BİLAL EMRE GİRİT

Bacholar of Science Thesis

PRU

T.C. PÎRÎ REİS UNIVERSITY

ROUTE FOLLOWING NAVIGATION APPLICATION IN AUTONOMOUS SURFACE VESSELS

BİLAL EMRE GİRİT

MAY 2024

A GRADUATION PROJECT REPORT SUBMITTED TO
ENGINEERING FACULTY
OF
PÎRÎ REİS UNIVERSITY

BY

BILAL EMRE GIRIT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF BACHELOR OF SCIENCE

IN
THE DEPARTMENT OF NAVAL ARCHITECTURE AND MARINE ENGINEERING

MAY 2024

Approval of the Engineering Faculty

Prof. Dr. İsmail Hakkı Helvacıoğlu
Dean

I certify that this Graduation Project Report satisfies all the requirements as a Graduation Project Report for the degree of Bachelor of Science.

Dr.Ahmet Ziya Saydam
Head of Department

This is to certify that we have read this Graduation Project Report and that in our opinion it is fully adequate, in scope and quality, as a Graduation Project Report for the degree of Bachelor of Science.

Dr. Şafak Cemal Karakaş
Supervisor

Examining Committee Members (first name belongs to the chairperson of the jury and the second name belongs to supervisor)

(Title and Name)	(PRU, EF)	
(Title and Name)	(Affiliation)	
(Title and Name)	(Affiliation)	

Bilal Emre Girit	....	....
Student’s Name	Signature	Date

Certified by:

....	....	....
Supervisor’s Name	Signature	Date

ABSTRACT

Route Following Navigation Application in Autonomous Surface Vessels

Girit, Bilal Emre
Bachelor of Science, Department of Naval Architecture and Marine Engineering
Supervisor: Dr. Şafak Cemal Karakaş

May 2024,70 pages

The autonomous systems are became so popular in recent years. With this popularity, the autonomy is influenced all the industries that it can affect. The maritime industry is also started affected from the autonomous systems and their innovations. The researchers from globe is trying to make some applications that makes the works easier in this industry with the help of autonomy. In this report i am going to make a project that includes a part of autonomy to the navigation applications in unmanned surface vessels. I am going to use a virtual model of a unmanned surface vehicle in a simulation with the help of Matlab SIMULINK. Also in this simulator going to make a virtual controllers to control the vessel in virtual environment. And also i am going to use some various simulators that includes real dynamics of the sea physics and the other environmental factors detailed. Then i am going to make an approach to making an algorithms that tries to make an route to the destinations most feasible and efficient way with a Python. The algorithm basically going to make a way trough two points that one of them is starting point the other one is destination point with the deep learning of neural networks with the help of the Neuroevolution of Augmenting Topologies. The goal of the algorithm is tring to make a most efficient way to the goal with avoiding the obstacles and various environmental effects that it can faced in real worldwith the help of the deep learning.

Keywords: Autonomy, Navigation,

ÖZ

Otonom Deniz Üstü Aracında Rota Takip Navigasyon Uygulaması

Girit, Bilal Emre
Lisans, Gemi İnşaatı ve Gemi Makineleri Mühendisliği Bölümü
Tez Yöneticisi: Dr. Şafak Cemal Karakaş

Mayıs 2024, 70 sayfa

Otonom sistemler son yıllarda oldukça popüler hale geldi. Bu popülerlik ile otomasyon etkileyebileceği tüm sektörleri etkilemektedir. Denizcilik sektörü de otonom sistemlerden ve getirdiği yeniliklerden etkilenmeye başlamıştır. Dünyanın her yerinden araştırmacılar, otomasyonun yardımıyla bu sektörde işleri kolaylaştıracak bazı uygulamalar yapmaya çalışıyorlar. Bu raporda insansız su üstü gemilerindeki navigasyon uygulamalarında otonmasyon içeren bir proje yapacağım. Matlab SIMULINK yardımıyla insansız bir yüzey aracının sanal modelini simülasyonda kullanacağım. Ayrıca simülatörde gemiyi sanal ortamda kontrol etmek için sanal kontrolörler bulunacak. Ayrıca deniz fiziğinin gerçek dinamiklerini ve diğer çevresel faktörleri ayrıntılı olarak içeren çeşitli simülatörler kullanacağım. Daha sonra Python uygulaması ile varış noktalarına en uygun ve verimli şekilde rota oluşturmaya çalışan bir algoritma oluşturulacak. Algoritma temel olarak Sinir Ağlarının Nöroevrimni baz alan bir Derin Öğrenme algoritması sayesindebiri başlangıç noktası, diğeri varış noktası olmak üzere iki nokta arasında yol oluşturacak. Algoritmanın amacı açık denizde karşılaşabileceği engellerden ve çeşitli çevresel etkilerden kaçınarak hedefe en verimli şekilde ulaşmaya çalışmak olacak ve bunu derin öğrenmenin getirileriyle en verimli şekilde yapmaya çalışacak.

Anahtar Kelimeler: Otonomi, Navigasyon

DEDICATION

To My Parents, to My Country,

ACKNOWLEDGMENTS

The author wishes to express his deepest gratitude to his supervisor Dr. Safak Cemal Karakas for their guidance, advice, criticism, encouragements and insight throughout the research.

This study was supported by the State Planning Organization (DPT) Grant No: BAP-09

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

USV	Unmanned Surface Vessel
AI	Artificial Intelligence
AIS	Automated Identification System
CEP	Circular Error Probability
GSM	Global System for Mobile Communications
DOF	Degrees of Freedom
COLREG	International Regulations for Preventing Collusions at Sea
ITTC	International Ship and Offshore Structures Congress and the International Towing Tank Conference
NEAT	Neuroevolution of Augmenting Topologies

1. INTRODUCTION

Autonomy is a concept that entered our lives in recent years. Especially with the rapidly growing hardware and software abilities of the computers the autonomy is became a technology that we can face in everywhere in common daily life. With that influence power of the autonomy, almost all of the industrial sectors are taking advantage of the autonomy, so it is inevitable to do not taking advantage of this technology in the marine sector. Affecting to the maritime sector is not easy as the other sectors because in this sector tolerance to the mistake is near to zero. So devoloping an algorithm that have control on the ships movements (or any critical assingement) is much more harder than devoloping any algorithm. But the researchers from globe is trying to make some applications that makes the works easier in this industry with the help of autonomy despite all these diffculties. There are several aim of this

There are several reasons that researchers and the engineers are tring the improve the autonomy in maritime industry. The reasons may be seen as not important that much for now but, it will be critically important when the time comes. The most important one of them is safety enhancement. The works about accidents in maritime industry (Acejo, Sampson, Nelson , & Ellis, 2018) shows that the main cause in accidents in maritime is inadaquate risk management. We can easily see that the main cause of the accidenrts in maritime industy is directly related with the human mistake. So we can say that if we can minimize the human factor in this industry, we can reduce the accidents that happens in this industry. The third most cause for the accidents is failure in communication. This subject is also directl related with the human factor. So again, if we can manage the recude the human factor and replace it with the autonomy system there will be visible differences appear in number of accidents in maritime industy.

Exept these safety concerns, this is not the only cause that we are trying to replace with the autonomy in the industry. There is a part of this industry that we can not ignore the effects on the industry. It is financial concerns. The integration of autonomy in the maritime industry yields financial benefits. By reducing reliance on human labor and optimizing operational efficiency, autonomous vessels cut labor costs and enhance resource utilization. Automated route planning minimizes fuel consumption, resulting in significant fuel savings. Moreover, there is no such thing like resting or work shift in system that turned automized, the continuous operation of autonomous vessels without crew rest intervals provides incrementation in productivity.

Also there is a topic that concerns all the people arund the globe. The global warming is the main focus of the all industries in last several years. A report from IMO claims that the international ships around the world are releasing the %2.2 of the total CO_2 release. Automating maritime operations holds promise in combating global warming. By optimizing routes and enhancing fuel efficiency, autonomous vessels reduce greenhouse gas emissions associated with traditional shipping. The precision of algorithmic navigation minimizes fuel consumption and carbon footprint, contributing to a greener and more sustainable maritime industry. Additionally, automated systems enable proactive response to environmental data, allowing vessels to navigate around ecologically sensitive areas. As the maritime sector transitions to automation, it becomes a key player in the global efforts to mitigate climate change, aligning with eco-friendly practices and fostering a more environmentally responsible approach to maritime transportation. So if we can convert the human affects into automized systems properly , we can reduce the global emissions significantly in following years.

In this comprehensive project, the primary focus is on devoloping a navigation algorithm for unmanned surface vessels, incorporating autonomy into their operational framework. For making this alghorithm i made a virtual environmet and a virtual unmanned surface vessel instead of making a model or a scaled model for the test phase fort he algorithm. This was the very first stage of the devoloping an algorithm and its test environment. For that environmental effects we used the algorithm that includes matematical models of the environmental effects. And added this various environmental effects to the our simulation environment Matlab SIMULINK. The simulation is designed to emulate real-world scenarios, incorporating dynamic sea physics and various environmental factors. Within this simulation, we have to get control of our unmanned surface vehicle so we have to implement the controller to the our vehicle. And than we can give the controllers of the vehicle to our algorithm and we can estimate that wheter it is successful or not. A key aspect of this project involves the development of algorithms aimed at optimizing navigation routes. And this is going to be maded with the deep learning neural networks, spesifically Neuroevolution of Augmenting Topologies. With this deep learning tool the vessel is able to pass all the obstacles around them and reach the target. And the algorithm's primary objective is going to find the most efficient route between two selected points: the starting point and the destination. The most crucial issue for the algorithm to intelligently navigate through the open sea, considering obstacles and diverse environmental effects that may be encountered along the way. And the algorithm is going to overcome those problems because it will be improve its performance with making mistake in virtual environment.

This project aims to improve and make a innovative approaches to the unmanned surface vessel technology, not making fully autonomous navigation system that can be work in real-world. The aim is, showing that the combination of the deep learning and the USVs is can be possible way to create a fully autonomous USV. The project combines virtual modeling, simulation, and algorithms to create a realistic and applicible solution for unmanned vessels navigating in open sea environments. By using computer models, simulations, and programs, the project includes that these vessels can operate safely and adapt to unpredictable situations at sea. The goal is to contribute to the development of advanced navigation systems, making unmanned surface vessels more reliable and capable in open sea conditions, for the reasons that i mentioned above, which is crucial for their successful deployment and operation.

2. LITERATURE REVIEW

The study that I will work on is not easy to be understood so I will mention a work that made by Bertram (Bertram, 2004). The study can be acceptable as introduction to the unmanned sea vessels, shows the general overview of automation of the ships. The work mentions about the decrease in crew members in particular work areas in the ship, increase in the computational power in recent years and accordignly we are at the threshold of the transitition between the manned-vessels to unmanned-vessels. This works main point is process of the transition so Bertram suggests several concept of how can we integrate the artificial intelligance to the maritime systems properly with the aspect of superiorities of the artificial intelligance to the humans or human to the artificial intelligance. Also mentions about the possible problems like communication misunderstanding between human and artificial intelligance In summarizing these key points, Bertram constructs a strong introduction to unmanned vessels in the maritime industry. The study not only highlights the progress made by the industry but also anticipates potential issues while proposing innovative approaches for future integration.

When we look at the collusions at the maritime industry we can easily see that the accidents is mainly caused by human mistakes (Atacan, 2022), and the author mentiones that the main reason causes the human mistake is mainly environmental factors. The paper tries to indicate the undercovered causes of the collusions and mainly looks at the fishing vessels. They made an risk analysis using Fine-Kinney methode and they found out that the generally cause of the collusions are caused by limited vision.

There are so many relative works about the topic that I am going to work on in the literature, the topic collusion avoidence is one of them. The researchers from China, (Niu, Zhu, & Zhai, 2023) proposes a Deep Reinforcement Learning alghorithm to solve the multi ship collusion scenarios. The paper compares three current Deep Learning methodes and found out the Double Deep Q Network with Prioritized Experience Replay has a significant advantages to the other alghoritms in the multi ship collusion scenraios. With all these advantages authors mentiones that the collusion avoidence is still is in his infancy. The integration of the artificial intelligence to the maritime sector is still long way to go.

The work of (Theunissen, 2014) is a complete guide to the how can autonomous surface vessels can work at open sea. The reccommendation of his paper is instead of making an fully autonomous ships, proposes to make an highly autonomous system supervised by a human. The idea behind this suggestion comes from the current state of art in the maritime industry. The paper mentiones all the tools that the industry uses are is not enough for a that kind of reliable system.

Some of the researchers (Baldauf, Fischer, Mehdi, & Gluch, 2017) are offered a system to the maritime industry a system like aviation industry An alghoritm that warns the officers if there is probability of a collusion in next several minutes. But the main problem in this topic is the lack of clarity of the limits unlike the aviation sector. Paper offers that there can be global alghorithm that takes inputs with the mandatory devices on the vessel (like AIS, Radar) and makes warnings to officers when there is no chance to escape the collusion. The authors are devoloping an approach to how can vessels feasibly adopt these kind of systems.

The transition between the conventional control systems to the autonomous systems is not going to be easy as it looks like There is a group of people that made an approach, a new idea for this transition process (Wu, Sæter, Hildre, Zhang, & Li, 2022). So it is going to be much more realistic to apply this rather than the direct conversion to the autonomous system. The system works like, there is a autonomous ship model that got ability to steer to the designed destination, but the difference is it has got a connection to the a shore control system. System can be controlled by artifical intellegence or by the human controller in anytime. The ratio of the controllers belongings are scaled into 4 degrees. The one side of the scale there is a human control with AI that gives dsicion support, at the other side of the scale there is no human only AI takes control of the ship. So in any time, human can take charge when its needed. This collabration of the human and AI makes the ship safer. If there's a tricky situation, a person can step in and make decisions. It's like having a smart helper that follows its own path but lets people give directions when it gets tricky. This way, the ship can sail smoothly on its own, but with the option for people to guide it along the way. It's a clever mix of computer help and human teamwork for safer sailing. This whole process can be a great transition between conventional ships to the fully autonomous ships.

Scientists in the maritime field, focusing on making ships run by themselves, believe that fully self-driving ships might happen in the future. But, before that, they need to solve a few challenges. One big challenge is making sure ships can navigate on their own without crashing into things. A researcher (Bahçe, 2019) worked on this. They created a special plan for self-driving ships and tested it in different situations. The plan worked and it made smart decisions and followed safety rules that comes from the COLREG regulations, showing that with more work, fully self-driving ships might become a reality.

There is also another obstacle that we have to overcome. This obstacle is especially about the international marine law that provides safety in sea traffic and researchers of University of Singapore (Karey, 2017) worked about this obstacles and how to avoid and overcome them. The fewness of the crew onboard in the vessel is obvious problem that conflicts the autonomy and the rules. This situation is one of the many conflicts in the study that mentioned above. The study describes the early solution as, two port countries can make agreements on safety with each other for only voyages between only two ports or countries. The following processes for open sea safety, we have seen plenty of situations that need to change for example the international maritime rules are generall puts responsibility on master as you may know, if there are no master in the ship, who or what can provide the This reliability in an unmanned vessel.

If we have had autonomous systems since the modern ship era, autonomous navigation systems could prevent the majority of ship collisions. But also there was too many accidents have prevented by humans with their elasticity of thinking. Researchers thinks that there are many problems going to occur we have not seen before in autonomous ships. For that thinking they sugggesting a safety verification program in that next-gen ships. Researchers proposes a hazard analysis methode that uses STPA (System-Theoretic Process Analysis). Authors (Rokseth, Haugen, & Utne, 2019) making a point that the benefits and why those systems demands these kind of system.

With all the yields of the autonomy, it is really hard to devolop such a system. Researchers and engineers from all over the workd are devoloped the learning algorithms to make things easy when they faced some complicated and many faced problems. The Deep Learning and Artificaial Neural Network is some of them. We can use those kind of applications to use in our context. But if we want to devolop new approach to the literature we have to increase the degree of the autonomy (Ye, Li , Wen, Ruiping, & Vasso, 2023) because all the main first step applications are made by the researcers from all around the world. So yes, there is a huge area that we can work on it, but this needs a great deal of effort from experts and researchers working in the fields of ship autonomy.

A researcher (Bayrak, 2023) tried to make a General Motion Planning Alghoritm with all these knowledge in this subject. The model is running at a simulation environment with some of scenarios. It made a model that primarily takes the COLREGs 13th, 14th and 15th rule. The paper shows that non prioritized navigation can be manufacturable and testable in virtual reality. Paper mentiones that there can be more complex and bigger simulation environments are open to work in future and present.

As mentioned earlier in the part of introduction, we are going to test our new approach on a virtual environment and we want it to show behaviours of the motion as close as possible to the real-life. Before creating the virtual environment of the tests, I want search some works that done about this topic. Firstly, there are expensive, commercial ship simulator applications that generally focused on the ship controls and movements for training and education. Those applications are great for sailors but not suitable for us. We are looking for the ones that more like ship motion prediction system. One of the work like that is done by researchers from University of Colombo (Sandaruwan, Kodikara, Keppitiyagama, & Rosa, 2010). They have used 6 degrees of freedom motions with C++ with Simulink for estimate the motions of the ship and they reached encouraging outcome. They have resulted in, little improvements in performance is provides more realistic results than the expected for some reasons so they are mentioned the parameters that future works can improve about this topic for the realistic results. They emphasized the critical significance of all six degrees of freedom motions, warns against a focus on only some primary motions such as surge and roll.

3. MATERIALS AND METHODES

In order to test and actualize these studies, we need to implement them on a surface vessel. Furthermore, this vessel should possess specific capabilities related with the requirements of the algorithms. These functionalities are comes from the equipment that we will install on our surface vessel. This integration is crucial for the successful execution of the experiments, as the vessel's performance will be directly influenced by those special equipments designed to meet the algorithmic needs. The careful selection and installation of these tools on the surface vessel will play a significant role in achieving accurate and meaningful results in our research. So in the next parts I am going to go in details about equipments, what are their abilities and why we especially chose them for our unmanned surface vehicle.

The main part of this project is the vessel. Without a vessel that can carry all the sensors, equipment, and the computer that runs the main algorithm that we are devolped, nothing else matters. The size of our unmanned vessel is going to be a small because no one is going to be on it, and if we make it bigger, it needs more propulsive power to make it move at same speed. There's also something about this vessel that helps to reduces the propulsive power need. It's hull type is like a catamaran with two parts, this gives us more space on top without making the boat heavier and without increasing the displacement volume. So, we have a catamaran boat that's 2 meters long and 1 meter wide. This is the boat we'll use for all our tests and the rest of the procedure.

3.1 Equipments

There is a sevearal equipments that we are going to implement in our vessel but first I would like to talk about the general layout of our unmanned surface vehicle. If you look at our components diagram you can easily see that there is a one main part that of our vessel that all the data sensors and all the communication modules are connected to it, and also It will run our main algorithm that we have devoloped than it will command the propulsion thrusters of the vessel. This part is going to be a brain of our vessel and all the thing is going to happen inside of it.

3.1.1 Main processor

The selected processing unit for this project is the Nvidia Jetson Nano. This processor is optimized at gathering, analyzing, and utilizing data obtained from connected sensors and modules. The decision of choosing this model is becaose of in its comprehensive support for various sensors and seamless communication capabilities with other modules. Essentially as a compact computer, the Nvidia Jetson Nano has got processing capabilities sufficient for executing the navigation algorithm, all while maintaining a less size and energy-efficient operational costs. The choice of this processor aligns with the project's objectives, as it provides compatibility with different sensors and proceeds effective communication among modules, ultimately contributing to the successful implementation of the unmanned surface vehicle's navigation system.

3.1.2 Thrusters

Focusing on the propulsion system, our unmanned surface vehicle is equipped with the two Blue Robotics T500 thrusters, with waterproof design and ensuring reliable operation in aquatic environments. Specifically chosen for their performance, these thrusters play crucial role in executing commands generated by our algorithm and central processor. The Blue Robotics T500 thrusters has got enough specifications, providing 1900 revolutions per minute and a thrust capability of 16.1 kilograms. This performance-based design not only applies precise and responsive control for navigating the planned course but also aligns with the requirements of our devoloped navigation algorithm. The incorporation of these waterproof thrusters is crucial, offering a reliable mechanical propulsion system for our unmanned surface vehicle to interact effectively with sea environment with less complexity.

3.1.3 Lidar sensor

For avoiding the environmental obstacles or anything to close vessel, it have to determine its own surroundings.The Rock Robotics R3 Pro lidar sensor, employs laser beams to measure distances and create detailed real-time map at any instant, so with that way it can determine its surroundings with this module. This sensor enhances navigation accuracy by providing real-time data, enabling the vehicle to make informed decisions and navigate safely in its surroundings.

3.1.4 AIS transponder

The Garmin AIS 800 transponder, a key component in our unmanned surface vehicle. This device enhances maritime safety by exchanging information with other vessels in general maritime usage. Operating on the Automatic Identification System (AIS), it broadcasts our vessel's position, speed, and course, while simultaneously receiving data from nearby vessels. This real-time exchange enables enhanced situational awareness, reducing the risk of collisions and contributing to safer navigation on waterways. With all these capabilities of this module, we are going to use the data that came from it. With an AIS transponder we can be aware of the other vessels that sensors can not see. This is the biggest reason of why we put this transponder in our vessel.

3.1.5 Accelerometer

DFrobot 6-axis accelerometer is a critical element in our unmanned surface vehicle. This accelerometer measures acceleration in 3 directions and angle values in 3 dimension, providing essential data on the vessel's basic movements. By detecting changes in speed and direction, it contributes to precise navigation, stability, and control. Integrating the accelerometer enhances our vessel's responsiveness and ability to being aware of itselfs movements. It is the one of the crucial components of the vessel because you cannot estimate the position status of the vessel while it is operating

3.1.6 GPS

The Garmin MSC 10 GPS module is one of crucial components in our unmanned surface vehicle. Operating with remarkable precision, this module employs a CEP (Circular Error Probability) value of less than 1 meter, ensures accurate positioning. With a 10 kHz update rate, it consistently provides up-to-date location information. By integrating this GPS module, our vessel gains reliable and real-time GPS data, enhancing overall navigation accuracy for optimal performance during autonomous operations.

3.1.7 GSM module

The WaveShare SIM7600G-H GSM module is a important component for communication in our unmanned surface vehicle. This module enables communication by utilizing the Global System for Mobile Communications (GSM) technology. It allows the vessel to establish a connection with mobile networks, facilitating data transfer and communication. This is particularly valuable for transmitting critical information, receiving commands remotely, and ensuring connectivity in areas with mobile network coverage. This GSM module with its capabilities, contributes to the overall communication system of our vessel, enhancing its versatility and allowing for effective operation and control from a remote location, or getting data from anywhere that got connection

3.1.8 Camera

The integration of the Intel RealSense D435i camera is prpvides unmanned surface vessel visual perception and environmental awareness. This camera captures depth and RGB imagery, enabling the vessel to see and be aware of its surroundings. With advanced depth sensing technology, it provides simplicity in object detection and obstacle avoidance.. This visual input is vital for image process and autonomous navigation, allowing the vessel to make informed decisions based on real-time visual information. The Intel RealSense D435i significantly enhances the situational awareness and overall capabilities of our unmanned vessel, ensuring safe and efficient operation in various environments.

3.1.9 Echo sounder

The Blue Robotics Ping2 echosounder is in our unmanned vessel is essential for underwater mapping and depth measurement. This technology enables our vessel to navigate safely in varying water depths, it measures the depth of the water in real-time.This helps to learn about the underwater and avoid the unexpected uınderwater danger.This model was choosed from many option because the Ping2 echosounder's 100-meter range ensures comprehensive sub-surface exploration, contributing to the vessel's ability to operate effectively in diverse aquatic environments, with the small size and with the affordable energy consumption.

3.2 Methodes

In our study, we aim to evaluate our project in a virtual environment To achieve results closely resembling real-world conditions, we employ precise mathematical models and methods. So if we are want it to give the most close results real environment, we have to use methodes that gives the most realistic and reliable results. The utilization of these methods enhances the accuracy and reliability of our findings in the virtual environment. In this project we have several mathematical methodes and models that we are going to use for various models. The mathematical methodes that we are going to use in this project differs at this point. The parts that we are going to apply consist of: model of behaviours of the vessel at the sea, environmental forces that comes from sea and the methode that we are going to create for the algorithm.

3.2.1 Mathematical model of the vessel

For creating the model of the vessel in a virtual environment used a fundamental mathematical models for my unmanned surface vessel (USV). Those models and methodes are provides us to ensure the status of the vessel at a spesific instance. This is important because the we must estimate the vessels actual behaviours in a virtual environment with that models.

First, specifically focused of its location with six degrees of freedom (DOF): surge, sway, heave, roll, pitch, and yaw. Through basic equations, this model takes into account hydrodynamic forces and external factors, provides that how the USV behaves in response to various conditions. The point is to refine the vessels design and control mechanisms, ensuring optimal performance in virtual scenarios. This mathematical foundation is going to used for predicting and enhancing the USV's movements in 3D space.

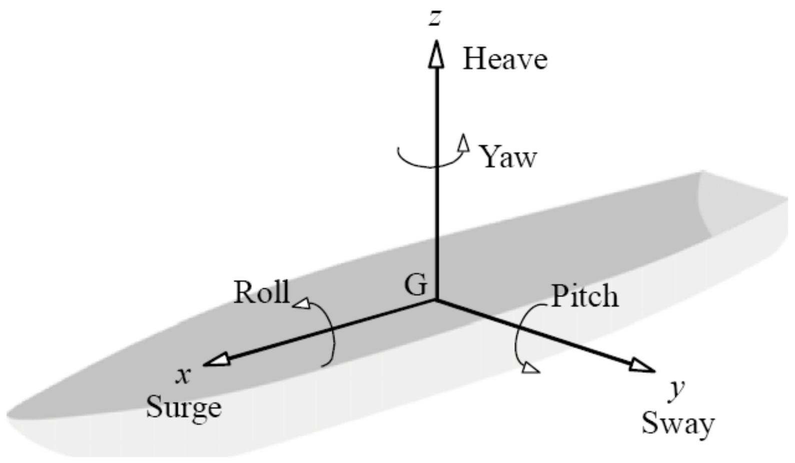


***Figure 1**: 6 DOFs in our vessel*

In addition to examining the various movements of the vessel, it's crucial to consider the orientation of the craft in related to its surroundings. It's not acceptable for the vessel's orientation to remain fixed while it's in motion. To fix this, we will employ Euler's Angle Transformation at some points. This method allows us to understand and represent the vessel's orientation as it moves through its course. Euler's Angle Transformation is a valuable tool that helps us account for the dynamic changes in the vessel's position and alignment, ensuring a more accurate representation of its movement in response to environmental factors. By incorporating this transformation, we aim to enhance the precision of our analysis and better capture the dynamics of the vessel in motion.

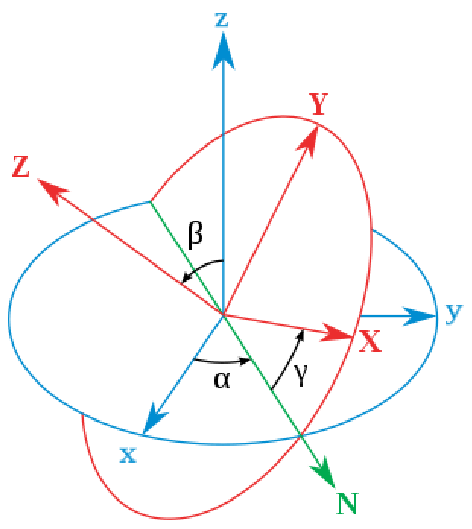


Figure 2: Euler Angle Transformation

In this approach the main dynamic of the methode is having three rotational axis at a one singular point, it is handy if you are not dealing with multiple movements. But if you are going to face a situation that requires multiple rotations at the same time the Euler Angle Transformation can stuck with the spesific problem called “gimbal lock”.Gimbal Lock is a main problem of this transformation methode it is mainly caused by the rotations that giong to happen after the spesific rotations or conditions.You can deeply anlyse this methode and the problem of it in the article that published in World Academy of Science (Aye, 2011) So we are going to use an additional methode that colled Unit Quanternions at some points in our study that represents the rotational movements of the vessel smoother than the Euler Transformation.

Unit Quanternions are basically a mathematical methode uses imaginary numbers that generally used for representing the rotational movements in three-dimensional space same as Euler methode. The advantages of it it is capable of estimating a rotation of an any point in any axis. The form of a quanterion is like Equation 1 where *a, b, c ,d* are real numbers and *i, j, k* are imaginary numbers of the part of the notation. For applying the rotation we are using we need Unit Quanternion as shown in Equation 2, where θ is the angle of rotation and *x*, y, *z* are the components of the axis of rotation. For finding the desired position we must multiply the quanternion and the unit quanternion with the quanternion multiplication. With that way we are getting the value of the location that rotation about axis that we are defined.

$$q = a + bi + cj + dk \tag{1}$$

$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)(xi + yj\ + zk) \tag{2}$$

In our virtual study, we will be also using hydrostatic equations and calculations to figure out how a boat behaves on water. This is really important because we want our virtual vessel to respond realistically to different situations. Our vessel can experience various forces, and how it reacts depends on its shape. If you push a container ship and a catamaran with the same force, they will not respond in the same way. Because they have different underwater hull types. So, we need to make our virtual world understand these differences to make the simulation more accurate by using these hydrostatic equations, we can estimate the real-world forces acting on the boat more realisticly. This makes our simulation more realistic, just like how actual ships respond differently to currents based on their hull types and their differences in hydrostatic values. The differences in hull design is changes the seakeeping behaviors of the vessel completely, and this is what we want basically.

Until now, we are trying to estimate the forces and responses of the vessel while it is stationarty, this part is about the motion how the vessel responds with a propulsion or a while it is in a maneuver. It is crucial because some points, there are nothing to analyze in this project while the vessel is stationary and the performance of the algorithm that we are going to make is depends directly on our vessels maneuvering capabilities. There are 3 different

Maneuvering models that seperated by speeds of the vessels. Low-speed, moderate-speed and high-speed. In this study we are going to choose the model of moderate-speed for our virtual model. And also there is another situation that I have to mention now, we are providing maneuver without any rudder unlike the majority of the sea vessels. We are providing all the maneuver by using two thuster motors that located seperately on the aft of the ship. So we will use a different mathematical method for maneuvering part.

3.2.2 Mathematical model of the virtual environment

The forces that makes our virtual environment more realistic are mainly cames from the environmental forces. Those forces seperates some parts and those are wind forces, wave forces and current forces. The effect of those forces to the vessel is not negligible so that we are going to add all these forces to the virtual environment and we will analyze the affects of those forces to the vessel and our algorithm.

First, we are going to analyze the wind forces. We have to mention that we are going to look at wind forces among the three separate axises but especially the axises about the x and y axis because effect of the z axis is negligible compare to the other axises. There are some coefficients and variables that we are going to use that effects all the calculations one of them is wind coefficient, it is generally usable for the vessels that symetrical to that axis. Another variables that we need to know is frontal projected area and lateral projected area. It is the two dimensional area of the vessel that above the waterline. One of them is side area called lateral projected area shown as A_{LW}, and the other one that obtained from the head called frontal projected area and shown as A_{FW}. With these variables we can estiamate the effect of the forces that act on the vessel in our virtual environment.

The next environmental effect in our virtual environment is wave forces. It is also effects our vessel and algorithm significanty, so again we can not ignore these forces. For making an environment with wheater conditions, we have to describe the sea conditions. The variables wave height, wavee period, peak period are directly related with the sea state conditionn. Beside these conditions and variables we have to select a wave spectra to use and for obtaining the most realistic waves in our simulation. The ITTC (International ship and offshore structures congress and the international towing tank conference) recommends to use the model of modified version of Pierson-Moskowitz spectrum. With this model we can estimate the most realistic responses of the vessel under the wave conditions that we have described.

And the last environmental effect in our virtual environment is current forces. In the realm of virtual sea environments, understanding the effect of current forces is fundamental. Current forces, often caused by ocean currents, play a crucial role in shaping the dynamics of the virtual sea. These forces mimic the real-world flow of water, influencing the movement and behavior of vessel with the simulated environment.

3.2.3. Subsystems of algorithm

Our primary goal is to achieve autonomy in our system, an algorithm to guide our model's movements. Instead of investing our time and energy into creating one comprehensive autonomous system, we can benefit from dividing it into smaller components. This approach enables us to use our resources efficiently. Besides, breaking down the system into manageable pieces facilitates understanding, increases upgradeability for future projects, and simplifies adaptation. By employing this strategy, we can focus our efforts, making the development process more simple. This modular approach not only optimizes our time and energy but also provides that our system is scalable and adaptable, ready to meet evolving requirements in time to time. Also, this method promotes efficiency, flexibility, makes a strong foundation for the development of autonomous systems tailored to our specific needs and objectives. So in that algorithm we are going to use some various small components and we are going to call them as subsystems in simulink environment. In this part of the thesis the working principles, inputs, outputs and objectives of the subsysyems will be intoduce deeply. Those subsytems are basically; the Angle Finder that finds the angle between two points, Location Control that checks if the vessel is close to the destination, Autopilot System that turns the vessels heading angle to the desired angle, Target Heading that makes vessel go to the desired position and PID controllers.

3.2.3.1. Subsystem Angle Finder

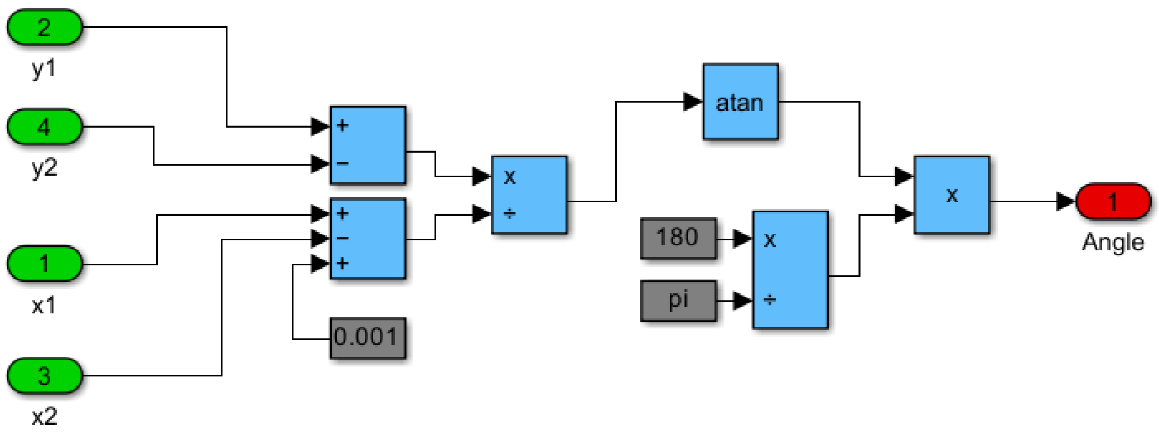


Figure 3: Schematic of subsystem angle finder

The subsytem Angle Finder's purpose is to find the angle between any two points in coordinate system. System inputs 4 data, coordinates of two position as x1 and y1 for first position and x2 and y2 as second position. There is one output that gives angle between two position.

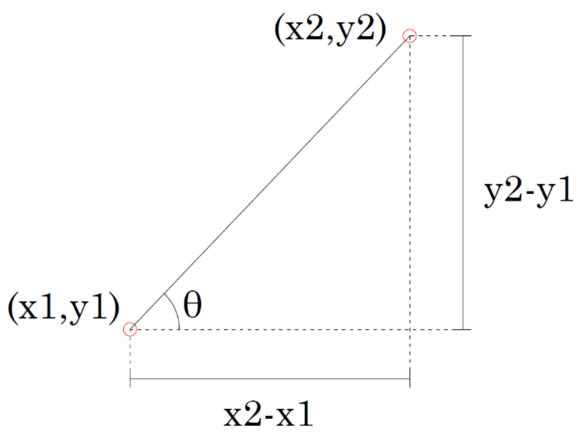


Figure 4: Coordinates and degrees

$$\arctan\left(\frac{(y2-y1)}{(x2-x1)}\right) \times \frac{180}{\pi} = \theta \qquad (3)$$

The working principle is very simple, first we are substract the values of the same coordinates than, the ratio of these two value gives us a result. If we put that result inside of a arctan() function we will get the value of the degree in radians. Than, if we multiply our value with the constant 180/π we will get the degree value for between two coordinates. This is going to be the general equation of this subsystem (3).Lastly to mention, there is a constant 0.001 that enters the system that provides the divisor by dividing with the zero at simulation start t = 0.

3.2.3.2. Subsystem Location Control

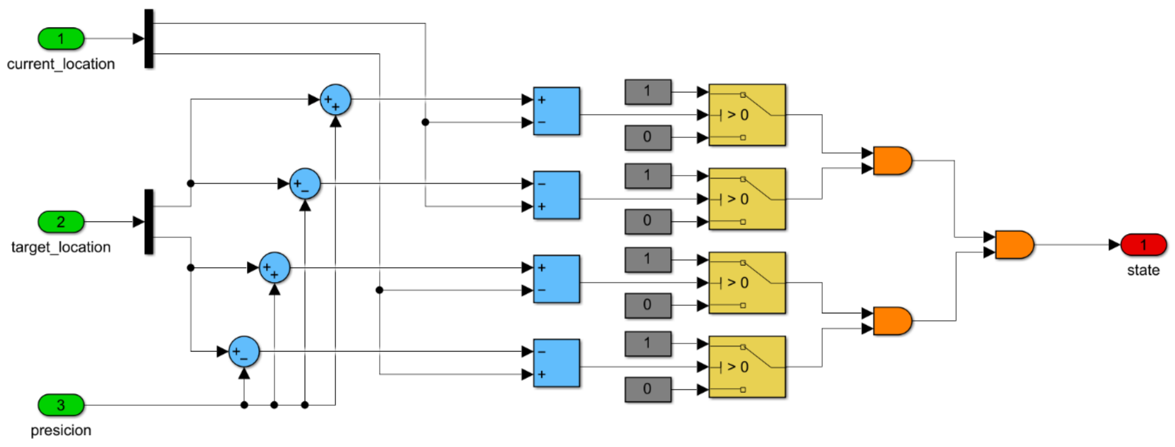


Figure 5: Schematic of subsystem location control

The subsystem Location Control basically checks if the current location is close to the target location with a spesific presicion value. This subsystem inputs 3 data, one of them is Current Location that gives data of current coordinates of the vessel, Target Location gives the coordinate values of the target location and final input is presicion, this one is adjusts the area of the target location.

The working principle of this system is based on the 2 inequalities for each x-axis and y-axis totally 4 inequalities works in the system. For example in x-axis, system compares three values; target location x-axis minus presicion, current location x-axis, target location x-axis plus presicion (4)(5). If the current location is between these two value signals extends from this part gives 1 otherwise system always gives 0. And this process is same for the y-axis (6)(7) and finally the signals are getting into AND logical operator. That means if the two axises are inside of the target area that defined with the presicion the system gives 1 signal to the output, unless there is a 0 signal from any section.

$$TargetLocation_x - Presicion < CurrentLocation_x \tag{4}$$

$$CurrentLocation_x < TargetLocation_x + Presicion \tag{5}$$

$$TargetLocation_y - Presicion < CurrentLocation_y \tag{6}$$

$$CurrentLocation_y < TargetLocation_y + Presicion \tag{7}$$

3.2.3.3. Subsystem PID controller

The main objective of our thesis is to move the vessel in desired location autonomously. For moving it, we have to control the power of the electrical motors so we need controllers. As usual we are going to use PID controllers in our system

A PID controller is a feedback control loop used in systems to maintain a desired setpoint. It consists of three terms: Proportional, Integral, and Derivative. The proportional term responds to the current error, the integral term eliminates steady-state error over time, and the derivative term dampens the system's response to prevent extraordinary conditions. And multiplayer constants for each term. By adjusting these constants, a PID controller can effectively minimizes oscillations and errors. This control mechanism ensuring stable and precise control in processes motion control.

We need to use PID controllers for our motor controls because sometimes our algorithm can give osscilated and sudden commands despite all the tests. This could be a big problem because motors can get damaged if they're given too many rapid changes in direction. PID controllers also help smooth out these commands, making sure the motors get steady instructions that won't harm them. It's important to keep our motors safe and running smoothly, and PID controllers help us do that.

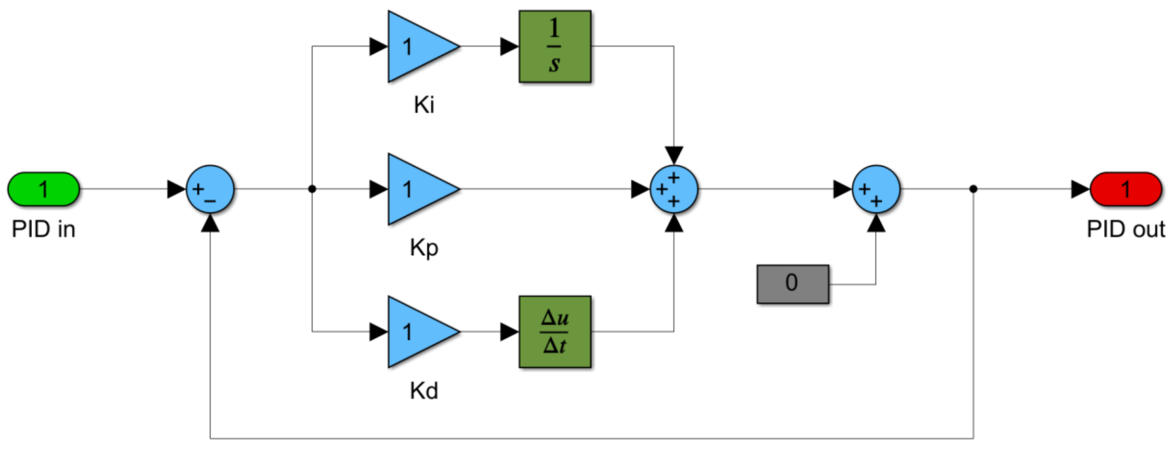


Figure 6: Schematic of subsystem PID controller

So, if we explain it again through the schematic. There is one input and output. System inputs the raw motor commands not smoothed. It goes trough 3 parts Ki, Kp and Kd and goes integrator and derivative block and sums all three value. Then it returns the error value to the starting point for correction. Finally process end with the output of the smoothed value.

3.2.3.4. Subsystem AutoPilot system

This subsystem is one of the most critical parts in our project because it allows us to keep our heading angle turn at desired point. And also it keeps the vessels heading angle still unless the input is changed.

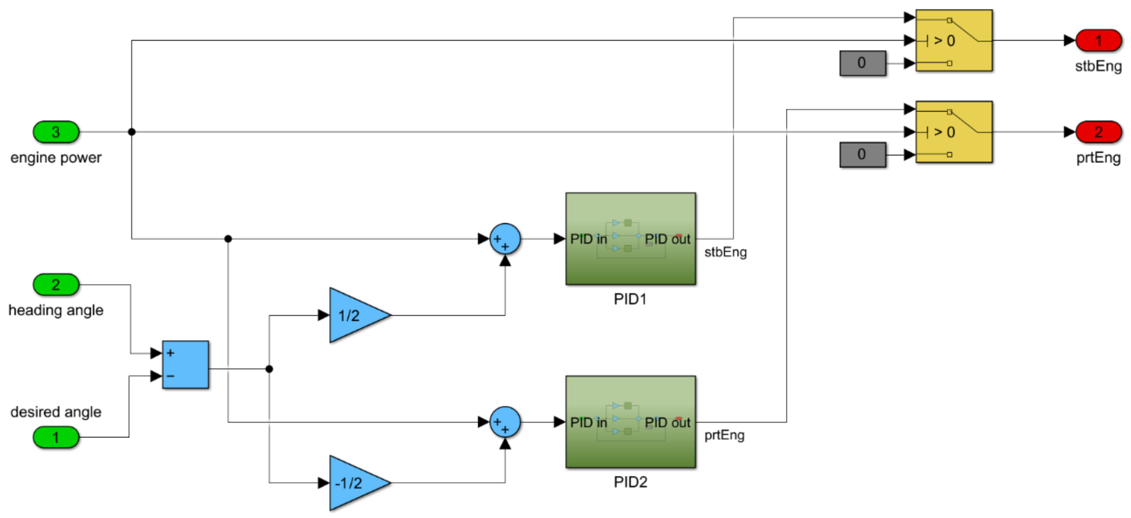


Figure 7: Schematic of Subsystem AutoPilot

This subsystem outputs 2 values that correspond to engine contol at starboard and port. And the inputs are: heading angle, current heading angle of the vessel; desired angle, desired heading angle value; engine power, the base value of our engine power. System basically substracts the two angle values heading angle and desired angle. And multiplies it with the 0.5 and -0.5 then it adds those two values to each motors. That way there is a power difference appears between two engines and that turns the vessel. Turns it until the difference between the two angle gets to zero. And system also tries to keep the value of the angle difference to the zero with the correction system that includes. And finally there is a limiter that prevents the engine values get negative values.

3.2.3.5. Subsystem Target Switcher

In this subsystem, the aim is changing the target if the vessel is inside the target position. The system indicates its position with the state variable. As I introduced before, state is variable that gets the 1 value when the vessel is close enough to the target position if else it gets 0, with that way we can indicate the position of the vessel. This subsystem basically choses the the next target if the state variable gets the value of 1.

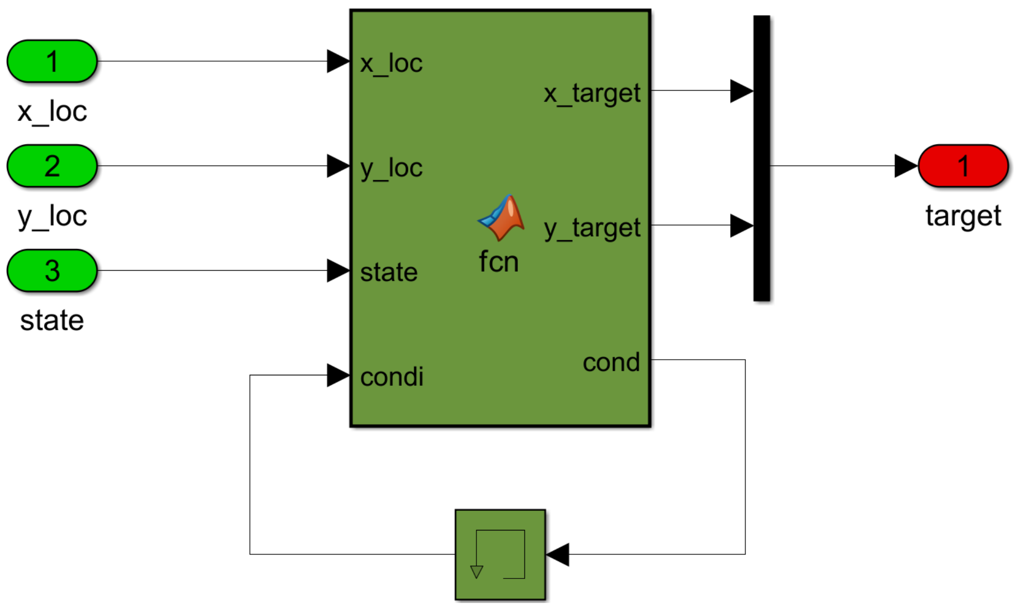


Figure 8: Schematic of subsystem target switcher

In this subsystem I prefer to use MatlabFunction block to make the algorithm the code. The main reason of this chose is the difference between input and output variables of the subsystem. It can also made with the blocks and other functions of the matlab but it is going to be much harder to analyze the situation. Also there is Memory block at the bottom of the schematic, I do not want to reset the value of the cond, this block provides the keeps the value of the variable cond stands same in the whole process. Our input values are x_loc, y_loc and state. State provides the position information as I explained before, x_loc and y_loc is the list of variables (8)(9) of coordinate points of the way that the vessel going to head.

$$x_{loc} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, \cdots x_n) \tag{8}$$

$$y_{loc} = (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, \cdots x_n) \tag{9}$$

The outputs of the block are x_target and y_target as shown in schematic, these two values are joining together and makes a list of one single coordinate point. And this coordinate point updates itself in time.

```
function [x_target,y_target,cond] = fcn(x_loc,y_loc,state,condi)

persistent order
cond   = condi;

if isempty(order)
    order = 0;
end

x_target = x_loc(order+1);
y_target = y_loc(order+1);

if state == 0
    cond = 0
end

if cond == 0 || cond ==1
    if state == 1
        cond = cond + 1
    end
end

if cond == 1
    order = order + 1;
    x_target = x_loc(order+1);
    y_target = y_loc(order+1);
    cond = cond + 1;
end

condi = cond;
```

As you can see in the code, you can easily determine the inputs and the outputs of the block in the first row, x_target, y_target and cond variables are going to be outputs of the block, x_loc, y_loc, state and condi variables are going to be inputs of the block. Next, we define order and cond variables, for later usages. Assigned the cond variable to output of the system as you can see, with that way the value of the cond variable does not changes whenever we call the function. Then we assign the value of the order variable, the difference of this definition process is variable is defined as persistent variable, if the variable is defined as persistent variable in Matlab it is not possible to attain its value in first definition row, That is the reason we used the codes between 6 and 8, so, we are checking that if the order variable is empty, if it is the value 0 is assinges to it. The whole process is gives an error at the beginning of the system without this code part. Then, we are defining our first x and y coordinate point with the help of order variable. Next, we are assinging zero value to the cond variable while the state variable is equals to zero. We are running this process to be sure that the cond variable is not changes until we notice a difference in state variable. Otherwise the cond variable can change in a instance that should not change and this causes to the vessel the pass to the next target without reaching the current target. And the rest of the code is looks for change in the state variable, when it changes the value of the condition increases one and that increasement runs the last code part. The last code part passes the x_target and y_target variables to the next x_loc and y_loc values and lastly we are increasing cond value one again to prevent the break the code part and this makes increase x_loc and y_loc values for only one time for one change in varible state. So with that way we can easily passing to the next target position with the help of the state and the order variable. And we are inserting two ordered x_loc and y_loc variable into one mux, finally it gives us the listed target position coordinates (10) until the next time the input variable state turns to the 1 value.

$$target = (x_{order}, y_{order}) \tag{10}$$

3.2.3.6. Subsystem Otter Usv

This part is the final and most crucial subsystem of the entire simulation. Sytem basically gives the behaviours of the vessel with the changeable inputs. The main part of the subsystem is created by the guides of the Thor I. Fossen and his work about hydrodynamics of the marine crafts (Fossen, 2011). The virtual model is based on the hydrodynamic equations that expalined thoroughly in this book. There is a subsystem that based on those equations and we are going to call them as core subsystem of the vessel

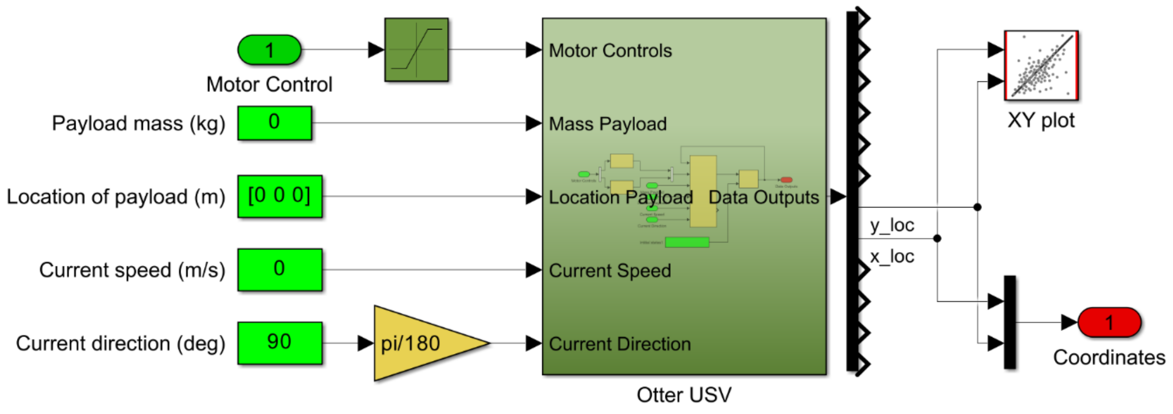


Figure 9: Schematic of subsystem otter USV

The unmanned vessel that we are going to export to the virtual environment is a vessel that produced and existed by the corparation Maritime Robotics, to briefly introduce, the vessel is a basicly catamaran type marine craft. It got length of about 2 meters, width about 1 meters and got 2 propellers to control and there are bunch of sensors on it. The sensor that we are mainly going to use is GPS sensor that gives the current coordinates of the vessel.

In the schematic you can easily determine the inputs and the outputs of the vessel. In the output part we are going to use only current x and y coordinates. In the real world those datas are coming from the sensors that located on the vessel, in this virtual environment we can easily accsess all the datas as the output of the subsystem. For the input part we got bunch of datas to explain. Input datas can be divided into two parts, constant inputs and variable inputs. For variable inputs we have only two data that listed into one as Motor Control. This data structure contains two data, one of them is power of the electrical motor at the starbord side and the other data is the power of the electrical motor at the port side. This two variable is listed together as one data and enters to the core of the subsystem. And these inputs are limited to the minimum -120 radyan/seconds to maximum +120 radyan/seconds with the help of the saturation block of the Matlab. Those maximum and minimum values are determined with the maximum ground speed of the vessel as 6 nautical miles as given in the information sheets by the Maritime Robotics. For the constant inputs we have 4 seperated constants, those are: payload mass, location of payload, current speed and current direction. The payload mass is initial payload mass in the vessel, it can affect the whole behaviour of the vessel significantly, but we got no need to add extra weight to the vessel so we accepted the payload mass as 0. And also coordinates are x is 0, coordinate y is 0 and coordinate z is 0. Other inputs are current speed and current direction. Initial speed of the vessel is equal to zero because we want vessel stand still in beginning of the simulation. The last input is current direction and it defines the initial orientation of the vessel. This value is given 90 degrees, the vessel is looking to the north of the virtual environment in initialization, and the core subsystem accepts the degree as radians so we are converting it with the multiplying it with the constant Π/180.

This subsystem is generally gives the behaviours of the unmanned vessels under the effects of the virtual environment and the motor commands. In the general layout of the algorithm the we are giving the commands and getting output as x and y coordinates from this subsytem. So that makes this subsystem the most crucial, the most critical one.

3.2.4. General layout of algorithm

The general layout of the algorithm is consists of subsystems that explained in detail above. All the system works in the Matlab Simulink environment with the simulation pacing in 1 second.This pacing setting makes simulator much slower and more realistic to the observer.While the simulation runs, The x and y coordinates are plotting as a result simultaneusly, so it gets much easier to remark any mistakes or possible problems.

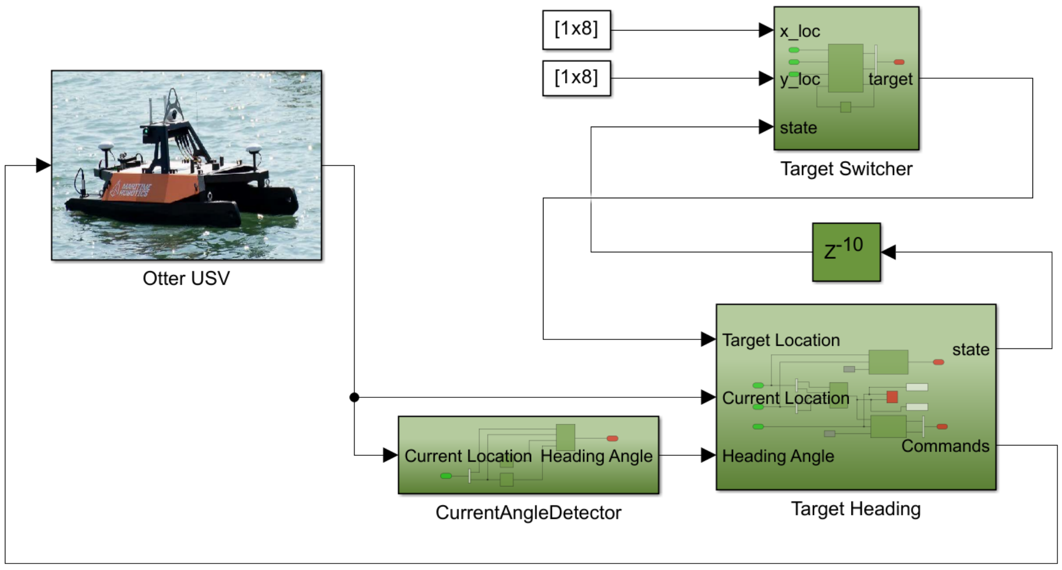


Figure 10: General schematic of algorithm

All the subsystems are connected or inside of each other as shown in schematic (Figure 9). So general algorithms working principal is relying on the presicion and the coordination of all these subsystems with each other.

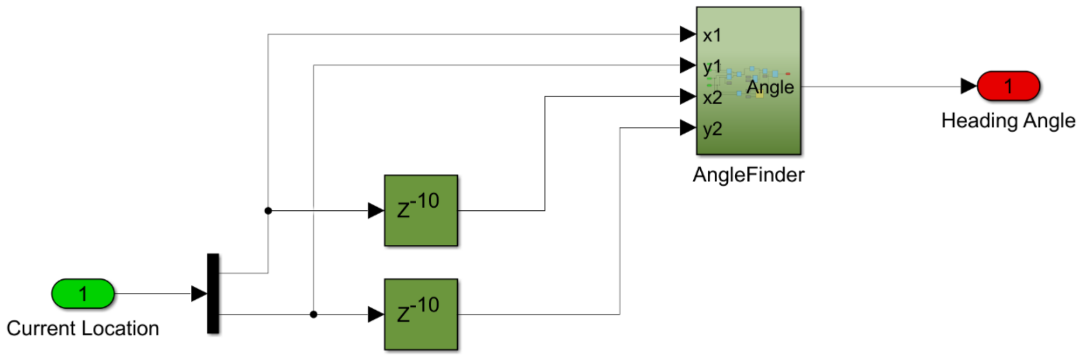


Figure 11: Subsystem current angle detector

To explain the whole process thoroughly, The Otter USV subsystem passes the coordination data of the vessel to the Current Angle dedector. The subsystem of Current Angle Detector is basically calculates the current heading angle of the vessel by using the subsystem Angle Finder that explained before. This subsystem gets input of the coordinate data and it passes that data to Angle Finder as point one, for point two it uses the coordinates of the vessel about a second ago with the help of the delay block and passes it to Angle Finder as second point. Angle Finder gets input of these two points to calculate the angle of the vessel. So with that procedures this subsystem (Figure10) determines the heading direction of the vessel and outputs it for the further usages.

So we have got coordinates of the vessel, and heading angle of the vessel so far. And the next process is Target Heading subsystem as you can see in the Figure 10. But before that we have to determine the location of the target position. The process of the determination of the target position is explained in the subsystem Target Switcher. It passes the current target coordinate with the inputs of the ordered target x and y values. These three values current coordinates, current heading angle and current target coordinate are inputs of the Target Heading Subsystem.

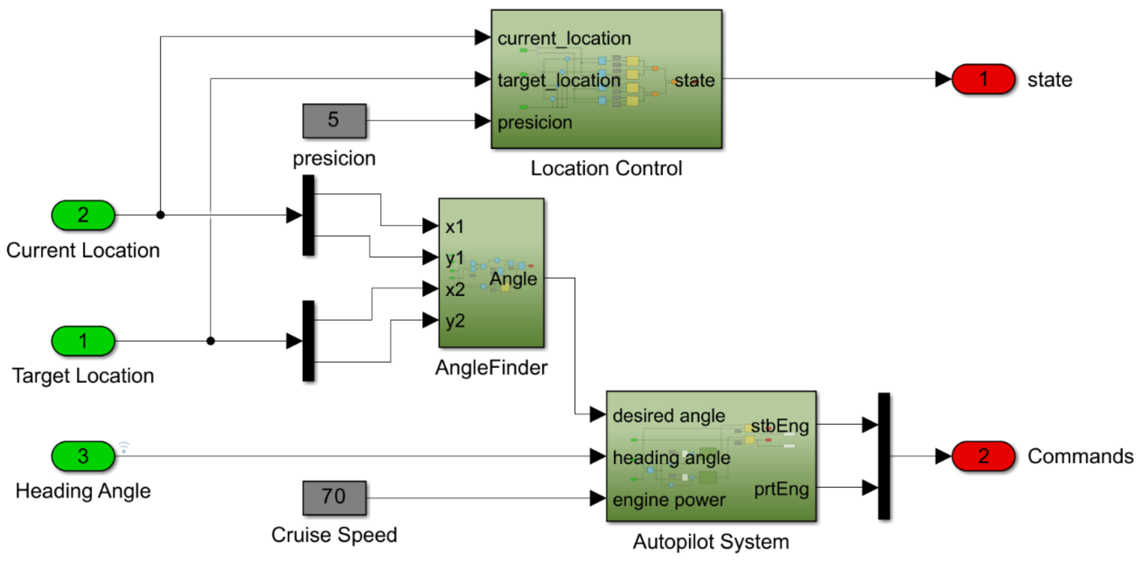


Figure 12: Subsystem target heading

In this subsystem (Figure 11), the first process is determination of the angle between current location and target location. This angle value is going to be called as our target angle value. The goal of this subsystem is to make the vessels heading angle as close as possible to our target angle value that we found recently. And it makes it with the subsystem Autopilot system, how the system work is also explained before. This subsystem inputs the desired heading angle, current heading angle and cruise speed and returns output as motor commands. This motor commands that system gave, changes the heading angle of the vessel step by step until the target heading angle. When the heading angle and target heading angle is equal or close enough the vessel is going to be headed to the target location. After a period of time it is obvious that the vessel is going to reach to the target coordinates.For further processes Target Heading subsystem uses Location Control subsystem to determine the vessels current location. The subsystem Location Control is explained in previous parts, it inputs the current location and target location and outputs the state variable. If the vessel is close enough to the target location the state value becomes one, if it is not close enough, state value returns zero value as output. With this Target Heading subsystem we got motor commands and checked if it is on to the target position.

After those processes we can look to the general algorithm(Figure 9), we are passing the motor commands to our unmanned surface vessel contionuously. So we can say that the vessel is able current go to a target point for now. When it arrives to the target point, the value of the state variable is becomes 1. And that makes change in the Target Switcher subsystem, the system changes the target position. With the new target location, all the values inside the Target Header changes and also motor commands. The same process that explained before is going to happen inside of the Target Header subsystem, until the vessel is gets close enough to the target position. And that loop runs until there is no target locations left to give to the vessel. There is a small detail we should not miss, there is a delay block between Target Header and Target Swithcer, this delay block prevents the error of algebric loop for the system and also it mandatory for clean initialization of the simulation.

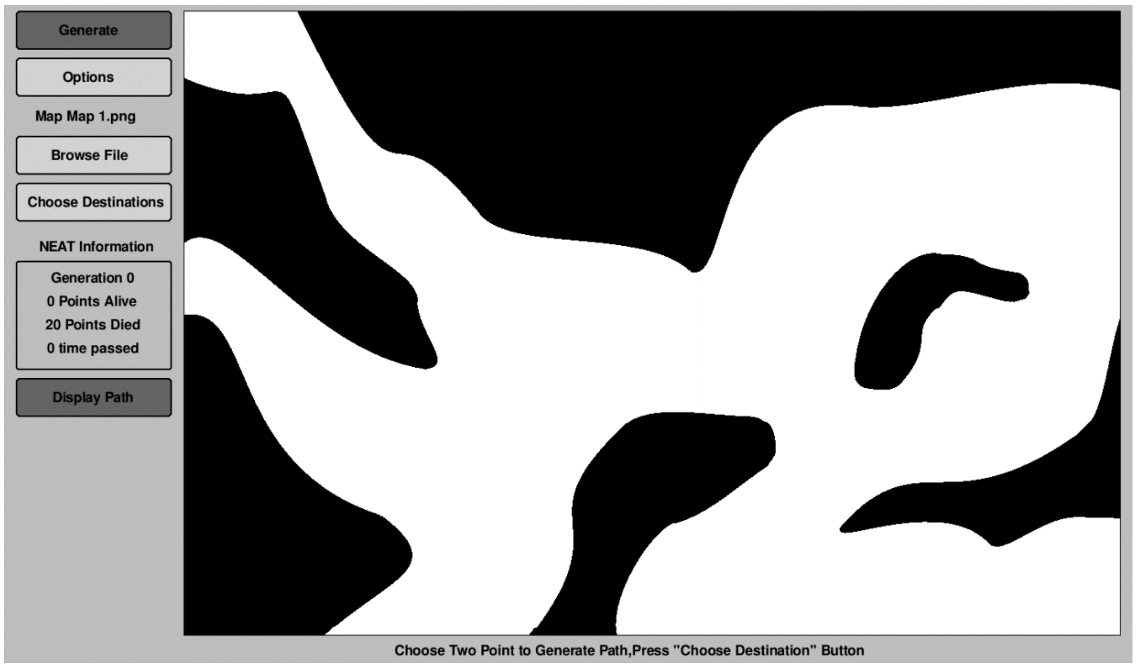
3.2.5. Path finder algorithm

As explained in previous sections, the whole algorithm in the simulink needs some coordinate values as input to start the whole process. In this part we are going to see how the system generates the coordinate values for the simulation with the application. The algorithm is totally seperate python application from the simulink. The reason behind that separation is, capacity of the simulink compared to the python. In further stages simulink can limitate progress of path planning, this is the reason why we using separate program
In this part, the application is going to be explained in detail but before, explanation of the main layout of the application is mandatory for preventing misunderstanding in next steps. The application is writed on Python coding language and based on Pygame library. This

library is tool in python that helps the people to make 2D video games, it provides tools to create graphics and control them. But in this project we are going to use it to the generation of virtual coordinate values. The foundation of the program is build with the pygame library, next is for the generation process, in this part we are going to use Python NEAT (Neuroevolution of Augmenting Topologies) that devoloped by researchers (McIntyre). It is a python library that provides users to train own artificial neural networks for their aims.It basically starts with basic connections and improves it with the responds of the system gradually. The path planning algorithm is mainly uses those two libraries of python. We will introduce it in detail below.

3.2.5.1. General layout of application

The application is based on pygame foundation as mentioned before. The main part of the application that we are going to focus is the map section. All the calculations and proceses are going to happen in this section. And the left side of the application, buttons and other informations are placed.



Map Map 1.png

Generation 0
0 Points Alive
20 Points Died
0 time passed

Choose Two Point to Generate Path,Press "Choose Destination" Button

Figure 13: General layout of the application

The map section of the application is consists adjustable image file, user can change the map that they are going to use with the Browse File button and insert to program a customized map. Choosen map has to be size of 1200x800 pixel and must have .png extension.

Other part of the system is points in the application. This points are represents the our unmanned surfacce vessel in the application. Each point that user defined has got some functions and variables that program is going to use. Thoose values are: current coordinates of the point, target coordinates of the point, absolute distance to the target coordinates,radar data of the point that calculates distances to obstacles for each direction. Functions for each point are: function that checks that the point is crashed or not, moves to the point to 4 directions. Now we must know what does some of those variables and functions really provide for the next steps.

The current coordinates of the point gives the current position coordinates of the point and this coordinates are variable. And target coordinates are the coordinates of the target position and those values are constant, user can not change the target position after the first definition of the point. Distance to the target point is a variable that gives us the absolute distance value of the current position and the target position. The mathematic behind this variable is pythagorean theorem. First difference of two x coordinates and y coordinates are obtaining first, then the difference of those two values second power enters to a square root and this calculation gives us the absolute distance of two point (11).

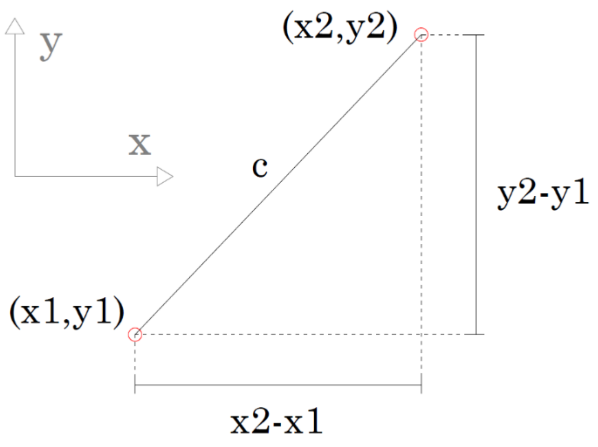


Figure 14: Figure of distance calculation

$$c_{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{11}$$

Another variable is radar variable that calculates the distance of the obstacles in evert 45 degree direction. This variable is gives us as a list of 8 different length, each index of a list gives the distance to a obstacles at degrees of 0° , 45°, 90°, 135°, 180°, 225°, 270° and 315°. each line start at the center of the point and ends when it reaches to the obstacle. If it does not encounter any obstacle until the 300 pixels line ends up there and does not goes further. And each length data is listed in order to degrees in a list (12). For instance the radar data of the point in Figure 14 is shown in the equation 13. The data starts by the 0 length of the 0 degree (north side) then it goes on with next length value of 45 degrees in clockwise.

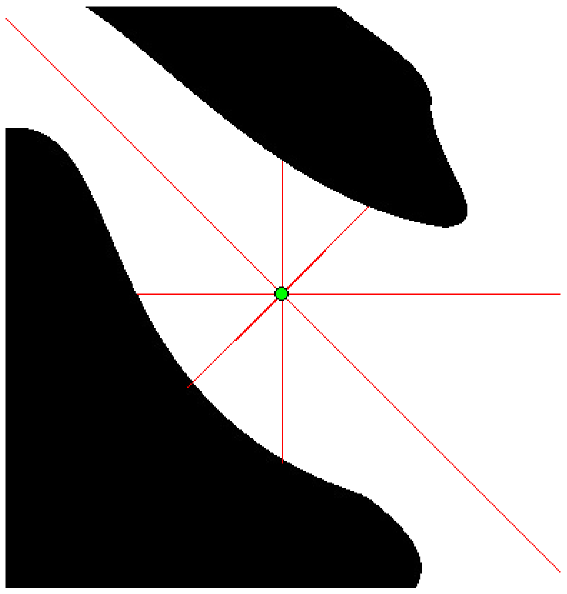


Figure 15: Radar data of each point

$$data = [\ell_{0°}, \ell_{45°}, \ell_{90°}, \ell_{135°}, \ell_{180°}, \ell_{225°}, \ell_{270°}, \ell_{315°}] \tag{12}$$

$$data_{fig14} = [\,94\,,87\,,300\,,300\,,119\,,94\,,102\,,275\,] \tag{13}$$

And the other function is move function. This function provides moving capabilities to each point. Each point can move to 4 directions, those are 0 degree, 90 degree, 180 degree and 270 degree. The function responds the input of numbers that 0 to 4. If the user enters 0 value to the function the point stands still. Other input values like 1,2,3 and 4 is moves the point to the directions in 90 degrees. the working principle of the function is, a value of amount of movement is added or substracted from the x or y coordinates of the system from the input of the system. For example user defined a movement amount and passes the input of 1 to the move function. The function, adds the value of amount of movement to the y coordinate and updates the points location. With that way point moves to the new position with the input of the user.

The last data that we have to understand for the next step is checking the point is crashed to any obstacle or not. The program determines it with the radar functions, if any obstacle is closer than the radius of the point or equals to it, that means the point is crashed to some obstacle. And after that time that individual point is not drawing to the screen anymore and program defines it as a dead point for the next phases. With those datas and knowledge about the points the processes of path planning algorithm is going to be much understandable.

3.2.5.2. Path finding with NEAT

Neat system is a tool based on of the artificial neural networks that used in reinforcement learning tasks that by devoloped by researchers from Texas (Kenneth O. Stanley). In the application of path following, generation of the path is based on this python extension. Let's start with basic introduction to the overall system, NEAT is a python library that helps users to create customized deep learning neural networks and tries to improve the performance of the neural network gradually. There is a input and output nodes in the neural network and the neural network responds randomly to the output nodes with the change in inputs, the improvement are made by some parameters that returned from functions called as activation function and fitness function. With the responds from the output nodes and the parameters from fitness function, the neural network tries to evolve itself to the best version.

The general working principle of the system is, the system creates a number of object that wants to improve its performance, in our example it is going to be points. All the objects has got seperate neural networks, and randomly generated neural connections between input nodes and the output nodes. And each object performs with the guide of therir randomly generated neural networks until a period of time or the extinction. When the time is over or all the objects goes to extinction all the objects that defined beginning of the program acquires a fitness value. The calculation of the fitness value is made by the fitness fucntion defined at the early beginning of the system. A number of objects that got better fitness value compared to the others are selecting by the system. And the neural networks of those selected objects are used for the generate the next generation of neural networks. So system evolves a number of neural networks from the objects that got better fitness value. And the new neural networks that generated from the best of the previous generation starts to run again. And again each object performs with the guide of its neural network. But this time the average fitness values of the objects are going to improve significantly. In this new generation of, the objects got new fitness values and system selects the best ones again. And this process runs over and over again to find the best solution to the problem. This is how the neural networks improves themselves gradually. From now on we will look how I implemented this system to in our context.

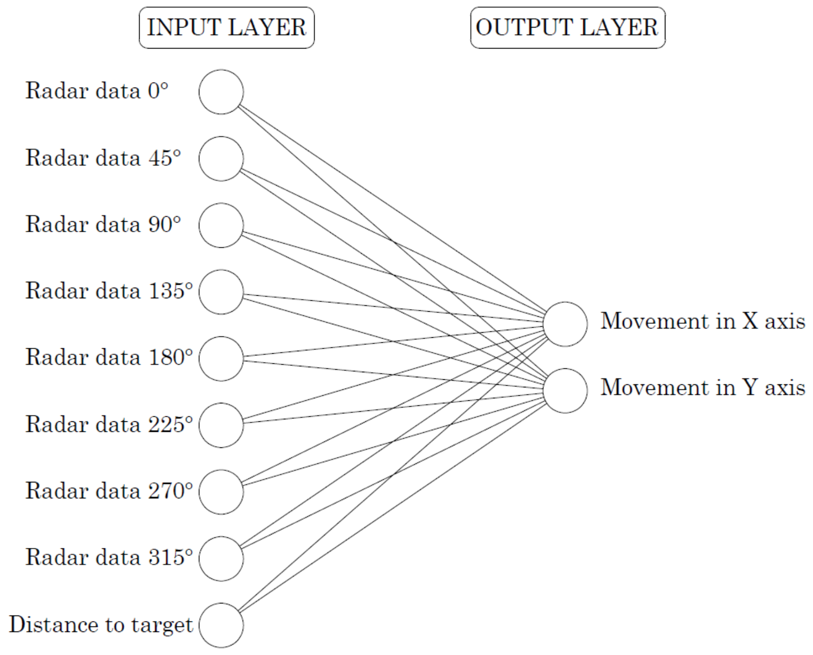


Figure 16: General layout of neural network

As mentioned before, the objects are going to be points in our algorithm. The inputs of the system is going to be radar data and the distance to the target. Output is going to be the the variable that passed to the move function. So the neural network is going to figure out that the relationship between the obstacles around itself and the distance to its target. For the output part, there is another function that not mentioned until now, it is called activation function and it defines the possible respond behaviours of the output. It is mandatory to use in these kind of systems because it greatly increases the stability of the output and this helps to learn the neural network much easy. In the Neat systems, activation function can be user customized step function or there are built-in functions that user can define as activation function such as sin, square, cube and exp. In this algorithm the most suitable activation function is hyperbolic tangent function called tanh. The reason of this choose is related to working principle of output nodes in this algorithm . In this algorithm, system got 2 output nodes one of them is related to x movements other one is related to y movements, each of the nodes are responds with the range of the activation function of hyperbolic tangent. As shown in the figure 16 the output of the tanh function is in range of -1 to the +1 symmetrically. And our each output node checks the result of the activation functions output, if the output is greater than zero it increases the related axis and that moves our point in that axis positively, if it is smaller than zero it moves the point at that axis negatively, finally if it is zero it stands in that axis with no movement. The reason behind the selection of the activation function as hyperbolic tangent is compress the output values to the [-1,+1] symmetrically, so the responses of the neural network is going to be symmetrical. So in general the system has 9 input nodes, 8 of them is radar data and 2 output nodes and that each of them is related to one axis

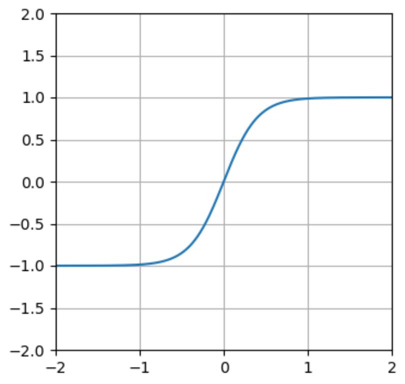


Figure 17: Hyperbolic tangent function

The next parameter in the neural network is fitness function. As explained before the objects in each generation is ordered with the values of fitness function in maximum to minimum at the end of the process some of the objects are eliminated as like in natural selection. In this part, we will see how the parts are getting fitness points in this virtual world. First the main objective is getting closer to the target point so we will focus on this part first. So each point is starting with the 10.000 fitness points. The square of distance to the target position is substracted from fitness point for each point. So the points that ends up closerto the target point than the other points are getting more points at the end of the generation. And we do not want to get closer to the obstacles in the map, so if the point is gets closer than 20 pixels get minus 20 points. With that way, the points are inclined to get away from any obstacle and that gives the point safe area as like the real world. Another parameter that affects the fitness of the points is time spended. We do not want that our vessel choses a path that increases the length of the way unnecesarily. So the fitness value of each point gets minus one for each second that passes. That means, the quicker the point reaches the destination gets more fitness points compared to other. And the last affect on the fitness point is checking that collision with any obstacles. If the point collides with the obstacle it gets -500 points. And that probably makes that point worst point in its generation that means no need to evolve this type of neural network anymore becaouse it does not figured out yet that collision is the worst thing that can happen to the vessel. So the general fitness function is given below as explained (14).

$$fitness\ point = 10000 - (distance\ to\ target)^2 - (time) \qquad (14)$$

All the methodes in this NEAT system is explained properly at the sections before. Now, we will look the process of the path planning in our algorithm. First user must choose the map that going to work on it, then application asks user for choosing two points. One of them is starting point other one is target point. Then generation starts when the user clicks the generate button. Each generation is starts with 20 different points and ends when the time is over or all of the points are collide with obstacles. And generation process ends when the algorithm finds a path between starting point to target point. The coordinates of the path is going to save in a list at the end of the process. The main objective of the algorithm is accomplished when the system finds the path saved the coordinates of the path to a list.

3.2.5.3. Transfer data Python to Simulink

The list of data that we founded from the python application, is needed by the simulink algorithm to move the vessel to the target point. So we have to transfer the data to the matlab workspace without any corruption. It is easy application becaouse the Python Matlab interaction is so easy to apply. But we are not just going to transfer the value we are going to modify the coordinate values when we transfer them. The reason that we are going to modify is because the coordinate values that came from Python is not starts from (0,0). But in Simulink the vessel starts at the point of (0,0). So we are going to change the coordinate values from Python that the coordinates inside the list starts from the zero point. Mathematically, we are going to attain the value differece of the current point and the next point to the current point. And add the coordinates of previous points. The starting point is not included in this formula, it is directly assinged as (0,0). With that formula (15)(16) the can gets the exactly the same path that start from the point of (0,0)

$$new\_x_{\ index} = (x_{index+1} - x_{index}) + x_{index-1} \tag{15}$$

$$new\_y_{\ index} = (y_{index+1} - y_{index}) + y_{index-1} \tag{16}$$

```
pyenv("Version","C:\Program Files\Python311\python.exe")
dataTransmitted  = pyrunfile("targetCoordinates.py","path_points");

x_coordinates = dataTransmitted{1};
y_coordinates = dataTransmitted{2};
length_of_list =max(length(x_coordinates),length(y_coordinates));

x_coordinates{1}=0;
y_coordinates{1}=0;

for index = 2:length_of_list-1
    x_coordinates{index} = (x_coordinates{index+1} - x_coordinates{index});
    y_coordinates{index} = (y_coordinates{index+1} - y_coordinates{index});
    x_coordinates{index} = (x_coordinates{index} + x_coordinates{index-1});
    y_coordinates{index} = (y_coordinates{index} + y_coordinates{index-1});
end

X = cell2mat(cell(x_coordinates));
Y = cell2mat(cell(y_coordinates));
```

You can see the code above, first the values are transferring from another Python file. And the data that came from python are assinging as x_coordinates and y_coordinates, then the first coordinate values are assinged as zero. The mathematical function is proceeding the for loop, the loop is starting from the index of 2 as you may notice. The reason behind is system already assigned the first coordinate values as zero. And lastly we are converting all those list values into format that suitable for Simulink and assiging them into X and Y variable for further usages in simulator. And this is all the code that we are going to use and with that way we are connecting and using those two systems together. And that’s all for the path folllowing application, the whole code is given in Appendix A. Readers can analyze and examine the code if they wanted to do.

4. RESULTS AND DISCUSSION

In this last part of the thesis we are going to test our system in a various scenarios and discuss about the results that cames from the test. For the testing we have various scenarios that from real map or fiction maps that does not exist. And also we will go over these steps of the processes again. And we are going to result the each test with the pros and cons of the system in that particular scenario.

For the first scenario, the map that USV is going to test is a does not exist in real world .I made a random map drawing for the testing in various conditions while I was devoloping this algorithm. In this map, I choosed random two points in the path finding algorithm. I choosed them far away enough to test the algorithm in hard conditions.The points that I choose are shown in Figure18 as green dots in the map.The point at the bottom left is starting point, the other one at the left top is target point. Then program waits for the command to start deep learning process. When the command comes algorithm automatically tries to find the path as explained in detail before. The screenshot while the learning process happening is given in Figure 19. When the algorithm find the path it automatically sends it to the connection file between Pytohn and Simulink and it displays it if the user wants to see what is it like to be. The results of this scenario is shown in Figure 20. The coordinates of the path is saved and sended to Matlab workspace. The Matlab workspace holds it and converts with the codes of the connection algorithm to the type of list that Simulink can handle.

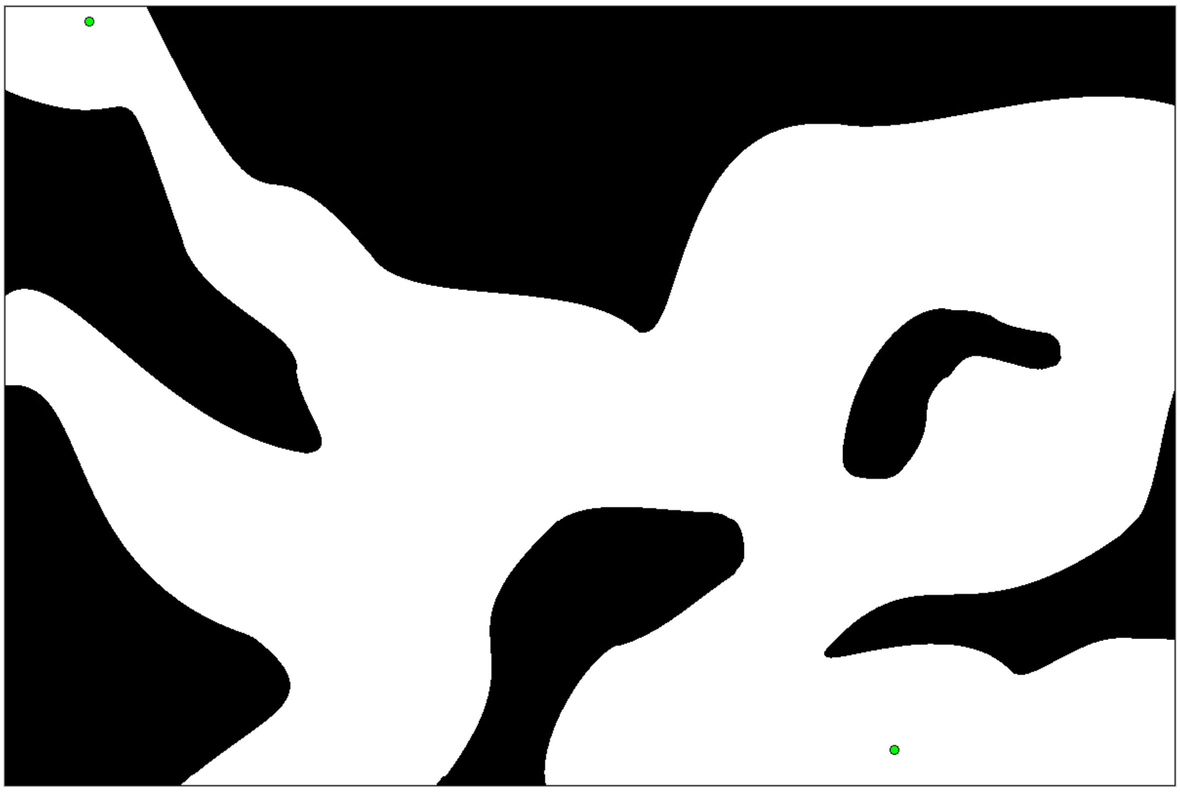


Figure 18: Scenario 1, starting point and target point

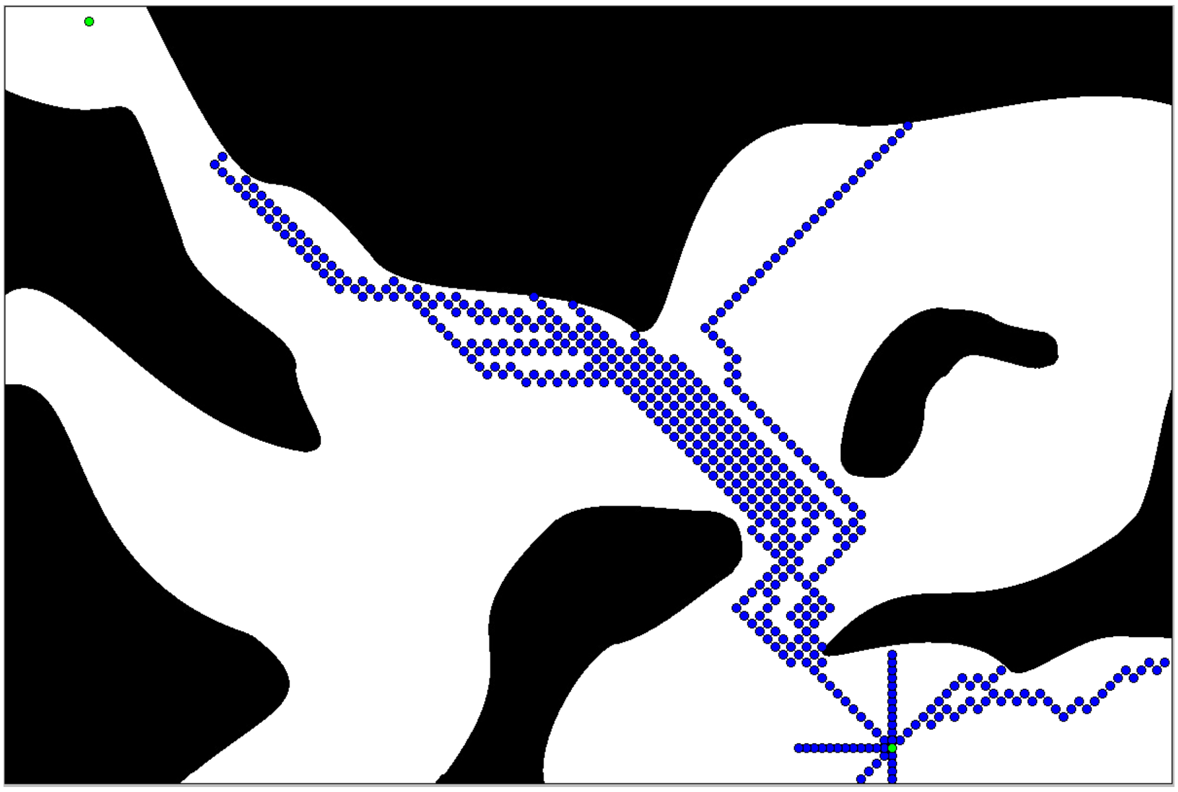


Figure 19: Scenario 1, learning process

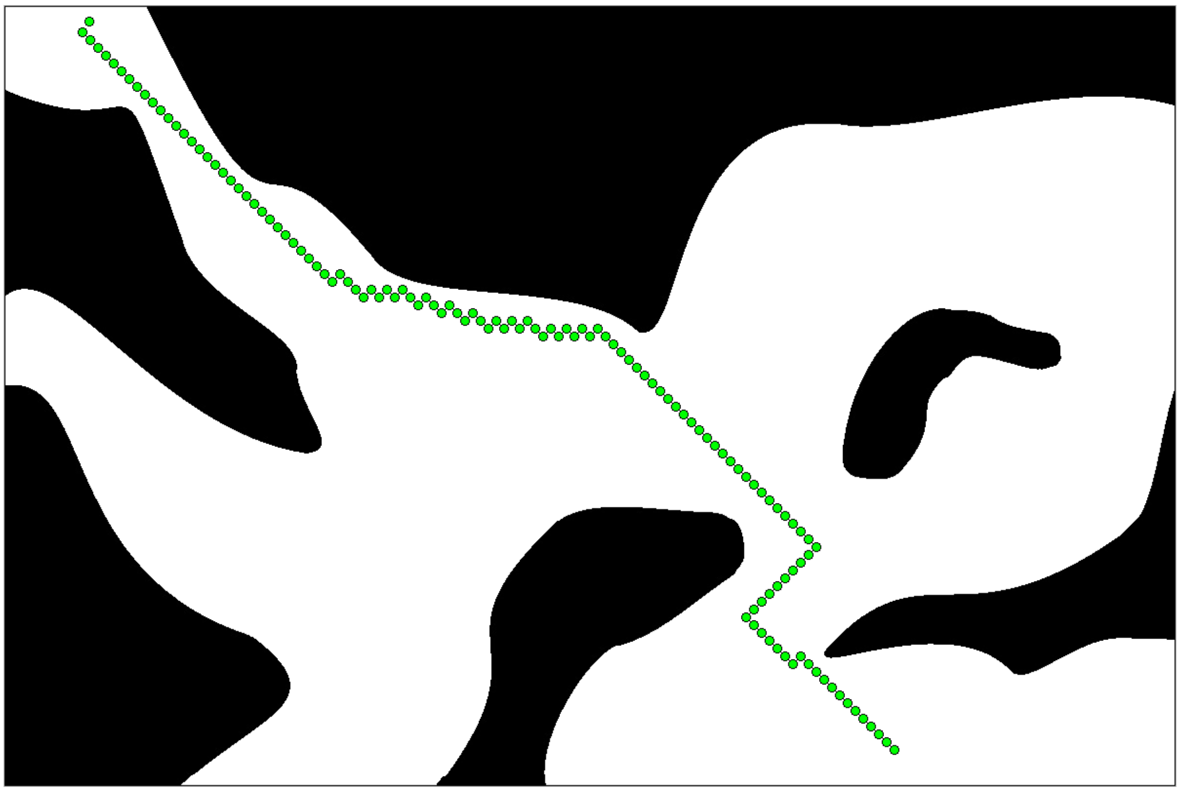


Figure 20: Scenario 1, generated path

For the scenario one, it is shown that how the algorithm looks like in the learning process at the Figure 19. How the algorithm rewards and give penalty to the each point with the fitness function is obvious in this figure. The points that get more fitness points are creating the newer versions of itselfs This is the reason why the way to the target point is ticker than the other directions. But the algorithm is not always creates points that closer to the target. it is possible to find a shorter way evertime, so the algorithm always tries to compleatly new moves for the chance of finding shorter way. If the new movements are not gathers any extra fitness points ends that mutation, if it provides extra fitness points, system concentrates about that points. In the process some of the points are moves to the meanless directions as shown Figure 19.

And the last part of this scenario one is using those path lines as the command to the virtual USV model of us. The values that saved as the coordinates of the path is converted to the type of list that the Simulink can use from the algorithm that in the matlab, and Matlab gives a command to start the simulation. And simulation starts, the general system in the simulink that explained in this project tries to reach the coordinates on the path with the variable motor commands, and program reaches each of the points with the help of subsystems in the Simulink. And reaching each points makes USV to follow the path that algorithm founded. While the path following process in Simulink user can see where the USV is heading trough as shown in Figure 21.

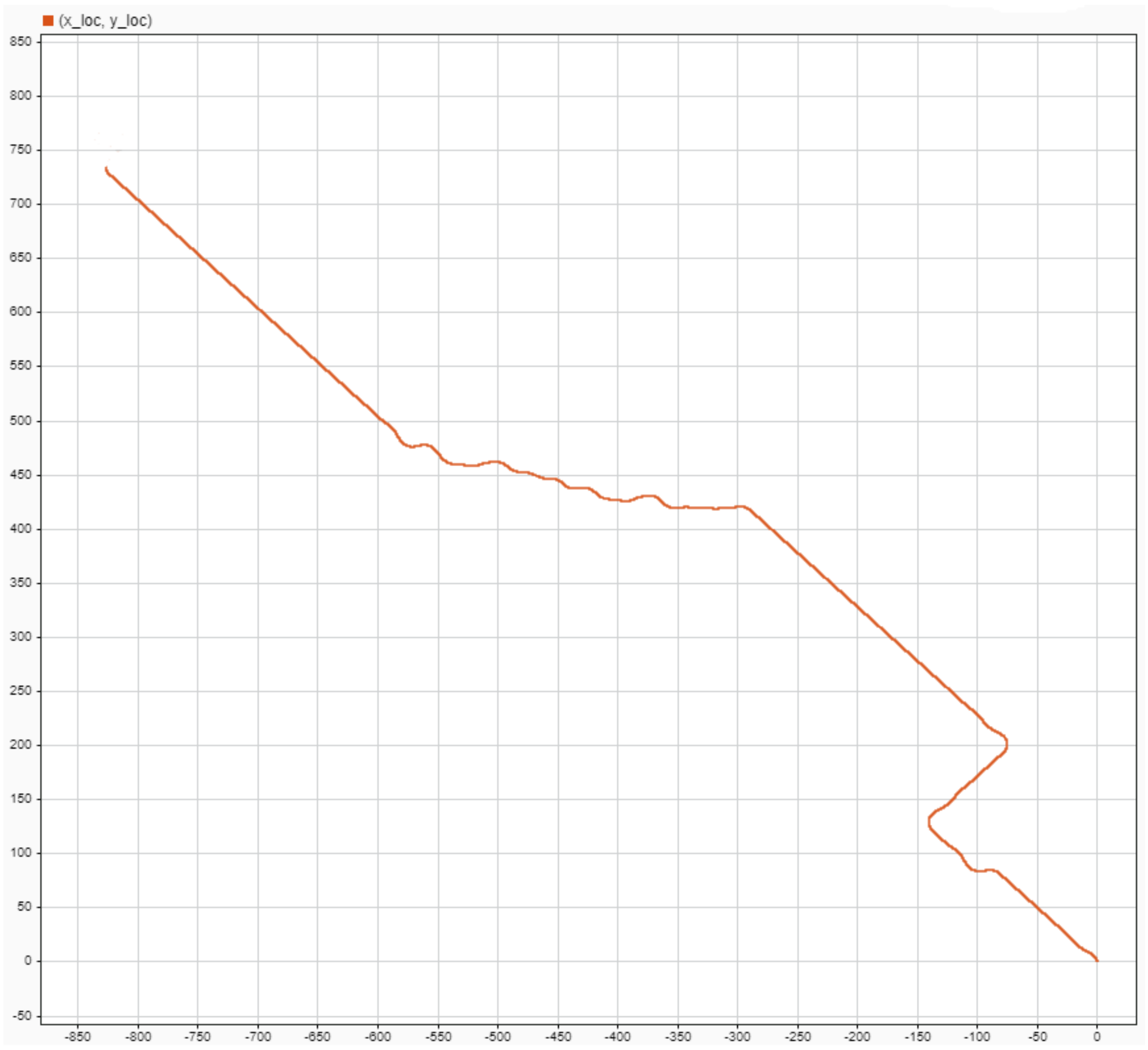


Figure 21: Simulink output, path following results

As you can see in the Figure 21, it is very similar to the path that generated by the Python application. The simulink gives the commands so precisely that the wobbling in the python commands are reflects to the path of the USV. The reason of the unconsistency at the middle of the path is caused by this presicion.

For the scenario 2, tests are going to be held in a real map for the comparison in real life. The tests are going to be held on the map of south entrance of the istanbul bosphorus. The choosen points are going to be a path of a ferry that locals of the istanbuls oftenly uses. those points that we are going to choose are the Kadikoy pier that lays bottom left of the map, and eminönü pier that positioned top right side of the map. The ferries of istanbul is uses that path for every day for transportation service to the people of istabul. The selection of this path is mainly because, we have acsess to the coordinate data of the ferry that works at this line. At the end of the navigation process of ours, a little comparision between our USVs coordinate data and the data of the ferry is going to be compared.

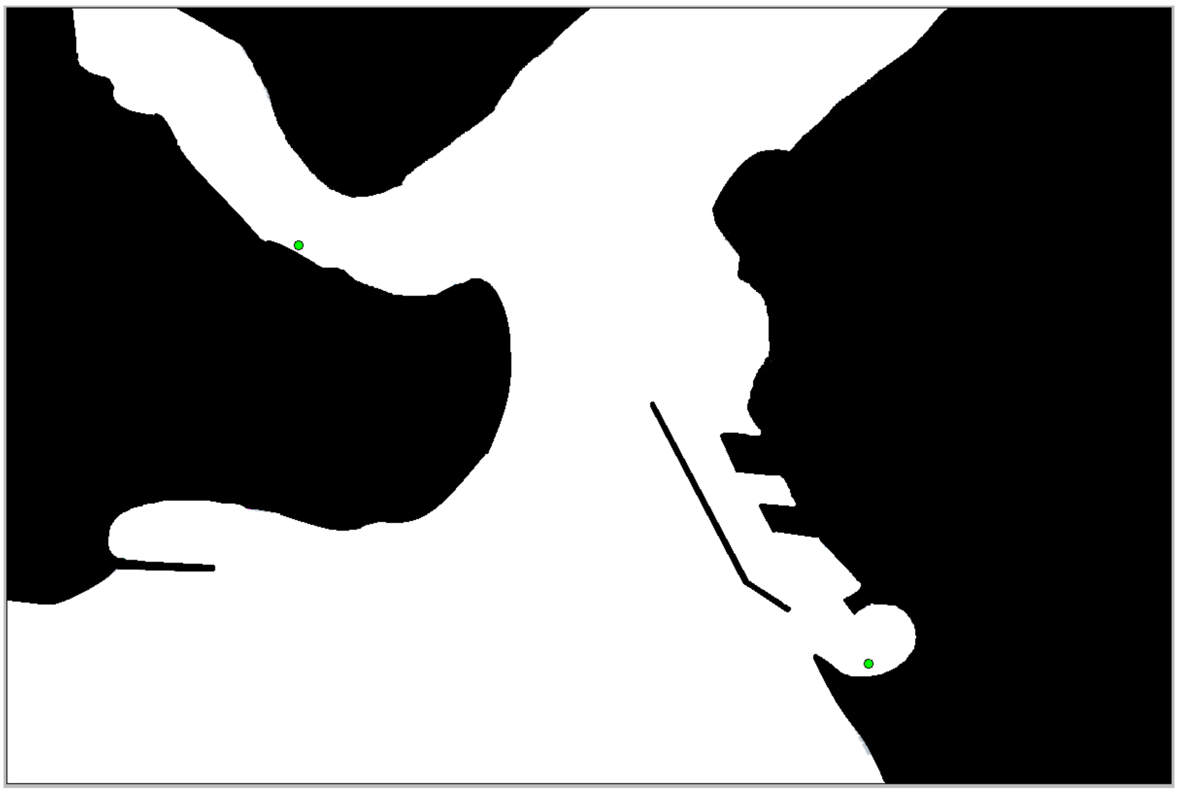


Figure 22: Scenario 2, starting point and target point

The path that algorithm generated is shown in Figure 23. And the Simulink output of the generated path is shown in Figıre 25. Those two figures shows that the algorithm of path generating and path following is working fine. In Figure 24 it shows the path of the ferry Beykoz when it is working on the Kadikoy - Eminonu – Karakoy. This is the same path that our vessel followed in the Simulink. And now, i am going to make a point with those 3 data.

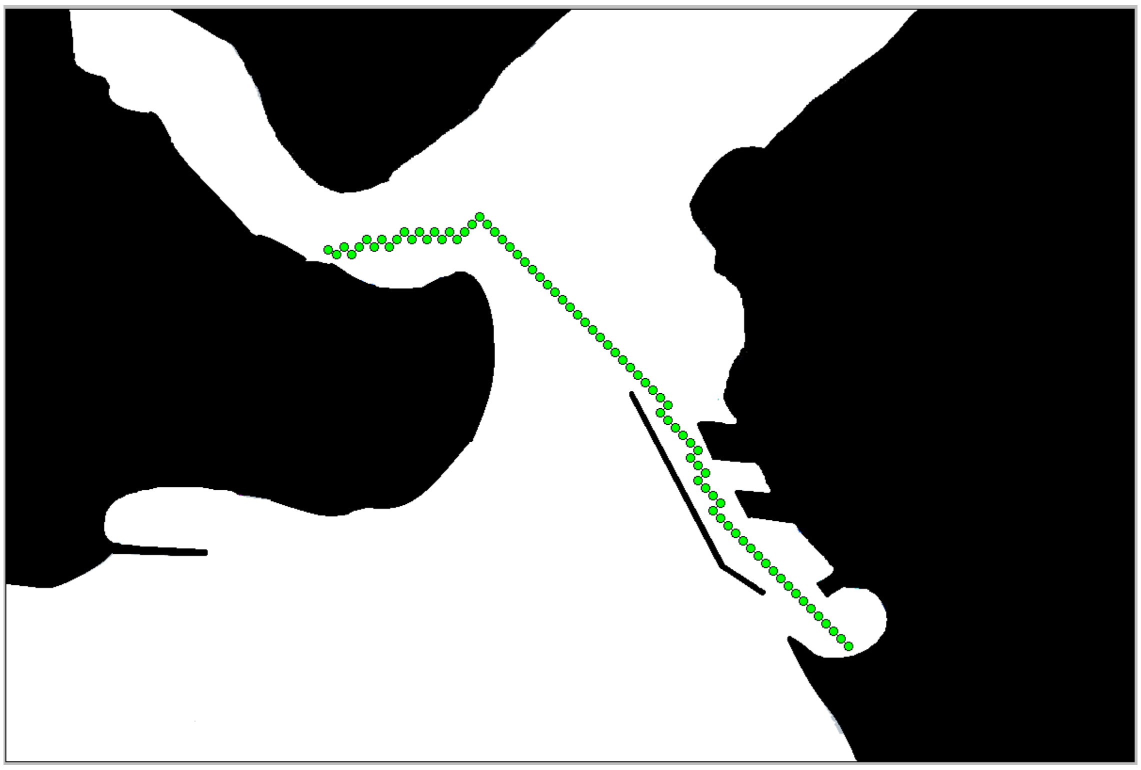


Figure 23: Scenario 2, generated path

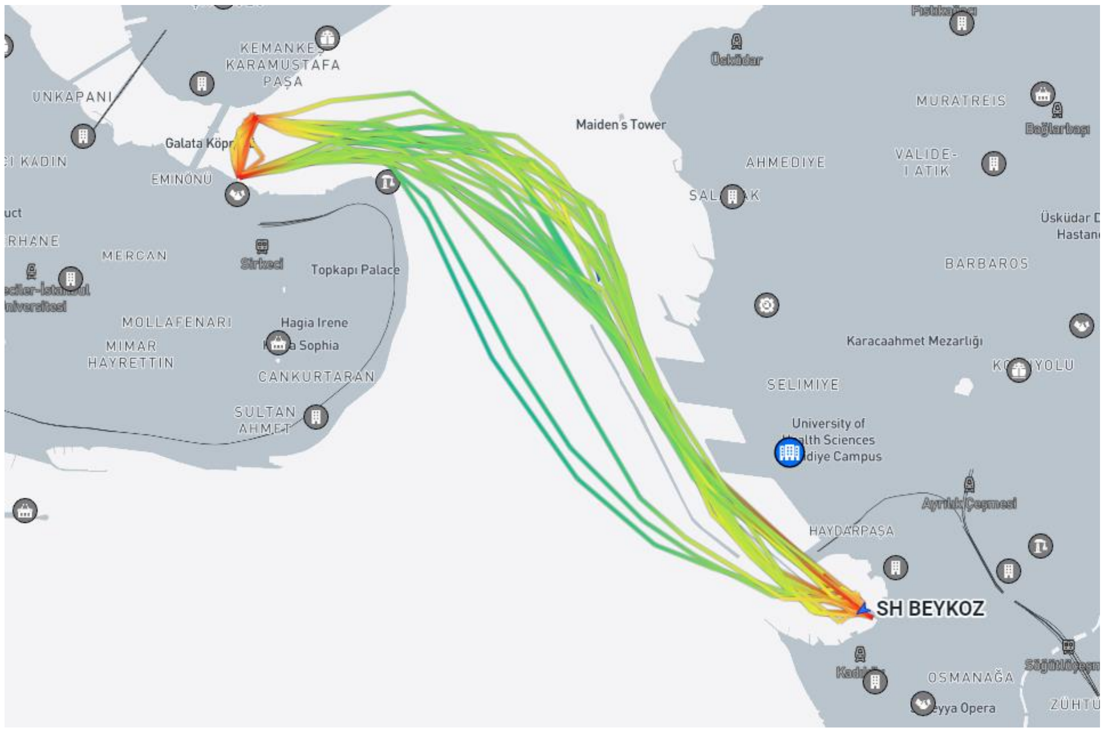


Figure 24: Daily path of Beykoz ferry

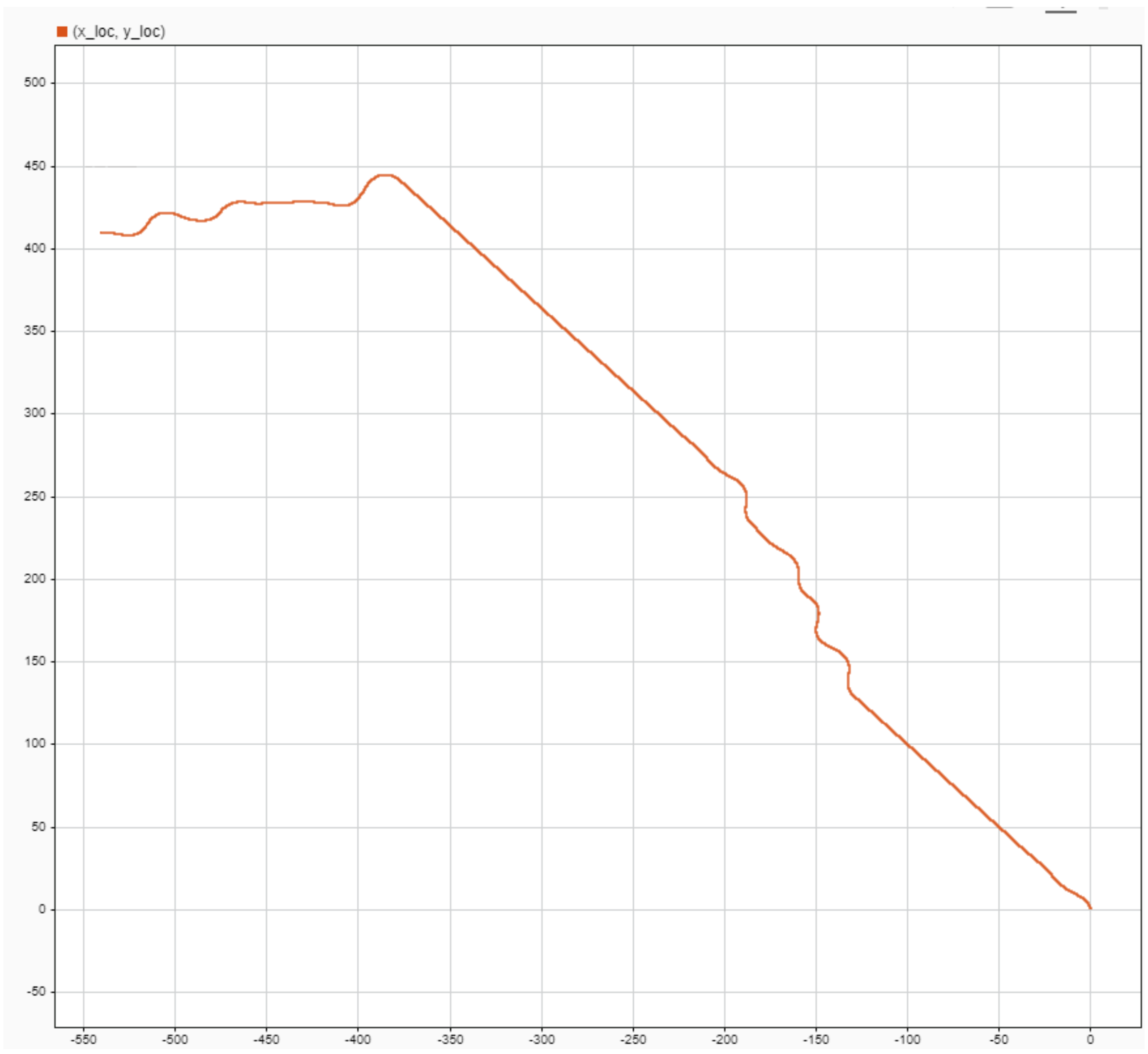


Figure 25: Scenario 2, simulink output.

There is a obvious consistency difference between Figure 23 and Figure 24. If the control of a ferry is an autonomous system, it will always follow the same path except extraordinary conditions, but when it is controlled by human controller it is not easy to follow same path for all day. Besides the unconsistency, autonomous system can reach the destination with less miles compared to the human controller. It is not possible to compete with autonomous system in this race because it is certaion that the mathematic of the system will be result much better than the human eyes, exept the extraordinary conditions. All the unconsistency of the human controller plus the mathematical performance of the autonomy when its calculating the most feasible route, reflects as extra fuel consumption at the end of the day. For the ferry that works each day of the year in the same path, the little effects of the fuel

consumption per each voyage, is enourmously great at the end of the year. And the carbondioxide emmission is going to be affected as same as fuel consumption, and also less mile the ferry travels, much better for the economical life of the machines of the ferry.
Whenall those effects are summed at the end of the year, there is huge advantage of the autonomous system is going to occur.

5. CONCLUSION

In conclusion, this project aims to eliminate the human work force on the USVs. Because these kind of vessels are already eliminates some of the workforce of humans in the process of the maritime. Minimizig the rest of the human work force is crucial because the rest of the process is not easy to do for ordinary algorithms. Until now, algorithms made for maritime failed or accepted as failure one way or another. In this project, the aim is break this boundary with the deep learning process. And shows that the cooparation of the simulator for the controlling the vessel and path planning algorithm uses deep learning systems can be succsesful together. But as mentioned before it is not easy for the traditional static algoritms to create variable algorithm that suitable for autonomy in maritime. the reason for that is, it is not something that can learnable for one time because of the variety of variables. The parameters in the maritime is changeable and the responds to these parameters are not same for each time. This is the reason that the system must be improve itself gradually. And the difference of this project is occurs with the usage of the deep learning of the neural networks. The vessel improves its navigation performance with the evolution of the neural network. This is the difference and the key of succses of it for this project.

As shown in scenario two, the project has got so many things to improve but even this time it is obvious that these kind of systems will provide significantly better fuel consumption and consistency compared to the vessels that uses human control. This is one the biggest advantages of those kind of projects that this project that can not ignored. And it is proved again that with the test of the scenario two, the values of fuel consumption, carbondioxide emission will be decrease significantly. That means, if researchers implement autonomy properly in maritime sector the advantages of the autonomy will be significantly great.

In general The aim of this project is to make a small introduction to how USVs can find their path to the target fully automated. The general layout of this project is, the python application generates the most efficient and feasible path with the deep learning in two marked points. When the path is generated the system transfers the path data to Simulink to virtual environment of the vessel. The vessel follows the path with the help of subsystem algoirthms that designed. And finally, Simulink displays the coordinate values of the vessel to the user. This is how the USV finds it own path with the help of algorithms that devoloped in this project.

The system works fine with the scenarios that generated and pros that can affect the maritime but it is obvious that there are so many things that must be improved for the next phases of the autonomy. The reality of the open sea is totally different from simulator system that devoloped in the project. Besides the difference between the reality and the virtual environment, the algorithm of path plannar must be improved with the responds of the real world. Because deep learning is improving itself in path planning without all the dymanics responds of the real world. There are too many other parameters that the USV have to look for. For instance, the affect of the nature is only ocuured as waves in this project, yet there are bunch of natural forces that affects the vessels that we have not mentioned like currents, wheather and winds. Other than nature also there is marine traffic that not mentined in this project the vessel must pass that sucsessfully.

At this time have highlight that this project is only looks for route planner and navigation algorithm not a fully automation algorithm for USV. There are so many things that researchers must improve for the pass the next automation phases. And i hope that this work lays a foundation for the people that tries to devolop a automation application for maritime.

REFERENCES

Acejo, I., Sampson, H., Nelson , T., & Ellis, N. (2018). *The causes of maritime accidents in the period 2002-2016.* Cardiff: Seafarers International Research Centre (SIRC).

Atacan, C. (2022). *Determination of Risks During Navigation on Fishing Vessels.* İzmir: Ege University.

Aye, M. M. (2011). Analysis of Euler Angles in a Simple Two-Axis Gimbals Set. *World Academy of Science, Engineering and Technology*, (pp. 389-394).

Bahçe, M. A. (2019). *Evaluation of Collusion Avoidance Alghoritms in Autonomous Ship Navigation.* İstanbul: İstanbul University - Cerrahpasa.

Baldauf, M., Fischer, S., Mehdi, R. A., & Gluch, M. (2017). *A perfect warning to avoid collisions at sea.* Szczecin: Maritime University of Szczecin.

Bayrak, M. (2023). *COLREGs-Compliant and Non-Prioritized Motion Planning for Autonomous Unmanned Surface Vehicles.* İstanbul: Istanbul Medeniyet University.

Bertram, V. (2004). *Technologies for Low-Crew/No-Crew Ships.* Hamburg.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control.* Trondheim, Norway: Norwegian University of Science and Technology.

Karey, L. (2017). *All Hands Off Deck The Legal Barriers to Autonomous Ship.* Johor Bahru: National University of Singapore.

Kenneth O. Stanley, R. M. (n.d.). *Efficient Evolution of Neural Network Topologies.* Texas: The University of Texas at Austin.

McIntyre, A. K. (n.d.). neat-python.

Niu, Y., Zhu, F., & Zhai, P. (2023). An Autonomous Decision-Making Algorithm for Ship Collusion Avoidance Based on DDQN. *The 7th International Conference on Transportation Information and Safety* (pp. 1174-1180). Xi'an,: China.

Rokseth, B., Haugen, O. I., & Utne, I. B. (2019). Safety Verification for Autonomous Ships. *MATEC Web of Conferences* (pp. 1-15). Trondheim,: DNV GL, Group Technology & Research.

Sandaruwan, D., Kodikara, N., Keppitiyagama, C., & Rosa, R. (2010). Perception Enhanced Virtual Environment for Maritime Applications. *GTFS International Journal on Computing vol:1 no:1*, 36-43.

Theunissen, E. (2014). *Navigation of unmanned vessels – History, Enablers, Challenges and Potential Solutions.* Den Helder: Netherlands Defence Academy, The Netherlands.

Wu, B., Sæter, M., Hildre, H., Zhang, H., & Li, G. (2022). Experiment Design and Implementation for. *IEEE International Conference on*, (pp. 1-6). Prague, Czech Republic.

Ye, J., Li , C., Wen, W., Ruiping, Z., & Vasso, R. (2023). Deep Learning in Maritime Autonomous Surface Ships: Current Development. *Journal of Marine Science and Application*, 584-601.

APPENDICES

APPENDIX A

```
import pygame, os, sys
from pygame.locals import *
from time import sleep
import math
import neat
import tkinter as tk
from tkinter import filedialog

pygame.init()
clock = pygame.time.Clock()
pygame.display.set_caption("Auto Route Planner")
screen = pygame.display.set_mode((1450,850),0,32)
screen.fill('white')
font = pygame.font.Font('freesansbold.ttf',18)


class Text:
    def __init__(self,text,x_pos,y_pos,
                 font,text_col,enabled):
        self.text = text
        self.x_pos = x_pos
        self.y_pos = y_pos
        self.font= font
        self.text_col = text_col
        self.enabled = enabled
        self.draw()

    def draw(self):
        text_disp = font.render(self.text,True,
                                self.text_col)
        screen.blit(text_disp,
                    (self.x_pos,self.y_pos))

def calcDist(x1, y1, x2, y2):
    distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
    return distance

def chooseMap():
    root = tk.Tk()
    root.withdraw()  # Hide the root window
    file_path = filedialog.askopenfilename()
    return file_path
```

```
class Button:
    def __init__(self,text,x_pos,y_pos,x_text,y_text,width,height,enabled):
        self.text = text
        self.x_pos = x_pos
        self.y_pos = y_pos
        self.x_text = x_text
        self.y_text = y_text
        self.width = width
        self.height = height
        self.enabled = enabled
        self.draw()

    def draw(self):
        button_text = font.render(self.text, True,'black')
        button_rect =pygame.rect.Rect((self.x_pos,self.y_pos)
                                      ,(self.width,self.height))
        if self.check_click():
            pygame.draw.rect(screen,(100,100,100,255),button_rect,0,7)
        elif self.enabled==False:
            pygame.draw.rect(screen,(100,100,100,255),button_rect,0,7)
        else:
            pygame.draw.rect(screen,'light gray',button_rect,0,7)
        pygame.draw.rect(screen,'black',button_rect,2,5)
        screen.blit(button_text,(self.x_pos+self.x_text,
                                 self.y_pos+self.y_text))

    def check_click(self):
        mouse_pos = pygame.mouse.get_pos()
        left_click = pygame.mouse.get_pressed()[0]
        button_rect = pygame.rect.Rect((self.x_pos,self.y_pos),
                                       (self.width,self.height))
        if left_click and button_rect.collidepoint(mouse_pos) and self.enabled:
            return True
        else:
            False

class Point:
    def __init__(self, x_pos,
                y_pos,x_target,
                y_target,color):
        self.x_pos = x_pos
        self.y_pos = y_pos
        self.x_move = 0
        self.y_move = 0
        self.x_limit = [5,1445]
        self.y_limit = [5,845]
        self.center = [x_pos + 5, y_pos + 5]
        self.isAlive = True
        self.x_target = x_target
        self.y_target = y_target
        self.x_tick = 0
        self.y_tick = 0
        self.color = color
```

```
def update(self,x_pos,y_pos):
        if self.isAlive:
            self.x_pos = x_pos + self.x_tick
            self.y_pos = y_pos + self.y_tick
            self.center = [x_pos + self.x_tick+ 5,
                           y_pos + self.y_tick+ 5]
        else:
            pass
        self.draw()

    def draw(self):
        self.survive()
        if self.x_pos>230 and self.y_pos>10:
            pointRect = pygame.rect.Rect((self.x_pos + self.x_move,
                                          self.y_pos + self.y_move),
                                          (10, 10))
            pygame.draw.rect(screen, self.color, pointRect, 0, 4)
            pointRect = pygame.rect.Rect((self.x_pos + self.x_move,
                                           self.y_pos + self.y_move),
                                           (10, 10))
            pygame.draw.rect(screen, (2,2,2,255), pointRect, 1, 4)


def radarData(self):
        distance = 1

        angles = [0, 45, 90, 135, 180, 225, 270, 315]
        distance_list = [1, 1, 1, 1, 1, 1, 1, 1]
        pointCenter = [self.center[0],self.center[1]]

        for i in range(8):
            checkRadar = True
            while checkRadar:
                distance_list[i] += 1
                if distance_list[i] > 300:
                    checkRadar = False
                x_end = int(self.center[0] +
                            (distance_list[i] *
                             math.sin(math.radians(angles[i]))))
                y_end = int(self.center[1] +
                            (distance_list[i] *
                             math.cos(math.radians(angles[i]))))
                if x_end>self.x_limit[1] or x_end <self.x_limit[0]
                    pass
                else:
                    if screen.get_at((x_end, y_end)) == (0, 0, 0, 255):
                        x_end = self.center[0]
                        y_end = self.center[1]
                        checkRadar = False
        return distance_list
```

```
def checkCollusion(self, distance):
        stateCollusion = False
        if any(d < distance for d in self.radarData()):
            stateCollusion = True
        else:
            stateCollusion = False
        return stateCollusion

    def move(self, direction):
        if direction == 0:
            pass
        if direction == 1:
            self.y_tick = -8
        elif direction == 3:
            self.y_tick=  +8
        elif direction == 2:
            self.x_tick = +8
        elif direction == 4:
            self.x_tick = -8

    def survive(self):
        if self.checkCollusion(9) or self.x_pos>1430 or self.x_pos<230 or
           self.y_pos>850 or self.y_pos<10:
            self.isAlive = False
        else:
            self.isAlive = True

    def dataOutput(self):
        data = self.radarData()
        dataAdd = int(calcDist(self.center[0],self.center[1],
                                self.x_target,self.y_target))
        endData = data.append(dataAdd)
        return data

    def distToTarget(self):
        distance = self.radarData()
        return distance[8]

destCounter = 0
run_main = True
genbtnState = False
gen_swState = False
choose_destState = False
destBtnState = True
dispPathState = False
engGenClicked = False
generalInfo = 0
total_generations = 2000

dest1 = [0,0]
dest2 = [0,0]

x_move = 0
y_move = 0
```

```
def run_generations(genomes,config):
    pygame.draw.rect(screen,(190, 190, 190, 255),
                            pygame.rect.Rect((0,0),(1450,800)),0,0)
    pygame.draw.rect(screen,'white',pygame.rect.Rect((230,10),(1200,800)),0,0)
    mapBlack = pygame.image.load(mapPath)
    text_fileState = Text(f'Map {mapPath} ',40,133,font,'black',True)
    pygame.draw.rect(screen,'black',pygame.rect.Rect((230,10),(1200,800)),1,0)
    screen.blit(mapBlack,(230,10))

    dispPathState = False
    btn_generate.draw()
    btn_options.draw()
    btn_browse.draw()
    btn_chooseDest.draw()
    btn_dispPath = Button("Display Path",15,480,47,13,200,50,dispPathState)

    point1.draw()
    point2.draw()

    print(engGenClicked)
    nets = []
    points = []
    ge =[]

    for id, g in genomes:
        net = neat.nn.FeedForwardNetwork.create(g, config)
        nets.append(net)
        g.fitness = 0
        ge.append(g)
        points.append(Point(dest1[0],dest1[1],dest2[0],dest2[1],'blue'))

    x_listAll = []
    y_listAll = []
    x_listBest = []
    y_listBest = []
    global generation
    generation += 1
    main_tick = 0
    time = 50
    best_index = 0
    choose_path = 0
  while True :
        main_tick +=1

        if generation < 7:
            time = 10
        if generation >= 7 and generation < 20:
            time = 50
        if generation >= 20:
            time = 150

        point1.draw()
        point2.draw()
```

```
def run_generations(genomes,config):
    pygame.draw.rect(screen,(190, 190, 190, 255),
                            pygame.rect.Rect((0,0),(1450,800)),0,0)
    pygame.draw.rect(screen,'white',pygame.rect.Rect((230,10),(1200,800)),0,0)
    mapBlack = pygame.image.load(mapPath)
    text_fileState = Text(f'Map {mapPath} ',40,133,font,'black',True)
    pygame.draw.rect(screen,'black',pygame.rect.Rect((230,10),(1200,800)),1,0)
    screen.blit(mapBlack,(230,10))

    dispPathState = False
    btn_generate.draw()
    btn_options.draw()
    btn_browse.draw()
    btn_chooseDest.draw()
    btn_dispPath = Button("Display Path",15,480,47,13,200,50,dispPathState)

    point1.draw()
    point2.draw()

    print(engGenClicked)
    nets = []
    points = []
    ge =[]

    for id, g in genomes:
        net = neat.nn.FeedForwardNetwork.create(g, config)
        nets.append(net)
        g.fitness = 0
        ge.append(g)
        points.append(Point(dest1[0],dest1[1],dest2[0],dest2[1],'blue'))

    x_listAll = []
    y_listAll = []
    x_listBest = []
    y_listBest = []
    global generation
    generation += 1
    main_tick = 0
    time = 50
    best_index = 0
    choose_path = 0
  while True :
        main_tick +=1

        if generation < 7:
            time = 10
        if generation >= 7 and generation < 20:
            time = 50
        if generation >= 20:
            time = 150

        point1.draw()
        point2.draw()
```

```
for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit(0)

        x_list = []
        y_list = []

        for n in range(population_size):
            x_list.append(points[n].x_pos)
            y_list.append(points[n].y_pos)
        x_listAll.append(x_list)
        y_listAll.append(y_list)


        if best_index !=0:
            if choose_path == 0:
                for n in range(len(x_listAll)):
                    x_listBest.append(x_listAll[n][best_index])
                    y_listBest.append(y_listAll[n][best_index])
                choose_path = 1


        for index, point in enumerate(points):
            output = nets[index].activate(point.dataOutput())
            i = output.index(max(output))
            if output[0] > 0:
                point.move(2)
            if output[0] < 0:
                point.move(4)
            if output[1] > 0:
                point.move(3)
            if output[1] < 0:
                point.move(1)


            point.move(output)
            point.update(point.x_pos,point.y_pos)


            distanceToTarget= point.dataOutput()[8]
            if distanceToTarget<20:
                    succeeded_points +=1
                    best_index = index


        remain_points = 0
        succeeded_points = 0
        point_fitness = []
        remain_points = 0
        succeeded_points = 0
        point_fitness = []
        for i, point in enumerate(points):
            if point.isAlive:
                remain_points +=1
                distanceToTarget= point.dataOutput()[8]
                genomes[i][1].fitness =10000-
                                       (distanceToTarget*distanceToTarget/100)-
                                       (main_tick/10)
                if point.x_pos == point1.x_pos and point.y_pos == point1.y_pos:
                    genomes[i][1].fitness -=500
                if point.checkCollusion(30):
                    genomes[i][1].fitness -=50
                point_fitness.append(genomes[i][1])
                if distanceToTarget<20:
                    genomes[i][1].fitness +=200000
                    dispPathState = True
                    genEnd = False
                    path_points = [x_listBest,y_listBest]
                    file = open("targetCoordinates.py","w")
                    file.write("path_points = " + str(path_points))
                    file.close()

        for point in points:
            point.draw()

        pygame.draw.rect(screen,(190, 190, 190, 255)
                         ,pygame.rect.Rect((0,290),(230,180)),0,0)
        pygame.draw.rect(screen,'black',
                         pygame.rect.Rect((15,330),(200,140)),2,3)
        text_title =Text('NEAT Information',45,300,font,'black',True)
        text_generationNumber =Text(f'Generation {generation}'
                                    ,60,340,font,'black',True)
        text_liveNumber =Text(f'{remain_points} Points Alive'
                                    ,55,370,font,'black',True)
        text_deadNumber =Text(f'{population_size-remain_points} Points Died'
                                    ,53,400,font,'black',True)
        text_passedTime =Text(f'{main_tick} time passed'
                                    ,55,430,font,'black',True)

        if remain_points == 0 or main_tick > time:
            pygame.draw.rect(screen,'white'
                             ,pygame.rect.Rect((230,10),(1200,800)),0,0)
            pygame.draw.rect(screen,'black'
                             ,pygame.rect.Rect((230,10),(1200,800)),1,0)
            screen.blit(mapBlack,(230,10))
            point2.draw()
            break
        if generalInfo == 2:
            pygame.draw.rect(screen,(190, 190, 190, 255)
                             ,pygame.rect.Rect((400,810),(1000,50)),0,0)
            text_generalInfo2 =Text('Wait For the Generation'
                                    ,500,817,font,'black',True)

        point1.draw()
        point2.draw()
        pygame.display.update()
        clock.tick(90)


mapPath = 'Map 1.png'
path_points = []
while run_main:

    mapBlack = pygame.image.load(mapPath)
    pygame.draw.rect(screen,'black',
                     pygame.rect.Rect((230,10),(1200,800)),1,0)
    screen.blit(mapBlack,(230,10))

    btn_generate= Button("Generate",
                         15,10,55,13,200,50,genbtnState)
    btn_options= Button("Options",
                        15,70,60,13,200,50,True)
    btn_browse = Button("Browse File",
                        15,170,45,13,200,50,True)
    btn_chooseDest = Button("Choose Destinations",
                            15,230,15,13,200,50,destBtnState)
    btn_dispPath = Button("Display Path",
                          15,480,47,13,200,50,dispPathState)

    pygame.draw.rect(screen,(190, 190, 190, 255),
                     pygame.rect.Rect((0,290),(230,180)),0,0)
    pygame.draw.rect(screen,'black',
                     pygame.rect.Rect((15,330),(200,140)),2,3)
    text_title =Text('NEAT Information',
                     45,300,font,'black',True)
    text_generationNumber =Text(f'Generation {generation}'
                                ,60,340,font,'black',True)
    text_liveNumber =Text(f'{0} Points Alive',
                          55,370,font,'black',True)
    text_deadNumber =Text(f'{population_size-0} Points Died'
                          ,53,400,font,'black',True)
    text_passedTime =Text(f'{0} time passed',
                          55,430,font,'black',True)
    pygame.draw.rect(screen,(190, 190, 190, 255),
                     pygame.rect.Rect((0,125),(230,40)),0,0)
    text_fileState = Text(f'Map {mapPath} ',
                          40,133,font,'black',True)

    mouse_pos = pygame.mouse.get_pos()
```

```
for event in pygame.event.get():
        if event.type == QUIT:
            run_main = False
            pygame.quit()
            sys.exit()
        if btn_chooseDest.check_click():
            choose_destState = True
            destBtnState = False
            generalInfo = 1
        if btn_generate.check_click():
            genbtnState = False
            gen_swState = True

    if btn_dispPath.check_click():
        import targetCoordinates
        path = targetCoordinates.path_points

        for index in range(len(path[0])):
            points = Point(path[0][index],path[1][index],0,0,'green')
            points.draw()


    if btn_browse.check_click():
        mapPath = chooseMap()


    if mouse_pos[0]>230 and mouse_pos[0]<1430 and choose_destState:
        if pygame.mouse.get_pressed()[0]:
            generalInfo = 1
            destCounter +=1
            if destCounter ==1:
                dest1=[mouse_pos[0]-5,mouse_pos[1]-5]
                print()
            if destCounter==2:
                dest2=[mouse_pos[0]-5,mouse_pos[1]-5]
                genbtnState = True
            sleep(0.1)
            if destCounter == 2:
                point2 = Point(dest2[0],dest2[1],dest1[0],dest1[1],'green')
                point1 = Point(dest1[0],dest1[1],dest2[0],dest2[1],'green')
                point1.update(dest1[0],dest1[1])
                point2.update(dest2[0],dest2[1])
                point1.draw()
                point2.draw()
                print(point1.x_pos,"  ",point1.y_pos)
                destBtnState = False
                choose_destState= False


    pygame.display.update()
    clock.tick(90)
    if genEnd:
        if __name__ == "__main__":
            config_path = "./config-feedforward.txt"
            config = neat.config.Config(neat.DefaultGenome,
                                        neat.DefaultReproduction,
                                        neat.DefaultSpeciesSet,
                                        neat.DefaultStagnation,
                                        config_path)
            p = neat.Population(config)
            p.add_reporter(neat.StdOutReporter(True))
            stats = neat.StatisticsReporter()
            p.add_reporter(stats)
            #print("RUN FUNC")
            if gen_swState:
                destBtnState = False
                generalInfo = 2
                pygame.draw.rect(screen,(190, 190, 190, 255),
                                 pygame.rect.Rect((400,810),(1000,50))
                                 ,0,0)
                p.run(run_generations,total_generations)
                gen_swState = False
                generalInfo = 3
                destBtnState = True
                dispPathState = True
```