Title

Human-Inspired Autonomous Navigation in Multi-Robot Systems: Behavioral Modeling, Cognitive Load, and Strategy Integration

Author

Mohaddesi, Seyed Amirhosein

Publication Date

2025

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA,
IRVINE

Human-Inspired Autonomous Navigation in Multi-Robot Systems: Behavioral Modeling, Cognitive Load, and Strategy Integration

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Sciences (CS)

by

Seyed Amirhosein Mohaddesi

Dissertation Committee:
Professor Jeff Krichmar, Chair
Professor Nikil Dutt
Associate Professor Elizabeth Chrastil

2025

DEDICATION

To my beloved wife, Narges, and my parents Kazem And Fariba
For their unconditional love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Page

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

Completing this dissertation has been a deeply rewarding and transformative journey, and it would not have been possible without the guidance, support, and encouragement of many people.

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Jeffrey Krichmar, for his invaluable mentorship, unwavering support, and continuous encouragement throughout my Ph.D. His deep knowledge, insightful feedback, and patient guidance have shaped not only this dissertation but my entire academic development. I am grateful for the freedom he gave me to explore my research ideas and for his trust in my abilities.

I would also like to thank the members of my dissertation committee (Professor Nikil Dutt, Professor Elizabeth Chrastil, and Professor Roy Fox) for their thoughtful feedback and constructive critiques, which have greatly improved the quality of this work. Their expertise and perspectives have been instrumental in refining my research.

I am indebted to my collaborators and lab mates in the Cognitive Anteater Robotics Laboratory (CARL), whose discussions, friendship, and teamwork made the research process enjoyable and stimulating. Special thanks to the Spatial Neuroscience Lab team for sharing their human navigation data, which was vital for my experiments.

To my friends and colleagues, thank you for the camaraderie, shared challenges, and encouragement, both inside and outside the lab. Your support helped me stay grounded through the highs and lows of graduate life.

I am forever grateful to my family for their unconditional love, patience, and belief in me. To my parents, your sacrifices and endless support have been my foundation. Thank you for instilling in me the values of hard work and perseverance. To my friends back home and around the world, your long-distance encouragement has meant so much.

Finally, I dedicate this work to the love of my life and my partner, Narges, whose unwavering support and presence at every step of this journey have meant everything to me. Together, we remain inspired by curiosity and driven by the desire to make a meaningful impact.

VITA

Seyed Amirhosein Mohaddesi

EDUCATION

Doctor of Philosophy in Information Computer Science (ICS) **2025**
University of California, Irvine *Irvine, CA*

Bachelor of Science in Computer engineering **2019**
Sharif University of Technology *Tehran, Iran*

PROJECTS

- **Platform to train and study navigation agents in ROS2 environment:** As a part of my Ph.D. thesis, I developed a realistic simulation environment using **ROS2 (Humble)** to assess the social aspects of human navigation strategies. I launched a simulation that enables real-time multi-agent navigation. Utilizing actual human navigation data collected by the Spatial Neuroscience Lab (SNL), I trained decision-making models in **PyTorch**, creating agents that employ Survey, Route and Graph strategies in their path planning. By implementing **SLAM**, **NAV2**, **Gazebo**, and **RVIZ** packages in **ROS2**, I emulated maze-shaped scenarios with agents equipped with obstacle avoidance, allowing agents to explore the environment at the same time. This was not possible before due to restrictions on human studies. My work applies human-inspired close-contact communication techniques to enhance agent interactions, providing a valuable toolbox for simulating and analyzing human navigation in complex environments. This project advances the study of human navigation and has the potential to enhance the performance of decentralized multi-agent rescue teams through improved communication methods.

- **Benefits of Varying Navigation strategies in Robot Teams:** Investigated the Benefits of Varying Navigation Strategies in Robot Teams by developing and conducting experiments inspired by human navigation research. In the **WEBOTS** environment, implemented obstacle avoidance and conflict resolution algorithms in a **(C++)** controller, and applied different human-inspired navigation strategies (Route, Survey, and Mixed) on simulated Clearpath PR2 robots. Analyzed how these strategies affect task completion time, environment exploration, and overall system effectiveness in multi-robot systems. Demonstrated that mixed navigation strategies offer a balanced trade-off between efficiency and coverage, highlighting the advantages of strategy variability. Contributed insights that can enhance the effectiveness of robotic teams in applications like exploration and search and rescue. Presented at "From animals to animats 17, SAB2024" The related poster was accepted at IEEE ICDL2024blue Publication.

- **Navigation and Cognitive Load in Telepresence Robots:** Explored the Impact of Autonomous Navigation on Cognitive Load in Telepresence Robots by leading a study comparing manual and autonomous navigation modes during a scavenger hunt task using telepresence robots. Developed and implemented autonomous navigation features, including **ROS SLAM** mapping for environment mapping and a user-friendly GUI using **PyQt** to facilitate interaction. Designed experiments to measure cognitive load, spatial awareness, presence, and user experience, and conducted user studies to evaluate performance. Demonstrated that autonomous navigation reduced cognitive load, improved movement efficiency, and improved memory and learning task performance, suggesting that incorporating autonomous features into telepresence robots can facilitate user operation and improve outcomes in educational, medical and workplace settings. Presented at blueIEEE ICDL2024.

- **8-bit quantization technique for spiking neural networks:** Developed an 8-bit Quantization Technique for Spiking Neural Networks to Reduce Embedded Device Power Consumption by implementing an 8-bit spiking neural network optimized for embedded computing using **PyTorch**. Designed and introduced an efficient 8-bit quantization method, which reduced power consumption by 12%–18% on **MNIST**, **CIFAR10**, and **CIFAR20** benchmarks, with only a minimal decrease in test accuracy of 3%–7%. Demonstrated significant energy efficiency improvements, contributing to the practical deployment of spiking neural networks on resource-constrained devices.

- Developed a Predictive Model for Lunar Lander Trajectory Using **Recurrent Neural Networks** and **Convolutional Autoencoders** by implementing a system in the **OpenAI Gym** environment. Designed and implemented a **convolutional autoencoder(CAE)** to compress 64×64 RGB image frames into compact representations, and integrated it with a **recurrent neural network (RNN)** to predict subsequent frames of the Lunar Lander trajectory controlled by random actions. Demonstrated that a simple RNN can learn Newtonian laws of motion and accurately predict complex spacecraft trajectories, providing insights that could enhance action policies in reinforcement learning methods and improve predictive modeling in physics-based applications.

- Developed a Bio-Inspired Obstacle Avoidance Controller for the **E-Puck** Robot Using Optic Flow in **WEBOTS** Simulation: Implemented a **(C++)** controller for the E-Puck robot that mimics real-life bee navigation by utilizing optic flow data from the robot's camera. Successfully enabled the E-Puck to navigate through a hallway filled with obstacles, demonstrating effective bio-inspired navigation strategies and contributing to advancements in autonomous robotic movement and obstacle avoidance techniques.

- Developed a **Convolutional Neural Network (CNN)** using **PyTorch** and **Torchvision** to classify house numbers from the Street View House Numbers (SVHN) dataset, achieving 98% accuracy by implementing data preprocessing techniques for image scaling and normalization, fine-tuning hyperparameters such as learning rate and batch size to optimize performance, and strengthening skills in neural network design and

implementation through hands-on experience with **deep learning** and **computer vision**.

PUBLICATIONS

Published

1. Navigation and Cognitive Load in Telepresence Robots Authors Grace Pan, Thea Weiss, **Seyed Amirhosein Mohaddesi**, John W Szura, Jeffrey L Krichmar Publication date 2024/5/20 Conference 2024 IEEE International Conference on Development and Learning (ICDL)
2. Benefit of Varying Navigation Strategies in Robot Teams Authors **Seyed A Mohaddesi**, Mary Hegarty, Elizabeth R Chrastil, Jeffrey L Krichmar Publication date 2024/9/7 Book International Conference on Simulation of Adaptive Behavior Pages 63-77 Publisher Springer Nature Switzerland

In review/preparation

1. A ROS2 Multi-Robot Simulation Platform for Human-Inspired Navigation Research

ABSTRACT OF THE DISSERTATION

Human-Inspired Autonomous Navigation in Multi-Robot Systems: Behavioral Modeling, Cognitive Load, and Strategy Integration

By

Seyed Amirhosein Mohaddesi

Doctor of Philosophy in Computer Sciences (CS)

University of California, Irvine, 2025

Professor Jeff Krichmar, Chair

This dissertation investigates the integration of human cognitive strategies into autonomous robot navigation across three interconnected studies. The first explores how varying levels of shared autonomy in telepresence robots impact user cognitive load and task performance. The second simulates diverse human navigation styles (route, graph, and survey knowledge) in robot teams and demonstrates the benefits of strategy diversity for task efficiency. The third presents a ROS2-based multi-robot simulation platform designed for scalable deployment and benchmarking of cognitive navigation behaviors. Performance evaluations confirm the system's scalability, and the platform is positioned to support future work in reinforcement learning, human-robot teaming, and cognitive modeling. Together, these contributions advance the understanding and implementation of human-like decision-making in autonomous robotic systems.

Chapter 1

Introduction

Over the past several decades, robotic navigation has made remarkable strides, from reactive path planning in constrained environments to robust autonomy in dynamic, unstructured settings. Yet, despite these advances, the field continues to grapple with fundamental challenges: aligning robotic behavior with human expectations, cognitive constraints, and collaborative needs. As robots are increasingly deployed in roles that require interaction with people, whether through direct telepresence or coordination in multi-agent teams, there is a growing need to design navigation systems that are not only technically sound, but also cognitively and socially compatible with human users.

This dissertation explores a human-centered perspective on robot navigation through a three-part investigation. Each part builds upon the previous part, forming a narrative arc that begins with individual user experience, expands into team-level strategic diversity, and culminates in the creation of a simulation framework capable of supporting both empirical and theoretical research on human-inspired navigation.

1.1 Part I: Telepresence Navigation and Cognitive Load

The journey begins with an exploration of telepresence robotics, where a remote user pilots a robot to interact with a distant environment. While telepresence systems promise greater accessibility and immersion, they also impose significant cognitive demands on the user, particularly in navigation. The first study in this dissertation evaluates how different levels of autonomous navigation support, ranging from manual control to fully autonomous point-to-point travel, affect a user's cognitive load, situational awareness, and task performance. Using objective performance metrics and subjective workload assessments, we find that appropriate levels of autonomy can substantially reduce user burden without compromising control or engagement. This work highlights the importance of shared autonomy as a design principle and sets the stage for investigating how robots can adaptively support humans in navigation contexts.

1.2 Part II: Benefits of varying Navigational Strategies in Robot Teams

While Part I focuses on a single human-robot dyad, Part II expands the scope to consider multi-robot teams operating autonomously. Inspired by cognitive psychology and spatial reasoning literature, we examine whether diverse navigation strategies, such as "route knowledge" (local paths) and "survey knowledge" (global maps), confer performance benefits when distributed across a team. Using simulation and controlled experiments, we evaluate teams of robots that follow homogeneous versus heterogeneous strategies during exploration and search tasks. The findings demonstrate that strategy diversity, where some agents follow locally efficient paths while others prioritize global coverage, yields synergistic effects: better overall efficiency, robustness to failure, and more flexible response to dynamic goals. These

results offer a compelling analogy to human teams, where different members may intuitively adopt different roles or strategies, and lay the groundwork for studying team cognition in robot collectives.

1.3 Part III: A ROS2 Simulation Platform for Human-Inspired Navigation

Having explored how humans interact with individual robots and how strategic variation benefits teams, the dissertation culminates in the development of a scalable ROS2-based simulation platform that unifies these insights. The platform enables researchers to simulate large teams of robots, each in its own namespace and equipped with SLAM, autonomous navigation, and behavioral monitoring capabilities, in a shared Gazebo environment. Crucially, the platform supports cognitive modeling and graph-based representations that reflect how humans form mental maps of space. This makes it possible to simulate human-inspired behaviors, test learning algorithms for navigation, and study the computational and communication tradeoffs of different team configurations. In addition, the system includes tools for performance benchmarking (CPU, memory, network load), enabling the evaluation of scalability and real-time feasibility.

This final component serves as both a synthesis of prior work and a launchpad for future investigations. It embodies the shift from studying individual human-robot interactions to engineering adaptive, human-aligned robot teams, and provides the infrastructure necessary for integrating learning-based models, reinforcement learning, and cognitive mapping techniques. By bridging empirical studies with system-level tools, this dissertation offers not just a series of findings, but a research framework for understanding and improving navigation in human-robot systems.

1.4 Research Goals and Contributions

Across all three parts, the central aim is to advance the understanding of how robotic navigation can be shaped by, and in turn support, human spatial cognition and team coordination. Specifically, this thesis makes the following contributions:

- Quantitative evidence that shared navigation autonomy reduces cognitive load in telepresence applications.
- Demonstration that strategic diversity in navigation styles improves performance in multi-robot teams.
- Development of a ROS2-based simulation tool that integrates mapping, navigation, and cognitive representations at scale.
- Performance analysis and reproducibility tools that validate the scalability and real-time feasibility of large-scale simulations.

In unifying these elements, the thesis presents a vision of navigation not merely as a technical challenge, but as a cognitive and social process, mediated by tools that enable robots to perceive, plan, and collaborate in ways that align with human understanding.

Chapter 2

Background

2.1 Cognitive Load and Human-Robot Interaction in Navigation

In human–robot interaction (HRI), particularly during navigation tasks, cognitive load refers to the mental effort required by a human operator to perceive information, make decisions, and control a robot. When teleoperating a mobile robot or telepresence system, the user often must divide attention between navigation (e.g. driving the robot, avoiding obstacles) and other goals (e.g. observing the environment or communicating), which can impose a high mental workload.[1][2] Cognitive load is a critical factor because excessive workload can lead to operator fatigue, errors, or reduced mission performance.

2.1.1 Measuring Cognitive Load

Researchers have developed various methods to quantify cognitive load in HRI. A common subjective instrument is the NASA Task Load Index (NASA-TLX), a post-task questionnaire

that asks participants to rate workload on multiple dimensions.[3] The NASA-TLX captures six factors – mental demand, physical demand, temporal demand, perceived performance, effort, and frustration – each rated on a scale (often 0–100)[3] An overall workload score can be obtained by averaging these factors[3]

This index has been widely used in teleoperation studies to gauge how demanding a navigation task is on the human operator[1]

In addition to subjective ratings, performance metrics serve as proxies for cognitive load. For example, longer task completion times, more frequent navigation errors or collisions, and lower success rates can indicate higher cognitive load. A recent study in robotic teleoperation recorded a rich set of measures – including brain activity via fNIRS, galvanic skin response (GSR), heart rate, subjective NASA-TLX scores, and task performance data – under varying workload conditions[1] The findings suggested that performance metrics (like task success and errors) were the most sensitive indicators of different workload levels, whereas many physiological signals did not reliably distinguish high vs. low workload conditions.[1]

Nonetheless, physiological proxies are also explored in HRI research: changes in heart rate variability, pupil dilation, EEG/fNIRS signals, and GSR can sometimes reflect changes in mental effort. These objective measures can be recorded continuously and non-intrusively[2], offering a real-time window into operator cognitive state. However, interpreting physiological data in complex tasks is challenging, and factors like stress or motion can confound signals. Thus, many researchers triangulate multiple metrics – subjective (NASA-TLX), behavioral (performance), and physiological – to assess cognitive load in robot navigation tasks.[1,2]

2.1.2 Autonomy to Reduce Workload

The level of robot autonomy significantly influences user workload and experience. In pure teleoperation (manual remote control), the human must make all navigation decisions and control all motions, which can be cognitively demanding, especially in cluttered or dynamic environments. Users often face limited situational awareness (due to narrow camera field of view or latency) and must perform low-level control (e.g. continuous steering and speed adjustments) on top of higher-level objectives[14] This high mental demand has been documented in telepresence robot studies: for instance, remote operators navigating a robot through crowds report difficulty due to the need to avoid pedestrians with only a partial view of surroundings.[1]

As a result, researchers have introduced shared autonomy approaches, where the robot's on-board intelligence assists the human in navigation. Shared autonomy lies between full manual control and full automation – the human and robot share control in a perception-action loop[45] One way to implement this is to offload low-level navigation tasks (obstacle avoidance, path following) to the robot, letting the human issue higher-level commands (like select a goal or general direction). This approach can markedly reduce the operator's cognitive load[14] In an illustrative study, Pang et al. enabled a telepresence robot to autonomously perform "high-level, human-like navigational behaviors" such as person-following and obstacle avoidance, adhering to social norms like proper following distance.[12]

In user trials, they found that when the robot handled its own navigation (autonomous mode), users completed routes faster and with zero collisions, and reported significantly lower workloads on the NASA-TLX.[1]

In fact, the mean self-reported workload was 45.7% higher under manual operation compared to autonomous navigation.[1], highlighting the large impact of autonomy on reducing human effort. Nearly all participants experienced less stress and effort when the robot drove itself,

allowing them to focus on interacting with people or observing the remote environment[1]. This evidence aligns with broader findings that adding autonomous assistance in teleoperation improves not only objective performance but also the user's subjective experience (lower frustration and effort)[2][4]

2.1.3 Teleoperation Challenges and Adaptive Support

Despite the benefits of autonomy, finding the right level of shared control is an active research area. On one hand, fully autonomous navigation can free the user from low-level tasks and thereby minimize workload[4] On the other hand, users sometimes prefer to remain "in-the-loop" for a sense of agency and trust[5]

If the autonomy is not transparent or reliable, operators may feel anxious or out of control, which could increase cognitive load due to monitoring the autonomy. Studies have noted that telepresence operators value having control over the robot's actions and may distrust fully self-driving behavior in certain scenarios.[1]

As a result, modern telepresence systems often allow an adjustable autonomy or shared control mode. For example, a robot might normally drive itself but immediately give control back to the human in ambiguous or critical situations (or whenever the user overrides). Adaptive support refers to dynamically tuning the level of assistance based on context or the user's state. If sensors detect that the user is overloaded (e.g. via physiological measures or degraded performance), the robot could increase assistance (taking over more functions); conversely, if the user is idle or inattentive, the system might engage them more or let them practice manual control. Preliminary research in HRI suggests that such adaptive autonomy can keep workload within manageable bounds, though it requires reliable detection of cognitive load in real time[4][5] Some approaches use secondary task performance or reaction-time tasks to gauge operator workload and trigger assistance when performance

drops[5] Others explore direct biofeedback (e.g., a brain-computer interface that switches autonomy modes when EEG indicates high mental effort). While still nascent, these adaptive shared-control systems aim to maintain situational awareness and engagement for the human operator without overwhelming them. In summary, managing cognitive load is crucial in human-robot navigation interfaces. By measuring workload through tools like NASA-TLX and performance analytics, researchers have shown that appropriate autonomy and shared control can dramatically improve the user experience. Telepresence and mobile robots that handle routine navigation tasks allow users to focus on strategic decisions and interaction, resulting in safer and more efficient operations with lower reported mental demand[1 4] These insights underscore the importance of human-centric design in robot navigation: the autonomy must be sophisticated enough to ease the user's burden, yet transparent and adaptable to keep the human in a comfortable "feedback loop" with the robot. This balance between human and robot control will be a recurring theme as we examine navigation strategies and multi-robot systems in the following sections.

2.2 Navigation Strategies in Humans and Robots

Human navigation is a rich cognitive process involving multiple strategies and representations. Psychologists distinguish between different types of spatial knowledge that humans acquire when learning an environment: route knowledge, graph (topological) knowledge, and survey knowledge[?] . Route knowledge consists of sequences of actions or landmarks (e.g., "turn left at the pharmacy, then right at the church") needed to go from one place to another, without requiring an overarching view of the environment[6]. It is a procedural, turn-by-turn understanding – one might follow a familiar route by remembering a chain of cues, yet have no idea of the global layout. Graph knowledge (also called topological knowledge) represents the environment as a network of interconnected places[?] . In this level of understanding, a

person knows which locations connect to which (the adjacency relationships), allowing them to take novel paths by following known links[?] . For example, you might realize you can go from A to C via B even if you've never traveled directly from A to C, as long as you know the connectivity graph. However, pure graph knowledge lacks precise distances or directions – it enables topologically valid shortcuts (using alternative routes between nodes) but not necessarily straight-line shortcuts through unexplored terrain[?] . Finally, survey knowledge is an integrated, metric map-like understanding of the environment's geometry[6]. With survey knowledge, one knows the approximate distances and angles between places and can point toward or directly navigate to unseen goals ("as the crow flies")[?] . This corresponds to having an internal cognitive map akin to a bird's-eye view, supporting efficient shortcuts and novel route planning. These three spatial representations form a progression from inflexible (route, which is sequential and rigid) to flexible (survey, which is global and allows creative navigation) representations[?] . Classic developmental theories in spatial cognition, such as Siegel & White's framework, propose that humans (especially children) initially encode routes, then gradually build up topological connections and eventually a full survey map of a space[7]. Subsequent research refined this view, noting that these forms of knowledge can develop in parallel to some extent and that graph-like representations often underlie human navigation decisions even when metric knowledge is incomplete[8].

2.2.1 Human Navigation Strategies

Beyond the knowledge one possesses, humans exhibit navigation strategies – preferences in how they use their knowledge to plan and execute routes. Strategies can be influenced by individual tendencies, cognitive abilities, and context. For instance, given the choice, some people consistently prefer a familiar route over an uncertain shortcut, even if they have survey knowledge of the area. Others are more inclined to attempt novel shortcuts leveraging their map knowledge. This dichotomy has been described as "route-following" vs. "survey-

based" strategy, or alternatively as response-based vs. place-based navigation[9]. Marchette et al. termed these extreme cases as "creatures of habit" (who rely on habitual routes – a response learning strategy) versus "cognitive mappers" (who engage place-learning to find new paths)[9]. Crucially, a person's chosen strategy is not always perfectly correlated with their knowledge; even a navigator who has acquired an accurate cognitive map might stick to a well-trodden path due to risk aversion or low confidence[10?] . Conversely, a person with only partial knowledge might still explore optimistically. Cognitive factors like working memory and executive function also play a role in strategy: planning a shortcut requires mentally simulating a new route and monitoring one's position, which is cognitively demanding, whereas following a known route can be more automated[?] . Personality traits and spatial anxiety levels have been found to influence strategy as well[10]. For example, those with high spatial anxiety or low confidence in their orientation skills might avoid detours and stick to routine paths (a conservative strategy), while those who are more adventurous or have a "growth mindset" about navigation tend to explore and learn new shortcuts[9,10]. Researchers have even quantified individual strategy bias using metrics like the "Solution Index," which compares one's propensity to take shortcuts versus known routes across trials[9,10].

Understanding these human strategies is important not only in psychology but also for designing robotic systems that interact naturally with people or emulate human-like decision making.

2.2.2 Implications for Robotic Navigation

Robotics researchers have long been inspired by human spatial cognition to improve autonomous navigation. Early robotic mapping paradigms drew directly from human concepts, most notably the idea of a cognitive map introduced by Tolman (1948)[7] and later supported by neuroscience (the discovery of place cells by O'Keefe and grid cells by Moser

et al.[11,12]). In robotics, this inspired the development of topological maps and hierarchical representations. A purely metric map (like an occupancy grid or detailed CAD model) allows precise localization and motion planning, but it can be computationally heavy and not very human-readable. A topological map, in contrast, models the environment as a graph of nodes (significant places, e.g., rooms, corridors, intersections) and edges (navigable paths between places). This is analogous to the human graph knowledge: it captures connectivity and order but not exact geometry. Topological navigation lets a robot plan at the level of "go from Lab to Office via Hallway" by following graph links, which is efficient and also aligns with how humans describe routes. Many robotic navigation systems combine both approaches – for example, build a metrical map for localization and obstacle avoidance, but also maintain a higher-level topological graph for efficient path planning and semantic understanding. Notably, Kuipers' Spatial Semantic Hierarchy (SSH) framework proposed layers of representation (distinctive states, topological graphs, and metrical layers) to mirror human-like spatial learning, bridging the gap from sensor data to cognitive map[13].

Modern SLAM (Simultaneous Localization and Mapping) techniques often produce a pose graph of the robot's trajectory: nodes represent robot poses or keyframes, and edges represent spatial constraints (distance, angle between poses, or loop closures). This pose graph is essentially a topological graph decorated with metric information – strikingly similar to the "labeled graph" hypothesis for human cognitive maps[2,8]. Chrastil and Warren provide evidence that people spontaneously form a mental representation of environments that is graph-like with metric labels. In their experiments, participants who freely explored a virtual environment could later navigate via novel shortcuts (implying a topological understanding beyond just learned routes) and tended to choose the shortest path to targets (implying metric distance knowledge)[?] . This supports the idea that humans internalize a cognitive graph – a network of places connected by paths annotated with approximate distances or travel costs[?] .

Robotic systems, likewise, benefit from storing hybrid maps that contain both topological structure and metric estimates (e.g., distance along each corridor). Such graph-based representations make path planning computationally efficient (as graph search) and can be more robust to dynamic changes than full metric maps. They also facilitate multi-robot coordination (as we will see in Chapter 5) by providing an abstract layer for sharing map information.

Another angle where human strategies inform robotics is in multi-agent exploration and team diversity. Humans exhibit cognitive heterogeneity – variations in spatial abilities and strategy preferences – and in group navigation tasks, different individuals may complement each other. For example, one team member might be adept at survey learning (quickly sketching a map of the area), while another excels at route following (remembering specific sequences). In analogous fashion, a team of robots could employ heterogeneous navigation strategies to improve overall performance. One robot could methodically follow frontier-based exploration (covering unknown areas grid by grid), while another uses a random-walk or landmark-seeking strategy; the combination might yield faster coverage and mapping than homogeneous behavior. Studies in multi-robot exploration suggest that strategy diversity can prevent robots from all getting stuck in the same local minima or redundant paths[14]. Furthermore, if robots share their learned maps, a robot that prefers thorough map-building can provide a reliable base map, whereas another that rushes ahead can extend the frontier – together achieving both speed and accuracy. In human teams, diversity in wayfinding strategies has been theorized to handle uncertain environments better, as some members ensure safe choices while others push for optimal shortcuts[9,10]. Translating this to robotics is an open research question, but it aligns with the concept of ensemble intelligence: multiple agents with different "cognitive styles" might outperform a uniform team.

In summary, human navigation research provides a taxonomy of spatial knowledge (route, graph, survey) and reveals that people use a spectrum of strategies from habitual route-

following to flexible map-based shortcutting[2,5]. These insights guide robotic navigation in two ways. First, they motivate representation – robots, like humans, can benefit from multi-layer maps that include topological graphs and metric information, enabling efficient and flexible path planning. Second, they inspire strategy design – robots might emulate human heuristics (e.g., favoring known safe paths when uncertain, or using exploration behaviors analogous to human curiosity) and even intentionally vary strategies within a multi-robot team to mirror the advantages of cognitive heterogeneity. By grounding robotic navigation strategies in human-inspired principles, we can build systems that are not only effective but also more predictable and intuitive when interacting with human operators or teammates. This human-centric perspective becomes especially pertinent when multiple robots are deployed together, as discussed next.

2.3 Multi-Robot Systems and Simulation Platforms

Robots rarely operate truly alone – many applications envision multi-robot systems where teams of robots coordinate to accomplish tasks more efficiently and robustly than a single robot could. In the context of navigation and exploration, multiple robots can cover more area in parallel, share sensor data to overcome individual limitations, and provide redundancy in case one unit fails. A cooperative multi-robot navigation scheme can alleviate problems that plague single-robot missions, such as cumulative localization drift, long mission times, and vulnerability to single-point failures. By distributing the work, multi-robot teams offer high efficiency (faster environment coverage) and high fault tolerance, as well as cost-effectiveness if many inexpensive robots replace one sophisticated machine. For example, in search-and-rescue, a swarm of ground robots can rapidly explore different wings of a building simultaneously, and if one gets stuck or lost, others can still fulfill the mission. Likewise, in mapping large areas, robots can split the environment, each mapping a portion

and then fusing their maps, reducing total time. Research shows that multi-robot SLAM (Simultaneous Localization and Mapping) approaches can reduce the computational load on each robot and limit error accumulation by sharing information[15]. One robot's detections of landmarks can help correct another's pose uncertainty, and loop closures found by any robot (when revisiting a previously seen area) can be broadcast to all, tightening the global map for everyone. In short, a team can leverage collective perception and memory to outperform isolated operation.

However, these benefits come with significant technical challenges. *Coordination and Communication:* Robots must communicate to share their knowledge (maps, positions, intentions) and to coordinate actions (like avoiding collisions or not duplicating effort). Limited bandwidth or unreliable communication links can bottleneck a multi-robot system. If each robot naively broadcasts large sensor data (e.g. high-res maps or video) to others, the network can become saturated, causing delays or dropouts. Thus, multi-robot algorithms often incorporate strategies to minimize communication load – for instance, sharing only processed map updates or sparse landmarks instead of raw sensor streams, or using event-driven communication (only send updates when significant changes occur). *Multi-agent coordination:* Additionally, multi-robot teams face the multi-agent coordination problem in deciding which robot goes where. Without planning, robots might clump together or explore the same area redundantly. Effective strategies like frontier allocation ensure robots spread out to cover different frontier cells (boundaries between known and unknown space) in exploration tasks. Robots may also need to negotiate priorities at intersections or narrow passages to avoid traffic jams, which introduces aspects of multi-robot path planning and even social conventions if humans are around. *Map merging and localization:* When each robot builds its own map, those maps must eventually be merged into a consistent global map. If the robots' starting relative positions are known (e.g., deployed from the same point or with a known offset), they can align their maps easily in a common coordinate frame. Often, teams start dispersed with unknown relative poses – then the system must identify when two robots have

an overlapping view of the same area (a map overlap) and compute a transformation to align their maps. This is akin to recognizing a rendezvous, where a robot might detect a feature or location that robot B has seen before; by matching these observations, the system infers how B's local map fits into A's map. Techniques for distributed SLAM handle this by sharing local sub-maps or keyframes and finding loop closures across robots[15]. Many algorithms adopt a centralized architecture where a central server collects data from all robots and runs a joint SLAM optimization (assembling a master map)[15]. Others pursue a fully decentralized approach where robots exchange data peer-to-peer and update their maps on the fly – the latter can be more scalable and robust to single-point failure, but is more complex to design.

Regardless of approach, careful handling of coordinate frames and transformations (TF) is required so that each robot knows its location with respect to the team or global frame.

2.3.1 ROS and ROS2 Frameworks

The Robot Operating System (ROS) has become a de facto platform for developing multi-robot systems, thanks to its flexible publish-subscribe architecture and extensive toolset. In ROS (often referring to ROS 1), multi-robot support is achieved via namespaces and node remapping. Each robot is launched with its own namespace (e.g., /robot1/ and /robot2/), so that topics like /robot1/scan and /robot2/scan separate the LiDAR data for robots 1 and 2. Namespacing ensures each robot's messages are isolated, preventing cross-talk, while still allowing intentional data sharing on specific topics if desired. Managing TF (coordinate transform frames) is also critical: one typically gives each robot its own TF tree (with frames like $\text{robot1}_{b}ase_{l}ink, robot1_{m}ap)toavoidconfusion.ROS1'scentralizeddesign(withasinglemasternode)meansal$

ROS 2, the newer generation version, addresses some issues via a decentralized middleware (DDS – Data Distribution Service). In ROS 2 there is no central master; instead, discovery and messaging are distributed. Multi-robot deployments in ROS 2 still rely on namespacing

to distinguish each robot's topics[?] , but additionally, DDS offers the option of using different Domain IDs for different groups of robots to segregate their communications. By configuring domain IDs, one can run multiple ROS 2 networks in the same physical space that do not interfere (useful if, say, two separate teams of robots operate nearby). Within a single ROS 2 domain, topic namespaces keep messages organized per robot, as noted in NVIDIA's Isaac Sim documentation: managing unique namespaces is "crucial for multi-robot simulations to ensure that each robot's topics are uniquely identifiable" [16].

In practice, a multi-robot ROS2 application might launch N instances of the navigation stack (one per robot), each under a namespace /robot$_i$/, $and all can run concurrently on one computer or across mult$
$robot use in mind – they encourage modular use of costmaps and planners per robot instance and support behavior$
$robot coordination^{?}$. $Nav2 can be run for multiple robots by launching duplicates of its servers (planners, contr$
$time capabilities to handle the increased load.$

2.3.2 Simulation Platforms

Simulating multiple robots is an important step before real-world deployment, used for debugging algorithms and conducting experiments. Traditional robot simulators like Stage and Gazebo have been used extensively for multi-robot simulation. Stage is a 2D simulator (no 3D physics) that was popular in early multi-robot research due to its simplicity and speed – it can simulate dozens of robots as moving point bots with range sensors in a flat environment. For instance, researchers used Stage to prototype multi-robot exploration and patrol algorithms where a high number of robots and large environments are needed, but fine-grained physics are not critical. The limitation is that Stage does not provide realistic dynamics or 3D sensing; it is essentially idealized (no wheel slip, perfect sensor data apart from user-defined noise). Gazebo, on the other hand, is a 3D physics simulator that can integrate with ROS and simulate realistic sensors (camera, LiDAR) and robot mechanics [17].

Gazebo (now evolving into Ignition Gazebo) allows high-fidelity tests of multi-robot systems, including checking for interference (robots pushing each other, sensor crosstalk like one robot's laser hitting another). Using Gazebo for multiple robots requires careful configuration: each robot model must be instantiated with a unique name and typically launched with a ROS namespace so that their plugin ROS interfaces (sensors, controllers) don't conflict[17,18]. Gazebo spawns all robots into a shared world where they can physically interact. One challenge encountered is resource contention – as the number of robots grows, the simulation can become very heavy to run. Multiple robots mean multiple sensor data streams and more physics calculations; for example, 10 robots each with a 2D laser scanner publishing 10Hz will produce 100 scans per second to process. Real-time performance might suffer, so users often reduce sensor update rates or level of detail when simulating large fleets. Another challenge is managing the ROS communication within a single Gazebo instance: if not properly namespaced, all robots' sensors might publish to the same topic (creating nonsense data)[18]. Best practices dictate launching each robot with its own namespace and remapping Gazebo plugin topics accordingly, which ROS launch files can facilitate. In ROS 1, some opted for a multi-master approach (running separate roscore for each robot and bridging between them) to scale better, but ROS 2's single domain approach often suffices with correct naming. Apart from Stage and Gazebo, other tools have been employed: Microsoft AirSim (for aerial robots, can simulate multiple drones)[19], Webots and CoppeliaSim (which allow multi-robot scenarios with scripting)[20,21], and custom simulators in research papers tailored to specific multi-robot setups (for instance, multi-UAV flight simulations).

2.3.3 SLAM and Navigation Software

Multi-robot simulation frequently involves each robot performing SLAM and navigation. There are off-the-shelf algorithms that support multi-robot mapping. Google's Cartographer SLAM, for example, includes a multi-trajectory mapping feature whereby data from multiple

robots (trajectories) can be fed into a single mapping backend to produce a unified map[22]. In practice, this is done by time-synchronizing the robots' pose graphs and adding inter-robot loop closure constraints when they observe the same areas. This requires a central Cartographer server or a robust communication of submaps between robots. The result is a merged occupancy grid covering all explored areas. Cartographer demonstrated that even without constant communication, distributed robots can periodically exchange compact submaps and still achieve a consistent global map when those submaps are matched and merged[22].

Another toolkit, SLAM Toolbox (a ROS2-friendly mapping suite), supports lifelong mapping and merging serialized maps. While primarily used for single-robot mapping, some users have extended it to multi-robot cases by having one robot's map server accept updates from others (assuming all in a shared coordinate frame)[23]. These tools, however, have limitations: robust multi-robot SLAM still struggles if the relative initial positions are unknown and there is no overlap in observations for a long time – the maps might remain disconnected until a meeting occurs. Also, computational load can spike when merging large maps, so often a powerful central computer is used in simulations to handle the SLAM back-end for all robots. For navigation, each robot typically runs its own path planner and local controller (like Nav2's planners). An interesting scenario is multi-robot path coordination, ensuring that robots do not collide with each other. Basic setups treat other robots as dynamic obstacles (each robot's sensor will detect its teammates, and the planner will route around them). More advanced approaches explicitly coordinate plans (e.g., scheduling crossing times in an intersection). In simulation, one must also consider if robots share a map or not for planning: if each robot plans on its own partial map, they might not anticipate a collision until sensors see each other; if they plan on a global shared map that includes other robots' positions, they can be more proactive. Frameworks like ROS Nav2 do not yet have native multi-robot coordination algorithms built in; those are usually custom additions in research. Still, simulation platforms enable testing such additions in scenarios with multiple

robots. In summary, multi-robot navigation combines algorithmic challenges (coordination, map merging, communication constraints) with software engineering challenges (configuring frameworks like ROS to handle multiple instances, and using simulators efficiently). ROS and ROS 2 provide the infrastructure (namespaces, distributed middleware) to get multiple robots running and talking, while tools like Gazebo provide a sandbox to experiment safely [17]. Prior research tools, such as Stage and newer techniques in distributed SLAM set the stage, but scaling to large teams remains a non-trivial task due to network, computation, and data association issues [22]. These issues are important to recognize as we develop human-inspired navigation algorithms – for instance, if we propose that robots share "cognitive maps" (graph-based representations) with each other, we must ensure the system can actually exchange and merge those representations in real time.

2.4 Cognitive Maps and Graph-Based Representations

The notion of a cognitive map – an internal representation of spatial layout – has been influential in both psychology and robotics. First popularized by Tolman in 1948 as an explanation for how rats learned mazes [7], the term captures the idea that organisms form an internal map of their environment to plan novel routes and navigate flexibly. In humans, cognitive maps are evident in our ability to infer shortcuts, point to unseen locations, or recognize a different path leads to the same destination [?] . Psychologically, a cognitive map is often considered an amalgam of topological structure (which places are connected) and metric information (approximate distances and directions). Neuroscience has provided compelling evidence for physical substrates of cognitive maps: the discovery of place cells in the hippocampus (neurons that fire when an animal is in a specific location) and grid cells in the entorhinal cortex (which fire in periodic spatial patterns, suggesting a distance measuring system) supports the existence of map-like representations in the brain [24,25]. Together, these

and other cells (boundary cells, head direction cells) form a neural positioning system that encodes both location and spatial relationships, essentially a graph with distances – strikingly similar to the properties of human cognitive maps hypothesized by behavioral studies[?] . For example, a person can mentally connect locations A–B–C (topology) and also estimate that A to C is, say, about 2 km northeast (metric), even if they've never traveled directly. This aligns with the "labelled graph" model of cognitive maps: a network of nodes (places) linked by edges that carry labels (e.g., distance or travel time)[8]. As mentioned earlier, Chrastil and Warren's experiments concluded that people learn an internal graph of the environment labeled with metric lengths – participants navigated to targets along novel routes that were both topologically correct and metrically near-optimal, indicating both aspects in their internal model[8]. In robotics, map representations echo these principles. Metric maps (like occupancy grids or point clouds) precisely chart the geometry of the environment. They are essential for low-level tasks such as obstacle avoidance and accurate positioning. However, pure metric maps are memory-intensive and more than what is necessary for high-level navigation decisions (like deciding where to go next). Graph-based maps (topological maps) abstract the environment into a graph of important states: for instance, each room could be a node, and a door between rooms would be an edge. Early mobile robots in the 1990s used topological maps for global path planning because it was computationally efficient and closer to how humans might describe a route (through a sequence of named places)[13]. A downside is that a topological map alone doesn't tell the robot how to execute the move along that edge – that's where metric information or control routines come in. Modern approaches often use hybrid maps that combine metric and topological elements. A common approach in SLAM is pose-graph SLAM, where each robot pose (or each keyframe of sensor data) is a node in a graph, and constraints (odometry measurements, landmark observations, loop closures) form the edges[26].

Solving SLAM is then a matter of optimizing this graph to find a configuration of poses that best satisfies all constraints – this is inherently graph-based and has become a dominant

paradigm due to its efficiency and elegance[26]. Once optimized, the node positions can be used to produce a metric map (e.g., by plotting sensor readings in the refined poses) for fine navigation. Notably, if we interpret the pose graph as places (nodes) with estimated distances (edge lengths), it mirrors the structure of a cognitive map; indeed, some researchers draw parallels between robot pose graphs and animal cognitive maps, even suggesting that loop closure detection in SLAM is analogous to an animal recognizing a familiar location via sensory cues[4,13]. There is also a line of work explicitly connecting human cognitive mapping theories to robotic mapping algorithms. For example, Kuipers (2000) proposed that a robot can build a cognitive map by first identifying distinctive places (e.g., corridor intersections or extreme points in sensor data) and then linking them topologically, subsequently enriching those links with metric estimates – effectively building a cognitive graph[13]. This approach was implemented in the Spatial Semantic Hierarchy and other cognitive robotics models, enabling robots to describe their environment in human-like terms ("I am in the kitchen, which is connected to the living room") while also knowing metric facts ("the kitchen and living room centers are 5 meters apart through a 2m doorway")[13].

Another connection to cognitive maps is the use of graph search algorithms on these maps: robots plan routes on topological graphs using Dijkstra or A*, analogous to how humans mentally search their cognitive map for the best route. If the graph's edges have weights corresponding to travel effort or distance, the robot can choose the shortest or safest path, similar to a human favoring the quickest route or perhaps a route that stays on main roads (if the weights are adjusted for preference). Topological vs. Metric Maps: There has been debate and research on the advantages of each. Topological maps are compact and facilitate large-scale planning (since complexity grows with number of places, not area size), but to follow an edge, the robot might rely on a local metric patch or reactive control which might fail if conditions change. Metric maps handle local detail but don't inherently capture the place-to-place relationships abstractly. The synergy of both is often ideal: many robotic systems use a two-layer navigation system, where a high-level planner computes a sequence

of waypoints or subgoals (using a graph of regions or rooms), and then a local planner moves the robot to the next waypoint using the metric occupancy grid[23]. This two-layer approach is efficient and aligns with human behavior – people often plan in terms of landmarks ("go to the library, then to the cafeteria") and only worry about exact turns when executing each segment[8].

2.4.1 Cognitive Maps in Multi-Robot Context

Using graph-based representations can be very efficient when multiple robots share information. Instead of exchanging full raw maps, robots could share a list of nodes (landmarks) and connections they have discovered. For example, Robot A might inform Robot B: "I found that Room1 connects to Room2 and Room3, and approximately how far apart they are." Robot B can integrate this into its own graph if it overlaps, or append it if those areas were unknown. This is significantly less data than sharing entire occupancy grids of Room1, Room2, etc. It also resonates with how multiple humans share spatial knowledge (e.g., giving each other directions by mentioning key intersections and paths). Some research in multi-robot SLAM leverages this by having each robot extract a topological skeleton from its metric map and then merging those skeletons[27]. The challenge, of course, is ensuring that when two robots talk about "Room1" they mean the same physical room – requiring some loop closure or place recognition across robots. If achieved, a shared cognitive graph allows distributed navigation: each robot knows the overall layout that the team has explored, which aids in task allocation (a robot can volunteer to go to an unexplored node near its position) and in avoiding conflicts (knowing where others are or will be on the graph). Furthermore, cognitive maps provide a common language for human-robot teams. A human supervisor could query the team's map ("show me how all the rooms are connected") or give commands like "All robots, converge in the lobby" which the robots translate to nodes on their graph. This is easier if the robots internally use a place/graph representation than if

they only had raw occupancy grids. In summary, cognitive maps form a unifying concept that links human spatial cognition and robotic navigation. Humans appear to navigate using a graph-like mental representation of space augmented with approximate distances[8], and robots effectively do the same when they perform graph-based SLAM or topological navigation. Graph representations enable efficient planning and natural communication, whereas metric representations ensure precise maneuvering – combining both yields the best of each. As we move forward in this dissertation, these ideas will come together: we aim to design human-inspired multi-robot navigation methods that borrow heavily from human cognitive maps and strategies, implemented on modern robotic platforms (ROS2, Nav2, etc.) and evaluated in simulation. The background threads covered – cognitive load in HRI, human vs robot navigation strategies, multi-robot coordination, and cognitive map representations – are all interrelated in our research question. For instance, reducing an operator's cognitive load might be achieved by having robots navigate in a human-like way (so their behavior is predictable and their map is understandable to the human). Likewise, leveraging multiple robots requires robust map-sharing, which could be accomplished by sharing cognitive graph data between robots and humans.

In the next chapters, we will build on this background: we will formulate the specific problem addressed by the thesis, describe the methodology for infusing human-inspired strategies into multi-robot systems, and present simulation frameworks (built on ROS2/Gazebo) developed to test these concepts. Ultimately, the goal is to bridge the gap between human spatial cognition and robotic navigation, improving multi-robot autonomy in a way that remains compatible with human operators and teammates. The insights from cognitive load, strategy diversity, and mapping reviewed here will guide the design and analysis of the proposed system as detailed in the following chapters.

Chapter 3

The Benefits of Autonomous Navigation in Telepresence Robots and its Effect on Cognitive Load

Note: This chapter was previously published in *G. Pan, T. Weiss, S. A. Mohaddesi, J. W. Szura, and J. L. Krichmar, "Navigation and Cognitive Load in Telepresence Robots", IEEE International Conference on Development and Learning (ICDL), 2024.*.

3.1 Abstract

Telepresence robots have become integrated into educational, medical, and workplace settings as the capabilities of remote technologies progress. Considering the recent pandemic, the employment of telepresence robots to digitally represent individuals in remote environments has become more important. The goal of this study was to investigate if implementing autonomous navigation through telepresence robots could reduce cognitive load and make

operation easier for the user. While many studies involving telepresence robots examine navigation, there has been little research on how different types of navigation affect cognitive load and user experience. The present study measured differences between autonomous and manual navigation on cognitive load, spatial navigation, presence, and user experience during a scavenger hunt task using a telepresence robot. We hypothesized that participants who used the autonomous navigation feature would experience reduced cognitive load and better task performance than those who manually navigated the robot. We found that in the autonomous navigation condition, participants were able to navigate around the environment more efficiently and performed better in a memory task than those in the manual navigation condition. These results suggest that incorporating autonomous navigation functions into telepresence robots may lessen cognitive load, enabling individuals to attend to more stimuli in their environment.

3.2 Introduction

With both the rise of the robotics industry and the recent pandemic, the applications and demand for remote technologies such as telepresence robots have increased tremendously[28]. With the limitations placed on social gatherings, telepresence robots have become a valuable option for in-person meetings and communication. In 2020, the telepresence robotics industry was valued at 165.4 million dollars, and it is now estimated to grow to 187.47 million dollars by 2026[29]. Telepresence robots continue to be used to participate and interact with others in various settings like schools, hospitals, and workplaces[30][31]. There is clearly a market for these telepresence technologies, and as the use of telepresence becomes more widespread, enhancing certain user experiences like ease of use, and efficient navigation have become imperative to telepresence functionality.

A key aspect in improving the telepresence experience is decreasing the amount of cognitive

load needed to operate the robot. Cognitive load is defined as the amount of working memory being occupied[32] and reducing this load during teleoperation may free users to better attend to tasks and people in their environments. In understanding the cognitive load of performing a task, the NASA Task Load Index (NASA-TLX) assessment is usually given to measure how much cognitive function is being employed. The assessment is a subjective measurement of cognitive load which estimates participants' responses across six dimensions: mental demand, physical demand, temporal demand, performance, effort, and frustration[33]. Many researchers use the NASA-TLX assessment as their main measurement of cognitive load, but its weakness lies in its subjective estimation of cognitive function. To supplement this, another possible tool to examine cognitive load, especially in embodied learning scenarios, is to implement a dual-task. Dual-tasks are a more objective measurement of cognitive load in that cognitive load is calculated through the participant's performance on a secondary task that has no connection to the primary task[34]. This secondary measurement of cognitive load may fill in the gaps of the NASA-TLX assessment by providing a better understanding of the amount of cognitive load used while completing a task via telepresence.

Since cognitive load pertains to how much information is present in working memory at one time, experiencing a high cognitive load can negatively impact an individual's learning and attention to salient stimuli[35]. Currently, there is a gap in knowledge concerning how cognitive load is impacted in task performance when using telepresence robots, and how useful autonomous navigation may be as a telepresence feature[36]. Therefore, it is important to factor in the demands on working memory and cognitive load while operating telepresence robots. While telepresence robots allow individuals to be present in their work or school environments, the overwhelming amount of effort necessary to operate the robot may inundate the individual's working memory and take away from the usability of this technology. In the case of homebound students, the cognitive load required to navigate the robot interferes with their ability to speak at the same time[31], and in the case of large-scale environments, users of telepresence robots experience more isolation due to the challenges of navigating

and avoiding obstacles in a large space[36].

Autonomous features like self-navigation may combat these problems of information overload as the navigation decisions are handed over to the robot. Having the robot be responsible for planning out the route to a desired destination removes much of the cognitive processing required of the user, freeing additional space in perception and working memory for other tasks like conversing with their instructors, peers, or colleagues while moving to another location, and attending to more stimuli in their environments.

In order to improve user experience of telepresence robots, the ease and type of navigation used while operating the robot also plays a crucial part. Manual navigation is a more traditional and user-focused mode of navigation where the robot relies on the user to operate the controls for movement. Autonomous navigation, on the other hand, allows the user to choose a goal location on the screen before the robot autonomously plans its movements and optimal path to the goal without additional human assistance. Assisted or autonomous navigation has been rising in popularity, and outside of telepresence, this form of navigation is seen in consumer vehicles through autopilot features. A major benefit of autopilot is avoiding crash collisions - Tesla reports that their vehicles without autopilot functions experienced one accident per every 1.2 million miles driven while vehicles with autopilot functions experienced one accident per every 4.19 million miles driven[37]. These autopilot features share the same benefits in the context of telepresence robots, and they are being implemented to facilitate more effective, safer, and hands-off navigation.

Studies on telepresence navigation have focused on obstacle avoidance in indoor spaces by using algorithms like Simultaneous Localization and Mapping (SLAM) to map environments[38] and algorithms like potential fields to avoid collisions[39]. These methods are important to avoid damage to the robot and its surroundings. However, while much research explores potential mechanisms for creating better autonomous navigation in telepresence robots, few studies focus on the implications on cognitive workload associated with the different modes

of navigation. One such study conducted by[40] focused on spatial navigation in telepresence robots and examined cognitive load in the context of assisted and unassisted teleoperation. Subjects participated in an obstacle course using both assisted and non-assisted teleoperation, with performance measured by time on task and number of errors made during both conditions. Takayama et al. found that participants in the assisted teleoperation condition performed better in avoiding more obstacles than in the non-assisted teleoperation condition. Even though the study conducted with Takayama et al. touches on how cognitive load interacts with assisted and non-assisted navigation, they only implemented the NASA-TLX assessment and an adjective questionnaire to assess cognitive load.

The present study draws from the concepts present in the current literature pertaining to the impact on cognitive load while operating telepresence robots but is designed to further understand cognitive load in the context of autonomous and manual navigation. We compared subject performance on a secondary memory task when operating a telepresence robot with autonomous or manual navigation. Our findings indicate that adding autonomous features, such as path planning and navigation, to telepresence robots can decrease cognitive load, improve performance on memory tasks, and aid in faster task completion times.

3.3 Method

3.3.1 Participants and Recruitment

The participants included n = 22 (11 males and 11 females) individuals between the ages of 18 and 48. Participants were recruited through digital flyers and physical flyers posted on the University of California, Irvine (UCI) campus. All experimental procedures were approved by the Institutional Review Board (IRB) at UCI. Due to the Covid-19 pandemic, participants accessed the Toyota Human Support Robot (HSR) remotely by logging onto the laptop

containing the robot's user interface via the TeamViewer remote desktop software. There were no in-person experimental sessions. Participants used both autonomous and manual navigation to navigate the space. The participants were randomly assigned to either start with autonomous or manual navigation in the first experimental session and then switched to the other navigation condition in the second experimental session. No order effects were observed. Participants chose their compensation method at the end of the study in the form of a $25 gift card, or 2 course credits granted by the UCI Human Subjects Lab Pool.

3.3.2 Telepresence Robot

The Toyota Human Support Robot (HSR) and its associated Robot Operating System (ROS) packages were used for this study. The Toyota HSR was designed to help with household tasks for the elderly and impaired[41 42 43]. Using the Toyota HSR, we mapped the floor of a university building using an existing SLAM algorithm[44]. ROS libraries provided with the Toyota HSR were used to plan paths. The robot's Lidar and camera detected obstacles and the path planner changed its route appropriately. The robot's onboard computer handled the mapping and navigation. A laptop computer mounted on the back of the Toyota HSR contained the user interface (Figure 3.1) that mapped button clicks to robot movements of the body and head. The camera feed from the robot was also visible on the Graphical User Interface (GUI). The software for controlling the robot was written in Python.

Participants accessed the Toyota HSR by logging onto the laptop via the TeamViewer remote desktop software – a remote access and remote-control online program. This allowed them to see the GUI and have access to the laptop's mouse and keyboard. The experimenter held an additional laptop, which the participant connected to via Zoom, so that the participant could communicate with the experimenter. The participant shared their screen, which allowed the experimenter to see the participant's view.

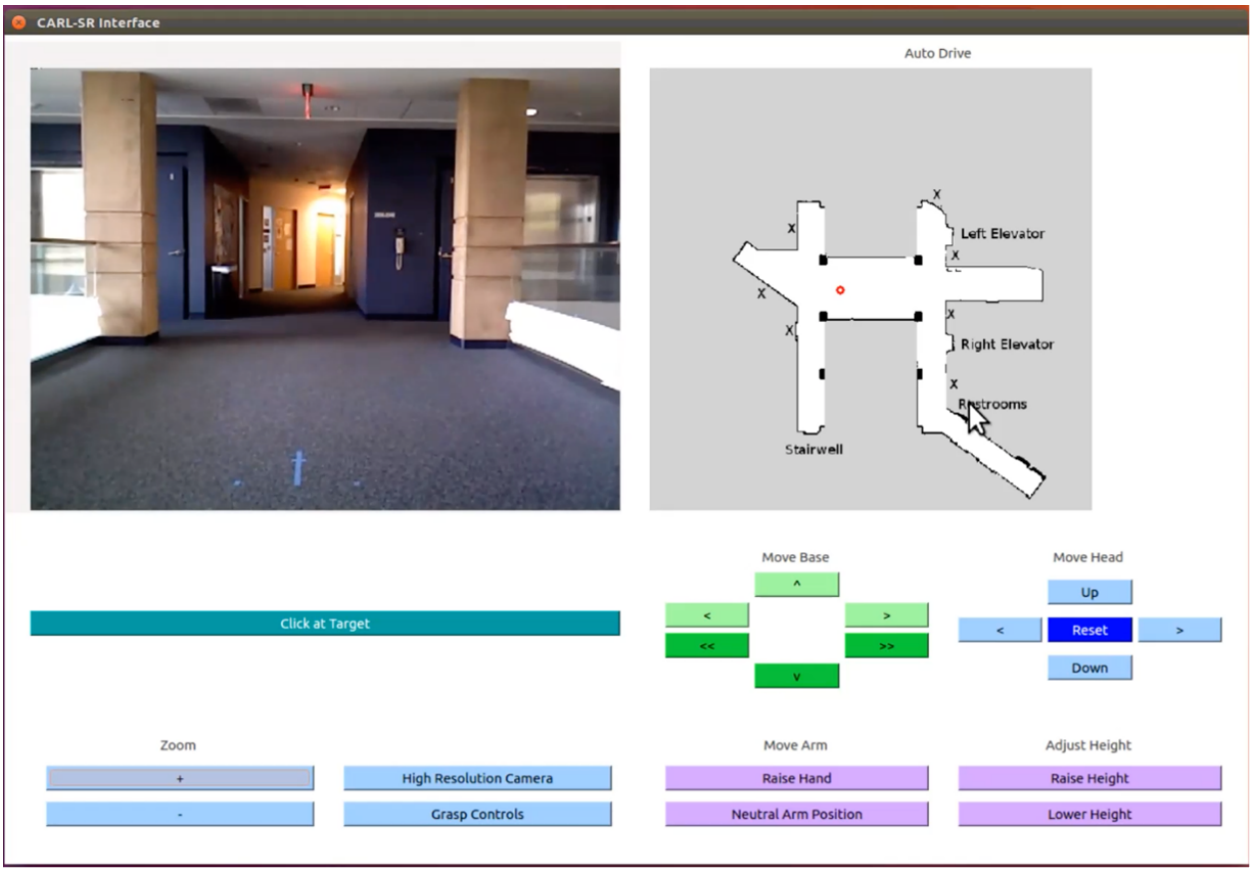

Figure 3.1: In the manual navigation condition, participants clicked on the green and blue buttons under the "Move Base" and "Move Head" functions to move the Toyota HSR. In the autonomous navigation condition, participants clicked on the white space on the map (top right) and the Toyota-HSR planned a path to the desired location. The desired location was marked with a green circle while the real-time location of the robot was marked with a red circle. The "X" marks on the map indicate where each cartoon face is located. The top left image showed the Toyota-HSR camera and what the robot saw in real time.

3.3.3 Experimental Design

This study measured the participant's cognitive load and performance on a search task when using autonomous and manual navigation to operate the Toyota HSR. This was a within-subjects design and subjects participated in both the autonomous and manual navigation conditions counter-balanced such that half of the participants experienced the autonomous navigation first and then the manual navigation condition, and half of the participants experienced the manual navigation condition prior to the autonomous navigation. The materials used were the Toyota HSR, the TeamViewer software, Zoom, 20 cartoon faces of people from an online database, and Google Forms. The experimental study took roughly two hours to complete.

The experimental study was separated into two main tasks and took roughly two hours to complete. In the primary task, participants were instructed to use the Toyota HSR in a scavenger hunt to find seven cartoon faces placed on the walls where "X" was marked on the map (Figure 3.1). There were two experimental sessions: one session completing the task with one set of cartoon faces using autonomous navigation and one session with another set of cartoon faces using manual navigation. Participants were also randomly assigned to begin with either Set A or Set B faces in the first experimental session and then switched to the other set of faces in the second experimental session. No order effects were observed. Participant's time during each experimental session was recorded, and the participants were encouraged to speak their process of navigating through the space out loud. When they finished with the scavenger hunt, they were presented with a secondary task where they were instructed to match the faces to the location where they had found them. The participants were not aware that they had a secondary task prior to starting the experimental session. This was done to obtain the most accurate results for the secondary memory task. Participants then were instructed to fill out the NASA-TLX, Presence, and Positive and Negative Affect Schedule (PANAS) assessments to measure their cognitive load

and user experience during that experimental session. After finishing these questionnaires, they moved onto the next experimental session where the process was repeated, this time using the other mode of navigation and another set of faces. After both experimental sessions were complete, participants filled out a post-study questionnaire about their experience using the Toyota-HSR.

3.3.4 Procedures

Before beginning the experiment, participants filled out a pre-study questionnaire asking for video game experience and telepresence robot experience. In addition, participants completed a locus of control questionnaire and mental rotation assessment task. After participants finished the pre-study questionnaire, they were given instructions on how to connect to TeamViewer and how to gain access to the Toyota-HSR user interface by remotely controlling the GUI for the robot (see Figure 3.1). Once connected, they underwent a short training session where the participants learned how to navigate the Toyota-HSR using both autonomous and manual controls. In the autonomous navigation condition, participants were instructed to click on the white space on the map causing the Toyota-HSR to move to that location. The robot planned a path to the desired location while avoiding obstacles. In the manual navigation condition, participants were instructed to use the green and blue buttons under the "Move Base" and "Move Head" functions to move the body and the head of the Toyota-HSR (Figure 3.1, bottom right). They could also use the keypad on their computer to move the robot (to move the head: "A" - move the head to the left, "D" - move the head to the right, "W" - move the head up, "S" - move the head down; to move the body: up arrow key - move the base forward, left arrow key - rotate the base to the left, right arrow key - rotate the base to the right). As soon as they felt comfortable with the controls, the experimenter set up one set of seven cartoon faces at each one of the "X" locations on the map which can be seen in the user interface of TeamViewer (Figure 3.1).

After the cartoon faces were set up, participants started the first experimental session where they would use the controls of their assigned navigation condition (autonomous or manual) to move to each "X" on the map. They were instructed to use the controls for that navigation condition to find each cartoon face at each "X" location on the map. In the autonomous navigation condition, they could only click on the map to move the body of the robot, but once they were at the desired location, they could move the robot's head using the blue buttons to look around their environment. In the manual navigation condition, they could only use the blue and green buttons or the keyboard to move the robot. They would use their assigned navigation controls to locate the face and to bring the face into "speaking distance" (close enough to see the features of the face through the Toyota-HSR camera) at each of the seven "X" locations on the map (Figure 3.2). The participants were also instructed to speak aloud, that is, "speak through" their experience of navigating the robot. For instance, if they wanted to navigate to the top right "X" on the map in the manual navigation condition, they could say something like "Now I'm trying to navigate to the top right 'X' on the map. I'm going to keep pressing on the up-arrow key until I reach the end of the hallway. Then I'm going to use the buttons on the interface to rotate the base of the robot until I'm facing the right. Then, I'll continue to press the up-arrow key to move in that direction." They were also encouraged to voice how they were feeling at any time during the experimental session and any issues they were having.

After completing the main experimental task (the scavenger hunt), the participants returned to the experimental session Google Forms link where they completed a secondary task. This secondary task was an image memory task where participants were given a set of 10 cartoon faces and had to match the faces to their locations during the scavenger hunt. Seven of the cartoon faces were target faces used in the scavenger hunt, and three of the cartoon faces were distractors that shared similar features to the target faces, but not used in the scavenger hunt. They had to match the correct face to the correct location, and respond "none" if the face was not present in the scavenger hunt. The map provided in the secondary task was a

Figure 3.2: Participants were instructed to bring the cartoon face close enough to the camera to see the features of the face. The space of the robot to the face on the door represents the desired distance for each face. The experimenter followed behind the Toyota-HSR during the experimental sessions to check that the robot would not crash into any walls or obstacles.

replication of the map on the user interface of the TeamViewer screen that the participants used to move around the space (Figure 3.3). Following the secondary task, participants then completed the NASA-TLX, Presence, and PANAS assessments. These assessments were to measure their cognitive load and user experience of the Toyota-HSR during that experimental session.

After the participants finished these assessments, they moved onto the second experimental session where they repeated the process but used a different type of navigation and a different set of faces. Following the second experimental session, participants filled out a post-study questionnaire on the same Google Forms link which asked questions regarding their experience with the robot. These questions were as follows:

- Was it easy to use the robot?
- Did you use the keypad or the interface more to navigate the robot?
- What was the best feature of the robot? Why?
- What was the worst feature of the robot? Why?
- What was a function you wish the robot had?
- Would you use this robot again?

The participants also had a free response space where they could share any additional thoughts regarding their overall experience using the Toyota HSR.

Given the circumstances of the pandemic, all subjects participated remotely through Zoom and accessed the Toyota-HSR through TeamViewer. Except for the experimenters, no one was allowed to enter into the lab space when participating in the study.

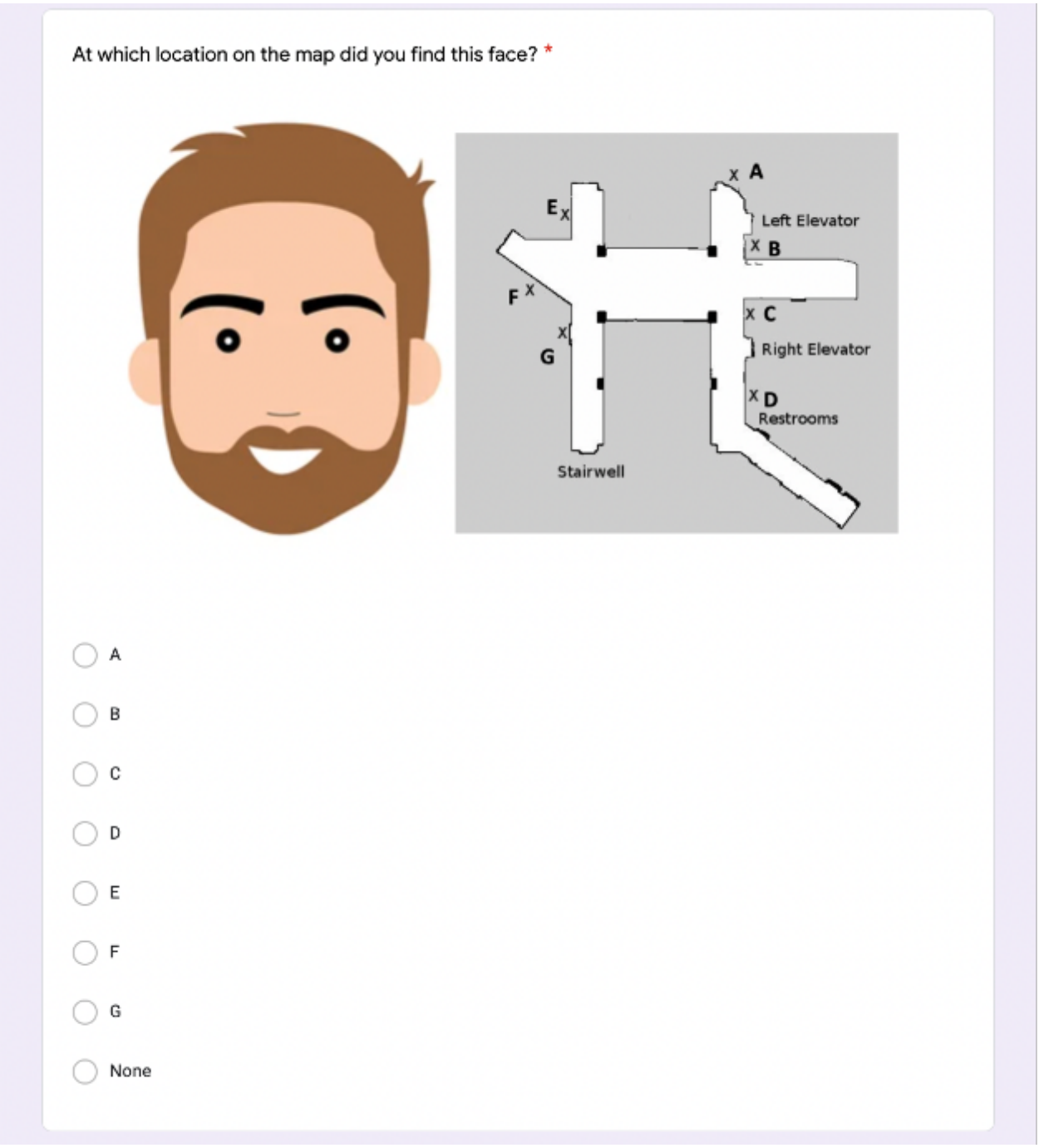

Figure 3.3: Example of what the participants see on the secondary task. They are instructed to match the face to the correct location on the map, and if the face was not present during the scavenger hunt, they would mark "none." There was a total of 10 questions with the same format, seven target faces and three distractor faces.

3.4 Results

3.4.1 Cognitive Load Assessments

Participants performed better on the secondary task during the autonomous navigation condition than the manual navigation condition. The number of correct responses on the secondary task were greater in the autonomous navigation condition than in the manual navigation condition (Figure 3.4). A Wilcoxon signed-rank test indicated that the median of the correct responses for the autonomous navigation condition, Mdn = 7, was significantly higher than in the manual navigation condition, Mdn = 4 with a p-value of less than 0.05. These results indicated that autonomous navigation may reduce the cognitive load on memory compared to manual navigation.

We found that participants were more efficient when navigating the Toyota-HSR autonomously. The time to complete the experimental sessions were compared between the autonomous navigation condition and the manual navigation condition (Figure 3.5). A Wilcoxon signed-rank test indicated that the median of the time taken for the experiential scavenger hunt task for the autonomous navigation condition, Mdn = 446 seconds, was significantly lower than in the manual navigation condition, Mdn = 594 seconds with a p-value of less than 0.0001. These results indicated that participants were faster in completing the experimental task in the autonomous navigation condition and this may contribute to a decrease in cognitive load compared to manual navigation.

3.4.2 Cognitive Load and User Experience Assessments

In addition to the correct responses and time taken in the experimental task, reduced cognitive load during autonomous operation was indicated through participant responses on the

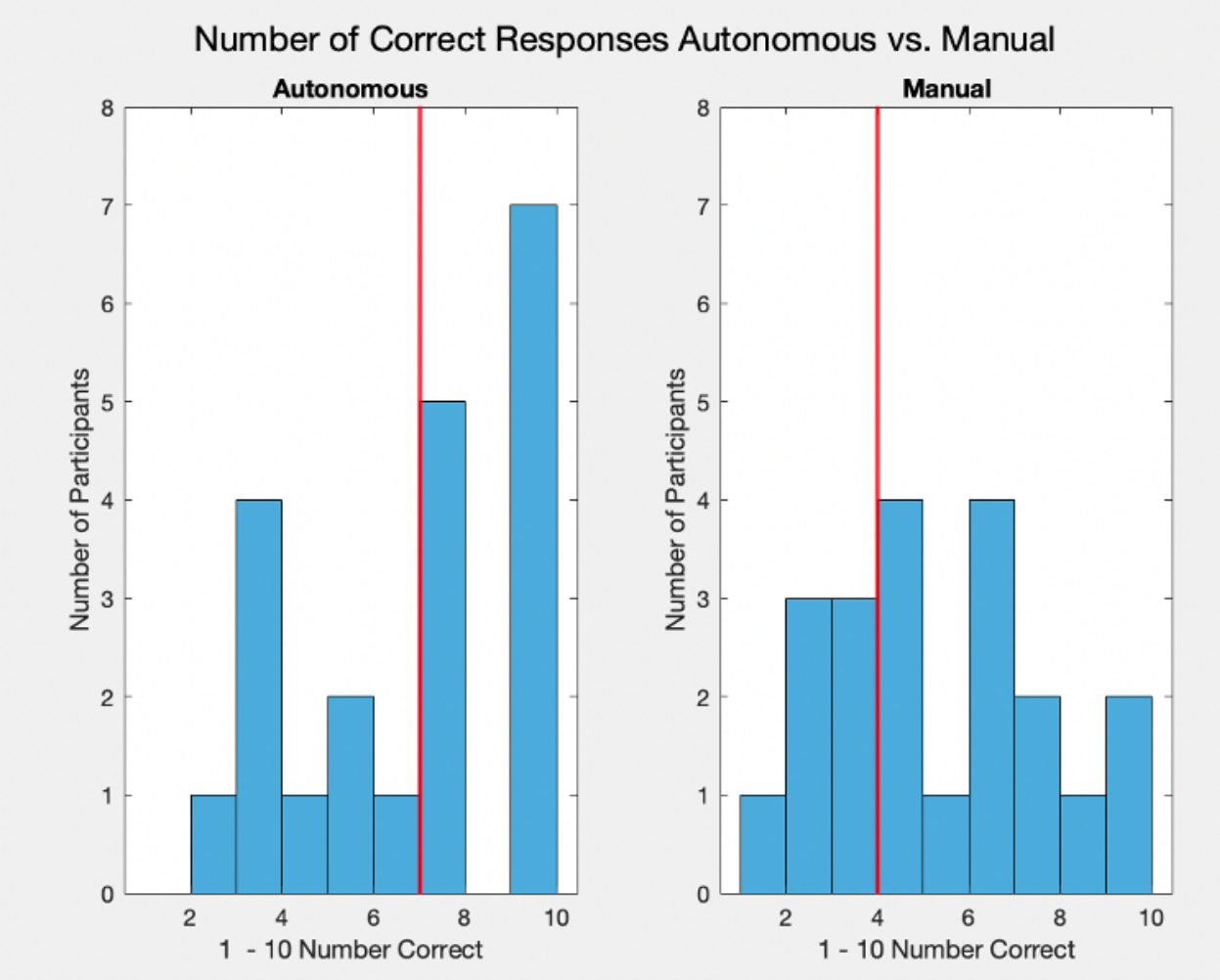

Figure 3.4: Histogram of the number of correct responses in the autonomous navigation condition and the manual navigation condition. The autonomous navigation condition had a higher median of 7 compared to the manual navigation condition which had a median of 4 (p-value ¡ 0.05 using a Wilcoxon signed-rank test).

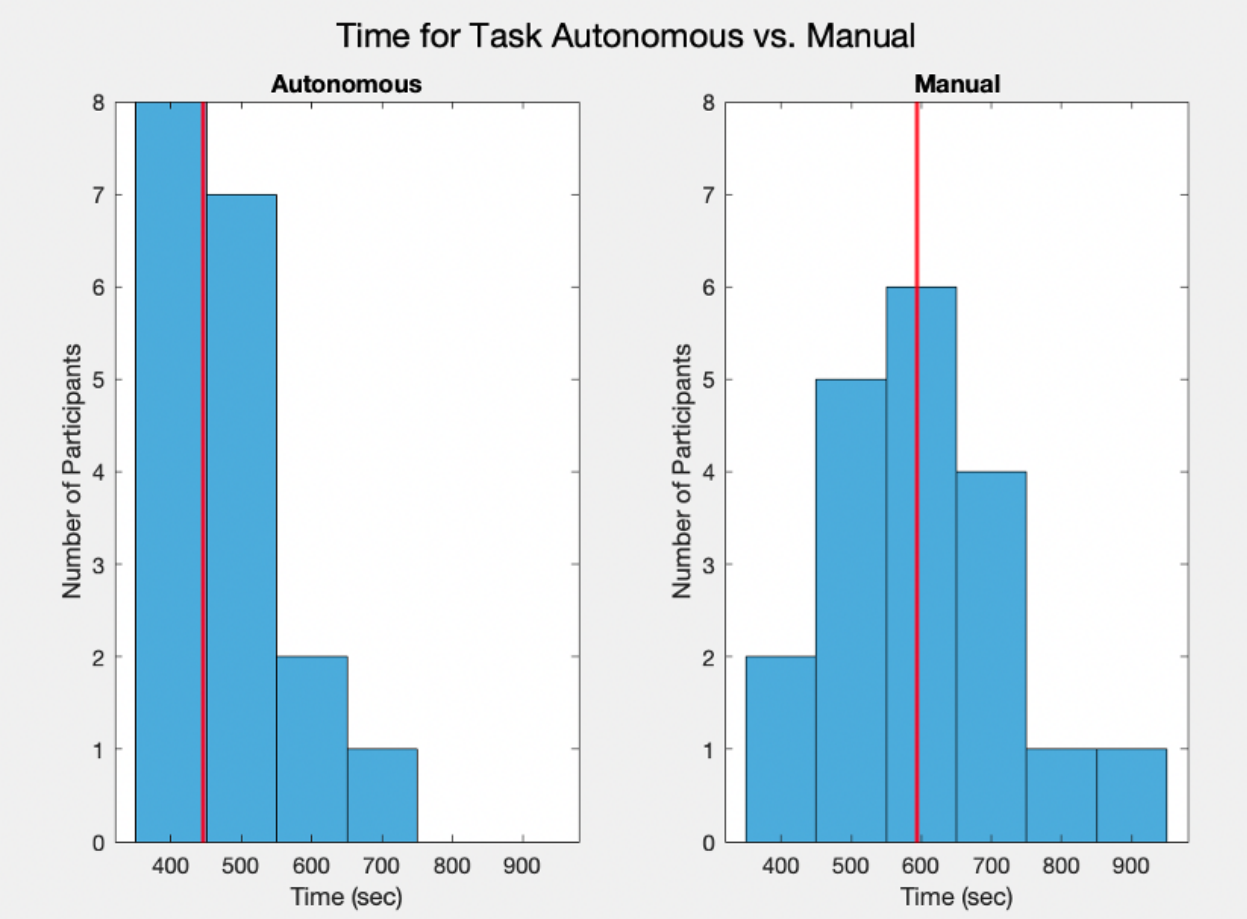

Figure 3.5: Histogram of the time taken to complete the experimental task for the autonomous navigation condition and the manual navigation condition. The median time was shorter for the autonomous navigation condition which was 446 seconds compared to the median time of the manual navigation condition which was 594 seconds (p-value ¡ 0.0001 using a Wilcoxon signed-rank test).

NASA-TLX assessment which was given immediately after each experimental session. The scores on the NASA-TLX assessment of the experimental sessions were compared between the autonomous navigation condition and the manual navigation conditions (Figure 3.6). A Wilcoxon signed-rank test indicated that the median of the NASA-TLX scores for the autonomous navigation condition, Mdn = 13, was significantly lower than in the manual navigation condition, Mdn = 17 with a p-value of less than 0.0005.

3.4.3 User Experience Assessment

In terms of emotional affect while navigating the Toyota HSR, the arousal and affect levels were lower in the autonomous navigation condition than in the manual navigation condition, indicating that subjects may have experienced less distractions that interfered with their task performance. The positive PANAS score was lower when using autonomous features suggesting that participants experienced more positive emotions when operating the Toyota HSR manually. The scores for the positive PANAS of the experimental sessions were compared between the autonomous navigation condition and the manual navigation condition (Figure 3.7). A Wilcoxon signed-rank test indicated that the median of the positive PANAS scores for the autonomous navigation condition, Mdn = 12, was significantly lower than in the manual navigation condition, Mdn = 19 with a p-value of less than 0.005.

The negative PANAS score was also lower when using autonomous features. Figure 3.8 shows a comparison of the negative PANAS of the experimental sessions between the autonomous navigation condition and the manual navigation condition. A Wilcoxon signed-rank test indicated that the median of the negative PANAS scores for the autonomous navigation condition, Mdn = 5, was significantly lower than in the manual navigation condition, Mdn = 6 with a p-value of less than 0.05.

Taken together, less emotional affect (both positive and negative as measured by PANAS) in

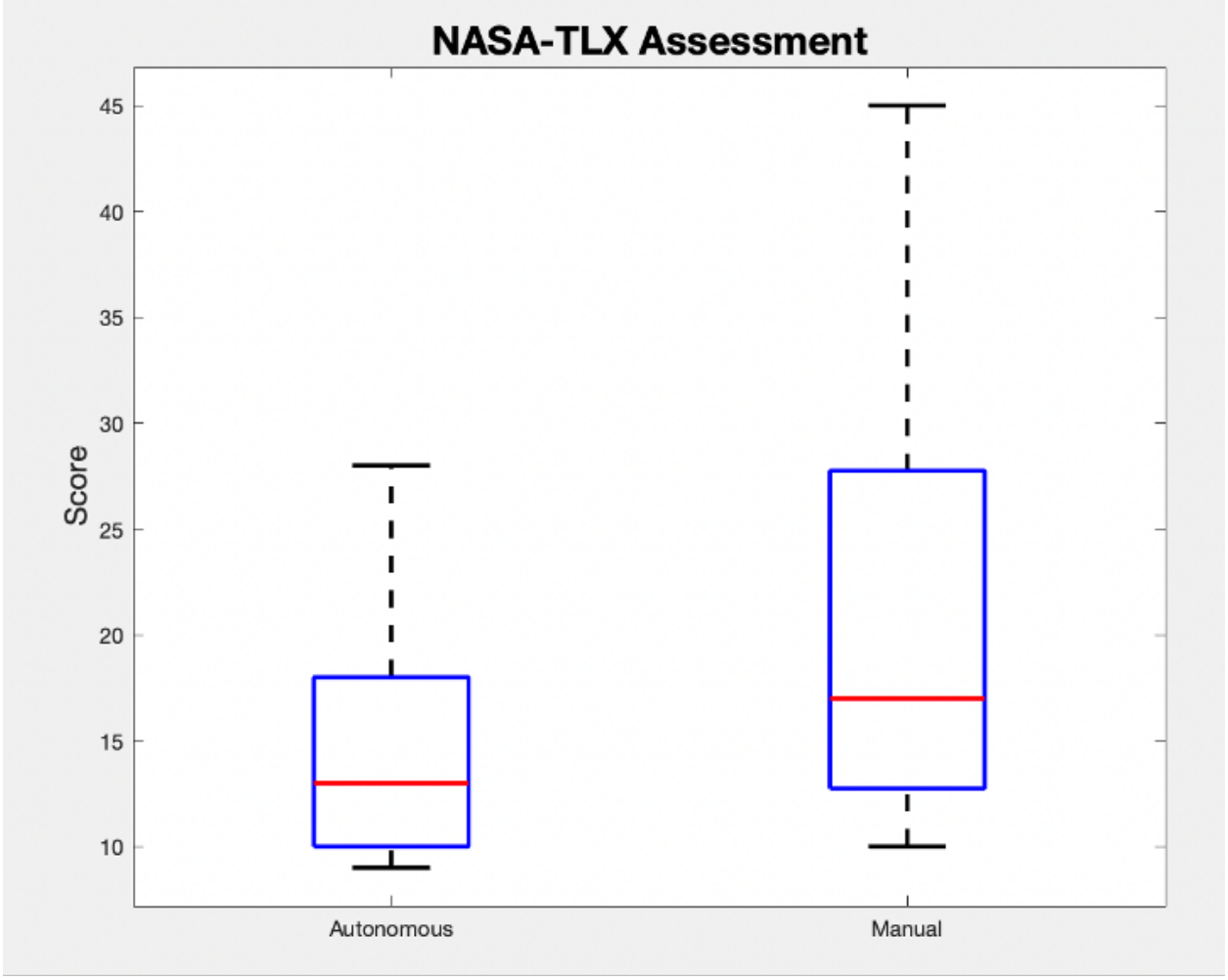

Figure 3.6: Boxplot of the participant responses on the NASA-TLX assessment. The NASA-TLX median scores were 13 and 17 for autonomous and manual navigation respectively (p-value ¡ 0.0005 using a Wilcoxon signed-rank test). On each box, the red central mark indicates the median, and the blue bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers.

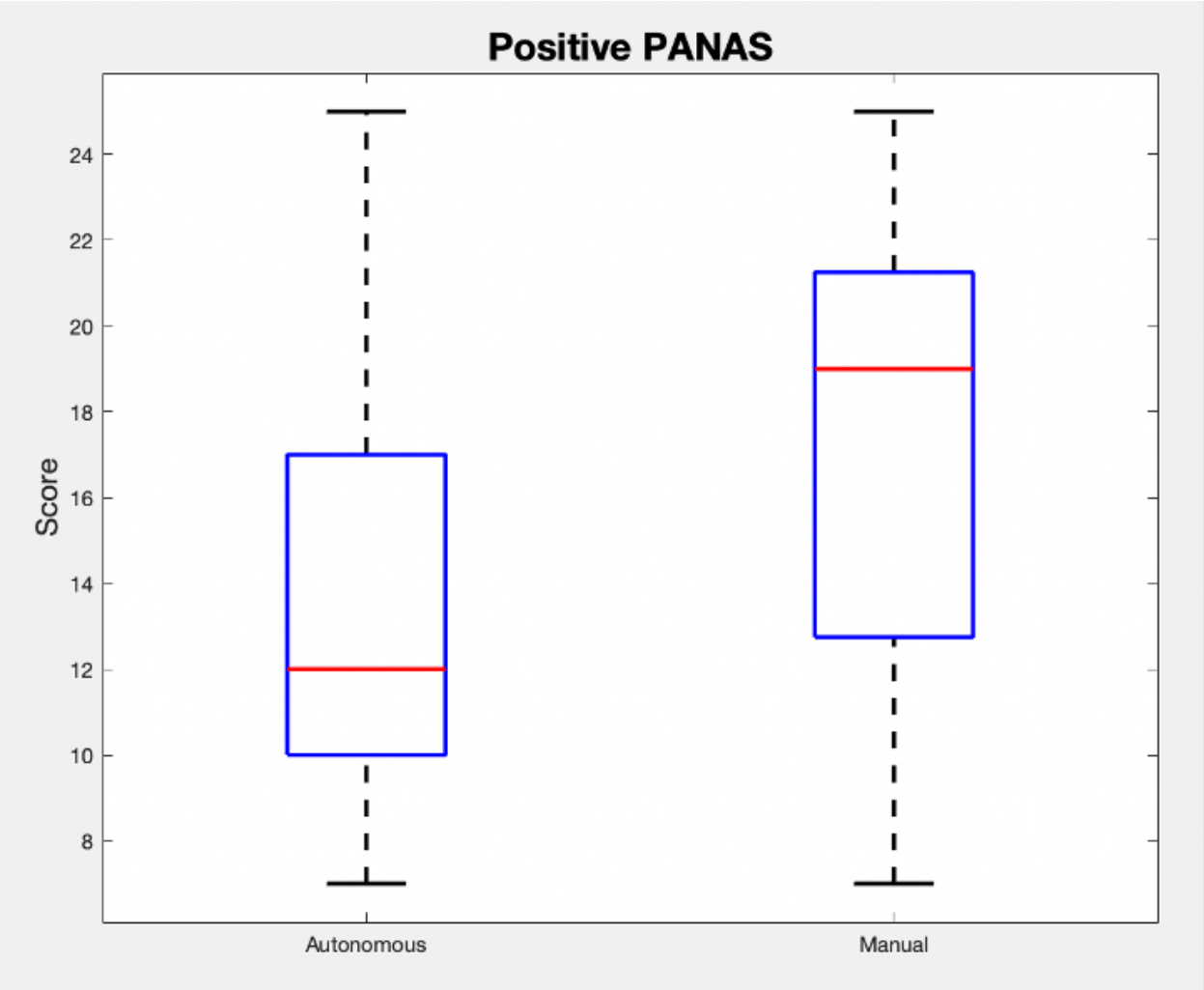

Figure 3.7: Boxplot of the positive PANAS scores for participants in the autonomous and manual navigation condition. The median positive PANAS scores were 12 and 19 for autonomous and manual navigation respectively (p-value ¡ 0.005 using a Wilcoxon signed-rank test).

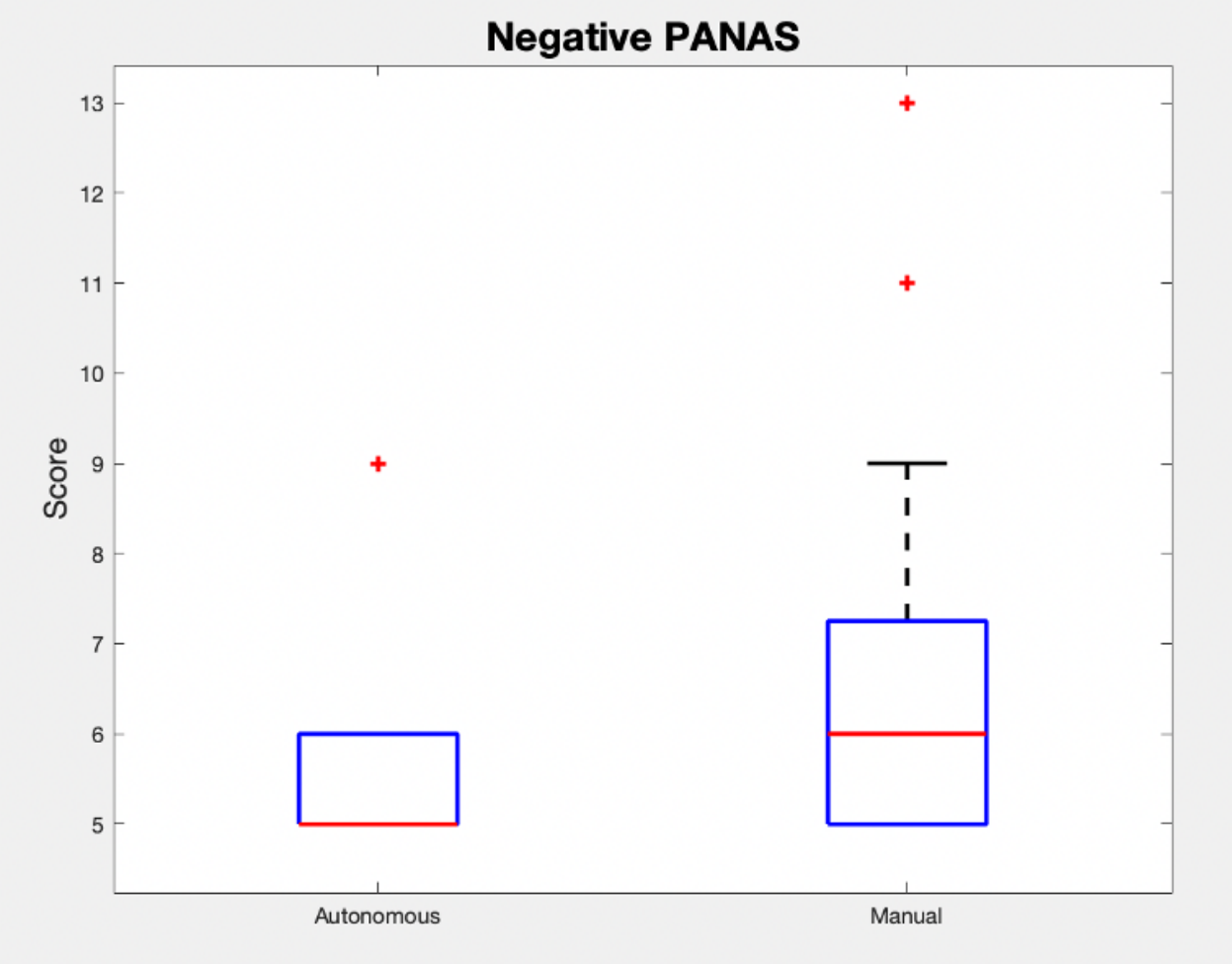

Figure 3.8: Boxplot of the negative PANAS scores for participants in the autonomous and manual navigation condition. The median negative PANAS scores were 5 and 6 for autonomous and manual navigation respectively (p-value ¡ 0.05 using a Wilcoxon signed-rank test). The red plus signs in the boxplot indicate outliers.

the autonomous navigation, may lead to less distractions that compete with attention and memory systems, which may in turn result in a cognitive load reduction.

3.4.4 Presence Assessment

There was no significant difference in the feelings of presence between the manual and autonomous navigation. Figure 3.9 shows the scores for the Presence assessment of the experimental sessions between the autonomous navigation and manual navigation conditions. A Wilcoxon signed-rank test indicated that the median of the Presence scores for the autonomous navigation condition, Mdn = 104, was lower than in the manual navigation condition, Mdn = 106 with a p-value greater than 0.10.

3.4.5 Comparative Analysis

There were no significant correlations between the number correct on the secondary task to video game experience, locus of control score, mental rotation score, NASA-TLX score, presence score, positive PANAS score, and negative PANAS score (Pearson's correlation coefficient; p ¿ 0.05). There were also no significant correlations between the time spent on the experimental sessions and the mental rotation score, NASA-TLX score, Presence score, positive PANAS score, and negative PANAS score (Pearson's correlation coefficient; p ¿ 0.05).

3.5 Discussion

The aim of this study was to test whether using autonomous navigation in telepresence robots could decrease cognitive load. The ability to autonomously navigate to locations

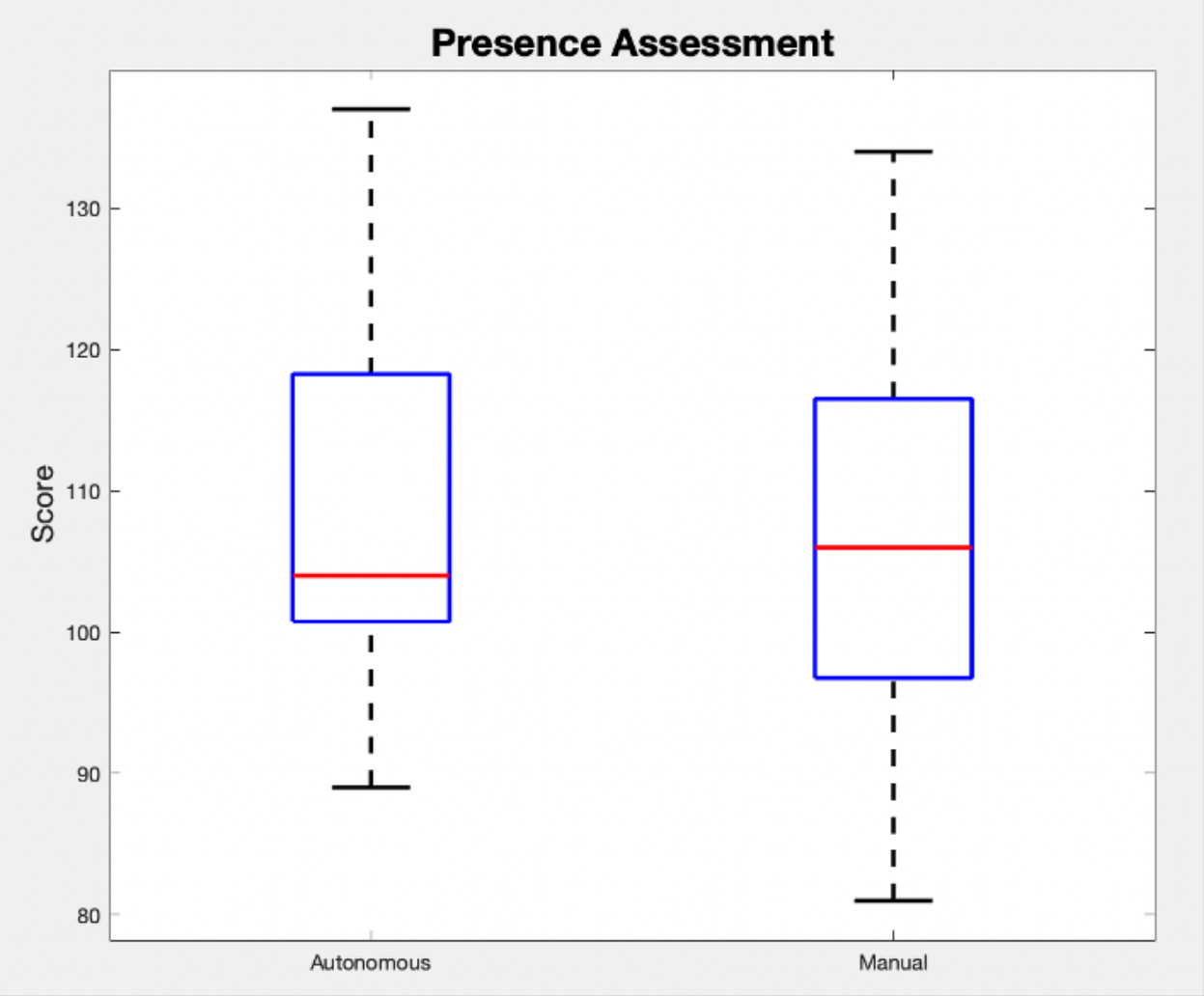

Figure 3.9: Boxplot of the Presence scores for participants in the autonomous and manual navigation condition. The median Presence scores were 104 and 106 for autonomous and manual navigation respectively. The p-value was greater than 0.10 and was not significant.

and around obstacles decreases the need to attend to these non-essential stimuli, which in turn may facilitate memory processes. As the recent pandemic has revealed, the demand for and usefulness of telepresence technology is growing as this technology allows home-bound individuals to be present even in remote settings[30][31]. Decreasing the cognitive load required to operate telepresence robots and increasing the ease of use may encourage the integration of these technologies into more classrooms and workplaces. Autonomous navigation in these telepresence robots may prove to be beneficial in school, work, and medical environments where individuals are afforded more cognitive space for other tasks.

In the present study, our main finding was that when participants were assigned to the autonomous navigation condition, cognitive load was reduced. Under the autonomous navigation condition, participants: 1) Experienced shorter experimental sessions implying more efficiency in locating target objects, 2) Scored higher on the secondary task, which required remembering the location and recognition of faces, and 3) Scored lower on the NASA-TLX assessment, which indicated less workload during the experimental session. All three findings support our hypothesis that autonomous navigation is beneficial in reducing cognitive load as cognitive load is assessed by examining time spent on a task, performance on a secondary task, and the NASA-TLX assessment. Therefore, we conclude that relieving the user of constant monitoring of the movement of a telepresence robot can significantly improve task performance and user experience.

3.5.1 Cognitive Load

In cognitive tasks, the time spent on a task can act as a measure of cognitive load[45]. Subjects take less time to complete the task when it is easier and more time when it is more difficult. In our study, the autonomous navigation feature allowed the participants to spend less time to complete the task since the path-planning decisions were assigned to the robot –

having this feature may have made the experimental task easier to complete and may have reduced the amount of cognitive processing necessary to complete the task. Additionally, since it took less time to complete the experimental task, participants may not have had to retain the cartoon faces in their working memory for as long which may have resulted in them performing better on the secondary memory task. Performance on secondary tasks, as well as responses from the NASA-TLX assessment offer valuable insight into cognitive workload. In our study, participants experienced better performance on the secondary memory task and lower estimates on the NASA-TLX assessment while operating the Toyota-HSR autonomously. These results supplement and support our hypothesis that cognitive load can be reduced through autonomous navigation.

3.5.2 Emotional Affect and Cognitive Load

When responding about their emotional affect during the experimental task, participants reported less affect in general while operating the Toyota-HSR in the autonomous navigation condition. Both the positive PANAS and the negative PANAS scores were lower in the autonomous navigation condition than in the manual navigation condition. These scores indicate that participants experienced less emotional response when operating the Toyota HSR in the autonomous condition, and these decreased emotions (both positive and negative) present in the autonomous condition may have been a result of the less hands-on navigation approach. In the autonomous controls, the participants simply had to click on the map to navigate the robot whereas the manual controls required them to push a button each time they needed to move the robot. Many participants reported during and after the experimental sessions that they felt that the manual navigation was more engaging and simulated playing a video game. Potential accidents and collisions in the environment during manual operation could also elicit more emotions and feelings from the participants[46]. In our study, there were very few collisions. However, subjects may have feared that a collision was

imminent when operating the robot manually. This could in turn require more engagement and concentration on moving the robot rather than memorizing face locations.

Increased engagement and arousal may come at a cost in the memory task performance. Emotional affect may compete with working memory and may become extraneous cognitive load[47]. Emotions are processed in the same cortical structures as other cognitive functions, which may cause interference[48]. Taken together, these findings support our results as emotion seems to contend for the same resources as cognitive processing. Thus, cognitive load may be impacted when higher emotional responses occur. In our study, participants performed better in the secondary memory task in the autonomous navigation condition where their emotional affect (both positive and negative) were lower than in the manual navigation condition. Participants experienced stronger positive and negative emotional reactions when they had to complete the scavenger hunt task using manual controls which could indicate higher cognitive load.

3.5.3 Related Telepresence Studies

One principal advantage of telepresence robots is the feelings of presence and embodiment that they offer[46]. In our study, participants reported no significant difference in feelings of presence between the autonomous and manual navigation conditions. Whether they were using the manual controls to move the Toyota-HSR themselves or if they were using the autonomous controls to let the Toyota-HSR plan out the route, they did not feel one mode of navigation induced more embodiment in the environment. This indicates that including autonomous navigation controls can serve to improve the usability of telepresence robots without taking away from the feelings of being present in their environment.

Recent telepresence studies have investigated how providing telepresence robots with autonomous functions can affect the user experience[49][46]. Similar to the focus of the present

work, these studies examined how to better integrate telepresence robots into social settings and improve user experience. Batmaz et al. found that autonomic speed control could help simplify navigation and could decrease cognitive load, but the decision to implement this was ultimately up to the user's personal preference. Their findings are similar to our study in the aspect that while participants had reduced cognitive load while the robot operated autonomously, some preferred the manual navigation because they enjoyed personally controlling the robot. However, while Batmaz et al. speculated that automatic speed control would reduce cognitive load, they did not directly test this themselves, and they did not analyze emotional affect while navigating. In the present study, we employed three measures of cognitive load and to further examine participants' emotional affect while operating the robot.

Yang et al. found that personality and presence were strongly conveyed through the telepresence robot, but there were strains on other aspects of interaction like responsibility, dependency, and contribution[46]. Their findings are similar to the findings of our study in that presence was expressed, and participants were more dependent on the experimenter during the manual navigation condition. However, while there were similarities between our results, their study did not measure cognitive load or implement different types of navigation in telepresence robots. Both studies[49][46] did not fully encompass cognitive load, presence, emotional affect, and user experience in terms of autonomous and manual navigation in telepresence robots. The present study supplements their work by looking more specifically at the effects of autonomous navigation on the areas of user experience, presence, emotional affect, and cognitive load. We find that autonomous functions can reduce cognitive load and emotional affect, thus leading to better performance on memory tasks.

3.5.4 Limitations and Improvements

Some limitations of our study include the latency in action from clicking on the GUI to the actual movement of the robot in the manual navigation condition. Many participants reported this as an issue for them in their attempt to navigate around the space and completing the primary task of the scavenger hunt. Another limitation was the camera quality and the size of the camera picture. Many participants commented on the quality of the video feed from the Toyota-HSR, and depending on the time of the day, the sunlight would greatly impact the visibility of the faces on the wall and the surrounding obstacles. Some participants also reported that they wished that the camera was wider angled to encompass more of the surroundings. These sentiments have also been reflected in past telepresence research where the scope and resolution of the camera have presented navigation challenges[50]. The impact of the camera feed quality was apparent when participants would occasionally run into walls or almost collide into the pillars in the environment when operating the Toyota HSR manually.

Unstable internet connection also impacted the flow of the primary experimental task. There were instances where the internet would disconnect, and the experimental session was interrupted. This did not appear to have an impact on the performance of the secondary task. However, it was disorienting for the participant and is a barrier for widespread use of telepresence robots in general. Another limitation was the speed of the Toyota HSR. Increasing the maximum speed in the manual and autonomous navigation may have improved participants' user experience as there were awkward pauses and moments of impatience during the experimental sessions. In the autonomous navigation condition, participants were clear where they wanted to go by clicking in the space, and some became very quiet as they waited for the robot to reach the destination. In the manual navigation condition, participants had to physically press the controls until the robot moved to their desired position, and some asked if the robot could move any faster.

3.5.5 Future Telepresence Recommendations

In addition to autonomous navigation, there may be other autonomous features that may improve experience and performance in the operation of telepresence robots. As Batmaz et al. illustrated in their study, automatic speed controls when approaching barriers or obstacles may simplify navigation and reduce the judgment work in the user's spatial cognition[49]. Autonomous battery recharge may also be another useful function. Currently in the field of autonomous drones, a practical consideration is hot-swapping batteries[51]. Having this function would enable users to continuously operate the telepresence robots instead downtime as their batteries recharge. These supplementary autonomous functions may serve to increase the usability and functionality of telepresence robots in real life settings.

Furthermore, autonomous reaching, grasping and gestures by the robot may also improve user experience. Grasping and holding objects are one of the most natural movements for humans to do in their environments, but robots are still struggling to replicate this[52]. Therefore, more research on how to engineer and implement the ability to autonomously reach and grab objects in telepresence may enable the user to be more interactive with their surroundings. In terms of autonomous gestures, there has been a recent effort in the sphere of self-driving cars to train them to recognize traffic hand signals[53]. This serves to increase safety on the road as hand signals are often used to direct traffic in construction zones and cyclists will often signal to indicate their next movements. Being able to automatically generate, recognize and categorize gestures may prove to be helpful in telepresence robots as well. Identifying certain movement cues like a stationary flat palm to stop may help prevent accidents while operating the robot, and even registering social cues like a handwave to say hello and then having the robot automatically wave back may increase the personability and humanness of the robot in its setting.

Ultimately, implementing autonomous features into telepresence robots has the potential to

decrease cognitive load and increase the quality of the user experience with this technology. Further studies should include various other telepresence robots to test the validity of the results. This would allow us to examine if these results and some of the complications are specifically related to the Toyota-HSR or if they universally impact many teleoperation devices.

Overall, the results of this study suggest that autonomous navigation in telepresence robots like the Toyota-HSR are an effective tool in decreasing cognitive load and improving the overall usability of telepresence robots in day-to-day interactions.

Chapter 4

Benefit of Varying Navigation Strategies in Robot Teams

Note: This chapter was previously published in: *S. A. Mohaddesi, M. Hegarty, E. R. Chrastil, and J. L. Krichmar, "Benefit of Varying Navigation Strategies in Robot Teams", in From Animals to Animats 17, LNAI 14993, Springer Nature, 2025.*

4.1 Abstract

Inspired by recent human studies, this paper investigates the benefits of employing varying navigation strategies in robot teams. We explore how mixed navigation strategies impact task completion time, environment exploration, and overall system effectiveness in multi-robot systems. Experiments were conducted in a simulated rectangular environment using Clearpath PR2 robots and evaluated different navigation strategies observed in humans: 1) Route (RT) knowledge where agents follow a predefined path, 2) Survey (SW) knowledge where agents take the shortest path while avoiding obstacles, 3) Mixed strategies with varying

proportions, such as 40% RT and 60% SW (0.4RT 0.6SW) and 60% RT and 40% SW (0.6RT 0.4SW), and 4) An additional strategy where agents switch from RT to SW 10% of the time (0.9RT 0.1SW). While SW strategy is the most time-efficient, RT strategy covers more of the environment. Mixed strategies offer a balanced trade-off. These findings highlight the advantages of variability in navigation strategies, suggesting benefits in both biological and robotic populations. Additionally, we have observed that human participants in a similar study would start on a route, and then 10% of the time switch to survey. Therefore, we investigate a 90% Route 10% Survey (0.9RT 0.1SW) strategy for individual team members. While a pure Survey strategy is the most efficient regarding time taken and a pure Route strategy covers more of the environment, a mixture of strategies appears to be a beneficial tradeoff between time taken to complete a mission and area coverage. These results highlight the advantages of population variability, suggesting potential benefits in both biological and robotic populations.

4.2 Introduction

Multi-robot navigation plays a vital role in advancing robotic technology, providing a dynamic solution to complex tasks that exceed the capabilities of individual robots. Its significance lies in enhancing efficiency, collaboration, and adaptability across various domains, including manufacturing, search and rescue operations, and autonomous vehicles. Coordinated movement among multiple robots enables them to collectively navigate intricate environments, share information, and optimize routes, addressing challenges that a single robot might find overwhelming.

Research suggests that people employ different types of knowledge to navigate[54]. Survey knowledge contains metric information that includes distances and directions between locations. This knowledge enables flexible path planning resulting in shortcuts or planned

trajectories over never-experienced paths. In contrast, route knowledge consists of sequences of actions associated with places or decision points. Typically, the routes are fixed paths and inflexible.

Boone and colleagues[55] investigated the knowledge people use during navigation in the Dual Solutions Paradigm. In the DSP, participants follow a fixed loop around a virtual environment that has several landmarks along the way. After several laps, they are tested by placing participants at a landmark and telling them to go to another landmark. If they take the fixed loop, they are applying *route* knowledge. If they take a novel shortcut, they are applying *survey* knowledge. When told to take the shortest path, the proportion of participants applying survey knowledge increased. This suggests that many participants had survey knowledge, but they might find it easier to take a learned route.

In a metadata analysis, Krichmar and he showed that the variation observed in human navigation is both between and within subjects. This variability might be explained by taking the cost of traversing an environment into consideration[56]. They found that when told to find a goal, roughly 60% of participants used a route strategy and 40% used a survey strategy. However when told to take the shortest path to a goal, those proportions were reversed (40% route, 60% survey). Furthermore, they found that subjects starting on a route switched to a survey 10% of the time.

In the present paper, we simulate teams of robots to test whether there are advantages to using human-inspired strategies for navigating (Fig. 4.2). Our present work makes the following contributions:

- Simulating human variation has advantages for robot navigation, and possibly for planning algorithms in self-driving vehicles and robotic swarms.
- Incorporating a simple navigation strategy inspired by human subjects, rather than finding an optimal solution, makes multi-agent systems easier to scale.

- A mix of route and survey strategies leads to a tradeoff between time to find goals and more exploration of an environment that may be advantageous for biological and robotic populations.
- In human studies, it is technically challenging to monitor multiple participants navigating at the same time. Simulating a human-sized environment with navigating robots can overcome this limitation.

To better assess how these findings transfer to the real-world, we used a physical robot simulation with human sized robots and local sensing. In the following sections, background and methods are described in further detail.

4.3 Related Work

Several studies have explored the benefits of varying navigation strategies in teams of robots, addressing various aspects of multi-agent path planning and cooperative behavior. Most of the related work falls into two categories: 1) Multi-Agent Path Finding (MAPF), and 2) Heterogeneous swarm navigation.

4.3.1 Multi-Agent Path Finding (MAPF)

In MAPF, the goal is to plan collision-free paths for many agents to reduce mission duration and maximize team productivity. An example is the Conflict Based Search (CBS) algorithm which addresses the challenge of finding optimal paths in multi-agent scenarios by considering conflicts among agents and employing efficient exploration techniques[57]. Another example is the branch-and-cut-and-price (BCP) algorithm that incorporates a shortest path pricing problem for finding paths for every agent independently and constraints for resolving

conflicts[58,59]. Both BCP and CBS are optimal but because of computational complexity, they don't scale well to a large number of agents[60]. Sub 1.5 MAPF algorithm on grids, optimizing time with a 1.5x longest-to-shortest path ratio constraint. This is a sub-optimal solution but it scales better[61]. Other algorithms scale well, by applying simple movement rules but are far from optimal[62,63]. MAPF-LNS is a hybrid approach that first creates a fast planning solution and then optimizes with a heuristic called large neighborhood search[64]. In the above cases, the solution is applied once for all agents. But a more realistic situation is a warehouse in which robots need to plan efficient, collision-free paths for long duration. The Rolling-Horizon Collision Resolution (RHCR) approach addressed the issue of lifelong multi-agent pathfinding in large-scale warehouses by applying their MAPF algorithm over different window periods[64].

This paper acknowledges the inherent limitations often found in human decision-making processes, in contrast to an emphasis on optimal solutions. While conflict resolution among robots remains a necessary aspect, it does not stand as the central objective within this paper's scope. However, future versions could take advantage of conflict resolution policies in the work discussed above. Unlike most MAPF algorithms, a centralized planner is absent in this work. Instead, each robot independently charts its path, driven by individual objectives, and limited knowledge of the other robot's state and intention.

4.3.2 Heterogeneous Swarm Navigation

Swarm robotics typically assumes that agents can communicate or interact with others in the vicinity. Heterogeneous teams of agents have been used to solve a wide range of problems[65–67]. In a heterogeneous robotic swarm, certain tasks can be solved efficiently through cooperation and functional specialization[68]. The agents or robots have different shapes or capabilities (e.g., different sensors or different locomotion). Unlike MAPF, the planning is

decentralized. In a typical navigating swarm, there might be leaders and followers and the task is to find a goal while circumventing obstacles and preventing collisions[69,70]. Because of the locality requirement, the swarm often resembles a flock of birds.

Although there are similarities to our approach, these swarms by definition stay together. Furthermore, the majority of the prior work in swarm navigation assumes a 'bird's eye' view, rather than local sensing that would be required in many field operations. We are interested in heterogeneous foraging, where the robots are independent, decentralized and use different strategies.

4.4 Methods

4.4.1 Path planning algorithms

We simulated the Dual Solutions Paradigm (DSP) developed to study human behavior with teams of 1, 3, and 5 robots navigating the environment given in Fig. 4.1. which had 14 landmarks. To simulate survey knowledge, any path planner that finds the shortest path between a start and destination would be sufficient[71]. The present paper uses a spiking wavefront propagation algorithm, which has been described in[72], to calculate the survey paths. The spikewave algorithm takes into consideration the cost of traversal, which is encoded as a propagation delay. Difficult to traverse regions have long delays and impassable areas have very long delays (Fig. 4.1(c)). It is important to note that every path generated by spikewave is a sequence of straight line segments, meaning that there are no curves in generated paths. The yaw needed for the robot to face towards and move forward along the line segment is then determined. In order to simulate route knowledge, the robots followed a fixed path that took them past all the goal locations (Fig. 4.1(a)). Specifically, route knowledge was simulated with spikewave by giving the robot closely spaced waypoints to

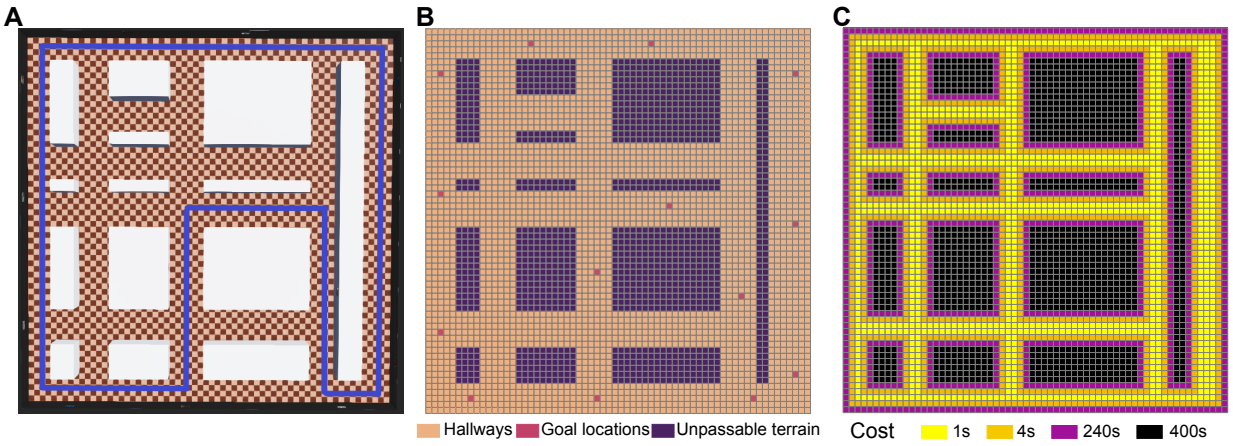

Figure 4.1: (**A**) A bird's eye view of the map, The blue line indicates the specified route (**B**) Grid map of the environment. Purple cells show impassable terrain locations, goal locations are indicated in red, and beige cells indicate the floor of the aisles. (**C**) Heatmap depicting traversal delay costs measured in seconds for each cell on the map. The delay increases as cells approach the walls. This map serves as a universal guide utilized by all robots.

ensure it stayed on the specified route. As described below, the robots could either use route knowledge, survey knowledge or a mixture of these to find the goal locations.

4.4.2 Simulation Environment

In this experiment, a 3D simulated environment was created using Webots[73], an open-source robot simulator. The environment contains a square-shaped maze, which is made of 4096(64×64) cells each representing a square meter, as shown in Fig. 4.2(a). Up to 5 simulated PR2 robots, provided in the Webots environment, were used to explore the maze (Fig. 4.2(b)). The PR2 robot is a mobile robot with advanced sensors and software that can perform various tasks such as navigation and perception with high accuracy and precision. Its unique design features a mobile base with two wheels and a caster and a robotic arm with seven degrees of freedom. The maze size was scaled such that the robot fit within a cell.

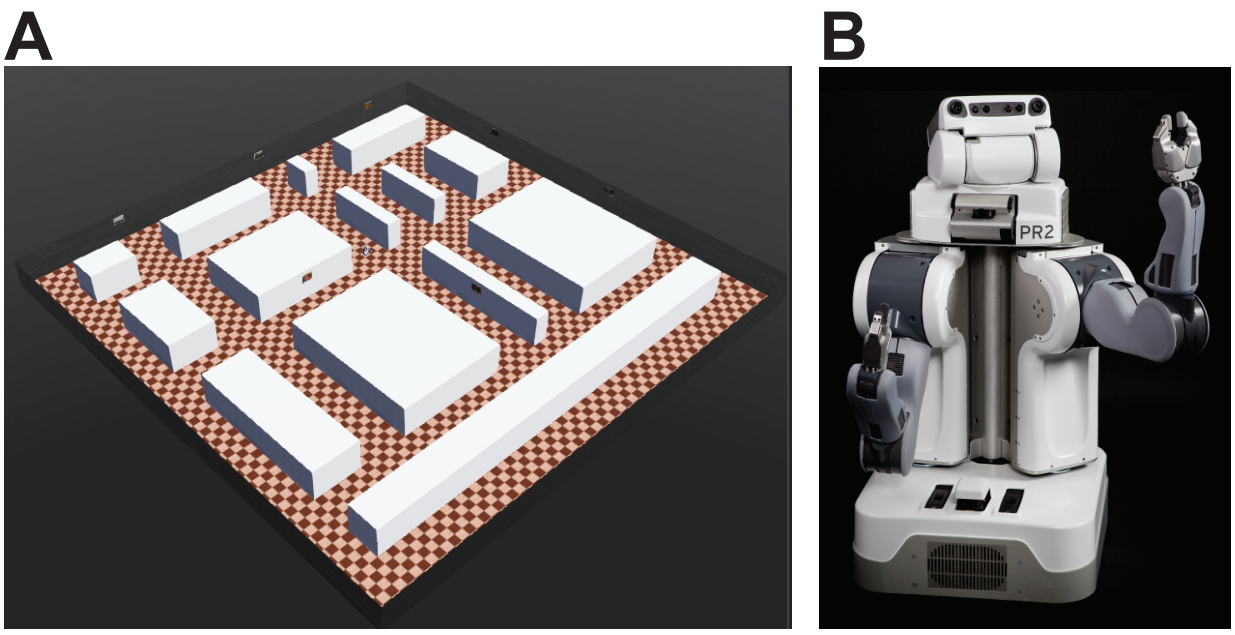

Figure 4.2: (**A**) Overall view of our maze in the Webots environment (**B**) Clearpath's PR2 robot

4.4.3 Experimental Control and Design

The experimental design followed the DSP[55], but with multiple agents simultaneously navigating the environment. Based on the team's size, each robot is assigned a task involving a subset of randomly designated goals chosen randomly from the 14 landmarks (Fig. 4.1(b)). These goals are distributed among the robot team to ensure that each goal is visited by at least one robot.

To conduct the simulation, several challenges need to be addressed. The first was to develop a subroutine that could handle movement between targets. This controller had to be capable of calculating a route between two targets that were free of obstructions and could be traversed by the robots. For the route strategy, an unobstructed path was provided to the robot as in the Dual Solution Paradigm used in human studies[55]. For the survey strategy, the spikewave algorithm uses the map with traversal delays to find the optimal path (Fig. 4.1(c)) as in[72]. An obstacle or environment boundary would have a long delay that would slow the wave

propagation, and a traversable path would have a short delay that results in the wave finding a short, obstacle-free path.

Due to the requirement of simulating physical robots in the present work, several challenges needed to be addressed. One challenge is that sometimes agents come so close to colliding with a wall. The PR2's base Lidar, with its scanning range of 270 degrees, was used to measure the distance to the wall. If this length is less than 60cm, a subroutine is triggered to take the robot back to its original path. The spikewave algorithm then computes a new path to the goal. Another challenge was that a rule had to be devised and implemented to handle bottleneck conflicts and determine which robot has the right of way. Fig. 4.3 is an example of the paths of 3 robots. For the purposes of this experiment, an ID was assigned to each robot (varying from 1 to 5). Two robots are in conflict when the distance between them is less than 6 meters. When conflicts arise (Fig. 4.4(a)), the robot with the higher ID pauses, allowing the other robot to move a distance of at least 12 meters away. This distance was empirically determined to effectively reduce the potential for subsequent collisions involving the PR2 robots. In cases where our defined route strategy is applicable, prohibiting rerouting, conflict resolution favors the robot that holds a positional lead on the route. Finally, since the problem of navigation strategies is similar to real-life scenarios, a single management unit could not be used to handle all conflicts between the robots. Instead, all decisions to control the disputes had to be either through predetermined rules or message passing among the agents. When a collision occurs, a signal is sent to all other robots within 25 meters of the collision point. The robots that receive this signal increase the delay of the cells around the collision point on the delay matrix. The delay is increased by 500 seconds for a 3×3 square block around the collision point. Each agent then runs the spikewave algorithm to reroute to the temporary goal using the new matrix. This helps robots avoid the collision point and prevents a third robot from interfering with two conflicting robots (Fig. 4.4(b)).

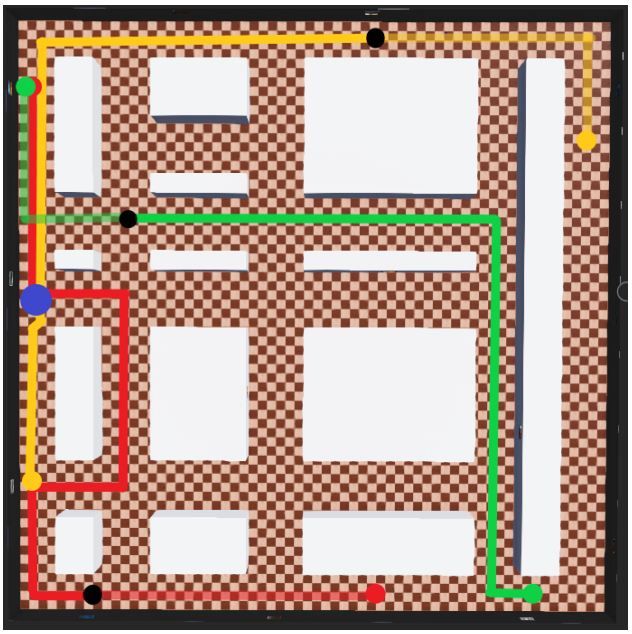

Figure 4.3: Example of 3 Robots navigating through the environment, Black spots indicate the current location of each robot. Bold colored lines show the path each robot has taken till now, and pale colored lines indicate the path planned. The blue circle is the conflict spot between the yellow and red robots

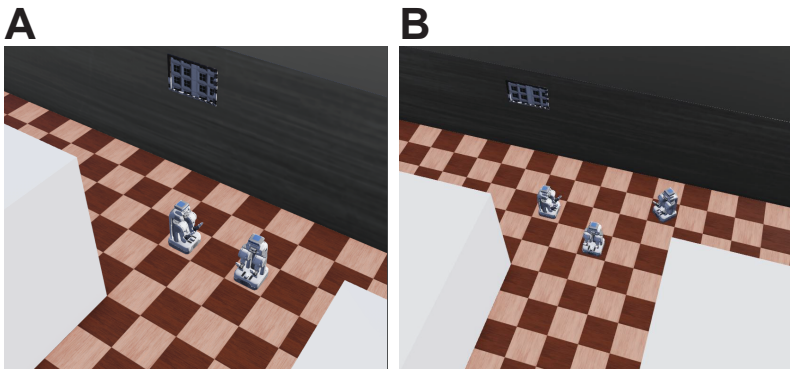

Figure 4.4: Types of robot collisions (**A**) Collision of two robots. (**B**) Collision of three robots.

In order to automate these routines we designed a standalone controller with real-time path planning and online conflict resolution for all the robots. The pseudo-code that demonstrates how this controller works is provided (see Supplementary Algorithm 1). All agents used the same C++ implementation of this pseudo-code in their controllers.

To evaluate and compare different navigation strategies, we used these metrics:

- Time taken in seconds (s) till all goals have been visited.
- Total number of collisions between robots.
- Overall area occupied by all robots as a percentage of the entire environment (%).
- Overall intersection of the occupied area between robots as a percentage of the entire environment (%).

We ran 5 trials with 1, 3 or 5 robots in the environment using these strategies:

- Route (RT): Familiar route strategy. All the paths to the targets have to be aligned to a pre-specified route.
- Survey (SW): The shortest, obstacle-free path to the target is calculated using the spikewave (SW) algorithm.
- Mixed (0.9RT 0.1SW): Agents start on a route and 10% of the time the paths they switch to the spikewave algorithm at the halfway point.
- Mixed (0.6RT 0.4SW): 40% of the time the paths to the targets are calculated using the spikewave algorithm and 60% of the time it has to go along the route.
- Mixed (0.4RT 0.6SW): 60% of the time (i.e., 3 of the 5 trials) the paths to the targets are calculated using spikewave algorithm and 40% of the time (i.e., 2 out of the 5 trials) it has to go along the route.

The 3 mixed strategies represent variability observed within and between individuals. The percentages for the mixed strategies were derived from a DSP meta-analysis[56] that showed when asked to go to a goal participants roughly used SW 40% and RT 60% and when told to take the shortest path participants roughly used SW 60% and RT 40%. The (0.9RT 0.1SW) strategy was derived from the observation that when starting on a route, 10% of the time participants switch to a survey strategy halfway towards their goal.

4.5 Results

We present the results obtained from a comprehensive set of simulations that aimed to compare the performance of 5 distinct navigation strategies, namely RT, 0.9RT 0.1SW, 0.6RT 0.4SW, 0.4RT 0.6SW, and SW. The experiments were conducted using different team sizes of 1, 3, and 5 robots, allowing us to investigate the impact of team size and strategies on navigation performance. To ensure the reliability of our findings, each combination of strategy and robot number was subjected to five independent trials. Various metrics were measured throughout the trials, encompassing the time taken, the number of collisions, the area covered (occupancy), and team intersection. The metrics obtained from the trials were then averaged across the five trials for each experimental condition.

Time taken, defined as the duration in seconds from the start of the simulation until the visit of the last remaining goal, served as a crucial performance indicator. The SW strategy had a significantly shorter time to complete the task across all tested robot numbers (Fig. 4.5(A)). Conversely, the RT strategy exhibited the longest average time. This suggests that robots employing the SW strategy were able to devise innovative shortcuts, enabling them to complete the task more swiftly. For the complete statistical comparisons, see Supplementary Table 4.1. Additionally, we observed that increasing the number of robots led to a significant reduction in the average time taken (see Supplementary Table 4.8 for statistical comparisons).

This outcome can be attributed to the distribution of the goal locations among multiple robots, thereby enhancing efficiency.

We examined the metric of robot collisions, which denotes the total number of encounters between two robots. Notably, for a single robot scenario, this value is zero. The RT strategy exhibited the highest average number of robot collisions when tested with five robots (Fig. 4.5(B)). Although this may be due to an RT trial with 5 robots in which many collisions occurred, Increasing the number of robots tended to increase the number of collisions. It appeared that mixed strategies and the SW strategy had fewer collisions. However, it is hard to compare due to high variability from trial to trial (see Supplementary Tables 4.3 and 4.4 for statistical comparisons).

We analyzed the metric of occupancy, which reflects the extent of the map area covered by the robots. As anticipated, an increase in the number of robots resulted in significantly more area coverage (Fig. 4.5(C)) for all strategies except for SW when comparing teams of 3 to teams of 5 (see Supplementary Table 4.6). The RT strategy and mixed strategies covered more area than SW (Supplementary Table 4.5).

Finally, we evaluated the metric of team intersection, which quantifies the extent of intersected areas between robots on the map. Note that there is no intersection with only one robot. As the navigation strategies went from pure RT to mixed strategies to pure SW, the intersected areas between robots diminished significantly (Fig. 4.5(D)). Conversely, an increase in the number of robots resulted in a significant rise in this metric (see Supplementary Tables 4.7 and 4.8 for statistical comparisons).

Taken together, these results suggest that there may be a sweet spot where varying navigation strategies, like 0.9RT 0.1SW, 0.6RT 0.4SW and 0.4RT 0.6SW, can lead to shorter search times than a pure RT strategy and more coverage of the environment than a pure SW strategy. Moreover, increased intersections, as seen with RT and mixed strategies, may

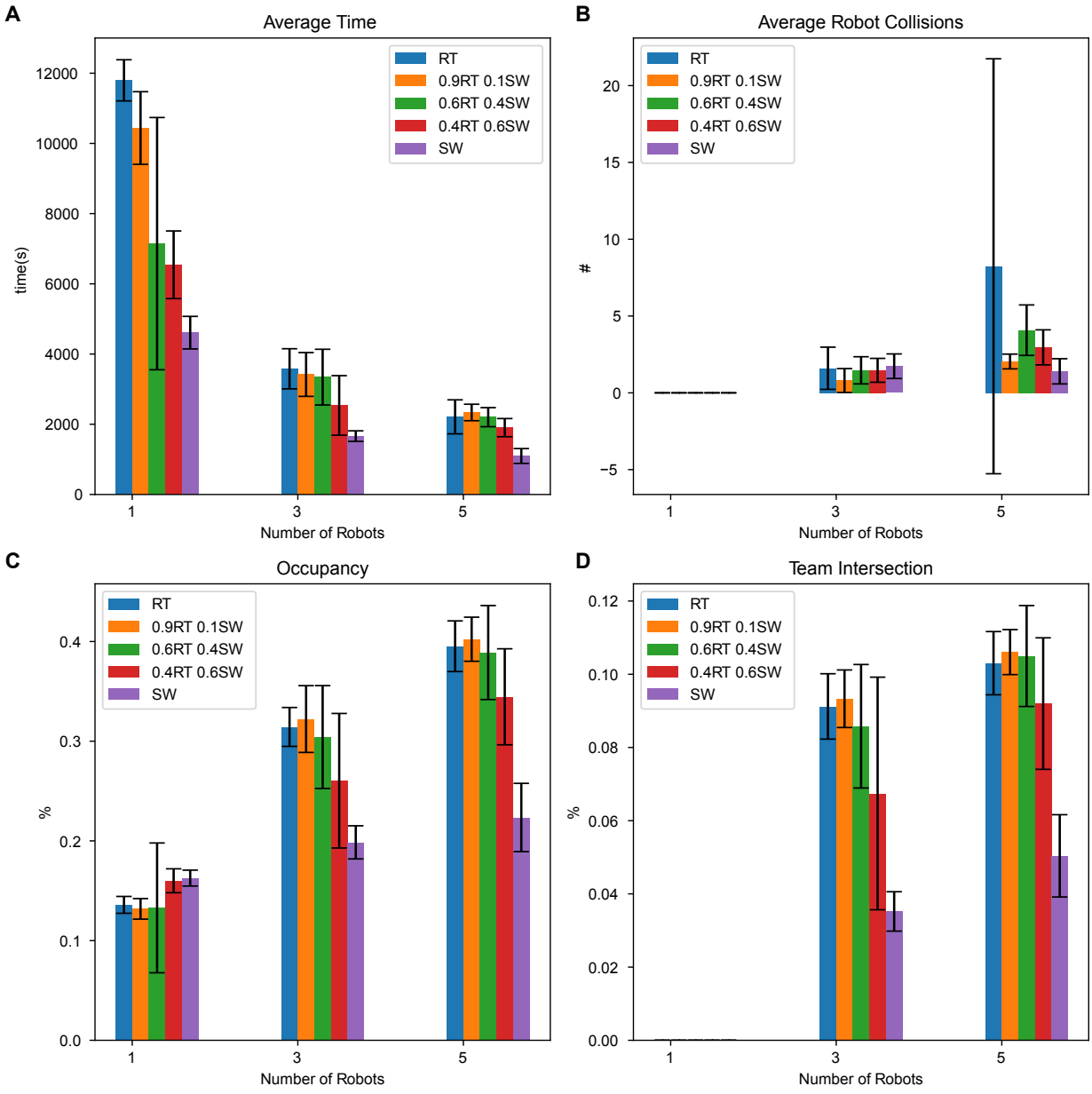

Figure 4.5: Results of experiments for 1, 3, and 5 robots employing 5 different strategies, Route (RT), Survey (SW), and the mixed strategies (0.9RT 0.1SW, 0.6RT 0.4SW, 0.4RT 0.6SW). These values were measured and averaged over 5 trials. (**A**) Time to completion. Cumulative time taken to accomplish all goals (**B**) Robot collisions (**C**) Occupancy, which is the proportion of the environment covered by the robot teams (**D**) Intersection, which quantifies the percentage of cells visited by two or more robots

facilitate team communication, which is especially important in situations where lines of communication are down (e.g., disaster zones).

4.6 Discussion

The main finding of the paper was that although survey knowledge was the most time-efficient strategy and route-based strategies explored most regions of the environment, mixtures that varied strategies struck a balance between being efficient and covering more of the environment. As the number of robots increases tasks are completed faster, and the area covered expanded. The mixed strategies were a sweet spot of efficient navigation, area coverage, and collisions as the team size increased. Robot teams that had varied strategies completed the task of finding all the goal locations in less time than a pure route strategy while visiting more of the environment than a pure survey strategy. This heterogeneous strategy increases efficiency while gaining more knowledge compared to a homogeneous team strategy. In human populations, and potentially in robot teams, the choice of strategy depends on the context. In conditions where efficiency is critical, shortcuts that take more cognitive processing should be favored[56]. In contrast, a familiar route alleviates cognitive load and leads to more interactions with others.

Our results have significant implications for the design of heterogeneous swarms. Rather than varying the agent's sensing or locomotion capabilities[69,70], the agents used a mixture of strategies, which demonstrated clear population benefits. Unlike many multi-agent studies, the present study required to incorporate the physics of the environment with an accurate simulation of commercially available robots[73].These computationally intensive simulations limited team size. In the future, it would be of interest to test the benefits of varying strategies with larger, physical robot teams, which might be possible in an environment like the Robotarium[74].

In summary, the present results explain why human populations vary their navigation strategies and demonstrate that this variation is beneficial. By employing a physical robot simulation that used realistic local sensing and spacing of physical robots, we can better assess how these results could transfer to the real world.

4.7 Supplementary Materials

4.7.1 Algorithms

Algorithm 1 Agent's controller pseudo-code

```
1: procedure CONTROLLER(self, robots, source, destination, strategy, costMap):
2:     while self → timeStep() do
3:         d[self] ← ∞
4:         newCM←costMap
5:         for agent: all agents except self do:
6:             d[agent] ← agent's distance from self
7:         end for
8:         [r,i] ← [min(d),argmin(d)]
9:         checkForObstructionSignal(&newCM)   ▷ checks if other robots have sent any
   obstruction signal, newCm was passed as reference to apply changes accordingly
10:        if r ≤ 6 then
11:            collision[self]++
12:            sendObstructionSignalTo(robots, 25)  ▷ Sends a signal to other robots within
   25 meters of the conflict point to update their costMap subsequently
13:            if self → id ≤ i then
14:                newCM←newCM + block(3, robots[i])   ▷ block(size,robot) makes
   an all-zero 64*64 matrix except for a size ∗ size square block centering at the location
   of robot with infinity value, used size=3 as an arbitrary value to avoid further collisions
   between these two robots
15:            else self→ waitFor(robot[i],12)   ▷ waitFor would stop this agent until its
   distance to robot[i] is at least 12 and then restores the original costMap for both robots
16:            end if
17:        end if
18:        path ← spikeWave( source, destination, strategy, newCM)  ▷ Shortest path from
   source to destination obeying the strategy
19:        navigate(path)   ▷ Navigate functions as a helper method that ensures traversal
   along a given path
20:    end while
21: end procedure
```

4.7.2 Statistical Comparisons

All statistical comparisons and p-values were generated using the two-sample Kolmogorov-Smirnov goodness-of-fit hypothesis test. Bonferroni corrections based on the number of

comparisons were used to test for significance level.

Table 4.1: **Time to Completion**. Comparing effects of strategy within same robot team. ** denotes $p < 0.01$ and * $p < 0.05$ after Bonferroni correction.

Strategy	**1 Robot**	**3 Robots**	**5 Robots**
RT vs. RT60%,SW40%	0.0038*	0.8899	0.9999
RT vs. RT40%,SW60%	0.0038*	0.0515	0.237
RT vs. SW	0.0038*	0.00001**	0.00001**
RT60%,SW40% vs. RT40%,SW60%	0.209	0.1359	0.237
RT60%,SW40% vs. SW	0.0361	0.00001**	0.00001**
RT40%,SW60% vs SW	0.0348	0.00001**	0.00001**

Table 4.2: **Time to Completion**. Comparing effects of strategy between different robot team sizes. ** denotes $p < 0.01$ and * $p < 0.05$ after Bonferroni correction.

Strategy	**1 vs. 3 Robots**	**1 vs. 5 Robots**	**3 vs. 5 Robots**
RT	0.0003**	0.0001**	0.0006**
RT60%,SW40%	0.0066*	0.0037*	0.0098*
RT40%,SW60%	0.0003**	0.0001**	0.0006**
SW	0.0003**	0.0001**	0.00001**

Table 4.3: **Collisions**. Comparing effects of strategy within same robot team. ** denotes $p < 0.01$ and * $p < 0.05$ after Bonferroni correction.

Strategy	**3 Robots**	**5 Robots**
RT vs. RT60%,SW40%	0.9983	0.237
RT vs. RT40%,SW60%	0.8899	0.4141
RT vs. SW	0.9983	0.237
RT60%,SW40% vs. RT40%,SW60%	0.9999	0.4141
RT60%,SW40% vs. SW	0.9983	0.0038*
RT40%,SW60% vs SW	0.9983	0.0038*

Table 4.4: **Collisions**. Comparing effects of strategy between different robot team sizes. ** denotes $p < 0.01$ and * $p < 0.05$ after Bonferroni correction.

Strategy	**3 vs. 5 Robots**
RT	0.5898
RT60%,SW40%	0.0904
RT40%,SW60%	0.0363
SW	0.8634

Table 4.5: **Occupancy**. Comparing effects of strategy within same robot team. ** denotes $p < 0.01$ and * $p < 0.05$ after Bonferroni correction.

Strategy	**1 Robot**	**3 Robots**	**5 Robots**
RT vs. RT60%,SW40%	0.209	0.6974	0.9996
RT vs. RT40%,SW60%	0.0361	0.209	0.209
RT vs. SW	0.0038*	0.0038*	0.0038*
RT60%,SW40% vs. RT40%,SW60%	0.9996	0.6974	0.209
RT60%,SW40% vs. SW	0.6974	0.0038*	0.0038*
RT40%,SW60% vs SW	0.6974	0.209	0.0361

Table 4.6: **Occupancy**. Comparing effects of strategy between different robot team sizes. ** denotes $p < 0.01$ and * $p < 0.05$ after Bonferroni correction.

Strategy	**1 vs. 3 Robots**	**1 vs. 5 Robots**	**3 vs. 5 Robots**
RT	0.0038*	0.0038*	0.0038*
RT60%,SW40%	0.0038*	0.0038*	0.0361
RT40%,SW60%	0.0361	0.0038*	0.0361
SW	0.0038*	0.0361	0.209

Table 4.7: **Intersection**. Comparing effects of strategy within same robot team. ** denotes $p < 0.01$ and * $p < 0.05$ after Bonferroni correction.

Strategy	**3 Robots**	**5 Robots**
RT vs. RT60%,SW40%	0.1359	0.0258
RT vs. RT40%,SW60%	0.1359	0.0001**
RT vs. SW	0.00001**	0.00001**
RT60%,SW40% vs. RT40%,SW60%	0.1359	0.0001**
RT60%,SW40% vs. SW	0.00001**	0.00001**
RT40%,SW60% vs SW	0.0047*	0.00001**

Table 4.8: **Intersection**. Comparing effects of strategy between different robot team sizes. ** denotes $p < 0.01$ and * $p < 0.05$ after Bonferroni correction.

Strategy	**3 vs. 5 Robots**
RT	0.0012**
RT60%,SW40%	0.0012**
RT40%,SW60%	0.00001**
SW	0.00001**

Chapter 5

A ROS2 Multi-Robot Simulation Platform for Human-Inspired Navigation Research

5.1 Abstract

We present a ROS2-based multi-robot simulation platform designed to advance research in human-inspired navigation strategies. The platform builds upon the Gazebo simulator and the ROS2 Navigation Stack (Nav2) to deploy and manage multiple TurtleBot3 robots concurrently, each operating within its own parameterized namespace and running independent SLAM and navigation pipelines. A modular launch configuration enables flexible deployment of 1 to 13 robots, incorporating features such as an initial pose publisher for assigning starting locations, integration with a custom graph construction node that represents the environment as a topological map, and an optional map merging node that facilitates multi-robot SLAM (see Fig. 5.1).

We evaluate the platform's performance by measuring scalability in terms of CPU, memory, and network usage as the robot team size increases. Results indicate that even with 13 robots, the simulation remains stable—though nearing the limits of typical research hardware—validating the platform's suitability for large-scale experiments. This tool accelerates multi-robot navigation research by providing a robust, extensible testbed, ready for integration with reinforcement learning (RL) algorithms, cognitive mapping techniques, and human–robot interaction frameworks.

The platform lays the groundwork for experiments in human-like navigation behavior, enabling studies that incorporate human-inspired route planning, cognitive map representations, and collaborative robot behaviors. This chapter details the design, implementation, and evaluation of the platform, contributing a valuable resource to the robotics research community.

5.2 Introduction

Robust simulation tools are essential for developing and testing advanced robotic navigation strategies, particularly those inspired by human behavior. Human way-finding and spatial strategies typically involve complex decision-making, memory utilization, and adaptation to uncertainty[75]. For instance, in telepresence robotics, it has been demonstrated that providing autonomous navigation assistance can significantly reduce an operator's cognitive load and improve overall user experience[76]. Similarly, in multi-robot teams, deploying varied navigation strategies inspired by human route planning leads to different performance trade-offs[77]. Our prior work showed that robots employing a "survey" (map-based) strategy complete tasks fastest, while a "route" (path-following) strategy provides broader area coverage; a mixture of strategies offers a balanced trade-off between efficiency and exploration[77].

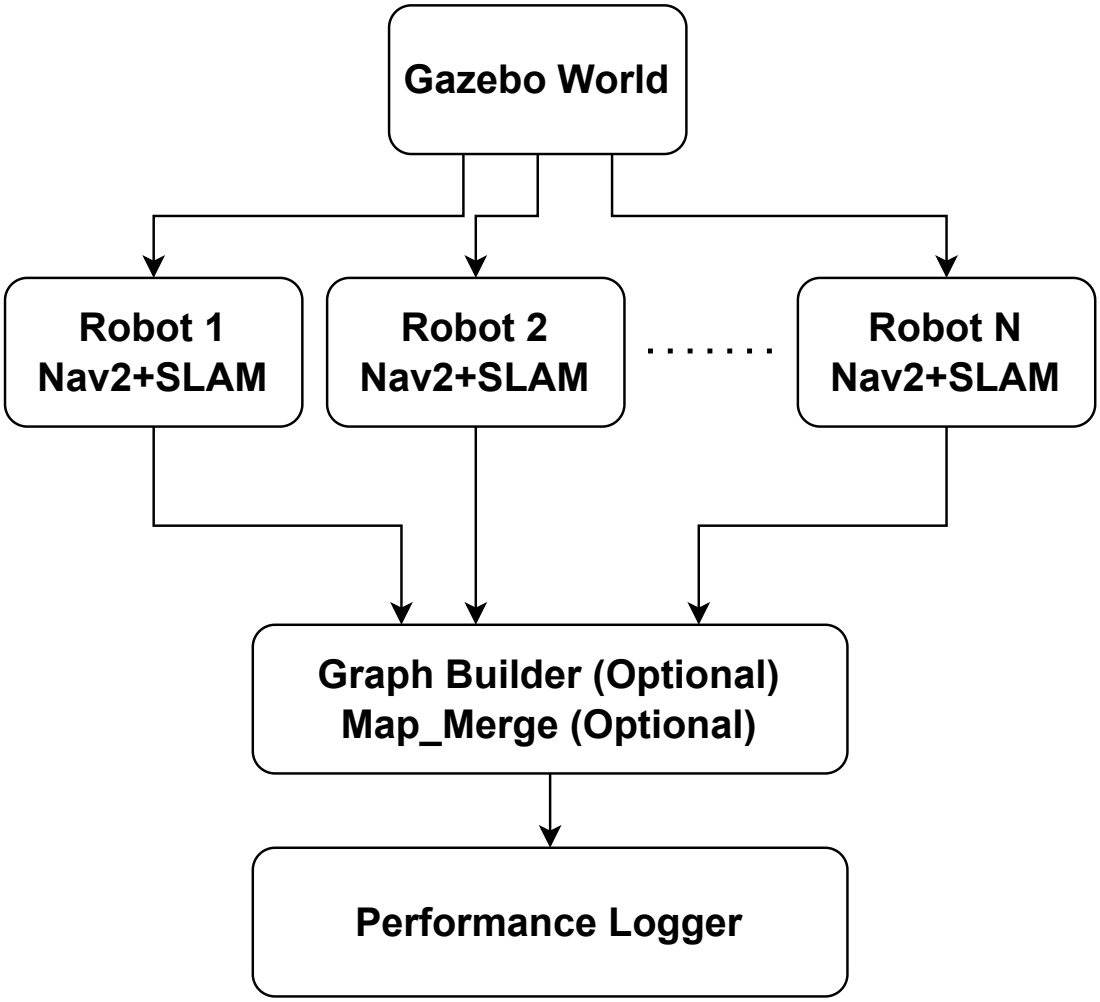

Figure 5.1: Overview of the ROS2-based multi-robot simulation platform architecture. Multiple TurtleBot3 robots are deployed in a Gazebo world, each running isolated instances of the Navigation2 (Nav2) and SLAM stacks within separate namespaces. Optional components, including the Graph Builder and Map Merge Node, enable collaborative mapping and topological data collection. Performance metrics such as CPU, memory, and network usage are logged to assess scalability.

These insights underscore the need for a flexible simulation platform where human-inspired strategies and behaviors can be implemented and evaluated at scale. However, existing robotics simulators and navigation frameworks often focus on single-robot scenarios or lack integrated support for cognitive modeling. Few readily available tools enable the deployment of large teams—up to a dozen robots—in a shared environment with synchronized SLAM, navigation, and data collection for the study of human-inspired navigation.

To address this gap, we developed a ROS2 multi-robot simulation platform that allows researchers to simulate multiple robots and experiment with diverse navigation strategies, map sharing, and collaborative behaviors. The platform builds on open-source ROS2 components (Gazebo, Nav2, SLAM) and extends them with additional modules for multi-robot coordination and performance analysis.

Key features of the platform include:

- **Scalable multi-robot launch:** A single ROS2 launch file can spawn between 1 and 13 TurtleBot3 robots (or more, on powerful computing hardware) in Gazebo, each within an isolated namespace to prevent topic collisions. This supports large-scale swarm simulations.
- **Integrated SLAM and Navigation:** Each robot runs its own SLAM (e.g., `slam_toolbox` or RTAB-Map) and the ROS2 Navigation stack (Nav2) for localization, path planning, and control, allowing each agent to navigate autonomously.
- **Parameterized configuration:** Robot names, sensor parameters, and navigation settings are managed through modular YAML configuration files. An initial pose publisher assigns distinct start positions to each robot to initialize their pose estimates and support downstream data fusion (e.g., map merging).
- **Graph construction node:** A custom node builds a graph-based representation of

the environment and exploration data in real-time. This topological mapping approach reflects human cognitive maps and supports the analysis of spatial relationships among robots and points of interest.

- **Optional map merging:** For multi-robot SLAM scenarios, an optional map merge node can combine individual occupancy grids from each robot into a global map. This feature addresses the lack of built-in ROS2 tools for multi-map merging, enabling cooperative mapping experiments.

- **Performance monitoring:** The system includes tools to log and measure resource usage (CPU, memory, network) as the number of robots scales. We report these metrics to validate that the platform can support computationally intensive experiments on typical research hardware.

In the remainder of this chapter, we detail the platform's design and components (Section 5.4), review related work (Section 5.3), and present empirical results on scalability (Section 5.5). We then discuss how this simulation framework enables new research directions in reinforcement learning and human navigation modeling (Section 5.6). We conclude that our ROS2 multi-robot platform is a valuable tool for the community, accelerating research on human-inspired navigation strategies in multi-robot systems.

5.3 Related Work

Early efforts in multi-robot simulation include 2D simulators such as Stage (for ROS1) and custom Gazebo setups, which required significant manual configuration. With ROS2, tools for multi-robot simulation are still evolving. A common approach is to use namespacing within ROS2 launch files to replicate robot instances. For example, Arshad et al.[78] provide a ROS2 package that launches multiple TurtleBot3 robots in Gazebo by leveraging unique

namespaces for each robot[78].

This approach ensures that topics like `/cmd_vel` and `/scan` are isolated per robot, preventing interference. Our platform adopts a similar namespacing strategy but extends it by incorporating synchronized SLAM and inter-robot coordination nodes (graph construction and map merging), which are not addressed by basic multi-robot launch examples. While multi-robot navigation and map merging were explored in the ROS1 era, ROS1 lacked an official map merging tool, a gap noted by users transitioning to ROS2[79].

Existing solutions often relied on external libraries or remained platform-specific. To address this, we implemented an in-house `merge_map_node`, enabling real-time map fusion during simulation. This facilitates research on cooperative exploration and shared situational awareness. Our approach aligns with prior research in multi-robot SLAM, which emphasizes building a common world model across team members. For instance, Cartographer has been extended to support multi-trajectory (multi-robot) mapping, and other frameworks have proposed merging maps through pose graph alignment[22].

Our contribution in this space is a ready-to-use ROS2 implementation that integrates seamlessly into the navigation launch sequence. Beyond engineering tools, our work is motivated by human-inspired navigation research. Humans are known to use both metric (map-based) and topological representations when navigating. Recent studies in robotics have begun leveraging topological maps to improve scalability and robustness. For example, Muravyev and Yakovlev (2024) demonstrate that topological maps (graphs of place adjacencies) can reduce drift and memory usage, enabling faster path planning through graph sparsity[80].

Likewise, cognitive science and AI research have introduced models in which agents build cognitive graphs of their environment and use them for planning[81]. These works highlight the significance of graph-based representations, inspiring our inclusion of the `graph_construction_node`. By logging points of interest (e.g., visited locations, frontiers, or landmarks) as graph nodes

and connections, our platform captures a high-level spatial structure akin to a human cognitive map. This enables testing of navigation algorithms that operate on a topological level or exploration of how humans might direct a robot team through waypoints.

In summary, while several building blocks of our platform exist independently (ROS2 Nav2 stack, multi-robot launch patterns, SLAM algorithms, etc.), our work integrates them into a cohesive tool specifically tailored for multi-robot, human-inspired navigation experiments. To our knowledge, this is one of the first ROS2-based simulation frameworks to combine multi-robot autonomy with cognitive mapping and map fusion capabilities in a single package. It provides researchers with a convenient, extensible environment to explore how human-like strategies can be implemented and evaluated in robot teams—a topic we have explored in prior studies[1,77] and which remains an active area of research.

5.4 Methods

System Overview: The simulation platform is built on ROS2 (tested with the Humble distribution) and uses Gazebo as the physics and sensor simulation environment. We selected the Clearpath Robotics TurtleBot3 as the robot model due to its popularity and lightweight footprint, enabling many instances to run in parallel. Figure 5.1 illustrates the software architecture, which is organized into modular ROS2 nodes launched via a single entry-point launch file. Each robot is spawned with a unique namespace (e.g., `/robot1_ns`, `/robot2_ns`, etc.) and runs a full stack of capabilities: localization and mapping, path planning, and motion control. A global coordination layer (optional) runs additional nodes (graph construction, map merging) that subscribe to data from all robots.

Figure 5.2 shows the Gazebo environment with two TurtleBot3 robots actively mapping the world using their LiDAR sensors. Figure 5.3 illustrates the corresponding RViz view, where

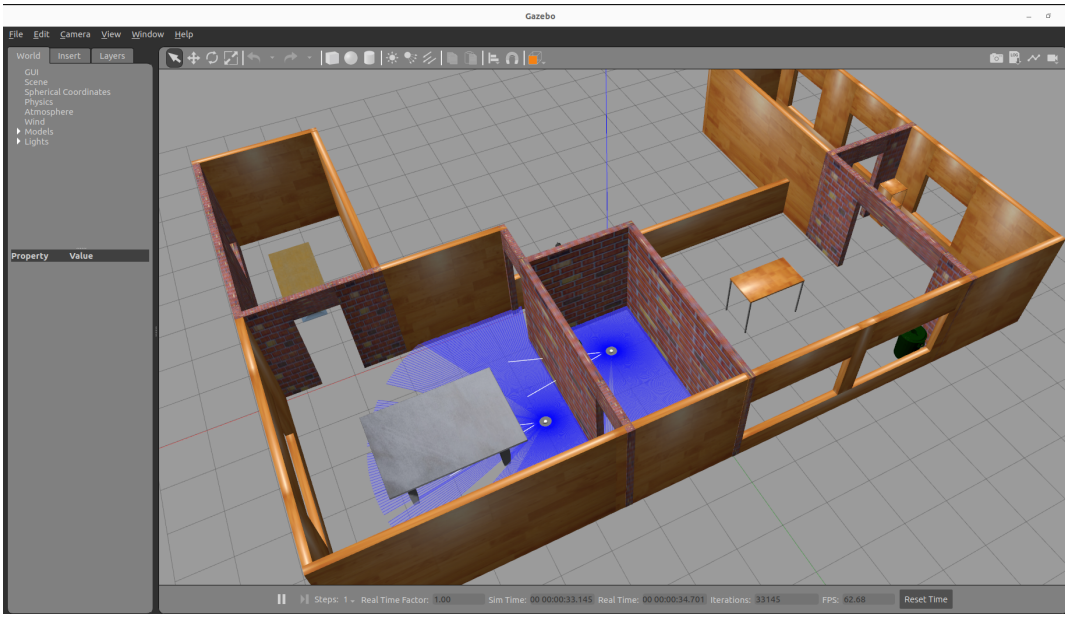

Figure 5.2: Gazebo simulation environment with two TurtleBot3 robots actively mapping the environment. The blue laser scan visualization shows LiDAR data used by the SLAM modules.

occupancy grids are visualized in real-time. Figures 5.2 and 5.4 provide a detailed look at the testing environment, and Figure 5.5 shows how multiple robots are instantiated and organized in namespaces.

5.4.1 Launch Configuration

The heart of the platform is a ROS2 launch script (`fully_integrated_swarm.launch.py`) that orchestrates the spawning of multiple robots and their associated processes. This launch file uses ROS2's Python launch API to generate multiple instance groups. Each robot's instantiation is enclosed within a group that specifies a unique namespace and remappings for its topics and services. The number of robots is defined via a launch argument (e.g., `num_robots:=N`), and a loop in the launch file dynamically creates N groups.

Within each group, we include standardized sub-launch files for:

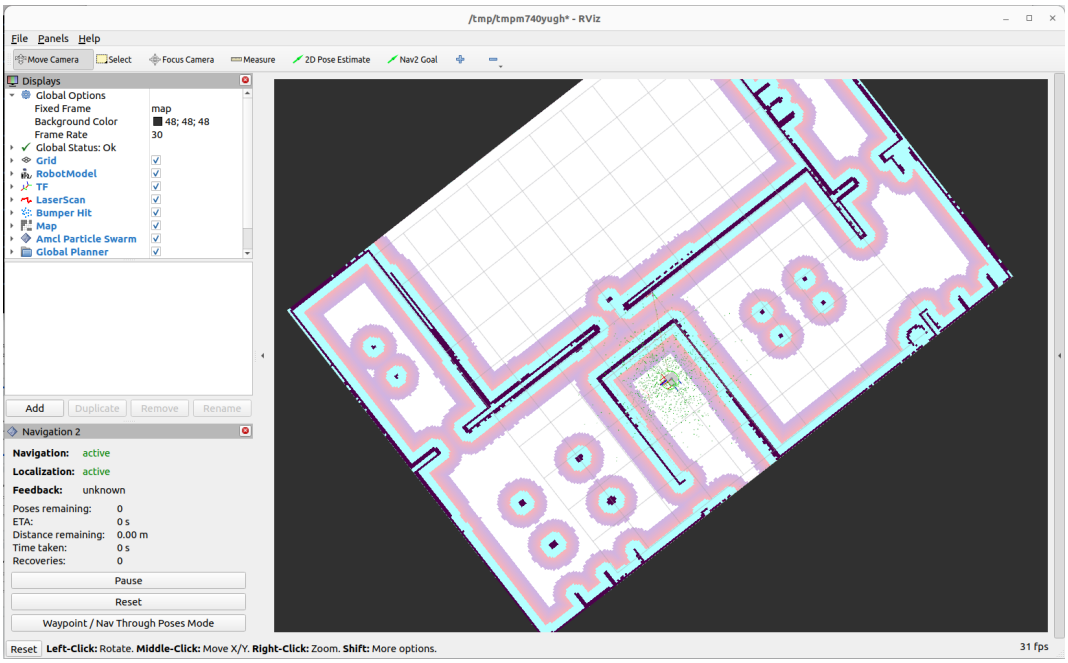

Figure 5.3: RViz visualization of local occupancy grids generated by the robots during mapping. This tool provides real-time feedback on SLAM progress and navigation status.

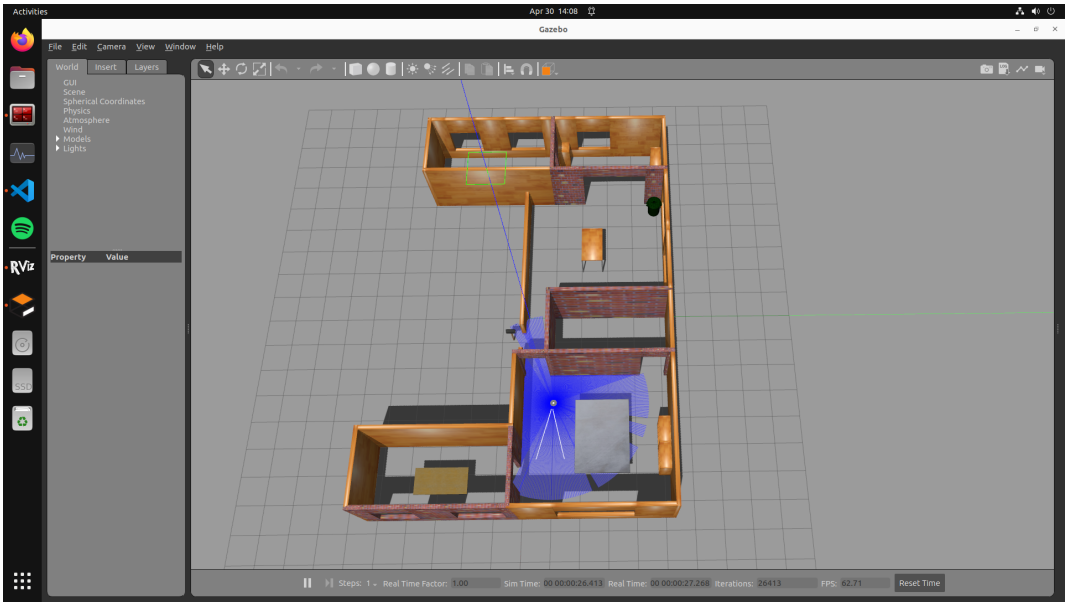

Figure 5.4: Zoomed-out Gazebo view of the full environment, providing context for the multi-robot exploration layout. The space includes long corridors and complex geometry to test scalability and robustness.

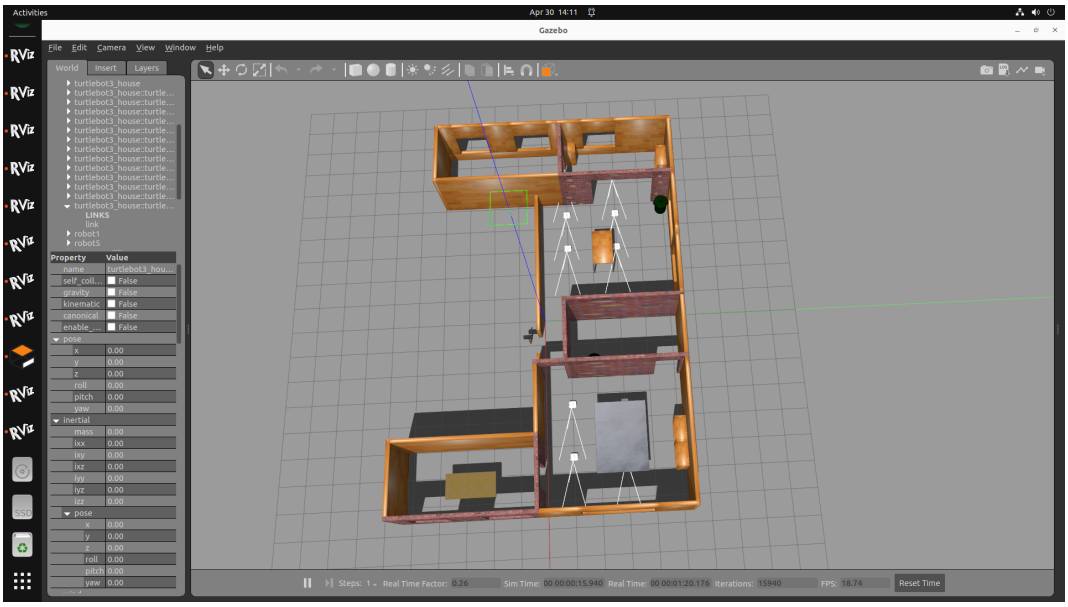

Figure 5.5: Gazebo environment with an overlay showing a fleet of robots. This illustrates how multiple robots are instantiated in separate namespaces using the modular launch system.

- Bringing up the TurtleBot3 model in Gazebo (loading its URDF model and controllers),
- Launching the SLAM node, and
- Launching the Nav2 stack.

We leverage ROS2 Nav2's flexibility to run multiple navigation instances by namespacing, ensuring that global topics (e.g., `/map`, `/odom`) are isolated per robot. Each Nav2 instance uses its own parameter set, which can be a shared YAML file (with relative topic names resolved per namespace) or separate files if differing behaviors are desired. For example, one robot can be configured with a different planner or local costmap size to test heterogeneous strategies. Robots are spawned in Gazebo using a service call that inserts the TurtleBot3 model at pre-defined coordinates per namespace.

We also developed an `initial_pose_publisher` node that publishes `geometry_msgs/PoseWithCovariance` messages to initialize each robot's pose estimate (for AMCL or SLAM). The initial poses are specified in a YAML file or generated procedurally (e.g., in a grid or circular formation).

This setup ensures that each robot starts at a distinct, non-overlapping position, minimizing sensor interference.

5.4.2 SLAM and Navigation

Each robot runs its own SLAM process to autonomously build a map of the environment. We integrated SLAM Toolbox (via `slam_toolbox/async_slam_toolbox_node`) as the default solution, selected for its ROS2 compatibility and ability to map while moving. Parameters (e.g., map resolution, update frequency) are loaded from `slam_online_async.yaml` within each robot's namespace. Optionally, the platform can switch to RTAB-Map for vision-based SLAM using `rtabmap_params.yaml`.

For navigation, each robot runs the ROS2 Nav2 stack, which includes:

- AMCL (for localization),
- A global planner (e.g., NavFn or SmacPlanner),
- A local planner/controller (e.g., DWB or TEB),
- Costmaps and behavior tree servers.

The robots share a common world in Gazebo, where physics and sensor data (LiDAR scans, odometry) are correctly namespaced before reaching ROS2 topics. This allows each Nav2+SLAM pipeline to function independently, as if running on a single-robot system. We ensure deterministic behavior by synchronizing simulation time across all namespaces using ROS2's `/clock`. The full system (e.i. 13 robots) launches approximately 100+ ROS nodes plus Gazebo processes, coordinated to avoid race conditions or resource spikes.

5.4.3 Map Merge Node

For cooperative mapping, we provide an optional `merge_map_node` that subscribes to occupancy grid maps (`/map`) from each robot's SLAM process. Assuming known initial poses (from the pose publisher), it aligns and overlays individual maps into a global occupancy grid published on `/merged_map`. Parameters like resolution and update rate are specified in `map_merge_params.yaml`.

This straightforward overlay approach sufficed in our static world tests but can be enhanced with more advanced merging techniques (e.g., feature-based map alignment) if required. Although fully automated merging remains a complex challenge, our node lays the groundwork for testing cooperative SLAM in multi-robot teams.

5.4.4 Modularity and Configuration

The system is highly modular, with a package structure that separates launch files, configuration files, and node implementations. YAML files are organized by function (SLAM, Nav2, map merging), simplifying parameter tuning. For example, we adjusted LiDAR update rates and sensor ranges to successfully scale up to 16 robots with minimal overhead. Configurable parameters for logging, graph frequency, and map merging allow researchers to fine-tune performance.

This modular design ensures that researchers can easily extend the platform—for example, swapping in a different navigation stack or adding new sensor configurations—without disrupting the rest of the system.

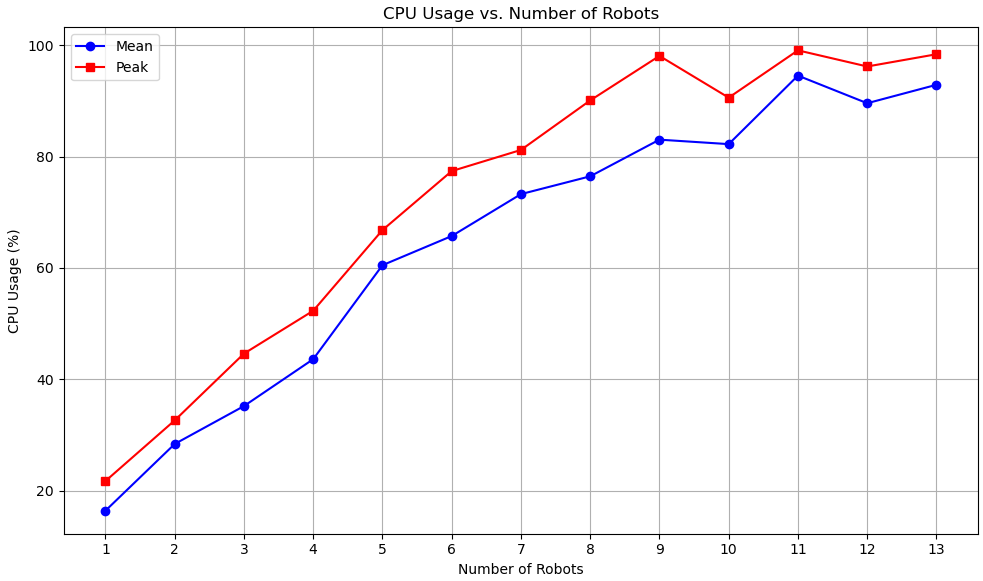

Figure 5.6: CPU usage versus number of robots. The graph illustrates mean and peak CPU utilization as the team size scales from 1 to 13 robots.

5.5 Results

To validate the platform, we conducted a series of simulation trials incrementally increasing the number of robots from 1 up to 13. In each trial, all robots were launched and set to autonomously explore a common environment (an indoor building-like world with obstacles and corridors) for a fixed duration. We monitored three key performance metrics: CPU utilization, memory consumption, and network throughput. The CPU and memory metrics were obtained from the host system (averaging across all cores for CPU and using total RSS for memory across all ROS2 and Gazebo processes). Network usage was measured by monitoring the ROS2 DDS traffic, i.e., the bytes per second transmitted between ROS2 nodes (which in our setup all run on the same machine, but still communicate through the DDS middleware as if over a network).

Figure 5.6 shows how both mean and peak CPU utilization increased with more robots. The mean CPU usage rose roughly linearly from 10% with 1 robot to over 90% with 13 robots. Peak CPU utilization reached 100% by around 9 robots, indicating that at least one CPU

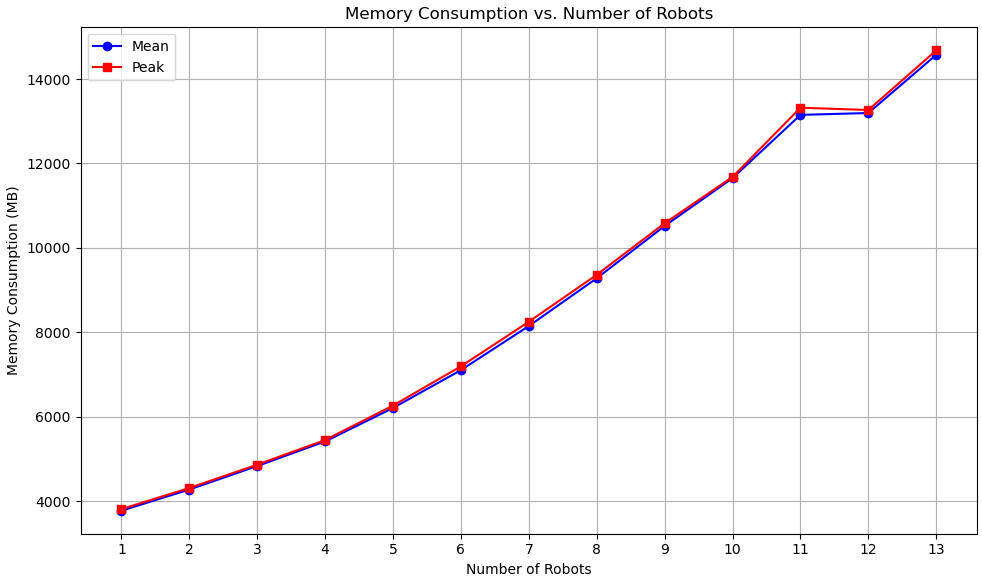

Figure 5.7: Memory consumption versus number of robots. RAM usage grows approximately linearly with team size, indicating predictable scalability.

core was fully saturated, and remained near maximum levels for larger team sizes. Minor dips (e.g., at 10 robots) likely reflect transient scheduling effects. This scaling trend highlights the computational demands of running multiple navigation stacks in parallel. Notably, the optional nodes (graph construction and map merge) contributed minimal overhead; the bulk of CPU usage came from the SLAM and Nav2 processes. Despite nearing system limits at 12 and 13 robots (with the simulation slowing to 0.2× real time), the platform remained functional, demonstrating its capability to support large-scale multi-robot simulations on a single high-performance PC.

Figure 5.7 illustrates the memory consumption trend. Memory usage scaled nearly linearly: starting from 3.7GB with 1 robot, climbing to 6.0GB with 5 robots, 10.0GB with 9 robots, and reaching 14.5GB with 13 robots. The gap between mean and peak memory was minimal, indicating stable usage without major spikes. Each robot adds roughly 0.9–1GB due to its independent SLAM instance, costmaps, and sensor data buffers. The map merge node added negligible memory overhead, as its merged map was not persistently stored by individual robots.

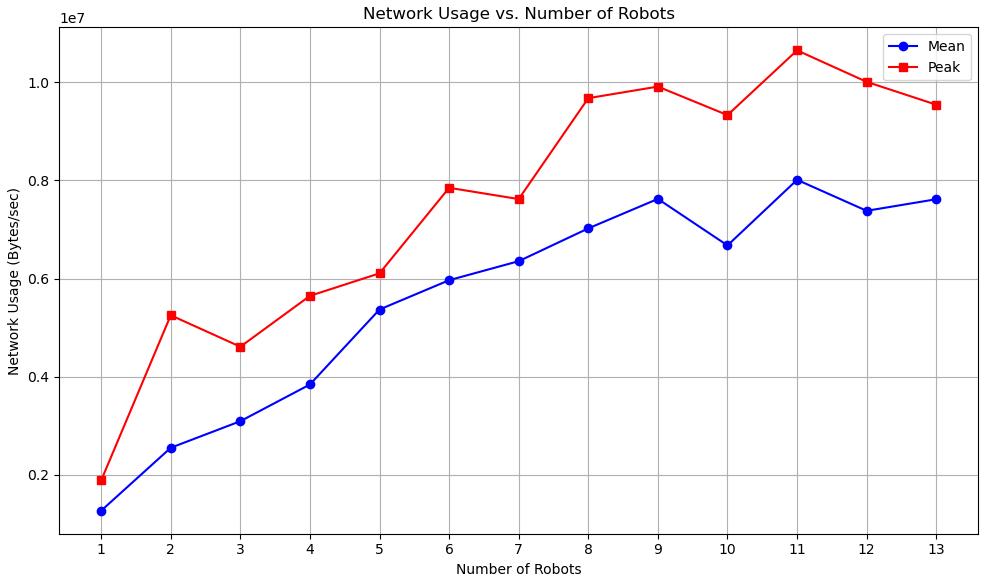

Figure 5.8: Network usage versus number of robots. DDS traffic (both mean and peak) is shown to assess communication load during multi-robot simulations.

The results show that a modern PC with 16GB RAM comfortably supports up to a dozen robots. To push beyond this, one could reduce each robot's map resolution, limit local costmap size, or lower sensor update frequencies.

Figure 5.8 shows how DDS traffic scales with team size. The network load grew steadily, approximately doubling when the second robot was added and continuing to rise to 10MB/s peak traffic at around 9–11 robots. Beyond this, the network traffic plateaued, likely due to DDS optimizations or natural bandwidth limits. For reference, with 1 robot, DDS traffic was 1–2MB/s (mostly LiDAR scans, TF transforms, and map updates). At 2 robots, this rose to 5MB/s, with each additional robot adding 0.5MB/s of continuous data flow. Occasional dips likely reflect planner pauses or idle robots awaiting commands. These results confirm that a standard Ethernet or Wi-Fi connection could handle multi-robot simulations of this scale without issue.

5.5.1 Summary of Performance

The platform demonstrates strong scalability for moderate team sizes. CPU is the first resource to approach saturation (reaching near full utilization at 10–13 robots), while memory and network load increase predictably and linearly. The integrated nature of the platform—spanning multiple robots, mapping, merging, and performance analysis—did not introduce significant overhead beyond what was expected from SLAM and Nav2 stacks.

We qualitatively verified that even at 13 robots, each robot successfully navigated and avoided collisions within Gazebo's physics engine (cross-namespace interactions via Gazebo worked as expected). These findings affirm the platform's robustness for large-scale simulations. Video recordings and detailed logs of the trials are available as supplementary material, showing smooth autonomous exploration even in dense multi-robot scenarios.

5.6 Discussion

The development of this ROS2 multi-robot simulation platform was driven by research questions at the intersection of robotics, human cognition, and autonomy. As a tool, the platform has matured to the point where it can catalyze a range of future investigations. In this section, we reflect on its capabilities in the broader context of human-inspired navigation research and highlight how it is ready to support reinforcement learning (RL) and advanced human behavior modeling.

5.6.1 Reinforcement Learning Integration

A key next step for the platform is serving as a training and testing ground for RL algorithms that aim to achieve human-like navigation policies. The simulation provides rich

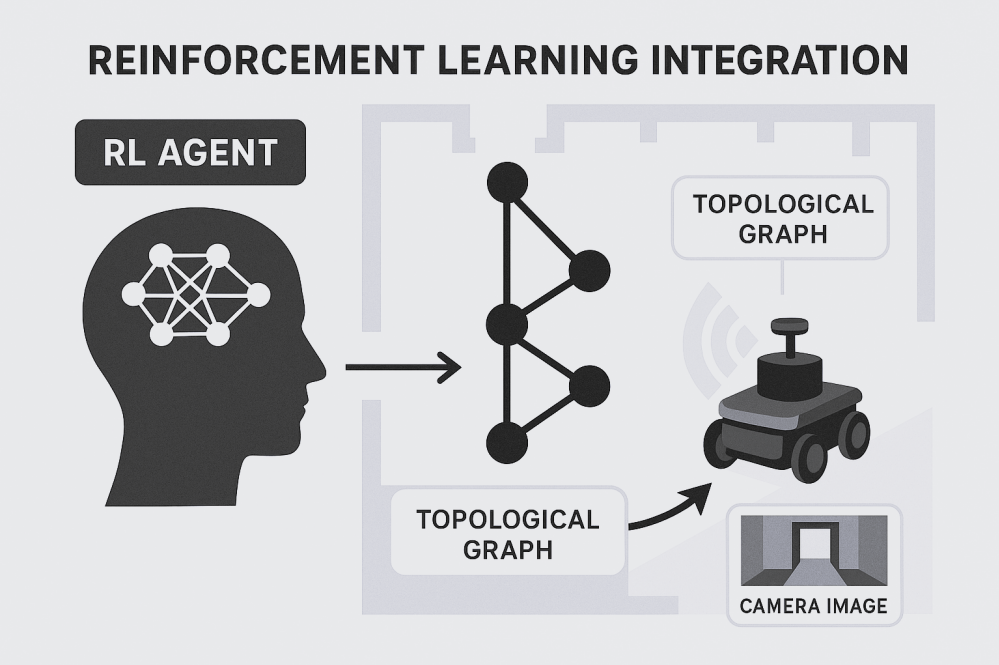

Figure 5.9: Reinforcement Learning Integration with ROS2 Platform

observations (LiDAR scans, camera images if enabled, odometry, etc.) and a safe environment for learning-based agents. We envision integrating an RL agent that processes not only raw sensor data but also higher-level state information—such as the real-time graph-based representation of the environment or even estimates of human cognitive load (see Fig 5.9). For instance, an RL policy could use the topological graph (from the graph node) to decide its next waypoint, mimicking how humans plan routes between landmarks. Prior work also indicates that navigation assistance can significantly influence operators' cognitive load[1].

An RL agent could be trained with a reward function designed to minimize a proxy for human cognitive load, encouraging the robot to navigate in ways that are predictably safe and low-stress for a human follower. The modularity of the platform makes such integration straightforward: the Nav2 planner can be replaced or augmented with a learned policy interfacing via ROS topics or a Behavior Tree (BT) plugin. Because each robot runs in its own namespace, it is feasible to train multiple RL agents in parallel (one per robot) for multi-agent learning, or to mix classical and learned planners within a single run to compare

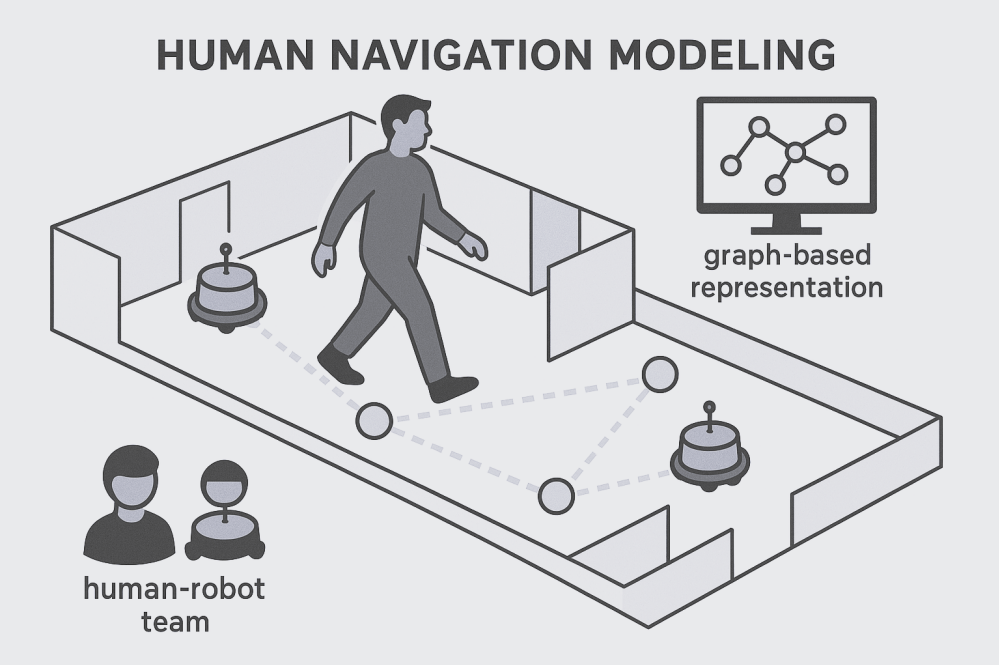

Figure 5.10: Modeling human navigation could be one of the future directions of this study. There could be a Human navigating with multiple robots or the integration of graph-based representation in robot teams.

performance. In short, the platform is RL-ready, offering a controlled yet realistic testbed to develop navigation algorithms that could transfer to physical robots or aid in interpreting human strategies.

5.6.2 Human Navigation Modeling

A central motivation for this work is to provide a testbed for models of human spatial navigation and team coordination (Fig 5.10). Our platform supports simulation of both robotic agents and aspects of human behavior. For example, a "virtual human" agent could be introduced—either as another robot or as a data stream influencing robot behaviors—to represent a human moving through the environment. The graph-based representation is particularly valuable here as a common language between human cognitive maps and robot-built maps. Researchers can inject human-like navigation patterns (e.g., frequently used

paths or a bias toward route-based navigation under cognitive load) and deploy controllers that account for them.

This enables scenarios such as human–robot teams where a human provides high-level guidance (via graph waypoints) while robots autonomously execute navigation to those targets. The platform's multi-robot support allows exploration of how a team might coordinate, mirroring human group behaviors—such as dispersing to cover area (exploration) versus clustering to share knowledge (communication), akin to how human search-and-rescue teams operate. Our earlier work explored how different people employ route knowledge versus survey knowledge, showing that mixed strategies can offer benefits[77].

The platform can extend this line of inquiry by dynamically switching strategies or adapting based on simulated cognitive load. For instance, if a robot (or human) experiences high cognitive load, the system could switch that agent to a simpler route-following mode; conversely, when surplus capacity is detected, the agent might engage in more complex survey-style exploration. These adaptive strategies can be implemented and tested via ROS2 dynamic reconfigure or by re-launching specific nodes. The graph node could also be used to detect overlapping robot knowledge and trigger strategy changes to avoid redundancy—similar to how a human team leader might reassign roles.

5.6.3 Applications and Extensions

The platform is immediately applicable to research in multi-robot navigation, human–robot interaction (HRI), and cognitive robotics. In HRI, one could simulate a remote telepresence scenario where one robot acts as the user's avatar and other robots roam autonomously. By toggling autonomy on and off, we can measure cognitive load and performance differences—similar to our prior telepresence study[1]—but now in a multi-robot context.

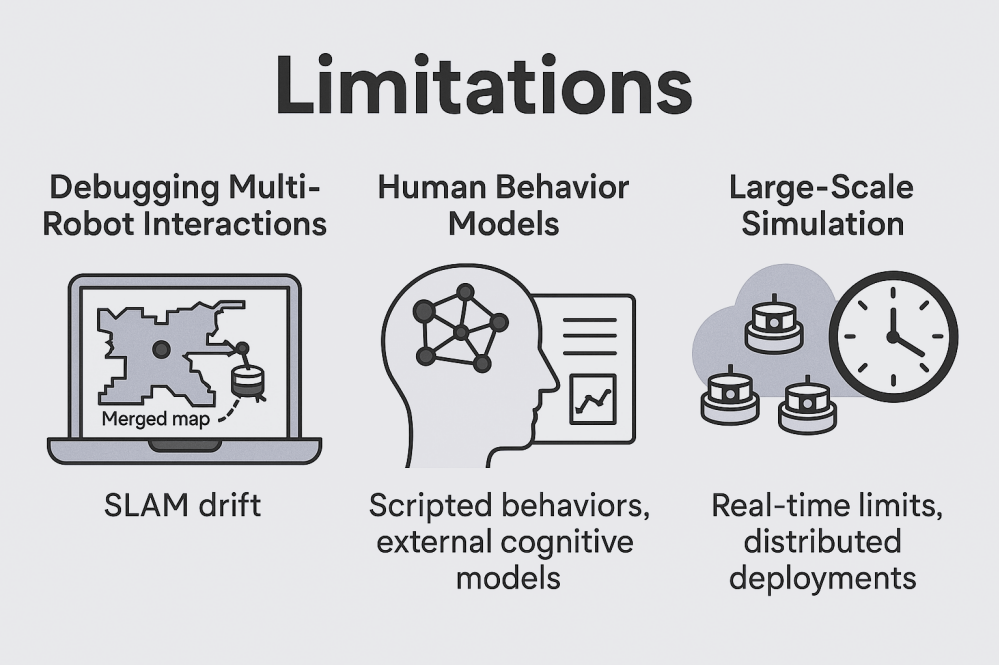

Figure 5.11: Limitation of the ROS2 platform

Another application is education and benchmarking: new multi-robot algorithms (for exploration, task allocation, etc.) can be evaluated in a realistic and repeatable setup (see Fig. **??**). We plan to release the platform as an open-source toolkit to enable reproducibility and further development by other researchers. The platform's foundation in standard ROS2 also enables deployment to physical robots with minimal adjustments; for instance, `fully_integrated_swarm.launch.py` can be adapted to bring up multiple real TurtleBot3s by swapping Gazebo for real hardware bring-up and configuring networking appropriately. This provides a smooth simulation-to-reality pipeline, which is essential for confirming that simulated behaviors translate to physical robots.

5.6.4 Limitations

Despite its capabilities, the platform has limitations (Fig 5.11. First, debugging multi-robot interactions can be complex—e.g., significant SLAM drift in one robot can misalign the merged map, affecting others. Scaling to larger teams may require careful tuning or advanced

techniques like decentralized mapping. Second, while the platform provides tools for human-inspired behavior (graph representation, strategy switching), it does not include built-in human behavior models; users will need to script behaviors or integrate external cognitive models[75]. Our goal is to lower the barrier to entry for such research. Third, performance tests showed that at 13 robots we were nearing real-time limits on a single machine. Researchers aiming for larger teams (20–50 robots) may need distributed deployments or cloud-based simulation. While ROS2 and DDS support this, we have not yet explored multi-machine configurations in depth.

5.6.5 Future Work

Future work will focus on integrating a learning-based navigation policy that leverages the graph-based cognitive map and potentially human feedback signals, bridging reinforcement learning and cognitive science within the platform Fig 5.12. We also plan to simulate user studies, where human participants operate robots (via a game-like interface) alongside autonomous agents, to study coordination and strategy adaptation. Data from these studies—including participant actions and cognitive load metrics—can inform further refinements. Additional extensions include incorporating human crowd simulation to study social navigation, where robots navigate in ways that respect human social norms. Thanks to the platform's modularity, these additions are feasible without overhauling the core architecture.

In conclusion, the ROS2 multi-robot simulation platform is a powerful research tool that unifies multi-robot autonomy and human-inspired navigation. It is ready for use in diverse experiments and can evolve with community contributions. By providing a robust, extensible foundation, we hope to accelerate progress in understanding and improving robot navigation strategies that are efficient, safe, and aligned with human expectations—whether that means reducing operator cognitive load or coordinating a team of robots with diverse strategies to

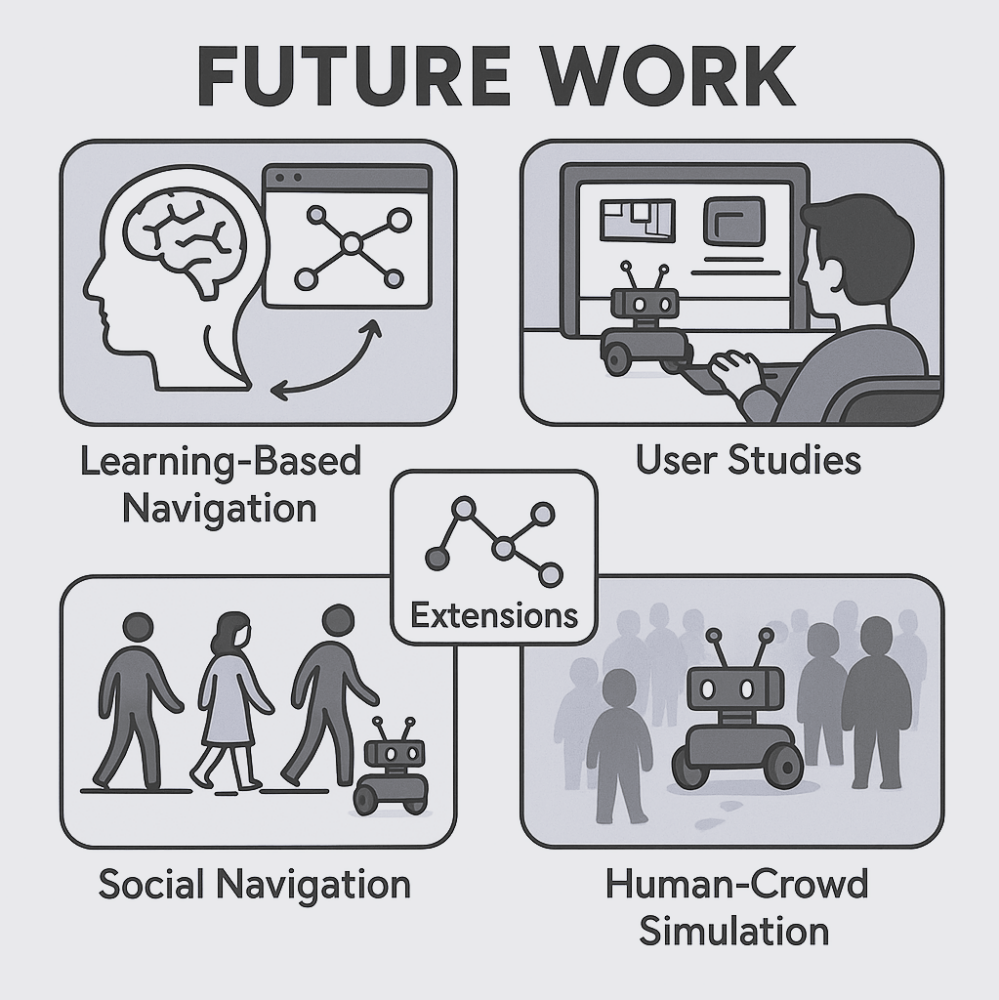

Figure 5.12: Future work including but not limited to Learning Based Navigation, User Studies, Social Navigation and Human Crowd Simulation

achieve shared goals.

5.7 Supplementary Materials and Code

- Video of the Gazebo environment (https://www.youtube.com/watch?v=nfTs7sWDnww)
- Github Repository (https://github.com/AmirMohaddesi/disaster_response_swarm)

Chapter 6

Future Directions

The culmination of this dissertation has brought forward a multi-faceted investigation into human-inspired robotic navigation, encompassing individual cognitive load in telepresence systems, the value of strategic diversity in multi-agent teams, and the development of a modular and scalable ROS2-based simulation platform. While the results presented provide substantial contributions to the domains of HRI, multi-robot systems, and simulation, they also open the door to numerous promising research directions.

This chapter outlines several of these future directions, categorized into three major themes: (1) advancing shared autonomy for telepresence systems, (2) extending strategic diversity and cognitive modeling in robot teams, and (3) expanding the capabilities and impact of the ROS2 simulation framework. In addition, cross-cutting themes such as human-robot teaming, explainability, and physical deployment are discussed as promising interdisciplinary frontiers.

6.1 Extending Shared Autonomy and Cognitive Load Modeling

One of the foundational insights from this thesis is that navigation autonomy in telepresence robots can significantly reduce user cognitive load, leading to better task performance and subjective experience. However, the scope of shared autonomy remains relatively narrow and task-specific in most current systems. The following suggestions could address this limitation.

6.1.1 Adaptive Autonomy and Real-Time Cognitive Feedback

Future work should explore adaptive shared autonomy systems that dynamically modulate assistance based on real-time estimates of the user's cognitive state. Building on recent developments in physiological sensing (e.g., EEG, eye-tracking, and galvanic skin response), a robot could detect indicators of high workload or stress and proactively take over certain functions, switching back when the user regains capacity. Incorporating closed-loop physiological feedback into the robot's behavior policy could make autonomy more responsive and personalized.

6.1.2 Multimodal Cognitive Load Estimation

While NASA-TLX and task performance remain the gold standards for workload estimation, there is increasing interest in multimodal fusion models that combine behavioral cues (e.g., hesitation, error rate), biosignals (e.g., HRV, pupil dilation), and contextual task information. Incorporating such fusion models in robotic systems may enable real-time workload-aware control policies, particularly relevant in telepresence where context switches (e.g., navigating

vs. observing) can be rapid and frequent.

6.1.3 Embodied Collaboration and Conversational Support

A long-term vision involves integrating natural language interaction and embodied social cues with shared autonomy. Robots that can communicate their intent, explain their actions, and receive user preferences conversationally could foster trust and collaboration. This aligns with ongoing work in explainable AI and co-adaptive interfaces, where both human and robot adjust to each other's behavior over time.

6.2 Cognitive Diversity and Strategy Modeling in Robot Teams

The second contribution of this dissertation highlighted that strategic diversity (emulating human route, graph, and survey navigation styles) can enhance team-level exploration and robustness. Future directions in this space involve refining the underlying models and extending the range of environments and tasks.

6.2.1 Learning Strategy Profiles from Human Trajectories

While the current implementation models strategies heuristically, future work could leverage imitation learning or inverse reinforcement learning (IRL) to learn strategy profiles from real human trajectory data. For example, the path datasets provided by SNL collaborators could be used to train agents to mimic the decision patterns of survey-based vs. route-based navigators. Combining this with psychometric data (e.g., spatial anxiety, working memory

scores) could reveal mappings between cognitive traits and navigational behaviors.

6.2.2 Strategic Diversity in Dynamic and Unstructured Environments

The existing strategy analysis focuses on static maze-like environments. Future simulations and real-world experiments should test strategic diversity in dynamic or partially observable environments, such as warehouses, disaster zones, or multi-level buildings. Robots could switch strategies based on environmental features or mission context, adapting over time using reinforcement learning or curriculum learning techniques.

6.2.3 Social Navigation and Heterogeneous Teaming

Strategic diversity becomes even more critical in human-robot teams. Robots that can recognize and complement the navigation style of a human partner (e.g., by being the "map-builder" while the human follows known routes) could support mixed-initiative collaboration. Further, modeling theory of mind for teammates, predicting what others know or intend to do, could enable socially aware path planning and division of labor in robot collectives.

6.3 Scaling and Extending the ROS2 Simulation Framework

The final component of this dissertation is the ROS2-based platform designed to simulate multi-robot navigation at scale. While the current system demonstrates support for SLAM, namespace-isolated agents, and basic map merging, several enhancements can expand its

scientific and practical utility.

6.3.1 Integration with Learning Frameworks

An important future direction is integrating the simulation environment with machine learning tools such as PyTorch and TensorFlow. This would allow direct training of navigation agents using reinforcement learning, behavior cloning, or hybrid methods. Specifically, policy-gradient-based methods (e.g., PPO, A3C) could be applied to learn decentralized team strategies under communication constraints. Support for gym-like interfaces and curriculum learning could make the platform accessible to the wider learning community.

6.3.2 Scalable Benchmarking and Dataset Generation

The platform can also serve as a benchmarking suite for multi-robot navigation under varying network, sensory, and control conditions. Future enhancements could include automated profiling of CPU, memory, and network load (partially demonstrated in this thesis), as well as dataset logging for SLAM, localization drift, and team-level performance metrics. The creation of an open-source multi-agent dataset generator, modeled after Habitat or CARLA, could foster community benchmarks and reproducibility.

6.3.3 Semantic and Graph-Based Extensions

One of the key contributions of the platform is its compatibility with graph-based cognitive representations. Future versions could implement full semantic labeling, where nodes in the topological graph represent rooms or functions (e.g., “kitchen”, “exit”) rather than just coordinates. This would support semantic planning, where goals are specified at a high level

(e.g., "go to the nearest safe zone"), and allow testing of models that combine symbolic and sub-symbolic reasoning which is critical for explainability and generalization.

6.4 Interdisciplinary Opportunities and Physical Validation

Several interdisciplinary research threads naturally extend from this work.

6.4.1 Neuroscience-Inspired and Bio-Inspired Navigation

Recent findings in cognitive neuroscience (e.g., grid cells, path integration) continue to inform navigation modeling. Incorporating such biologically inspired mechanisms into multi-agent systems (e.g., using vector-based navigation, ego-centric memory, or replay-based learning) could enhance performance in sparse reward environment.

6.4.2 Large-Scale Human-Robot Experiments

With access to datasets like those from UCSB/UCI's SNL studies, future experiments could simulate human-robot interaction in large virtual environments or test real-world robot teammates that emulate human behavior profiles. Mixed-reality setups, where humans control avatars alongside autonomous robots in shared environments, would offer a unique testbed for spatial coordination and communication.

6.4.3 From Simulation to Physical Deployment

Finally, transitioning from Gazebo-based simulation to real-world deployment on TurtleBot3 or similar platforms would validate the platform's utility. Efforts would focus on SLAM robustness, navigation policy transfer, and system latency. Incorporating ROS2 multi-host networking, real-time operating support, and minimal-resource configurations would bridge the gap between lab and field.

6.5 Conclusion

This dissertation has explored navigation from three complementary angles; cognitive load in telepresence, strategic diversity in robot teams, and scalable ROS2 simulation. Each of these areas, while independently valuable, offers rich potential for future integration and interdisciplinary growth. By drawing from cognitive science, HRI, reinforcement learning, and systems engineering, the future of human-inspired multi-robot navigation promises more adaptive, efficient, and intuitive robot systems. The tools and findings developed here lay a strong foundation for continued exploration at this nexus of human and machine spatial intelligence.

Bibliography

[1] Grace Pan, Thea Weiss, Seyed Amirhosein Mohaddesi, Joseph W. Szura, and Jeffrey L. Krichmar. Navigation and cognitive load in telepresence robots. In *Proceedings of the IEEE International Conference on Development and Learning (ICDL)*, pages 1–6. IEEE Explore, 2024.

[2] Susan L. Epstein. Navigation, cognitive spatial models, and the mind. *Common Model of Cognition Bulletin*, 1(2):321–326, Dec. 2017. URL `https://ojs.library.carleton.ca/index.php/cmcb/article/view/2628`.

[3] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati, editors, *Human Mental Workload*, pages 139–183. North Holland Press, 1988.

[4] Raphael Kaplan and Karl J. Friston. Planning and navigation as active inference. *Biological Cybernetics*, 112(4):323–343, 2018. ISSN 1432-0770. doi: 10.1007/s00422-018-0753-2. URL `https://doi.org/10.1007/s00422-018-0753-2`.

[5] Michael Peer, Iva K. Brunec, Nora S. Newcombe, and Russell A. Epstein. Structuring knowledge with cognitive maps and cognitive graphs. *Trends in Cognitive Sciences*, 25 (1):37–54, 2021. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2020.10.004. URL `https://www.sciencedirect.com/science/article/pii/S1364661320302503`.

[6] Patrick Foo, William H. Warren, Andrew Duchon, and Michael J. Tarr. Do humans integrate routes into a cognitive map? map- versus landmark-based navigation of novel shortcuts. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 31 (2):195–215, 2005. doi: 10.1037/0278-7393.31.2.195. URL `https://doi.org/10.1037/0278-7393.31.2.195`.

[7] Edward C. Tolman. Cognitive maps in rats and men. *Psychological Review*, 55(4): 189–208, 1948.

[8] E. R. Chrastil and W. H. Warren. From cognitive maps to cognitive graphs. *PLoS One*, 9(11):e112544, 2014. ISSN 1932-6203. doi: 10.1371/journal.pone.0112544.

[9] Steven A. Marchette, Arnold Bakker, and Amy L. Shelton. Cognitive mappers to creatures of habit: Differential engagement of place and response learning mechanisms predicts human navigational behavior. *Journal of Neuroscience*, 31(43):15264–15268, 2011. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.3634-11.2011. URL `https://www.jneurosci.org/content/31/43/15264`.

[10] A. P. Boone, X. Gong, and M. Hegarty. Sex differences in navigation strategy and efficiency. *Memory Cognition*, 46(6):909–922, 2018. ISSN 1532-5946. doi: 10.3758/s13421-018-0811-y.

[11] Edvard I. Moser, Emilio Kropff, and May-Britt Moser. Place cells, grid cells, and the brain's spatial representation system. *Annual Review of Neuroscience*, 31:69–89, 2008. doi: 10.1146/annurev.neuro.31.061307.090723. URL `https://doi.org/10.1146/annurev.neuro.31.061307.090723`.

[12] Trygve Solstad, Charlotte N. Boccara, Emilio Kropff, May-Britt Moser, and Edvard I. Moser. Representation of geometric borders in the entorhinal cortex. *Science*, 322(5909): 1865–1868, 2008. doi: 10.1126/science.1166466.

[13] Benjamin Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119(1-2): 191–233, 2000. doi: 10.1016/S0004-3702(00)00017-5.

[14] Dileep George, Rajeev V. Rikhye, Nishad Gothoskar, J. Swaroop Guntupalli, Antoine Dedieu, and Miguel Lázaro-Gredilla. Clone-structured graph representations enable flexible learning and vicarious evaluation of cognitive maps. *Nature Communications*, 12(1):2392, 2021. doi: 10.1038/s41467-021-22559-5. URL https://doi.org/10.1038/s41467-021-22559-5.

[15] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. doi: 10.1109/TRO.2021.3075644.

[16] NVIDIA. Isaac sim multi-robot navigation documentation. https://docs.nvidia.com/isaacsim/latest, 2023. Accessed: 2025-04-30.

[17] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004. doi: 10.1109/IROS.2004.1389727.

[18] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009.

[19] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, 2017. URL https://arxiv.org/abs/1705.05065.

[20] Olivier Michel. Cyberbotics ltd. webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):5, 2004.

[21] Eric Rohmer, Surya P. N. Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326, 2013. doi: 10.1109/IROS.2013.6696520.

[22] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016. doi: 10.1109/ICRA.2016.7487258. URL https://google-cartographer.readthedocs.io.

[23] Steve Macenski and Ivona Jambrecic. Slam toolbox: Slam for the dynamic world. *Journal of Open Source Software*, 6(61):2783, 2021. doi: 10.21105/joss.02783. URL https://doi.org/10.21105/joss.02783.

[24] J. O'Keefe and J. Dostrovsky. The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1):171–175, 1971. ISSN 0006-8993. doi: https://doi.org/10.1016/0006-8993(71)90358-1. URL https://www.sciencedirect.com/science/article/pii/0006899371903581.

[25] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I. Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005. doi: 10.1038/nature03721. URL https://doi.org/10.1038/nature03721.

[26] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, December 2016. ISSN 1941-0468. doi: 10.1109/tro.2016.2624754. URL http://dx.doi.org/10.1109/TRO.2016.2624754.

[27] Mario Bourgault, Nathalie Drouin, and Emilie Hamel. Decision making within distributed project teams: An exploration of formalization and autonomy as determinants of success. *Project Management Journal*, 39(1_suppl):S97–S110, 2008. doi: 10.1002/pmj.20063. URL https://doi.org/10.1002/pmj.20063.

[28] R. Davey. Telepresence robotics: An overview, 2021. URL https://www.azorobotics.com/Article.aspx?ArticleID=414. Retrieved March 18, 2022.

[29] Mordor Intelligence. Telepresence robots market: 2022–27. URL https://www.mordorintelligence.com/industry-reports/telepresence-robots-market. Retrieved March 14, 2022.

[30] V. Newhart and M. Warschauer. Virtual inclusion via telepresence robots in the classroom: An exploratory case study. *The International Journal of Technologies in Learning*, 23(4):9–25, 2016. doi: 10.18848/2327-0144/CGP/v23i04/9-25.

[31] V. Ahumada-Newhart and J. S. Olson. Going to school on a robot. *ACM Transactions on Computer-Human Interaction*, 26(4):1–28, 2019. doi: 10.1145/3325210.

[32] J. Fuhrman. Cognitive load theory: Helping students' learning systems function more efficiently, 2017. URL https://www.franklin.edu/institute/blog/cognitive-load-theory-helping-students-learning-systems-function-more-efficiently. Retrieved March 14, 2022.

[33] AHRQ. Nasa task load index. URL https://digital.ahrq.gov/health-it-tools-and-resources/evaluation-resources/workflow-assessment-health-it-toolkit/all-workflow-tools/nasa-task-load-index. Retrieved March 10, 2022.

[34] S. Esmaeili Bijarsari. A current view on dual-task paradigms and their limitations to capture cognitive load. *Frontiers in Psychology*, 12:648586, 2021. doi: 10.3389/fpsyg.2021.648586.

[35] J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285, 1988. doi: 10.1207/s15516709cog1202_4.

[36] I. Rae and C. Neustaedter. Robotic telepresence at scale. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017. doi: 10.1145/3025453.3025855.

[37] Tesla. Tesla vehicle safety report, 2022. URL https://www.tesla.com/VehicleSafetyReport. Retrieved March 14, 2022.

[38] M. T. Sqalli, K. Tatsuno, K. Kurabe, H. Ando, H. Obitsu, R. Itakura, T. Aoto, and K. Yoshino. Improvement of a tele-presence robot autonomous navigation using slam algorithm. In *IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2016.

[39] W. K. Chung, T. L. Lam, and Y. Xu. A natural assisted navigation motion for telepresence robots. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2014. doi: 10.1109/ROBIO.2014.7090675.

[40] L. Takayama, E. Marder-Eppstein, H. Harris, and J. M. Beer. Assisted driving of a mobile remote presence system: System design and controlled user evaluation. In *2011 IEEE International Conference on Robotics and Automation*, 2011. doi: 10.1109/ICRA.2011.5979637.

[41] M. Quigley. Ros: An open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[42] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase. Development of the research platform of a domestic mobile manipulator utilized for international competition and field test. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018. doi: 10.1109/IROS.2018.8593798.

[43] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase. Development of human support robot as the research platform of a domestic mobile manipulator. *ROBOMECH Journal*, 6(1), 2019. doi: 10.1186/s40648-019-0132-3.

[44] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011. doi: 10.1109/SSRR.2011.6106777.

[45] J. Lee. Time-on-task as a measure of cognitive load in tblt. *The Journal of AsiaTEFL*, 16(3):958–969, 2019. doi: 10.18823/asiatefl.2019.16.3.12.958.

[46] L. Yang, B. Jones, C. Neustaedter, and S. Singhal. Shopping over distance through a telepresence robot. *Proceedings of the ACM on Human-Computer Interaction*, 2 (CSCW):1–18, 2018. doi: 10.1145/3274460.

[47] J. L. Plass and S. Kalyuga. Four ways of considering emotion in cognitive load theory. *Educational Psychology Review*, 31(2):339–359, 2019. doi: 10.1007/s10648-019-09473-5.

[48] J. E. LeDoux and R. Brown. A higher-order theory of emotional consciousness. *Proceedings of the National Academy of Sciences*, 114(10), 2017. doi: 10.1073/pnas.1619316114.

[49] A. U. Batmaz, J. Maiero, E. Kruijff, B. E. Riecke, C. Neustaedter, and W. Stuerzlinger. How automatic speed control based on distance affects user behaviours in telepresence robot navigation within dense conference-like environments. *PLOS ONE*, 15(11), 2020. doi: 10.1371/journal.pone.0242078.

[50] C. Neustaedter, G. Venolia, J. Procyk, and D. Hawkins. To beam or not to beam. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, 2016. doi: 10.1145/2818048.2819922.

[51] Blackberry. Autonomous systems: Ultimate guides. URL `https://blackberry.qnx.com/en/ultimate-guides/autonomous-systems`. Retrieved April 4, 2022.

[52] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour, and S. Schaal. Learning of grasp selection based on shape-templates. *Autonomous Robots*, 36(1-2): 51–65, 2013. doi: 10.1007/s10514-013-9366-8.

[53] C. Weaver. Self-driving cars learn to read the body language of people on the street, 2021. URL `https://spectrum.ieee.org/selfdriving-cars-learn-to-read-the-body-language-of-people-on-the-street`. Retrieved April 14, 2022.

[54] E. R. Chrastil and W. H. Warren. Active and passive spatial learning in human navigation: Acquisition of graph knowledge. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 41(4):1162–1178, 2015. ISSN 1939-1285. doi: 10.1037/xlm0000082.

[55] A. P. Boone, B. Maghen, and M. Hegarty. Instructions matter: Individual differences in navigation strategy and ability. *Memory Cognition*, 47(7):1401–1414, 2019. ISSN 1532-5946. doi: 10.3758/s13421-019-00941-5.

[56] J. L. Krichmar and C. He. Importance of path planning variability: A simulation study. *Topics in Cognitive Science*, 15(1):139–162, 2023. ISSN 1756-8757. doi: 10.1111/tops.12568.

[57] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2014.11.006.

[58] Edward Lam and Pierre Le Bodic. New valid inequalities in branch-and-cut-and-price for multi-agent path finding. *Proceedings of the International Conference on Automated Planning and Scheduling*, 30(1):184–192, 2020. doi: 10.1609/icaps.v30i1.6660.

[59] Edward Lam, Pierre Le Bodic, Daniel Harabor, and Peter J. Stuckey. Branch-and-cut-and-price for multi-agent path finding. *Computers and Operations Research*, 144: 105809, 2022. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2022.105809.

[60] Jiaoyang Li, Zhe Chen, Daniel Harabor, Peter J. Stuckey, and Sven Koenig. Anytime multi-agent path finding via large neighborhood search. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4127–4135. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/568. Main Track.

[61] Teng Guo and Jingjin Yu. Sub-1.5 time-optimal multi-robot path planning on grids in polynomial time. *CoRR*, abs/2201.08976, 2022.

[62] Qandeel Sajid, Ryan Luna, and Kostas E. Bekris. Multi-agent pathfinding with simultaneous execution of single-agent primitives. In *Fifth Annual Symposium on Combinatorial Search*, 2012.

[63] Ko-Hsin Cindy Wang and Adi Botea. Mapp: a scalable multi-agent path planning algorithm with tractability and completeness guarantees. *J. Artif. Int. Res.*, 42(1): 55–90, 2011. ISSN 1076-9757.

[64] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W. Durham, T. K. Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. *CoRR*, abs/2005.07371, 2020.

[65] A. P. Engelbrecht. Heterogeneous particle swarm optimization. In M. Dorigo, M. Birattari, G. A. Di Caro, et al., editors, *Swarm Intelligence*, pages 191–202. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15461-4.

[66] A. Hara, K. Shiraga, and T. Takahama. Heterogeneous particle swarm optimization including predator-prey relationship. In *The 6th International Conference on Soft Computing and Intelligent Systems and The 13th International Symposium on Advanced Intelligence Systems*, pages 1368–1373, 2012. doi: 10.1109/SCIS-ISIS.2012.6505194.

[67] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. Hernandez, and R. G. Harley. Particle swarm optimization: Basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Computation*, 12(2):171–195, 2008. ISSN 1941-0026. doi: 10.1109/TEVC.2007.896686.

[68] M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, et al. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics and Automation Magazine*, 20(4):60–71, 2013. ISSN 1558-223X. doi: 10.1109/MRA.2013.2252996.

[69] R. Maeda, T. Endo, and F. Matsuno. Decentralized navigation for heterogeneous swarm robots with limited field of view. *IEEE Robotics and Automation Letters*, 2(2):904–911, 2017. ISSN 2377-3766. doi: 10.1109/LRA.2017.2654549.

[70] Yoshinori Sano, Takahiro Endo, Takumi Shibuya, and Fumitoshi Matsuno. Decentralized navigation and collision avoidance for robotic swarm with heterogeneous abilities.

Advanced Robotics, 37(1-2):25–36, 2023. ISSN 0169-1864. doi: 10.1080/01691864.2022.2117996.

[71] S. M. LaValle. Motion planning part i: The essentials. *IEEE Robotics and Automation Magazine*, 18(1):79–89, 2011. doi: 10.1109/Mra.2011.940276.

[72] T. Hwu, A. Y. Wang, N. Oros, and J. L. Krichmar. Adaptive robot path planning using a spiking neuron algorithm with axonal delays. *IEEE Transactions on Cognitive and Developmental Systems*, 10(2):126–137, 2018. ISSN 2379-8920. doi: 10.1109/TCDS.2017.2655539.

[73] O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.

[74] Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1699–1706, 2017. doi: 10.1109/ICRA.2017.7989200.

[75] Daria de Tinguy, Tim Verbelen, and Bart Dhoedt. Learning dynamic cognitive map with autonomous navigation. *Frontiers in Computational Neuroscience*, Volume 18 - 2024, 2024. ISSN 1662-5188. doi: 10.3389/fncom.2024.1498160. URL `https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2024.1498160`.

[76] Kazjon Grace and Mary Lou Maher. Specific curiosity as a cause and consequence of transformational creativity. In *Proceedings of the Sixth International Conference on Computational Creativity (ICCC)*, pages 259–266, 2015. URL `https://computationalcreativity.net/iccc2015/proceedings/11_3Grace.pdf`.

[77] Seyed A. Mohaddesi, Mary Hegarty, Elizabeth R. Chrastil, and Jeffrey L. Krichmar. Benefit of varying navigation strategies in robot teams. In Oliver Brock and Jeffrey Krichmar, editors, *From Animals to Animats 17*, pages 63–77, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-71533-4.

[78] Arshad Lab. turtlebot3_multi_robot: Multi-robot simulation package for ros2. `https://github.com/arshadlab/turtlebot3_multi_robot`, 2024. Accessed: 2025-05-01.

[79] Lukasz Kuczynski and contributors. multirobot_map_merge: Merge maps from multiple robots. `https://github.com/lukicdarkoo/multirobot_map_merge`, 2018. ROS1 package, accessed: 2025-05-01.

[80] Kirill Muravyev and Konstantin Yakovlev. Maintaining topological maps for mobile robots. In Wolfgang Osten, editor, *Sixteenth International Conference on Machine Vision (ICMV 2023)*, volume 13072, page 130720W. International Society for Optics and Photonics, SPIE, 2024. doi: 10.1117/12.3023459. URL `https://doi.org/10.1117/12.3023459`.

[81] Benjamin Kuipers. The spatial semantic hierarchy. *Artif. Intell.*, 119(1–2):191–233, May 2000. ISSN 0004-3702. doi: 10.1016/S0004-3702(00)00017-5. URL `https://doi.org/10.1016/S0004-3702(00)00017-5`.