PECMAN: Perception-enabled Collaborative Multi-Agent Navigation in Unknown Environments

Tianchonghui Fang[1], Shaunak Roy[1] and Shalabh Gupta[1]

***Abstract*—Most path planners assume fully known, static environments, assumptions that fail when robots navigate in dynamic and partially observable environments. SMART-3D addresses these issues by real-time replanning, where it morphs the underlying RRT* tree whenever new obstacles or structures are discovered in the environment. Instead of rebuilding the tree entirely from scratch, SMART-3D prunes invalid nodes and edges and subsequently repairs the disjoint subtrees at hot-nodes to find a new path, thus providing high computational efficiency for real-time adaptability. We extend SMART-3D to perception-enabled collaborative multi-agent navigation (PECMAN) in unknown environments. PECMAN is built upon distributed tree morphing and shared perception strategies, where each agent reacts to environmental changes and morphs its respective tree to replan its path, while simultaneously broadcasting newly discovered structures to other agents, thus enabling them to proactively replan even in areas that have not yet been explored by them. This approach reduces redundant reactions and unnecessary replannings of the agents due to improved situational awareness. The performance of PECMAN was evaluated by $28{,}000$ multi-agent simulations on seven 2D scenarios with different case studies. The results show that PECMAN achieves up to 52% reduction in the team-completion time, while maintaining $\sim 100\%$ success rates. Finally, PECMAN was tested by real experiments on two autonomous robots in a building environment.**

I. INTRODUCTION

Recent years have witnessed an unprecedented increase in the applications of unmanned vehicles (e.g., unmanned air vehicles (UAVs) and unmanned underwater vehicles (UUVs)) including marine life exploration [1][2], seafloor mapping [3], [4], coverage planning [5], [6], ocean sensing [7], agriculture [8], [9], bridge monitoring [10], oil spill cleaning [11], [12], human-robot interaction [13], target tracking [14], [15], [16], mine countermeasures [17], [18], energy-constrained exploration [19], terrain mapping [20], and surveillance [21].

A. Motivation

In most complex environments, safe and reliable navigation [22] becomes challenging because robots have to navigate around unknown static (e.g., walls and rocks) and dynamic (e.g., pedestrians and other robots) obstacles. In this regard, a recent algorithm, called Self Morphing Adaptive Replanning Tree (SMART) [23], introduced an efficient mechanism for real-time replanning when dynamic obstacles enter the robot's Local Reaction Zone (LRZ). SMART-3D [24] extended SMART to 3D workspaces. Instead of rebuilding the tree entirely from scratch, it morphs the current tree by first pruning the affected nodes and edges, and then repairing the resulting disjoint subtrees by making local reconnections at hot-nodes (nodes adjacent to different subtrees), to find a new path. This local morphing provides high computational efficiency for real-time adaptability. SMART-3D is validated on a single robot in dynamic, partially observable environments, where obstacles are discovered online through onboard sensing. While SMART-3D handles unknown and dynamic environments for a single robot, its multi-robot deployment introduces several inter-agent coordination and information-sharing challenges [25]. For example, due to a lack of coordination and information sharing strategy, it is possible that i) multiple agents decide to go towards the same narrow corridor and face a deadlock, or ii) each agent independently rediscovers the same structures that have already been observed by neighboring agents, thus wasting resources and delaying planning. To address the above problems, this paper extends the single-agent adaptive navigation strategy of SMART-3D to develop a novel algorithm called perception-enabled collaborative multi-agent navigation (PECMAN).

B. Related Work

Three lines of multi-agent research are relevant. **Multi-Agent Path Finding (MAPF).** MAPF [26], [27], [28] methods are scalable for coordinating hundreds of agents on known graphs with synchronized discrete timesteps. Hönig et al. [29] note that both these assumptions break down on physical robots in real environments. Conflict-based Search (CBS) [26] finds optimal solutions, but it suffers from computational explosion in narrow corridors [30]. Priority Inheritance with Backtracking (PIBT) [27] scales to thousands of agents via priority inheritance. Lazy Constraints Addition search for MAPF (LaCAM) [28] adds lazy constraints to the search problem. All the above methods require a known grid and discrete timesteps. Lifelong MAPF [31] handles persistent task streams but still needs a known graph. In contrast, PECMAN handles unknown continuous environments, a regime that the above methods do not address.

Sampling-based replanning. Sampling-based multi-agent planning [32], [33] methods handle continuous workspaces using RRT-based exploration, but some of these methods assume that the environment is known. Some methods can also handle dynamic obstacles. Dynamic RRT (DRRT) [34] prunes the invalid tree-structures due to dynamic obstacles and regrows the tree to replan the path. RRTX [35] replans by propagating the cost changes globally. Extended RRT (ERRT) [36] grows

[1] T. Fang (email: tianchonghui.fang@uconn.edu), S. Roy (email: shaunak.roy@uconn.edu) and S. Gupta (email: shalabh.gupta@uconn.edu) are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269, USA. Corresponding author: S. Gupta.

the tree using a bias toward the old plan. These methods require significant modifications to the tree structure every time a change occurs. On the other hand, SMART-3D [24] prunes invalid nodes and edges in a local Critical Pruning Region (CPR) and reconnects disjoint subtrees via hot-nodes, thus preserving most of the tree structure. PECMAN keeps the local prune and repair properties of SMART-3D and builds a multi-agent coordination layer on top.

Reactive coordination. Optimal Reciprocal Collision Avoidance (ORCA) [37] computes pairwise velocity constraints. It operates in open continuous spaces but can face deadlocks in narrow corridors where no mutually feasible velocity exists. Dynamic Window Approach (DWA) [38] has the same limitation. In contrast, PECMAN introduces a Global King/flee/push coordination layer to resolve such conflicts by sequencing robots through priority inheritance, where the King agent passes first while the others flee, with the RRT* tree providing fallback paths after displacement.

C. Contributions

This paper develops a novel algorithm called PECMAN for collaborative multi-agent navigation in unknown environments. The key contributions of this paper are as follows:

1) **Shared perception.** PECMAN enables shared perception in unknown environments. This means that if any agent discovers a static structure (e.g., a wall) in the environment using its LiDAR, it broadcasts this information to all other agents who, upon receiving such early notice, react accordingly and update their plans proactively before detecting this structure themselves. This can significantly reduce the trajectory lengths and times of these agents due to early planning and by not traveling in the wrong direction in the unknown realm.
2) **Distributed tree morphing.** PECMAN scales the single agent tree morphing (i.e., prune-and-repair) strategy of SMART-3D to distributed tree morphing for a team of robots. In this distributed strategy, each agent morphs its own RRT* tree in response to a) nearby dynamic obstacles and b) static obstacles discovered either by its own sensor or based on the information received from other agents. In case b), it permanently deletes the tree-portions that are blocked by static obstacles.
3) **Narrow corridor coordination.** PECMAN introduces a Global King Priority layer to resolve deadlocks at narrow corridors through flee/push behaviors and multi-directional retreat without the need to modify any planning trees. This mechanism is similar to PIBT's priority inheritance [27] but in continuous coordinates rather than on a grid.

The performance of PECMAN was validated on seven procedurally generated 2D scenarios, ranging from $32\,\text{m}\times32\,\text{m}$ to $192\,\text{m}\times192\,\text{m}$ with up to 4 agents and 30 pedestrians, totaling $28{,}000$ multi-agent trials, considering two different repair strategies and two different sensing modes: shared and independent. Finally, PECMAN was tested by real experiments on two autonomous robots in a building environment.

II. The PECMAN Algorithm

A. Problem formulation

Consider a set $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, ...\mathcal{R}_N\}$ of $N \in \mathbb{Z}_+$ agents operating in a workspace $\mathcal{W} \subset \mathbb{R}^2$ with unknown static structures $\mathcal{O}_{\text{st}} \subset \mathcal{W}$ (e.g., walls) and a set $\mathcal{O}_{\text{dy}} = \{\mathcal{O}_1, \mathcal{O}_2, ...\mathcal{O}_M\}$ of $M \in \mathbb{Z}_+$ dynamic obstacles (e.g., pedestrians). An agent $\mathcal{R}_i \in \mathcal{R}$ has a position $\mathbf{x}_i(t) \in \mathbb{R}^2$, radius $r_i \in \mathbb{R}_+$, speed $v_i \in \mathbb{R}_+$, goal $\mathbf{g}_i \in \mathbb{R}^2$, and LiDAR range $r_s \in \mathbb{R}_+$. It maintains a map $\hat{\mathcal{M}}_i(t) \subseteq \mathcal{O}_{\text{st}}$ and constantly updates it based on newly discovered structures. At $t{=}0$, $\hat{\mathcal{M}}_i(0) = \emptyset$. Furthermore, each agent maintains an RRT* tree $\mathcal{T}_i(t)$, that is constantly morphed based on the latest information about static and dynamic obstacles. At $t{=}0$, $\mathcal{T}_i(0)$ is built using the initial (partial) knowledge about static structures. Finally, each agent also maintains a plan P_i, which is the path to $\mathbf{g}_i$. The path is constantly replanned using the updated tree $\mathcal{T}_i(t)$. In addition, each agent $\mathcal{R}_i$ also receives information $\mathcal{I}_{i'}(t)$ from the other agents $\mathcal{R}_{i'} \in \mathcal{R}, i' \neq i$, including their current positions $\mathbf{x}_{i'}(t)$ and the portions of the map $m_{i'} \subseteq \mathcal{O}_{\text{st}}$ that have been newly discovered by them.

Objective: All agents reach their respective goals successfully, avoiding all static and dynamic obstacles, while minimizing $\max_i T_i^{\text{goal}}$, where T_i^{goal} is the time taken by agent $\mathcal{R}_i$ to reach its goal $\mathbf{g}_i$.

B. Algorithm phases

The system runs a 4-phase loop every time frame ($\Delta t{=}0.05\,\text{s}$). These phases are described below.

1) Phase 1: LiDAR Scan: In order to map the environment, each agent $\mathcal{R}_i$ performs a 360° scan of the surroundings using 180 sensor rays of range r_s. This scan detects both static and dynamic obstacles within the sensing range but is unable to detect occluded items behind other obstacles. If it detects any new portions of static structures $m_i \subseteq \mathcal{O}_{\text{st}}$, then it updates its own map as $\hat{\mathcal{M}}_i(t) \leftarrow \hat{\mathcal{M}}_i(t) \cup m_i$.

2) Phase 2: Shared Perception: In this phase, each agent broadcasts information about newly discovered structures to a central coordinator, which in turn broadcasts it to all other agents. As such, if agent $\mathcal{R}_i$ receives information from any other agent $\mathcal{R}_{i'}$, then it updates its map as $\hat{\mathcal{M}}_i(t) \leftarrow \hat{\mathcal{M}}_i(t) \cup m_{i'}, \forall i'$. This process is repeated for all agents $\mathcal{R}_i \in \mathcal{R}$, until all maps are synchronized. On the other hand, in independent operation mode, each agent relies solely on the map generated by its own sensor for planning and does not receive information from other agents. The advantage of shared discovery is that each agent can make proactive planning decisions, even for regions that it has never scanned using its own sensors, thus utilizing the potential to save significant travel time. For example, if the goal is behind an unknown wall that has a door on its left, then due to the lack of information, the agent would travel in a straight line to the goal and not towards the door until realizing it later upon detecting the wall.

3) Phase 3: Distributed Tree Repair: Each agent performs its tree pruning and repair when a) its map is updated to reveal new static structures and/or b) it detects dynamic obstacles.

Algorithm 1: Eager

Input: Tree $\mathcal{T}$, new wall w.
EdgeValidityCheck($\mathcal{T}$, w);
$\mathcal{E}_{\text{bad}} \leftarrow \{e \in \mathcal{T} \mid e \text{ intersects } w\}$
for *each* $e \in \mathcal{E}_{bad}$ **do**
  Prune e; *tree fragments into subtrees*
end
$\mathcal{H} \leftarrow$ HotNodeSearch($\mathcal{T}$, LSR)
for $h \in \mathcal{H}$ **do**
  if *reconnection through* h *valid* **then**
    Reconnect,
    Rewiring cascade.
  end
end
return BuildPath($\mathcal{T}$).

• **Tree morphing for static structures**- To accommodate new static structures, we describe four strategies as follows.

1. *Full tree rebuild.* In this strategy, a new tree is built from scratch after discarding the previous tree (15,000-nodes). This guaranties an optimal RRT* tree, but costs $\sim$ 40 ms per rebuild and expends significant time in replanning. For example, on the campus scenario ($\sim$300 walls), this means $\sim$140 rebuilds per run, totaling $\sim$5.6 s of replanning overhead. Furthermore, this could be orders of magnitude slower on an edge device and does not scale with the scenario size. We use this as a baseline.
2. *Eager.* In this strategy, all edges E of the tree are scanned, then the edges that intersect with the newly mapped structure are permanently pruned, and finally, local reconnections are made via hot-nodes within the Local Search Radius (LSR) of radius 80 m. This preserves most of the tree structure and costs $\sim$5 ms per replanning. Algorithm 1 describes the Eager strategy.
3. *Lazy Eager (LE).* In this strategy, if the current path is blocked by the new structure, then Eager is triggered; otherwise, replanning is skipped. This prevents unnecessary replannings if the new structures dont obstruct the path, thus saving significant replanning time.
4. *Swift.* This strategy is similar to Lazy Eager except that only the edges near the current path are scanned, and pruned if invalid, followed by tree repair and path search.

Lazy Eager and Swift strategies are both computationally cheaper; however, they can leave invalid edges elsewhere in the tree. This could cause problems during other replanning incidents or when an agent is pushed towards invalid edges by another robot during King coordination, resulting in a full rebuild. In multi-agent mode with shared perception, such invalid edges can accumulate fast, as new structures are discovered by remote agents.

• **Tree morphing for dynamic obstacles**- This is unchanged from the original SMART-3D algorithm: compute obstacle hazard zones OHZs for each dynamic obstacle, prune invalid nodes and edges inside OHZs of all dynamic obstacles intersecting with the agent's LRZ, reconnect via hot-nodes to morph the tree, and finally find a new path to the goal.

TABLE I
SCENARIO SPECIFICATIONS.

Scenario	Size	Walls	RRT* iters	Agents
Building	32×32 m	41	5K	2–4
Office	32×32 m	42	5K	2–4
Warehouse	32×32 m	28	5K	2–4
Hospital	96×96 m	200	30K	4
Airport	160×160 m	250	50K	4
Campus	128×128 m	300	80K	4
University	192×192 m	400	160K	4

4) Phase 4: Narrow Corridor Coordination: This layer resolves deadlocks in narrow corridors without morphing the planning tree of any agent by operating at the control level.

King selection. The king is the highest-priority active agent. The priorities could be assigned based on different factors, such as task criticality, distance to goal, battery level, and health status. When the king reaches its goal, the next highest priority agent becomes the king. The same priority ties are handled by random assignment.

King behavior. The king follows its path through the narrow corridor, ignoring all other agents whose paths also pass through the corridor. If a collision situation arises, the king pushes all agents backward, which is called a chain push. In this case, if the chain of agents approaches a wall, the first blocker is scattered sideways. After a push, SMART replans.

Non-king behavior. If a non-king agent arrives within 5 m of the king, it flees away (to the left, right, or in the first non-wall direction). If it is beyond 5 m of the king, it follows its own path, treating other agents as dynamic obstacles.

Overall, there are three decoupled layers: (1) RRT* tree- global path via pure pursuit. (2) SMART repair- updates the path when walls/OHZs invalidate edges, and (3) King/flee/push- direct position adaptation.

III. RESULTS

This section presents the comparative evaluation results of PECMAN through simulations and experiments.

A. Validation by Simulations

We evaluate the performance on seven procedurally generated 2D floorplans. Table I provides the scenario specifications. All scenarios run on an i9-14900K processor with 1,000 trials per scenario per strategy.

Fig. 1 shows snapshots of the multi-agent simulations on seven scenarios: Building, Office, Warehouse, Hospital, Airport, Campus and University. The simulations are conducted in cross mode, where the agents start at the corners and navigate to the opposite corners. The pedestrians follow a random motion at $0.5 - 2.0$ m/s.

1) Lazy Eager vs. Swift strategies: We compare two repair strategies: Lazy Eager and Swift. As discussed earlier, Lazy Eager scans all edges, but it does that only when the current path is blocked. On the other hand, Swift only scans the edges that are within 10 m of the current path. Although both of these methods can leave invalid edges on the map, Lazy Eager has the opportunity to clean them when the path is

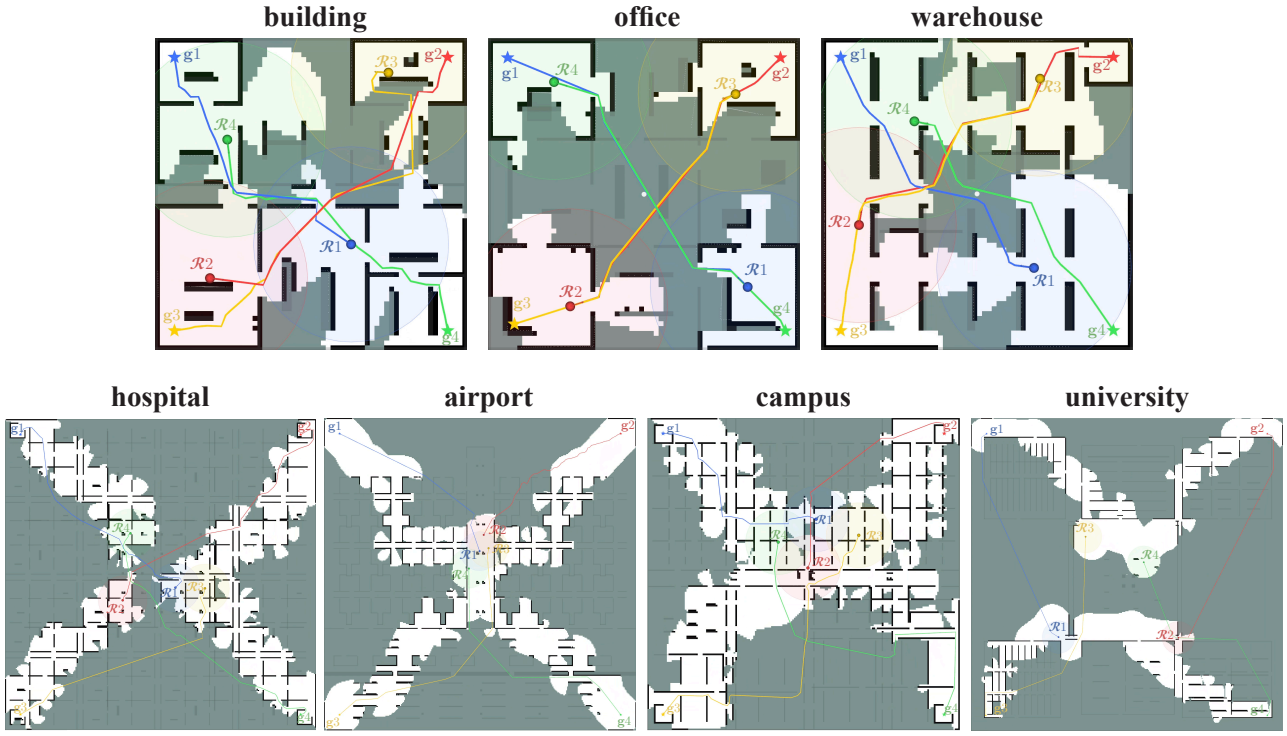


Fig. 1. 2D scenarios ranging from 32 m×32 m (building, office, warehouse) to 192 m×192 m (university). Walls (black) and Unexplored (grey).

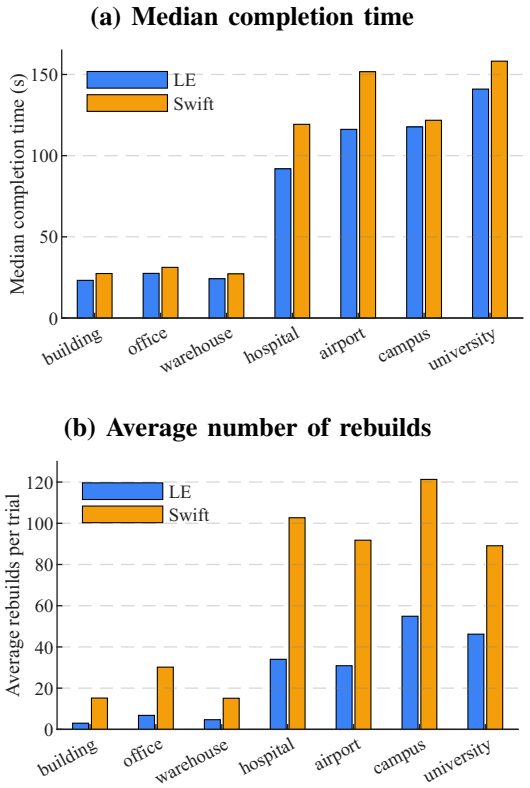


Fig. 2. Comparison of Lazy Eager vs. Swift strategies for 4-agent simulations with 1,000 trials per scenario.

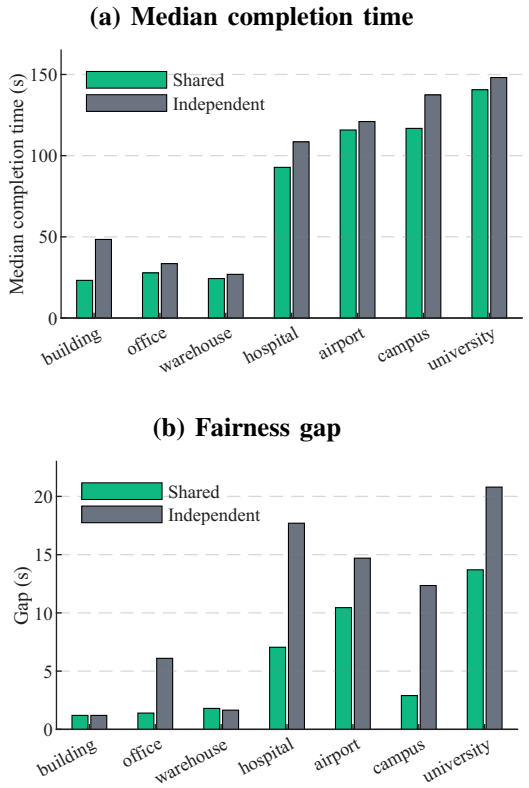


Fig. 3. Comparison of Shared perception vs Independent modes for 4-agent simulations with 1,000 trials per scenario.

blocked; thus, it triggers a lower number of full tree rebuilds. Fig. 2 visualizes the median completion times and the average number of rebuilds over all trials for each scenario. It is observed that Swift triggers ∼2–5× more full rebuilds due to invalid edges. When King pushes an agent into an unexplored region, it encounters invalid edges and must do a full rebuild.

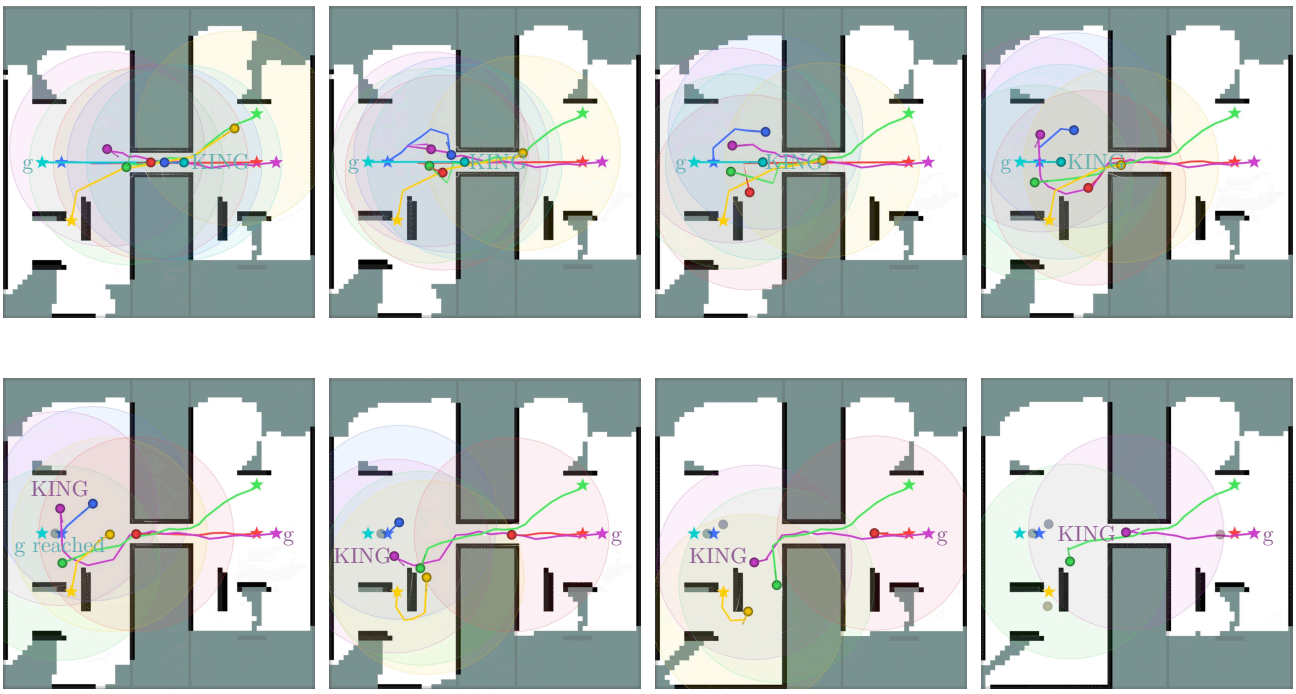


Fig. 4. Narrow corridor stress test: six agents approach a narrow corridor from both ends under Global King coordination. Eight time-stamped snapshots are shown from one trial. Panels 1–4: the light-blue agent is the active King; the other agents yield by fleeing, being pushed in a chain, or scattering sideways when blocked, allowing the King to pass through the corridor. Panels 5–8: after the previous King reaches its goal, the purple agent becomes the next King; the yellow and green agents yield while it passes.

This effect is clearer on large maps, e.g., hospital where the rebuild count jumps from 34.0 to 102.7 and campus where it jumps from 54.9 to 121.3. Lazy Eager delivered a $\sim$100% success rate, while the team finishes faster in all scenarios. The completion time gap widens on the larger maps, while the rebuild gap is already pronounced even on the small layouts.

2) Shared perception vs. Independent operation modes: Next, we compare the performance of the Shared perception and Independent operation modes while using the Lazy Eager repair strategy. In independent operation mode, each agent reacts to an unknown structure when its LiDAR sensor detects it; in contrast, in shared perception mode, a structure discovered by any agent is broadcast to all teammates, who then react accordingly by morphing their respective trees.

Fig. 3 visualizes the median completion times and the fairness gap over all trials for each scenario. The fairness gap is the difference between the slowest and fastest agent median completion times within each scenario. It is observed that the shared perception mode lowers the team completion time in all scenarios and also decreases the fairness gap in most cases.

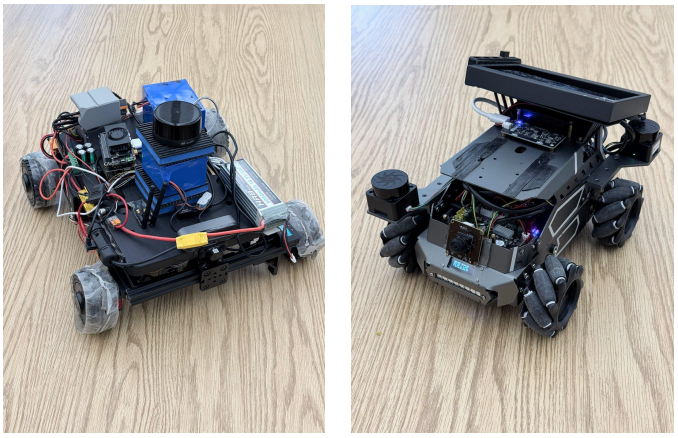
The success rate remains $\sim$100% for both shared perception and independent operation modes in all scenarios. On university, the shared perception mode records an average of 22.8 rebuilds per trial vs 11.3 for the independent operation mode, yet it completes slightly faster (140.6 s vs. 148.1 s). This is because shared perception allows proactive replanning rather than reactive replanning.

3) Narrow corridor coordination: The Global King layer resolves the failure mode that appears when two or more agents converge in a corridor narrower than twice the robot radius. In such situations, the reactive methods such as ORCA [37] deadlock because no feasible avoidance velocity exists. As a stress test for this layer, we constructed a 2 m wide and 8 m long corridor with agents entering from both ends, as shown in Fig. 4. The King sequencing (i.e., King passes first, others flee; then the next-priority agent becomes the king after the previous one exits) successfully sequenced all robots through the corridor in every trial. It is observed that the King layer handles the corridor case where reactive methods cannot. Fig. 4 shows a representative run where six agents are sequenced through the corridor under King coordination.

B. Testing by Real Experiments

We implemented PECMAN on a team of two heterogeneous robots using ROS2 platform as a final validation. Fig. 5 shows the two physical robots.

Hardware. Each robot carries a 2D LiDAR and a Raspberry Pi 5 (4-core, 2.4 GHz) running Cartographer SLAM, the pose publisher, and a low-level velocity controller. A central Alienware i9-14900K runs SMART-3D planner per robot in Rust. ROS2 Humble with CycloneDDS over WiFi connects the two sides. Running Cartographer SLAM on Pi 5 consumes $\sim$1.5 cores, while RRT* takes $\sim$300 ms on the Pi 5 versus $\sim$40 ms on the i9. Since running both onboard is not feasible,



(a) Robot 1 (b) Robot 2

Fig. 5. Two ROS2-integrated robots used for the experiments. Each robot runs Cartographer SLAM on its onboard Raspberry Pi 5 and streams observations to the central i9 planner.

the i9 hosts the planner while Pi 5 runs the local SLAM, path following controller, and emergency stopping.

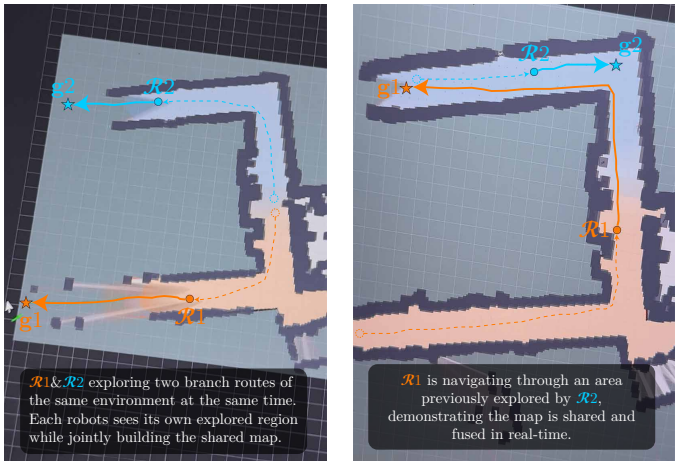
Map alignment. Each robot runs SLAM independently, and the i9 aligns the two local maps into a shared planning frame through an external rigid transform that is set from the known deployment geometry at startup. The i9 then transforms occupied cells from each robot's occupancy grid into this shared frame, which becomes the static-obstacle input to SMART-3D. Thus, map sharing is done at the planner level, and not by merging the underlying SLAM graphs.

Shared perception and execution. The newly observed static obstacles by one robot are broadcast to the other robot's planner, which updates its tree through incremental repair before it physically encounters that region. The static obstacles are wall portions extracted from the SLAM occupancy map and shared between robots. The dynamic obstacles are distinguished by map consistency. The live LiDAR scan is compared against the currently known static walls. Then, the LiDAR scan points that cannot be explained by the static map are treated as residual points, clustered together (min 8 points, 1.0 s persistence), and fed to the planner as dynamic obstacles. The position of the other robot is communicated directly from that robot and not inferred from LiDAR.

Fig. 6 shows a U-shaped corridor exploration in RViz. First, during the exploration stage, each robot explores one arm of the corridor, maps it, and shares it with the other robot. Then, in the tasking stage, each robot is assigned different goal points such that Robot 1 is assigned a goal in the arm that was scanned by Robot 2, while Robot 2 is assigned a goal in its own arm. It is seen that both Robots 1 and 2 successfully plan their paths to their respective goals and navigate to them, thus confirming the end-to-end success of the shared-perception.

IV. Conclusion

This paper extends SMART-3D, which is a single-robot real-time tree-repair planner, to perception-enabled collaborative multi-agent navigation planner (PECMAN). It is shown that PECMAN works in unknown environments through shared perception between agents. This enables agents to



(a) Exploration stage (b) Tasking stage

Fig. 6. Two-robot team tasking in a U-shaped corridor in a building. (a) Exploration phase- each robot maps its own arm of the corridor; the two partial maps are shared in the planning frame. (b) Tasking phase- Robot 1 (red) finds a path to a goal in the region explored by Robot 2, while Robot 2 (blue) navigates to a goal in its own region.

conduct proactive replanning in areas not observed by them, thus making more efficient plans and saving computation times. PECMAN was validated via extensive Monte-carlo simulations for seven 2D scenarios. It is seen that Shared perception achieves significant speed ups in the team-completion times, while maintaining near-100% success rates. Future work involves extending the PECMAN algorithm to consider curvature constrained robots [39] and environmental currents [40].

References

[1] R. K. Katzschmann, J. Del Preto, R. MacCurdy, and D. Rus, "Exploration of underwater life with an acoustically controlled soft robotic fish," *Science Robotics*, vol. 3, no. 16, 2018. [Online]. Available: https://doi.org/10.1126/scirobotics.aar3449

[2] J. Yuh, G. Marani, and D. R. Blidberg, "Applications of marine robotic vehicles," *Intelligent Service Robotics*, vol. 4, no. 4, pp. 221–231, 2011. [Online]. Available: https://doi.org/10.1007/s11370-011-0096-5

[3] Z. Shen, J. Song, K. Mittal, and S. Gupta, "CT-CPP: Coverage path planning for 3D terrain reconstruction using dynamic coverage trees," *IEEE Robot. Autom. Lett.*, vol. 7, no. 1, pp. 135–142, 2022. [Online]. Available: https://doi.org/10.1109/LRA.2021.3119870

[4] N. Palomeras, N. Hurtós, M. Carreras, and P. Ridao, "Autonomous mapping of underwater 3-D structures: From view planning to execution," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1965–1971, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8299494

[5] Z. Shen, J. P. Wilson, and S. Gupta, "C*: A coverage path planning algorithm for unknown environments using rapidly covering graphs," *IEEE Transactions on Robotics*, vol. 42, pp. 1233–1253, 2026. [Online]. Available: https://doi.org/10.1109/TRO.2026.3661719

[6] J. Song and S. Gupta, "ϵ^*: An online coverage path planning algorithm," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 526–533, 2018. [Online]. Available: https://doi.org/10.1109/TRO.2017.2780259

[7] T. Somers and G. A. Hollinger, "Human–robot planning and learning for marine data collection," *Autonomous Robots*, vol. 40, no. 7, p. 1123–1137, 2016. [Online]. Available: https://doi.org/10.1007/s10514-015-9502-8

[8] S. Moradi, A. Bokani, and J. Hassan, "Uav-based smart agriculture: a review of uav sensing and applications," in *2022 32nd International Telecommunication Networks and Applications Conference (ITNAC)*, 2022, pp. 181–184. [Online]. Available: https://doi.org/10.1109/ITNAC55475.2022.9998411

[9] G. G. R. d. Castro, G. S. Berger, A. Cantieri, M. Teixeira, J. Lima, A. I. Pereira, and M. F. Pinto, "Adaptive path planning for fusing rapidly exploring random trees and deep reinforcement learning in an agriculture dynamic environment uavs," *Agriculture*, vol. 13, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2077-0472/13/2/354

[10] T. Panigati, M. Zini, D. Striccoli, P. F. Giordano, D. Tonelli, M. P. Limongelli, and D. Zonta, "Drone-based bridge inspections: Current practices and future directions," *Automation in Construction*, vol. 173, p. 106101, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580525001414

[11] J. Song, K. Qiu, S. Gupta, and J. Hare, "Slam based adaptive navigation of auvs for oil spill cleaning," in *2014 Oceans - St. John's*, 2014, pp. 1–6. [Online]. Available: https://doi.org/10.1109/OCEANS.2014.7003028

[12] S. V. Kumar, R. Jayaparvathy, and B. Priyanka, "Efficient path planning of AUVs for container ship oil spill detection in coastal areas," *Ocean Engineering*, vol. 217, 2020. [Online]. Available: https://doi.org/10.1016/j.oceaneng.2020.107932

[13] J. Yang, J. P. Wilson, and S. Gupta, "Dare: Diver action recognition encoder for underwater human–robot interaction," *IEEE Access*, vol. 11, pp. 76 926–76 940, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3298304

[14] K. Shojaei and M. Dolatshahi, "Line-of-sight target tracking control of underactuated autonomous underwater vehicles," *Ocean Engineering*, vol. 133, pp. 244–252, 2017. [Online]. Available: https://doi.org/10.1016/j.oceaneng.2017.02.007

[15] J. Z. Hare, S. Gupta, and T. A. Wettergren, "Pose: Prediction-based opportunistic sensing for energy efficiency in sensor networks using distributed supervisors," *IEEE Transactions on Cybernetics*, vol. 48, no. 7, pp. 2114–2127, 2018. [Online]. Available: https://doi.org/10.1109/TCYB.2017.2727981

[16] J. Z. Hare, J. Song, S. Gupta, and T. A. Wettergren, "POSE.R: Prediction-based opportunistic sensing for resilient and efficient sensor networks," *ACM Transactions on Sensor Networks*, vol. 17, no. 1, 2020. [Online]. Available: https://doi.org/10.1145/3419755

[17] E. U. Acar, H. Choset, Y. Zhang, and M. Schervish, "Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 441–466, 2003. [Online]. Available: https://doi.org/10.1177/02783649030227002

[18] K. Mukherjee, S. Gupta, A. Ray, and S. Phoha, "Symbolic analysis of sonar data for underwater target detection," *IEEE J. Oceanic Eng.*, vol. 36, no. 2, pp. 219–230, 2011. [Online]. Available: https://doi.org/10.1109/JOE.2011.2122590

[19] Z. Shen, J. P. Wilson, and S. Gupta, "$\epsilon^\star+$: An online coverage path planning algorithm for energy-constrained autonomous vehicles," in *Global Oceans 2020: Singapore – U.S. Gulf Coast*, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1109/IEEECONF38699.2020.9389353

[20] F. O. Coelho, M. F. Pinto, I. Z. Biundini, G. G. R. Castro, F. A. A. Andrade, and A. L. M. Marcato, "Autonomous uav exploration and mapping in uncharted terrain through boundary-driven strategy," *IEEE Access*, vol. 12, pp. 92 464–92 483, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3422834

[21] K. Mukherjee, S. Gupta, A. Ray, and T. A. Wettergren, "Statistical-mechanics-inspired optimization of sensor field configuration for detection of mobile targets," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 3, pp. 783–791, 2011. [Online]. Available: https://doi.org/10.1109/TSMCB.2010.2092763

[22] J. P. Wilson, S. Gupta, and T. A. Wettergren, "Generalized multispeed dubins motion model," *IEEE Transactions on Robotics*, vol. 41, pp. 2861–2878, 2025. [Online]. Available: https://doi.org/10.1109/TRO.2025.3554436

[23] Z. Shen, J. P. Wilson, S. Gupta, and R. Harvey, "SMART: Self-morphing adaptive replanning tree," *IEEE Robot. Autom. Lett.*, vol. 8, no. 11, pp. 7312–7319, 2023. [Online]. Available: https://doi.org/10.1109/LRA.2023.3315210

[24] P. Agrawal, S. Gupta, and Z. Shen, "SMART-3D: Three-dimensional self-morphing adaptive replanning tree," 2025. [Online]. Available: https://arxiv.org/abs/2509.16812

[25] J. Song and S. Gupta, "CARE: Cooperative autonomy for resilience and efficiency of robot teams for complete coverage of unknown environments under robot failures," *Auton. Robots*, vol. 44, pp. 647–671, 2020.

[26] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015. [Online]. Available: https://doi.org/10.1016/j.artint.2014.11.006

[27] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artificial Intelligence*, vol. 310, p. 103752, 2022. [Online]. Available: https://doi.org/10.1016/j.artint.2022.103752

[28] K. Okumura, "LaCAM: Search-based algorithm for quick multi-agent pathfinding," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 37, no. 10, 2023, pp. 11 655–11 662. [Online]. Available: https://arxiv.org/abs/2211.13432

[29] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and robust execution of MAPF schedules in warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019. [Online]. Available: https://doi.org/10.1109/LRA.2019.2894217

[30] J. Li, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "Symmetry-breaking constraints for grid-based multi-agent path finding," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6087–6095. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4565

[31] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 272–11 281. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17344

[32] M. Čáp, P. Novák, J. Vokřínek, and M. Pěchouček, "Multi-agent RRT*: Sampling-based cooperative pathfinding (extended abstract)," in *Proc. International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2013, pp. 1263–1264. [Online]. Available: https://dl.acm.org/doi/10.5555/2484920.2485174

[33] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," *International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016. [Online]. Available: https://doi.org/10.1177/0278364915615688

[34] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *IEEE Int. Conf. Robot. Automat.*, 2006, pp. 1243–1248. [Online]. Available: https://doi.org/10.1109/ROBOT.2006.1641879

[35] M. Otte and E. Frazzoli, "RRT$^{\text{X}}$: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *Int. J. Robot. Res.*, vol. 35, no. 7, pp. 797–822, 2016. [Online]. Available: https://doi.org/10.1177/0278364915594679

[36] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *IEEE Int. Conf. Intell. Robots Syst.*, vol. 3, 2002, pp. 2383–2388. [Online]. Available: https://doi.org/10.1109/IRDS.2002.1041624

[37] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*. Springer, 2011, pp. 3–19.

[38] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997. [Online]. Available: https://doi.org/10.1109/100.580977

[39] J. Song, S. Gupta, and T. A. Wettergren, "T*: Time-optimal risk-aware motion planning for curvature-constrained vehicles," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 33–40, 2019.

[40] K. Mittal, J. Song, S. Gupta, and T. A. Wettergren, "Rapid path planning for Dubins vehicles under environmental currents," *Robot. Auton. Syst.*, vol. 134, p. 103646, 2020. [Online]. Available: https://doi.org/https://doi.org/10.1016/j.robot.2020.103646