UÇAN ARABA SİMÜLASYON YARIŞMASI
FİNAL TASARIM RAPORU
LİSE KATEGORİSİ

Proje Adı: Vecihi

Takım Adı: HezarFen

Takım ID: #582510

Başvuru ID: #3062379

TABLE OF CONTENTS

1. TEAM CHART AND TASK DISTRIBUTION

Our team consists of experienced members who have previously worked in the fields of Python, algorithms and Riders. Tasks are distributed according to each member's area of expertise, and it is aimed to use individual competencies in the most effective way. Thus, it is aimed to achieve optimal results by revealing the strengths of each member. In addition, our team adopts a holistic approach and exhibits a coordinated and harmonious working environment. Team introduction and task distribution scheme is presented in Diagram-1.

Scheme-1: Team Presentation Scheme

2. SCENARIO TEST WITH SANDBOX INTERFACE

2.1 Designing Scenario with Riders SandBox Interface

Within the scope of the project, a sandbox was created in order to showcase the tasks requested from us. This sandbox was chosen to contain the most challenging variations of the tasks in order to show as many features of the code developed by us as possible during the video presentation. In addition to performing all the specified tasks, the sandbox implements the most challenging variations of these tasks in the simulation environment. Through the code we developed, detailed studies were carried out on Image-1 and Image-2 to determine the most difficult variations of the desired tasks. In Figure-1 and Figure-2, green marked areas represent landing zones, blue marked areas represent charging stations, red marked areas represent hospital zones, and purple marked areas represent both charging station and hospital zones.

Image-1: Map Region Marking

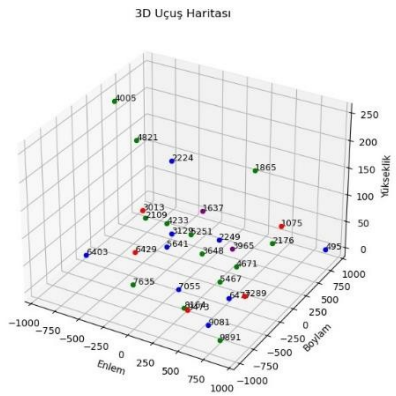


Image-1: 3d Map Region Marking

The 3-4-5-6 route determined for the Aljazari flying car requested in the competition is presented in Image-3. In this image, the orange circle represents the highway drone, the red circle represents Cezeri, the green circle represents the charging point and the white circle represents the target. The most important feature of the selected route is that it is one of the most difficult routes on the map. The reason why the selected route is a difficult route is that the highway drone, which is one of the most challenging problems encountered in the simulation, is close to Cezeri during take-off and creates traffic. This situation was effectively demonstrated with the sandbox environment. In addition, the sharp turn zones on the route and the fact that it is not possible to proceed from point 3 to point 6 without any charging process have been effective in choosing this route in order to best demonstrate the capabilities of the developed absolute code. The features set for Aljazari are detailed in Image-4 and the features of the highway vehicle are detailed in Image-5.

Image-3: Al Jazari Route **Image-4:** Aljazari Feature **Image-5:** Highway Vehicle

The main purpose of choosing the sandbox of the fire brigade drone is to perform the requested tasks and simulation within the limits of a 5-minute video presentation; outside of these time and task limitations, the absolute code developed can fulfill any task. Accordingly, the sandbox environment was designed to include harsh conditions, such as near mountain areas. In addition, the initial amount of water for the fire department is set to 20 (Figure-7), and the strength of the fire is set to 40 (Figure-8), which requires the fire department to return to the headquarters after consuming water, and demonstrates the return of the fire truck to the headquarters to replenish water or recharge. At the same time, the mountainous nature of the selected fire zone allows us to demonstrate the effectiveness of the absolute code we have written. The route of the fire brigade drone is shown in Figure 6, where the red square represents the fire and the green circle represents the fire truck.

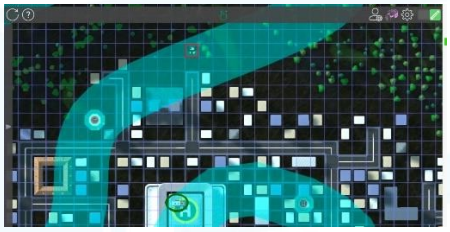


Image-6: Fire Brigade Drone Route **Image-7:** Fire Brigade Feature **Image-8:** Fire Feature

The main reason for choosing the cargo drone in a sandbox environment is that it represents one of the most challenging tasks that can be performed within the maximum duration of the simulation, which is 5 minutes of video presentation. The route of the cargo drone is shown in Figure 9, where the cargo drone is colored green and the target delivery point is colored blue. In order to create the most difficult sandbox environment suitable for the mission, the take-off and landing points of the drone must be low and surrounded by high-rise buildings, which brings the risk of hitting the buildings if the drone tries to move without enough elevation. In addition, the fact that the delivery area is located far away from the drone and in a high building, together with the flat landing area, allows the drone to exhibit its fast take-off feature. All these processes are managed by the absolute code developed by us, and even if only a single delivery point is requested, thanks to the absolute code, the mission can be carried out smoothly with single or multiple delivery drones. Image-10, cargo

drone's features; and Image-11 shows the features of the cargo delivery point in detail.

Image-9: Cargo Drone Route

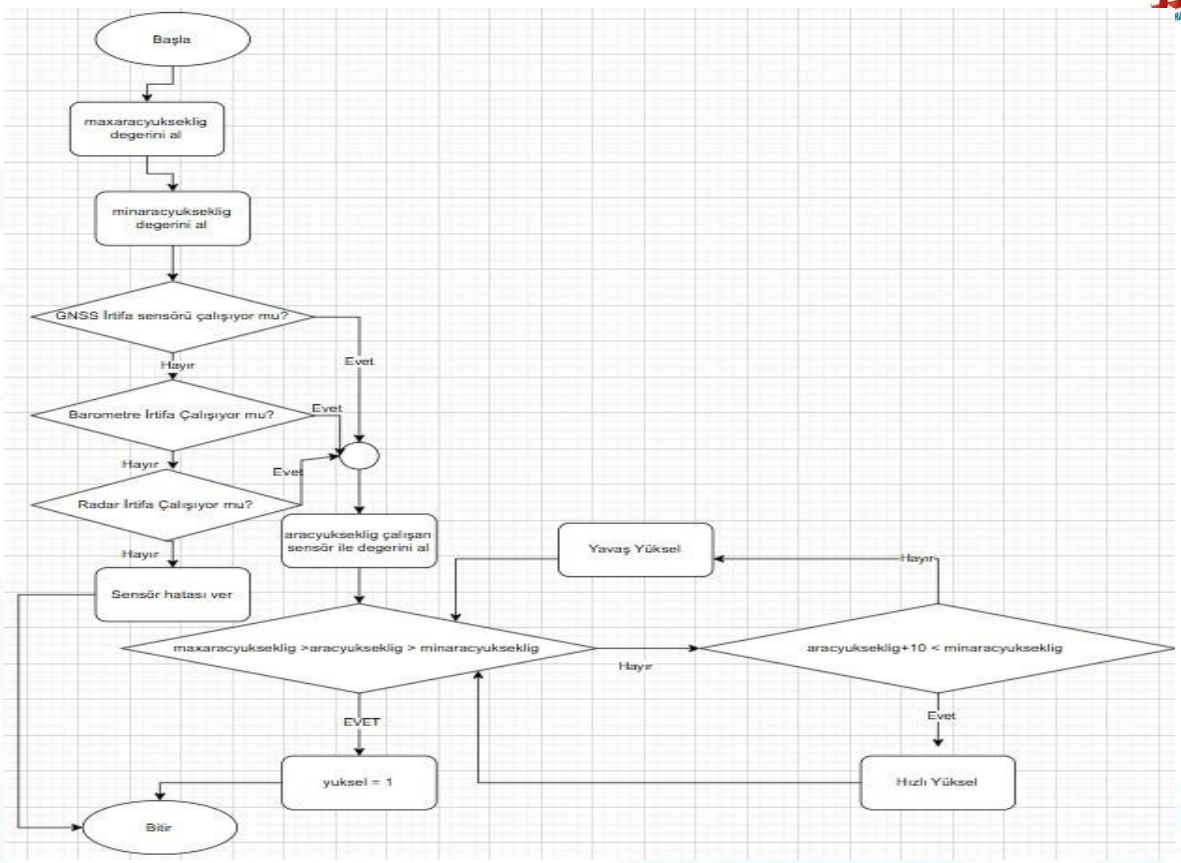
Image-10: Cargo Drone Features

Image-11: Cargo Delivery Specifications

At the same time, the fire brigade and cargo vehicles are tasked with landing back at their take-off points, while the Aljazari flying car is responsible for landing at the target point; Aljazari earns 10 points for a successful landing. In this context, the main reason for choosing the sandbox environment in this way is to ensure that the absolute code prepared is able to demonstrate the power of the sandbox and the code in the best way within the scope of the most difficult variations of the tasks requested from us.

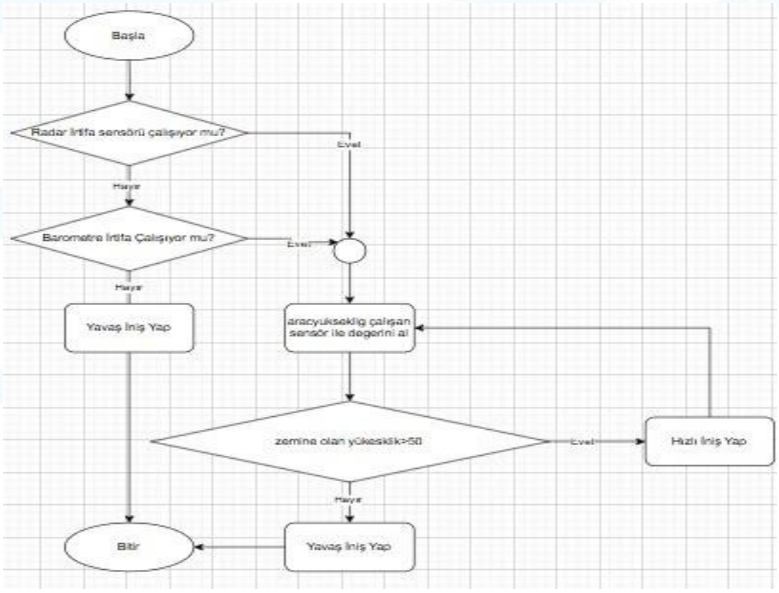
2.2 Creating the Software Flow Diagram

The take-off algorithm is the basic function that enables flying cars to take off. Thanks to this function, the flying cars can reach the desired height in a healthy and fast way. This algorithm is presented in Algorithm-1.



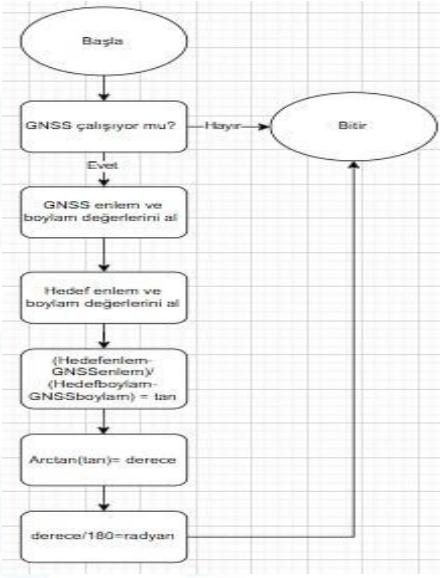
Algorithm-1: Take-off Function

The landing algorithm contains the basic function that makes it possible for flying cars to land safely, soundly and quickly. This function allows the flying cars to perform a safe landing by providing the necessary controls during the landing process. This algorithm is presented in Algorithm-2.



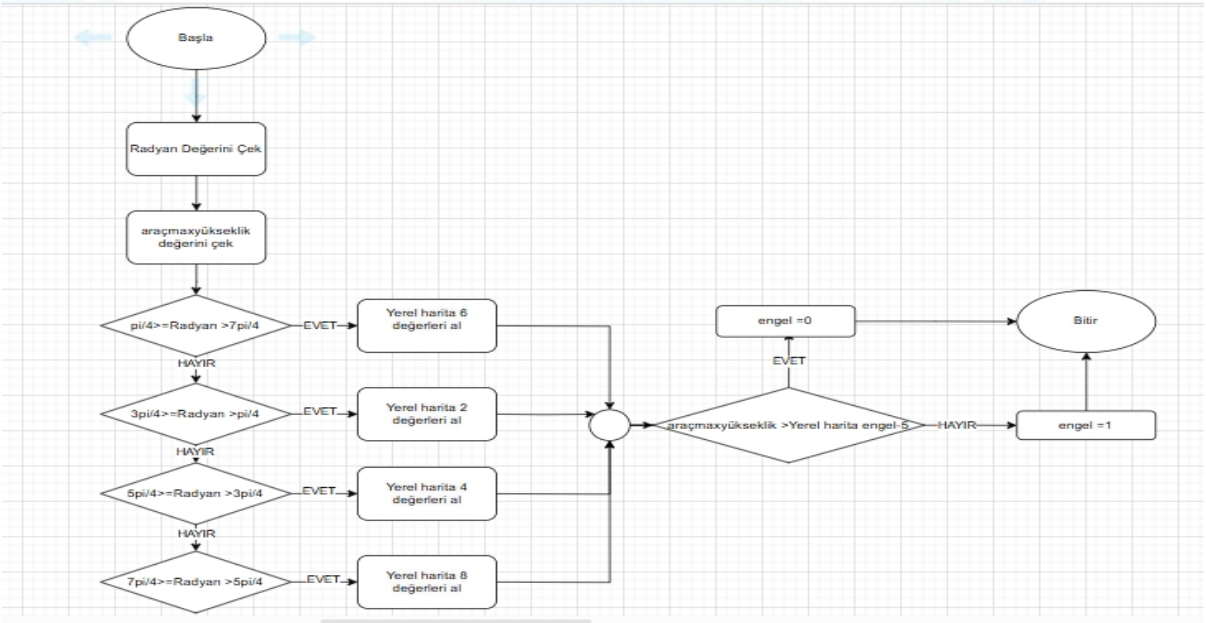
Algorithm-2: Landing Function

The target turn radius finding algorithm is used to determine the value of the turn radius that will allow the vehicle to turn towards its destination. This function evaluates the geometric relationship between the current position of the vehicle and the target position to calculate the required radian value for the turn. This algorithm is presented in Algorithm-1.



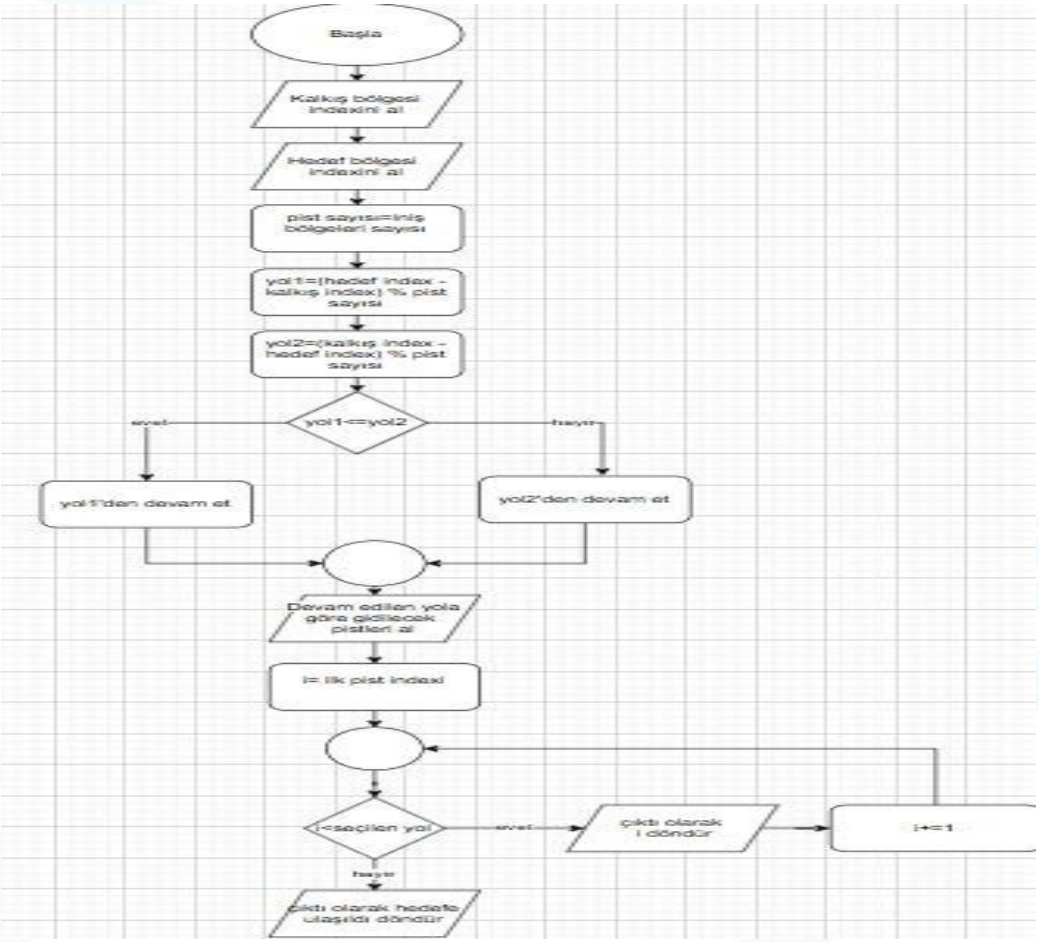
Algorithm-3: Target Turn Radian Finding Function

The obstacle detection algorithm includes a function that evaluates the maximum height of the region in the direction of the vehicle with the help of the local map and other region states and decides whether to pass through this region or not. Thus, a safe and optimal route is determined for the vehicle. This algorithm is presented in Algorithm-4.



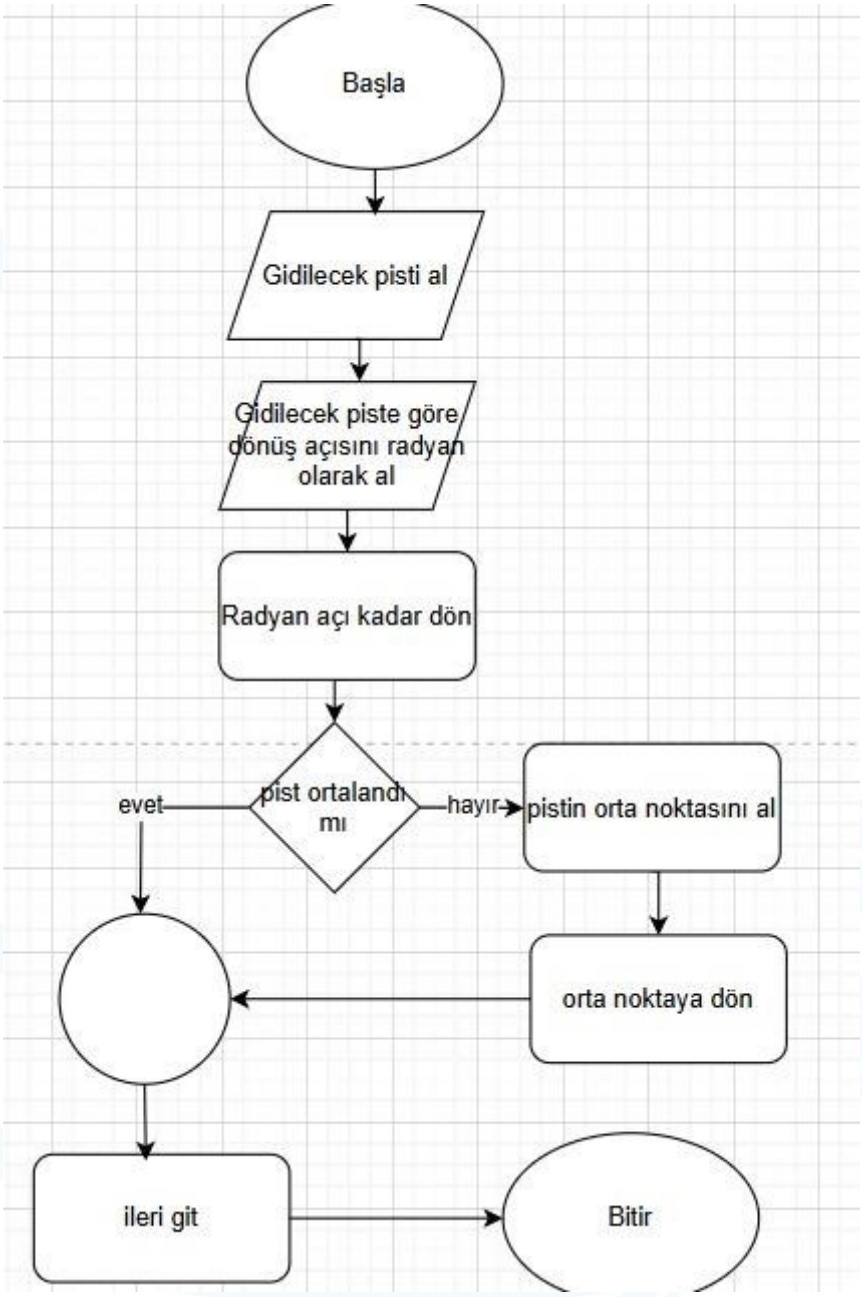
Algorithm-4: Obstacle Detection Algorithm

This algorithm, which forms the basis of the road guide that the Aljazari cruise flying car should use when traveling from one place to another, provides the information that the route to be traveled will be forward lane or backward lane. The path manager algorithm performs the appropriate lane selection based on the vehicle's current position, target position and environmental parameters, allowing the route to be followed accurately during the flight; algorithm-5 is available.



Algorithm-5: Path Manager

This algorithm, which ensures that the Al Jazari flying car reaches its destination without deviating from the set route, contains the basic function of keeping the vehicle on course. The path folower algorithm guarantees the preservation of the route by continuously controlling and guiding the vehicle along the designated path so that it does not leave the center; algorithm-6 is available.

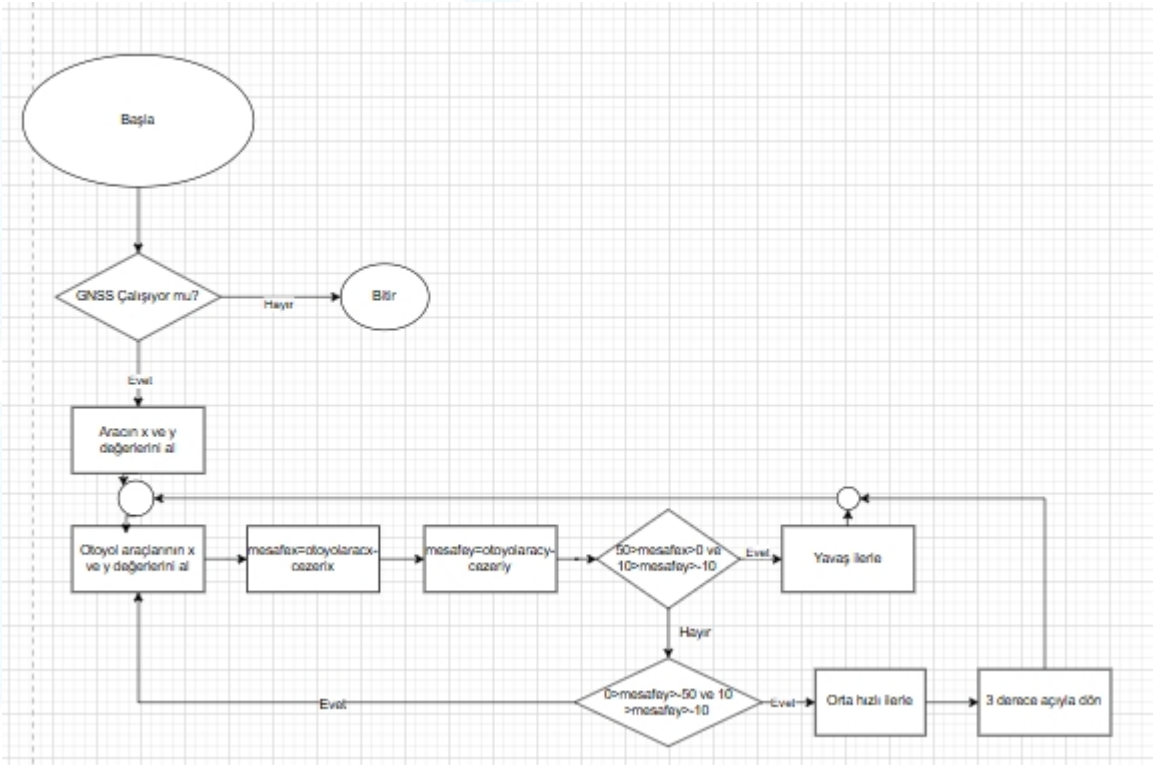


Algorithm-6: Path Folower Algorithm

Our developed algorithm provides a critical advantage in the path tracking processes of flying vehicles and stands out when compared to current methods in the literature in terms of real-time data processing, adaptive control, computational efficiency and fault tolerance; For example, our algorithm, which can dynamically adapt to environmental changes by instantaneous processing of sensor data, offers a more flexible and environmentally sensitive performance compared to the fixed parameter structures of classical PID control methods, while at the same time providing similar advantages to predictive methods such as Model Predictive Control (MPC), supported by the statement "enabling rapid adaptation to system dynamics" (EUSPA, 2021); in addition, thanks to its high computational efficiency, the high computational

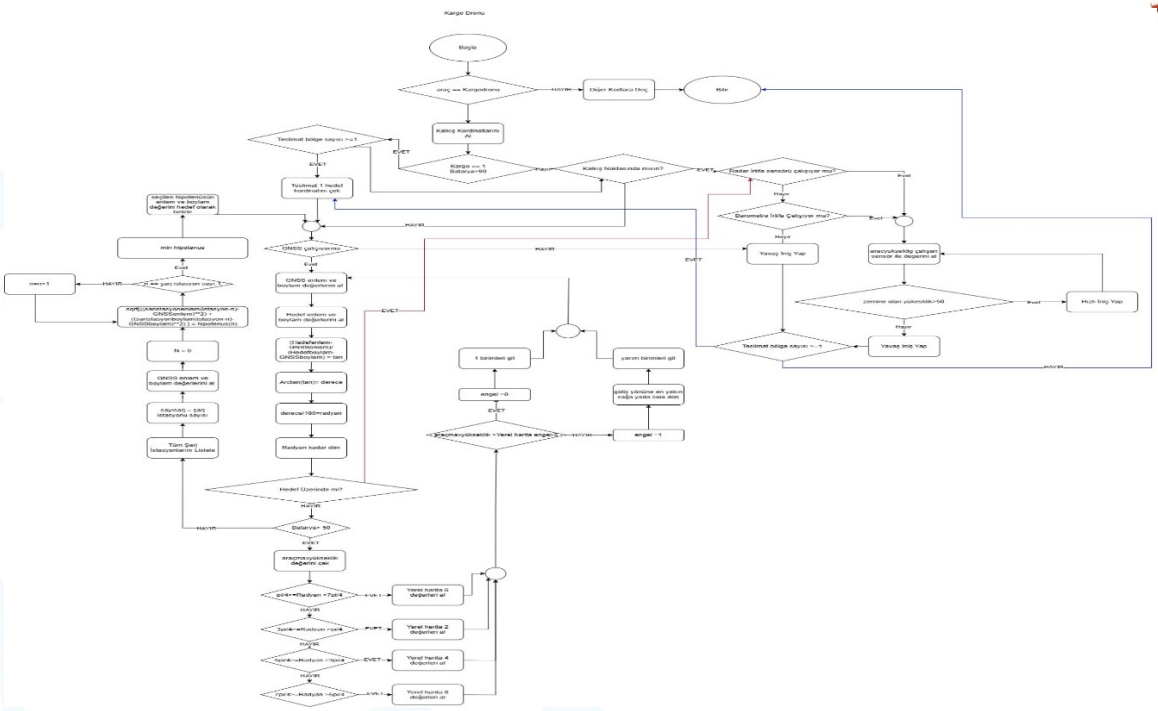
(GÜNHAN, 2014); the lack of transparency and difficulties in error analysis that deep learning-based methods face in complex environmental data analysis are overcome by the more explainable structure offered by our algorithm, which can be expressed as "superior performance in computational efficiency and fault tolerance" (Mathew, 2017); In this context, the innovative solutions offered by our algorithm, both in terms of practical application and theoretical evaluation, can be considered as an important step in optimizing the road following performance of flying vehicles.

This algorithm, which has been developed for the Aljazari flying car to detect the traffic situation around it during flight, contributes to a safe flight and accurate route following process by analyzing the presence and movements of other vehicles around the vehicle. The traffic detection algorithm analyzes environmental data to improve flight safety; algorithm-7 is available.



Algorithm-7: Traffic Detection Algorithm

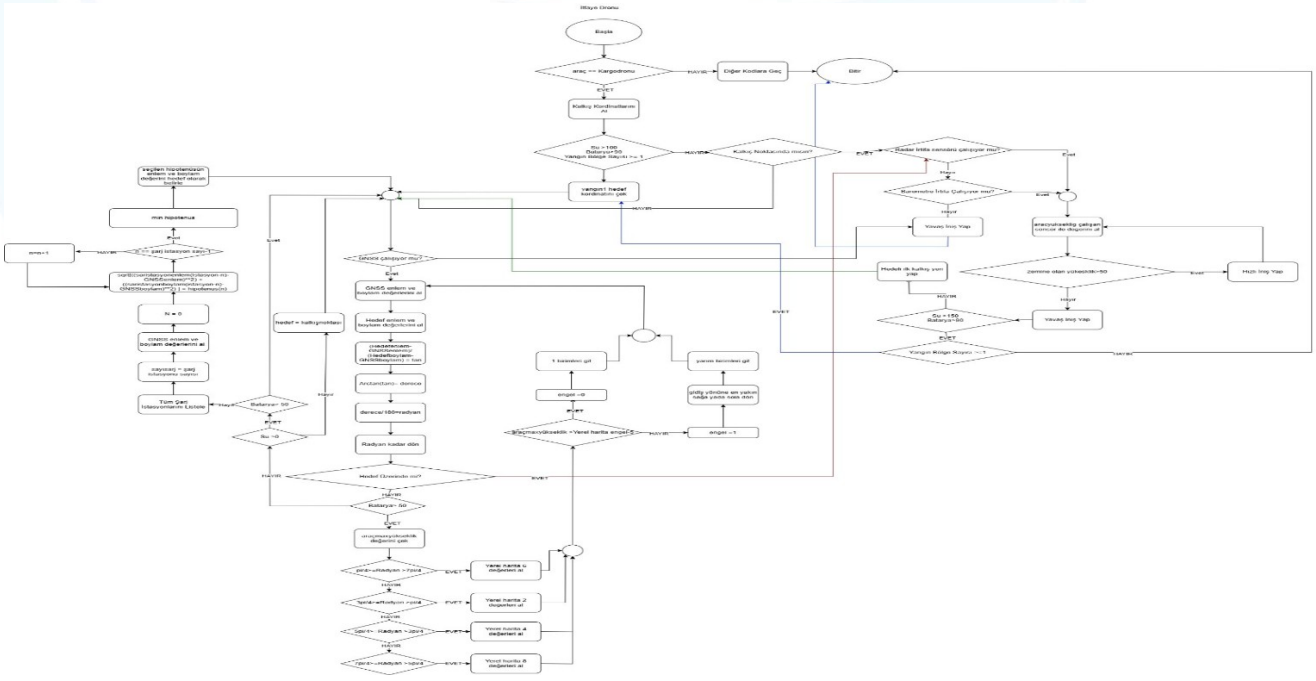
Designed to ensure that the cargo drone can perform its missions in any simulation environment, this algorithm provides an absolute code structure that can operate under variable conditions such as different battery states, package weights and weather events. The cargo flying vehicle absolute code guarantees mission continuity and system adaptability; algorithm-8 is available.



Algorithm-8: Cargo Flying Vehicle Absolute Code

This algorithm, which enables the fire brigade drone to perform its tasks smoothly and safely in simulation environments, has the ability to operate in different battery, fire supply and weather conditions. T h e absolute code of the fire brigade flying v e h i c l e

It has a flexible structure that supports its reliable fulfillment.



Algorithm-9: Fire Truck Absolute Code

The sub-parameters of the Aljazari drone are specified in these algorithms. This algorithm, which ensures that it stays on the specified route and reaches the target, is basically based on the integration of path manager and path folower algorithms. The absolute code of the Aljazari flying vehicle, which is designed to operate under variable conditions such as different batteries, routes, targets, road conditions and weather events, reveals the overall adaptability and durability of the system. At the same time, these algorithms are set to work for all vehicles that will be added in the future.

2.3 Shooting the Simulation Proof Video

The presented proof video (https://youtu.be/cEvQ3RQkKEs) demonstrates that the absolute code we have developed has been successfully executed in the prepared Sandbox environment and that all the envisaged functions have been fully realized. This video documents that all the tasks specified by us have been fulfilled, and in particular that even the most challenging variation between tasks has been successfully completed. This result is a proof that the absolute code we have prepared can show the same functionality in any similarly structured simulation environment.

3. REFERENCES

Cao, Y., Ni, K., Kawaguchi, T., & Hashimoto, S. (2024). *Path Following for Autonomous Mobile Robots with Deep Reinforcement Learning*. Sensors, 24(2), 561. https://doi.org/10.3390/s24020561

Nguyen, H., Rego, F., Quintas, J., Cruz, J., Jacinto, M., Souto, D., Potes, A., Sebastiao, L., & Pascoal, A. (2022). *A review of path following control strategies for autonomous robotic vehicles: theory, simulations, and experiments*. arXiv preprint arXiv:2204.07319.

Patnaik, A., Patel, M., Mohta, V., Shah, H., Agrawal, S., Rathore, A., Malik, R., Chakravarty, D., & Bhattacharya, R. (2020). *Design and Implementation of Path Trackers for Ackermann Drive based Vehicles*. arXiv preprint arXiv:2012.02978.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge University Press.

Paden, B., Čáp, M., Yong, S. Z., Yershov, D., & Frazzoli, E. (2016). *A survey of motion planning and control techniques for self-driving urban vehicles*. IEEE Transactions on Intelligent Vehicles, 1(1), 33-55.

Falcone, P., Borrelli, F., Asgari, J., Tseng, H. E., & Hrovat, D. (2007). *Predictive control for autonomous vehicle maneuvering*. IEEE Transactions on Control Systems Technology, 15(3), 566-580.

Schouwenaars, T., de Moor, B., Feron, E., & How, J. P. (2001). *Mixed integer programming for multi-vehicle path planning*. In Algorithmic Foundations of Robotics VI (pp. 1-14). Springer.

EUSPA. (2021, December 3). *What is GNSS?* Retrieved March 23, 2025, from https://www.euspa.europa.eu/european-space/eu-spaceprogramme/what-gnss

GÜNHAN, Y. (2014, June 27). *Identification and Correction of Inertial Measurement Unit Errors*. Retrieved March 23, 2025, from