Academic Summary

Reinforcement Learning for Underwater Robots

Author: Azadeh Kooshesh 220200909| Nico Trebbin 218204402 | Iman Sharifitabaar 221202331 | Mohammad Mahmudul Islam 220100157| Md Golam Azam 220100200

Supervisor: Dr. rer. nat. Sebastian Bader

Academic year: Summer semester 2022

---

1. Introduction

Because getting to the depths of the ocean is a bit difficult and dangerous, scientists try to have secure access with the robots. The first underwater robot is "RU27", which can dive to depths of 200 meters to collect data. As for any robotics application, one of the most important questions concerns system reliability, in order to provide the necessary confidence before letting it goes on populated routes. Therefore, in this topic, we want to focus on how a robot can learn like a human and behave flexibly under the water. Our agent explores a virtual environment using a simulator. The communication is handled by ROS, short for Robot Operating System. The agent can calculate the best action to explore the environment using his perceptions, i.e. the sensor and map data, and a reinforcement learning method. The sensor data must be requested continuously due to the ever changing environment.

2. Purpose and Scope

The goal of this program is to explore unknown environments with an underwater robot. Various nearby obstacles can influence the agent's path. That's why we want to implement a software that can explore a high percentage of the environment. While driving, the agent follows a collision-free strategy thus obstacles are avoided. A map is created during the exploration. This map contains sufficient information about the area. After exploring, a map is drawn.

3. User of Product

Our program is only for experienced people. This program is especially designed for underwater researcher and explorer, maritime institutes and the MMIS institute.

4. Functional Requirements

- Get sensor data as input (e.g. Sonar, GPS, Camera, Compass, Pressure sensor)
  For the correct positioning of the robot we will receive the several data from different sensors.
  Large Baseline (LBL) - is a method based on sonar, which works very similar to GPS, just using acoustics instead. For this you need transponders that ping at known intervals. The robot can than calculate its position based on the run length intervals.
  Ultra Short Baseline (USBL) - works the same as the LBL, just that the transponders are very close to each other (usually one device). In this way you get an angle and range measurement. Less accurate, but more compact.
  Doppler Velocity Log (DVL) - uses the doppler effect in multiple directions to give you the velocity against ground.
  Pressure Sensor - you can directly get a measurement of your depth in the water level with this.

Attitude Heading Reference System (AHRS) - based on IMU and Compass, you can get a good estimate of your orientation. The system get the sensor data of the robot.
The end user only gets the resulting map. The data is fetched from HoloOcean, an open source simulator with an OpenAI Gym like API. If time is left also a direct communication using ROS can be established.

- Program explores the map, i.e. travels to unvisited places
- Program creates a map both as a data structure and graphically
- A 3D space is explored
- Experiments can be run from the command line

5. Non-Functional Requirements

- Program should work for a variety of unknown environments provided by the simulator HoloOcean. In the best case, real environments can be considered too but these are not officially supported.
- Executable on common operating systems
  - Focused on Linux, proprietary OS' aren't officially supported
  - The system should be able to run in a Docker container
- Performance evaluation
  The robot performance could be measured whether it is able to detect the collision using sonar sensor. The sonar data needs to be purged from noise. To provide immediate feedback to the user during the exploration, the system is required to run in real time. If possible, real time performance should be maintained even on weak hardware. e.g robot hardware. Further the exploration task should not take too long. A coverage precision cannot be given. HoloOcean unfortunately does only include a compiled environment map thus we can only access it with HoloOcean itself. However, we can provide a 3D scatter plot of the explored environment. Further we can plot the explored points directly into HoloOcean.
- *Precision*
  If our systems fails to acknowledge the current location due to the technical challenges or the connection with robot is lost the user is likely to abort the exploration. Therefore, the systems precision should be comparable to state-of-the-art recognition systems.

6. Constraints and Assumptions

- We assume that sensors won't fail for a long time
- We assume that the current isn't too strong
- We assume that the robot is manually moved to the given depth
- We assume that the input data is not too noisy
- We assume that the robot isn't too big
  - If the robot exceeds a certain size, the collision prevention might fail

We strive for high coverage of the environment within the simulator used. The map can be less defined when real environments are used. Due to the broad variety of different environments, perfect coverage cannot be achieved.

7. Comparing the HoloOcean simulator with Unmanned Underwater Vehicle Simulator (UUV)

Based on the information in marine industry we have two good simulators between all choices, HoloOcean and UUV. In this project we preferred to use HoloOcean due to variety of options, that are provided to us. Therefore, we will mention some of the options of both simulators below.

HoloOcean

- Multi-agent systems
- Acoustic and optical communications
  - Imaging or forward-looking sonar
  - Side scan sonar
  - Multibeam profiling sonar
  - Echo sounders
- High-fidelity camera imagery
- Common underwater navigational sensors
  - Doppler velocity log
  - Depth sensors
  - Inertial measurement units
  - Magnetometer

    - Pose sensors
- Configuration through deck services
- Various underwater agents
- Various realistic testing environments and worlds

But infront we have the UUV simulator which is almost similar to HoloOcean but in "Realistic Sonar Implementations" it has a less options. Also the UUV simulator has fewer sensor options than HoloOcean.

UUV

- Realistic sonar implementations
    - Side scan sonar
    - Multibeam sonar
- Common underwater navigational sensors
    - Camera
    - DVL
    - IMU inertial measurement unit
    - Subsea pressure
    - Positioning transponder
- Gazebo world plugin for generation of current velocity
- 3D current velocity
- Configuration through ROS services

Installation Requirements

Requirements for HoloOcean installation:

- >= Python 3.6
- Several gigabytes of storage
- pip3
- Linux or Windows 64bit
- Preferably a competent GPU
- For Linux: OpenGL 3+

Requirements for UUV installation:

- Ubuntu 14.04.4 LTS Linux
- ROS Indigo

As you can see, HoloOcean can run not only on Windows, but also on Linux. HoloOcean is open-source and has a simple python interface. Therefore, we choose it in our project.

8. State of the Art

Exploring unknown underwater environments is a tough task. Most current algorithms use a combination of frontier-based exploration and view planning methods. Frontier-based exploration divides the map into known and unknown areas. During view planning valid viewpoints are calculated. The agent then drives, actually collision-free, to the best viewpoint. The used methods for the viewpoint and path generation might differ from paper to paper. The usage of deep neural networks can also be of interest for these tasks. A more detailed explanation can be found in the Summaries appendix. A comparison of the used papers can be found in Figure 3 of the visualisations appendix.

9. Proposed Architecture

Our proposed system gathers the sensor data, in particular sonar data to build an internal state, that is also called the internal world map. Based on the agent's internal state and perceptions the system detects any nearby obstacles. Afterwards the coordinates of the most promising points are calculated. This task also includes the transforming of polar to Cartesian coordinates. Furthermore, the next starting point is calculated. The system must create a path for the agent to follow. This process is repeated until the environment is explored. A visualisation can be found in Figure 4. The UML class diagram can be found in Figure 5.

Components of the Robot

Our robot consists of several components. A pose sensor calculates the current position and rotation of the robot. When starting the agent, the position is at the origin. The second sensor is a forward looking sonar sensor. Using that sensor, the robot can scan the environment for possible collisions. The last sensor is a sonar sensor as well, but it's pointing down instead of forward. That sonar scans the underground of the environment. Additionally, if time is left, a camera can be added for a colourized environment representation.

Internal World Map

While driving through the environment, the robot has to keep track of the explored points. Therefore, we need to save the points we've already visited. Each of the points has one of the following labels: `empty`, `collision`, `covered` and `unexplored`. A point has the label `empty`, iff it is covered by the forward looking sonar and there is no collision at this point. The label `collision` is given to each point, that is covered by the forward looking sonar and is obscured by

an obstacle. A point is `covered`, iff the downward looking sonar has scanned this point. An `unexplored` point is every point not covered by the downward looking sonar. For better performance, we store the points in octrees. An octree divides the environment in multiple bounding boxes. The structure of an octree allows us to easily calculate all points inside a border. Using a nearest neighbour algorithm the the next point of approach is sought, i.e. the closest unexplored point. An octree has one more advantage, we can easily skip redundant and very close points. A sonar sensor can generate quite a bit of points. The storage of all the, partly redundant, points isn't efficient at all.

Nearest Neighbours

While exploring the environment, we need to create a path from the current point to the target point. The target point is the closest unexplored point.

Path Generation

To find a path from one point to another, we will use the RRT algorithm, short for rapidly-exploring random trees. The RRT algorithm builds a tree starting from the current point. His goal is to find a path to the target point. The algorithm needs to be aware about the obstacles in the environment, so our robot needs to have a forward looking sonar sensor. If the robot chooses a path with a collision that is not yet known, than the robot will find it out while driving. When he sees the obstacle, the path is changed.

Visualization

After exploring the environment, the map must be visualized. The point cloud generated by the downward looking sonar sensor can be mapped in a 3D environment. The visualization is a 3D scatter plot, a mesh plot can be created if time is left.

A. Summaries

We found 59 paper from IEEE, springer link, google scholar search engines and their are not older than 2017. 11 papers are selected as a final selection. Our criteria for selection were Used method, Agent, source code available, Efficiency, Accuracy, Robot type, Simulator type, Algorithm as pseudo code, Model structure for deep learning, Used sensor, 2D 3D exploration.

Online Robotic Exploration for Autonomous Underwater Vehicles in Unstructured Environments [4]

This paper presents a single-agent exploration algorithm for unknown underwater environments. Using view planning methods, the proposed algorithm has high efficiency. The agent firstly generates the map as a grid map. The sonar data is incorporated into the grid map using a fast cell traversal algorithm. Sensor noise is taken into account using the hit-and-miss approach. After creating the grid map the viewpoints must be generated. The viewpoints with the smallest weighted Euclidean distance are selected. After generating the best viewpoints, a path must be found. A rapidly-exploring random tree is used for this task.

Optimized environment exploration for autonomous underwater vehicles [3]

This paper presents a single-agent exploration algorithm for unknown underwater environments. The proposed algorithm shows high efficiency using frontier-based exploration and view planning methods. In the first step, the algorithm inserts data from the sonar sensor and an optical camera into the map. This map is used for generating viewpoints for further exploration. After the viewpoints are generated a path to the best viewpoint must be defined. Therefore, a rapidly-exploring random tree will be used. In the end, a line of sight controller moves the agent along the path.

Target Search Control of AUV in Underwater Environment With Deep Reinforcement Learning [10]

This paper presents a single- and multi-agent exploration algorithm for unknown underwater environments. With the combination of frontier-based exploration and deep reinforcement learning the proposed algorithm can explore a variety of environment with a high success rate. By integrating deep reinforcement learning, the frontier-based exploration can search a wider environment in less time. When exploring the environment, the algorithm needs to go through three tasks. Firstly, a grid map will be created using the sonar sensor data. This grid map gets normalized by a few operations. In the second step, the next target point must be found. Using the grid map as well as location of the agent and the frontier, a new target is created using a deep neural network. When the next point is calculated, the robot has to find a collision-free path to get there. This task is done by deep reinforcement learning.

Two-Dimensional Frontier-Based Viewpoint Generation for Exploring and Mapping Underwater Environments [5]

This article introduces an single-agent exploration algorithm for unknown underwater environments. Using frontier-based exploration and view planning methods in combination with suitable data structures, the proposed algorithm aims for high efficiency. As it drives through the environment the agent scans the surroundings with a camera and sonar sensors. Due to these sensors a grid map will be generated. For high efficient access the grid map is internally stored in several quad trees. By filtering of sonar signals, even noisy data can be used. After the data has been inserted into the map, the next starting points are calculated. When the next point has been computed, the robot must find a collision-free path to it. This task is done by a rapidly-exploring random tree path planner. After the path is calculated a trajectory tracking controller is used to move the agent to the starting point.

Research on the Stability Motion Control Method of Underwater Vehicle Driven by Data [8]

This article is about the independent navigation capabilities of underwater robots, which is an important prerequisite for their successful underwater operations. According to the conventional stand-alone navigation algorithm in complex underwater environments, the calculation value for real-time navigation is very high, while reinforcement learning faces the next disasters and requires a lot of time to teach learning. This paper proposes the use of an improved DDQN algorithm to study the independent navigation of underwater robots in unfamiliar environments. This algorithm gives underwater robots the ability to learn independently and improves their adaptability in different environments. This bottleneck solves traditional independent navigation algorithms and achieves independent navigation of underwater robots in an environment without map information.

ORBSLAM2 and Point Cloud Processing Towards Autonomous Underwater Robot Navigation [6]

In this paper, they used ORBSLAM2, a vSLAM algorithm that creates a super-point map of Oriented FAST and Rotated BRIEF (ORB) features from video images and localizes the video source. We evaluate the feasibility of cloud point processing in implementing a basic navigation method. The cloud is processed into three-dimensional planes to abstract the surroundings, where the path is easily created, and a robot point controller can steer the robot according to its current position as feedback.

Acoustic Camera-Based Pose Graph SLAM for Dense 3-D Mapping in Underwater Environments [12]

In this paper, an acoustic camera with a rotator was used for dense three-dimensional mapping of the underwater environment. The proposed method first applies a three-dimensional occupation mapping framework based on the rotation of the audio camera around the acoustic axis using a rotator in a fixed position to produce three-dimensional local maps. Then, the scanning of adjacent local maps is performed to calculate the odometer without the intervention of internal sensors, and an approximate real-time compact global map is created. Finally, based on a diagram optimization scheme, offline refinement is performed to create a final condensed global map.

Stonefish: An Advanced Open-Source Simulation Tool Designed for Marine Robotics, With a ROS Interface[1]

This article is about an advanced open source simulator for marine robots using ROS interface called Stonefish.It presents a new software tool, focused on, but not limited to, simulation of intervention autonomous underwater vehicles (I-AUV) and consists of a library written in C++ and a Robot Operating System (ROS) package.The Stonefish library also contains classes implemented directly in OpenGL to represent different types of virtual sensors and actuators used in surface, underwater and even flying robots. All commonly used robot sensors can be simulated with it. The library implements support for keyboard and joystick input as well as a simple GUI. According to the article the most unique feature of this library is the computation of buoyancy and hydrodynamics based on actual geometry.

Autonomous underwater vehicles (AUVs) path planning based on Deep Reinforcement Learning [11]

Researchers in this paper study the path planning of autonomous underwater vehicles by combining deep learning and reinforcement learning techniques, modeling the seabed with WL interpolation surfaces, and proposing a deep reinforcement learning-based path planning model for underwater vehicles. A model is trained in the simulation environment, so the underwater robot can adapt its movements to changing conditions. This article utilizes the AUVS path planning method that combines sensors and global information. WL is used to model the seabed and simulate its general shape, and then environmental information near the location of the AUVS is collected using sensors while the AUVS

is operating in order to construct an AUVS path planning model using deep reinforcement learning.

Underwater robot sensing technology: A survey[2]

In this article it is intended to contribute to this growing area of research in underwater robot sensing. Therefore, it survey the related works of underwater robot sensing technologies including underwater acoustic sensing, underwater optical sensing, underwater magnetic sensing, as well as underwater bionic sensing. Finally, it also proposes some valuable suggestions and future challenges and directions for future researches.

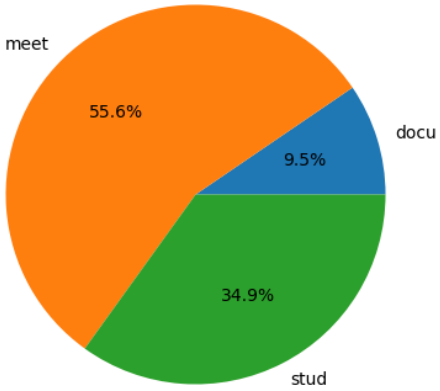
Meta Reinforcement Learning Based Underwater Manipulator Control [7]

In this paper, an underwater manipulator control method based on model-based meta reinforcement learning is introduced that is advantageous in terms of its capability to generate flexible responses to various environmental changes. This learning method rapidly updates the model based on actual application program experiences. The model was then transmitted to the model predictive control, which analyzes the manipulator's control input to achieve the target position. The entire system architecture using ROS Gazebo is organized into three sections: Agent, Trainer, and Run. MuJoCo and Gazebo were used to provide a simulation environment for model-based meta reinforcement learning, and the suggested method was validated and confirmed by introducing model uncertainty. This was accomplished by considering an underwater construction robot under a real-world control environment.

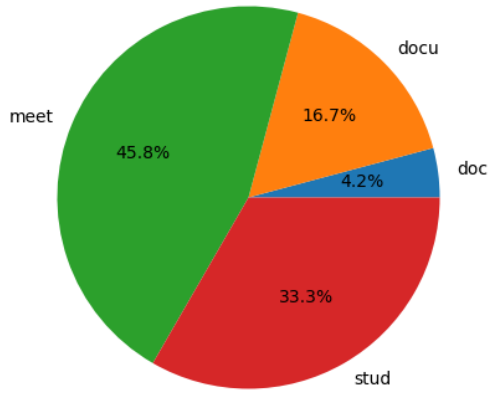
Gradient-Based Reinforcement Learning Techniques For Underwater Robotics Behavior Learning [9]

RL theory surveys the two main classes of RL algorithms: Value Function (VF)-based methods and Policy Gradient (PG)-based techniques. A particular class of algorithms, Actor-Critic methods, born of the combination of PG algorithms with VF methods, is used for the final experimental results: a real underwater task in which the underwater robot Ictineu AUV learns to perform an autonomous cable tracking task.

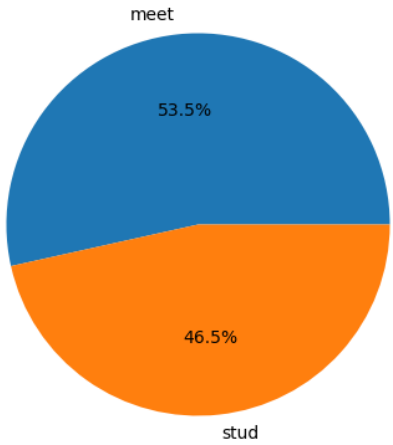
B. Visualisations



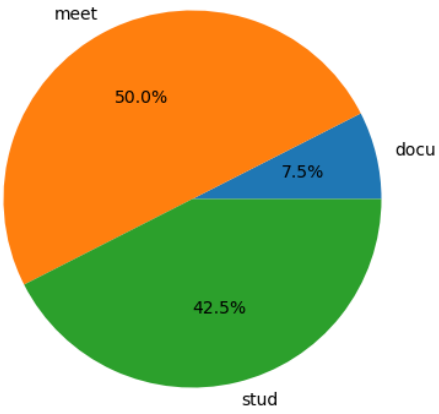
(a) Md Golam Azam



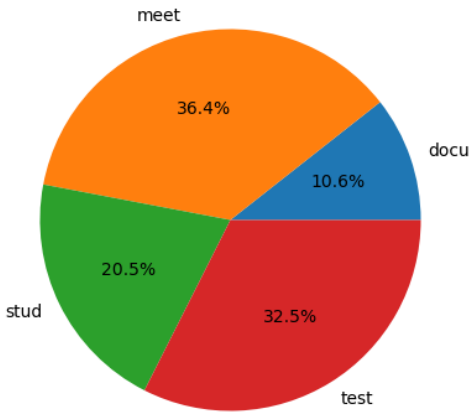
(b) Azadeh Kooshesh



(c) Iman Sharifitabaar



(d) Mohammad Mahmudul Islam



(e) Nico Trebbin

Figure 1: time sheets

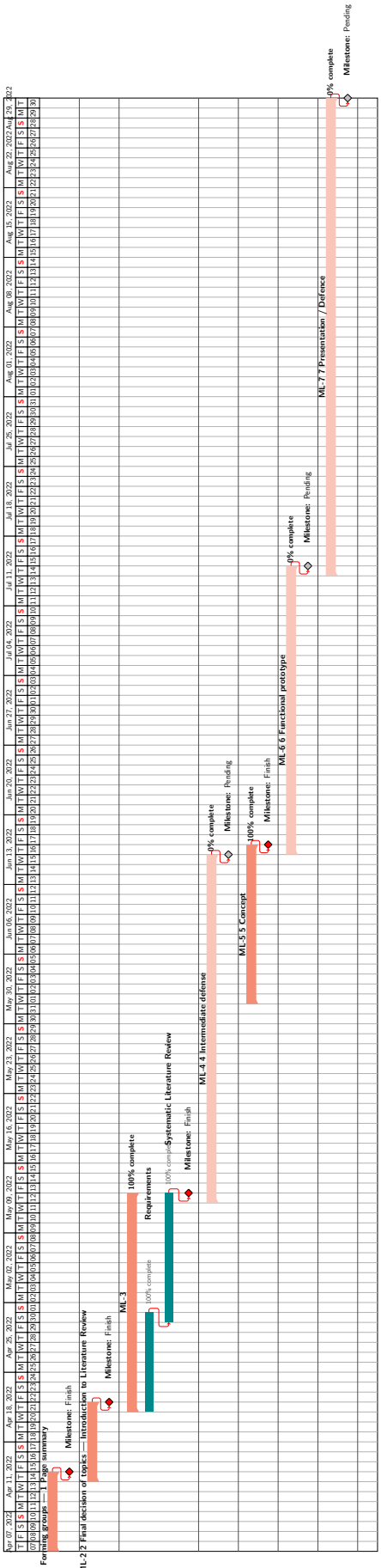


Figure 2: GANTT chart

Figure 3: comparison table

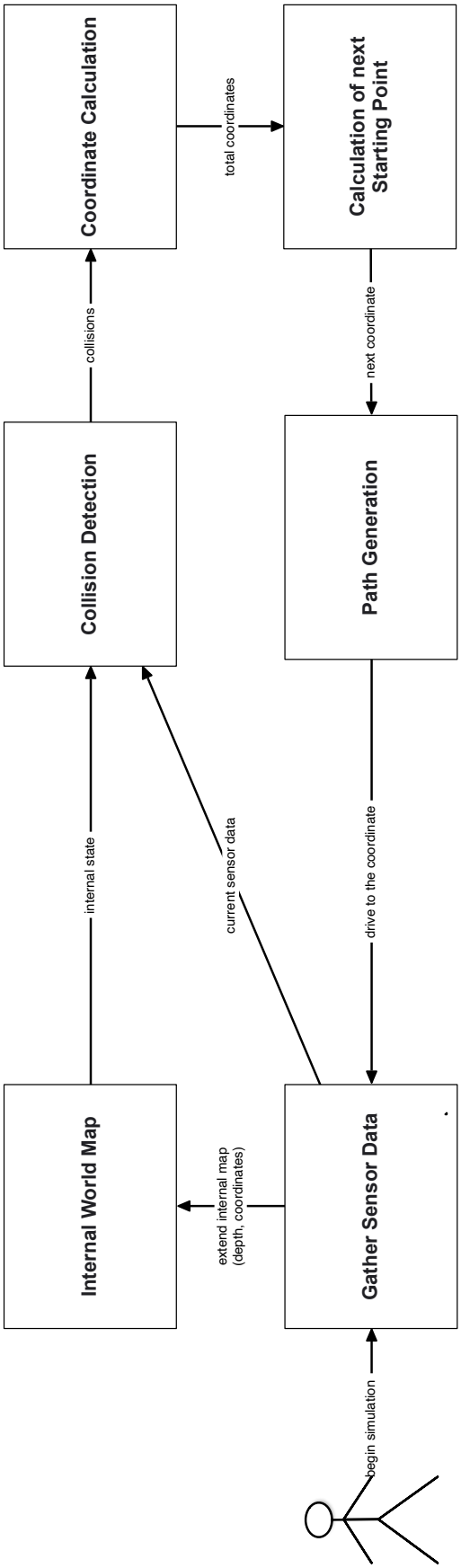
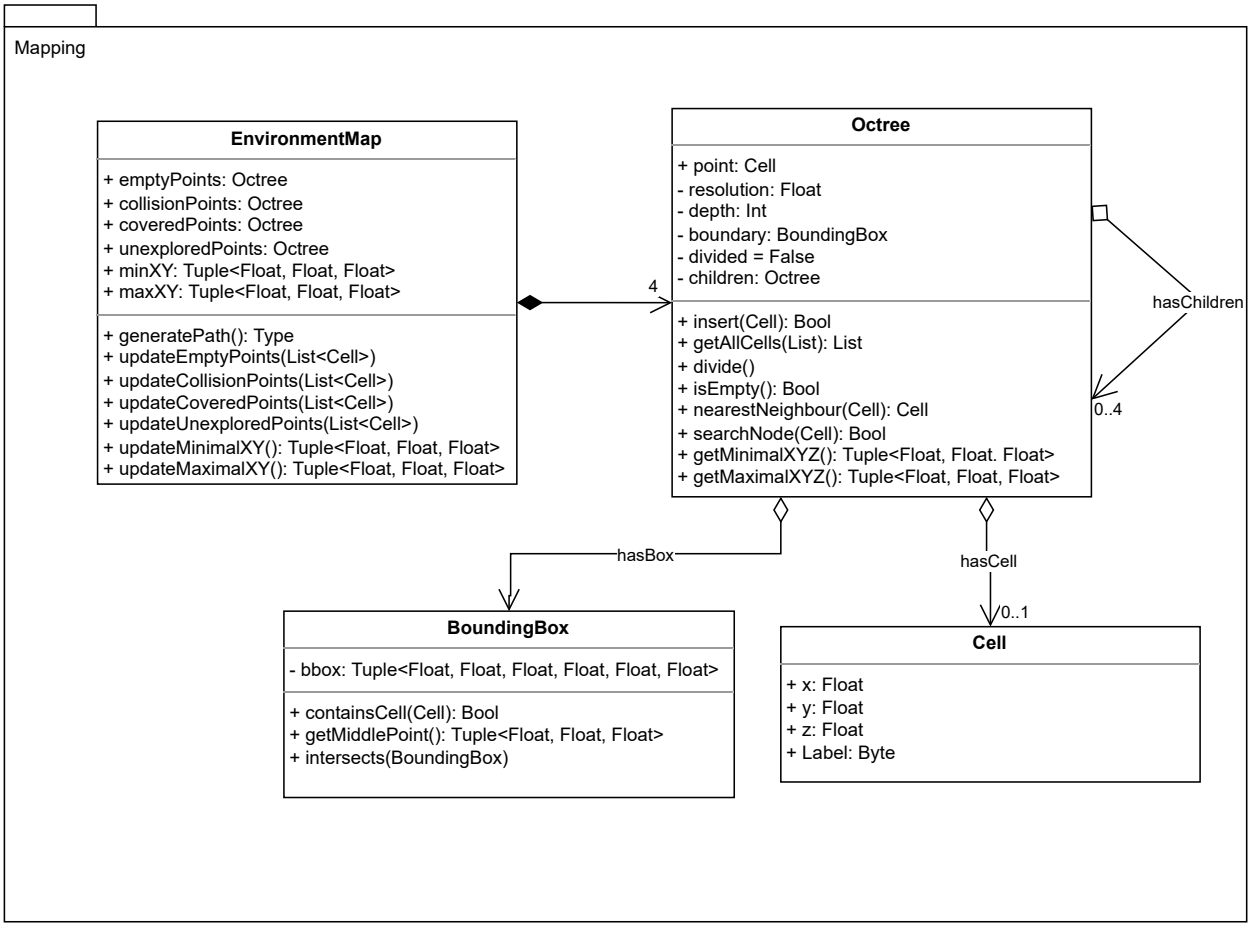
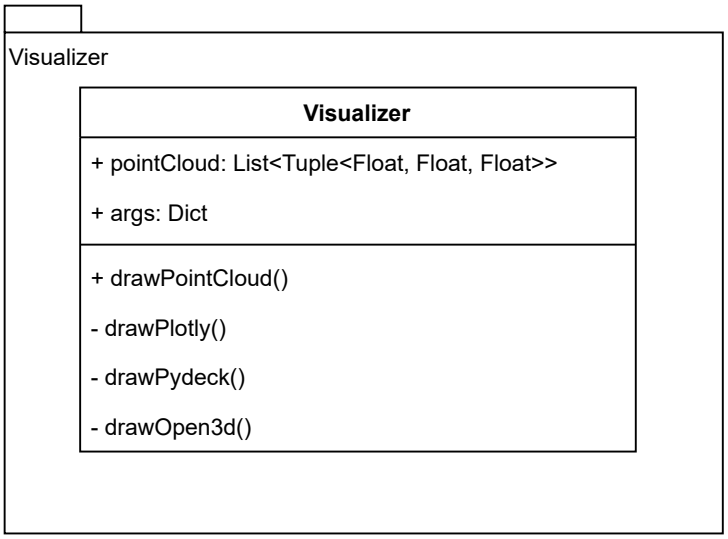


Figure 4: Proposed Architecture



(a) Mapping Package



(b) Visualizer Package

Figure 5: UML Class Diagram

References

[1] Patryk Cieślak. Stonefish: An advanced open-source simula- tion tool designed for marine robotics, with a ros interface. 2019.

[2] Yang Cong. Underwater robot sensing technology: A survey. 2021.

[3] Vidal Eduard, Hernández Juan David, Istenic Klemen, and Carreras Marc. Optimized environment exploration for autonomous underwater vehicles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6409–6416, 2018.

[4] Vidal Eduard, Hernández Juan David, Palomeras Narcís, and Carreras Marc. Online robotic exploration for autonomous underwater vehicles in unstructured environments. In *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*, pages 1–4, 2018.

[5] Vidal Eduard, Palomeras Narcís, Istenič Klemen, Hernández Juan David, and Carreras Marc. Two-dimensional frontier-based viewpoint generation for exploring and mapping underwater environments. *Sensors*, 19(6), 2019.

[6] Franco Hidalgo. Orbslam2 and point cloud processing towards autonomous underwater robot navigation. 4, 2020.

[7] Moon Jiyoun, Bae Sung-Hoon, and Cashmore Michael. Meta reinforcement learning based underwater manipulator control. In *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pages 1473–1476, 2021.

[8] Cheng Peng. Research on the stability motion control method of underwater vehicle driven by data. 4, 2021.

[9] Andres EL-FAKDI SENCIANES. Gradient-based reinforcement learning techniques for underwater robotics behavior learning. 200, 2010.

[10] Cao Xiang, Sun Changyin, and Yan Mingzhong. Target search control of auv in underwater environment with deep reinforcement learning. *IEEE Access*, 7:96549–96559, 2019.

[11] Li Zhaolun Luo Xiaonan. Autonomous underwater vehicles (auvs) path planning based on deep reinforcement learning. pages 1–5, 2021.

[12] Yonghoon Ji Member IEEE Hanwool Woo Member IEEE Yusuke Tamura Member IEEE Hiroshi Tsuchiya Atsushi Yamashita Senior Member IEEE Yusheng Wang Graduate Student Member IEEE and Hajime Asama. Acoustic camera-based pose graph slam for dense 3-d mapping in underwater environments. 19, 2021.