Real-Time Drone System for Detecting, Tracking, and Following of a Mobile Robot in ROS/Gazebo

Ali Salman[1], Mohammad Dika[1], Hassan Diab[1], Michel Owayjan[1,2], Roger Achkar[1]

[1]*Faculty of Engineering and Computer Science, American University of Science & Technology, Beriut, Lebanon*
[2] *Institut de Recherche en Energie Electric de Nantes Atlantique (IREENA), Nantes University, Saint-Nazaire, France*
ahs20045@students.aust.edu.lb, mdika@aust.edu.lb, hdiab@.aust.edu.lb, mowayjan@aust.edu.lb, rachkar@aust.edu.lb

***Abstract*—Surveillance drones equipped with video transmission capabilities play a crucial role in modern security systems, with the integration of OpenCV for object detection marking a significant advancement. This study evaluates the detecting accuracy by varying the distance between the drone and the target object, addressing gaps in current research through a detailed practical insight. This research explores key aspects of drone-based video surveillance, including object detecting, tracking and real-time following, aiming to enhance understanding and methodology in Computer Vision (CV) applications with Unmanned Aerial Vehicles (UAVs). The Simulation of a surveillance drone system in Gazebo simulator within the ROS2 framework. The drone is programmed to dynamically follow and track a mobile robot, enabling systematic analysis of the object detection algorithm's performance. The findings contribute to advancing the reliability and effectiveness of drone-based video surveillance systems for future innovations in security and monitoring applications.**

Keywords— UAV, Control Systems, Drone, Object Following & Detecting, Tracking System, Mobile Robot, PID, Ros2, Gazebo Environment

I. Introduction

Drones, also known as Unmanned Aerial Vehicles (UAVs), have emerged as a rapidly evolving technology, offering immense value in various sectors such as military, surveillance, and public safety [1]. In surveillance applications, drones offer an ability to access areas that are otherwise difficult or dangerous for humans and have revolutionized surveillance operations. For instance, in military contexts, drones are deployed to detect landmines, survey battlefields, and carry out reconnaissance missions without putting human lives at risk. The precision and reach that drones offer significantly enhance the efficiency and safety of such critical operations, making them indispensable tools for modern surveillance and security tasks [2].

In the proposed system, a surveillance drone is simulated to showcase its capabilities in identifying and tracking specific targets—in this case, a mobile robot. The system utilizes the OpenCV Library for real-time image processing and object detection, enabling the drone to follow and monitor the target effectively. Additionally, to maintain stable flight during these complex operations, a PID control algorithm is implemented, ensuring smooth and accurate adjustments to the drone's movements. This combination of real-time visual processing and stable flight control makes the system especially valuable in situations where continuous monitoring, situational awareness, and security are paramount [3].

The integration of UAVs with advanced technologies like computer vision is further enhancing their capabilities. By using OpenCV for image recognition, drones can autonomously detect, classify, and track multiple objects in real time, minimizing human intervention and expanding their utility in modern surveillance applications.

II. Literature Review

Originally, drones were conceived and developed with military objectives to minimize human involvement in combat. Early versions of drones have been in use since the Civil War (1861-1865). Over time, the scope of drone applications expanded to include a variety of civilian uses such as search and rescue operations, surveillance, traffic monitoring, and more [4].

Given the significance and wide-ranging applications of drones, there is a growing interest among scientists to explore this field further. While researchers have made substantial progress, there remain numerous challenges that warrant additional investigation. Earlier approaches predominantly relied on statistical learning models for tasks like object detection and recognition. However, the evolving landscape of drone technology continues to present opportunities for more advanced research and innovation in these areas [5,6].

Surveillance tasks can be categorized based on the type and number of objects to be monitored, as well as the importance of the task. Typically, the objective of surveillance is to collect specific data about the targets, including their trajectory, and activities. Traditionally, ground-based video cameras and other sensors have been used for these tasks [7,8].

The advent of UAVs, along with advancements in computer vision and imaging technologies, has led to the development of stable and efficient UAV surveillance systems. These systems are increasingly replacing traditional methods and operating autonomously in a wide range of applications [9-11].

For cost-effective surveillance missions that do not require extensive coverage, a single UAV can be sufficient. By employing path-planning algorithms, a single UAV can sequentially monitor areas, adjusting its flight altitude to increase its surveillance range. However, for large-scale missions, a single UAV may not be adequate to meet the demands efficiently [12]. In such cases, a network of multiple

2025 International Conference on Control, Automation, and Instrumentation (IC2AI) | 979-8-3315-2960-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/IC2AI62984.2025.10932288

UAVs becomes essential. By sharing information and designing optimized allocation and deployment strategies, these UAV networks can significantly enhance mission efficiency through coordinated path planning and resource distribution [13].

III. Methodology

A. UAV Design

UAVs typically feature four propellers. When modeling its dynamics, it is assumed that these propellers are symmetrically positioned and oriented around the drone's center. The torque generated by the propellers can be calculated using the formula:

$$\tau = rFsin\theta \quad (1)$$

In this equation, τ represents the torque, r is the distance from the center of mass to the point of force application, F is the force produced by the propeller, and θ is the angle between the force vector and the arm.

The overall thrust produced by a drone can be calculated using the following formula:

$$L = T_{net} = \rho A \sum_{i=1}^{4} v_i^2 \quad (2)$$

In the above equation, T_{net} represents the total net thrust generated by all propellers, A denotes the cross-sectional area of each propeller, ρ is the air density, and v_i indicates the velocity of each individual propeller [14].

The thrust generated by each propeller contributes to the drone's lift, enabling it to hover, ascend, or descend. The cross-sectional area 'A' plays a crucial role in determining how much air is displaced by the propeller, while the air density 'ρ' affects the overall efficiency of thrust production. The velocity 'v_i' of each propeller, squared and summed, reflects the combined effect of all four propellers working together to generate the necessary lift.

B. PID Controller

Figure 1 illustrates a feedback control loop for a drone system, including a Self-Tuning PID controller in a ROS2 simulation environment. The system is designed for the drone to follow, track, and detect a target. The target input for the control loop is derived from the difference between the target position and the drone's current position, calculated through Pose Estimation. This difference is the error term $e(t)$, which is processed by the Self-Tuning PID controller. The PID control law for this system is given by:

$$u(t) = K_p e(t) + K_i \int e(t)d(t) + K_d \frac{de}{dt} \quad (3)$$

The controller's output $u(t)$ drives the drone's movement, helping it follow the mobile robot smoothly and accurately. The State Feedback loop provides continuous updates of the drone's current state, which is essential for adjusting the control output in real-time [15].

1) Roll and Pitch Control

For roll and pitch, the error is based on angular deviations from the desired roll and pitch angles:

$$u_{roll/pitch}(t) = K_p * e_{roll/pitch}(t) + K_d * \frac{de_{roll/pitch}(t)}{dt} \quad (4)$$

This controller aims to quickly correct roll and pitch errors to maintain stability, while the limit keeps the slope within safe bounds.

2) Yaw Control

Yaw control adjusts the drone's orientation to stay aligned with the target direction. The control output for yaw is:

$$u_{yaw}(t) = K_p * e_{yaw}(t) + K_d * \frac{de_{yaw}(t)}{dt} \quad (5)$$

This controller provides moderate responsiveness to yaw errors, ensuring smooth orientation adjustments (Fig. 1).

C. Robots Charchratistic & Enviorment

The ROS2 framework (Fig. 2) enables all-in-one communication between various nodes, such as the vision processing node, the control node, and the simulation environment. This integration ensures that all components operate in a coordinated and synchronized manner, allowing for precise and accurate following behavior.

By employing ROS2, developers can establish robust communication channels that facilitate real-time data exchange between nodes. For instance, the vision processing node can continuously relay visual data to the control node, which in turn adjusts the UAV's movements based on the processed information.

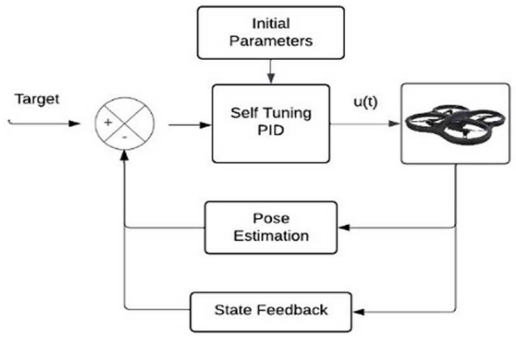

Fig. 1. Feedback Loop of the Proposed System

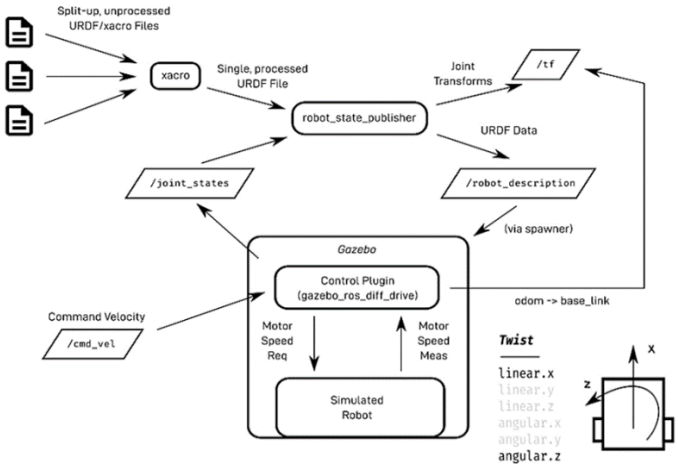

Fig. 2. Mobile's Robot Architecture Workflow in ROS2 Simulation

The workflow for simulating a mobile robot in ROS and Gazebo involves defining the robot's physical properties using URDF/Xacro files, with the state publisher managing joint states and coordinate transforms (Fig. 3). A Gazebo Control Plugin processes velocity commands from the /cmd_vel topic to simulate realistic movement based on the robot's physical characteristics, ensuring accurate updates to the robot's state through feedback like position and odometry [16].

For drones, the control system architecture integrates components that exchange real-time data on the drone's parts and movements to update its position and orientation. The control center manages activities like rest, takeoff, and velocity adjustments via the /drone/control node, while manual control is possible through the /drone/teleop_twist_keyboard node.

D. Framework & Detecting

1) Detecting & Allocation

The Parrot AR.Drone features a forward-facing and a downward-facing camera for navigation, stabilization, and video streaming. The cameras detect multiple markers, focusing on the one with the highest confidence to hover near the target. Using ROS2 nodes, video feeds are captured and processed with image transport libraries for subscribing to and publishing image topics. These cameras support tasks like marker tracking, and general image processing.

To determine the range between a drone and a certain object, the following equations are used to calculate the height from a designated distance d:

$$b = 2d[\tan\left(\frac{\varphi}{2}\right)] \quad (6)$$

$$h = 2d[\tan\left(\frac{\theta}{2}\right)] \quad (7)$$

In these equations,(b) represents the camera's point of view, (h) is the height, (d) is the horizontal distance between the drone and the target, ϕ is the vertical angle of the camera, and θ is the horizontal angle of the camera (Fig. 4).

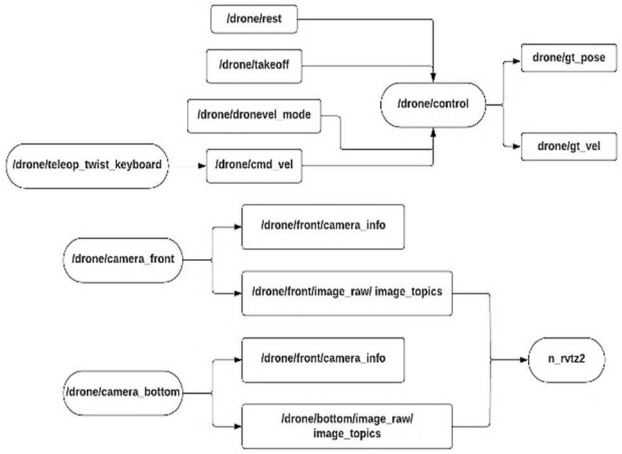

Fig. 3. Drone's Architecture Workflow in ROS2 Simulation

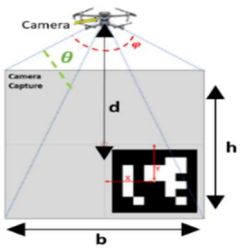

Fig. 4. Drone's Tracking/Detecting Representation

2) OpneCV Center of Contour

Fig. 5 demonstrates contour approximation using the Ramer-Douglas-Peucker (RDP) algorithm. This technique simplifies a contour by reducing the number of vertices while preserving the overall shape. The process is guided by a threshold value, which determines how closely the simplified shape should match the original contour [17].

The centroid of a shape represents the average position of all the points within that shape. For a shape made up of n unique points x1,…, x_n, the centroid can be calculated as:

$$c = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (8)$$

Image moments are specific weighted averages of pixel intensities that allow us to calculate certain properties of an image, such as area, radius, and centroid. To find the centroid, the image is typically converted to binary format before identifying its center. As the algorithm applies the simplification process, certain points are removed, creating a simplified version with fewer vertices. The formulas for the centroid coordinates are:

$$C_x = \frac{M_{10}}{M_{00}} \quad (9)$$

$$C_y = \frac{M_{01}}{M_{00}} \quad (10)$$

where C_x and C_y represent the x and y coordinates of the centroid, respectively, and M denotes the moments.

IV. Navigation Development Framework

A. Simulation

The workflow starts with launching Gazebo and configuring both the drone and target. Once the system is initialized, the camera images are captured to detect the target. If the target is not detected, the drone continues to search for it. Upon detection, the system saves the target's position and begins tracking it. If the target is lost, the drone calculates the target's last known position and moves in that direction in an attempt to reacquire it. The process continues until the tracking is either successful or the program ends.

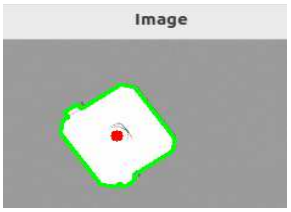

Fig. 5. Contour Center of the Mobile Robot

B. Input Source

The primary input for the system is the real-time camera feed from the UAV. A ROS2-based node is implemented to subscribe to the "camera" topic, which receives the real-time camera feed from the UAV, ensuring the feed is filtered to avoid any distortion during simulation.

Once the frames are captured, they are converted from the ROS format to the OpenCV format. This conversion is crucial for further processing and analysis, as OpenCV provides extensive capabilities for image processing and computer vision tasks (Fig. 6).

C. Process Flow

The drone's movements are dynamically adjusted based on the object's position within the video frame. This is accomplished through the use of two ROS2 nodes (Fig. 7).

The first node, known as the Vision node, processes the incoming video feed to generate the necessary control signals. It analyzes the visual data to determine the object's location and movement within the frame. The second node, called the Controller node, takes these control signals and uses a low-level controller to execute the corresponding movements.

The Vision node is managing the UAV drone's movement by tracking an object's position within a video frame. Its primary function involves calculating a direction vector, which determines how the drone should adjust its movements to stay aligned with the tracked object as,

$$d = [\frac{x}{W} - 0.5, -\left(\frac{y}{H} - 0.5\right), \theta\ ], \tag{11}$$

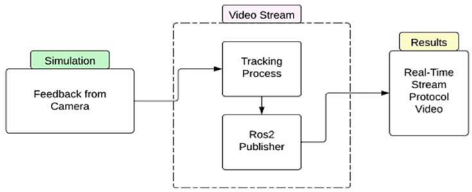

Fig. 6. Input Source Process

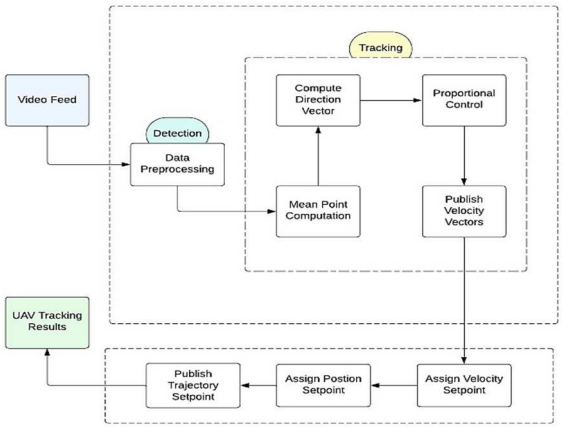

Fig. 7. UAV Tracking Process Flow

The coordinates x and y are used to find the object's central point in the video frame, allowing for precise tracking. The frame's dimensions, W and H, provide a reference for the object's relative position within the visual field. The heading angle θ, derived from the arctangent of the object's position coordinates, indicates the direction in which the drone needs to move to stay aligned with the object.

$$\theta = -\tan^{-1}(\frac{\Delta x}{\Delta y}), \tag{12}$$

In scenarios where the object is lost, the system clears the positional queue and sends a reset signal to the ROS node. This allows the drone to re-establish tracking of the lost object. The yaw control is utilized to ensure precise heading adjustments, which are calculated by:

$$\theta = -\tan^{-1}(\frac{\Delta x}{\Delta y}) * \frac{180}{\pi}, \tag{13}$$

The systematic approach combines real-time object detection, proportional control, and filtering, which enables precise and stable drone maneuvering, thus ensuring effective object tracking even in dynamic environments. When the node receives the updated data, it calculates new positional setpoints by integrating the target velocities over a discrete time interval Δt. This process updates the UAV's current position to the new target position:

$$P_{target}(t) = P_{current} + V_{target} * \Delta t \tag{14}$$

Here, V_{target} denotes the target's velocity components, represented as $[V_x, V_y, V_z]$, where correspond to the velocity along the x, y, and z axes, respectively.

V. Simulation & Experiments

A. Detecting & Tracking

The drone's camera feed is processed to identify contours, with the largest contour representing the target. The code calculates errors in the x and y directions (error_x & error_y) based on the target's position relative to the image center. These errors generate velocity commands for the drone, using the y-error for linear velocity and the x-error for angular velocity, helping the drone realign with the target.

In Fig. 8, the target moved towards the bottom left, as indicated by error_x = 118 and error_y = -19. These errors show the target is off-center, requiring adjustments to the right and upward to recenter it. The program detected two contours (contours.size() = 2), likely corresponding to the main target and a smaller region or boundary. The minimal errors suggest that target reflections may cause the detection algorithm to mistakenly identify additional contours associated with the main target.

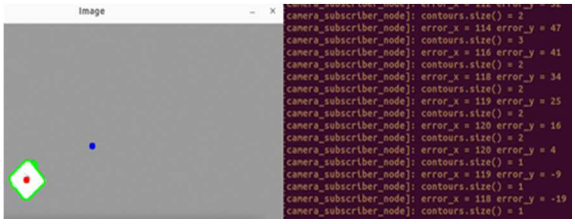

Fig. 8. Real-Time Moving Object Detection Results

Each set of the results provided in the terminal gives a real-time valuable feedback on the target's position relative to the center of the target, essential for maintaining alignment in an object-tracking and tracking application.

On the other hand, fig. 9 shows the terminal output results as error_x = 2 and error_y = 0, which indicating that the target is almost centered horizontally in the image and is well-aligned vertically. With contours.size() = 1, meaning only one primary contour was detected, likely encompassing the entire target area.

B. Movements

The Teleop_Twist_Keyboard interface allows users to control a mobile robot's movement and speed by mapping keys to specific directional commands. Operators can adjust the robot's linear and angular velocities to fine-tune its movement, including speeding up, slowing down, or stopping the robot.

When the mobile robot increases its speed, the drone may have difficulty keeping up due to the robot's sudden movements, requiring the drone to quickly reorientate itself. This can result in temporarily losing sight of the target. To address this, equations that save the last known position of the target (last_velocity_x & last_velocity_z) help the drone relocate the target by preserving its last known trajectory.

These saved velocities allow the drone to continue moving in the last direction where the target was seen. This approach increases the probability of the target coming back into the camera's field of view, as the drone doesn't stop searching but rather moves in a predictive manner based on the last known position.

1) Error Calculation

The error in the x and y directions, which represents the difference between the target's centroid and the image center, is calculated as:

$$error_x = mid_x - cX \tag{15}$$

$$error_y = mid_y - cY \tag{16}$$

where:

The coordinates cX and cY indicate the x and y positions of the target's centroid, while midx and midy represent the center of the image.

2) Velocity Calculation

Based on the errors, the drone's linear and angular velocities are adjusted to keep the target centered in the camera view:

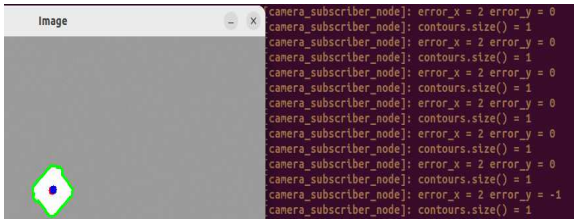

Fig. 9. Camera Node Indicating Well-Aligned Results

$$velocity\_linear.x = error_y - 0.001 \tag{17}$$

$$last\_velocity\_x = velocity\_msg.linear.x \tag{18}$$

The linear velocity in the x-direction is proportional to the y-error, guiding the drone forward or backward to align vertically with the target.

$$velocity\_linear.z = error_x - 0.005 \tag{19}$$

$$last\_velocity\_z = velocity\_msg.angular.z \tag{20}$$

The angular velocity (rotation) in the z-direction is proportional to the x-error, rotating the drone to center horizontally on the target.

last_velocity_x and last_velocity_z in equations [18, 20], are reused if the target is lost to keep the drone moving along its last known trajectory, increasing the chances of reacquiring the target.

C. Positioning

Based on the tracking information, the drone's position is adjusted in real-time to maintain a desired relative position to the mobile robot. This step is critical for ensuring the drone can follow the robot smoothly and accurately.

In the below Figure, the drone is positioned relatively close to the object, and the image center is not perfectly aligned with the target's centroid. This misalignment creates an error in both the x and y directions, which is visible in the terminal output as non-zero error_x and error_y values.

Since the contour is clearly defined, with contours.size() showing a count of 1, the object is well-detected. However, the drone's close proximity results in a larger error value, indicating that the drone may need adjustments to center itself on the target. This situation highlights that when the drone is too close, small positional shifts can lead to significant errors, requiring frequent re-centering commands.

On the other hand, Fig.11 shows the drone appears to be at a higher altitude or further away from the target. The red dot aligns almost perfectly with the blue dot, resulting in error_x and error_y values of 0. The contours.size() remains 1, confirming that the object is well-detected at this distance as well. This shows that when the drone is positioned at an optimal distance, it can more easily maintain alignment with the target, as indicated by the lack of error.

At a higher altitude, the errors tend to stabilize, which may reduce the need for frequent adjustments. Thus, this setup suggests that maintaining a stable distance allows for smoother target tracking and consistent detection without rapid error fluctuations.

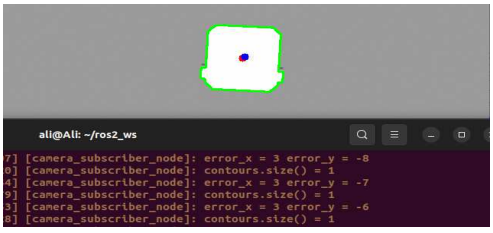

Fig. 10. First Positioning Case

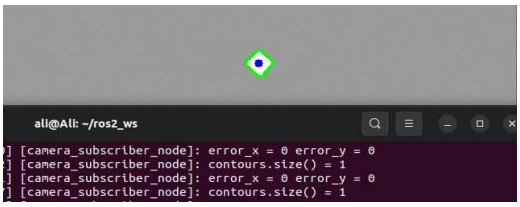

Fig. 11. Positioning Second Case

D. Simulation and Visualization

In Fig. 12, the simulation environment successfully detects multiple objects in its field of view. The main target object is the larger shape frame. The additional smaller object (obstacle) visible in the frame is also detected but is not actively tracked by the drone due to its smaller size. This filtering behavior demonstrates the simulation's capability to distinguish between objects and prioritize tracking the larger, more prominent target.

The simulation focuses on selectively tracking larger, potentially moving targets while ignoring smaller or static objects. This ensures the drone remains focused on the primary object of interest, even in complex environments with multiple detected objects. The system's ability to filter out secondary objects and maintain focus on the main target is crucial for real-world applications, confirming the success of the tracking system in this test.

VI. Conclusion & Future Work

A. Conclusion

This project successfully integrates a UAV with ROS2 and OpenCV for real-time detection, tracking, and following of a mobile robot in a Gazebo simulation. Using a robust feedback loop and ROS2's communication framework, the drone dynamically adjusts its position to maintain optimal tracking. The simulation results validate the system's effectiveness, demonstrating a seamless UAV-based surveillance method. This project advances UAV technology for various applications, including agriculture, land surveying, law enforcement, and search and rescue operations.

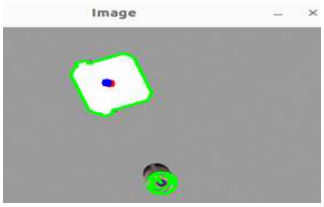

Fig. 12. Object Detecting, Tracking, and Showcasing

B. Future Work

Future work on the drone project includes integrating additional sensors like LiDAR, and thermal cameras to improve detection and environmental awareness, especially in challenging conditions. Advanced AI models, such as reinforcement learning, will enhance object detection and tracking, enabling the system to handle more complex scenarios. The development of advanced path-planning and obstacle avoidance algorithms will allow for autonomous navigation in complex environments. Transitioning from simulation to real-world testing in areas like agriculture, urban settings, and disaster response will validate performance. Expanding to multi-UAV operations, improving re-detection algorithms for occlusion, and enhancing energy efficiency will further enhance the system's capabilities for various industries and advance autonomous robotics technology.

References

[1] Adli, S. E., Shoaran, M., & Noorani, S. M. S. (2019). GSPnP: simple and geometric solution for PnP problem. The Visual Computer .

[2] Ferreira, D., & Basiri, M. (2024). Dynamic Target Tracking and Following with UAVs Using Multi-Target Information: Leveraging YOLOv8 and MOT Algorithms. *Drones*, *8*(9), 488.

[3] Tae-Yoon Kim;Jae-Hyun Kim; (2020). Implementation of unmanned aerial system for surveillance mode . 2020 International Conference on Information and Communication Technology Convergence (ICTC).

[4] Monocular Visual Autonomous Landing System for Quadcopter Drones Using Software in the Loop. (2022, May 1). IEEE Journals & Magazine | IEEE Xplore.

[5] Al-Dosari, K., Hunaiti, Z., & Balachandran, W. (2024). Drone Safety and Security Surveillance System (D4S).

[6] Hinga, S. K. (2021). Design and development of an Aerial Surveillance Security System. *arXiv (Cornell University)*.

[7] Dande, B.; Chang, C.Y.; Liao, W.H.; Roy, D.S. MSQAC: Maximizing the surveillance quality of area coverage in wireless sensor networks. IEEE Sens. J. 2022, 22, 6150–6163. [CrossRef]

[8] Fei, Z.; Li, B.; Yang, S.; Xing, C.; Chen, H.; Hanzo, L. A Survey of Multi-Objective Optimization in Wireless Sensor Networks: Metrics, Algorithms, and Open Problems. IEEE Commun. Surv. Tutor. 2017.

[9] Elmokadem, T.; Savkin, A.V. Towards fully autonomous UAVs: A survey. Sensors 2021, 21, 6223.

[10] Rezwan, S.; Choi, W. Artificial intelligence approaches for UAV navigation: Recent advances and future challenges. IEEE Access 2022.

[11] Bai, Y.; Zhao, H.; Zhang, X.; Chang, Z.; Jäntti, R.; Yang, K. Towards autonomous multi-UAV wireless network: A survey of reinforcement learning-based approaches. IEEE Commun. Surv. Tutor. 2023.

[12] Li, X.; Savkin, A.V. Networked Unmanned Aerial Vehicles for Surveillance and Monitoring: A Survey. Future Internet 2021.

[13] Wang, J.Y.; Su, D.P.; Feng, P.; Liu, N.; Wang, J.B. Optimal Height of UAV in Covert Visible Light Communications. IEEE Commun. Lett. 2023.

[14] Teja, H. S., Chawan, G., Nilay, S., Eswaraiah, D., & Jyothi, U. (2023). Design and analysis of drone propeller by using aluminium and nylon materials. E3S Web of Conferences, 391, 01032.

[15] Hoang, M. L. (2023). Smart drone surveillance system based on AI and on IoT communication in case of intrusion and fire accident.

[16] ros2_control extra bits | Articulated Robotics. (n.d.).

[17] Rosebrock, A. (2021, April 17). OpenCV center of contour - PyImageSearch. PyImageSearch.