AIR COMBAT DECISION MAKING WITH FUZZY LOGIC AND AIRCRAFT TRACKING

by

151220202178   BURAK ÖZDEMİR

151220192040   AYLA BİLGİN

151220194029   ŞEREF KARAKUŞ

A Graduation Project Report

Electrical Electronics Engineering Department

JUNE 2024

AIR COMBAT DECISION MAKING WITH FUZZY LOGIC AND AIRCRAFT TRACKING

by

151220202178 BURAK ÖZDEMİR

151220192040 AYLA BİLGİN

151220194029 ŞEREF KARAKUŞ

A Report Presented in Partial Fulfilment of the Requirements for the Degree Bachelor of Science in Electrical Electronics Engineering

ESKISEHIR OSMANGAZI UNIVERSITY

JUNE 2024

AIR COMBAT DECISION MAKING WITH FUZZY LOGIC AND AIRCRAFT TRACKING

by

151220202178 BURAK ÖZDEMİR

151220192040 AYLA BİLGİN

151220194029 ŞEREF KARAKUŞ

has been approved by

Supervisory Committee

[illegible]

Assoc. Prof. Dr. Hasan Serhan YAVUZ

[illegible]

Asst. Prof. Dr. Faruk DİRİSAĞLIK

[illegible]

Asst. Prof. Dr. Burak KALECİ

ABSTRACT

Air combat relies on complex strategies balancing attack and defense maneuvers. This project aims to enhance autonomous decision-making capability in air combat scenarios by developing an algorithm combining advanced techniques such as game theory, fuzzy logic, and dynamic programming. The algorithm to be developed focuses on enabling autonomous systems to make informed decisions during air combat. To achieve this, the developed algorithm will be tested in a simulation environment, and its success rate will be measured. The performance of the algorithm will be evaluated by testing it on various simulated combat scenarios. Success metrics will include mission success rates, resource utilization efficiency, and adaptability to changing environments. This study aims to present a new and effective approach that can be used to enhance the decision-making capability of autonomous systems in air combat situations.

Keywords: *air combat, fuzzy logic, dynamic programming, decision-making algorithms*

ÖZET

Hava muharebeleri, saldırı ve savunma manevralarının dengeli bir şekilde yapıldığı karmaşık stratejilere dayanmaktadır. Bu projede, hava muharebelerinde otonom karar alma yeteneğini artırmak için oyun teorisi, bulanık mantık ve dinamik programlama gibi ileri düzey tekniklerin birleşiminden oluşan bir algoritma geliştirilmesi hedeflenmektedir. Geliştirilecek olan algoritma, hava muharebelerinde bilinçli kararlar alabilen otonom sistemlerin geliştirilmesine odaklanmaktadır. Bu amaçla, geliştirilen algoritma simülasyon ortamında test edilecek ve başarı oranı ölçülecektir. Algoritmanın performansı, çeşitli simüle edilmiş muharebe senaryoları üzerinde test edilerek değerlendirilecektir. Başarı metrikleri arasında, misyon başarı oranları, kaynak kullanım verimliliği ve değişen ortamlara uygunluk gibi faktörler bulunacaktır. Bu çalışma, hava muharebelerinde otonom sistemlerin karar alma yeteneğini artırmak için kullanılabilecek yeni ve etkili bir yaklaşım sunmayı amaçlamaktadır.

Anahtar Kelimeler: *hava muharebesi, bulanık mantık, dinamik programlama, karar alma algoritması*

ACKNOWLEDGEMENT

We would like to thank our supervisor, Assoc. Prof. Dr. Hasan Serhan Yavuz for their useful comments and on-going support. We are also very grateful to Turkish Aerospace Industries for their financial and technical support; and Enis Kaan Keskin for his support in the development of the simulation and algorithm.

This project has been supported by TUBITAK with 2209-B 2024-2.

Assoc Prof. Dr. Hasan Serhan Yavuz

Electrical Engineer, Dr.

Profession: Image Processing,

Experience: 20+ years,

Organization: Eskisehir Osmangazi University

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviation	**Explanation**
ROS:	Robot Operating System
EEE:	Electrical and Electronics Engineering.
YOLO:	You Look Only Once
FIS:	Fuzzy Inference System

1. INTRODUCTION

Today's military operations demand sophisticated decision-making from military aircraft due to the increasing complexity and dynamic nature of air combat environments. Despite reduced direct human intervention in modern military operations, the rapidly evolving scenarios and uncertainties, such as changing enemy tactics and radar data, necessitate advanced decision-making mechanisms. This complexity poses significant challenges for the safety and success of military assets on the battlefield. With rising military expenditures and technological advancements, the demand for capable aircraft is growing. However, for these aircraft to be effective, they must quickly and accurately process information to make strategic decisions. Thus, enhancing the decision-making capabilities of warplanes in air combat is a primary concern for military strategists and engineers. In this context, the aim of this project is to develop a comprehensive decision-making mechanism to enhance the effectiveness of warplanes in air combat environments. This mechanism will support traditional military operations, allowing aircraft to evaluate targets and threats efficiently and make strategic decisions. With advancements in technology, the development of such algorithms has accelerated. One commonly used approach in these developments is fuzzy logic. According to studies in this field, fuzzy logic is a valuable method for analyzing and making decisions in complex and uncertain situations that cannot be described with simple binary values like true/false [1]. Fuzzy logic is particularly useful in air combat due to the numerous variables and uncertainties involved, such as enemy tactics and radar data. These factors can significantly affect the outcome of engagements. By processing complex and uncertain data, fuzzy logic can assess the threat levels of target aircraft. It takes into account various factors like the target's location, speed, and maneuvering capabilities to determine how dangerous a particular target aircraft is. This information allows military aircraft to make strategic decisions and respond appropriately to threats [2]. Additionally, the project plans to incorporate deep reinforcement learning to develop a dogfight scenario. Deep reinforcement learning enables military aircraft to improve their decision-making processes by learning from their experiences over time. This technique employs deep neural networks to estimate Q-values, which represent the total expected reward for each state-action pair, guiding the agent in choosing the best actions. The learning process occurs through trial and error, allowing the agent to learn which actions maximize its rewards [3]. Furthermore, the aircraft is expected to follow a path towards the enemy aircraft as determined by fuzzy logic. Initially, this path will be learned through reinforcement learning, followed by the realization of a dogfight scenario. As an alternative, the project aims to determine an optimal route using the A* algorithm, integrate a camera into the aircraft, and utilize object detection algorithms. This combination will enable the aircraft to detect enemy planes and engage in dogfight strategies as needed. By integrating these algorithms and techniques, the project strives to ensure that aircraft can successfully complete their missions and achieve strategic goals in air combat, effectively responding to the complex and uncertain conditions they encounter.

2. REQUIREMENTS SPECIFICATION

This thesis aims to develop autonomous aircraft systems during air combat. For this purpose, fuzzy logic-based decision-making algorithms have been developed and implemented using the Gazebo simulation environment and ROS to capture and process data. This algorithm is designed to detect enemy aircraft and direct them to the most dangerous enemy aircraft. Once the location of the enemy aircraft is determined, the aircraft will be detected with the YOLOv8 algorithm and tracking will be initiated. This pursuit is intended to resemble a dogfight scenario. Additionally, the A* path planning algorithm will be used to ensure the safe and efficient movement of the aircraft. These requirements include the basic steps necessary for the successful completion of the project.

2.1. Simulation Environment Requirements

- Installation and configuration of the environment to ensure compatibility of Gazebo and ROS platforms.
- Determination of requirements for simulating physical obstacles and weather conditions in the simulation environment.
- Gazebo and ROS platforms run on the Linux operating system [3].

2.2. Decision Making Algorithm Requirements

- Development of fuzzy logic based decision making algorithm.
- Fuzzy logic based detection system requirements to detect enemy aircraft.
- Decision making algorithms are written using Python. The 'skfuzzy' library in the Python programming language was used [4].

2.3. Enemy Aircraft Detection

- Selection and installation of appropriate camera hardware for the integration of image processing algorithms.
- Detecting enemy aircraft and determining their locations with the YOLOv8 algorithm.

2.4. Path Planning Algorithm Requirements

- Conducting reinforcement learning trainings and determining whether they work efficiently.
- Integration of the A* path planning algorithm for cruise planning towards the designated location of the enemy aircraft as an alternative solution.
- Development of motion control systems so that the aircraft can perform the necessary maneuvers to track enemy aircraft.

2.5. Performance and Reliability Requirements

- Determination of performance metrics related to the speed and accuracy of the decision-making algorithm.
- Providing automatic debugging and security measures to detect and fix any errors in the system.

2.6. Integration and Testing Requirements

- Seamless integration of all components (simulation environment, perception system, path planning and motion control).

- Planning and execution of system tests, component tests and integrated system tests.

3. STANDARDS

The following standards were used during the project development process.

- **IEC 29119-1** (*International Electrotechnical Commission's international standard on specifies general concepts in software testing and presents key concepts*): This standard was taken into account when testing the written algorithms and software.

- **IEEE 24641** (*IEEE Standard for specifies interrelationships between the components of the reference model*): The interoperability of the algorithms is done in accordance with this standard.

- **Software Language Standard- Python 3.8.x**: The software languages used in the project were used in accordance with the necessary standards.

4. PATENTS

Some similar patents in the project are listed below.

- Prioritizing use of flight attitude controls of aircraft, Patent NO: US11124289B2
- Unmanned vehicle route planning method based on improved A-star algorithm and deep reinforcement learning: CN111780777B

5. THEORETICAL BACKGROUND

5.1 Simulation Environment

Simulation environments play an important role in robotic studies. It provides a platform for testing and improving software in a controlled manner. Gazebo and Robot Operating System (ROS) are widely used for this purpose. Gazebo has created a 3D, dynamic, multi-robot environment that can recreate the complex worlds that next-generation mobile robots will encounter. Its open source status, fine grained control, and high fidelity place Gazebo in a unique position to become more than just a stepping stone between the drawing board and real hardware: data visualization, simulation of remote environments, and even reverse engineering of blackbox systems are all possible applications [6]. ROS is not an operating system in the traditional sense of process management and scheduling; rather, it provides a structured communications layer above the host operating systems of a heterogenous compute cluster [7]. The fundamental concepts of the ROS implementation are nodes, messages, topics, and services [8]. Integration between ROS and Gazebo is provided by a set of Gazebo plugins that support many existing robots and sensors. Because the plugins present the same message interface as the rest of the ROS ecosystem, you can write ROS nodes that are compatible with simulation, logged data, and hardware [8].

In the project, a ROS supported camera was used and integrated on the aircraft. OpenCV is the premier computer vision library, used in academia and in products around the world. OpenCV provides many common computer vision algorithms and utilities that you can use and build upon. ROS provides tight integration with OpenCV, allowing users to easily feed data published by cameras of various types into OpenCV algorithms, such as segmentation and tracking. ROS builds on OpenCV to provide libraries such as image_pipeline, which can be used for camera calibration, monocular image processing, stereo image processing, and depth image processing [8].

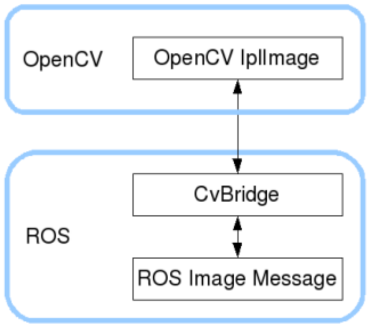


Figure 1. *ROS OpenCV integration [8]*

Figure 1 shows how ROS integrates with the OpenCV library. At the same time, in the project, the positions of the aircraft were taken through the odom node of ROS. The coordinate frame called odom is a world-fixed frame [9]. Figure 2 shows the integration of odom data with ROS.

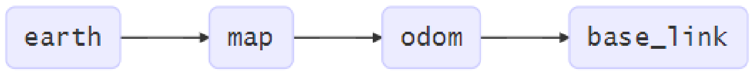


Figure 2. *ROS odom integration [9]*

5.2 Decision Making Algorithm (Fuzzy Logic)

Fuzzy logic algorithms have great importance in decision-making algorithms used in aviation. Fuzzy logic is basically an analysis and decision-making method used to address more complex and uncertain situations that cannot be defined by extreme values such as true/false.

In classical logic, propositions are accepted to be absolutely true or absolutely false. This is mathematically represented by two values: one (1) for true and zero (0) for false. This dual approach forms the basis of traditional logic systems and digital computing. However, the situation is different in fuzzy logic; Here, the degree of accuracy can take an infinite number of values between zero (0) and one (1), instead of being absolutely right or absolutely wrong. This approach allows to better express uncertainties and intermediate states, thus helping to model complex and dynamic systems more realistically. An example illustrating the working principle of fuzzy logic is provided in Figure 3.

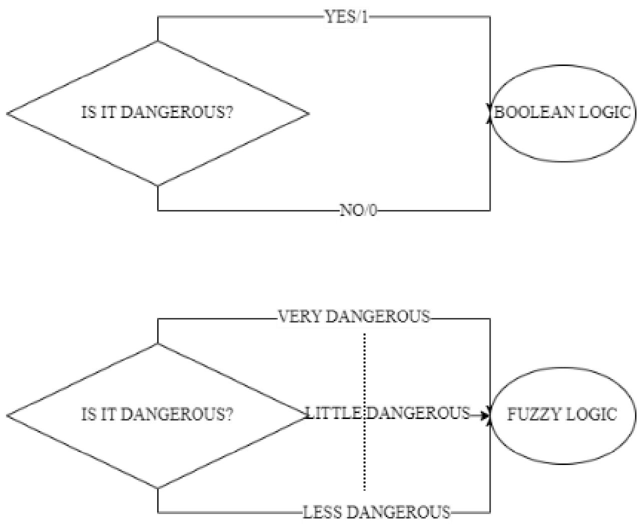


Figure 3. *Fuzzy Logic System*

A Fuzzy Inference System (FIS) transforms input data into output data using fuzzy logic principles to handle uncertain or imprecise information. The system comprises four main stages: fuzzification, inference, rule base, and defuzzification. In the fuzzification stage, a specific input value is transformed into fuzzy sets through corresponding membership functions. The membership of an element x in a fuzzy set A in a universal set is expressed by **$\mu A(x) \in [0,1]$**. In this stage, for instance, the membership degrees of a distance can be calculated for categories such as very close, close, normal, far, and very far.

In the inference stage, outputs are computed using the fuzzy rules defined in the system. The rules are typically expressed in the "IF-THEN" format. If x is A, then y is B. Here, the input sets A and B can be determined using the minimum operator (min) in equation (1).

$$\min\big(\mu A(x), \mu B(y)\big) \tag{1}$$

Outputs determine how strongly each rule is applied. Another method used in this stage is the combination of various rules using the maximum operator (max) in equation (2) among fuzzy sets.

$$\mu\mathrm{R}(\mathrm{z}) = \max\,(\min\big(\mu A1(x), \mu B1(y)\big), \min(\mu A2(x), \mu B2(y)), \dots, \min(\mu An(x), \mu Bn(y))) \tag{2}$$

In the defuzzification stage, a single crisp output value is obtained from the combined fuzzy sets. The most commonly used defuzzification method is the calculation of the centroid. In this method, the centroid of the output is found as in equation (3).

$$z = \frac{\int \mu C(z) \cdot z\, dz}{\int \mu C(z)\, dz} \tag{3}$$

This formula provides a defuzzified crisp value by calculating the centroid of the fuzzy output sets. Alternatively, other defuzzification methods, such as the peak of the fuzzy output or weighted average, can also be used [10].

The Mamdani fuzzy inference system is a commonly used method in fuzzy logic, preferred particularly because it aligns more closely with human perception. To fully define the operation of this system, various operators need to be assigned. These are:

- The AND operator used in the calculation of rule output (typically a T-norm).
- The OR operator (typically a T-conorm).
- The Implication operator used in the calculation of resultant MFs..
- The Aggregate operator used to combine qualified resultant MFs into a single overall output MF (typically a T-conorm).
- The Defuzzification operator, which transforms an output MF into a single crisp value.

The centroid of area (COA) method used to calculate the final crisp output in the Mamdani fuzzy inference system is expressed as equation (4).

$$Zcoa = \frac{\int z\ \mu C(z) \cdot z\ dz}{\int z\ \mu C(z)\ dz} = \frac{\omega 1a1z1 + w2a2z2}{w1a1 + w2a2} \tag{4}$$

In this formula, ai and zi are the area and centroid of the resultant MF, respectively, and ωi is the output of the rule. The defuzzification unit combines the outputs of all the rules generated by a specific input and produces a crisp output. Thus, the fuzzy output is converted back into a precise number. The centroid of area method is the most common way of defuzzification, where the centroid of the fuzzy set is measured and projected onto the z-axis to obtain a crisp result. The system logic is shown in Figure 4.

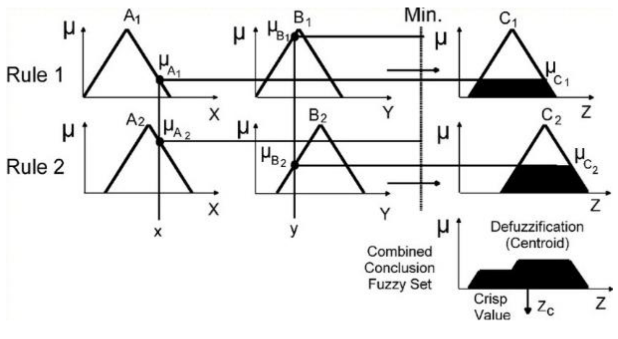


Figure 4. *Mandani Fuzzy Inferance System*

In the literature, similar studies have been conducted in air combat scenarios. For example, the rapid and accurate threat evaluation (TE) of incoming aerial targets has a significant impact on air defense. Two new generalized intuitionistic fuzzy soft set (GIFSS) methods have been proposed for the threat evaluation of certain aerial targets. Specifically, a model based on the generalized λ-Shapley Choquet integral has been proposed for threat evaluation of aerial targets. This model mitigates uncertainties by using weighted averages of different parameters to more accurately determine threat levels [11]. Similarly, in our project, parameters such as the aircraft's distance, speed, and position are evaluated to determine the threat level; however, our approach is based on fuzzy logic and uses fuzzy membership functions. Moreover, Beyond Visual Range (BVR) air combat is a significant trend in future warfare tactics. In this scenario, a fighter aircraft can attack the enemy before a direct encounter. The complexity of this tactic arises from the need to consider the target's maneuvers, the missile's characteristics, and the positional advantages of the fighter aircraft. We can approach this complexity with an algorithm that takes into account some critical parameters. These parameters are processed to determine the superiority of the fighter aircraft and the threat factor of the enemy. The data obtained from this process is used as input for a fuzzy logic algorithm to determine the optimal combat strategy [12].

5.3 Path Planning Algorithm

5.3.1 Deep Q-Learning (Reinforcement Learning)

Reinforcement Learning is one of the modern artificial intelligence algorithms used in path planning. In the reinforcement learning model, an agent improves itself based on its environment through perception and action, as illustrated in Figure 5. At each step of development, the agent receives an input (i) from the current state (s) of the environment. It then selects an action and produces this action as an output. The action changes the state of the environment, and the value of this state transition is provided to the agent as a scalar reinforcement signal (r/reward). The agent's behavior (B) should be inclined to choose actions that increase the total value of the reinforcement signal [13]. In other words, the goal is to maximize the reward in the reward-punishment system.

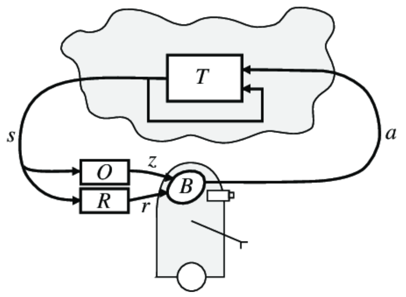
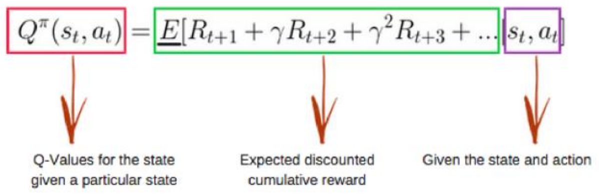


Figure 5. *Standard Reinforcement Learning Model*

In Deep Reinforcement Learning, a deep neural network is used to estimate Q-values. Q-values are estimated values of the expected total reward for each state-action pair. These Q-values represent the expected reward an agent can obtain by taking any action in a given state. The deep neural network is used to predict these Q-values and assist the agent in selecting the best actions. Q-learning aims to choose actions that maximize the sum of the immediate reward and the value of the subsequent state. While performing action A in state S, it leads to states where the reward is r. Figure 6 shows the formula for calculating the Q-value. [14]



$$Q^{\pi}(s_t, a_t) = \underline{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | s_t, a_t]$$

Q-Values for the state given a particular state
Expected discounted cumulative reward
Given the state and action

Figure 6. *Q-Value*

The decision-making steps of the Deep Q-Learning algorithm are outlined below. The block diagram summarizing these steps is provided in Figure 7.

1) Set the parameters.
    - Initialize the parameters for the Deep Q-Learning algorithm.
2) Update the parameters through training.
    - Train the model and update the parameters based on the learning process.
3) If the maximum Q-value corresponds to the airplane's behavior, calculate the Q-value.
    - Compute the Q-value for the action that corresponds to the maximum Q-value associated with the airplane's behavior.

4) Based on the $Qmax$ value, output the decision-making behavior that corresponds to the $Qmax$ value.
   - This represents the best decision-making behavior for the airplane in the current environment.
5) The environment is rewarded or penalized based on the behavior, and the decision-making process is updated.
   - Adjust the decision-making process by providing rewards or penalties to the environment based on the actions taken.

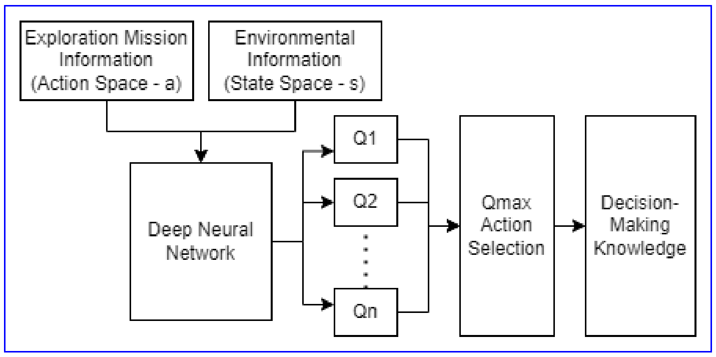

Exploration Mission Information (Action Space - a)
Environmental Information (State Space - s)
Deep Neural Network
Q1
Q2
Qn
Qmax Action Selection
Decision-Making Knowledge

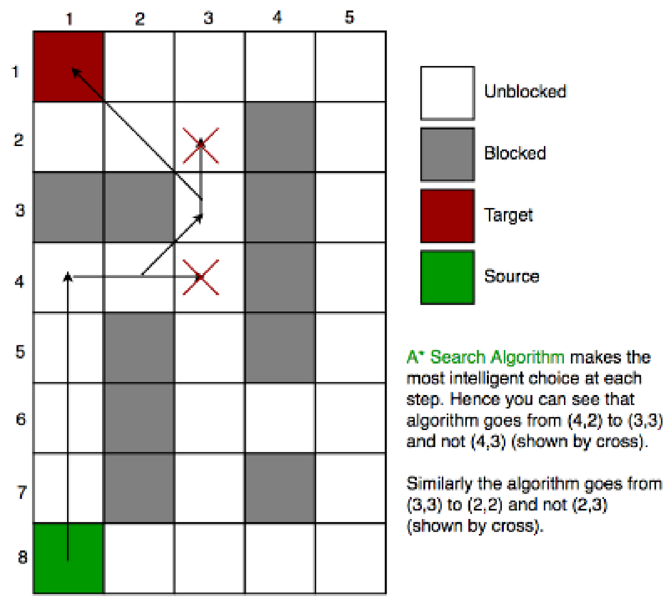
Figure 7. *Block diagram of the Deep Q Learning based decision making model [15]*

5.3.2 A* Algorithm

A* is a search algorithm that has long been used in the pathfinding research community. Its efficiency, simplicity, and modularity are often highlighted as its strengths compared to other tools [16]. The A* algorithm, unlike other algorithms, has brains. So, it is a really smart path finding algorithm compared to other algorithms. What A* Search Algorithm does is that at each step it picks the node according to a value-‘f’ which is a parameter equal to the sum of two other parameters – ‘g’ and ‘h’. At each step it picks the node/cell having the lowest ‘f’, and process that node/cell [17]. Below are the steps of the A* algorithm.

1) An open list is initialized.
2) A closed list is initialized. The starting point is added to the open list.
3) Until the open list is empty, the node with the smallest 'f' value is found in the open list. This node is called 'q'. The value 'q' is extracted from the open list.
4) If target and 'q' are equal, the algorithm is stopped.

5) If not, 'g' and 'h' values are calculated again. This can be done in many ways. Figure 8 gives a visual of how the A* algorithm works.



1
2
3
4
5
6
7
8
Unblocked
Blocked
Target
Source
A* Search Algorithm makes the most intelligent choice at each step. Hence you can see that algorithm goes from (4,2) to (3,3) and not (4,3) (shown by cross).
Similarly the algorithm goes from (3,3) to (2,2) and not (2,3) (shown by cross).

Figure 8. *A* Algorithm [17]*

5.4 Object Detection and Tracking

The YOLO algorithm stands out as a critical component in many fields such as real-time object detection, autonomous vehicles, robotics, video surveillance and augmented reality. The most important feature that distinguishes it from models such as RCNN, FasterRCNN etc. is that it can provide fast results. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes [18]. The algorithm divides the image into NxN sized frames to minimize errors and make better decisions. Each frame is individually scanned to check whether an object is present or not. If the frame contains an object, it is checked whether the object's center point lies within that frame. The frame that contains the object's center point is defined as the decisive frame. [19] The frame containing the object's center point returns specific parameters. These parameters include the coordinates of the bounding box surrounding the object, the confidence score, and the object's class. The mechanism of how the YOLO algorithm works on an image is illustrated in Figure 9.

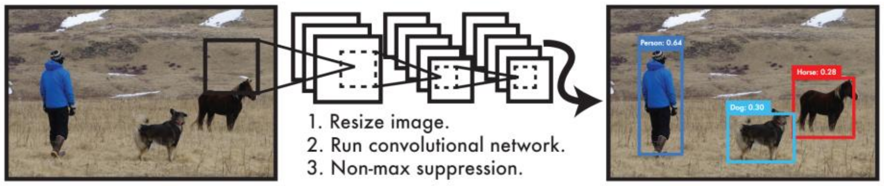


Figure 9. *The YOLO Detection System [18]*

Average Precision (AP) and confidence score are important metrics that evaluate the model's accuracy and coverage. Precision is the probability that the objects it detects are correct among all objects. Recall is how many of the detected objects are correct. These equations are included in equation 5. AP measures the relationship between precision and recall by calculating the area under the precision-recall curve. An example of this graph is shown in figure 10. This metric summarizes the performance of the model for different confidence thresholds and determines the overall performance by evaluating each object category separately. [18]

$$Recall = \frac{TP}{TP+FN} \quad Precision = \frac{TP}{TP+FP} \tag{5}$$

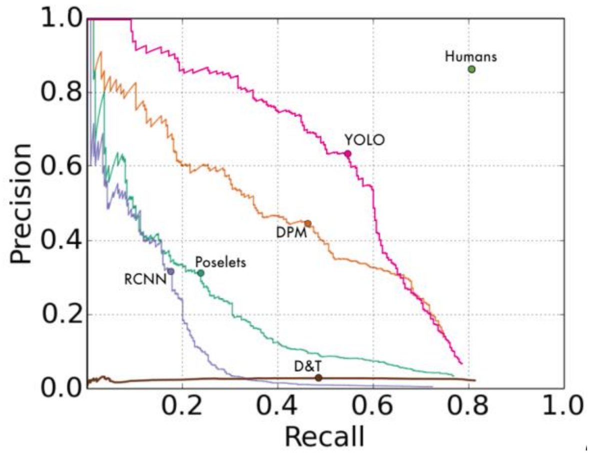


Figure 10. *mAP Graph on example Dataset [18]*

Confidence score expresses the probability of the model finding an object in a frame. This score is obtained by multiplying the probability of the object (Pr(Object)) by the Intersection over Union (IoU) value. IoU is calculated as the ratio of the intersection between the predicted box and any ground truth box to the union. The trust score takes a value between 0 and 1; 0 indicates that the object definitely does not exist, and 1 indicates that it definitely exists. The algorithm can sometimes return unnecessary parameters and draw multiple boxes. Or it can find more than one center point of an object. The box with the highest detection probability (with a high IOU value) is selected and other boxes with lower IOU values are removed. This process is called Non-Maximum Suppression. [20] The network can detect an object when there is no object in the frame, draw the bounding frame incorrectly, or make errors in classification. To prevent such situations, the loss function in equation 6 must be used. This function is created by summing the loss values in the box coordinates and confidence score value for classification [18].

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \tag{6}$$

As a result, the image divided into frames of size NxN returns a total of 5 values (4 box coordinates and a confidence score) and object class probability for each bounding box. As seen in the general yolo architecture in figure 11 The total output size on each image is expressed as NxNx(5B+C).

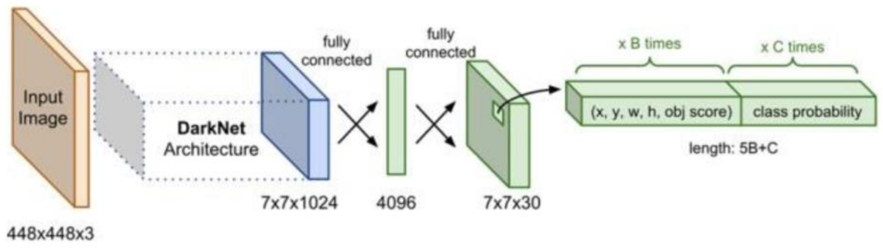


Figure 11. *General YOLO Architecture*

Within the scope of the project, the YOLOv8 model, whose architecture is shown in Figure 12, is used. Improvements over previous models are as follows: [21]

1. Decoupled Head structure: With this structure, the model can process object detection and classification tasks independently of each other. In this way, each model produces a better score on its own.
2. Activation Functions: While sigmoid is used for object detection, softmax is used for class prediction.
3. Loss Functions: CIoU and DFL are used for the object box. In this way, performance is increased when detecting small objects. Binary cross entropy is used as classification loss.
4. Anchor-Free Model: Direct anchor detection is performed and thus more accurate object detection can be achieved.

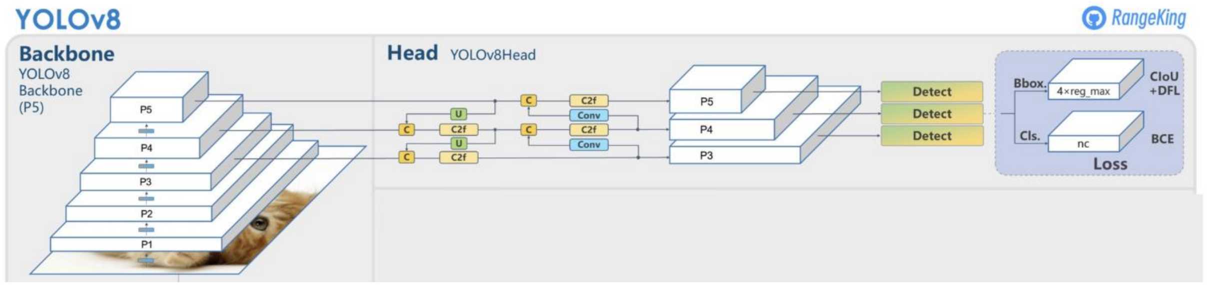


Figure 12. *Yolov8 Architecture [21]*

6. METHODOLOGY

6.1 Simulation Environment

The simulation environment was created on the gazebo tool as shown in figure 13 and figure 14. There are a total of 5 fixed-wing unmanned aerial vehicles in the environment. 4 of them are enemy planes in red. The other one is the user's plane in blue. Additionally, there are constantly moving clouds in the sky. A gray background is placed at the bottom of the environment.

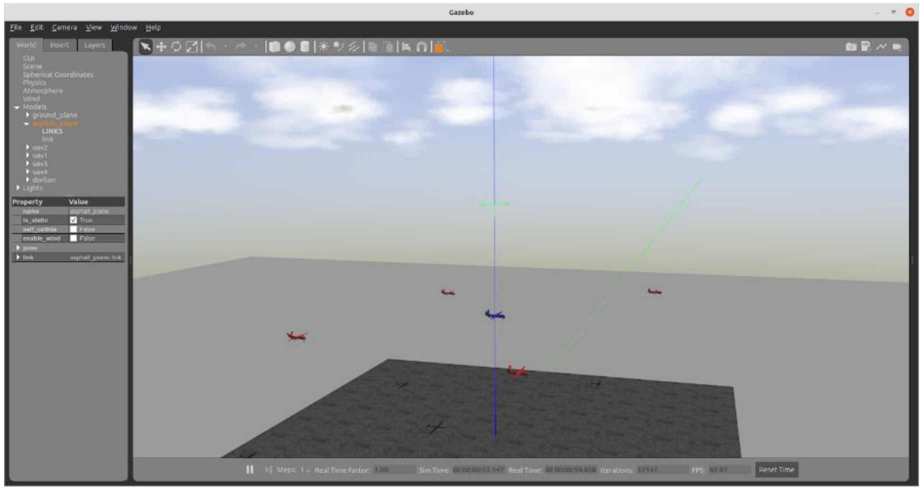


Figure 13. *Simulation Environment*

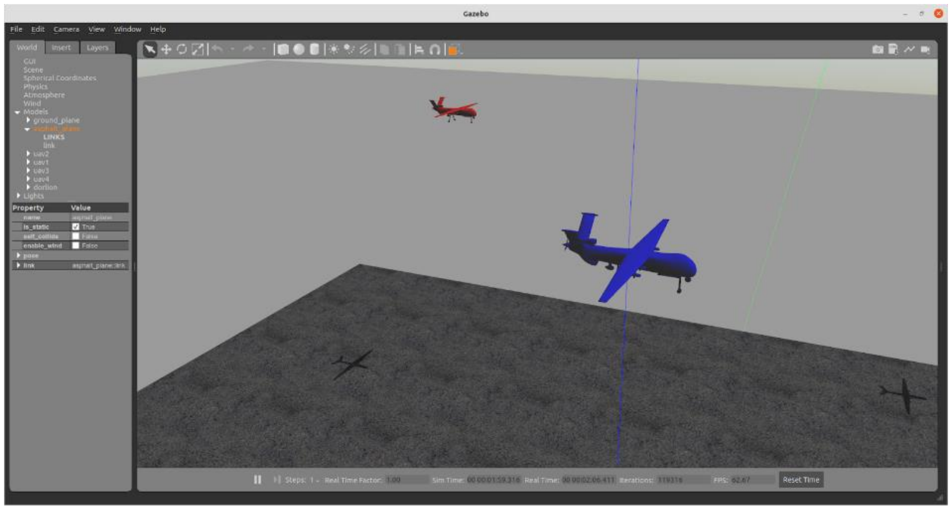


Figure 14. *Simulation Environment*

6.2 Decision Making Algorithm with Fuzzy Logic

A Fuzzy Inference System with 3 inputs was designed to find the most dangerous aircraft in the project. The system determines a danger level for each enemy aircraft using the speed, location, and position data from the aircraft.

The first of the inputs is distance information. The distance between the enemy aircraft's location and the user aircraft is determined. This location information is read through the odom topic with RosPy. A distance membership function value is calculated based on this distance. The graph of these membership functions is shown in figure 15. While the closest membership function value is calculated for the range of 0 - 6 meters, the farthest membership function value is calculated for 17 meters and above. This distance value is normalized between 0-1 to obtain more accurate results before being sent to the fuzzy system.

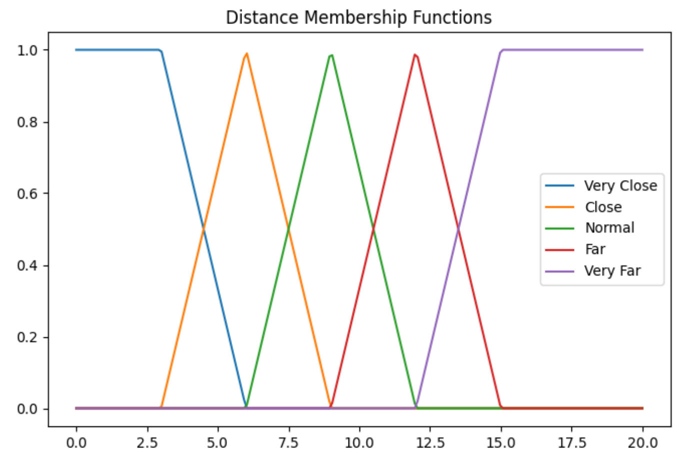


Figure 15. Distance Membership Functions

The second input value is the forward speed of the enemy aircraft at the time of calculation. With RosPy, Twist type speed data is sent and read to the aircraft via the cmd_vel topic. The speed data in this twist type is in the range of 0-1. For this reason, fast, slow and medium membership functions in the range 0-1 in Figure 16 were created.

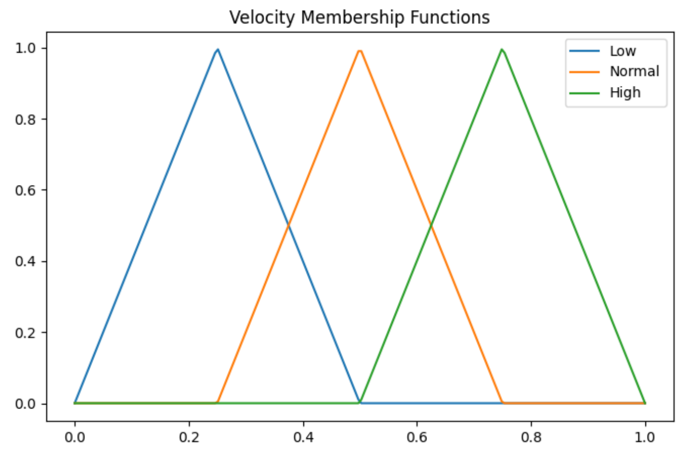


Figure 16. Velocity Membership Functions

As the final input, information about the position of the enemy aircraft relative to the user aircraft is used. An algorithm has been developed to calculate this position information. The algorithm reads the yaw angle and position information of the aircraft using odom data.

When we divide the simulation environment into four regions, the algorithm first determines in which region the enemy aircraft is relative to the user aircraft. Afterwards, the position value is normalized between 0-1 in accordance with the direction and region. And this value is used in the membership functions in Figure 17. The min-max normalization formula in equation (7) is used for this normalization process.

$$position_{normalization} = \left(\frac{position - min_{old}}{max_{old} - min_{old}} * (max_{new} - min_{new}) \right) + min_{new} \qquad (7)$$

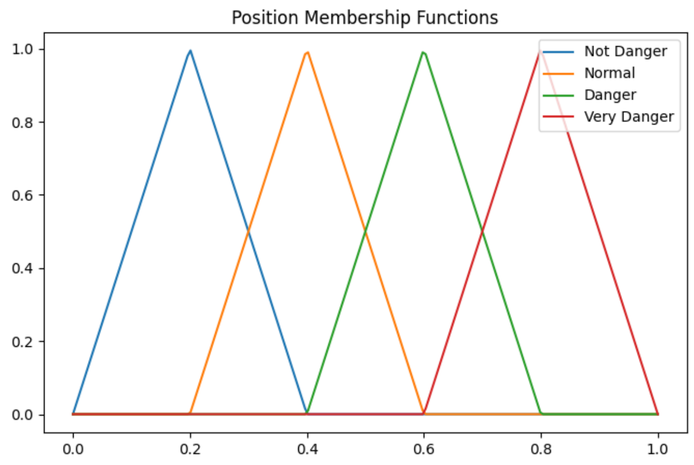
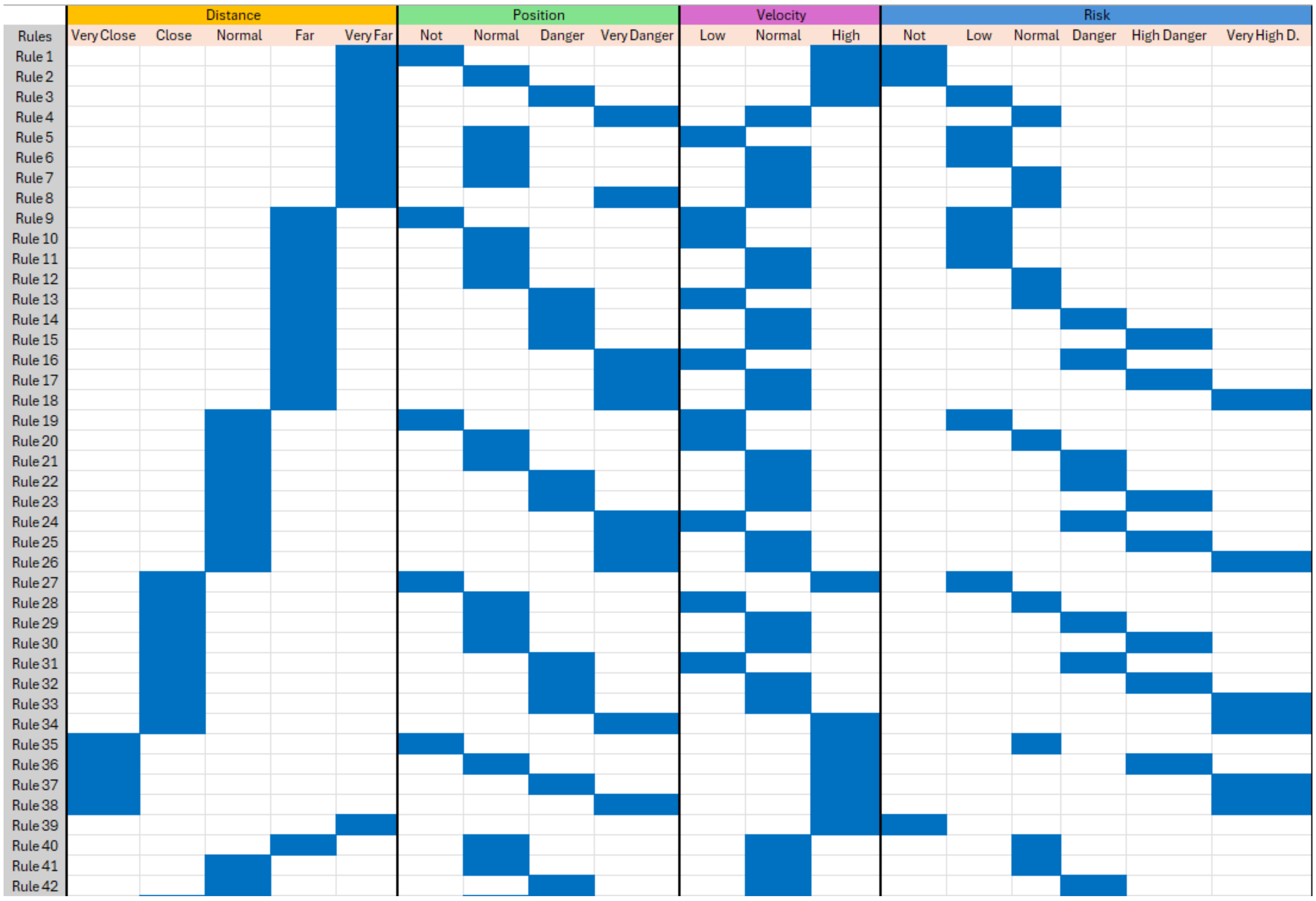


Figure 17. *Position Membership Functions*

These three membership values obtained are processed according to the 70's min-max rule system shown in Figure 18. The minimum of each value is determined, and then the maximum of those belonging to the same output membership function is determined.



	Distance					Position				Velocity			Risk					
Rules	Very Close	Close	Normal	Far	Very Far	Not	Normal	Danger	Very Danger	Low	Normal	High	Not	Low	Normal	Danger	High Danger	Very High D.
Rule 1					■	■						■	■					
Rule 2					■		■					■	■					
Rule 3					■			■				■		■				
Rule 4					■				■		■				■			
Rule 5					■		■			■				■				
Rule 6					■		■				■			■				
Rule 7					■		■				■				■			
Rule 8					■				■		■				■			
Rule 9				■		■				■				■				
Rule 10				■			■			■				■				
Rule 11				■			■				■			■				
Rule 12				■			■				■				■			
Rule 13				■				■		■					■			
Rule 14				■				■			■					■		
Rule 15				■				■			■						■	
Rule 16				■					■	■						■		
Rule 17				■					■		■						■	
Rule 18				■					■		■							■
Rule 19			■			■				■				■				
Rule 20			■				■			■					■			
Rule 21			■				■				■					■		
Rule 22			■					■			■					■		
Rule 23			■					■			■						■	
Rule 24			■						■	■						■		
Rule 25			■						■		■						■	
Rule 26			■						■		■							■
Rule 27		■				■						■		■				
Rule 28		■					■			■					■			
Rule 29		■					■				■					■		
Rule 30		■					■				■						■	
Rule 31		■						■		■						■		
Rule 32		■						■			■						■	
Rule 33		■						■			■							■
Rule 34		■							■			■						■
Rule 35	■					■						■			■			
Rule 36	■						■					■					■	
Rule 37	■							■				■						■
Rule 38	■								■			■						■
Rule 39					■							■	■					
Rule 40				■			■				■				■			
Rule 41			■				■				■				■			
Rule 42			■					■			■					■		

Figure 18. *Rule System*

Figure 19 shows these output membership functions. These class values are passed through the defuzzifier function to determine a danger level. Mamdani fuzzy inference system is used in this process. In this way, the membership values obtained are processed in the rule base and then the degree of danger is calculated.

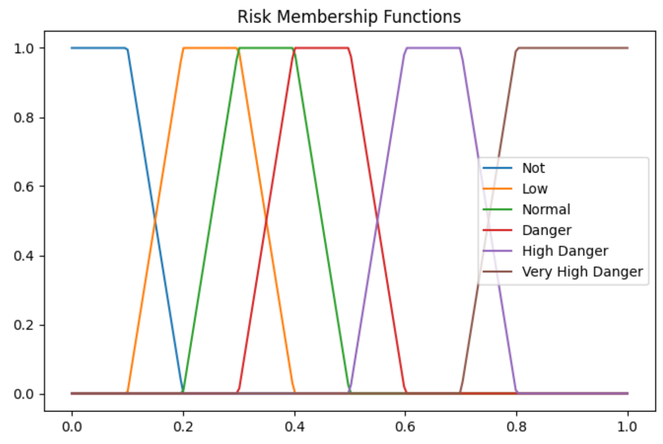


Figure 19. *Risk Membership*

6.3 Path Planning Algorithms (RL, A*)

In the project, two different algorithms were studied for the path determined when heading towards the most dangerous aircraft. The first studies were on reinforcement learning. However, since this method does not work efficiently on 3 axes, an A* algorithm has also been developed.

6.3.1 Determine Path Reinforcement Learning

At the beginning of the studies, a Deep Q RL model was developed to reach the target point safely. The model takes the instantaneous distance of the aircraft to the target as input and sends speed data to the aircraft as output.

The reward system of the model is structured according to how far the aircraft is from the target location. Each time the plane reaches the target it gets +100 points. If the plane cannot reach the target within a certain time or the agent hits an obstacle, -100 points are awarded. At the time of departure, the reward is structured according to how far the aircraft is from the target location. The main purpose of the training phase is to try to maximize the reward in the reward-punishment system. Reward values in a training phase are displayed in Figure 20.

$$r = -0.001 * \sqrt{(UAVx - ENEMYx)^2 + (UAVy - ENEMYy)^2 + (UAVz - ENEMYz)^2} \quad (8)$$

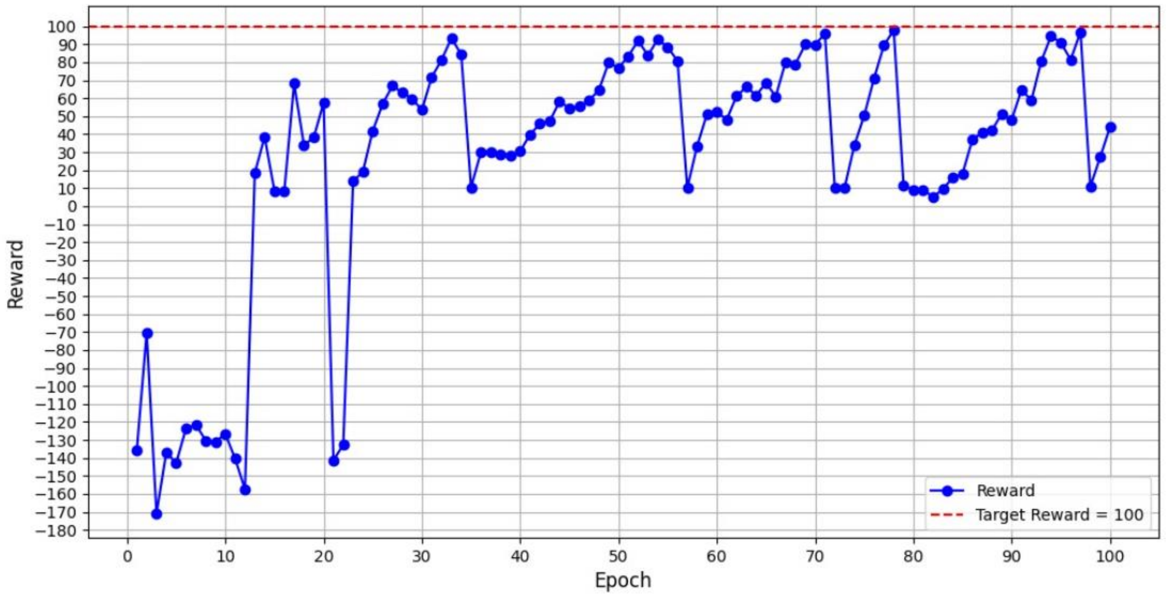


Figure 20. *Reinforcement Model Traning Proecesses*

6.3.2 Determine Path A* Algorithm

The A* algorithm takes four arguments when running: Starting point, end point, obstacles and obstacle radius. The algorithm takes the (x,y,z) position of the user aircraft as the starting point, and the (x,y,z) position of the most dangerous enemy aircraft as the ending point. Obstacles are the locations of other enemy aircraft in the environment. In addition, since it is not desired to approach these planes, an obstacle radius is determined and a path within that radius is prevented.

When a demo scenario is examined as an example, the shortest and safest path in figure 21 is calculated. In this example, the position of the user aircraft in the Gazebo is displayed as (0,0,7) and the position of the most dangerous enemy aircraft is displayed as (-5,-5,10). In this scenario, the algorithm planned a path that reaches the target by passing through a total of 5 nodes.

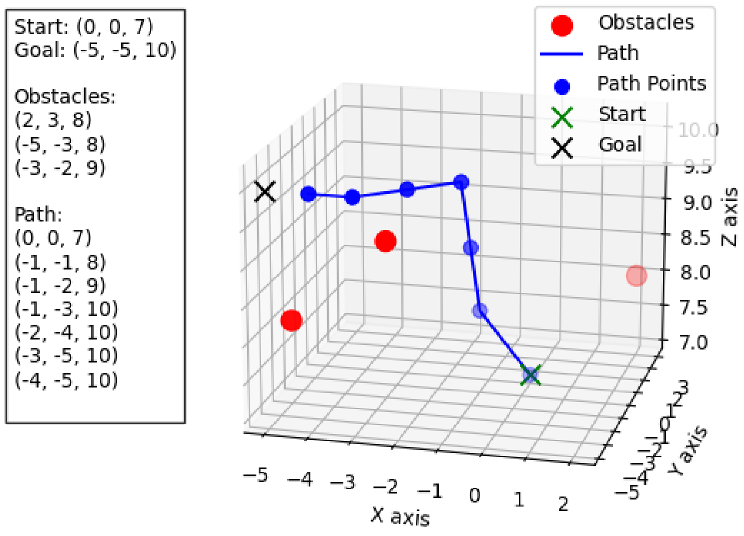


Figure 21. *Example Path with A* Algorithm*

6.4 Object Detection and Tracking

In the project, YOLOv8 architecture is used as the object detection model. The model was trained in the Gazebo simulation environment with images taken from the user aircraft's camera. The images were labeled using the LabelImg program and the model was trained with the computer's own graphics card.

During the simulation, when the user aircraft reaches the target location, the detection process begins on the image taken from the aircraft camera. The user aircraft is asked to follow the enemy aircraft for 10 seconds. A controller algorithm was developed to align the midpoint of the aircraft's camera with the midpoint of the detected aircraft. For example, if the enemy plane is to the right of the camera, the angular velocity value of the plane is updated to move to the right. When this follow-up process is done for 10 seconds, the scenario is completed successfully. An example camera view is seen in figure 22.

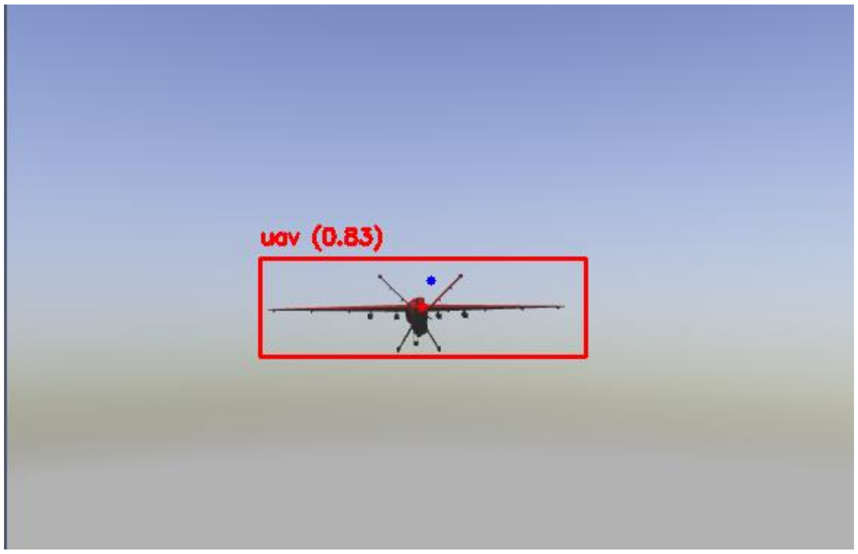


Figure 22. *Enemy UAV Detection*

6.5 Interface

An interface was developed using the PyQt library to examine the simulation scenario. The initial screen of the interface is shown in Figure 23. The status of the instant simulation phase can be examined on the simulation frame screen on the developed interface. In the boxes at the bottom of the interface, the fuzzy logic input information and the danger level of each aircraft at the moment the scenario starts are displayed.

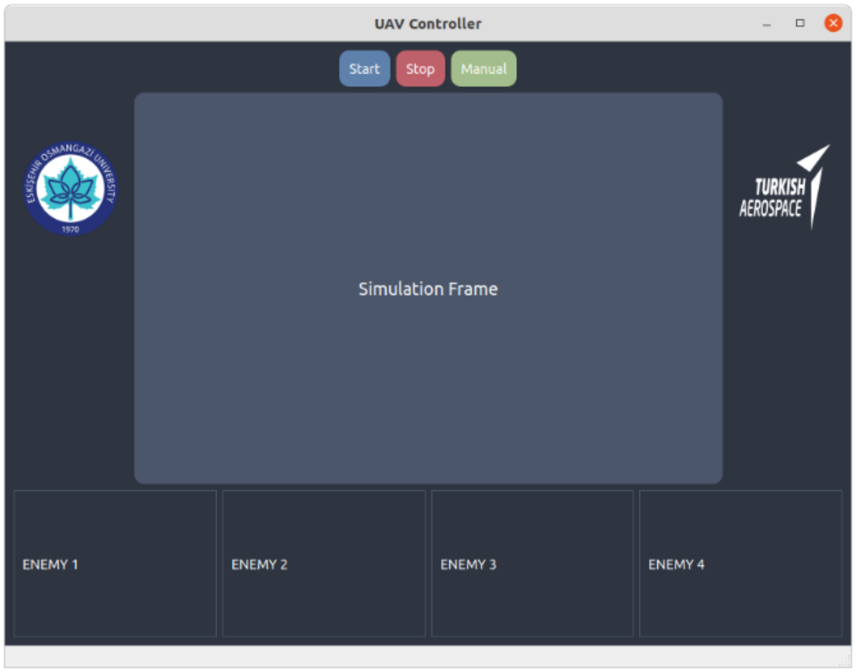
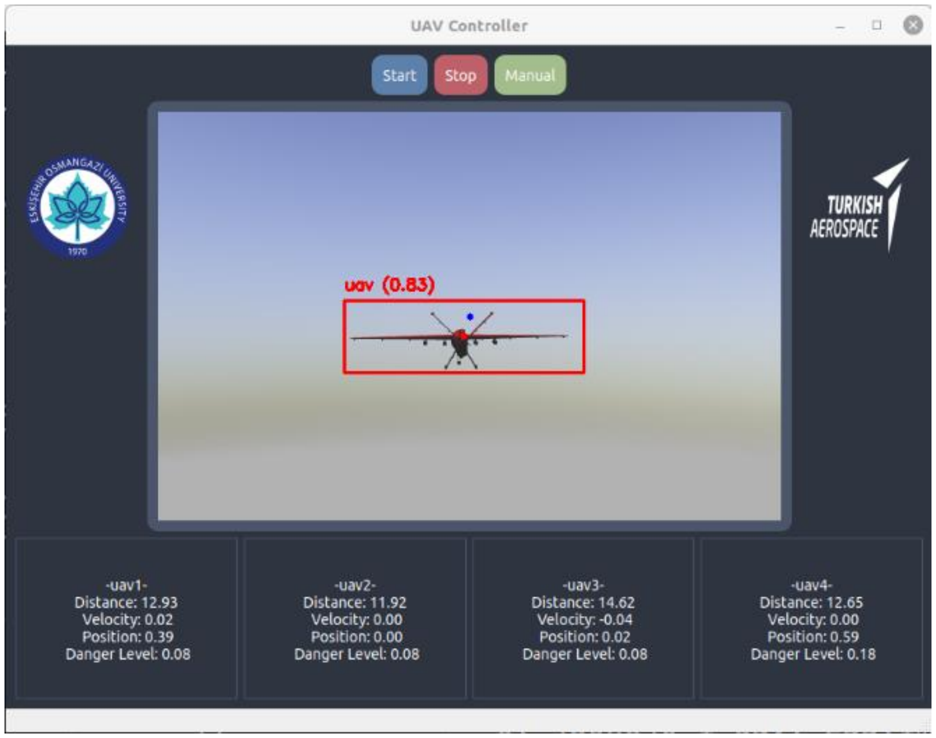


Figure 23. *Interface*

When the user plane reaches the target, as shown in Figure 24, the image from the plane's camera appears on the simulation frame screen. In this way, it can be easily displayed that the enemy aircraft has been detected and is being followed by the user aircraft. When the detection process starts, it counts down from 10 in the upper left corner of the screen and the success message is printed on the screen.



-uav1-
Distance: 12.93
Velocity: 0.02
Position: 0.39
Danger Level: 0.08

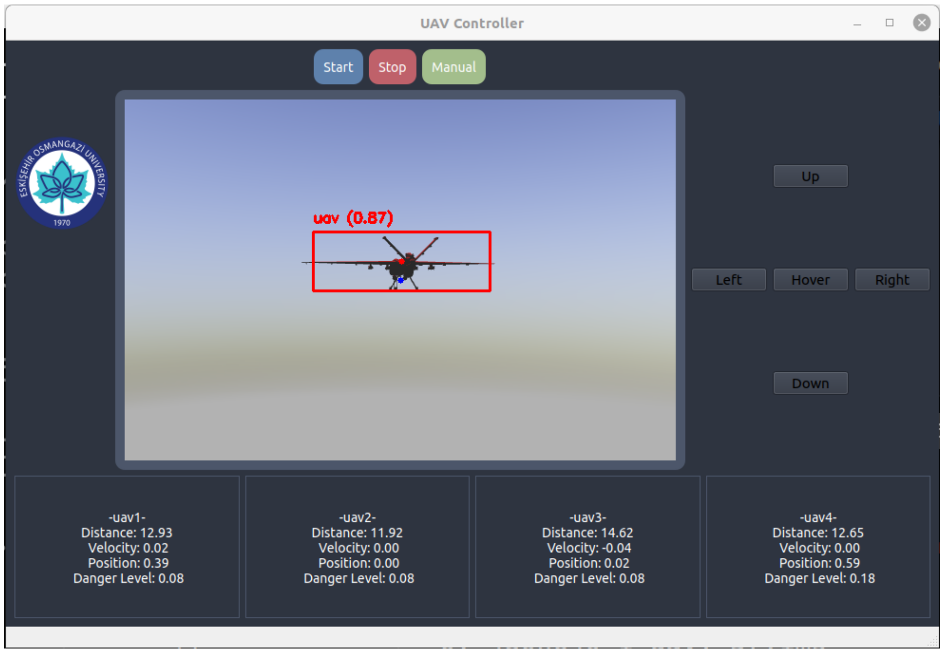
-uav2-
Distance: 11.92
Velocity: 0.00
Position: 0.00
Danger Level: 0.08

-uav3-
Distance: 14.62
Velocity: -0.04
Position: 0.02
Danger Level: 0.08

-uav4-
Distance: 12.65
Velocity: 0.00
Position: 0.59
Danger Level: 0.18

Figure 24. *Enemy UAV Detection on Interface*

There are 3 buttons at the top of the interface, as seen in figure 25. Start and stop buttons start and stop the scenario. The manual button opens a controller on the right side of the interface. With these controller buttons, enemy aircraft can be controlled and the tracking status of the user aircraft can be observed.



-uav1-
Distance: 12.93
Velocity: 0.02
Position: 0.39
Danger Level: 0.08

-uav2-
Distance: 11.92
Velocity: 0.00
Position: 0.00
Danger Level: 0.08

-uav3-
Distance: 14.62
Velocity: -0.04
Position: 0.02
Danger Level: 0.08

-uav4-
Distance: 12.65
Velocity: 0.00
Position: 0.59
Danger Level: 0.18

Figure 25. *Interface with Contoller Buttons*

6.6 Tools

Give the list of tools that used in the project

- ROS Noetic
- Gazebo / Rviz
- Scikit-Fuzzy
- OpenCV
- Matplotlib
- Numpy
- PyTorch
- PyQt / Qt-Designer
- Microsoft Visual Studio

7. EXPERIMENTS

Experiments were carried out according to the positions of the aircraft in different directions and at different distances. In the experiments, the user plane is at position 0,0,10. When the simulation starts, the planes move at random speeds for 10 seconds. Afterwards, the most dangerous plane is found and the shortest route is calculated.

7.1 Experiment 1

In the first experiment, the enemy planes are at positions (3,-7), (-4,5), (6,8), (-6,-5) and their height is 13 meters. In addition, at the beginning of the experiment, all of the planes were facing forward. That is, the direction of enemy 2 and enemy 4 planes is towards the user plane. Figure 26 shows the simulation environment.

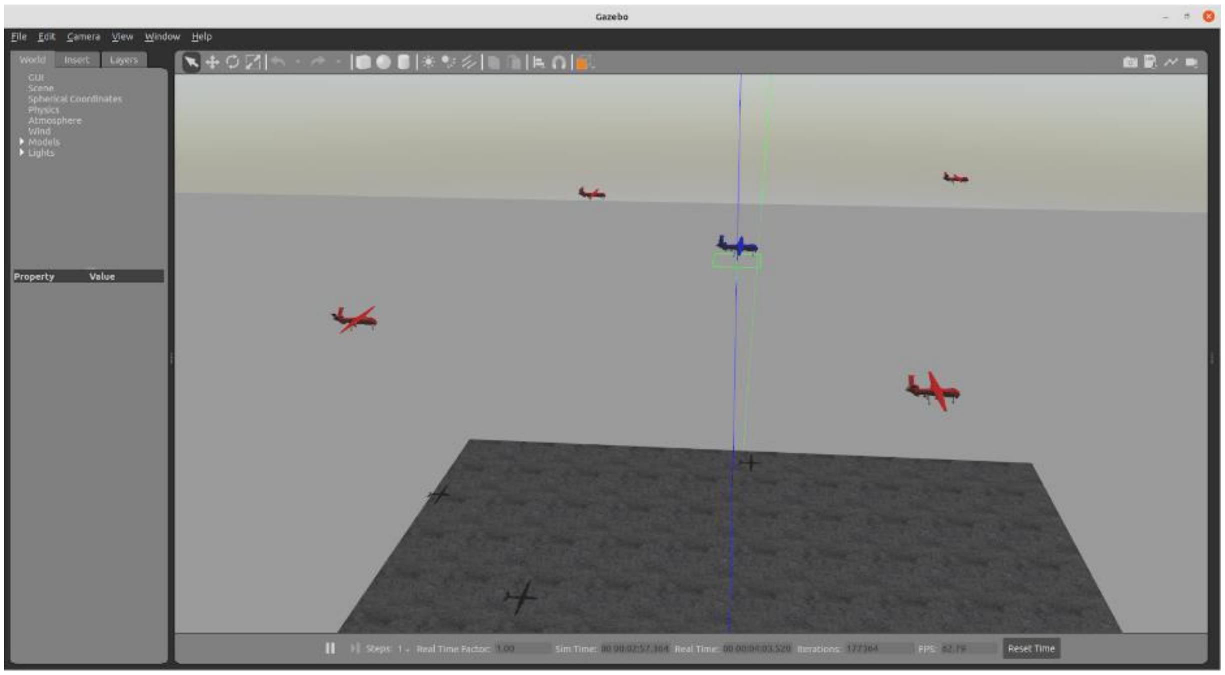
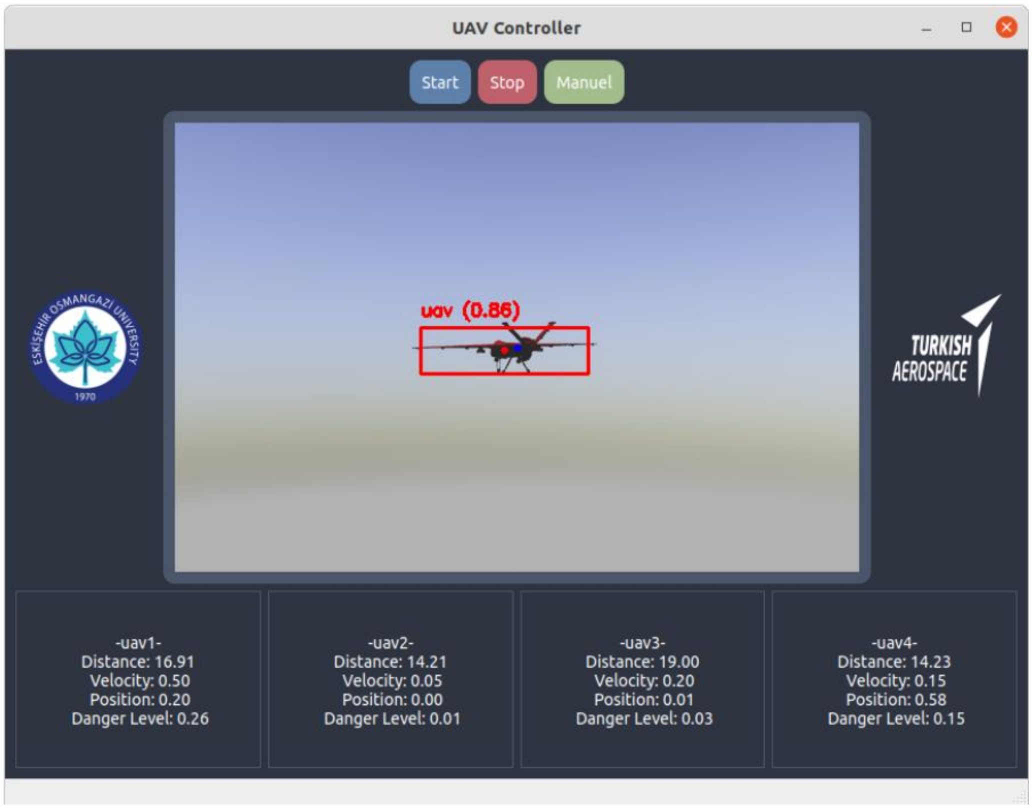


Figure 26. *Simulation environment*

The results are shown in figures 27 and 28. Looking at the danger levels of the enemies in figure 27, it can be seen that the most dangerous aircraft is enemy1. With the A* algorithm, a path planning was made from the user plane to the enemy1 plane as shown in figure 28. The aircraft went to the target and performed detection and tracking.



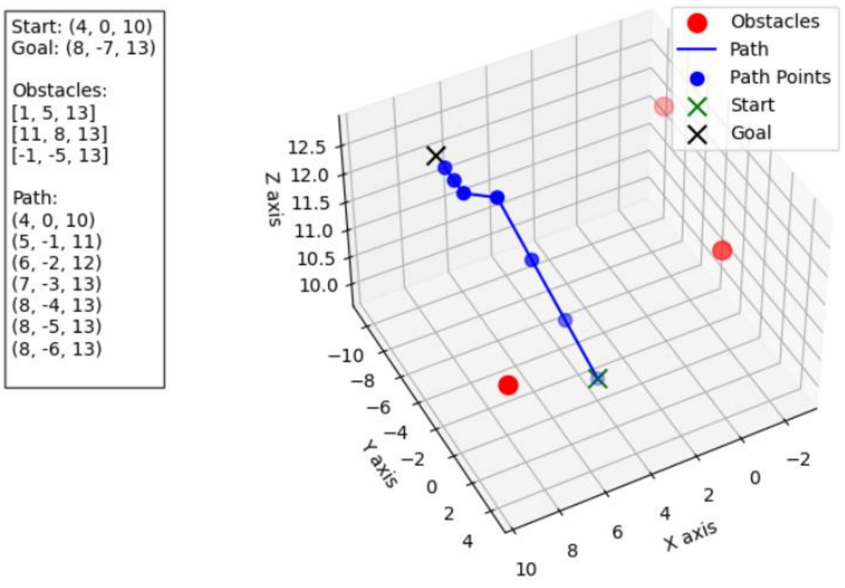
-uav1-
Distance: 16.91
Velocity: 0.50
Position: 0.20
Danger Level: 0.26

-uav2-
Distance: 14.21
Velocity: 0.05
Position: 0.00
Danger Level: 0.01

-uav3-
Distance: 19.00
Velocity: 0.20
Position: 0.01
Danger Level: 0.03

-uav4-
Distance: 14.23
Velocity: 0.15
Position: 0.58
Danger Level: 0.15

Figure 27. *Experiment 1*



Start: (4, 0, 10)
Goal: (8, -7, 13)

Obstacles:
[1, 5, 13]
[11, 8, 13]
[-1, -5, 13]

Path:
(4, 0, 10)
(5, -1, 11)
(6, -2, 12)
(7, -3, 13)
(8, -4, 13)
(8, -5, 13)
(8, -6, 13)

Figure 28. *Experiment 1 Result*

7.2 Experiment 2

In the second experiment, the enemy planes are in positions (2,-3), (-4,3), (4,3), (-3,-2) and their height is 13 meters. Enemy aircraft are brought closer to the user aircraft. In addition, at the beginning of the experiment, all of the planes were facing away from the user plane. Figure 29 shows the simulation environment.

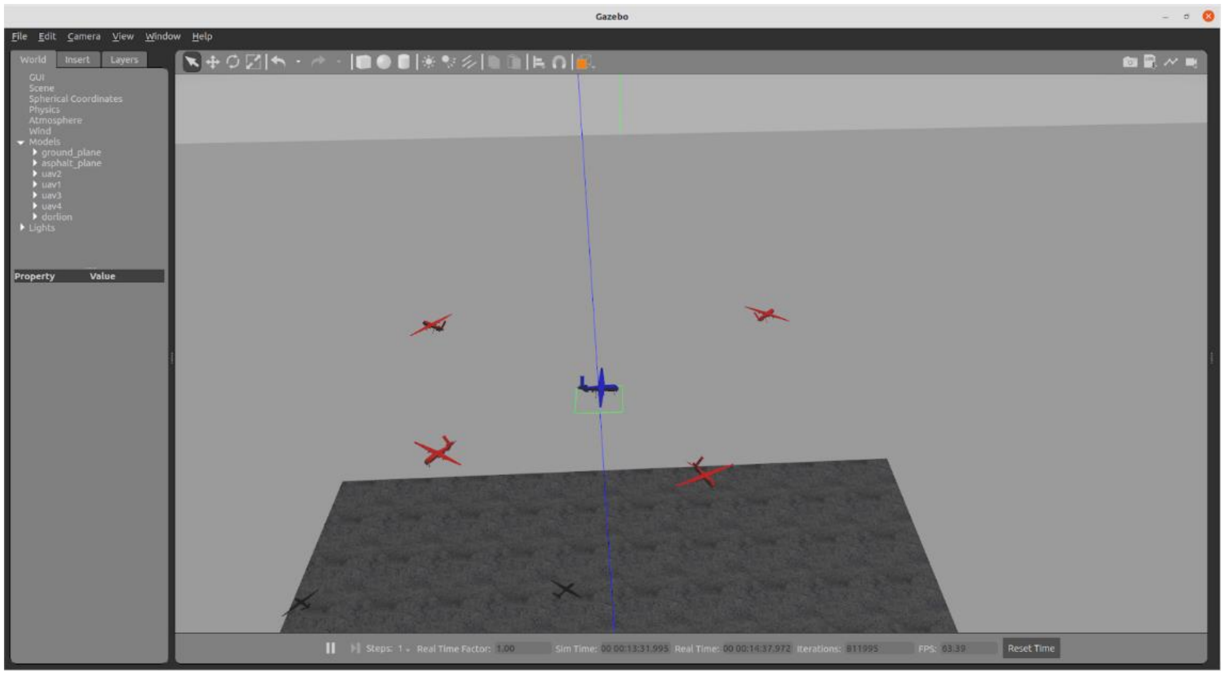
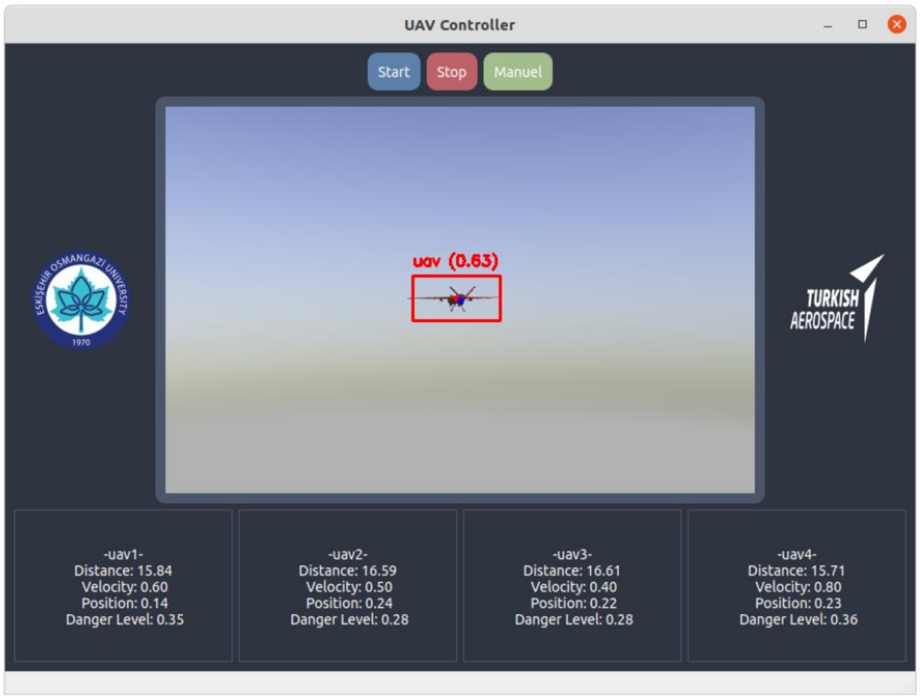


Figure 29. *Simulation Environment*

The results are shown in Figures 30 and 31. Looking at the danger levels of the enemies in figure 30, it can be seen that the most dangerous aircraft is enemy 4. With the A* algorithm, a path planning was made from the user plane to the enemy 4 plane as shown in figure 30. The aircraft went to the target and performed detection and tracking.



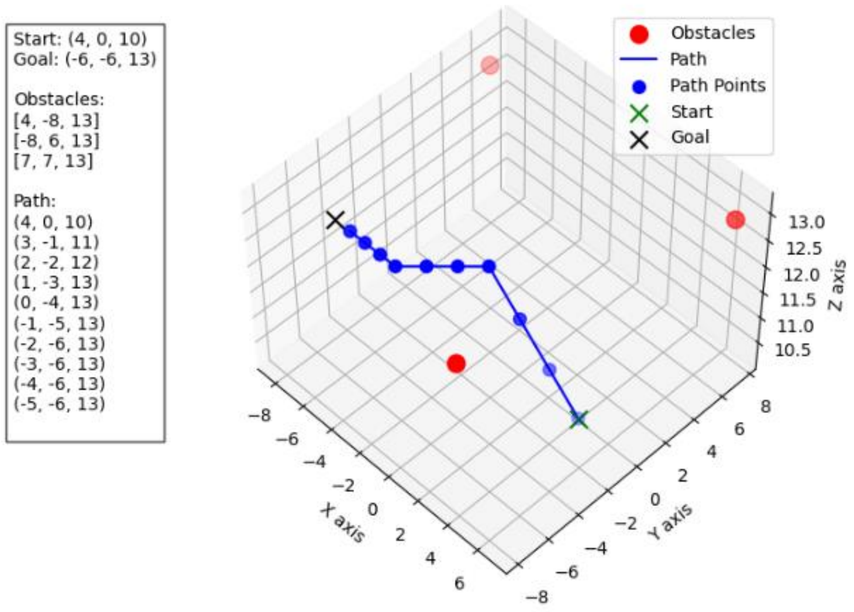
-uav1-
Distance: 15.84
Velocity: 0.60
Position: 0.14
Danger Level: 0.35

-uav2-
Distance: 16.59
Velocity: 0.50
Position: 0.24
Danger Level: 0.28

-uav3-
Distance: 16.61
Velocity: 0.40
Position: 0.22
Danger Level: 0.28

-uav4-
Distance: 15.71
Velocity: 0.80
Position: 0.23
Danger Level: 0.36

Figure 30. *Experiment 2*


Start: (4, 0, 10)
Goal: (-6, -6, 13)

Obstacles:
[4, -8, 13]
[-8, 6, 13]
[7, 7, 13]

Path:
(4, 0, 10)
(3, -1, 11)
(2, -2, 12)
(1, -3, 13)
(0, -4, 13)
(-1, -5, 13)
(-2, -6, 13)
(-3, -6, 13)
(-4, -6, 13)
(-5, -6, 13)

Figure 31. *Experiment 2 Result*

8. PROJECT PLAN

Table 1 shows the steps by which the study was carried out.

Table 1. *Working Stage*

No	Name and Objectives of Work Packages	By Whom It Will Be Conducted	Time Period (..-.. Month)	Success Criteria and Contribution to the Project's Success
1	Literature Research	Burak Özdemir Ayla Bilgin	Month 1.	A literature review has been conducted, and an approach concept for the project has been planned.
2	Preparing a Simulation Environment	Şeref Karakuş	Months 2 and 3.	With this work package, the plan is to create a simulation environment that closely resembles reality. An aircraft model will also be integrated into the simulation environment. This will enable the conduct of tests and training within the created environment.
3	Decision Making Algorithms with Fuzzy Logic	Burak Özdemir Ayla Bilgin Şeref Karakuş	Months 4 and 5.	This work package aims to determine the threat levels of enemy aircraft using a Fuzzy Inference System based on aircraft speed, position, and location data. Distance, speed, and position information are obtained and processed using RosPy. The results are evaluated using the Mamdani fuzzy inference system and appropriate membership functions to determine danger levels
4	Enemy Aircraft Detection	Ayla Bilgin	Months 4.and 5.	In this work package, the focus is on selecting and installing suitable camera hardware for integrating image processing algorithms. This hardware will be crucial for implementing tasks such as detecting enemy aircraft and determining their locations using the YOLOv8 algorithm.
5	Development of Path Planning Algorithms	Burak Özdemir Ayla Bilgin Şeref Karakuş	Months 5 and 6.	In this work package, the aim was to study two different algorithms for determining the path towards the most dangerous aircraft in the project. Initially, research focused on reinforcement learning; however, due to inefficiencies observed on all three axes, an A* algorithm was subsequently developed to address these limitations.

6	Object Detection and Tracking	Burak Özdemir Şeref Karakuş	Months 6 and 7.	This work package used YOLOv8 for object detection. The model trained on images from the user aircraft's camera in Gazebo, labeled with LabelImg, and using the computer's GPU. During simulations, the aircraft tracked enemy aircraft for 10 seconds, adjusting angular velocity based on their position relative to the camera.
7	Implementation of Integration and Testing	Burak Özdemir	Month 8	This work package aims to ensure the integration and testing of all simulations in the simulation environment. This will enable the anticipation and resolution of various issues that the aircraft may encounter in a real-world environment.
8	Reporting	Burak Özdemir Ayla Bilgin Şeref Karakuş	Month 9	This work package ensures comprehensive reporting of the project process. It documents all stages, presents project objectives, methods, findings, and results clearly. Additionally, it evaluates challenges faced, measures taken, and proposed improvements in detail.

Figure 32 shows the gantt diagram of the project.

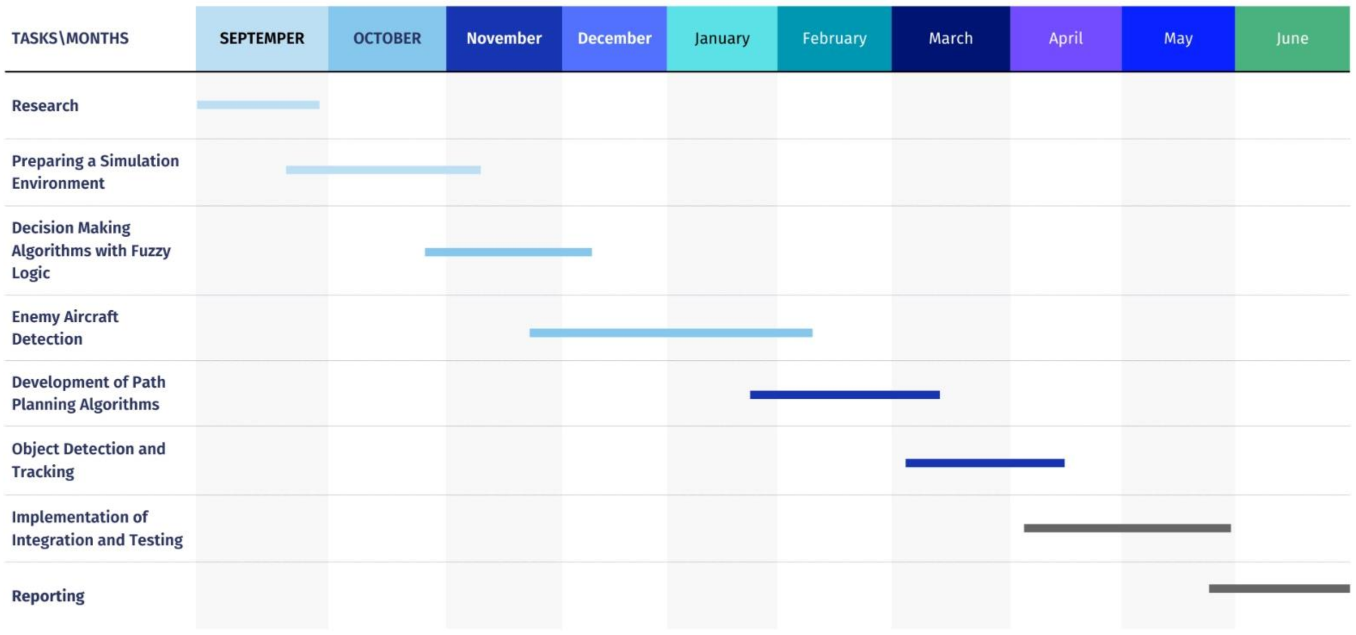


Figure 32. *Gantt Diagram*

9. CONCLUSION

Within the scope of the project, an autonomous delivery system was successfully implemented in an air combat. Thanks to the designed system, the aircraft can determine the most dangerous aircraft around it, direct it to the aircraft and follow it. Using ROS, the necessary data is obtained through the tools in the gazebo environment. These data are seamlessly transferred as input to the FIS algorithm and a hazard value can be obtained. Since a dog fight is then requested with the most dangerous plane, the user calculates a path in the most efficient way with the A* algorithm. When the user plane reaches the enemy plane, the object detection process begins and the dog fight system is activated. When the plane detects the enemy plane, it adjusts its speed data according to the position-to-position relationship between them, and if the enemy plane is too far away, it accelerates. The simulation scenario is completed successfully when the enemy aircraft is tracked for a total of 10 seconds. Thanks to this project, an aircraft will be able to safely escape from the planes around it during air communications and successfully destroy the most dangerous plane. In this way, the life safety of our pilots will be ensured. During the course of the project, efforts will be made to make these processes faster and to make map planning with reinforcement learning. Additionally, the algorithms used will be tested on development cards.

REFERENCES

[1] Grigorie, T. L. (2011). Fuzzy Controllers, theory and applications. InTech.

[2] Yılmaz, H., & Şahin, M. E. (n.d.). Bulanık Mantık Kavramına Genel Bir Bakış - DergiPark. https://dergipark.org.tr/en/download/article-file/3001236

[3] Li, Y. (2017). Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274.

[4] S. N. Bokhari, "The Linux operating system," in Computer, vol. 28, no. 8, pp. 74-79, Aug. 1995, doi: 10.1109/2.402081. keywords: {Linux;Operating systems;Workstations;Personal communication networks;Education;Hard disks;Internet;Costing;Condition monitoring;Power engineering and energy}

[5] Spolaor, S., Fuchs, C., Cazzaniga, P., Kaymak, U., Besozzi, D., & Nobile, M. S. (2020). Simpful: A user-friendly python library for Fuzzy Logic. International Journal of Computational Intelligence Systems, 13(1), 1687. https://doi.org/10.2991/ijcis.d.201012.002

[6] Koenig, N., & Howard, A. (n.d.). Design and use paradigms for gazebo, an open-source multi-robot simulator. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). https://doi.org/10.1109/iros.2004.1389727

[7] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). ROS: An open-source Robot Operating System. Retrieved from http://robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf

[8] NATO Science and Technology Organization. (2015). Advances in aerodynamic modeling and simulation. Retrieved from https://www.sto.nato.int/publications/STO%20Educational%20Notes/STO-EN-SCI-271/EN-SCI-271-06.pdf

[9] "Robot Operating System (ROS): an open-source, meta-operating system for your robot," REP 105, Version 3.5, Jan. 14, 2010. [Online]. Available: https://www.ros.org/reps/rep-0105.html. [Accessed: Jun. 14, 2024].

[10] N. Sabri, S. A. Aljunid, M. S. Salim, R. B. Badlishah, R. Kamaruddin, and M. F. Abd Malek, "Fuzzy Inference System: Short Review and Design," https://www.researchgate.net/profile/Naseer-Sabri/publication/280739444_Fuzzy_inference_system_Short_review_and_design/links/5900293daca2725bd71e8630/Fuzzy-inference-system-Short-review-and-design.pdf

[11] Liu, X., et al. "Threat Evaluation of Air Targets Based on the Generalized λ-Shapley Choquet Integral of GIFSS." MDPI, https://www.mdpi.com/2226-4310/8/5/144

[12] K. Ummah, H. Setiadi, H. M. Pasaribu, and D. Anandito, "A Simple Fight Decision Support System for BVR Air Combat Using Fuzzy Logic Algorithm." ResearchGate, https://www.researchgate.net/publication/347877567_A_Simple_Fight_Decision_Support_System_for_BVR_Air_Combat_Using_Fuzzy_Logic_Algorithm

[13] A. P. Pope, J. M. Banks, C. T. Hoy, and T. J. Durst, "Hierarchical Reinforcement Learning for Air Combat At DARPA's AlphaDogfight Trials," in *IEEE Transactions on Artificial Intelligence*, 2022, doi: 10.1109/TAI.2022.3222143.

[14] M. Langlois and R. H. Sloan, "Reinforcement Learning via Approximation of the Q-Function," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 22, no. 3, pp. 219-235, 2010, doi: 10.1080/09528130903157377.

[15] J. Xu, Q. Guo, L. Xiao, Z. Li and G. Zhang, "Autonomous Decision-Making Method for Combat Mission of UAV based on Deep Reinforcement Learning," in *Proceedings of the 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chengdu, China, 2019, pp. 538-544, doi: 10.1109/IAEAC47372.2019.8998066.

[16] Foead, D., Ghifari, A., Kusuma, M. B., Hanafiah, N., & Gunawan, E. (2021). A systematic literature review of a* pathfinding. Procedia Computer Science, 179, 507–514. https://doi.org/10.1016/j.procs.2021.01.034

[17] A* search algorithm. GeeksforGeeks. (2024, March 7). https://www.geeksforgeeks.org/a-search-algorithm/

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," arXiv preprint arXiv:1506.02640v5, 2016. [Online]. Available: http://pjreddie.com/yolo/.

[17] L. Weng, "Object Detection Part 4: Fast Detection Models," Lil'Log, Dec. 27, 2018. [Online]. Available: https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html.

[18] Terven, J. R., & Cordova-Esparza, D. M. "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS." ArXiv, https://ar5iv.labs.arxiv.org/html/2304.00501v1

[19] D. Reis, J. Hong, J. Kupec, and A. Daoudi, "Real-Time Flying Object Detection with YOLOv8," arXiv preprint arXiv:2305.09972v2, May 22, 2024. [Online]. Available: https://arxiv.org/abs/2305.09972.