

Modular, Resilient, and Scalable System Design Approaches - Lessons learned in the years after DARPA Subterranean Challenge

Prasanna Sriganesh, James Maier, Adam Johnson, Burhanuddin Shirose, Rohan Chandrasekar, Charles Noren, Joshua Spisak, Ryan Darnley, Bhaskar Vundurthy and Matthew Travers

Abstract—Field robotics applications, such as search and rescue, involve robots operating in large, unknown areas. These environments present unique challenges that compound the difficulties faced by a robot operator. The use of multi-robot teams, assisted by carefully designed autonomy, help reduce operator workload and allow the operator to effectively coordinate robot capabilities. In this work, we present a system architecture designed to optimize both robot autonomy and the operator experience in multi-robot scenarios. Drawing on lessons learned from our team’s participation in the DARPA SubT Challenge, our architecture emphasizes modularity and interoperability. We empower the operator by allowing for adjustable levels of autonomy (“sliding mode autonomy”). We enhance the operator experience by using intuitive, adaptive interfaces that suggest context-aware actions to simplify control. Finally, we describe how the proposed architecture enables streamlined development of new capabilities for effective deployment of robot autonomy in the field.

Index Terms—Field Robotics, Robot Autonomy, System Design, Multi-Robot Exploration

I. INTRODUCTION

Field robotics encompasses a wide range of applications in which robots must operate in large, unknown and often harsh environments. One such application is a search-and-rescue mission, where an operator deploys a robot to map and explore hazardous environments. These missions have traditionally relied on teleoperation, for instance, robots were deployed at the Fukushima Daiichi nuclear reactor disaster [1] or the snake robots used in the 2017 Mexico earthquakes [2]. However, direct teleoperation places a high cognitive burden on the operator, especially in complex environments with limited communications [3]. Robot autonomy becomes a crucial tool in these scenarios, allowing the operator to focus on high-level mission goals.

Robot autonomy has received promising improvements to the performance driven by recent advances in robot perception, localization, planning, and artificial intelligence (AI) systems. The 2021 DARPA Subterranean (SubT) Challenge [4] provided a compelling showcase for these advancements. SubT involved a single human operator deploying a team of heterogeneous robots for coordinated mapping and exploration in unknown subterranean scenarios. While the challenge highlighted the potential of robot autonomy, it exposed the teams to problems of real-world deployment. The analysis of DARPA SubT Challenge results [4] identified several key challenges, particularly: operator cognitive overload

from managing multiple robots, robot attrition, and ensuring interoperability within a heterogeneous team. In order to address the identified challenges, we posit that developing a holistic systems-level design (e.g., from individual robots to the operator experience/UI) is a crucial step for the effective deployment of autonomous systems in the field

We present a system architecture designed to optimize both robotic autonomy and the operator experience within the context of multi-robot deployments. We draw from our group’s experience as part of Team Explorer in SubT and outline the lessons learned from the field. These lessons drive core elements of our design philosophy for the presented architecture, namely: modularity and adaptive autonomy. Designing a system with a focus on modularity enables the rapid configuration and deployment of heterogeneous robotic systems. Coupling this design focus with tools for adaptive autonomy, which includes both operator-adjustable levels of autonomy (for example, switching between assisted teleoperation and waypoint navigation) and behavior-tree-based intuitive user-interfaces, yields a system where operator workload does not scale as poorly with additional robots. Finally, we outline how different components of the architecture work in tandem to enable easier development of future multi-robot capabilities without the need for extensive reconfiguration.

II. RELATED WORK

In this section, we first describe relevant work and insights from field robotics research, with a focus on search-and-rescue and subterranean environments. We next present the different approaches used by the participating teams in the DARPA SubT challenge that highlight lessons learned in the deployment of multi-robot autonomy.

Autonomous exploration in subterranean environments has been a significant focus in field robotics. These include works involving mapping of mine environments [5] [6], disaster response [1] [2] [7] and planetary exploration [8]. Recent advances have focused on improving specific aspects of subterranean robotics such as perception, localization and navigation. Robots operating in these environments place constraints on compute available, and demand development of lightweight architecture for object detection [9] [10]. The degraded nature of the environment have led to development of localization algorithms that can fuse information from multiple sources [11] [12]. Novel algorithms have been developed for long-range exploration and planning for such environments [13] [14].

All the authors are affiliated with Carnegie Mellon University. Prasanna Sriganesh (pkettava@andrew.cmu.edu) is the corresponding author.

One of the biggest challenges presented by search-and-rescue scenarios [15] [16] is the low-bandwidth and limited communication range. Robots are required to build their own networks, and exchange data only when necessary. This has led to algorithms to enable multi-robot exploration with limited communications [17]. There have also been solutions to ensure that the robots maintain persistent communications by monitoring the network and dropping repeater nodes when necessary [18]. The DARPA Subterranean challenge brings all these various advancements together and exposes the challenges present in deployment.

In the analysis of the 2021 DARPA SubT Final Event by Dr. Timothy Chung [4], the authors highlight the inevitability of robot attrition in real-world deployments. They emphasize on the importance of streamlined system integration to enable fast team reconfiguration. The authors also state that robot heterogeneity was crucial to team performance, and was partially enabled by the increasing availability of various mobile robot platforms. Finally, they underscore the importance of the human-robot interface as “the human teammate is often the critical performance limiter”. While a comprehensive survey of DARPA SubT technical approaches warrants a separate work, here we focus on key details from Team Explorer and other relevant teams that address these critical system-level insights.

Team Explorer in the DARPA SubT Challenge: Team Explorer’s approach to SubT [19] comprised of heterogeneous robot team controlled by a single operator. They emphasised the “sliding mode autonomy framework” [20] [21], that allowed the human operator to adjust the level of autonomy for different tasks. This framework enabled different robot operational modes with varying levels of autonomy (e.g., exploration, go-to-waypoint, manual etc). A behavior tree [22] was used to arbitrate different behaviors on the robot.

Team Explorer mitigated issues with high operator cognitive load by using a GUI which was highly tailored to the format of the SubT competition, while also employing human aids to the single operator who would remind the operator to complete certain tasks, thus letting the operator focus solely on operating robots. Additionally, robot capabilities were defined through module-specific configuration files, enabling heterogeneity in the fleet. However, this approach limited on-the-fly reconfiguration, highlighting the need for enhanced interoperability in future systems.

Other teams in the DARPA SubT Challenge: The work described by other DARPA Subterranean Challenge performer teams provide great insights to address challenges in human-robot interfacing, and overall system resiliency. Team CoSTAR [23], describe a copilot assistant that manages the interface between the human operator and the autonomous system. This copilot assistant can create and schedule tasks to both the human as well as the robots by treating the operator and the robots as a single team and helps reduce the cognitive load on the operator.

Other approaches to human-robot collaboration include the use of finite state automata for decision-making, as seen in

Team CSIRO Data61’s work [24]. They employed a market-based task allocation approach to coordinate the actions of different robots in their team. The authors note that tuning the rewards for task allocation was a challenge, and needed to be tuned for various operational environments. Additionally, they also described the need for incorporating manual tasks to incorporate operator guidance into this framework. This motivates the need for adjustable levels of autonomy, to avoid conflicts between the operator’s intended actions and robot’s decision making.

The CTU-CRAS-NORLAB team [25] [26] acknowledge that component-level failures are inevitable, and is a result of working with cutting-edge hardware which is operating at its limits. In order to address this, they design their stack to continually log system state and automatically restart components from the last known safe state when a failure is detected. This inspires us to carefully craft information flow in the system to ensure restarts do not cause conflicting commands to the robot.

III. LESSONS LEARNED

Our experiences, combined with the analysis of prior work described in the previous section, have highlighted five crucial lessons learned for the effective deployment of robot autonomy in the field. They are described below.

- (L1) **Operator-Adjustable Autonomy:** Ensuring the operator has the ability to adjust the level of robot autonomy is crucial for flexible and adaptable operation
- (L2) **System Interoperability:** Seamless interoperability between heterogeneous robots is essential. This allows for dynamic robot deployment depending on environmental factors
- (L3) **Centralized Control Flow:** Careful management of system-wide control flow is necessary to prevent conflicting commands and ensure coordinated robot behavior
- (L4) **Adaptive User Interface:** An interface that dynamically adjusts to present only valid actions based on context can significantly reduce the operator’s cognitive workload
- (L5) **Extensible System Design:** Prioritize modularity and flexibility during system integration to accommodate future changes and the addition of new capabilities

In the next section, we described our developed architecture that is designed with these learned lessons in mind

IV. PROPOSED ARCHITECTURE

This section describes the proposed architecture for a multi-robot system. Figure 1 provides a visual overview. The architecture is designed to enable efficient coordination and control of a multi-robot system and consists of several modules serving different purposes. The architecture supports multiple operators (basestations) and multiple heterogeneous robots. We use Robot Operating System (ROS) [27] for implementation and DDS for communication between the operator’s basestation and the robots. This allows us to

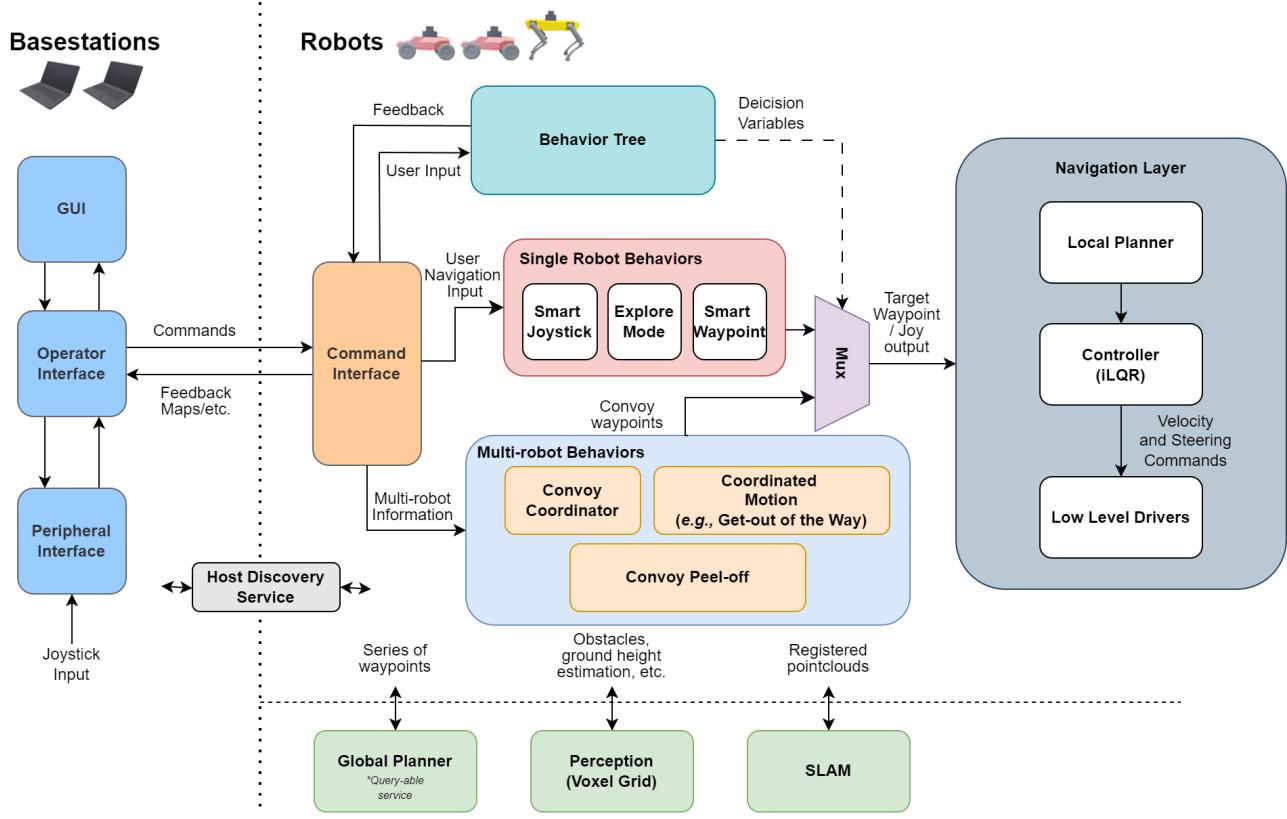


Fig. 1: A overview of the proposed system architecture

control information exchange and monitor communication bandwidth.

The architecture supports dynamically adding or removing robots in real-time. This is made possible by a host discovery service, which shares information about active agents (robots and basestations) on the network, and their capabilities (e.g., legged or wheeled robot). This is discussed further in section IV-A.

We embrace the concept of sliding mode autonomy [20] [21] to ensure the operator can adjust the level of autonomy needed for a task while maintaining high-level control. This aligns with (L1) and as a result, creates different operational modes (similar to Team Explorer’s implementation [19]) with increasing degrees of autonomy:

- **Full Manual Mode:** Operator commands directly control the robot
- **Smart Joystick Mode:** User-commanded direction is used in conjunction with a planner to avoid obstacles and navigate through narrow spaces
- **Waypoint Mode:** The robot navigates to a target location provided by the operator
- **Exploration Mode:** The robot autonomously explores an area and operates with the highest level of autonomy

Translating the user inputs to navigation commands is handled in the single robot behavior block, while the behavior tree ensures that the robot can be set to a specific mode. This is discussed further in section IV-C.

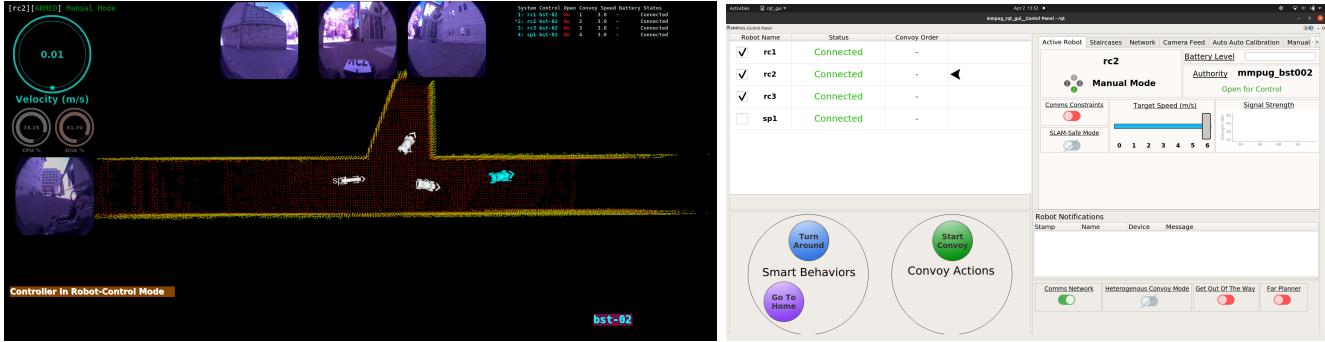
In addition to these modes, our architecture enables easy

addition and development of new modes for different co-ordinated behaviors [28]. One such example is the convoy mode where multiple robots form a convoy formation. The operator controls the lead robot using the modes mentioned above while the other robots follow-the-leader while staying in formation, extending the sliding mode autonomy across multiple robots. This is further discussed in section IV-F.

A. Host discovery service

To enable effective interoperability, our architecture requires knowledge about connected agents and their capabilities. We implement a host discovery service on each agent (robots and basestations) that provide this information. Each agent maintains a local server that stores identification information and its specific capabilities. Clients can query this server to obtain necessary information, allowing the selective launch of platform-specific modules. For example, a different controller and planner can automatically be launched in case of a quadruped.

Obtaining knowledge from other agents follows a “request-response” communication model. The requesting agent queries the server of the responding agent, who replies with its identification and capabilities. To support discovery of an arbitrarily large number of agents, requests are sent in a multicast fashion. Responding agents package the requested information and reply directly (unicast) to the requesting agent. This host discovery service optimizes communication on bandwidth-constrained networks while enabling discovery and transfer of robot-specific parameters.



(a) RViz visualization providing the operator with real-time status: map, robot locations, camera feeds of the active robot, its operational mode, and summary data for other robots on the top right

(b) Touch-enabled GUI example: three robots selected (top left), active robot information (top right), valid actions (bottom left), and notifications/feature toggles (bottom right)

Fig. 2: Operator Interface: Example of RViz window, and the touch-enabled GUI

A published list of robots and basestations are provided over a ROS topic, ensuring system-wide awareness of active agents. We mandate that all applicable nodes must dynamically update their data streams based on this list. This host discovery process enables the seamless deployment of new robots and allows for better robot interoperability (**L2**), a significant improvement over our previous work, which lacked this flexibility.

B. Command Interface

The command interface module serves as the component enabling multi-operator, multi-robot interaction. Its primary function is to receive all messages transmitted over the communications channel and determine whether they should be rejected or distributed to other modules within a robot’s system. It also publishes robot telemetry data back to the basestation. The data sent to the basestation includes the robot state, the behavior tree state, downsampled map and camera streams.

Upon the host discovery service’s updates, a list of the available devices is published throughout the network of robots and operators. The command interface monitors this list to track all available operators in the system. To ensure single-operator control at a time, the module keeps track of the current authorized basestation and grants the robot’s control to any basestation upon request. Once authorized, only messages originating from the authorized basestation are forwarded to the rest of the stack. Messages from other basestations are discarded.

The command interface also serves as the arbiter of control authority in the presence of multiple basestations. During operations, any basestation may place a request for control over a particular robot. Before the authority is assigned to a requesting basestation, the request is evaluated by the command interface to ensure that no other basestation currently holds a “lock” on the current robot. This “lock” designates that the vehicle is currently under the control authority of a different basestation and can only be overwritten when the basestation that is current authorized to command the robot releases the active lock.

The command interface ensures that there is only a single

active interface point from different operators to a robot. This ensures that no conflicting commands are sent to the robot from different operators, addressing our lessons learned from (**L3**). The use of the host discovery service alongside the command interface allows for clear interoperability with multiple operators (**L2**).

C. Behavior Tree and Mux

Behavior trees are central to our approach, managing complex system-mode transitions, enhancing modularity, and ensuring a clear structure [29]. Within our proposed architecture, the behavior tree acts as the central authority, determining the active mode on the robot. The behavior tree receives requests from the command interface, assesses their viability, switches the robot to the requested mode and enables that corresponding channel on the mux.

Each robot runs its own behavior tree, validating requests against predefined conditions. These conditions can be based on various things such as hardware availability, environmental constraints, or subsystem health or statuses. For instance, if the robot is not receiving joystick commands or if SLAM (Simultaneous Localization and Mapping) is not initialized, a ‘smart joystick mode’ request will be discarded, as the ‘smart joystick mode’ requires SLAM to function correctly. This structure of the behavior tree makes it easier and faster to define and implement new modes, along with their activation conditions (**L5**).

Once the behavior tree determines the operational mode, it communicates this decision to the mux. The mux filters the outputs from different subsystems given the current robot mode specified by the behavior tree. This ensures that the navigation layer does not receive conflicting targets, promoting a centralized control flow in the system (**L3**). It also facilitates seamless integration of different subsystems (**L5**).

The current behavior tree state (operational mode, authorized basestation, convoy order etc.) is sent back to the basestation as feedback. This helps in creating an adaptive interface that presents valid options to the user based on the operator’s selection and the feedback from robot behavior trees. (**L4**)

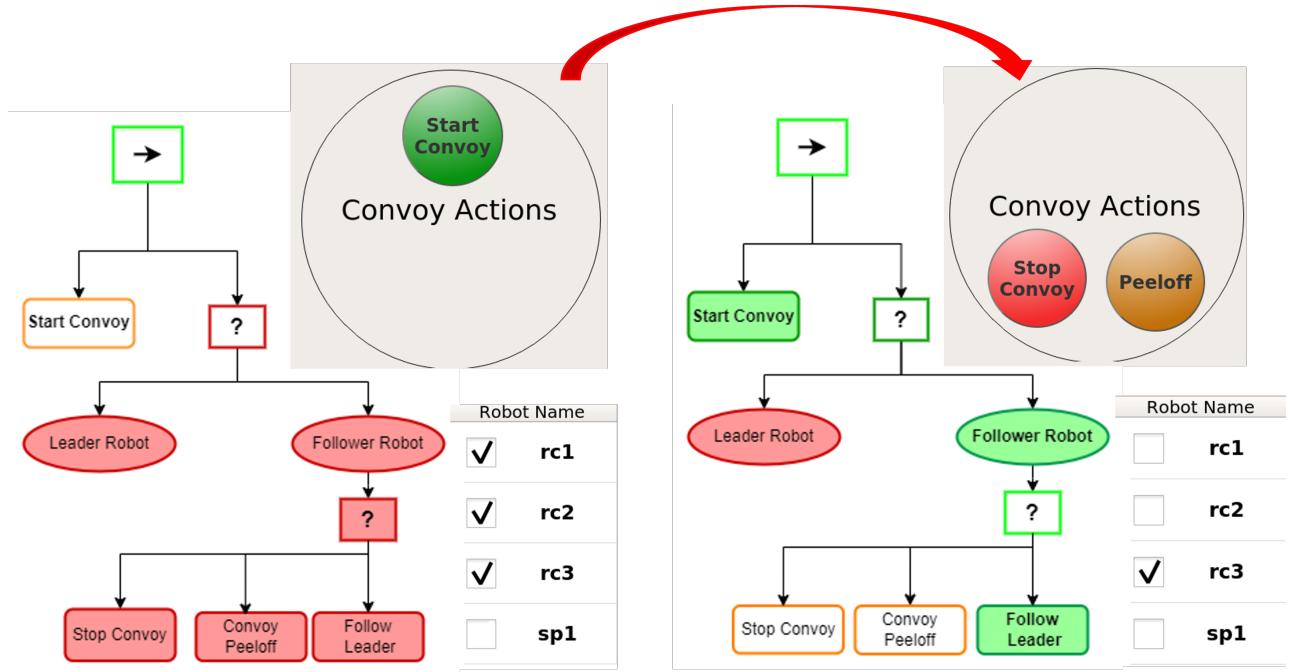


Fig. 3: The left image shows the behavior subtree for the convoy with multiple robots selected, enabling the ‘start convoy’ action. The right image demonstrates how the GUI dynamically updates based on the changes in the behavior subtree when a follower robot in an active convoy is selected, presenting actions like ‘peeloff’ and ‘stop convoy’.

D. Operator Interface and GUI

The operator interface acts as the central hub for communication and control, providing the operator with real-time feedback from the connected robots. It features two screens:

- **RViz Window:** Displays a 3D map, camera streams, robot modes, velocities, and overall system status (see Fig. 2a). This provides crucial situational awareness.
- **Custom GUI (Touch-Enabled):** Delivers subsystem feedback and enables rapid command actions through its touch-optimized design. (see Fig. 2b)

Figure 4 shows the representation of the operator setup to control the robots.

Commands can be issued through either a physical human

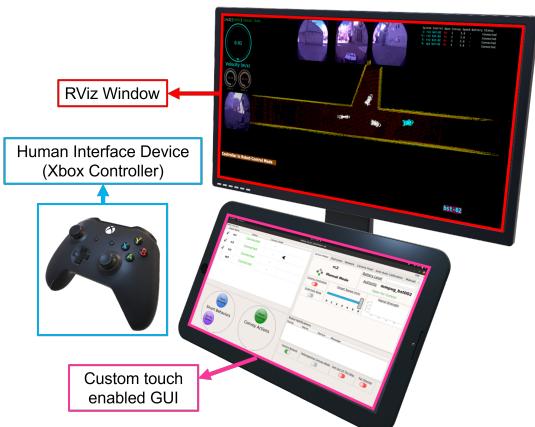


Fig. 4: A pictorial representation of the operator setup showing the two screens

interface device (e.g., xbox controller) or the touch-enabled GUI on the basestation. The GUI dynamically updates based on the selected robots and their state feedback from the behavior tree. It presents only valid actions in real-time, guiding the operator’s choices and minimizing errors. For example, the ‘start convoy’ action becomes available only when multiple robots are selected, and actions such as ‘stop convoy’ and ‘peeloff’ are displayed when a follower robot in an active convoy is selected (Fig. 3).

The GUI also displays information about the active robot such as the mode of operation, the operator in control, and communication signal strength. The GUI features a notification panel for prioritized messages from the robot, such as control requests from other operators or subsystem failures. Additionally, the GUI has tabs for specialized information like staircase detection, network monitoring etc. These tabs are intentionally categorized and hidden, allowing the operator to access relevant details without being overwhelmed.

Our architecture acknowledges the reality of component-level failures. A dedicated tab displays the health status of individual nodes. This empowers the operator to choose between automatic or manual restarts. This custom GUI directly addresses the fourth lesson learned: the importance of adaptive user interfaces (**L4**).

E. Navigation Layer

The navigation layer serves as the component responsible for translating high-level operator commands into actionable directives for the robot’s hardware. The navigation layer is typically made up of a local planner that compute a path towards the goal (e.g., trajectory library [30]), a controller to

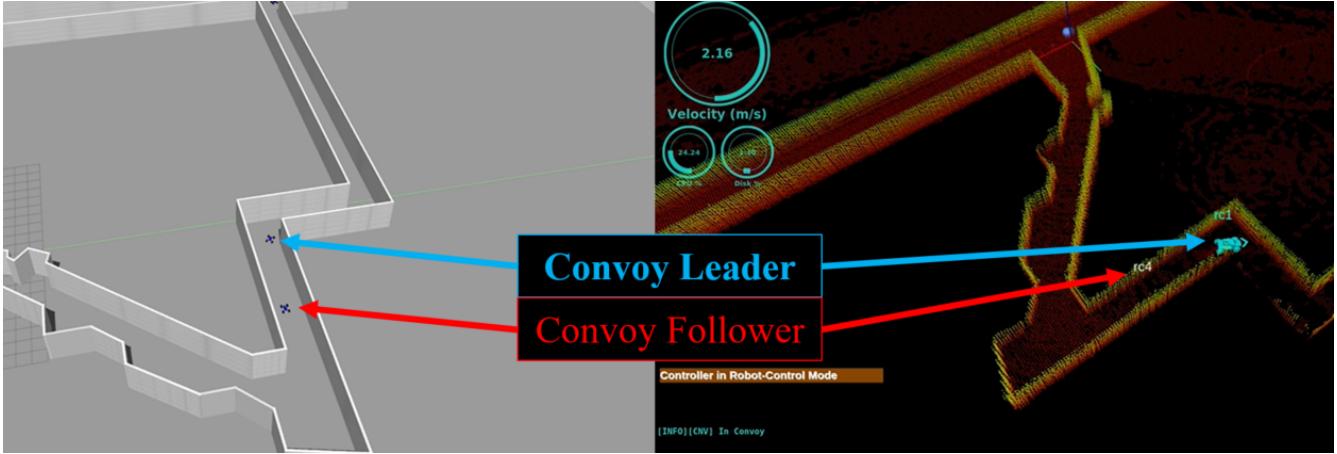


Fig. 5: Simulation of two robots in convoy formation

execute this plan (e.g., iLQR controller [31]), and the low-level controllers (drivers) that translate these control actions to the actuators. By keeping these components separate, the architecture allows for flexibility and easier testing. One can swap out the planner or controller for different robots or experiment with new navigation algorithms without overhauling the entire system (L5).

A single interface point to the navigation layer from the mux prevents unintended commands. This structure ensures that conflicting directives from the autonomy stack cannot be received by the navigation layer. The mux, controlled by the behavior tree, determines the true input source, promoting a centralized control flow and enhancing safety (L4). Figure 6 shows how trajectory libraries are used to select a path for the robot controller to track.

F. Advanced Multi-agent Coordination

The proposed architecture simplifies the development and implementation of coordinated multi-agent behaviors, contributing to reduced operator workload. The use of behavior trees and the mux as a single interface point provides a streamlined framework for implementing features, includ-

ing multi-robot convoying and get-out-of-the-way maneuvers [28], [32], [33].

In the convoy scenario, the operator directly controls the lead robot in either the smart joystick or waypoint mode. The follower robots autonomously maintain formation, tracking the lead robot's movement. This is achieved through the dedicated ‘convoy coordinator’ block in the architecture (1). This block handles information about all the robots in convoy and computes the target waypoints for the follower robots. These waypoints are then sent to the mux, and the behavior tree directs the mux to forward inputs from this block to the navigation layer. Figure 5 shows a simulated convoy with 2 robots.

Developing new capabilities is simplified with this structure. For example, developing a ‘get-out-of-the-way’ behavior involves two main steps. The first step is to implement the ‘get-out’ algorithm as an independent module and connecting its output to the mux. The next step is to add the triggering condition of this maneuver in the behavior tree. This ties back to our fifth learned lesson: (L5), which promotes faster development of future capabilities. Both convoy mode and get-out-of-the-way behaviors enable one-to-many control and automatic coordination, which significantly reduce the operator’s burden when managing multiple robots. (L4)

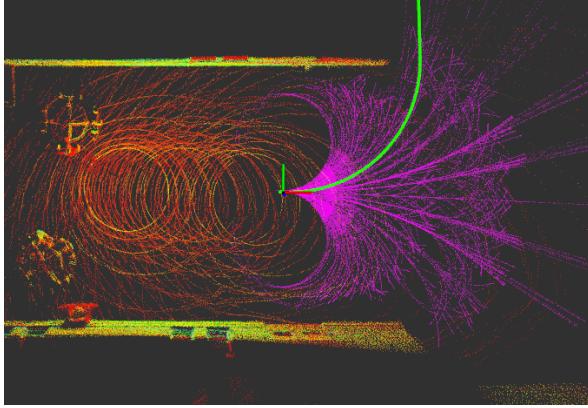


Fig. 6: Robot employing a trajectory library based local planner. All feasible paths are shown in magenta, and selected path is shown in green

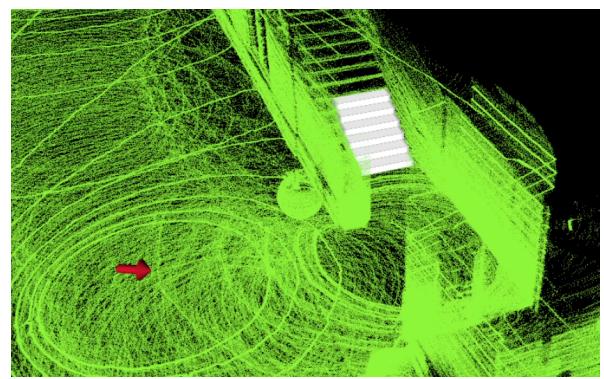


Fig. 7: Detected staircases depicted using white markers in the 3D map

G. System-wide services

While our proposed architecture has many definite data channels for passing information between modules, certain components are declared as a system-wide service from which any robot can utilize their output. These services fall into one of two categories: base-level knowledge or queryable services. In our architecture, base-level knowledge incorporates both SLAM and perception, whose outputs are considered requisite for autonomy. For example, a staircase detection algorithm [34] runs on all robots to automatically inform the operator about staircases in the environment. Figure 7 shows detected staircases as markers in the RViz map. A query-able service, however, is complex function or behavior, such as a global planner, which may be required by multiple different modules of the system. In both cases, restricting access to the data channel either requires inefficient passing of information through multiple modules, or a module accepting the entirety of the output while it is not needed. Instead, we embrace direct access to these specific system-wide services in which any module can obtain their outputs or call their functions as necessary. This design supports modularity, allowing independent implementation of algorithms for different platforms that adhere to a standardized data format (L5).

V. CONCLUSION

In this paper, we present a system architecture for multi-robot operations. Drawing on lessons learned from our experience in the DARPA SubT challenge, and real-world deployments of multi-robot teams, our approach prioritizes operator control, interoperability, control flow management, adaptive interfaces, and modular design for rapid development. We outline how the key components, such as behavior trees, the mux, and the command interface, work in concert to provide flexible autonomy levels, streamline coordination of heterogeneous robots, and reduce the operator's cognitive workload.

The proposed architecture, with its emphasis on usability and adaptability, holds significant promise for advancing effective robot deployment in challenging, unstructured environments. By carefully considering the human-robot interaction and the practical realities of system failures, our work contributes to the development of more capable and reliable autonomy solutions. In the future, we plan to present the field reports from deploying and testing this architecture on real heterogeneous multi-robot systems. Future investigations will also explore scalability to larger teams and further refinement of adaptive interface elements.

REFERENCES

- [1] Y. Yokokohji, “The use of robots to respond to nuclear accidents: Applying the lessons of the past to the fukushima daiichi nuclear power station,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 681–710, 2021.
- [2] J. Whitman, N. Zevallos, M. Travers, and H. Choset, “Snake robot urban search after the 2017 mexico city earthquake,” in *2018 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, IEEE, 2018, pp. 1–6.
- [3] J. Y. Chen, E. C. Haas, and M. J. Barnes, “Human performance issues and user interface design for tele-operated robots,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1231–1245, 2007.
- [4] T. H. Chung, V. Orehov, and A. Maio, “Into the robotic depths: Analysis and insights from the DARPA subterranean challenge,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 6, pp. 477–502, 2023.
- [5] C. Baker, A. Morris, D. Ferguson, et al., “A campaign in autonomous mine mapping,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, IEEE, vol. 2, 2004.
- [6] S. Thrun, S. Thayer, W. Whittaker, et al., “Autonomous exploration and mapping of abandoned mines,” *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 79–91, 2004.
- [7] K. Nagatani, S. Kiribayashi, Y. Okada, et al., “Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots,” *Journal of Field Robotics*, vol. 30, no. 1, pp. 44–63, 2013.
- [8] A. Agha, K. Mitchell, and P. Boston, “Robotic exploration of planetary subsurface voids in search for life,” in *AGU Fall Meeting Abstracts*, vol. 2019, 2019, P41C–3463.
- [9] W. Liu, D. Anguelov, D. Erhan, et al., “Ssd: Single shot multibox detector,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I* 14, Springer, 2016, pp. 21–37.
- [10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [11] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, “Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 8729–8736.
- [12] M. Palieri, B. Morrell, A. Thakur, et al., “Corrections to “locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time”[apr 21 421–428],” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3760–3760, 2021.
- [13] F. Yang, C. Cao, H. Zhu, J. Oh, and J. Zhang, “Far planner: Fast, attemptable route planner using dynamic visibility update,” in *2022 ieee/rsj international con-*

- ference on intelligent robots and systems (iros), IEEE, 2022, pp. 9–16.
- [14] A. Bouman, M. F. Ginting, N. Alatur, *et al.*, “Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 2518–2525.
- [15] J. Delmerico, S. Mintchev, A. Giusti, *et al.*, “The current state and future outlook of rescue robotics,” *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, 2019.
- [16] R. R. Murphy, S. Tadokoro, and A. Kleiner, *Disaster robotics*. Springer, 2016.
- [17] F. Amigoni, J. Banfi, and N. Basilico, “Multirobot exploration of communication-restricted environments: A survey,” *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 48–57, 2017.
- [18] M. Tatum, “Communications coverage in unknown underground environments,” *Masters’s thesis, The Robotics Institute, Carnegie Mellon University, USA*, 2020.
- [19] S. Scherer *et al.*, “Resilient and modular subterranean exploration with a team of roving and flying robots,” *Field Robotics Journal*, pp. 678–734, May 2022.
- [20] M. B. Dias, B. Kannan, B. Browning, *et al.*, “Sliding autonomy for peer-to-peer human-robot teams,” 2008.
- [21] F. W. Heger and S. Singh, “Sliding autonomy for complex coordinated multi-robot tasks: Analysis & experiments,” 2006.
- [22] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, “A survey of behavior trees in robotics and ai,” *Robotics and Autonomous Systems*, vol. 154, p. 104 096, 2022.
- [23] A. Agha, K. Otsu, B. Morrell, *et al.*, “Nebula: Quest for robotic autonomy in challenging environments; TEAM costar at the DARPA subterranean challenge,” *CoRR*, vol. abs/2103.11470, 2021.
- [24] A. Agha, K. Otsu, B. Morrell, *et al.*, “Nebula: Quest for robotic autonomy in challenging environments; TEAM costar at the DARPA subterranean challenge,” *CoRR*, vol. abs/2103.11470, 2021.
- [25] T. Roucek, M. Pecka, P. Cízek, *et al.*, “System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA subterranean challenge,” *CoRR*, vol. abs/2110.05911, 2021.
- [26] J. Bayer, P. Čízek, and J. Faigl, “Autonomous multi-robot exploration with ground vehicles in darpa subterranean challenge finals,” *Field Robotics*, vol. 3, pp. 266–300, Jan. 2023.
- [27] M. Quigley, K. Conley, B. Gerkey, *et al.*, “Ros: An open-source robot operating system,” in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.
- [28] N. Bagree, C. Noren, D. Singh, M. Travers, and B. Vundurthy, “Distributed optimal control framework for high-speed convoys: Theory and hardware results,” *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 2127–2133, 2023.
- [29] M. Colledanchise and P. Ögren, *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.
- [30] J. Zhang, C. Hu, R. G. Chadha, and S. Singh, “Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation,” *Journal of Field Robotics*, vol. 37, no. 8, pp. 1300–1313, 2020.
- [31] D. Q. Mayne, “Differential dynamic programming—a unified approach to the optimization of dynamic systems,” in *Control and dynamic systems*, vol. 10, Elsevier, 1973, pp. 179–254.
- [32] P. Toth and D. Vigo, *Vehicle Routing*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014.
- [33] S. Nahavandi, S. Mohamed, I. Hossain, *et al.*, “Autonomous convoying: A survey on current research and development,” *IEEE Access*, vol. 10, pp. 13 663–13 683, 2022.
- [34] P. Sriganesh, N. Bagree, B. Vundurthy, and M. Travers, “Fast staircase detection and estimation using 3d point clouds with multi-detection merging for heterogeneous robots,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 9253–9259.