Distributed drone networks for target search and rescue

Luca Zardini *luca.zardini@studenti.unitn.it* ID 229366

***Abstract*—This project explores a distributed system of autonomous drones designed for search and rescue missions in unknown environments. The proposed system leverages decentralized coordination, enabling drones to collaborate and share information. The architecture integrates multi-agent communication protocols, path-planning algorithms, and real-time target detection using onboard sensors and computer vision techniques. Each drone is equipped with a camera system capable of recognizing and identifying targets through real-time image processing and computer vision algorithms. Additionally, an integrated GPS module provides accurate localization within the environment, enabling efficient navigation and coordination with other drones in the system.**

I. Introduction

In recent years, autonomous drone technology has emerged as a promising solution for addressing critical challenges in search and rescue missions. Deploying unmanned aerial vehicles (UAVs) equipped with advanced sensing and communication technologies can significantly enhance the efficiency and safety of rescue operations. Distributed drone systems offer the potential for rapid large-scale environmental exploration through coordinated, autonomous behavior and can operate continuously without risking human lives.

A. Problem formulation

This project aims to design and implement a distributed drone system able to search and rescue targets in unknown environment.

Each drone operates autonomously to locate and rescue targets, while simultaneously sharing relevant information with other drones in real time. The drones must collaborate effectively to complete the rescue of each target and the mission is considered completed when all targets are rescued. A parameter r specifies the minimum number of drones required to rescue a single target.

To navigate the search area, each drone estimates its position using onboard sensors such as GPS and calculates its distance to identified targets. The simulation incorporates realistic conditions, including sensor update rates and noise. Furthermore, to emulate practical constraints, the drones are subject to a communication range limitation, ensuring that data transmission occurs only between drones within the predefined communication range, calculated as

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{1}$$

where (x_i, y_i, z_i) and (x_j, y_j, z_j) represent the positions of drones i and j, respectively.

II. Adopted models

In this section, the communication system adopted and the robot model, sensors and actuators are presented.

A. Communication system

The communication system in the developed project relies on the Robot Operating System (ROS), a widely used middleware framework in robotics. ROS provides a decentralized architecture, which is particularly beneficial for multi-agent systems like the autonomous drones in the search and rescue mission. This architecture ensures that the system components, known as nodes, can operate independently while exchanging information as needed.

ROS is based on a publish/subscribe messaging approach, where nodes that produce data (publishers) send it to specific topics, and other nodes (subscribers) access the data by subscribing to these topics. Each node in the system is dedicated to a particular function, such as sensor data processing, flight control, or mission management. By organizing communication into topics, ROS simplifies data management and allows for efficient coordination between multiple drones, even as the number of drones and complexity of the mission increases.

B. System Model

In this project, a quadrotor drone is employed due to its versatility and ease of control. Its ability to perform vertical takeoff and landing allows operation in confined spaces, while its hovering and maneuvering capabilities facilitate precise navigation and target detection in the search and rescue scenario. The quadrotor is equipped with four motors, which provide thrust and control over its movement. The state vector μ represents the position and orientation of the drone, which can be defined as

$$\mu = \begin{bmatrix} x \\ y \\ z \\ roll \\ pitch \\ yaw \end{bmatrix} \tag{2}$$

where x, y, z are the position coordinates, and $roll, pitch, yaw$ represent the orientation angles.

The discretized model of the position state is defined as

$$x(t + \Delta t) = x(t) + v_x(t) \cdot \Delta t \tag{3}$$

$$y(t + \Delta t) = y(t) + v_y(t) \cdot \Delta t \tag{4}$$

$$z(t + \Delta t) = z(t) + v_z(t) \cdot \Delta t \tag{5}$$

where v_x, v_y, v_z are the velocity components along the respective axes, and Δt is the time step. The quadrotor moves in the arena only with linear velocity, without any angular velocity, as rotation is not needed for this task. Its orientation is assumed to remain stable, making the effect of angles on the motion negligible, thereby relying solely on linear motion.

The drone is outfitted with several sensors that provide essential data for its autonomous operation. These sensors include:

- GPS, for localization within the environment, enabling the estimation of the drone's position;
- camera, for detecting and identifying rescue targets, providing visual information for the search;
- IMU, for measuring drone's orientation, acceleration, and angular velocity, supplying data that helps the controller navigate while compensating for gravity and external environmental factors.

Each sensor is responsible for publishing data on a specific topic.The nodes subscribe to relevant topics based on their role in the mission, enabling them to react in real-time to changes in the environment. For example, the controller is subscribed to IMU topic, making the sensed data accessible for applying the desired velocity.

The controller subscribes to a topic named *cmd_vel*, which contains messages with the desired linear and angular velocities. Upon receiving these messages, the controller applies the corresponding forces to maneuver the drone within the environment. It consists of three PID that keep the quadrotor stable and reactive to published movements.

This simplified model assumes ideal conditions for the drone's movements, without considering complex dynamics such as wind resistance or motor thrust variations. The primary focus remains on simulating the drone's trajectory and the coordination required for target detection and rescue.

III. Solution

In the following subsections, the assumptions made, the estimator approach used, and the plan and rescue algorithm implemented for the drone system are presented.

A. Assumptions

To approach the search and rescue task, a set of assumptions was made to design a practical and efficient solution. These assumptions help simplify the problem while maintaining realism in the context of the system's objectives.

First, each drone is assumed to know the total number of targets to rescue. This assumption simplifies the search process by enabling the drones to focus on locating and rescuing the targets without the need for exhaustive exploration of the entire arena. The drones will continue searching until all targets have been discovered and rescued.

Second, it is assumed that messages sent between drones within the communication range are always successfully delivered, and packet loss is not considered in the system. This assumption ensures reliable communication for coordination and data exchange during the rescue process.

Third, the number of drones and the number of targets are configurable parameters. However, the relationship between these parameters must satisfy the following condition to ensure that the target rescue process is feasible:

$$n_d \geq n_t \times (\mathrm{r} - 1) + 1 \tag{6}$$

where:

- n_d is the total number of drones available for the mission.
- n_t is the total number of targets to be rescued.
- r is the minimum number of drones required to rescue each target.

This condition ensures that enough drones are available to successfully rescue all targets within the constraints of the system.

Fourth, the drone's velocity is assumed to be constant at 0.5 m/s for each direction.

Additionally, the height of the target is assumed to be known. This information aids the drones in computing the distance to the target, improving their ability to navigate accurately and effectively during the rescue operation.

B. Kalman filter

The drone estimates its position using the Kalman filter estimator. It consists of two main phases, the prediction and the correction. The prediction step starts when the linear velocity gained from the GPS and the orientation velocity obtained from the IMU sensor are published on the topics. The correction step is invoked when the GPS position data is received.

1) Prediction step: In the prediction step of the Kalman filter, the objective is to estimate the new state of the system based on the current state and the control inputs, while also updating the uncertainty associated with the state estimate. Given the state vector defined in Eq 2 and assuming a constant orientation (i.e., no changes in angular components), the predicted state is defined by applying Eq 3, Eq 4 and Eq 5 where Δt corresponds to the inverse of the GPS publishing frequency.

The predicted state vector μ_{new} is then expressed as

$$\mu_{\text{new}} = \begin{bmatrix} x_{\text{new}} \\ y_{\text{new}} \\ z_{\text{new}} \\ \text{roll}_{\text{new}} \\ \text{pitch}_{\text{new}} \\ \text{yaw}_{\text{new}} \end{bmatrix} \tag{7}$$

In addition to the state prediction, the covariance matrix Σ, which quantifies the uncertainty in the state estimate, must also be updated. The updated covariance matrix Σ_{new} is calculated using the following equation

$$\Sigma_{\text{new}} = A \cdot \Sigma \cdot A^T + G \cdot Q \cdot G^T \tag{8}$$

where Σ represents the current covariance matrix, and the terms A, G, and Q are defined as follows.

The matrix A is the state transition matrix, which describes how the state evolves based on the control inputs. This matrix can be expressed as

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

It is a diagonal matrix, assuming that the state variables are independent of each other in terms of their evolution over time.

The matrix G is the kinematic model noise redistribution matrix, which represents how the noise in the control inputs affects the state evolution. Here, the *yaw* angle has been considered to account for possible orientation inaccuracies. In particular, G is given by

$$G = \begin{bmatrix} \cos(\text{yaw}) & -\sin(\text{yaw}) & 0 & 0 & 0 & 0 \\ \sin(\text{yaw}) & \cos(\text{yaw}) & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

The matrix Q is the process noise covariance matrix, which captures the uncertainty associated with the control inputs and system dynamics. It is defined as

$$Q = \begin{bmatrix} \sigma^2_{\mathrm{v}_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{\mathrm{v}_y} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\mathrm{v}_z} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\mathrm{w}_r} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\mathrm{w}_p} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{\mathrm{w}_w} \end{bmatrix} \tag{11}$$

where $\sigma_{\mathrm{v}_x}, \sigma_{\mathrm{v}_y}, \sigma_{\mathrm{v}_z}$ represent the standard deviations of the velocity components, and $\sigma_{\mathrm{w}_r}, \sigma_{\mathrm{w}_p}, \sigma_{\mathrm{w}_w}$ represent the standard deviations of the angular velocity components.

2) Correction step: In the context of the Kalman filter, the correction step refines the predicted state estimate using new sensor measurements. In this case, the sensor data is provided by the GPS system, which reports the drone's position in three-dimensional space. Only the position is considered during the correction step, while orientation remains unaccounted for. The correction step adjusts the state vector and its covariance matrix by integrating these measurements.

The GPS provides measurements of the drone's altitude, longitude and latitude in the world. These values are converted into coordinates x, y and z, which define the measurement vector, expressed as:

$$\mathbf{z} = \begin{bmatrix} z_x \\ z_y \\ z_z \end{bmatrix} = \begin{bmatrix} gps_x \\ gps_y \\ gps_z \end{bmatrix} \tag{12}$$

where gps_x, gps_y, and gps_z are the coordinates of the drone as provided by the GPS sensor.

The measurement noise covariance matrix R captures the uncertainty associated with the GPS measurements. Knowing the accuracy of the GPS allows better correction (estimation). It is computed as follows

$$R = \sigma^2_{\text{GPS}} \cdot I_3 \tag{13}$$

where I_3 is the 3x3 identity matrix, and the diagonal elements represent the variances (squared standard deviations) of the GPS measurements in the x, y, and z directions.

The Jacobian matrix H is the derivative of the measurement function with respect to the state vector. It describes how the state vector influences the measurements. In this case, the GPS measurement is directly related to the position components x, y, and z, and is independent of the orientation (*roll*, *pitch*, *yaw*). Thus, the Jacobian matrix H is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{14}$$

This matrix essentially extracts the position components from the state vector, as the GPS measurements are only concerned with the position.

The Kalman gain K determines how much influence the new measurement should have on the updated state estimate. It is computed by minimizing the uncertainty in the updated state estimate. The Kalman gain is given by

$$K = \Sigma H^T (H \Sigma H^T + R)^{-1} \tag{15}$$

where Σ is the predicted state covariance matrix, representing the uncertainty in the predicted state, H is the Jacobian matrix and R is the measurement covariance matrix. The Kalman gain provides a measure of the relative confidence in the predicted state w.r.t. the new measurement.

Once the Kalman gain is computed, the state estimate is updated by incorporating the new measurement. The updated state vector is given by

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{predicted}} + K\mathbf{y} \tag{16}$$

where y is the difference between the estimated position and the measurement vector.

Finally, the covariance matrix Σ is updated to reflect the reduced uncertainty after incorporating the GPS measurement. The updated covariance matrix is computed as

$$\Sigma_{\text{new}} = (I - KH)\Sigma \tag{17}$$

where I is the identity matrix of the appropriate size. This update reduces the uncertainty in the state estimate, particularly in the position components x, y, and z, which have been corrected by the GPS measurement.

C. Planning and searching

Each drone autonomously navigates the environment using a random exploration strategy, randomly choosing a point inside the environment and moving towards it. This strategy is easy and not optimal, but it ensures the coverage of the environment without centralized control and without any more complex strategy. This method balances simplicity and robustness in dynamic search-and-rescue scenarios.

During the movement, drones continuously scan their surroundings using camera sensor. When a target is detected, its location is estimated using sensor data and the pinhole camera model. This estimation can be computed since the parameters of the camera, resolution and field-of-view, are known. The detection triggers the next phase of the operation—requesting assistance and initiating rescue.

D. Cooperation

Upon detecting a target, the drone broadcasts its position estimate and the estimated position of the target within its communication range. Nearby drones receiving this signal determine if they are available to assist based on their status. This decentralized approach reduces communication bottlenecks and increases fault tolerance.

When the needed number of drones arrives, they coordinate the rescue operation. Until all the drones do not see the target, they are not ready to rescue it. Once all the needed rescue drones localize the target, it is considered rescued.

IV. Implementation details

The project was developed using Gazebo 11 and ROS1 Noetic, leveraging an existing environment[1]. Gazebo provides a high-fidelity robotics simulator with a robust physics engine that models real-world phenomena such as gravity, collisions, and aerodynamics. It supports essential sensors like cameras, IMUs, and GPS modules, enabling realistic drone navigation and target detection.

The quadrotor drone is based on the *hector quadrotor*[2] package. Since this library is not officially released for ROS Noetic, it was cloned and built within the project. Modifications were made to the URDF files to adapt the drone's structure, adding a downward camera to detect targets. The target and arena configurations remain consistent with the existing task for continuity.

A. Drone initialization

The simulation of the search and rescue task starts when the drones start the engines. The drones and the targets spawn inside the arena and the drones initially perform a vertical lift to reach a desired height. This first phase is called initialization, which ends when the drone reaches the starting point. To avoid collision, each drone has a predefined height for which it can moves in all directions without incurring in possible collisions. Since the heights are different, some drones start before others the search and rescue task. The higher the drone position, the higher the possibility to detect a target because it covers more space, but the higher also the uncertainty on the estimation of the target position.

[1]https://github.com/Marcosterlo/Distributed_project

[2]https://github.com/tu-darmstadt-ros-pkg/hector_quadrotor

B. GPS coordinate conversion

The drone's GPS module publishes position data in the WGS84 geodetic coordinate format, consisting of latitude, longitude, and altitude. However, for effective navigation and motion planning within the simulation environment, these coordinates must be converted into a Cartesian coordinate system. This transformation is performed in two stages.

First, the geodetic coordinates are converted into Earth-Centered, Earth-Fixed (ECEF) coordinates, a global 3D Cartesian system centered at Earth's center of mass. This conversion uses the WGS84 ellipsoidal model, with the semi-major axis and the first eccentricity squared defining Earth's shape. The resulting ECEF coordinates are calculated using trigonometric functions applied to the latitude, longitude, and altitude values.

Second, the ECEF coordinates are transformed into a local Cartesian frame relative to a reference point, chosen as the origin of the environment. This transformation is achieved by computing the vector difference between the drone's ECEF coordinates and the reference point's ECEF coordinates. A rotation matrix based on the reference latitude and longitude is then applied, aligning the local axes with the environment's coordinate system.

The resulting local coordinates are expressed as (x, y, z) and are published through the ROS infrastructure, allowing the motion planner to perform pathfinding and rescue operations based on accurate positional data.

C. Target distance computation

To estimate the relative position of a detected target, the drone uses its onboard camera and estimated altitude. The pixel coordinates of the target in the image frame are processed using the camera's field of view and resolution. By computing the camera focal length, the target position is normalized relative to the image center. This normalization accounts for the camera's optical properties and corrects for orientation.

The normalized coordinates are then scaled by the drone altitude, yielding the target relative position in meters within the local Cartesian frame. This approach enables the drone to determine the target location using only visual data and its altitude, facilitating navigation and rescue operations.

D. Drone states

Every drone has a current state which determines the possible actions. Six states have been defined to govern the simulation: WAITING_FOR_START, SEARCHING, MOVING_TO_TARGET, WAITING_FOR_SUPPORT, SEARCHING_AROUND and FINISH.

The WAITING_FOR_START state is set when the robot starts to move on the z axis to reach the starting position. In this phase, only messages sent by the neighbours to collect their position are received, the others are ignored.

During the SEARCHING phase, the robot moves inside the arena to reach a random point, simultaneously looking for targets and being available to assist another robot in rescuing a found target.

The MOVING_TO_TARGET state is set when the robot receives a message to rescue a target and starts moving towards

the received point or when it detects a new target. During this phase, for simplicity, it ignores other discovered targets and continues to move towards the original one.

When the robot finds the target and is near it, it waits for support to rescue it, setting its current state to WAITING_FOR_SUPPORT. A robot in the MOVING_TO_TARGET state moves to its assigned position, where it may or may not detect the target. If the target is visible, the robot switches to the WAITING_FOR_SUPPORT state. Otherwise, it transitions to SEARCHING_AROUND and begins moving in a spiral pattern centered on the assigned point to locate the target. This behavior accounts for the fact that each robot's position estimates are subject to error, meaning the same global point may correspond to different local points for each robot due to inaccuracies in their position estimation.

When the number of robots in state WAITING_FOR_SUPPORT is higher or equal than needed, the target is considered rescued and the state of the drones is set to SEARCHING if there are other targets to rescue, FINISH otherwise.

Every robot shares the following information:

- Its position at every estimation/correction step;
- The position of the target when it is detected by the camera;
- When it is in state WAITING_FOR_SUPPORT, it continues to share the message that it is ready to rescue;
- It assigns the nearest robots to help it when a new target is detected;
- When a target is rescued, the message is broadcasted;
- When all the targets are rescued, the drones broadcast the mission completed message.

E. Target position estimate

Each drone has a localization error, which contributes to send possibly wrong target positions. In order to improve the localization of the target, every time the target is localized by the other drones, this value is averaged with the sensed one. In this way, it is possible to reduce the target location estimate error.

V. Results

To evaluate the system's performance, two experiments focusing on the average rescue time and one experiment focusing on the impact of noise have been conducted. Due to the high computational demands of each simulation and the limited processing power of the available machine, only three simulations were performed for each experiment. As the planning algorithm relies on a random movement selection, the results are not statistically representative. However, they allow for the identification of trends and provide a basis for interpreting the outcomes.

In the first experiment, n_d is set to 5, n_t is set to 4 and r is set to 2. This experiment explores all four combinations of two factors: communication range (infinite or limited to 6 meters) and the presence or absence of noise. The noise values are reported in Table I.

TABLE I
Noise applied to sensors

	noise
GPS	0.2 m
Sonar	0.5 m
IMU	0.1 m/s^2

TABLE II
Exp1 average rescue time with n_d = 5, n_t = 4 and r = 2.

	Noise absence	Noise presence
Infinite communication range	1m 4s	1m 41s
Limited communication range	1m 23s	1m 2s

Noise can significantly impact the validity of messages exchanged between drones, including critical data such as each drone's position and the location of the target to be rescued. Furthermore, the communication range introduces additional uncertainty for drones that are outside its boundaries. This limitation can result in missed detections of targets, delayed requests for assistance, uncommunicated rescue completions, and failures to notify all drones that the mission is complete.

Among the simulations conducted, the scenario without noise and with an infinite communication range represents the simplest and most idealized case. In contrast, the scenario incorporating noise and a limited communication range mirrors a more realistic and challenging operational environment. The differences in complexity between these scenarios are reflected in the results presented in Table II. As expected, the easiest scenario requires the least amount of time to complete. Conversely, the longest simulation time is observed in the scenario with noise and an infinite communication range.

The second experiment was performed with a communication range of six meters, the presence of noise, and n_t equal to 2. The variables in this experiment are the number of drones and the number of drones required to rescue a target (Table III).

Table IV presents the average, minimum, and maximum rescue times recorded during the experiments. The rescue time starts when the drones begin flying and ends when the last target is rescued. Instead, Table V displays the rescue time calculated from the moment the drones take off to the moment when all drones are informed that every target has been rescued.

As expected, looking at the average of rescue time of Table IV and Table V, limiting the communication range introduces a delay, as drones need time to enter the communication range and receive information. When a drone receives the signal that the mission is complete, it halts movement to conserve energy and begins broadcasting the completion message within the communication range. Other drones must enter this range to receive the message and stop as well.

In these settings, a higher number of drones does not always lead to faster mission completed message sharing. This is because the required number of drones per rescue remains

TABLE III
EXP2 SCENARIOS.

Scenario1	Scenario2	Scenario3
$n_d = 3$ $n_t = 2$ $r = 2$	$n_d = 5$ $n_t = 2$ $r = 2$	$n_d = 5$ $n_t = 2$ $r = 3$

TABLE IV
EXP2 AVERAGE, MIN AND MAX RESCUE TIME CONSIDERING WHEN ALL THE TARGETS HAVE BEEN RESCUED.

	Average	Min	Max
Scenario1	1m 48s	1m 10s	2m 14s
Scenario2	1m 21s	39s	2m 23s
Scenario3	1m 20s	52s	1m 58s

TABLE V
EXP2 AVERAGE, MIN AND MAX RESCUE TIME CONSIDERING WHEN ALL THE DRONES ARE AWARE THE MISSION IS COMPLETED.

	Average	Min	Max
Scenario1	1m 52s	1m 10s	2m 14s
Scenario2	1m 31s	1m 6s	2m 25s
Scenario3	1m 28s	1m 7s	1m 58s

in the same location and does not move. Although the limited number of simulations makes the results less reliable, it is reasonable to observe that the time required to share the mission completion message among all drones is lower when there are three drones compared to five. This is largely due to the fact that only one drone needs to enter the communication range versus three.

However, if drones were to move throughout the environment after completing the mission to distribute the completion message, the expected time would likely decrease as the number of drones increases.

On the other hand, it is reasonable that higher number of drones allows to faster complete the mission, thanks to higher area coverage in the same time (scenario 1 and scenario 2).

Comparing the experiments with varying numbers of drones for rescue (scenario 2 and scenario 3), the results indicate that the average rescue time remains relatively consistent. With additional experiments and more extensive data, it is expected that reducing the number of drones required to rescue a target would lead to faster completion of the rescue mission. Indeed, when fewer drones are needed to assist with a discovered target, the remaining drones can continue exploring, enhancing overall efficiency.

Figure 1 illustrates the target position estimation error over time for one simulation in the third scenario. The red vertical line marks the moment when a new drone detects the target and becomes ready to perform the rescue operation. As shown in the plot, the initial detection of the target coincides with a sharp peak in estimation error, indicating a sudden increase. This phenomenon occurs primarily because the detecting drone is in motion towards the target, and its camera oscillates as the drone decelerates and stabilizes to initiate the rescue process. This pendulum-like movement affects the camera's ability to accurately estimate the target's position, leading to an increased error.

The target position is updated based on a rule that averages

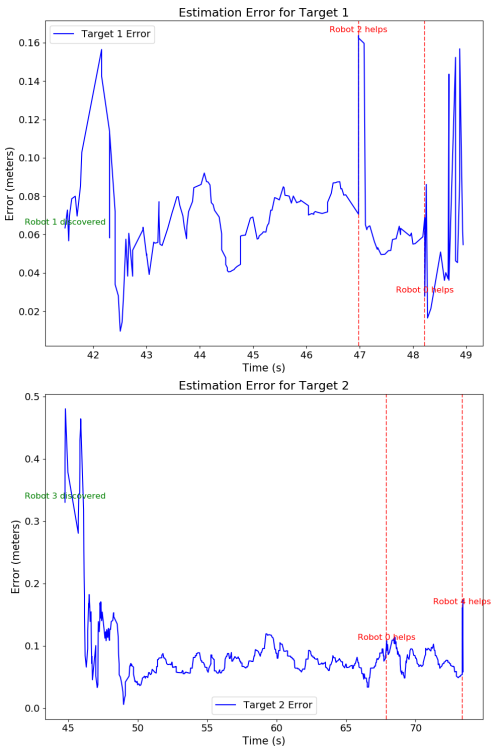


Fig. 1. Estimation error for the two targets with $n_d = 5$ and $r = 5$ (scenario 3 of Exp2).

TABLE VI
EXP3 GPS NOISE WITH $n_d = 5$, $n_t = 4$, $r = 2$ AND COMMUNICATION_RANGE = 6.

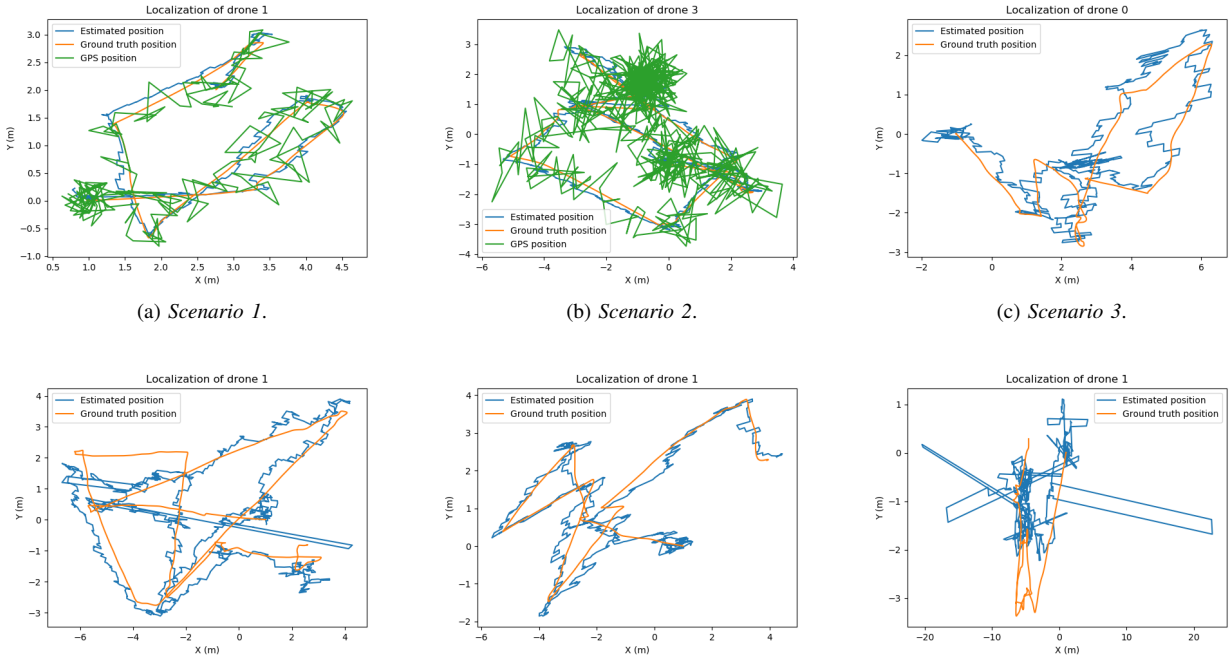
	Position noise	Velocity noise
Scenario 1	0.2 m	0.0 m/s
Scenario 2	0.5 m	0.0 m/s
Scenario 3	1.0 m	0.0 m/s
Scenario 4	0.5 m	0.3 m/s
Scenario 5	0.8 m	0.1 m/s
Scenario 6	0.8 m	0.5 m/s

the current estimated position with the newly detected position. Consequently, even the more accurate predictions from stable drones are influenced by the noisy estimate from the oscillating drone. This effect propagates through the system, temporarily increasing the overall estimation error.

Additionally, the rescue operation requires all three drones to maintain visual contact with the target. During this process, the drones with the higher point of view may have their view obstructed by the drones below them. To regain sight of the target, they slightly adjust their position, introducing another source of noise and further impacting the estimation accuracy. These combined factors contribute to the observed fluctuations in target estimation error during the rescue.

Finally, an analysis about the impact of the noise on the GPS has been conducted. In this third experiment, no repetitions were executed due to high complexity of the simulation. For this analysis, the simulations with the values displayed in Table VI have been executed.

In this project, the Kalman filter is utilized for state estimation, employing the GPS position value during the correction



(a) *Scenario 1.* (b) *Scenario 2.* (c) *Scenario 3.*

(d) *Scenario 4.* (e) *Scenario 5.* (f) *Scenario 6.*

Fig. 2. Ground truth position, GPS position, and estimated position of a single drone throughout the simulation for each scenario. In the (a) and (b) images, GPS coordinates are displayed, while in the remaining ones they have been omitted to avoid excessive visual clutter.

phase and the GPS linear velocity value during the prediction phase. Although the drone's velocity is assumed to be constant, noise has been intentionally introduced to the velocity values to analyze its impact on the drone's behavior and the system's robustness.

Figure 2 shows the ground truth position and the estimated position of one drone for each simulation. A plot presenting the average localization error across all drones in the simulation has been omitted due to its less informative nature. This plot contained fluctuations due to large discrepancies in error values, with one drone showing significant error while the others exhibited much smaller errors, making it difficult to interpret. Despite this, the general trend is that as velocity noise increases, localization accuracy decreases. When velocity noise is present, the algorithm predicts the drone's next position using these noisy velocity values, resulting in larger deviations from the true position. Conversely, when velocity noise is set to zero, the Kalman filter performs quite good despite the position noise, allowing the drone to localize itself without significant errors. This is highlighted in scenario 3.

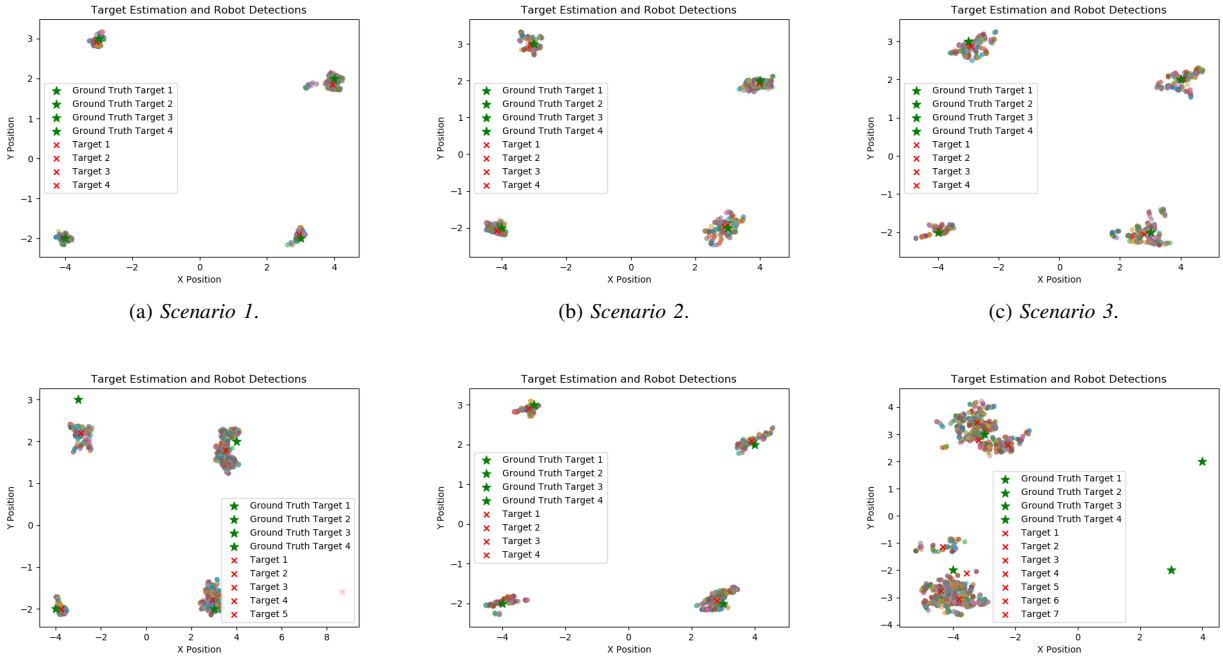
Furthermore, Figure 3 highlights the impact of velocity noise on target localization. In scenario 4, one drone detects a target and shares the detected target position $(8.5, -1.6)$, which is significantly far from the actual target location. This demonstrates how velocity noise can lead to substantial errors, complicating the coordination and decision-making processes for the drones. Additionally, in scenario 6, multiple drones detect the same target; however, the estimated positions are too noisy for the system to reconcile and recognize it as the same target. As a result, the system treats the same target as multiple distinct targets. Consequently, when the target is rescued, it is incorrectly counted as the rescue of several targets instead of just one.

VI. Conclusion

In this report, the design and implementation of a simulation environment specifically developed for drone-based search and rescue tasks is presented. The system includes a quadrotor drone model with its associated control mechanism, allowing for the deployment of a variable number of drones and targets within the simulated environment. Drones navigate using a random planning algorithm and, upon detecting a target, initiate the rescue operation. The environment supports configurable parameters such as sensor noise and communication range, enabling the simulation of various real-world challenges under controlled conditions.

A series of experiments was conducted to study the average rescue time in different scenarios. These experiments examined the effects of varying the number of drones, the number of targets, the number of drones required for rescue operations, the presence and magnitude of noise, and the communication range. Additionally, the impact of sensor noise on drone performance was analyzed, with a specific focus on localization accuracy and its influence on the overall effectiveness of the rescue missions.

The results demonstrate that handling search and rescue tasks in real world scenarios is complex. Indeed, noise and limited communication range significantly affect system performance, causing delays in target detection, miscommunication among drones, and increasing mission durations.



(a) *Scenario 1.* (b) *Scenario 2.* (c) *Scenario 3.*

(d) *Scenario 4.* (e) *Scenario 5.* (f) *Scenario 6.*

Fig. 3. Target detections of all drones. The green stars represent the real target positions, the colored circles represent the target position estimates, and the red crosses represent the centers of the estimate clusters with radius 1 (an arbitrary value used by drones to distinguish between different targets). The clustering algorithm is employed just for visual purposes.

The project is available on a public Github repository[3]. Detailed instructions for executing the code are provided in the README file.

A. *Future improvements*

To address the challenges of this project, several assumptions were made to simplify its implementation. The first assumption involves neglecting the orientation of the robot and focusing solely on its x, y, and z coordinates. While this approach simplifies the task, incorporating orientation control would make the project more realistic. This could be achieved by including the orientation angles in the drone model [1] and employing a Kalman filter to account for and correct potential estimation errors caused by orientation inconsistencies [2].

Additionally, the chosen search algorithm, which relies on random movements, could be significantly improved. Implementing systematic exploration techniques that ensure complete area coverage while maintaining a memory of previously explored regions would enhance efficiency. A further improvement would involve enabling the drones to collaboratively construct a shared map, where each drone shares its position and discoveries. This collaborative mapping approach would support more efficient coordination during the mission.

To prevent collisions, the current design requires drones to operate at different altitudes, which in turn affects their coverage and target estimation accuracy. While higher altitudes allow drones to survey larger portions of the map, they also increase the error in target estimation. Incorporating advanced collision avoidance algorithms would eliminate the need for height-based separation.

Another assumption is that all communication messages are delivered correctly to drones within the communication range. To address message delivery failures, a robust communication protocol could be implemented in order to handle errors and ensure reliable data exchange.

Regarding positional updates, each drone currently relies on its GPS sensor or, alternatively, Ultra-Wideband (UWB) sensors placed within the environment. However, drones do not share positional data with each other to refine their own estimates. A possible improvement, inspired by approaches such as the one described by Kia et al. [3], involves leveraging inter-drone communication to collaboratively enhance localization accuracy.

Finally, the target estimation process could be further refined by adopting more robust algorithms that account for noise and inaccuracies. These enhancements would significantly increase the reliability and precision of the system, enabling it to better address real-world challenges.

[3]https://github.com/lucaZardini/ros-environment-simulation-system

REFERENCES

[1] Randal W Beard. Quadrotor dynamics and control. *Brigham Young University*, 19(3):46–56, 2008.

[2] S. A. Belokon', Yu. N. Zolotukhin, K. Yu. Kotov, A. S. Mal'tsev, A. A. Nesterov, V. Ya. Pivkin, M. A. Sobolev, M. N. Filippov, and A. P. Yan. Using the kalman filter in the quadrotor vehicle trajectory tracking system. *Optoelectronics, Instrumentation and Data Processing*, 49(6):536–545, Nov 2013.

[3] Solmaz S. Kia, Stephen Rounds, and Sonia Martinez. Cooperative localization for mobile agents: A recursive decentralized algorithm based on kalman-filter decoupling. *IEEE Control Systems Magazine*, 36(2):86–101, 2016.