Improving Search and Rescue Response Times by Using Aerial-Supported Ground Search and Rescue Robots

Edward Silvey, Alfian Fadhlurrahman, Shailee Kampani, Alvin Afilla

Abstract—**Search and rescue (SAR) missions demand rapid victim location. Both UAVs and UGVs have been successfully used independently for SAR, but each has clear limitations. UAVs provide fast, wide-area coverage but cannot directly assist victims, while UGVs can deliver on-site support yet take longer to locate them. This work evaluates whether combining both platforms can improve overall rescue efficiency.**

To test this, we simulate an autonomous UAV–UGV team in Webots and compare it against a SLAM enabled ground robot as a baseline. The UAV performs a boustrophedon coverage pattern, detects victims using colour-based vision, and transmits their coordinates to the UGV. The ground robot then navigates directly to each victim using waypoint control and obstacle avoidance.

Across varied SAR scenarios, the UAV-supported system completes missions significantly faster, taking an average 56% of the time the UGV-only takes. These results show that aerial guidance can substantially reduce search times and improve the effectiveness of ground-based rescue robots.

Index Terms—**Aerial Systems: Applications; Autonomous Vehicle Navigation; Multi-Robot Systems; Search and Rescue Robots; SLAM**

I. Introduction

SEARCH and rescue (SAR) requires finding as many victims of disasters as fast as possible. Both ground and aerial search and rescue robots have become commonly used strategies in order to improve search and rescue [1], [2]. Both of these methods though, have their downsides along with their upsides. Unmanned Aerial Vehicles (UAVs) or drones are able to provide coverage of a wide area quickly whilst avoiding any obstacles on the ground, but are largely unable to provide help to victims once found. Unmanned Ground Vehicles (UGVs) or rovers, are able to provide the assistance once they find the victim, but have to avoid obstacles on the ground.

There has been extensive research on multi-robot systems for search and rescue purposes [3], [4] and how best they can be coordinated [5].

Combining the two methods could lead to a quicker response times and more victims being found successfully, thus improving search and rescue overall.

II. Methods

In order to test if a UAV-UGV search party was able to perform more efficiently, we decided to simulate both a combined UAV-UGV search party and a lone UGV search party in Webots [6]. Using this approach we were able to time how long it took for both methods to complete different search and rescue missions. The code repository for this project can be found on GitHub [7].

A. Drone Movement and Flight Control

The purpose of the drone is to fly over the designated area, locating victims in need of rescue and sending their location to the ground rovers so that they can provide the needed help.

The UAV is controlled by an adapted version of the *mavic2pro.c* sample controller [8] that has been changed in order to navigate to predetermined GPS waypoints, as opposed to being remotely controlled by a pilot. For this use case these waypoints follow a Boustrophedon path [9] over the target area, moving forwards only once it has reached the other end of a row. This allowed all areas of the target area to be covered by the drone.

The movement of the drone is determined by the error between it's own GPS location and the GPS location of the target waypoint. This is calculated as:

$$\boldsymbol{E} = \boldsymbol{t} - \boldsymbol{p} \tag{1}$$

where $\boldsymbol{E}$ is the position error vector and $\boldsymbol{t}$ and $\boldsymbol{p}$ are the position vectors of the target location and drone location respectively. This error in position is then converted into disturbance, calculated as:

$$\boldsymbol{D} = \begin{bmatrix} \min(\max(-2\boldsymbol{E}_0, -2), 2) \\ \min(\max(2\boldsymbol{E}_1, -2), 2) \\ 0 \\ \min(\max(\boldsymbol{E}_2 + 0.6, -1), 1) \end{bmatrix} \tag{2}$$

where $\boldsymbol{D}$ is the disturbance vector for pitch, roll, yaw and altitude respectively. These values are clamped in order to prevent exaggerated movements which could cause the drone to loose control. Since the drone does not need to rotate when moving along its path the yaw will always be set to 0.

A waypoint is considered to be visited when both the euclidean distance in horizontal plane is less than 0.2m and the altitude difference is less than 0.1m. These constraints can be expressed in the forms $\sqrt{(E_0^2) + (E_1^2)} < 0.2$ and $\mid E_3 \mid < 0.1$ respectively. Once a waypoint is reached the next waypoint is considered as the target location. Once the final waypoint is reached the drone will hover in place.

B. Victim Detection and Communication

This section operates on two primary tasks: (1) colour-based victim recognition using the drone's onboard camera and (2) communication of the detected victim coordinates to the Surveyor ground robot.

1) Colour Based Detection: Webots does not natively integrate frameworks such as TensorFlow [10] or PyTorch [11], and implementing such systems would exceed the real-time processing constraints of the simulation time step. As a result all victims in this scenario wear green (RGB: 0, 0.8, 0) to ensure a consistent and distinctive visual signature, as a result the detection method relies on RGB colour segmentation rather than more advanced computer vision approaches.

At every image update, the controller samples the pixel values at the centre of the camera frame ($\frac{width}{2}, \frac{height}{2}$). This significantly reduces the computational load while maintaining detection reliability during systematic area coverage.

A victim is flagged when the following condition is satisfied:

$$(G > R + 30) \land (G > B + 30) \land (G > 100) \tag{3}$$

where R, G, and B represent the green, red, and blue channel intensity values respectively, ranging from 0 to 255.

This threshold ensures that the green channel is more dominant than both the red and blue channels, while also requiring the green intensity to be above 100. This helps avoid false positives from dark areas or low-saturation terrain features.

Once a green region is detected, the controller checks that the object corresponds to an untagged victim by looking at an internal registry that tracks the detection status of each victim.

2) Coordinate Extraction and Communication: The wireless communication architecture utilises the Webots emitter-receiver device pair. In order to provide a dedicated communication link separate from other devices, the drone's emitter was set up to broadcast on Channel 1. Correspondingly, the Surveyor's receiver was configured to monitor Channel 1, allowing for asynchronous message reception.

Once a victim is verified, the system extracts its spatial coordinates and transmits them immediately via Channel 1. Coordinates are formatted as ASCII strings:

```
"X,Y\0"
```

Coordinates are sent with two decimal places of precision (e.g., `2.45,1.83`), which is sufficient for centimetre-level localisation required by the ground robot. Each coordinate pair is transmitted exactly once to prevent redundant transmissions. The null-terminated string format ensures reliable message framing at the receiver.

This communication protocol is deliberately lightweight: the message contains only numerical coordinates and a null-terminator, minimising bandwidth and preventing desynchronisation between the robots. The Surveyor robot, upon receiving the coordinates, enters into targeted navigation mode and uses its internal motion-planning logic to travel to the exact location.

3) Final Tagging and Output Summary: After detection, the drone tags the victim by changing their shirt color to red, ensuring the system does not re-process victims that have already been identified. When all waypoints are traversed, the detection subsystem outputs a complete summary of found victims and their coordinates for quantitative assessment.

C. Ground Robot Navigation, and Obstacle Avoidance

The ground robot we use is a simulation model of the Surveyor SRV-1 available in Webots. The controller facilitates autonomous navigation through a waypoint-based trajectory using waypoints received dynamically from the drone while maintaining the capability to reactively avoid unexpected obstacles.

The system architecture is built on a Finite State Machine (FSM) that governs the transition between navigation, obstacle detection, and avoidance manoeuvres [12]. The following is the image of the FSM.

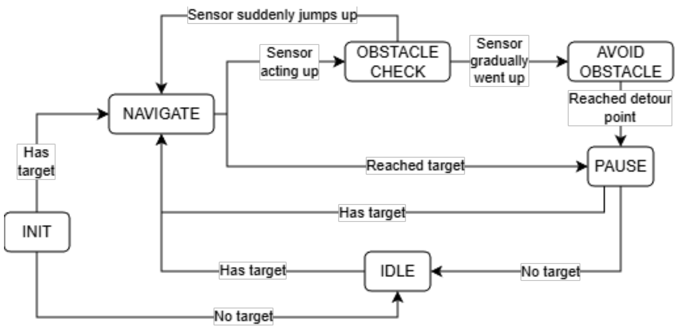

Fig. 1. Ground Robot FSM

1) Waypoint Navigation: The robot travels to target coordinates (x_{target}, y_{target}) received from the drone using a proportional control loop. The navigation logic continuously calculates the heading error α between the robot's current orientation θ and the desired bearing to the target:

$$\alpha = \text{atan2}(y_{target} - y, x_{target} - x) - \theta \tag{4}$$

The motor velocities are adjusted differentially to minimize α. A proportional gain is applied to the heading error to determine the turning speed, while the forward speed is dynamically scaled based on proximity to the target. To ensure precision, the robot decelerates when the distance to the target is less than the defined threshold $D_{slow} = 1.0\,\text{m}$.

2) Obstacle Avoidance: The obstacle avoidance used here is a reactive "swerve" manoeuvre. The system monitors the front distance sensor (d_{front}). To filter out sensor noise, the system employs a temporal filter: an obstacle is confirmed only if d_{front} remains below the safety threshold ($1.0\,\text{m}$) for $N = 5$ consecutive simulation steps.

Upon confirmation, the robot transitions to the obstacle avoidance state. This state calculates a temporary local detour point (x_d, y_d) at a fixed lateral offset relative to the robot's current pose.

We set the detour point to $0.3\,\text{m}$ (30 cm) to the robot's right (perpendicular to the current heading). The coordinates for this manoeuvre are derived using the transformation:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + 0.3 \cdot \begin{bmatrix} \sin(\theta) \\ -\cos(\theta) \end{bmatrix} \tag{5}$$

This calculation effectively places a virtual waypoint 30 cm to the immediate right of the robot. The robot navigates to this point to bypass the obstruction before reverting to navigation state to reassess the path to the original target. This method assumes that the environment consists of discrete obstacles rather than continuous walls, as the fixed lateral displacement allows the robot to side-step small objects without complex map processing.

D. Ground Robot SLAM Assisted Navigation

For our benchmark baseline, we developed a ground robot that uses a state-machine-based controller for autonomously moving to predefined coordinates; the logic behind moving to these predefined coordinates would be similar to the previous section. The robot is also assisted by SLAM and an Obstacle Avoidance algorithm that is provided by the previous section to ensure it reaches the victims efficiently.

1) Map Representation and World-Grid Transformation: This section would be beneficial in helping the SLAM picture a 2-D occupancy grid. This grid would be the source of keeping and tracking any changes that it found when traversing through the map. This grid will interact with the other algorithms and sensors within this SLAM approach, path-checking, detour generation, sensors, and victim detection all read or save information onto this grid.

The grid is made by defining fixed limit boundaries (in meters) of the map, and anything that is beyond these limits would be treated as *"out of bounds"* and would not be considered as a traversable area.

At anytime the robot wants to query or update the map, it needs to convert those coordinates (x,y) into indices (i,j). This is done through normalising and scaling coordinates using this function:

$$r_x = \frac{x - X_{\min}}{X_{\max} - X_{\min}}, \qquad i = \lfloor r_x \cdot N \rfloor \tag{6}$$

$$r_y = \frac{y - Y_{\min}}{Y_{\max} - Y_{\min}}, \qquad j = \lfloor r_y \cdot N \rfloor \tag{7}$$

- r_x, r_y: a normalised position, where the point is between the maximum and minimum coordinates of the x, y axis as a ratio in [0,1]
- x, y: the actual world coordinates in meters
- $X_{\max}$, $X_{\min}$: the maximum and minimum bounds of the map along the x-axis
- $Y_{\max}$, $Y_{\min}$: the maximum and minimum bounds of the map along the y-axis
- N: the map size

This transformation function will be used at anytime the robot moves to provide updates to the grid map; it will play a significant role in helping the robot map the victims and check for clear paths and obstacles.

2) Sensor Model and Ray-Based Map Updates: The slam mode also features four IR distance sensors, placed at the front, left, right, and back of the robot. Those sensor readings would be converted to a clipped distance using this formula:

$$d = \operatorname{clip}\big(0.4\,(1 - \tfrac{\text{raw}}{154}),\, 0,\, 0.4\big) \tag{8}$$

where it treats the raw sensor reading into a clamped distance [0,0.4]. Those readings are then projected into a global frame through ray-casting, for each sensor along a distance (d) and direction (θ), the coordinates along the ray are:

$$x(d) = x_r + d\cos\theta \qquad y(d) = y_r + d\sin\theta \tag{9}$$

Before the sensor hits an obstacle, the belief would be set to:

$$v_{t+1} = \max\big(0,\ v_t - 0.05\big) \tag{10}$$

After a hit, the belief would be updated again to:

$$v_{t+1} = 1.0 \tag{11}$$

And along the way, the belief would keep getting updated in the case that after hitting the space is free again, which the process would jump back to the belief before it hits an obstacle.

3) Path Collision Checking on the Occupancy Map: For checking and planning ahead in cases where the path to a current target intersects any occupied cells in the near future. The controller will call a path checking routine, which takes points between the robot and the target over a fixed look-ahead distance. Each of those points can be formulated to:

$$\mathbf{p} : [0.1, d_{\text{check}}] \to \mathbb{R}^2, \qquad \mathbf{p}(d) = \begin{bmatrix} x(d) \\ y(d) \end{bmatrix} = \begin{bmatrix} x_r \\ y_r \end{bmatrix} + d\,\hat{\mathbf{u}} \tag{12}$$

where $\hat{\mathbf{u}}$ is the unit vector towards the target, and d is the distance that is incremented in fixed steps. That position would then be mapped onto the occupancy grid, and its cell value would be tested against an occupancy threshold. If it exceeds that threshold, it would return the world coordinate of the blocking cell and pass those coordinates to the detour planner for steering around the predicted obstacle.

4) Local Detour Planner: if the path collision checking function detects an obstacle, the planner would create a detour around the waypoint to avoid and go around the obstacle

To avoid those obstacles, we rotate the vector (u_x, u_y), rotate positively and negatively by 90 degrees. This will make two perpendicular unit vectors:

$$\hat{\mathbf{n}}_L = (-u_y,\ u_x) \tag{13}$$

$$\hat{\mathbf{n}}_R = (u_y,\ -u_x) \tag{14}$$

Given a coordinate (x_0, y_0), the robot will create two candidates for detour, either going left or right:

$$\mathbf{D}_L = (x_o, y_o) + \delta\hat{\mathbf{n}}_L, \qquad \mathbf{D}_R = (x_o, y_o) + \delta\hat{\mathbf{n}}_R \tag{15}$$

An offset of $\delta = 0.3\,\text{m}$, is used for creating a safety limit such that the robot will not skim the obstacle.

5) Integration With Waypoint Traversal and Victim Revisit: The slam approach has 6 specific states: Detour, Revisit, Align, and Approach. Each of those states is covered within two phases: Initial Scan and Revisit. At the start, the robot will start with the Initial Scan phase, where it will start traversing the map in a boustrophedon lawn-mower-like pattern from top to bottom. During this phase, the robot will continuously mark obstacles, free spaces, and victims onto the grid.

Once finished with its traversal, the robot will switch into the Revisit Mode, where it takes the coordinates of the previously stored victims and starts visiting them again one by one, starting from the first victim spotted. Once it gets to the position where it spots the victim, it will switch states to

Align and align itself with the victim until the heading error falls below the tolerance for alignment:

$$e_\theta = \text{normalize}\big(\theta_v - \psi\big) \tag{16}$$

Then it will switch state to approach the victim, not stopping until it stops detecting the colour green, in which it would update its belief that it has already gone through that victim. After it finishes revisiting all the victim coordinates, the robot will stop and conclude its run.

E. Comparing Approaches

To compare the UAV-UGV search party to the UGV only approach we simulated different SAR scenarios of varying complexity. In order to obtain a results that best reflected the randomness of SAR outside of a simulated environment we varied the quantity and location of obstacles as well as the positions of the victims, however, to ensure comparable results both methods were used on each variation. To ensure complete testing of the detection algorithm, custom victim prototypes were designed to represent casualties in various positions such as sitting and lying down.

Both search parties were tasked with finding three victims in each of the different scenarios. Since a victim cannot be considered to be helped until the UGV reaches them, the time taken to complete the search was defined as the time the UGV reached the third victim. This was repeated multiple times for each scenario and an average calculated.

III. Results

A. Scenario 1

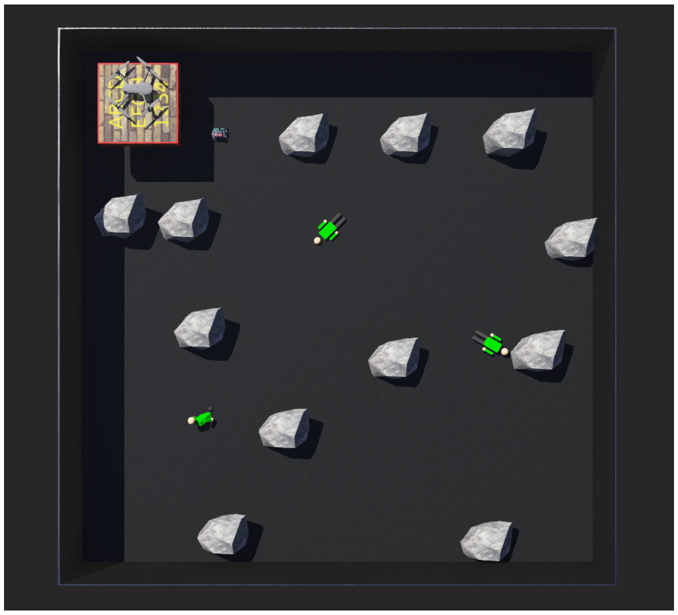

Fig. 2. Search and rescue scenario 1 with obstacles and victims

1) SLAM Assisted Ground Robot: The average time that the SLAM-assisted ground robot took to completes its search and rescue mission was 209.4s. It took an average of 115s for it to complete the search for the victims, and an average of 114.4s for it to then visit all of the victims after that.

2) Aerial Supported Ground Robot: The average time that the aerial supported ground robot took to completes its search and rescue mission was 118.2s. It took an average of 71s for the drone to complete the search for the victims, and the rover took an average of 47.2s for it to pursue and visit all those victims.

B. Scenario 2

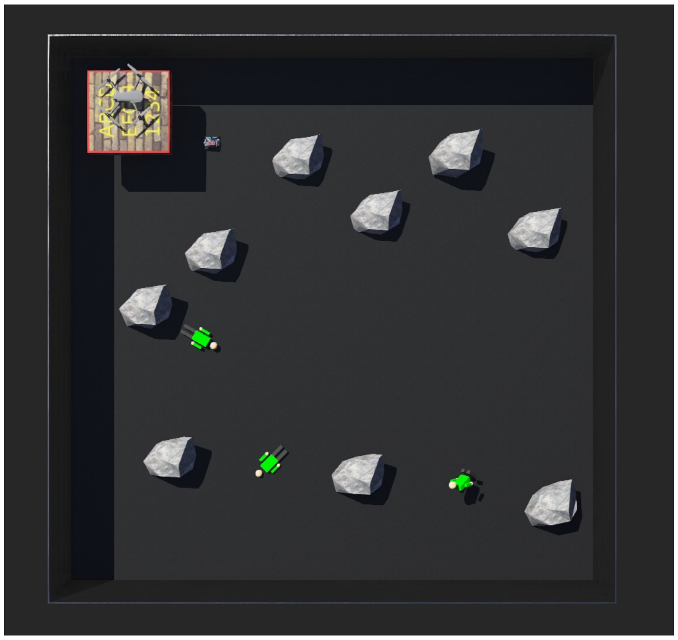

Fig. 3. Search and rescue scenario 2 with obstacles and victims

1) SLAM Assisted Ground Robot: The SLAM only implementation was unable to locate all the victims meaning no time could be recorded, and no result obtained.

2) Aerial Supported Ground Robot: The average time that the aerial supported ground robot took to completes its search and rescue mission was 116s. It took an average of 71s for the drone to complete the search for the victims, and the rover took an average of 45s for it to pursue and visit all those victims.

IV. Conclusion

This study demonstrates that, in comparison to ground-only methods, combining UAV and UGV platforms greatly increases search and rescue efficiency. According to the quantitative results, the aerial-supported system completed objectives in an average of 118.2 seconds as opposed to 209.4 seconds for the stand-alone SLAM-assisted robot, resulting in a 43.5% reduction in the overall mission time.

The aerial drone's capacity to quickly scan the environment and send victim coordinates in real time is the source of these performance improvements. While the stand-alone SLAM-assisted robot required 115 seconds to scan the area, the drone completed the same coverage in just 71 seconds which is a 38% reduction in the search phase alone. The UAV reduces both the uncertainty and traversal burden placed on the ground robot. This allows the UGV to navigate directly towards the confirmed victims rather than relying solely on exhaustive exploratory mapping.

These efficiency increases stem from two key factors:

- **Terrain Complexity:** The hybrid system handled ground clutter well. The aerial unit completely avoided the obstacles on the ground, while the SLAM-based rover had to navigate around each one to ensure full coverage. This efficiency gap is anticipated to grow as terrain complexity rises (e.g., in debris-heavy disaster zones).
- **Distance from Initial Location:** The advantages increase with the size of the search area. The UAV can cover large distances quickly, allowing it to locate victims in the search grid much earlier than a ground robot could travel the same distance. This eliminates the need for blind searching and helps the ground unit execute a direct, energy-efficient route to the targets.

Ultimately, this research provides strong evidence that combining complementary robotic platforms maximises the strengths of each system while mitigating their individual weaknesses. This provides a convincing framework for operations of the future, where significant time savings may directly result in higher victim survival rates.

Future work could extend this approach by incorporating more complex detection pipelines, cooperative multi-drone deployments, or real-time adaptive path planning to further enhance robustness in real-world disaster scenarios.

A. Limitations

1) Aerial-supported Ground Robot Navigation and Obstacle Avoidance: The implementation of the ground robot navigation while being assisted by a drone is made to be simple. However, this design choice results in specific functional limitations:

- **Blind detour:** The robot shifts 0.3 m to the side without checking the space. It may hit wide objects.
- **Unused sensors:** Only the front sensor is used, so side hits can occur.
- **No memory:** The robot cannot recall past steps. It can loop inside U-shaped traps.
- **Uneven Terrain Limits:** Slopes tilt the front sensor and give false readings. Slip also affects turns.

2) Ground Robot SLAM Assisted Navigation: Our SLAM-assisted ground navigation was designed to be as straightforward as the aerial-supported method. However, this decision led to a few restrictions on this strategy:

- **Victim marking:** The robot is able to mark and store the victim's location on the grid. However, any slight changes to the belief, such as how far the victim was spotted, can cause the victim to be marked on the grid more than once. This leads to the behaviour of the robot to go to a single victim's coordinates during the revisit stage multiple times.
- **Ideal Localization:** The robot's pose is only obtained from the GPS and compass. This approach assumes perfect position and heading, which does not reflect real-world scenarios where GPS would not work.
- **Victim Detection:** Victim detection relies only on green colour detection, making it prone to lighting and specific camera viewpoints. This also makes the robot believe that no obstacle is coloured green and every victim has a shared shirt colour.
- **Victim Storage:** Victims are stored based on the position that they were spotted, which makes the revisiting phase slightly more tedious, where the robot must go back to the angle it spotted the victim and drive forward from that angle until green is no longer seen. After not seeing green any more, it would assume that it already approached the victim closely. This approach is unconventional, as if the robot already gone to the victim before going to the coordinates where it was spotted, it would not count as the victim being visited.

APPENDIX

The following sections are attributed to the corresponding authors:

- Edward Silvey — Abstract; Introduction; Methods (Drone Movement and Flight Control; Comparing Approaches); Appendix; Referencing
- Alfian Fadhlurrahman — Methods (Ground Robot Navigation and Obstacle Avoidance); Conclusion (Limitations)
- Shailee Kampani — Methods (Victim Detection and Communication); Conclusion
- Alvin Afilla — Methods (Ground Robot SLAM-Assisted Navigation); Results

REFERENCES

[1] N. Li, J. Cao and Y. Huang, "Fabrication and testing of the rescue quadruped robot for post-disaster search and rescue operations," 2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI), Changchun, China, 2023, pp. 723-729, doi: 10.1109/ICETCI57876.2023.10176723.

[2] S. Waharte and N. Trigoni, "Supporting Search and Rescue Operations with UAVs," 2010 International Conference on Emerging Security Technologies, Canterbury, UK, 2010, pp. 142-147, doi: 10.1109/EST.2010.31.

[3] L. Li, M. Bai, "Multi-Robot Cooperation for Search and Rescue: A Review of the State of the Art," in IEEE Access, vol. 8, pp, 181553-181567,2020.

[4] J. Delmerico, E. Mueggler, J. Nitsch and D. Scaramuzza, "Active Autonomous Aerial Exploration for Ground Robot Path Planning," in IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 664-671, April 2017, doi: 10.1109/LRA.2017.2651163.

[5] T. Reimer, B. Olivieri, M. Cavalcanti and M. Endler, "From Air to Ground: Coordinating UAVs and UGVs in SAR Missions," 2025 28th International Conference on Information Fusion (FUSION), Rio de Janeiro, Brazil, 2025, pp. 1-6, doi: 10.23919/FUSION65864.2025.11124130.

[6] Webots. http://www.cyberbotics.com. Open-source Mobile Robot Simulation Software.

[7] A. Afilla, E. Silvey, A Fadhlurrahman, S Kampani, "AGSRR," GitHub repository, 2025. Available: https://github.com/alvin-zhaf/AGSRR/

[8] Cyberbotics Ltd., mavic2pro.c, Webots Sample Controllers. Available: https://cyberbotics.com

[9] H. Choset, "Coverage of Known Spaces: The Boustrophedon Cellular Decomposition," Autonomous Robots, vol. 9, no. 3, pp. 247–253, Dec. 2000, doi: https://doi.org/10.1023/a:1008958800904.

[10] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," TensorFlow.org, 2015. Available: https://www.tensorflow.org

[11] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Advances in Neural Information Processing Systems, vol. 32, 2019.

[12] R. Brooks, ”A robust layered control system for a mobile robot,” in IEEE Journal on Robotics and Automation, vol. 2, no. 1, pp. 14-23, March 1986, doi: 10.1109/JRA.1986.1087032.