Universität Rostock Traditio et Innovatio

Autonomous Underwater Vehicle Simulation

by

Lanka Veera Vishnu Pavan	: 220200602
Purits Andrei	: 221202242

Reviewer: Dr. -Ing. habil. Peter Danielis Universität Rostock

Abstract

In the last century, people began to consider creating autonomous underwater vehicles. As we all know the underwater sea is 90 percent not studied. And this is often logical, cause of the enormous water pressure and dangerous habits down there. So in 1957, the primary AUV was developed. It gave a start to endless research. Nowadays, people and doing their best to succeed in creating programs that make it easier to figure with AUV. Path planning and obstacle awareness play a significant role in safe travel through obstacles.
So path-planning and obstacle awareness are the main objective of using the Bonn-Motions mobility model for accurate tracing purposes. We choose the Global path planning for a constrained environment.And techniques to avoid safely the stationary obstacles. Taken into account using an algorithm for the considered problem. Energy consumption is the main concern for AUV path planning designers. Keeping in mind energy consumption, designed an algorithm for finding the shortest path for a given constrained area called Dijkstra Algorithm. For a given area shortest path is calculated and the results are presented below.

Contents

List of Figures

1 Introduction

We were to find the best mobility model and implement suitable Path planning for it. The main purpose is to explore the global Path plannig on a constrained environment. In our project we used methods for planning of an AUV from the existing solutions and improve them and simulate this path using the OMNeT++ 3D environment. Our first objective was to find the best mobility model. There are many moving methods described in OMNeT++ tutorials. We have checked them all, and finally chose BonnMotionMobility. The second goal is to choose Path planning method and prove why it is the best. Then we needed to implement it all to the OMNeT++. Afterwards we created a simulation 3D environment with obstacles with OSGearth model support. The respective files of the simulations can be found in the git repository of this project.
Chapter 2 explains all the basic simulation tools on which we perform our simulation. It is also given the basic understanding of our mobility models in the INET framework. In our case it is a BonnMotionMobility. It also explains the idea of how 2D case mobility actually works and the algorithms in a 3D case for path planning of AUV, that we used in our project.
Chapter 3 shows which algorithm for path planning did we choose for our problem and why.
Chapter 4 shows the implementation of our code to the program and the theoretical information about OMNet++.
Chapter 5 explains how we manage to implement obstacles in our environment. And the pros and cons of our mobility in working with obstacles.
Chapter 6, last but not least, explains the idea of implementing our project to real problem with real coordinates, also couples ideas are provided there for someone who is going to improve our project in the nearest future.

2 Fundamentals

This project is based on the mobility models showcased in the INET framework of the OMNet++ environment. This project's mainly aim is to use in-built mobility model with some modifications.

2.1 OMNeT++

OMNeT++ is a modular and component-based C++ simulation library framework mainly used for building network simulations. Firstly we chose OMNeT++ version 6.0, but unfortunately the INET model library, that this version has, is hardly working with 3D environment. We have been trying to solve this problem for 2 weeks, and we were not succeeded. As we chose the main aim for this project to implement the solution for the real 3D case the decision to use the older version was made. In this project the OMNeT++ has 5.7 version. The purpose of using OMNeT++ is to create an environment for the simulation of AUV based along with integrating real-life factors for a more realistic simulation model. In our project we have used a set of Ned files, C++ source codes, INI file to run a simulation and .movements file to create a trace of needed way. It has pre-built functionalities helpful for simulation of wired and wireless communication networks, queueing networks, performance aspects of complex software systems.

2.2 INET Framework:

INET Framework is a model library for the OMNeT++ simulation environment which provides protocols, agents, and other models for researchers and student working with communications networks. It supports a wide class of communication networks, node mobility, advanced visualisation and network emulation which are the main area of interest for this project. The INET 4 package which was used in this project consists of pre-coded simulation environments along with tutorials and showcas

2.3 Mobility Models in INET :

The INET provides a classification of mobility models based based on the type of motion of a node along its path. The selection of a specific model depends on the type of motion. There are many mobility models provided in INET showcases, we have mostly work with Turtle mobility and BonnMotionMobility, because their characteristic perfectly suits this project. At the very end we decided to work only with BonnMotionMobility, cause it is better, when we need to apply precise coordinates of sea border territory. We would mot talk much about other motions, as we focused on only one and the previous group has already provided us the hole list of them. BonnMotion-Mobility is used to trace files made using the BonnMotion scenario generator. These scenarios can also be exported for several network simulators, such as ns-2, ns-3, COOJA and ONE.

3 Path Planning Methods for AUV

AUV path planning refers to planning a secure and feasible path from the initial state (position, attitude) to the target state (position, attitude) under certain evaluation criteria. consistent with the scope of path planning, AUV path planning are often divided into global path planning and local path planning. it's the foremost commonly used classification of path planning. Global path planning obtains all environmental information of the AUV during the whole navigation process prior to to search out the globally optimal path. Local path planning relies on a range of sensors carried by the AUV to gather real-time environmental information (such because the distribution of obstacles) to plan a locally optimal path for obstacle-free navigation.
So our entire projects mainly concentrates on the Global path planning.at the end we have tried out the Local path planning.The methods and details are discussed below.So for path planning the vehicle should follow some algorithm to reach end point or target position.And this algorithm should be effective and optimum so that AUV will reach its target position without any obstacle with less time period.They are several type of algorithm. We carried out our project by Dijkstra Algorithm.
In below presented some of the terminology and parameters used.

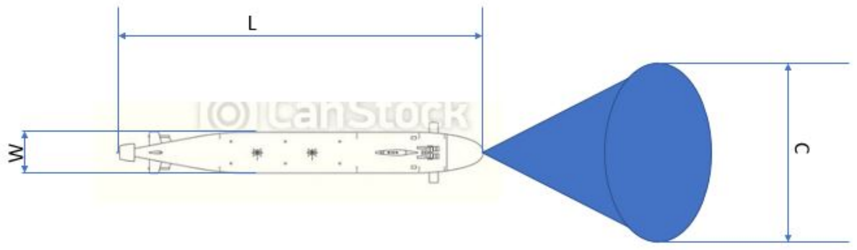


Figure 1: Parameters

L = length of AUV.
C = search range of sensor or Area of sensing range.
W = Width of AUV.
S = Safe distance from obstacle.

3.1 Dijkstra algorithm

Dijkstra algorithm could be a typical global shortest path planning algorithm. It starts from the start line and uses the strategy of the greedy algorithm to traverse the adjacent nodes which are closest to the place to begin and not visited. When it reaches the endpoint, it can find the shortest path from the place to begin to the endpoint. Dijkstra algorithm is principally accustomed solve the single-source shortest path problem in weighted directed or undirected graphs.

Because of limited energy storage for the AUV. It is always recommended to find a path that is very short to reach the target, which makes it more clear that finding the shortest path also plays an important role. In this algorithm the main task is to select the node points that leads to the target node point. How we select the node points makes so much difference ,and we need to keep in mind that in between there are obstacles .So that our AUV should travel between these obstacles in a safe and secure manner.

so in our project we have taken a area with obstacles which is same as realistic. So in this area we have used Dijkstra Algorithm to find out the shortest route.Below figure shows that 3 to 4 ways ,but Dijkstra Algorithm helps to find shortest and secure path.

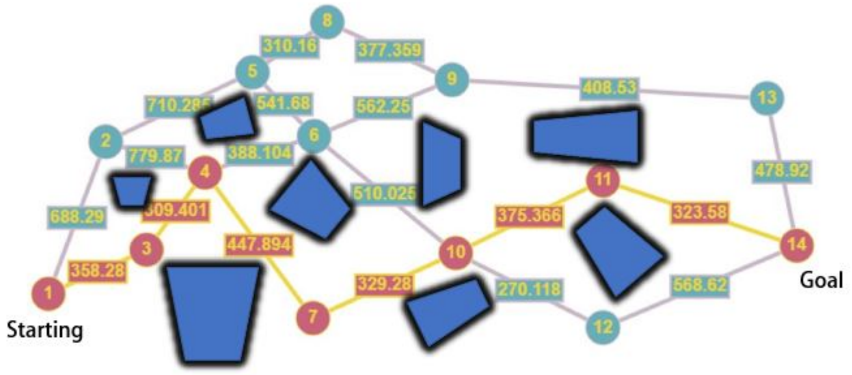


Figure 2: Path planning using Dijkstra Algorithm(software :Onlinegraph.ru)

Above this diagram is drawn in software called Graphonline.ru,one can use this software to calculate different algorithms and sketch accordingly .In above figure one can observe there are several alternatives ways from starting vortex 1 and end goal vortex 14 .The path representing blue lines are the other alternatives ways for AUV, way represented in red shows the path planned by Dijkstra Algorithm.
Selecting the vertices is big task for path planning ,because safe travel of AUV should be considered.So it depends on the sensor search range ,lets consider the search area of AUV as "c".In below diagram its shown that ,in sensor range area .How the safe distance is taken within the account.Path planning

of AUV involves multi-objects optimization. For this, several factors must be considered, which include energy consumption, time period, safety, and smoothness of trajectory. The length of the planned path nearly reflects the energy consumption and time expenditure, and energy consumption is a crucial factor which must be taken into consideration for limited energy carrying AUV. Energy is consumed within the process of navigation, changing flight direction, and diving. Safety is additionally paramount, which is embodied to keep a secure distance far from obstacles (safe margin). Sometimes it's necessary to think about the period of time in an exceedingly certain task.

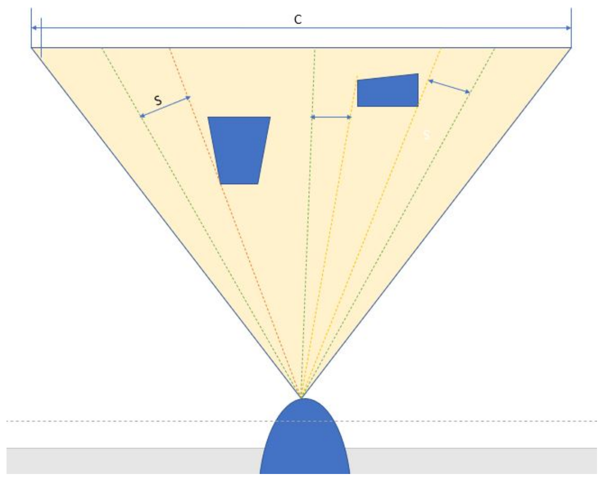


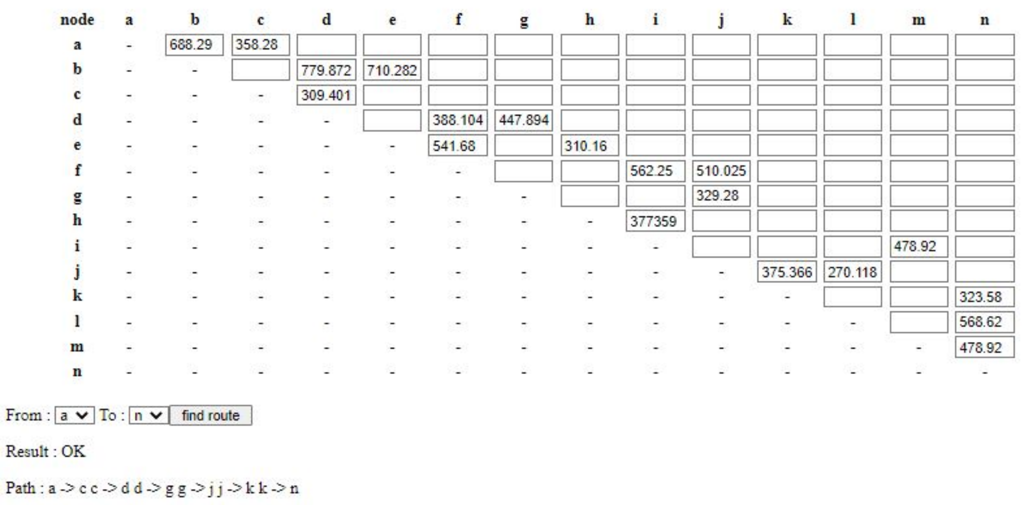
Figure 3: Sensor search area of AUV and avoidance of obstacle

First, select one in every of the sides of obstacle toward the left or right. And take the safe distance into tho account. Consider safe distance "S" which reads in angle degree. differing types of sensors have the various ranges of searching areas. Let's the sensor search area is 120 degrees. .so the safe distance is taken into consideration 30 degrees.that is rotation of direction to wards no obstacles zone of 30 degree.or one may take S in meter .when the obstacle position is know in as 3d point .then it may taken in meter as distance .We have taken safe distance in meters, we have taken 80 percent of width as a safe distance .

3.2 Calculating shortest path using Dijkstra Algorithm

In the above section, we went through how we selected vertex points in between the obstacles in an optimized way. during this section, we are going to discuss .how the shortest path is calculated. One can calculate the shortest manually for the very small number of vertices. But it's not the identical case if there are more numbers. Since it's pre-planned or one can say global path planning, we all know the locations of the obstacles fine. Then after we are going to choose the vertices through the obstacles. we want to attach of these vertices altogether possible ways.

So we've all vertices and therefore the distance between them. The software is named Blogger VIEIRA Emmanuel. Which we'd like to allow vertices, distances, and connections between them.



node	a	b	c	d	e	f	g	h	i	j	k	l	m	n
a	-	688.29	358.28											
b	-	-		779.872	710.282									
c	-	-	-	309.401										
d	-	-	-	-		388.104	447.894							
e	-	-	-	-	-	541.68		310.16						
f	-	-	-	-	-	-			562.25	510.025				
g	-	-	-	-	-	-	-			329.28				
h	-	-	-	-	-	-	-	-	377359					
i	-	-	-	-	-	-	-	-	-				478.92	
j	-	-	-	-	-	-	-	-	-	-	375.366	270.118		
k	-	-	-	-	-	-	-	-	-	-	-			323.58
l	-	-	-	-	-	-	-	-	-	-	-	-		568.62
m	-	-	-	-	-	-	-	-	-	-	-	-	-	478.92
n	-	-	-	-	-	-	-	-	-	-	-	-	-	-

From : a To : n find route

Result : OK

Path : a -> c c -> d d -> g g -> j j -> k k -> n

Figure 4: Calculation of Dijkstra Algorithm using VIEIRA Emmanuel

One can wonder why above the vertices are defined in alphabets.Yes, here we can only define them as alphabets .we just need to replace 1-14 to a-n.

4 Implementation and Case Study

4.1 Basics

OMNeT++ is Objective Modular Network Testbed written with the code language C++. It is a simulation framework with network protocols, which can simulate and perform data. An OMNeT++ model consists of modules that communicate with message passing. The active module code is written in C++ with simulation class library. Messages can contain complex data structures. Simple modules can be grouped into compound modules, the number of hierarchy is not limited.
The user defines the structure of the model in OMNeT++ description language, NED. It supports the structure of the model to be parametrized and supports separating the module interfaces., but OMNeT++ modules are reusable if some care was take writing them. There is also a graphical editor which works directly with NED files.

4.2 BonnMotionMobility

4.2.1 Theory

The BonnMotionMobility mobility model is a mobility model, which means that contains a text file with preplaned motion (t, x, y, [z]), where t is time and x, y, z are coordinates of our AUV. Z is in the square brackets, because this motion could work in both 2D and 3D cases.
An example of implementation BonnMotion to the ini file.

```
**.host[*].mobility.typename = "BonnMotionMobility"
**.host[*].mobility.traceFile = "bonnmotion.movements"
**.host[*].mobility.is3D = false
**.host[*].mobility.nodeId = -1
```

In bonnmotion.movements file you could add the same quantity of lines as many as you want AUV in your simulation.
The time that we chose is conditional. In terms of simulations it is much easier, when AUV completes the route in several seconds. The coordinates in every second of the simulation will be calculated later with some Path Planning algorithm. In our project we use 4 files in total and I would like to talk about each one of them separately a little bit more.

4.2.2 The Ini file

The most important part of our program is .ini file. There we set the value of the constrainted area, in our case it is cuboid 2x2x2 km. We have done this with osgVisualizer, that is attached to some version of INET.

```
.visualizer..mobilityVisualizer.displayPositions = true
.visualizer..mobilityVisualizer.displayOrientations = true
.visualizer..mobilityVisualizer.displayVelocities = true
.visualizer..mobilityVisualizer.displayMovementTrails = true
.visualizer..mobilityVisualizer.animationSpeed = 1
*.physicalEnvironment.config = xmldoc("indoor.xml")
**.constraintAreaMinX = 0m
**.constraintAreaMinY = 0m
**.constraintAreaMinZ = 0m
**.constraintAreaMaxX = 2000m
**.constraintAreaMaxY = 2000m
**.constraintAreaMaxZ = 2000m
**.networkConfiguratorModule = ""
```

Then we can set exact config, in which we are interested in. We set here number of hosts (AUVs) and attached all necessary files together. Also we are applying, if our mobility is provided on 2D or 3D case.

```
[Config Project]
*.numHosts = 3
.host[].osgModel = "3d/glider.osgb.50.scale.0,0,180.rot"
*.host[].mobility.typename = "BonnMotionMobility"
*.host[].mobility.traceFile = "bonnmotion.movements"
*.host[].mobility.is3D = true
*.host[].mobility.nodeId = -1
```

4.2.3 Indoor file

All physical obstacles are described in indoor file. There are many options and variables to set exact shape and position that you need. First of all, you set coordination of its position. Then it comes rotation options. You can rotate an obstacle in all 3 dimensions. There are also 3 shapes, that you could choose: cuboid, sphere and prism. These variety of geometrical forms provide us an opportunity to perform obstacle environment in the way, that we have planned. Afterwards, we choose the size of our obstacle, its color and opacity with texture. You could see an example of the obstacle implementation further.

```
<object position="min 1660 640 0" orientation="90 0 0" shape="cuboid
300 300 1750" material="brick" fill-color="236 236 236" opacity="1" tex-
ture="brick.jpg"/>
```

4.2.4 Showcase file

Here we attached physical environment to our project. With important paramentrs and submoduls.

```
import inet.node.inet.StandardHost;
import inet.visualizer.integrated.IntegratedVisualizer;
import inet.environment.common.PhysicalEnvironment;
parameters:
int numHosts;
@display("bgb=1000,1000");
submodules:
    physicalEnvironment: PhysicalEnvironment
@display("p=580,425");
visualizer: IntegratedVisualizer
@display("p=100,50");
host[numHosts]: StandardHost
@display("i=misc/node$_v s$”);
```

4.2.5 Movements file

Last but not least, .movements file. In our project we use 3 drones, so in the file there are 3 strings with time and coordinates in form (t, x, y, z). From 1 to 30 seconds our drones avoid all obstacles in the shortest way and finish at the end point.

```
1 90 200 0 5 205 430 400 10 670 390 800 15 975 120 1000 20 1200 240 1200
25 1500 540 1400 30 1820 492 1600
1 1580 50 0 5 1890 143 265 10 1890 216 530 15 1890 270 795 20 1890 344
1060 25 1890 408 1320 30 1820 492 1600

    1 1000 2000 0 5 1085 1675 265 10 1170 1350 530 15 1265 1025 795 20 1350
700 1060 25 1585 596 1320 30 1820 492 1600
```

5 Simulation results

We tried our greatest to visualise the simulation environment as like acoustic. The seen of simulation looks like sky blue which is representing water. the form of the node should seem like a submarine or AUV, but it's sort of a drone. The drone represents an AUV. the realm outside of the acoustic zone is constrained.

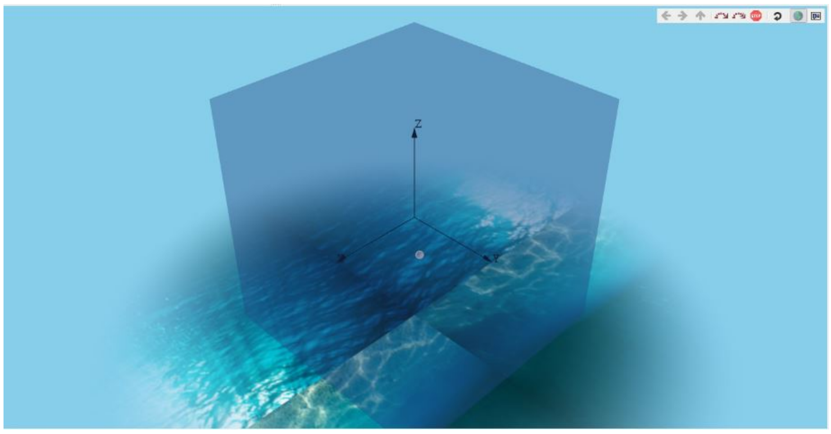


Figure 5: Acoustic simulation background

5.1 Mission planning with one AUV

In the below figure one can observe the trail which is taken by AUV. the path is shown in red colored. the trail which follows is that the shortest path calculated by using the Dijkstra Algorithm. It reaches from one vertex to a different vertex within a time span of what we've got given, because it's following the Bonn Motion mobility Model. We used a time span between the vertex of 5 seconds.

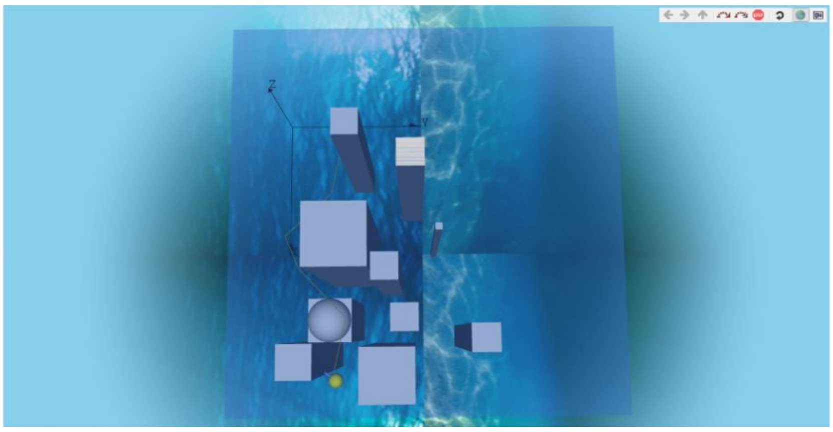


Figure 6: Path planning with one AUV

5.2 Mission planning using multiple AUVs

The figure shows that multiple AUVs take alternative paths. one amongst them is following Dijkstra Algorithm for a higher path. Others are using the subsequent best paths, which started from different starting location. The yellow color path represents the AUV following the Dijkstra Algorithm. The blue color represents the choice paths. From the figure, one can observe that goal position which is representing as yellow sphere.

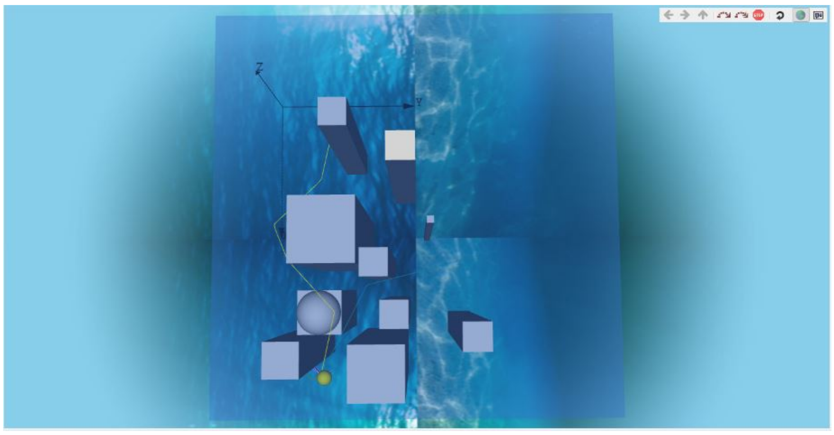


Figure 7: Path planning with multiple AUV

6 Future Areas of Research

6.1 Obstacle Awareness in Real time

The algorithm works fine for finding path for the pre-path planning. But how do we select node points between the obstacles when it is real time simulation when obstacles move in dynamic pattern? The selection of node points we need to make sure that AUV should be safe.

Below are some of the ideas one can do further research.What we have researched all are motioned below.

6.2 Obstacle detection

In the way to path planning in terms of real time. It is not so easy to go through the path.In reality the AUV may experience some static and dynamic obstacles. When the obstacle is static we may use process called pre-path planning. But in the case dynamic obstacles the AUV should be more cautious about the environment. Or else their may be situations that serious damage to AWV. So obstacle awareness places important role in path planning.
While going through designing the algorithm for obstacle detection. We went through some of the own methods for reaching the obstacle awareness.

6.3 Constraint area

The main purpose of constraint area is to define a cubic volume that the node can not leave, so outside of that area we cannot perform any simulations. When the node reaches the boundary of the constraint area, the mobility component should prevent the node to exit.

- reflect of the wall
- reappear at the alternative edge (torus area)
- placed at a randomly chosen position of the realm
- stop the simulation with miscalculation

So using one of this feature "Reflect of the wall".To declare obstacle as constraint small area, so when node reaches to this obstacle eventually to get reflected. But the disadvantage of declaring constraint area, we can declare area to wards outside and most specifically we can constraint only one area. So there is chance by using this concept to find concept of reflection for obstacle.

6.4 Input while run-time simulation

One more alternative is to give a input command while running the simulation, when it reaches the obstacle. The point is here working with the Bonn-motions. So in this direction we tried to find a way to build it. It ended in no solution.one cannot give run-time input commands while simulating using Bonn-motions.But if you can try to build your own mobility model it is possible.It is suggested to using own mobility model for this approach.

6.5 Obstacle detection with many node

In this procedure one need to create so many nodes between obstacles. So in the process of packet exchange, since obstacle stops the radio signals passing through them ,so one need to write an algorithm which path is constructed through the packets exchanging lane .So in below diagram it is green dotted line representing the packets are successfully exchanged between nodes(a,c) and AUV.Red dotted line representing unsuccessful transmission(b,d,e). It is one promising way to get awareness of obstacles.

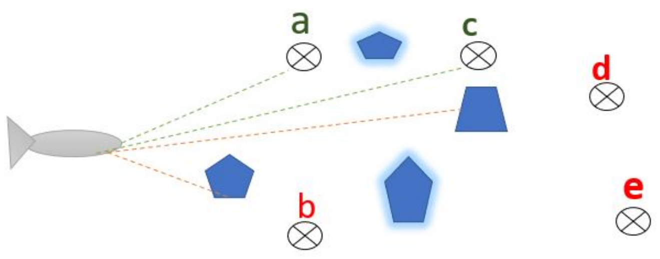


Figure 8: Local path planning using many nodes.

6.6 The design

We have done a lot of work to perform our project in the best way. As the assets for OMNeT++ and not free, we implemented water environment with blue cuboid with half capacity and a picture of the deep sea from the internet. As AUV we have chosen a 3D model of the glider, that left a trail on his way, so it would help to indicate the way, passed by our vehicles, a lot if we are working with several hosts. The obstacles were represented as a special system of cuboids and spheres.

6.7 Urban Environment

It would be more presentable, if a real map of the seaside would be performed in the project.

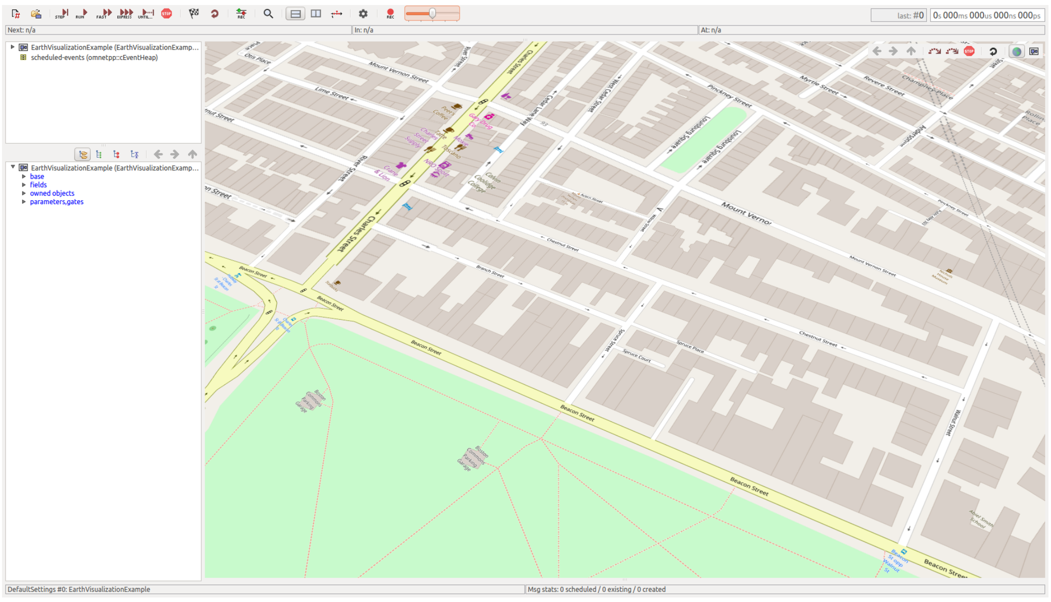


Figure 9: The example of urban environment in OMNeT++

It is often better for simulations to have a real world connection. And there is a way to manage this, there is a possibility to put network nodes on a real world map. This map doesn't effect the simulation, it only defines network's visuals. However, objects can be added to represent water ships, which can be used as obstacles to signal propagation.

6.8 Implementation 3D map to the project

To create a map, the visualizer needs a .earth file. This is also an XML file that tells how the data is turned into a map and how to take data from the Net. There is a possibility to create Rostock's seaside with underwater environment by yourself.
Locations are identified with geographical coordinates, longitude and latitude. In INET there is a module OsgGeographicCoordinateSystem, that is

responsible for converting between Cartesian and geographical coordinates and backwards.

7 Conclusion

Based on the results of the simulations performed on the project we are able to make some basic conclusions.

7.1 Mobility model

We have chosen BonnMotionMobility in our project, cause this Motion Mobility can be used both in 2D and 3D cases and it has path preplanning option that was very useful in this project, Compared to other BonnMotionMobility is accurate.

7.2 Dijkstra Algorithm

One of the most challenges with wireless AUVs is proscribed energy. Keeping energy consumption a vital target. We've calculated the shortest path to achieve the goal position. which the results are very accurate. The minor discrepancy observed is because the from the path using BonnMotionMobility, because XML file does not support floating numbers and hence the results of the coordinates are rounded which results in a minor deviation from the path.

7.3 Implementation

The implementation of our code was successful and it works fine on OMNeT++ version 5.7 and it was presented in Chapter 4.

7.4 Environment and obstacles

We created simulation environment in the shape of a cube with size of 2000m. Also we added obstacles and calculated the shorted path through them. You can see the results in Chapter 5.

7.5 Global path planning

In different varieties of path planning, we focused on Global path planning with static obstacles. Worked on the algorithm for a path called Dijkstra for locating the shortest path keeping in our mind about less energy consumption. The results are pretty promising. We achieved working simulation for it.

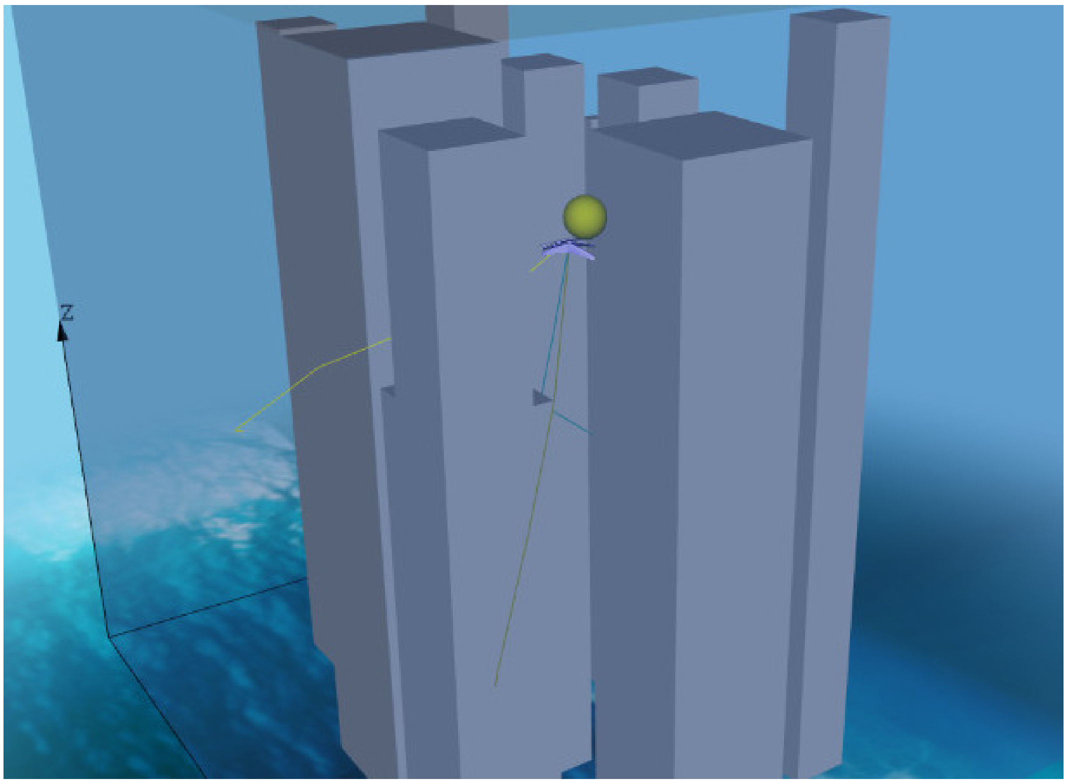


Figure 10: The project final appearance

8 References

[1] OMNeT++ Developer Team, „OMNeT++ Documentation,“ [Online]. Available: https://inet.omnetpp.org/docs/users-guide/index.html.

[2] Manhar R. Dhanak, Nikolaos I. Xiros, Material and Springer Handbook of Ocean Engineering.

[3] OMNeT++ Developer Team, OMNeT++ tutorials: , https://inet.omnetpp.org/docs/showcas

[4] Pranavkumar Ghoghari and Muhammad Azubi Qureshi (2022) Autonomous Underwater Vehicle Simulation.

[5] OMNeT++ Developer Team, https://inet.omnetpp.org/docs/showcases/mobility/basic/doc/

[6] OMNeT++ Developer Team, https://inet.omnetpp.org/docs/showcases/visualizer/osg/ea

[7]Zheping Yan, Jiyun Li,* Yi Wu, and Gengshi Zhang (2018) A Real-Time Path Planning Algorithm for AUV in Unknown Underwater Environment Based on Combining PSO and Waypoint Guidance.

Affidavit

This project was completed by the efforts of 2 participants. Some of the tasks were completed individually but most of them were made collectively by many personal meeting and briefings. The overall efforts made by both participants were equal. We also like to thank Dr.-Ing habit. Peter Daniel's for being involved in this project, helping us to understand our final aim and without whom this project could not being done.
We hereby confirm that we have written the present work independently and have not used any tools other than those indicated but this work is based on the success of the previous group's work. The passages of the work which are taken from the wording or the meaning of other works have been identified with an indication of the source.

Andrei Purits	Vera Vishnu Pavan Lanka
Place, date Signature	Place, date Signature