scirobotics.aan4809.

[27] A. Raj and A. Thakur, "Fish-inspired robots: design, sensing, actuation, and autonomy—a review of research," *Bioinspiration and Biomimetics*, vol. 11, no. 3, p. 031001, 2016/04/13 2016, doi: 10.1088/1748-3190/11/3/031001.

[28] J. Fras, Y. Noh, M. Macias, H. Wurdemann, and K. Althoefer, "Bio-Inspired Octopus Robot Based on Novel Soft Fluidic Actuator," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 21-25 May 2018 2018, pp. 1583-1588, doi: 10.1109/ICRA.2018.8460629.

[29] M. Sfakiotakis, A. Kazakidi, N. Pateromichelakis, and D. P. Tsakiris, "Octopus-inspired eight-arm robotic swimming by sculling movements," in *2013 IEEE International Conference on Robotics and Automation*, 6-10 May 2013 2013, pp. 5155-5161, doi: 10.1109/ICRA.2013.6631314.

[30] J. Zhu, C. White, D. K. Wainwright, V. Di Santo, G. V. Lauder, and H. Bart-Smith, "Tuna robotics: A high-frequency experimental platform exploring the performance

space of swimming fishes," *Science Robotics,* vol. 4, no. 34, p. eaax4615, 2019/09/18 2019, doi: 10.1126/scirobotics.aax4615.

[31] R. J. Clapham and H. Hu, "iSplash-II: Realizing fast carangiform swimming to outperform a real fish," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 14-18 Sept. 2014 2014, pp. 1080-1086, doi: 10.1109/IROS.2014.6942692.

[32] D. Chen, Z. Wu, Y. Meng, M. Tan, and J. Yu, "Development of a High-Speed Swimming Robot With the Capability of Fish-Like Leaping," *IEEE/ASME Transactions on Mechatronics,* vol. 27, no. 5, pp. 3579-3589, 2022, doi: 10.1109/TMECH.2021.3136342.

[33] T. Bujard, F. Giorgio-Serchi, and D. Weymouth Gabriel, "A resonant squid-inspired robot unlocks biological propulsive efficiency," *Science Robotics,* vol. 6, no. 50, p. eabd2971, 2021/01/20 2021, doi: 10.1126/scirobotics.abd2971.

[34] G. Li *et al.*, "Self-powered soft robot in the Mariana Trench," *Nature,* vol. 591, no. 7848, pp. 66-71, 2021/03/01 2021, doi: 10.1038/s41586-020-03153-z.

[35] A. Chemori, K. Kuusmik, T. Salumäe, and M. Kruusmaa, "Depth control of the biomimetic U-CAT turtle-like AUV with experiments in real operating conditions," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 16-21 May 2016 2016, pp. 4750-4755, doi: 10.1109/ICRA.2016.7487677.

[36] A. Konno, T. Furuya, A. Mizuno, K. Hishinuma, K. Hirata, and M. Kawada, "Development of Turtle-like Submergence Vehicle," *Journal of The Japan Institute of Marine Engineering,* vol. 41, 01/01 2006, doi: 10.5988/jime.41.SI_158.

[37] M. Kemp, B. Hobson, and J. J. Long, "Madeleine: An agile AUV propelled by flexible fins," in *Proceedings of the 14th International Symposium on Unmanned Untethered Submersible Technology*, USA New Hampshire, 2005, vol. 6.

[38] G. Dudek *et al.*, "AQUA: An Amphibious Autonomous Robot," *Computer,* vol. 40, no. 1, pp. 46-53, 2007, doi: 10.1109/MC.2007.6.

[39] Underwater-GPS(UGPS). "Water-Linked Underwater GPS G2 Systems." https://waterlinked.com/underwater-gps (accessed 27 March 2023, 2023).

[40] Understanding-Lidars. "Velodyne Lidar - What is lidar?" https://velodynelidar.com/what-is-lidar/ (accessed 27 March 2023, 2023).

[41] Depth-Camera-D435. "Intel® RealSense™ Depth Camera D435." https://www.intelrealsense.com/depth-camera-d435/ (accessed 10 April 2023, 2023).

[42] Stereolabs-Camera-ZED-2. "Stereolabs-Depth Sensing Solutions." https://www.stereolabs.com/zed-2/ (accessed 22 March 2023, 2023).

[43] A. Crespi and A. J. Ijspeert, "Online Optimization of Swimming and Crawling in an Amphibious Snake Robot," *IEEE Transactions on Robotics,* vol. 24, no. 1, pp. 75-87, 2008, doi: 10.1109/TRO.2008.915426.

[44] D'AoUT and Aerts, "A kinematic comparison of forward and backward swimming in the eel anguilla anguilla," *The Journal of experimental biology,* vol. 202 (Pt 11), pp. 1511-21, 1999.

[45] X. Niu, J. Xu, Q. Ren, and Q. Wang, "Locomotion Learning for an Anguilliform Robotic Fish Using Central Pattern Generator Approach," *IEEE Transactions on Industrial Electronics,* vol. 61, no. 9, pp. 4780-4787, 2014, doi: 10.1109/TIE.2013.2288193.

[46] K. H. Low, "Current and future trends of biologically inspired underwater vehicles," in *2011 Defense Science Research Conference and Expo (DSR)*, 3-5 Aug. 2011 2011, pp. 1-8, doi: 10.1109/DSR.2011.6026887.

[47] G. V. Lauder, "Fish Locomotion: Recent Advances and New Directions," *Annual Review of Marine Science,* vol. 7, no. 1, pp. 521-545, 2015/01/03 2015, doi: 10.1146/annurev-marine-010814-015614.

[48] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, "A review on modelling, implementation, and control of snake robots," *Robotics and Autonomous Systems,* vol. 60, no. 1, pp. 29-40, 2012/01/01/ 2012, doi: https://doi.org/10.1016/j.robot.2011.08.010.

[49] D. T. Roper, S. Sharma, R. Sutton, and P. Culverhouse, "A review of developments towards biologically inspired propulsion systems for autonomous underwater vehicles," *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment,* vol. 225, no. 2, pp. 77-96, 2011/05/01 2011, doi: 10.1177/1475090210397438.

[50] F. Xie *et al.*, "Designs of the Biomimetic Robotic Fishes Performing Body and/or Caudal Fin (BCF) Swimming Locomotion: A Review," *Journal of Intelligent & Robotic Systems,* vol. 102, no. 1, p. 13, 2021/04/19 2021, doi: 10.1007/s10846-021-01379-1.

[51] R. Wang, S. Wang, Y. Wang, L. Cheng, and M. Tan, "Development and Motion Control of Biomimetic Underwater Robots: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 52, no. 2, pp. 833-844, 2022, doi: 10.1109/TSMC.2020.3004862.

[52] Y. Li *et al.*, "A comprehensive review on fish-inspired robots," *International Journal of Advanced Robotic Systems,* vol. 19, no. 3, p. 17298806221103707, 2022.

[53] N. Gravish and G. V. Lauder, "Robotics-inspired biology," *Journal of Experimental Biology,* vol. 221, no. 7, p. jeb138438, 2018, doi: 10.1242/jeb.138438.

[54] M. Triantafyllou and G. Triantafyllou, "An Efficient Swimming Machine," *Scientific American - SCI AMER,* vol. 272, pp. 64-70, 03/01 1995, doi: 10.1038/scientificamerican0395-64.

[55] K. A. Morgansen, P. A. Vela, and J. W. Burdick, "Trajectory stabilization for a planar carangiform robot fish," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, 11-15 May 2002 2002, vol. 1, pp. 756-762 vol.1, doi: 10.1109/ROBOT.2002.1013449.

[56] S. Saimek and P. Y. Li, "Motion Planning and Control of a Swimming Machine," *The International Journal of Robotics Research,* vol. 23, no. 1, pp. 27-53, 2004/01/01 2004, doi: 10.1177/0278364904038366.

[57] J. Ostrowski and J. Burdick, "The Geometric Mechanics of Undulatory Robotic Locomotion," *The International Journal of Robotics Research,* vol. 17, no. 7, pp. 683-701, 1998/07/01 1998, doi: 10.1177/027836499801700701.

[58] T. Salumäe and M. Kruusmaa, "Flow-relative control of an underwater robot," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences,* vol. 469, no. 2153, p. 20120671, 2013/05/08 2013, doi: 10.1098/rspa.2012.0671.

[59] K. EunJung and Y. Youngil, "Design and dynamic analysis of fish robot: PoTuna," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 26 April-1 May 2004 2004, vol. 5, pp. 4887-4892 Vol.5, doi: 10.1109/ROBOT.2004.1302492.

[60] R. Zhang, Z. Shen, and Z. Wang, "Ostraciiform Underwater Robot With Segmented Caudal Fin," *IEEE Robotics and Automation Letters,* vol. 3, no. 4, pp. 2902-2909, 2018, doi: 10.1109/LRA.2018.2847198.

[61] W. Li, W. Tianmiao, W. Guanhao, and L. Jinlan, "A novel method based on a force-feedback technique for the hydrodynamic investigation of kinematic effects on robotic fish," in *2011 IEEE International Conference on Robotics and Automation*, 9-13 May 2011 2011, pp. 203-208, doi: 10.1109/ICRA.2011.5979907.

[62] T. Y.-T. Wu, "Swimming of a waving plate," *Journal of Fluid Mechanics,* vol. 10, no. 3, pp. 321-344, 1961, doi: 10.1017/S0022112061000949.

[63] M. J. Lighthill, "Note on the swimming of slender fish," *Journal of Fluid Mechanics,* vol. 9, no. 2, pp. 305-317, 1960, doi: 10.1017/S0022112060001110.

[64] M. J. Lighthill, "Large-Amplitude Elongated-Body Theory of Fish Locomotion," *Proceedings of the Royal Society of London. Series B, Biological Sciences,* vol. 179, no. 1055, pp. 125-138, 1971. [Online]. Available: http://www.jstor.org/stable/76021.

[65] S. D. Kelly and R. M. Murray, "Modelling efficient pisciform swimming for control," *International Journal of Robust and Nonlinear Control,* vol. 10, no. 4, pp. 217-241, 2000/04/15 2000, doi: https://doi.org/10.1002/(SICI)1099-1239(20000415)10:4<217::AID-RNC469>3.0.CO;2-X.

[66] J. E. Colgate and K. M. Lynch, "Mechanics and control of swimming: a review," *IEEE Journal of Oceanic Engineering,* vol. 29, no. 3, pp. 660-673, 2004, doi: 10.1109/JOE.2004.833208.

[67] K. A. Morgansen, B. I. Triplett, and D. J. Klein, "Geometric Methods for Modeling and Control of Free-Swimming Fin-Actuated Underwater Vehicles," *IEEE Transactions on Robotics,* vol. 23, no. 6, pp. 1184-1199, 2007, doi: 10.1109/LED.2007.911625.

[68] Z. Chen, S. Shatara, and X. Tan, "Modeling of Biomimetic Robotic Fish Propelled by An Ionic Polymer–Metal Composite Caudal Fin," *IEEE/ASME Transactions on Mechatronics,* vol. 15, no. 3, pp. 448-459, 2010, doi: 10.1109/TMECH.2009.2027812.

[69] V. Kopman, J. Laut, F. Acquaviva, A. Rizzo, and M. Porfiri, "Dynamic Modeling of a Robotic Fish Propelled by a Compliant Tail," *IEEE Journal of Oceanic Engineering,* vol. 40, no. 1, pp. 209-221, 2015, doi: 10.1109/JOE.2013.2294891.

[70] V. Kopman and M. Porfiri, "Design, Modeling, and Characterization of a Miniature Robotic Fish for Research and Education in Biomimetics and Bioinspiration," *IEEE/ASME Transactions on Mechatronics,* vol. 18, no. 2, pp. 471-483, 2013, doi: 10.1109/TMECH.2012.2222431.

[71] J. Yu, J. Yuan, Z. Wu, and M. Tan, "Data-Driven Dynamic Modeling for a Swimming Robotic Fish," *IEEE Transactions on Industrial Electronics,* vol. 63, no. 9, pp. 5632-5640, 2016, doi: 10.1109/TIE.2016.2564338.

[72] D. Barrett, M. Grosenbaugh, and M. Triantafyllou, "The optimal control of a flexible hull robotic undersea vehicle propelled by an oscillating foil," in *Proceedings of Symposium on Autonomous Underwater Vehicle Technology,* 2-6 June 1996 1996, pp. 1-9, doi: 10.1109/AUV.1996.532833.

[73] Y. Junzhi, T. Min, W. Shuo, and C. Erkui, "Development of a biomimetic robotic fish and its control algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part*

B (Cybernetics), vol. 34, no. 4, pp. 1798-1810, 2004, doi: 10.1109/TSMCB.2004.831151.

[74] Q. Yan, Z. Han, S.-w. Zhang, and J. Yang, "Parametric Research of Experiments on a Carangiform Robotic Fish," *Journal of Bionic Engineering,* vol. 5, no. 2, pp. 95-101, 2008/06/01/ 2008, doi: https://doi.org/10.1016/S1672-6529(08)60012-8.

[75] M. Hultmark, M. Leftwich, and A. Smits, "Flowfield measurements in the wake of a robotic lamprey," *Experiments in fluids,* vol. 43, pp. 683-690, 11/01 2007, doi: 10.1007/s00348-007-0412-1.

[76] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks,* vol. 21, no. 4, pp. 642-653, 2008/05/01/ 2008, doi: https://doi.org/10.1016/j.neunet.2008.03.014.

[77] J. Zheng, T. Zhang, C. Wang, M. Xiong, and G. Xie, "Learning for Attitude Holding of a Robotic Fish: An End-to-End Approach With Sim-to-Real Transfer," *IEEE Transactions on Robotics,* vol. 38, no. 2, pp. 1287-1303, 2022, doi: 10.1109/TRO.2021.3098239.

[78] T. Zhang *et al.*, "From Simulation to Reality: A Learning Framework for Fish-Like Robots to Perform Control Tasks," *IEEE Transactions on Robotics,* pp. 1-18, 2022, doi: 10.1109/TRO.2022.3181014.

[79] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature,* vol. 518, no. 7540, pp. 529-533, 2015/02/01 2015, doi: 10.1038/nature14236.

[80] S. Yan, Z. Wu, J. Wang, M. Tan, and J. Yu, "Efficient Cooperative Structured Control for a Multijoint Biomimetic Robotic Fish," *IEEE/ASME Transactions on Mechatronics,* vol. 26, no. 5, pp. 2506-2516, 2021, doi: 10.1109/TMECH.2020.3041506.

[81] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," in *International Conference on Machine Learning*, 2015, pp. 1889-1897.

[82] R. Thandiackal *et al.*, "Emergence of robust self-organized undulatory swimming based on local hydrodynamic force sensing," *Science Robotics,* vol. 6, no. 57, p. eabf6354, 2021/08/11 2021, doi: 10.1126/scirobotics.abf6354.

[83] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Netw,* vol. 21, no. 4, pp. 642-653, 2008.

[84] Z. Zhengxing Wu Junzhi Yu Min Tan Jianwei, "Kinematic Comparison of Forward and Backward Swimming and Maneuvering in a Self-Propelled Sub-Carangiform Robotic Fish," *J Bionic Eng,* vol. 11, no. 2, pp. 199-212, 2014.

[85] A. Crespi and A. J. Ijspeert, "Online Optimization of Swimming and Crawling in an Amphibious Snake Robot," *TRO*, vol. 24, no. 1, pp. 75-87, 2008.

[86] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, 2013, doi: 10.1162/NECO_a_00393.

[87] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*, 12-17 May 2009 2009, pp. 763-768, doi: 10.1109/ROBOT.2009.5152385.

[88] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263-279, 2013/03/01 2013, doi: 10.1177/0278364912472380.

[89] W. Kim, C. Lee, and H. J. Kim, "Learning and Generalization of Dynamic Movement Primitives by Hierarchical Deep Reinforcement Learning from Demonstration," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1-5 Oct. 2018 2018, pp. 3117-3123, doi: 10.1109/IROS.2018.8594476.

[90] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 31-36.

[91] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019/01/16 2019, doi: 10.1126/scirobotics.aau5872.

[92] E. Marchesini, D. Corsi, and A. Farinelli, "Benchmarking Safe Deep Reinforcement Learning in Aquatic Navigation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 27 Sept.-1 Oct. 2021 2021, pp. 5590-5595, doi: 10.1109/IROS51168.2021.9635925.

[93] S. B. Behbahani, J. Wang, and X. Tan, "A dynamic model for robotic fish with flexible pectoral fins," in *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 9-12 July 2013 2013, pp. 1552-1557, doi: 10.1109/AIM.2013.6584316.

[94] R. S. Sutton and A. G. Barto, *Reinforcement learning : an introduction* (Adaptive computation and machine learning.). Cambridge, Mass: MIT Press, 1998.

[95] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238-1274, 2013/09/01 2013, doi: 10.1177/0278364913495721.

[96] E. O. Neftci and B. B. Averbeck, "Reinforcement learning in artificial and biological systems," *Nature Machine Intelligence*, vol. 1, no. 3, pp. 133-143, 2019/03/01 2019, doi: 10.1038/s42256-019-0025-4.

[97] Link. "Spring 2020 - Reinforcement Learning." https://cs.uwaterloo.ca/~ppoupart/teaching/cs885-spring20/schedule.html (accessed 1 April 2023, 2023).

[98] U. Berkeley. "Advanced Policy Gradient Methods." https://rll.berkeley.edu/deeprlcoursesp17/docs/ (accessed 21 Feb 2023, 2023).

[99] T. Wang. "Trust Region Policy Optimization." http://www.cs.toronto.edu/~tingwuwang/trpo.pdf (accessed 15 March 2023, 2023).

[100] MATLAB. "Trust Region Policy Optimization (TRPO) Agents." https://ww2.mathworks.cn/help/reinforcement-learning/ug/trpo-agents.html (accessed 18 March 2023, 2023).

[101] J. Hui. "RL — Trust Region Policy Optimization (TRPO) Explained." https://jonathan-hui.medium.com/rl-trust-region-policy-optimization-trpo-explained-a6ee04eeeee9 (accessed 15 March 2023, 2023).

[102] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2016.

[103] M. Volodymyr *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," presented at the International Conference on Machine Learning, 2016/06/11, 2016.

[104] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv e-prints*, p. arXiv:1707.06347, 2017. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2017arXiv170706347S.

[105] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 24-28 Sept. 2017 2017, pp. 31-36, doi: 10.1109/IROS.2017.8202134.

[106] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 24-28 Sept. 2017 2017, pp. 2371-2378, doi: 10.1109/IROS.2017.8206049.

[107] J. Choi, K. Park, M. Kim, and S. Seok, "Deep Reinforcement Learning of Navigation in a Complex and Crowded Environment with a Limited Field of View," in *2019 International Conference on Robotics and Automation (ICRA)*, 20-24 May 2019 2019, pp. 5993-6000, doi: 10.1109/ICRA.2019.8793979.

[108] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *International Conference on Machine Learning*, 2018, pp. 1861-1870.

[109] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-Supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 21-25 May 2018 2018, pp. 5129-5136, doi: 10.1109/ICRA.2018.8460655.

[110] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. v. d. Panne, "Feedback Control For Cassie With Deep Reinforcement Learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1-5 Oct. 2018 2018, pp. 1241-1246, doi: 10.1109/IROS.2018.8593722.

[111] T. Li, H. Geyer, C. G. Atkeson, and A. Rai, "Using Deep Reinforcement Learning to Learn High-Level Policies on the ATRIAS Biped," in *2019 International Conference on Robotics and Automation (ICRA)*, 20-24 May 2019 2019, pp. 263-269, doi: 10.1109/ICRA.2019.8793864.

[112] C. Hubicki *et al.*, "ATRIAS: Design and validation of a tether-free 3D-capable spring-mass bipedal robot," *The International Journal of Robotics Research*, vol. 35, no. 12, pp. 1497-1521, 2016/10/01 2016, doi: 10.1177/0278364916648388.

[113] J. Yu, Z. Wu, X. Yang, Y. Yang, and P. Zhang, "Underwater Target Tracking Control of an Untethered Robotic Fish With a Camera Stabilizer," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 10, pp. 6523-6534, 2021, doi: 10.1109/TSMC.2019.2963246.

[114] T. G. Brown and C. S. Sherrington, "The intrinsic factors in the act of progression in the mammal," *Proceedings of the Royal Society of London. Series B, Containing Papers of a Biological Character*, vol. 84, no. 572, pp. 308-319, 1911/01/01 1911, doi: 10.1098/rspb.1911.0077.

[115] T. G. Brown, "On the nature of the fundamental activity of the nervous centres; together with an analysis of the conditioning of rhythmic activity in progression, and a theory of the evolution of function in the nervous system," (in eng), *J Physiol*, vol. 48, no. 1, pp. 18-46, Mar 31 1914, doi: 10.1113/jphysiol.1914.sp001646.

[116] M. MacKay-Lyons, "Central Pattern Generation of Locomotion: A Review of the Evidence," *Physical Therapy*, vol. 82, no. 1, pp. 69-83, 2002, doi: 10.1093/ptj/82.1.69.

[117] A. I. Selverston, "Invertebrate central pattern generator circuits," (in eng), *Philos Trans R Soc Lond B Biol Sci,* vol. 365, no. 1551, pp. 2329-45, Aug 12 2010, doi: 10.1098/rstb.2009.0270.

[118] B. R. Noga and P. J. Whelan, "The Mesencephalic Locomotor Region: Beyond Locomotor Control," (in English), *Frontiers in Neural Circuits,* Review vol. 16, 2022-May-09 2022, doi: 10.3389/fncir.2022.884785.

[119] P. S. G. Stein, *Neurons, networks, and motor behavior* (Computational neuroscience.). Cambridge, Mass: MIT Press, 1997.

[120] A. H. Cohen, P. J. Holmes, and R. H. Rand, "The nature of the coupling between segmental oscillators of the lamprey spinal generator for locomotion: A mathematical model," *Journal of Mathematical Biology,* vol. 13, no. 3, pp. 345-369, 1982/01/01 1982, doi: 10.1007/BF00276069.

[121] W. Zhao, J. Yu, Y. Fang, and L. Wang, "Development of Multi-mode Biomimetic Robotic Fish Based on Central Pattern Generator," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 9-15 Oct. 2006 2006, pp. 3891-3896, doi: 10.1109/IROS.2006.281800.

[122] C. Stefanini *et al.*, "A Mechanism for Biomimetic Actuation in Lamprey-like Robots," in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, 20-22 Feb. 2006 2006, pp. 579-584, doi: 10.1109/BIOROB.2006.1639151.

[123] D. Lachat, A. Crespi, and I. Auke Jan, "BoxyBot: a swimming and crawling fish robot controlled by a central pattern generator," in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, 20-22 Feb. 2006 2006, pp. 643-648, doi: 10.1109/BIOROB.2006.1639162.

[124] J. Cheng, R. B. Stein, K. Jovanovic, K. Yoshida, D. J. Bennett, and Y. Han, "Identification, Localization, and Modulation of Neural Networks for Walking in the Mudpuppy (Necturus Maculatus) Spinal Cord," *The Journal of Neuroscience,* vol. 18, no. 11, pp. 4295-4304, 1998, doi: 10.1523/jneurosci.18-11-04295.1998.

[125] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 7-12 Oct. 2012 2012, pp. 5026-5033, doi: 10.1109/IROS.2012.6386109.

[126] Y. Zhong, Z. Li, and R. Du, "A Novel Robot Fish With Wire-Driven Active Body and Compliant Tail," *IEEE/ASME Transactions on Mechatronics,* vol. 22, no. 4, pp. 1633-1643, 2017, doi: 10.1109/TMECH.2017.2712820.

[127] L. Wen, T. Wang, G. Wu, and J. Liang, "Hybrid undulatory kinematics of a robotic Mackerel (Scomber scombrus): Theoretical modeling and experimental investigation," *Science China Technological Sciences*, vol. 55, no. 10, pp. 2941-2952, 2012/10/01 2012, doi: 10.1007/s11431-012-4952-0.

[128] I. Hameed, X. Chao, D. Navarro-Alarcon, and X. Jing, "Training Dynamic Motion Primitives using Deep Reinforcement Learning to Control a Robotic Tadpole," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 23-27 Oct. 2022 2022, pp. 6881-6887, doi: 10.1109/IROS47612.2022.9981112.

[129] T. Manderson *et al.*, "Vision-Based Goal-Conditioned Policies for Underwater Navigation in the Presence of Obstacles," in *Robotics: Science and Systems*, 2020.

[130] S. Yan, Z. Wu, J. Wang, M. Tan, and J. Yu, "Marine Autonomous Navigation for Biomimetic Underwater Robots Based on Deep Stereo Attention Network," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 27 Sept.-1 Oct. 2021 2021, pp. 8418-8423, doi: 10.1109/IROS51168.2021.9636159.

[131] W. Ubellacker, N. Csomay-Shanklin, T. G. Molnar, and A. D. Ames, "Verifying Safe Transitions between Dynamic Motion Primitives on Legged Robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 27 Sept.-1 Oct. 2021 2021, pp. 8477-8484, doi: 10.1109/IROS51168.2021.9636537.

[132] A. Pervez, Y. Mao, and D. Lee, "Learning deep movement primitives using convolutional neural networks," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 15-17 Nov. 2017 2017, pp. 191-197, doi: 10.1109/HUMANOIDS.2017.8246874.

[133] C. Li, R. Lowe, and T. Ziemke, "A Novel Approach to Locomotion Learning: Actor-Critic Architecture using Central Pattern Generators and Dynamic Motor Primitives," *Frontiers in Neurorobotics*, Original Research vol. 8, 2014. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2014.00023.

[134] M. Travers, J. Whitman, and H. Choset, "Shape-based coordination in locomotion control," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1253-1268, 2018/09/01 2018, doi: 10.1177/0278364918761569.

[135] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328-373, 2013, doi: 10.1162/NECO_a_00393.

[136] J. Palmisano, R. Ramamurti, K. Lu, J. Cohen, W. Sandberg, and B. Ratna, "Design of a Biomimetic Controlled-Curvature Robotic Pectoral Fin," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 10-14 April 2007 2007, pp. 966-973, doi: 10.1109/ROBOT.2007.363110.

[137] P. E. Sitorus, Y. Y. Nazaruddin, E. Leksono, and A. Budiyono, "Design and Implementation of Paired Pectoral Fins Locomotion of Labriform Fish Applied to a Fish Robot," *Journal of Bionic Engineering*, vol. 6, no. 1, pp. 37-45, 2009/03/01/ 2009, doi: https://doi.org/10.1016/S1672-6529(08)60100-6.

[138] J. S. Palmisano, J. D. Geder, R. Ramamurti, W. C. Sandberg, and B. Ratna, "Robotic Pectoral Fin Thrust Vectoring Using Weighted Gait Combinations," *Applied Bionics and Biomechanics*, vol. 9, p. 802985, 1900/01/01 2012, doi: 10.3233/ABB-2012-0064.

[139] D. Chen, Z. Wu, P. Zhang, M. Tan, and J. Yu, "Performance Improvement of a High-Speed Swimming Robot for Fish-Like Leaping," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1936-1943, 2022, doi: 10.1109/LRA.2022.3142409.

[140] C. Zhou and K. H. Low, "Design and Locomotion Control of a Biomimetic Underwater Vehicle With Fin Propulsion," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 1, pp. 25-35, 2012, doi: 10.1109/TMECH.2011.2175004.

[141] Y. Meng, Z. Wu, H. Dong, J. Wang, and J. Yu, "Toward a Novel Robotic Manta With Unique Pectoral Fins," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 3, pp. 1663-1673, 2022, doi: 10.1109/TSMC.2020.3034503.

[142] K. Togawa, M. Mori, and S. Hirose, "Study on three-dimensional active cord mechanism: development of ACM-R2," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, 31 Oct.-5 Nov. 2000 2000, vol. 3, pp. 2242-2247 vol.3, doi: 10.1109/IROS.2000.895302.

[143] M. Mori and S. Hirose, "Three-dimensional serpentine motion and lateral rolling by active cord mechanism ACM-R3," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 30 Sept.-4 Oct. 2002 2002, vol. 1, pp. 829-834 vol.1, doi: 10.1109/IRDS.2002.1041493.

[144] H. Yamada and S. Hirose, "Development of Practical 3-Dimensional Active Cord Mechanism ACM-R4," *J. Robotics Mechatronics*, vol. 18, pp. 305-311, 2006.

[145] H. Yamada *et al.*, "Development of Amphibious Snake-like Robot ACM-R5," in *36th, International symposium on robotics*, Tokyo, 2005, vol. 36: Japan Robot Association;, pp. 133-133. [Online]. Available: https://www.tib.eu/de/suchen/id/BLCP%3ACN066725101. [Online]. Available: https://www.tib.eu/de/suchen/id/BLCP%3ACN066725101

[146] Y. Shumei, M. Shugen, L. Bin, and W. Yuechao, "An amphibious snake-like robot with terrestrial and aquatic gaits," in *2011 IEEE International Conference on Robotics and Automation*, 9-13 May 2011 2011, pp. 2960-2961, doi: 10.1109/ICRA.2011.5979869.

[147] Z. Li, X. Chao, I. Hameed, J. Li, W. Zhao, and X. Jing, "Biomimetic omnidirectional multi-tail underwater robot," *Mechanical Systems and Signal Processing*, vol. 173, p. 109056, 2022/07/01/ 2022, doi: https://doi.org/10.1016/j.ymssp.2022.109056.

[148] C. Wright *et al.*, "Design of a modular snake robot," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 29 Oct.-2 Nov. 2007 2007, pp. 2609-2614, doi: 10.1109/IROS.2007.4399617.

[149] C. Wright *et al.*, "Design and architecture of the unified modular snake robot," in *2012 IEEE International Conference on Robotics and Automation*, 14-18 May 2012 2012, pp. 4347-4354, doi: 10.1109/ICRA.2012.6225255.

[150] D. Rollinson *et al.*, "Design and architecture of a series elastic snake robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 14-18 Sept. 2014 2014, pp. 4630-4636, doi: 10.1109/IROS.2014.6943219.

[151] P. Liljebäck, S. Ø, K. Y. Pettersen, and J. T. Gravdahl, "Mamba - A waterproof snake robot with tactile sensing," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 14-18 Sept. 2014 2014, pp. 294-301, doi: 10.1109/IROS.2014.6942575.

[152] T. Takemori, M. Tanaka, and F. Matsuno, "Ladder Climbing with a Snake Robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1-5 Oct. 2018 2018, pp. 1-9, doi: 10.1109/IROS.2018.8594411.

[153] Z. Huang *et al.*, "Impact of caudal fin geometry on the swimming performance of a snake-like robot," *Ocean Engineering*, vol. 245, p. 110372, 2022/02/01/ 2022, doi: https://doi.org/10.1016/j.oceaneng.2021.110372.

[154] Z. Huang, S. Ma, H. Bagheri, C. Ren, and H. Marvi, "The Impact of Dorsal Fin Design on the Swimming Performance of a Snake-Like Robot," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4939-4944, 2022, doi: 10.1109/LRA.2022.3153903.

[155] Z. Huang, D. Kong, C. Ren, S. Li, and S. Ma, "Performance Study of an Underwater Snake-like Robot with a Flexible Caudal Fin," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, 4-7 Aug. 2019 2019, pp. 1-5, doi: 10.1109/ICMA.2019.8816412.

[156] A. Crespi, K. Karakasiliotis, A. Guignard, and A. J. Ijspeert, "Salamandra Robotica II: An Amphibious Robot to Study Salamander-Like Swimming and Walking Gaits," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 308-320, 2013, doi: 10.1109/TRO.2012.2234311.

[157] Camera. "Hero 10 Black." https://gopro.com/en/us/shop/cameras/hero10-black/CHDHX-101-master.html (accessed 11 Mar 2023, 2023).

[158] Power-Supply. "Elektro-Automatik Bench Power Supply, EA-PS 2042-20 B, 320W." https://hken.rs-online.com/web/p/bench-power-supplies/7568821 (accessed 24 Mar 2023, 2023).

[159] Y. Cong, C. Gu, T. Zhang, and Y. Gao, "Underwater robot sensing technology: A survey," *Fundamental Research*, vol. 1, no. 3, pp. 337-345, 2021/05/01/ 2021, doi: https://doi.org/10.1016/j.fmre.2021.03.002.

[160] Y. Meng, Z. Wu, Y. Li, D. Chen, M. Tan, and J. Yu, "Vision-Based Underwater Target Following Control of an Agile Robotic Manta With Flexible Pectoral Fins," *IEEE Robotics and Automation Letters*, pp. 1-8, 2023, doi: 10.1109/LRA.2023.3250004.

[161] M. S. Triantafyllou, G. S. Triantafyllou, and D. K. P. Yue, "Hydrodynamics of Fishlike Swimming," *Annual Review of Fluid Mechanics*, vol. 32, no. 1, pp. 33-53, 2000/01/01 2000, doi: 10.1146/annurev.fluid.32.1.33.

[162] G. V. Lauder, E. J. Anderson, J. Tangorra, and P. G. A. Madden, "Fish biorobotics: kinematics and hydrodynamics of self-propulsion," *Journal of Experimental Biology*, vol. 210, no. 16, pp. 2767-2780, 2007, doi: 10.1242/jeb.000265.

[163] E. D. Tytell, I. Borazjani, F. Sotiropoulos, T. V. Baker, E. J. Anderson, and G. V. Lauder, "Disentangling the functional roles of morphology and motion in the swimming of fish," (in eng), *Integr Comp Biol*, vol. 50, no. 6, pp. 1140-54, Dec 2010, doi: 10.1093/icb/icq057.

[164] J. L. Tangorra, C. J. Esposito, and G. V. Lauder, "Biorobotic fins for investigations of fish locomotion," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 10-15 Oct. 2009 2009, pp. 2120-2125, doi: 10.1109/IROS.2009.5353953.

[165] S. Verma, G. Novati, and P. Koumoutsakos, "Efficient collective swimming by harnessing vortices through deep reinforcement learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 23, pp. 5849-5854, 2018/06/05 2018, doi: 10.1073/pnas.1800923115.

[166] O. Xie, B. Li, and Q. Yan, "Computational and experimental study on dynamics behavior of a bionic underwater robot with multi-flexible caudal fins," *Industrial Robot: An International Journal*, vol. 45, no. 2, pp. 267-274, 2018, doi: 10.1108/IR-06-2017-0122.