



Coordination-Aware Task Allocation Via a Lightweight Supervisory Layer
An Event-Driven Approach to Multi-Agent Search and Rescue

T.R. Schutgens¹

Supervisors: A. Jamshidnejad¹, S. Schoonebeek¹

¹EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: T.R. Schutgens
Final project course: CSE3000 Research Project
Thesis committee: A. Jamshidnejad, S. Schoonebeek, L.C. Siebert

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Abstract

Task allocation in multi-robot Search and Rescue (SaR) is challenging: the environment is unknown, time-critical, and too dynamic for continuous centralized planning. This paper asks how a lightweight supervisory framework can dynamically allocate sub-tasks within a multi-robot team to balance exploration and victim retrieval without incurring prohibitive computational costs. We propose an event-driven, auction-based supervisory layer that intervenes only on discrete triggers; spatial conflicts, agent stagnation, and victim discovery, leaving robots to operate autonomously otherwise. Compared to a purely decentralized baseline across 216 simulated missions, the supervisor raises the victim discovery success rate from 50.5% to 91.7%, locates all victims roughly 24% faster without meaningfully increasing computation time. When extended to a heterogeneous fleet including a dedicated rescue agent, these exploration gains persist, but the increased rate of victim discovery outpaces single-agent rescue capacity, revealing a trade-off between exploration efficiency and victim retrieval throughput that the current allocation mechanism does not yet resolve.

1 Introduction

Search and Rescue (SaR) missions force autonomous robots to operate in highly volatile environments. Whether mapping unknown structures, navigating around spreading fire, or rescuing trapped victims, these operations carry strict safety constraints. As mission complexity grows to include cooperative exploration [1], coverage path planning [2], and navigating to retrieval waypoints, simply avoiding physical hazards is no longer enough. The core challenge shifts to task allocation. Controllers must efficiently distribute complex assignments across a robotic team that must continuously adapt to shifting risk and uncertainty.

Deploying autonomous teams in disaster zones keeps human responders out of inaccessible or dangerous spaces. An effective task allocation system guarantees that the rush to locate and retrieve victims does not compromise safety.

Previous literature addresses several isolated aspects of this problem. At the foundational level, control methods like Model Predictive Control (MPC) and shielded reinforcement learning maintain local safety margins against dynamic disturbances [3, 4]. At the coordination level, research covers task allocation for cooperative exploration [5], dynamic search and retrieval [6], and supervisory control in urban environments [7]. Recent architectures also manage assignment uncertainty [8], integrate hierarchical search strategies [9], and model human cognitive states [10].

A gap remains in achieving event-driven reallocation between exploration and rescue using a lightweight supervisor with single-task capacity and local MPC execution. While frameworks exist for isolated search or foundational safe control, the field lacks a unified and lightweight supervisory layer

to bridge them. Existing allocators struggle to shift agents between assignments quickly. Pulling a robot from map exploration to assist in a newly discovered rescue often incurs heavy computational costs. Solving this requires a supervisory mechanism that mediates between physical safety constraints and mission goals without slowing down the system.

This paper investigates the following primary research question:

How can a lightweight supervisory framework dynamically allocate sub-tasks within a robotic team to optimize the trade-off between safety constraints and performance objectives under environmental uncertainty?

In multi-robot coordination, a 'lightweight' framework is characterized by low computational overhead and minimal communication bandwidth. This primary question is divided into two targeted subquestions:

- **Sub-question 1 (The Allocation Trade-off):** How can a supervisory layer effectively coordinate global area exploration while remaining responsive to immediate victim retrieval needs?
- **Sub-question 2 (Performance Evaluation):** To what extent does dynamic task reallocation improve overall mission efficiency, particularly the time required to locate victims, compared to purely decentralized approaches?

The remainder of this paper is structured as follows. Section 2 reviews relevant existing literature, followed by a detailed description of the modeled simulation environment in Section 3. Section 4 outlines the methodology governing the supervisory control system and the local controllers. The experimental results are analyzed in Section 5. Section 6 addresses responsible research and ethics, and Section 7 concludes the study with directions for future work.

2 Literature Review

This section reviews the theoretical foundations of the proposed architecture. It examines existing multi-robot coordination solutions and highlights promising task allocation mechanisms to identify the specific knowledge gap this work addresses.

In multi-robot Search and Rescue (SaR) missions, the operational environment is typically unknown at deployment. This requires the primary search strategy to be coverage oriented [2]. The robot swarm must explore the maximum amount of unknown territory in the shortest possible time to locate trapped victims rapidly. A foundational method for driving this behavior is frontier based exploration. This directs robots to the boundaries between explored and unexplored space [11]. If several robots greedily pursue the nearest frontiers, spatial conflicts inevitably arise. These conflicts cause swarm clustering and physical interference. To mitigate this, task allocators often employ utility functions that penalize robots for selecting frontiers physically close to other agents [1]. Alternatively, they utilize environmental segmentation to force the swarm into distinct topological regions [5, 12].

The architectural design of these coordination mechanisms bifurcates into centralized and decentralized control approaches. In centralized multi-agent control, a singular system determines the mission plans for all agents simultaneously [13]. This approach benefits from a global vision of the entire system. It naturally prevents redundant exploration and yields optimal search paths. Centralized systems suffer from severe computational bottlenecks as the swarm scales and remain inherently vulnerable to single points of failure. Decentralized control relies on local onboard decision making for each robot [8]. These systems are computationally efficient and highly robust but struggle to maintain reliable communication and lack a cohesive global vision, frequently leading to suboptimal area coverage.

Recognizing the limitations of both paradigms, recent research advocates for hierarchical supervisory architectures [7]. For example, de Koning and Jamshidnejad [9] introduced a framework that intervenes with local controllers only when specific conflicts arise. Agents operate autonomously the rest of the time. This event driven approach successfully leverages the global optimality of a centralized supervisor while retaining the computational speed of decentralized execution.

To navigate the actual assignment of targets within these architectures, modern Multi-Robot Task Allocation (MRTA) frameworks frequently employ market based or auction driven strategies [6]. Tasks are treated as commodities. Robots calculate scalar bids based on local utility functions factoring in travel distance and expected information gain [1]. Traditional Sequential Single Item (SSI) Auctions are computationally lightweight but can yield suboptimal long term routing. Researchers have developed bundle based approaches like the Consensus Based Bundle Algorithm (CBBA) to allow agents to bid on sequences of tasks [8]. Calculating extended task sequences creates dynamic fragility in highly volatile SaR environments because a newly discovered obstacle or a spreading fire can instantly invalidate a precomputed sequence [6]. This forces computationally paralyzing recalculations.

Realistic SaR operations introduce the critical challenge of robot heterogeneity and competing task typologies [14]. A deployed swarm rarely consists of identical units: Agents may possess varying sensor ranges or specialized hardware capabilities. An entire swarm might be capable of spatial exploration while only a subset of specialized agents is equipped to rescue a victim. The allocator must efficiently allocate low priority high volume exploration while staying responsive to time sensitive rescue tasks.

2.1 Synthesis and Identified Knowledge Gap

Progress has been achieved in isolated domains of SaR robotics leaving an important intersection unaddressed. Existing frameworks feature established hierarchical structures for mitigating spatial conflicts [7, 9] alongside robust auction methods for task distribution [1]. They lack a unified lightweight supervisory architecture capable of managing highly volatile heterogeneous missions. Systems currently fall back on computationally heavy sequence planners.

A knowledge gap exists regarding how to combine environment exploration and victim rescuing into a responsive al-

location system within highly dynamic environments. This research addresses this gap by proposing an event driven task allocation architecture. The supervisor intervenes under three conditions: discrete physical conflicts, agent stagnation, or victim discovery. This allows the proposed supervisor to act as a lightweight conflict resolver. It dynamically modulates task weights based on capabilities and utility to ensure global mission optimality. The framework prioritizes critical rescues while maintaining the computational speed of decentralized auction based execution.

3 Problem Formulation

In order to address the identified gap, this section formally defines the multi-robot coordination problem for SaR missions. Specifically, it details the modeling of the dynamic SaR environment, the kinematic and sensing capabilities of the robotic agents, and the behavioral model governing the victims.

3.1 SaR Environment

Following the framework established by [9], the SaR environment E is modeled as a bounded, discretized, two-dimensional cellular grid of dimensions $L_x \times L_y$ (see Fig. 1). Each cell, located at coordinates (x, y) , can be vacant or occupied by static obstacles, fire and SaR robots or victims. Furthermore, each cell possesses a specific structural vulnerability score. The global state of these elements is represented by a set of $L_x \times L_y$ matrices within the simulator.

These environments are procedurally generated using a room-and-tunnel based algorithm inspired by [15]. This generator constructs structured indoor environments characterized by interconnected rooms and tunnels with a starting room always present in the top left corner (see Fig 2). The generation process is highly parameterized via a central configuration module, allowing for systematic variations in global map size, room dimensions, structural vulnerabilities, and initial fire spawning. The complete set of configuration parameters used to guide the procedural generation is defined in Table 1.

Vulnerability is modeled discretely as one of {SAFE, VULNERABLE, HIGH_RISK}. Similarly, a cell in the fire matrix transitions through states defined as {SAFE, FLAMMABLE, BURNING, BURNT}. A burning tile sets all adjacent non-burning tiles to flammable, and at each time step, a flammable tile has a set probability (p_f) of igniting. Once a burning tile has burned for a specified duration (T_f), it is extinguished and becomes permanently burnt.

3.2 SaR Robots

The multi-robot SaR system is composed of a fleet of N agents, denoted as a_i for $i \in \{1, \dots, N\}$. At each discrete simulation time step t , an agent a_i selects a movement action from a finite action space $A = \{\text{Left, Down, Right, Up, Wait}\}$. Based on this selection, the agent's coordinates within the grid environment are deterministically updated.

Each robot is equipped with a local sensor characterized by a scan radius r . At each time step, the robot observes all non-occluded grid cells within this radius to update the perceived

Table 1: Procedural Environment Generation Parameters

Parameter	Description
N_r	Number of rooms to generate
p_{un}	Probability of a room remaining unconnected
w_{start}, l_{start}	Width and length of the agent starting room
N_v	Total number of victims placed
$w_{r,min}, w_{r,max}$	Minimum and maximum room width
$l_{r,min}, l_{r,max}$	Minimum and maximum room length
t_{min}, t_{max}	Minimum and maximum tunnel thickness
$p_{v,r}$	Probability of a room being vulnerable
$s_{v,r}$	Probability a room is HIGH_RISK
$p_{v,t}$	Probability of a tunnel being vulnerable
$s_{v,t}$	Probability a tunnel is HIGH_RISK
N_f	Number of initial fire ignition points
p_f	Probability of fire spreading to a flammable cell
T_f	Duration (time steps) a cell remains burning

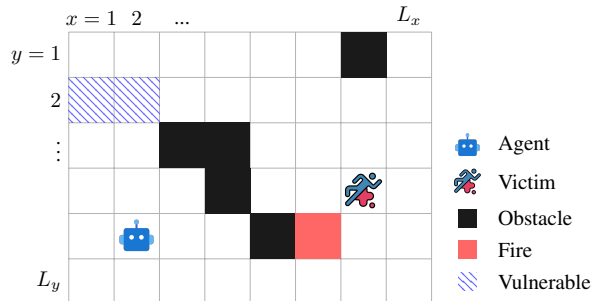


Figure 1: Schematic view of all elements in the SaR environment.

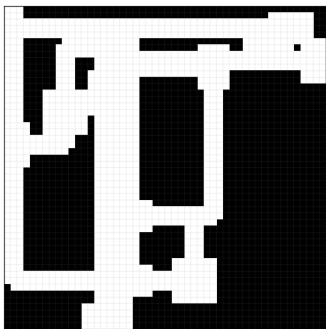


Figure 2: Example 50x50 room-and-tunnel environment.

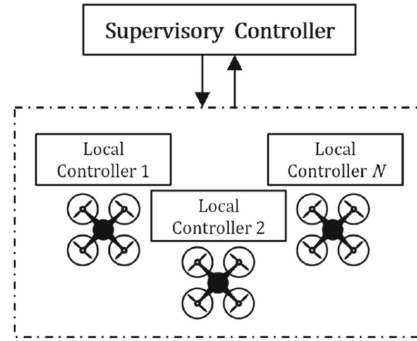


Figure 3: Architecture of the supervisory system [9]

state of the environment, identifying hazards, obstacles, and victim presence. To account for the dynamic nature of the SaR environment (e.g., fire propagation) and the unreliability of older data, the information gathered from each cell is assigned a confidence value equal to s_{acc} . Upon observation, a cell's confidence is set to the robot's scan accuracy. This confidence metric then decays at each subsequent time step with σ , reflecting the increasing spatial uncertainty of regions which have not been observed for a certain amount of time steps.

3.3 Victim Modeling

The initial quantity and distribution of victims within E are unknown to the SaR robots. To simulate realistic human behavior in disaster scenarios, victim agents are modeled using the Game-Theoretic Belief-Desire-Intention (GT-BDI) architecture. This is adopted unchanged from [16].

4 Supervisory Control System

This section details the proposed architecture of the supervisory control system. The search behavior of the SaR swarm is governed by a two-level hierarchical control architecture, building upon the framework established in [9]. The lower operational level relies on decentralized controllers that allow individual robots to autonomously navigate and execute assigned tasks. To ensure the swarm operates cooperatively, a higher-level centralized supervisor oversees the mission. Instead of sharing data directly with one another, the local agents communicate exclusively with this supervisor by submitting scans and triggering discrete events (see Fig 3). The supervisor then processes these events to detect overlaps, resolve spatial conflicts, and auction new targets. By restricting communication to this centralized hub while offloading task execution to the local agents, the architecture benefits from the robustness of decentralized systems with the coordination capabilities of a centralized planner.

4.1 Local MPC Controllers

At the local control level, the agents are controlled using Model Predictive Control (MPC). MPC is chosen as the local controller because its receding-horizon formulation allows agents to optimize competing objectives such as exploration, safety, and path adherence over a short planning

window without committing to long task sequences, consistent with the event-driven philosophy of the supervisory layer. The formulation follows [17], adapted with task-specific objective functions.

Formally, at each time step t , the local controller solves the following finite-horizon optimization problem:

$$\max_{u(t:t+H-1)} \sum_{\tau=t}^{t+H-1} L(\tau) + V(t+H) \quad (1)$$

subject to the environment's grid dynamics and the discrete action space constraints $u(\tau) \in A$, where A is the finite action space defined in Section 3.2. Here, H is the prediction horizon, u is the sequence of control actions, $L(\tau)$ is the stage objective evaluated at each step, and $V(t+H)$ is the terminal objective penalizing the final predicted state.

Depending on the agent's current assignment, the specific formulations of $L(\tau)$ and $V(t+H)$ adapt dynamically. If an agent is idle, meaning that it has not been assigned a task, it functions according to the following stage objective function to be maximized:

$$L_{\text{idle}} = w_d S_d + w_{c,\text{idle}} S_c - w_{s,\text{idle}} P_s - w_{r,\text{idle}} P_r \quad (2)$$

where S_d represents the local discovery score, S_c is the confidence score and P_s denotes the safety penalty. The weights w_d (discovery), $w_{c,\text{idle}}$ (confidence), $w_{s,\text{idle}}$ (safety) and $w_{r,\text{idle}}$ (repulsion) are constant weights and to prioritize actively uncovering new areas while mitigating physical risk, the discovery weight is set significantly higher compared to the safety weight. A smaller confidence weight is applied to fine-tune the agent's certainty in its local state estimation. The repulsion penalty ensures agents avoid coming too close to each other. Additionally, the full MPC formulation for an unassigned agent incorporates a terminal objective to penalize stagnation over the prediction horizon H . Because the agent lacks a targeted goal, the terminal function applies a heavy exploration penalty:

$$V_{\text{idle}} = -w_{\text{exp}} P_{\text{exp}} H \quad (3)$$

where P_{exp} represents the terminal exploration penalty, heavily weighted and scaled by the horizon length. Together, these objective functions ensure that idle agents act as independent mappers, safely and continuously expanding the swarm's known territory until the supervisory controller intervenes with a definitive task.

For an exploration task, the agent gets handed over a path constructed by the supervisor and this path is used in its objective function:

$$L_{\text{target}} = w_{\text{path}} S_{\text{path}} + w_{c,\text{tar}} S_c - w_{s,\text{tar}} P_s - w_{r,\text{tar}} P_r \quad (4)$$

where S_{path} represents the path adherence score and the other scores are as with the function for an idle agent. Because the primary goal during a targeted task is to reach the assigned coordinate efficiently, the path adherence score is prioritized. To avoid getting stuck with other agents, $w_{r,\text{tar}}$ (repulsion) is weighted heavily, while confidence is given lower weight so it does not overpower the trajectory tracking. The path adherence score, S_{path} , dynamically evaluates how well the agent is tracking its assigned route. The

agent calculates the Euclidean distance to every waypoint in the current path to find the closest point. Let $d_{\min}$ be this minimum distance and i_{closest} be the index of that closest waypoint along the path array. To evaluate path adherence, the score is computed as a continuous bounded function:

$$S_{\text{path}} = \min \left(1.0, \max \left(0.0, 0.5 + 0.1 i_{\text{closest}} - 0.05 d_{\min} \right) \right) \quad (5)$$

This continuous formulation creates a flexible, "magnetic" trajectory that facilitates dynamic obstacle avoidance. The 0.5 base score provides headroom for the utility to increase via a forward-driving progress gradient $0.1 i_{\text{closest}}$ or decrease via a spatial detour penalty $0.05 d_{\min}$. By restricting i_{closest} to a local 10-step look-ahead window ($i_{\text{closest}} \in [0, 9]$), this progress gradient is inherently normalized, preventing the score from scaling indefinitely on long paths. Because the detour penalty is weighted lower than the progress reward, the formulation is designed so that the local MPC is able to stray from the path where necessary (for example to avoid another agent) in order to maintain forward momentum. This behavioral tendency follows from the relative weighting of the two terms; we do not separately verify it through trajectory-level analysis, and leave empirical confirmation of detour behavior under hazard encounters to future work. Finally, bounding constraints normalize the score within $[0, 1]$.

Rescuing Capability

The agent may also have the capability to rescue a victim, which is adopted unchanged from [16], where the agent guides the victim to the exit based on the perceived trust of the victim. Unless assigned a rescue task, an agent with this capability functions as mentioned before using MPC.

4.2 Supervisory Auction-Based Controller

The core function of the centralized supervisory controller is to maintain a cohesive global strategy without succumbing to the computational bottlenecks typical of continuous multi-agent path planners. At each time step, the supervisor aggregates discrete local sensor scans transmitted by the agents to continuously update a global belief map and a global exploration map of the environment. Confidence functions the same as with agents, where the supervisor has a decay value σ_{sup} .

This architecture does not allow inter-agent communication. All data flows bidirectionally between individual robots and the central supervisor. This design choice provides three major advantages: (1) it reduces the computational overhead required on local onboard processors, (2) it lowers hardware costs by removing the need for complex mesh-networking modules, and (3) it enhances systemic robustness, as the loss of any individual robot does not fracture a local communication chain.

The supervisor is 'lightweight' because it remains dormant during normal operation and only reallocates tasks when specific events occur. These events are triggered under three scenarios:

- **Spatial Conflict:** If an agent's local sensors detects another agent within its scan radius, it raises a spatial con-

flict. Both agents do this. This spreads the robots out over the environment.

- **Impatience:** To resolve "soft" conflicts, the supervisor monitors the information gain of each agent. If the supervisor detects that an active agent has failed to discover new territory (territory that has never been within the scan radius of any agent) for the global exploration map over a set threshold T_{imp} , it flags the agent with "impatience".
- **Victim Discovery:** If a victim is identified within any transmitted local scan, the supervisor raises a high-priority intervention event.

Once the event queue is populated, the supervisor initiates a resolution phase by generating a discrete set of tasks. For spatial conflicts and impatience events, the supervisor generates *exploration tasks*. This is achieved by extracting the current global frontiers [11], distributing weighted points across these boundaries, and clustering them into navigable exploration goals. Conversely, a victim discovery event instantaneously generates a *rescue task* localized at the victim's coordinates.

With the task set generated, the supervisor allocates them using a modified auction-based framework. An algorithmic overview is detailed in Appendix A. A Sequential Single-Item (SSI) auction is used because it can adapt to dynamic task generation well. More algorithmically optimal allocation methods such as the Hungarian method do not scale well with high amount of tasks and agents. In standard MRTA, SSI auctions often allow agents to bid on and accumulate a queue of multiple future tasks. However, as demonstrated by Wei et al. [6], allowing agents to queue tasks severely decrements objective performance in highly dynamic Search and Retrieval scenarios. Pre-allocated task queues result in rigid, heavy workloads that prevent agents from responding efficiently to suddenly discovered targets, increasing overall mission time.

To capture the performance benefits of responsive decentralization while maintaining hierarchical coordination, the proposed allocator modifies the traditional SSI auction by enforcing a strict *single-task capacity limit* per agent. Agents are only permitted to hold one active task at a time. During an auction, the supervisor only considers agents that are currently engaged in an event (i.e., those that are "impatient" or too close to some other agent) as eligible bidders. However, if a critical rescue task is generated, this restriction is lifted, and all agents are pulled into the auction to ensure the fastest possible response.

Task assignment is formulated as a utility maximization problem. The supervisor calculates a utility score $U_{i,j}$ for each eligible agent i bidding on task j , defined as:

$$U_{i,j} = (B_j \cdot C_{i,j}) - T_{i,j} - P_{i,j} \quad (6)$$

Where:

- B_j (**Base Reward**): An inherent non-negative scalar reward for completing the task. The reward value is defined by the task type, denoted as $R_{explore}$ for mapping

and R_{rescue} for victim retrieval. Rescue tasks are assigned a much higher base reward to guarantee they take priority over exploration.

- $C_{i,j}$ (**Capability Multiplier**): A strictly non-negative scalar value $C_{i,j} \in [0, 1]$ representing the capability of agent i to execute task j . This multiplier directly scales the base reward. It also serves as a hard boundary for agents to execute tasks. If an agent is incapable of executing a task (i.e., $C_{i,j} = 0$), it is excluded from the bidding pool for that specific assignment.
- $T_{i,j}$ (**Travel Cost**): The Euclidean distance from agent i to the location of task j . While Euclidean distance is an under-approximation of the true path length in cluttered environments, it serves as a computationally lightweight heuristic sufficient for the supervisor's high-level auction. This standard cost term naturally favors assigning tasks to spatially closer capable agents.
- $P_{i,j}$ (**Spatial Penalty**): Inspired by Burgard et al. [1], this non-negative penalty drives swarm dispersion by reducing the utility of tasks located near other agents. It decays exponentially with distance (scaled by λ_s), with its maximum magnitude governed by proximity weights to an active target (w_{target}) or an idle agent (w_{agent}).

Rather than utilizing normalization, the terms $(B_j, T_{i,j}, P_{i,j})$ are scaled against each other using empirically tuned parameters. This scaling ensures that spatial penalties and travel distances do not accidentally overpower the base rewards and maintain the high-level goals of the mission.

By evaluating this utility function, the supervisor resolves spatial and algorithmic conflicts, dispersing the swarm for optimal exploration while retaining the authority to dynamically snap the most capable agents toward suddenly discovered victims.

5 Results

This section presents a comparative evaluation of the proposed supervisory framework against two unsupervised, decentralized baselines. The simulations are implemented via Python 3.12.13 on a plugged-in laptop with an Intel Core i7 Processor with a 2.40 GHz frequency. The results are divided into two phases: exploration and target discovery, and the complete mission pipeline incorporating physical victim rescue.

5.1 Exploration and Target Discovery

To isolate exploration efficiency, the simulation was constrained to a maximum of 300 time steps, terminating early if all victims were successfully located. We compare the proposed Supervised architecture against two baselines: a standard Unsupervised swarm relying solely on local, isolated perception, and a Shared variant in which agents operate without a central allocator but instantaneously share a global perception map. It is included to isolate the benefit of active task allocation from the benefit of passive shared awareness alone. By contrast, the proposed supervisor only requires one-directional scan uplinks from agents, with sparse task

assignments sent back only on intervention; a star topology with $O(N)$ communication complexity, versus the $O(N^2)$ dense broadcasting required to maintain a globally shared map across all peers. While the supervisor still relies on a stable link to function, its bandwidth footprint is drastically lower, making it a substantially more feasible middle ground between fully isolated autonomy and continuous global synchronization.

All three configurations used the agent architecture from Section 4.1, with 4 agents initialized from a shared starting region with an equal 1.0 capability multiplier for exploration tasks, and were evaluated on the same 216 randomized map seeds. The maps had a size of 60x60 and victims are placed and remain at static locations; full parameters are listed in Appendix B.

To ensure a direct, paired comparison, all three configurations were evaluated on the same set of 216 randomized map configurations (seeds), with $N_f = 1$. Table 2 illustrates the average time steps required to hit specific exploration milestones.

Table 2: Average number of simulation time steps necessary to reach certain exploration milestones, rounded

Milestone	50%	70%	80%	85%	90%
Supervised	81	108	126	136	151
Unsupervised	92	137	165	183	201
Shared	80	108	128	139	153

Endgame Bottleneck and Swarm Dispersion

Table 2 shows a widening gap between Supervised and Unsupervised over the course of a mission: only 13% at the 50% exploration mark, but 25% by 90%. This reflects an “endgame bottleneck” in decentralized swarms, where agents redundantly re-search cleared areas or become trapped in constrained geometries as unmapped space shrinks. Shared tracks Supervised closely through the 50% and 70% milestones (80 and 108 steps), showing that early dispersion is driven mainly by global awareness; the two only diverge from 80% onward, where active conflict resolution begins to outperform passive awareness slightly.

This dispersion effect is reflected in the Mean Inter-Agent Distance (MIAD), which increased 9.3% under supervision relative to Unsupervised (19.69 vs. 18.02, $p < 0.001$), though the gain over Shared was smaller and not significant ($p = 0.139$). A more direct measure of wasted effort is *Exploration Redundancy*, the fraction of an agent’s observed cells that duplicate the rest of the swarm’s coverage, normalized against total global coverage:

$$R_{\text{normalized}} = \frac{\left(\sum_{i=1}^N E_i\right) - E_{\text{global}}}{E_{\text{global}} \cdot (N - 1)} \quad (7)$$

where E_i is the number of cells observed by agent i and E_{global} is the number of distinct cells observed by the swarm. A score of 0 indicates no duplicate coverage; values closer to 1 indicate increasing overlap. The Supervised architecture achieved an absolute redundancy of 0.355, versus 0.623

for Unsupervised and 0.461 for Shared, representing a 43% reduction in duplicate coverage relative to the Unsupervised baseline. Because absolute redundancy naturally accumulates over time, this metric is susceptible to early-stoppage bias, as the Supervised fleet completes missions faster and is less prone to timeout. To decouple spatial efficiency from mission duration, the redundancy accumulation rate per time step was evaluated. Controlling for time, the Supervised fleet maintained a lower rate of redundant coverage (0.0019 per step) compared to both Unsupervised (0.0024 per step, $p < 0.001$) and Shared (0.0022 per step, $p < 0.001$). This indicates that shared awareness alone cannot prevent agents from gravitating toward the same frontiers; an active allocation mechanism is required.

Target Discovery Reliability and Speed

We also report *Exploration Velocity*, the percentage of the map cleared per time step. Supervised achieved 0.549%/step versus 0.378%/step for Unsupervised ($p < 0.001$, a 45% relative increase), while Shared reached an intermediate 0.495%/step, an 11% improvement over Unsupervised, placing roughly three-quarters of the supervisor’s advantage on active allocation rather than shared awareness.

This is reflected directly in victim discovery, detailed in Table 3.

Supervised achieved a 91.7% success rate within the 300-step limit, versus 50.5% for Unsupervised and 75.9% for Shared. Among successful runs, Supervised located all victims 24% faster than Unsupervised and still 4% faster than Shared ($p = 0.027$). Supervised also showed tighter variance in completion time (43.93 vs. 53.37 and 50.03 steps), indicating more predictable performance across map layouts. Together, these results show that while much of the supervisor’s benefit over a naive decentralized swarm comes from global awareness, this alone is insufficient to match supervised performance, which is achieved with far less communication overhead than full map synchronization requires.

Computation Time

Supervision introduced no meaningful computational overhead relative to the Unsupervised baseline (1.076 s vs. 1.084 s, $p = 0.338$). The supervisor intervened an average of 50.6 times per mission, resolving taking an average of just 11 milliseconds. This is well below typical actuation and communication delays in physical robotic systems, meaning the supervisory response time would not be the limiting factor in a real deployment.

Figure 4b and 4c emphasize the operational superiority of the supervised approach: Supervised consistently locates nearly all victims with minimal variance, while Unsupervised shows a wider spread of detection times and a higher chance of leaving victims undiscovered. Figure 4a shows the corresponding reduction in spatial conflicts under supervision.

Safety Outcomes

The local MPC objective penalizes unsafe behavior via P_s and P_r , reflecting the safety/performance trade-off central to the research question; we verify this directly via collision and hazard statistics. Supervised reduced inter-agent collisions by 94% relative to Unsupervised (0.99 vs. 15.58,

Table 3: Average victim discovery performance and paired statistical analysis across 216 simulated runs. *Time to All Found* and its standard deviation are computed over successful runs only.

Architecture	Absolute Performance		Paired Analysis (vs. Supervised)		
	Success Rate	Time to All Found	Paired Diff. (μ , M)	95% CI	p -value
Supervised	91.7%	177.59 ± 43.93	—	—	—
Unsupervised	50.5%	220.81 ± 53.37	$-52.51, -51.00$	$[-64.21, -40.82]$	< 0.001
Shared	75.9%	184.02 ± 50.03	$-9.30, -8.00$	$[-17.55, -1.05]$	0.027

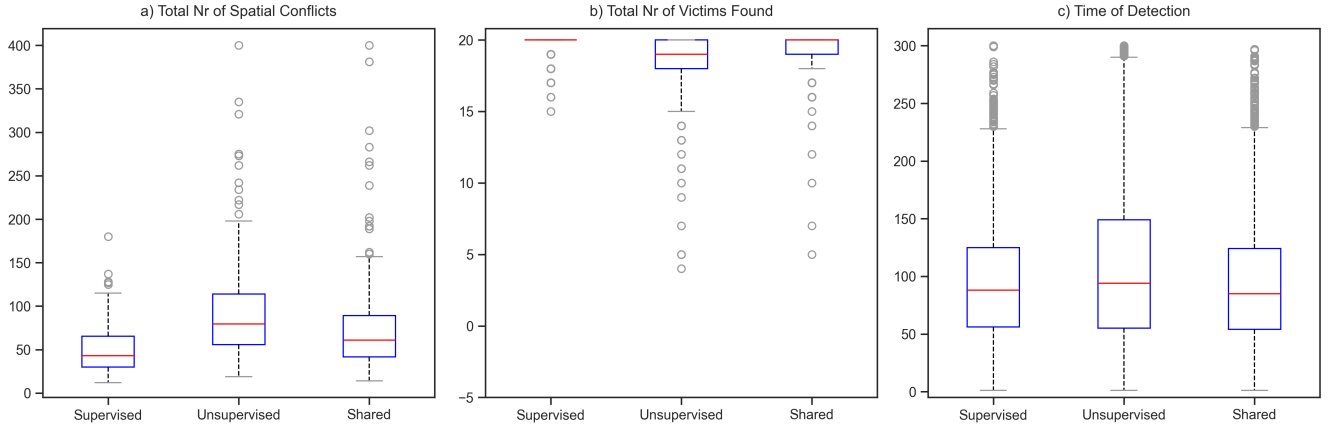


Figure 4: Comparative distributions across 216 simulation runs. **a)** Number of spatial conflicts, **b)** Total victims found, **c)** Individual victim detection times

$p = 0.008$) and total damage (measured as per-episode harm count as detailed in [18]) by 25% ($p < 0.001$), with similar but non-significant reductions in fire collisions and infeasible states. Relative to Shared, Supervised still reduced total damage by 6% ($p = 0.039$) and collisions by 86% (not significant, $p = 0.068$), again suggesting that active conflict resolution, not shared awareness, drives most of the safety benefit. This benefit holds as the environment becomes more hazardous: across a sweep of number of initial fire points ($N_f \in \{1, 3, 5, 10\}$), Supervised reduced inter-agent collisions by 92–94% relative to Unsupervised at every level tested, showing negligible degradation even as overall success rates declined for all architectures.

One result runs counter to this trend: the fraction of mission time in vulnerable terrain was higher under supervision ($p < 0.001$ vs. both baselines), a consequence of shorter missions rather than greater absolute exposure; absolute time in vulnerable terrain remains lower under supervision (286.68 vs. 364.29 steps for Unsupervised). Overall, supervision improves safety alongside performance rather than trading one for the other.

Robustness to Increased Fire

Beyond safety, broader mission performance also degrades gracefully under increasing hazard. Using the homogeneous fleet from Section 5.1, as N_f increases from 1 to 10, success rate declines for all architectures (Supervised: 91.7% $\rightarrow$ 54.6%; Unsupervised: 50.5% $\rightarrow$ 31.9%; Shared: 75.9% $\rightarrow$ 44.4%), and the supervisor’s relative advantage narrows in

both success rate (41 to 23 percentage points) and exploration velocity (45% to 22%), as fire-induced obstructions limit how far agents can disperse. Supervised nonetheless outperforms both baselines at every fire density tested.

5.2 Heterogeneous Fleet

To evaluate whether the supervisory framework generalizes beyond a homogeneous swarm, we repeated the exploration evaluation with a heterogeneous fleet of four agents: Two agents as used in Section 5.1, one agent with the same objective functions but with SDD-TMPC [17] and one rescue-capable agent. Here the capabilities for exploration and rescuing are set to 1.0 and 0.0 respectively, except for the rescue agent, which had a rescuing capability of 1.0. This was again compared against the Unsupervised and Shared baseline across 100 paired runs, now always running for 300 simulation steps, the simulation parameters are detailed in Appendix B. Victims are configured to follow a rescue agent immediately upon contact, isolating allocation performance from victim trust dynamics.

Exploration Performance

The exploration-coordination benefits established in Section 5.1 persist under heterogeneity. Supervised achieved a 100% success rate within the mission limit, compared to 83% for Unsupervised and 90% for Shared. Exploration milestones were reached faster under supervision (time to 50%: $p < 0.001$; time to 90%: $p < 0.001$), and exploration velocity was higher than both baselines ($p < 0.001$ in each case).

MIAD increased relative to Unsupervised (21.82 vs. 20.45, $p < 0.001$), though the difference relative to Shared was not significant ($p = 0.788$), consistent with the earlier finding that early dispersion is driven substantially by shared awareness rather than active allocation alone. These results confirm that introducing agent heterogeneity does not erode the supervisor’s core exploration-coordination benefit.

Rescue Performance

Rescue outcomes tell a different story. Supervised located more victims than either baseline (7.05 vs. 6.17 for Unsupervised, 6.23 for Shared) but saved fewer (2.21 vs. 2.83 and 2.88, $p < 0.001$ in both cases). This gap reflects a capacity bottleneck: with only one rescue-capable agent, the exploration gains from supervision do not translate into a corresponding rescue gain, and in this configuration come at the expense of it.

6 Responsible Research

This section discusses the reproducibility of the methods utilized in this study and reflects upon the ethical implications of deploying an autonomous supervisory system in dynamic SaR environments.

6.1 Reproducibility

To ensure the transparency and replicability of the findings presented in this study, the methodologies, simulation environment, and multi-agent controllers have been comprehensively detailed. Furthermore, the complete simulation framework, including the environment generator, MPC controllers, and data-evaluation scripts, has been made open-source and is publicly available at <https://github.com/tigo-sc/rp-2026-autonomous-SaR>. All critical simulation parameters, including procedural environment generation rules, objective function weights, and supervisory parameters, are explicitly quantified Appendix B.

Furthermore, to mitigate the variability of procedural map generation and ensure statistical significance, the comparative evaluations were conducted across 216 fixed, randomized map configurations (seeds). By documenting the exact formulas for derived metrics such as Exploration Velocity and MIAD, future researchers have the complete theoretical and parametric framework required to independently verify, reconstruct, and build upon the proposed architecture.

6.2 Ethics

Deploying autonomous systems in life-critical SaR scenarios raises ethical questions about accountability. In the event of a system failure, whether it lies with the system engineers, an impossible environment or the supervisory system itself, it remains a complex ethical dilemma.

However, an ethical advantage of the proposed architecture is its explainability. Unlike black-box machine learning models, the event-driven auction mechanism relies on explicit mathematical utility functions. Every allocation decision made by the supervisor can be traced back to the specific robot data and cost weights present at that exact time step, which facilitates failure analysis.

Finally, there is an inherent ethical risk associated with the “sim-to-real” gap. Evaluating life-critical systems solely within a grid simulation abstracts away real-world physical hazards, sensor noise, and communication degradation. Acknowledging these limitations prevents overconfidence in the system’s operational reliability. Consequently, any future transition from this simulated framework to physical deployment must involve rigorous testing and maintain a human override protocol to guarantee operational safety and responsible disaster response.

Generative Artificial Intelligence was used to improve the grammatical structure of the paper. Output was always rigorously checked.

7 Conclusions

7.1 Conclusion

The central objective of this research was to determine how a lightweight supervisory framework can dynamically allocate tasks within a multi-robot team to balance global area exploration and immediate, targeted victim retrieval. This was achieved through an event-driven, centralized Sequential Single Item (SSI) auction mechanism that remains dormant until triggered by spatial conflicts, agent stagnation, or victim discovery, allowing the system to override decentralized, selfish agent behavior only when necessary.

The proposed architecture was benchmarked against a decentralized, unsupervised baseline and an idealized Shared-map baseline across 216 simulated SaR missions. The supervisor outperformed the unsupervised baseline on victim discovery, raising success rate from 50.5% to 91.7% and reducing completion-time variance by 21%. It also outperformed the Shared baseline despite the latter’s far heavier communication requirements, indicating that the observed gains stem from active allocation rather than situational awareness alone. This dispersion benefit is supported quantitatively by Mean Inter-Agent Distance and Exploration Redundancy, with active allocation reducing duplicate coverage by 43% relative to the unsupervised baseline, and persisted when the fleet was extended to heterogeneous agents.

Extending the evaluation to the complete rescue pipeline, however, revealed a trade-off: while supervision increased the rate of victim discovery, fewer victims were ultimately saved, as a single rescue agent could not keep pace with the accelerated discovery rate. This indicates that the proposed mechanism does not yet translate its exploration gains into improved retrieval throughput under the simulated conditions.

7.2 Limitations and Future Work

A primary limitation is the assumption of an idealized communication network, in which the supervisor receives all environmental scans instantaneously. In practical deployments, latency, intermittent connectivity, or packet loss would degrade the accuracy of the supervisor’s belief map and auction weights. Future work should investigate task allocation under lossy or localized communication. We note that the proposed architecture’s star-topology communication, where

agents transmit only local scans and receive sparse assignments, is inherently more robust to such degradation than the dense synchronization required by the Shared baseline, though this was not evaluated directly.

A second limitation concerns the scope of the baseline comparison: the present evaluation exclusively benchmarks against decentralized, no-supervisor baselines. While this demonstrates that active supervision improves performance relative to fully decentralized coordination, it does not establish superiority over other supervisory or auction-based allocation methods, such as CBBA-style bundle bidding or alternative event-triggering schemes. Benchmarking against such methods is left for future work.

The rescue-capacity bottleneck identified in this study presents a third, more immediate direction for future work. The current utility function does not account for rescue-agent throughput when prioritizing discovery, allowing exploration to outpace retrieval capacity. Future work should investigate utility formulations that explicitly reason about rescue-agent availability, and evaluate larger or more capable rescue sub-teams able to match the swarm’s exploration throughput. More broadly, the task profiles examined here remain limited relative to real-world deployment needs; expanding the auction framework to accommodate a wider range of specialized capabilities and complex tasks, such as debris excavation or structural reinforcement, is an important direction for scaling this framework toward real-world search and rescue operations.

A Algorithmic Framework

This appendix details the comprehensive pseudo-code for the event-driven supervisory task allocation framework discussed in Section 4.2. Algorithm 1 outlines the process of event resolution and the subsequent SSI auction.

B Simulation Parameters

This appendix consists of the parameters used for the simulations of section 5.1 and 5.2 in Table 4 and Table 5 respectively.

Table 4: Simulation parameters homogeneous simulations

Parameter	Value	Parameter	Value
L_x, L_y	60, 60	$p_{v,r}$	0.3
N_r	12	$s_{v,r}$	0.2
p_{un}	0.0	$p_{v,t}$	0.4
w_{start}, l_{start}	3, 3	$s_{v,t}$	0.2
N_v	20	N_f	1
$w_{r,min}, w_{r,max}$	4, 8	p_f	0.02
$l_{r,min}, l_{r,max}$	4, 8	T_f	100
t_{min}, t_{max}	2, 3		
$w_d, w_{c,idle}$	100, 3	$w_{path}, w_{c,tar}$	100, 1
$w_{s,idle}, w_{r,idle}$	10, 50	$w_{s,tar}, w_{r,tar}$	5, 200
s_{acc}, r	0.9, 4	σ_{agent}	0.01
T_{imp}	8	σ_{sup}	0.01
$R_{explore}, R_{rescue}$	100, 10000	w_{target}, w_{agent}	500, 200

Algorithm 1 Event-Driven Supervisory Task Allocation

Require: Set of agents A , Set of unresolved events E , Current task backlog B

```

1:  $T_{new} \leftarrow \emptyset$ ,  $A_{bid} \leftarrow \emptyset$ , ExtractFrontiers  $\leftarrow$  False
2: // Phase 1: Event Resolution
3: while  $E \neq \emptyset$  do
4:    $e \leftarrow E.pop()$ 
5:   if  $e.type \in \{\text{SpatialConflict}, \text{LowProgress}, \text{PathBlocked}\}$  then
6:      $A_{bid} \leftarrow A_{bid} \cup \{e.agent\}$ 
7:     ExtractFrontiers  $\leftarrow$  True
8:   else if  $e.type == \text{VictimFound}$  then
9:      $T_{new} \leftarrow T_{new} \cup \{\text{RescueTask}(e.location)\}$ 
10:     $A_{bid} \leftarrow A$   $\triangleright$  All agents pulled into auction
11:   else if  $e.type \in \{\text{TaskCompleted}, \text{VictimRescued}, \text{VictimDied}, \text{RescueAbandoned}\}$  then
12:      $e.agent.status \leftarrow \text{IDLE}$ 
13:   end if
14: end while
15: if ExtractFrontiers then
16:    $T_{new} \leftarrow T_{new} \cup \text{ExtractGlobalFrontiers}()$ 
17: end if
18:  $T_{avail} \leftarrow T_{new} \cup B$ 
19: // Phase 2: Utility-Based Auction
20: while  $T_{avail} \neq \emptyset$  and  $A_{bid} \neq \emptyset$  do
21:    $U_{max} \leftarrow -\infty$ ,  $a_{best} \leftarrow \text{null}$ ,  $t_{best} \leftarrow \text{null}$ 
22:   for  $t \in T_{avail}$  do
23:      $P_{spatial} \leftarrow \text{CalculateBurgardPenalty}(t, A \setminus A_{bid})$ 
24:     for  $a \in A_{bid}$  do
25:       if  $a.capability(t) == 0$  then continue
26:     end if
27:      $R_{base} \leftarrow \text{BaseReward}(t) \times a.capability(t)$ 
28:      $C_{travel} \leftarrow \text{Distance}(a, t)$ 
29:      $U \leftarrow R_{base} - C_{travel} - P_{spatial}$ 
30:     if  $U > U_{max}$  then
31:        $U_{max} \leftarrow U$ ,  $a_{best} \leftarrow a$ ,  $t_{best} \leftarrow t$ 
32:     end if
33:   end for
34: end for
35: if  $a_{best} \neq \text{null}$  then
36:   Assign( $a_{best}, t_{best}$ )
37:    $A_{bid} \leftarrow A_{bid} \setminus \{a_{best}\}$ 
38:    $T_{avail} \leftarrow T_{avail} \setminus \{t_{best}\}$ 
39: else
40:   break  $\triangleright$  No valid paths/capabilities remaining
41: end if
42: end while
43:  $B \leftarrow T_{avail}$   $\triangleright$  Unassigned tasks return to backlog
44: for  $a \in A_{bid}$  do
45:    $a.status \leftarrow \text{IDLE}$   $\triangleright$  Unassigned agents map locally
46: end for

```

Table 5: Simulation parameters heterogeneous simulations

Parameter	Value	Parameter	Value
L_x, L_y	50, 50	$p_{v,r}$	0.3
N_r	15	$s_{v,r}$	0.2
p_{un}	0.0	$p_{v,t}$	0.4
w_{start}, l_{start}	3, 3	$s_{v,t}$	0.2
N_v	10	N_f	1
$w_{r,min}, w_{r,max}$	4, 8	p_f	0.01
$l_{r,min}, l_{r,max}$	4, 8	T_f	100
t_{min}, t_{max}	2, 3		
$w_d, w_{c,idle}$	100, 3	$w_{path}, w_{c,tar}$	100, 1
$w_{s,idle}, w_{r,idle}$	10, 50	$w_{s,tar}, w_{r,tar}$	5, 200
s_{acc}, r	0.9, 4	σ_{agent}	0.01
T_{imp}	8	σ_{sup}	0.01
$R_{explore}, R_{rescue}$	100, 10000	w_{target}, w_{agent}	500, 200

References

- [1] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, Jun. 2005. DOI: 10.1109/tro.2004.839232.
- [2] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013, ISSN: 0921-8890. DOI: <https://doi.org/10.1016/j.robot.2013.09.004>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S092188901300167X>.
- [3] F. Surma and A. Jamshidnejad, "State-dependent dynamic tube MPC: A novel tube MPC method with a fuzzy model of disturbances," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 4, pp. 1319–1354, 2025.
- [4] O. Bastani, S. Li, and A. Xu, "Safe RL via statistical model predictive shielding," in *Robotics: Science and Systems*, 2021, pp. 1–13.
- [5] C. Wang et al., *Multi-robot system for cooperative exploration in unknown environments: A survey*, 2025. arXiv: 2503.07278 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2503.07278>.
- [6] C. Wei, K. Hindriks, and C. Jonker, "Dynamic task allocation for multi-robot search and retrieval tasks," English, *Applied Intelligence: the international journal of artificial intelligence, neural networks, and complex problem-solving technologies*, vol. 45, no. 2, pp. 383–401, 2016, ISSN: 0924-669X. DOI: 10.1007/s10489-016-0771-5.
- [7] Y. Liu, M. Ficocelli, and G. Nejat, "A supervisory control method for multi-robot task allocation in urban search and rescue," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 254–265, 2015.
- [8] B. Rossano, J. Lim, and J. P. How, *Uncertainty-aware multi-robot task allocation with strongly coupled inter-robot rewards*, 2025. arXiv: 2509.22469 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2509.22469>.
- [9] C. de Koning and A. Jamshidnejad, "Hierarchical integration of model predictive and fuzzy logic control for combined coverage and target-oriented search-and-rescue via robots with imperfect sensors," *Journal of Intelligent and Robotic Systems*, vol. 107, no. 40, 2023.
- [10] M. L. M. Patrício and A. Jamshidnejad, "Dynamic mathematical models of theory of mind for socially assistive robots," *IEEE Access*, vol. 11, pp. 103 956–103 975, 2023.
- [11] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, IEEE, 1997, pp. 146–151.
- [12] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 1160–1165. DOI: 10.1109/IROS.2008.4650734.
- [13] P. Yao and Z. Zhao, "Improved glasius bio-inspired neural network for target search by multi-agents," *Information Sciences*, vol. 568, pp. 40–53, 2021, ISSN: 0020-0255. DOI: <https://doi.org/10.1016/j.ins.2021.03.056>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025521003078>.
- [14] L. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998. DOI: 10.1109/70.681242.
- [15] G. O. Flores-Aquino, J. D. D. Ortega, R. Y. A. Arvizu, R. L. Muñoz, O. O. Gutierrez-Frias, and J. I. Vasquez-Gomez, *2d grid map generation for deep-learning-based navigation approaches*, 2021. arXiv: 2110.13242 [cs.R0]. [Online]. Available: <https://arxiv.org/abs/2110.13242>.
- [16] M. Grouev, "Human-robot interaction in search and rescue: Modeling victim behaviour and trust," Bachelor's Thesis, Delft University of Technology, 2026.
- [17] F. Keerberg, "Perception-driven tube scaling for state-dependent dynamic tube-based MPC," Bachelor's Thesis, Delft University of Technology, 2026.
- [18] J. Dimsa, "Shielded rl in partially observable search-and-rescue," Bachelor's Thesis, Delft University of Technology, 2026.