ABSTRACT

Title of Thesis: STIGMERGY-BASED MULTI-AGENT MOTION PLANNING USING QR CODES

Miles Jones
Master of Science, 2025

Thesis Directed by: Professor Michael W. Otte
Department of Aerospace Engineering

This thesis takes a stigmergy-inspired approach to asynchronous multi-agent motion planning in communication-restricted environments. Building on sampling-based motion planning, it introduces a framework for encoding paths within QR codes that are physically placed in the environment. Feasible paths generated by a robot are post-processed through path pruning and ellipsoidal containers to reduce data size while preserving essential trajectory information. QR codes allow later robots to decode, reconstruct, and replan directly within the corresponding ellipsoid, supported by algorithms for occupancy grids, greedy ellipsoid expansion, and uniform ellipsoidal sampling.

Simulation results show that the proposed approach reduces computation time and memory requirements while maintaining feasible path quality, highlighting the potential of geometry-based stigmergy as a lightweight, communication-free mechanism for sharing trajectory information among autonomous agents.

STIGMERGY-BASED MULTI-AGENT MOTION PLANNING USING QR CODES

by

Miles Jones

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2025

Advisory Committee:
Professor Michael W. Otte, Chair/Advisor
Professor John Martin
Professor Huan Xu

Acknowledgments

First, I would like to give my sincerest thanks to Dr. Otte for providing me the opportunity to do this work with him and those in the Motion and Teaming Laboratory. His guidance throughout this experience has been invaluable, and he has always been available to provide it when I needed it. I appreciate the numerous opportunities he has provided me leading up to this thesis. I would also like to extend my gratitude to Dr. Mumu Xu and Dr. John Martin for participating in my thesis committee and giving me very valuable feedback and perspectives on this work.

I would also like to show my deepest appreciation and gratitude to my family, who have always given me unconditional support, guidance, and motivation every step of the way. None of this would have been possible without that. I would like to thank my friends for giving me advice and feedback when I needed it, as well as keeping me sane when I was trying to debug something. Thank you as well to those in the Motion and Teaming Laboratory for all the kindness, feedback, questions, and support, especially for my defense.

Table of Contents

Acknowledgments ii

Table of Contents iii

List of Tables v

List of Figures vi

List of Abbreviations viii

Chapter 1: Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Contribution . . . 4
- 1.3 Outline of thesis . . . 5

Chapter 2: Literature Review 6
- 2.1 Background . . . 6
- 2.2 Stigmergy . . . 7
- 2.3 Sampling based motion planning . . . 10
- 2.4 Path reconstruction . . . 12
- 2.5 Knowledge sharing . . . 14
- 2.6 Mapping and localization . . . 15
- 2.7 Closely related work . . . 16

Chapter 3: Overview 18
- 3.1 Nomenclature . . . 19
- 3.2 Definition of the multi-agent stigmergy problem . . . 20
  - 3.2.1 Assumptions and Constraints . . . 20
  - 3.2.2 Formal problem definition . . . 21

Chapter 4: Approach 23
- 4.1 Preliminary motion planning . . . 23
- 4.2 Post-processing . . . 24
  - 4.2.1 Ellipsoid construction . . . 25
  - 4.2.2 QR codes . . . 26

4.3 Using stored data . . . 27

Chapter 5: Processing motion planning information 30
5.1 Ellipsoid properties . . . 30
5.1.1 Ellipsoid definitions and properties . . . 30
5.1.2 Sampling ellipsoids directly . . . 36
5.2 Occupancy grids . . . 38
5.3 Generating the QR code . . . 40
5.3.1 Node scoring . . . 41
5.3.2 Path pruning . . . 42
5.3.3 Ellipsoid Construction . . . 43

Chapter 6: Stigmergic motion planning 47
6.1 Parsing data . . . 47
6.2 Motion planning within an ellipse . . . 51

Chapter 7: Experiments 55
7.1 Experiment set-up . . . 55
7.2 Parameter selection . . . 57

Chapter 8: Results and discussion 58
8.1 Computation time comparison . . . 58
8.2 Path length results . . . 65
8.3 Ellipsoid results . . . 67
8.4 Memory usage results . . . 69

Chapter 9: Future Work 74

Chapter 10: Conclusions 76

Bibliography 78

List of Tables

7.1 Experimental parameters 56

8.1 Statistics for feasible solution time and processing time 59
8.2 Statistics for feasible solution path length 66

List of Figures

2.1 High-level flowchart demonstrating how the concepts in the literature review fit together in the structure of this thesis 6

3.1 This visual overview visualizes the end result of each phase. The first robot (on the left) generates a feasible path, which then gets post-processed. In the post-processing (center box), the first robot prunes the path to eliminate nodes. If the path is too long to fit into a specific QR code, an ellipse is computed to eliminate more nodes from the path and encapsulate the area they were in. Finally, a second robot decodes this information and replans solely in the ellipsoid, if one exists. . . 19

3.2 Labeled version of figure 3.1 that shows where each variable in the problem is located. On the left, the first robot generates a path P. Path P is modified and turned into a path P', which can contain an ellipsoid D (shown in the center). This path P' and ellipsoid D is used by a second robot (on the right) to complete a new path between x_s and x_g similar to the original P. 22

5.1 Construction of an ellipsoid from an arbitrary path 31

5.2 Ellipsoid formed from an arbitrary path with a point q on the boundary 33

5.3 Example of drawing direct samples using algorithm 2. Samples are drawn from a unit-disk, which is then transformed into an area-uniform ellipse. 37

5.4 Example of drawing direct samples using algorithm **??**. Cholesky decomposition is used to map the unit-disk to the desired ellipse. A sample is randomly drawn from the unit disk and mapped to the ellipse. 38

7.1 Visualization of the total collection of data run in this experiment. 56

8.1 Per-environment boxplots of the solution time data for robot 1. 60

8.2 Per-environment boxplots of the solution time data for the robot 2 without assistance from robot 1. 60

8.3 Per-environment boxplots of the solution time data for the robot 2 using Q. 60

8.4 Percentage of successful solutions over time (robot 1) 61

8.5 Percentage of successful solutions over time (robot 2 without Q) 61

8.6 Percentage of successful solutions over time (using Q) 62

8.7 Mean and standard deviation of QR decoding time in each environment 62

8.8 Plot showing the p-value from a K-S test as additional environments are added into the data set. Each K-S test uses solution time data between robot 1 and robot 2. 63

8.9 Plot showing the p-value from a K-S test as additional environments are added into the data set. The K-S test is performed using the solution time data from trials with and without Q. 64

8.10 Plot showing the p-value from a K-S test as additional environments are added into the data set. The overall computation times from the first robot are compared to the second robot in each K-S test. 64
8.11 Path length mean and standard deviation for each environment (robot 1) 66
8.12 Path length mean and standard deviation for each environment (robot 2 without Q) 66
8.13 Path length mean and standard deviation for each environment (robot 2 with Q) . 67
8.14 Histogram of ellipse areas as a percentage of the environment bounds 68
8.15 3D Histogram of ellipsoid area and solution time 69
8.16 Mean and standard deviation of ellipsoid area in each environment 69
8.17 Box plot showing memory usage, comparing the robot 1 path to Q and the robot 2 path. 70
8.18 Box plot showing memory usage of the robot 1 path data 71
8.19 Box plot showing memory usage of the robot 2 path data (without Q) 71
8.20 Box plot showing memory usage of the robot 2 path data 72
8.21 Box plot showing memory usage of Q data 73
8.22 Histogram comparing memory usage between the paths from robot 1 and robot 2. 73

List of Abbreviations

RRT	Rapidly-Exploring Random Tree
RRT*	Rapidly-Exploring Random Tree Star
RRT#	Rapidly-Exploring Random Tree Sharp
SBMP	Sampling-based Motion Planning
AO	Asymptotically Optimal

Chapter 1: Introduction

With the rapid rise of autonomous systems across numerous public and private domains, it has become more important to make autonomous motion planning and decision-making more flexible and robust. In industrial, commercial, and space domains, autonomous systems must gather and process information from their surroundings and logically apply it to complete their prescribed tasks, such as exploration, search and rescue, or the transportation of specific objects. Motion planning is the fundamental method an autonomous agent uses to complete these tasks in practice. One of the primary goals of motion planning is for an agent to calculate a feasible path, where "feasible" refers to a trajectory that is collision-free and is viable for the agent to follow. Sampling-Based Motion Planning (SBMP) has become a popular solution to solving the feasible path planning problem. Traditionally, an agent uses onboard sensors and instruments in combination with known dynamics to inform motion planning through unknown environments and identify a feasible path. Additionally, multi-agent systems may use standardized networking and design structures to share information across agents in a controlled environment.

There are contexts where autonomous agents cannot directly share information through wireless communication. Tasks that take place underwater, underground, in disaster areas, or in environments with significant interference may face issues with inconsistent or unreliable wireless communication. In nature, creatures such as ants, bees, or termites work around similar

constraints by leaving pheromones in the environment. This form of interaction falls under a category known as "stigmergy" [1, 2]. *Stigmergy* is the general term for the mechanism of indirect coordination among multi-agent systems in which traces left in the environment stimulate subsequent actions. In nature, stigmergy commonly manifests itself in the form of molecular signals and chemical pheromones, with ants and termites being primary examples. Information about role assignments, localization, path planning, and more can be asynchronously communicated through the environment itself.

Humans have also recently started using technology reminiscent of stigmergy. Barcodes, QR codes, and other fiducial markers have become a common method of storing digital information in our surroundings. QR codes have quickly become a common method for sharing pedestrian navigational information, menus, important links, announcements, and more. Considering this fact, these tools are also potentially useful as a medium for robots to collaborate.

This thesis presents recent work on using QR-code compatible stigmergy to enable cooperative multi-robot sampling-based motion planning. Autonomous agents that can store and distribute information within the physical environment when communication is restricted are potentially invaluable. As such, we address this problem by experimenting with a versatile method of sharing motion planning data between agents via physical codes in the environment.

1.1 Motivation

The motivation for using modern human forms of stigmergy for cooperative motion planning is supported by a few points. The reasons why we chose to study this problem are as follows.

1. **Asynchronous operation:** A direct benefit of using stigmergic communication to supple-

ment motion planning is the potential for asynchronous multi-robot collaboration. Storing data in the environment itself means that robots are free to act independently and exchange valuable knowledge indirectly. Robots within a team have the opportunity to work in the same environment at very different times and leave lasting physical records of their progress, regardless of the task.

2. **No wireless communication:** Aside from asynchronous operation, multi-robot collaboration without direct wireless communication is also possible. Additionally, taking advantage of the environment enables multi-robot collaboration that is not dependent on wireless communication. Implementing effective wireless communication for a team of robots can be very costly, especially in unknown environments.

3. **Versatility:** One of the valuable aspects of stigmergy-based communication and planning is the potential flexibility it provides over existing methods. To our knowledge, existing research similar to the work in this thesis usually involves some form of regularized static tags or markers. However, stigmergy and the general concept of sharing information asynchronously between systems can manifest in many different forms, and we believe there is significant potential for physical markers to be used to facilitate autonomous tasks. This thesis focuses specifically on replicating a shared trajectory, but there is ample opportunity to focus on other specific applications: mapping areas, identifying obstacles, localization, and more.

4. **Access to environmental information:** With regards to SBMP in an unknown environment, receiving any kind of data can immediately be used to improve the performance of a team and reduce overall computation time. Agents can use data from a past agent

and avoid redundant work, therefore allowing for the opportunity to reallocate for better purposes beyond feasible path planning.

5. **Potential efficiency:** By sacrificing some time spent on processing and encoding data, agents benefit a lot from the computation and experience of past robots. Moreover, using geometry to contain part of a path reduces the area being sampled. Considering that a feasible path is assumed to exist within this geometry, future agents can focus the motion planning on a much smaller area compared to the entire path and environment.

1.2 Contribution

As a proof of concept, this work makes a few key contributions. The primary contributions come from the manner in which future agents use prior motion planning data, rather than the motion planning itself. The following points describe the contributions made by this thesis.

1. To our knowledge, this is the first proof of concept of using stigmergy-based QR-code data packets in an unstructured environment for robot collaboration. There is no need for multiple QR codes to be placed in any structured or regularized manner. Only one is needed, and it can be placed wherever necessary.

2. This thesis proposes a framework for leveraging prior motion planning experiences that asynchronously benefits the autonomous agents that use it. Agents unpack encoded data from persistent physical markers in the environment and apply it to help avoid any redundant work.

3. Autonomous agents can save on time and memory usage, even with very minimal data and

small environments. Receiving data from prior motion planning attempts gives agents a head start such that a solution will be found earlier, if it exists.

4. A method for determining which part of a path should be reconstructed by sampling from an ellipsoid. Local collision rate estimates are used to inform the process of ellipsoid placement. During the process, regions with greater recorded collisions are prioritized, meaning more planning time is dedicated to these regions.

1.3 Outline of thesis

This document covers the concept, theory, and experiments involved in our research and proceeds as follows. Chapter 2 is a literature review, discussing prior research that supports and relates to the context of our work. Chapter 3 provides an overview of the background to this work and walks through each "phase" of the proposed work. Chapters 4 and 5 then go into the technical details of geometric and motion planning tools, demonstrating how the relevant theories and properties are leveraged in the experiments. Chapters 6 and 7 discuss the details of the experiment and the corresponding results. Chapters 8 and 9 then present the future work for this research and the conclusion.

Chapter 2: Literature Review

In this thesis, multiple existing research concepts related to motion planning are integrated in a novel way. This section, which surveys work related to this thesis, is broken into seven subsections that focus on general background information, stigmergy, sampling-based motion planning, sampling bias, path reconstruction, knowledge sharing, and mapping and localization. Finally, we conclude with a subsection that covers the most closely related work.

2.1 Background

This thesis plays with concepts rooted in both stigmergy and sampling-based algorithms. The previous works that introduced these ideas are foundational for the presented work. The work in [3] introduced Rapidly-exploring Random Trees (RRT) and became a turning point toward modern SBMP. Extensions to RRT and other SBMP methods are also collected and presented

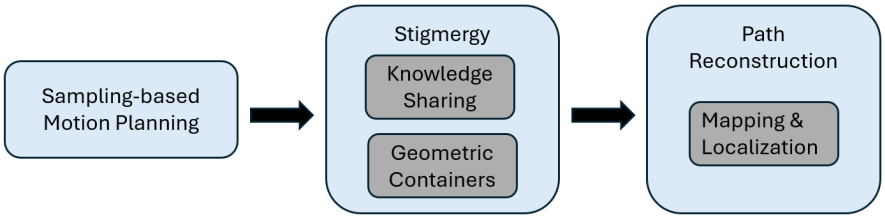


Figure 2.1: High-level flowchart demonstrating how the concepts in the literature review fit together in the structure of this thesis

in [4]. Beyond path planning for feasible paths, [5, 6] improves RRT and transitions to optimal path planning for sampling-based scenarios.

The study of stigmergy goes at least as far back as 1959, but it is still a new mechanism being explored in autonomous systems [2]. The underlying mechanics behind the behavior of different insects are studied in [1, 2, 7] to analyze and define stigmergy. Research into stigmergy is further broken down in [8], splitting it into components labeled as action, medium, and trace. In the stigmergy feedback loop, a trace is left by some action in a medium, promoting more action(s). Stigmergy and its features have since been applied to multi-robot research to improve planning algorithms [9].

In this thesis, biased and adaptive sampling are part of how stored information is leveraged. Inspired by some of the same ideas in [5], [10] proposes algorithms that focus on information gain, using adaptive sampling to direct an agent's efforts to areas that help accomplish this goal. Additionally, [11] studies different methods for uniformly sampling directly from an ellipsoid, which is closely related to the sampling method used in this thesis.

2.2 Stigmergy

Stigmergy has been implemented in motion planning in numerous ways, involving localization, navigation, learning, and more. Considering how physical codes are used in the real world to share information, localization and positioning are commonly explored applications. In addition to its data storage capabilities, a QR code's unique orientation makes it naturally useful for localization. One method used in [12–14] is creating a regularly distributed array of QR codes containing coordinates that serve as a guide to an agent. Observing the relative orientation and

position of the QR code with respect to the agent provides an easily estimated pose for the agent. The work in [13] takes this further by encoding a path matrix in each QR code, detailing the path information of neighboring QR codes. Alternatively, large QR codes are used to store detailed floor plans as a full navigation graph or a simpler edge list in [15]. Instead of using QR codes to store any coordinates or floor plan, QR codes can be customized to assist agents in localization [16–18]. QR codes serve as a visual signal in [16], and when paired with SLAM technology, more accurate maps are generated. A modified QR code is used in [17], simplifying its unique orientation to improve detection accuracy and pose estimation. Alternatively, in [18], drones use image processing on a QR code to get a quick pose estimation. By aligning the camera center point to the center point of the QR code, the agent can notably reduce error in the pose estimation derived from its relative position to the QR code.

Other works represent stigmergy via "pheromones" and apply it to different learning and optimization algorithms. For space applications, the research in references [19,20] uses pheromones as a basis to train individual satellites in a swarm. The research from [19] proposes a hierarchical deep reinforcement learning routing algorithm to train, control, and optimize LEO satellites for network routing. Alternatively, reference [20] uses behavior trees, based on hierarchical control, to differentiate behaviors the reactive agent uses as it identifies pheromones in the environment. Pheromone-based mechanics are implemented into deep reinforcement learning models to improve UAV efficiency in [21], significantly improving the efficiency of coverage applications. A new form of deep reinforcement learning based on the ant colony optimization covered in [1] is also used in [22]. In this case, the reward function is a local pheromone function that supports stigmergy between agents and encourages cooperation. Instead of improving the reinforcement learning aspect itself, the research presented in [23, 24] chooses to improve communication be-

tween agents by using indirect communication via stigmergy in the form of "digital pheromones". These digital pheromones work together with reinforcement learning models to enhance decentralized multi-agent collaboration.

More work that is close to this thesis applies stigmergy as a means of sharing environmental and navigational information. Virtual stigmergy is implemented in [25] to share tuples of key-value pairs, allowing agents in a swarm to share information about different agents and the environment they are in. On the other hand, in [26], the problem of localization is avoided by using RFID tags to contain internal maps. By having agents store internal map information directly into an RFID floor, other robots can navigate safely without needing to spend resources on localization and mapping. In another approach, [27–29] use physical codes to aid pedestrian indoor navigation. In [27, 28], indoor floor plans and features are built into QR codes to save money compared to GPS and RFID and provide more accurate navigational assistance where GPS and other services might fail. Unlike floor plans, [29] focuses on navigation network models to improve QR code placement, thus improving QR-based pedestrian indoor navigation.

While stigmergy is not the main focus in [30], they experiment with (virtual) stigmergy to facilitate local communication and effectively create a shared memory that improves the scalability of a decentralized system. Even if agents are having congestion issues hindering their movement, they can still propagate information to one another and update their belief information. Adding on that these communicated updates are synchronized across agents, virtual stigmergy becomes a way the system can avoid congestion and collision.

The research in [31] tries to make up for the disadvantages that simple reflex agents display in a multi-agent system with marker-based stigmergy. By having agents place color marks to indicate paths for deliveries, agents get information on task progress and path information without

needing any information on the environment itself. In this work specifically, the color marks operate similarly to pheromones, implicitly allowing for gradients to guide agents when they are close to a mark. Given a set of rules built around these color marks, the system of agents can more effectively complete their task with performance levels comparable to agent-based and model-based approaches.

2.3 Sampling based motion planning

Numerous important and innovative algorithms are in the field of sampling-based motion planning research that have greatly improved autonomous operations. As previously discussed in 2.1, the RRT algorithm from [3, 4] has formed a cornerstone to many modern SBMP algorithms. RRT simply provides a feasible path solution, but was extended and improved in [5] with the RRT* algorithm, introducing guarantees of asymptotic optimality to solutions at little extra computational cost. Additionally, nodes are iteratively rewired so they are always connected to the lowest-cost path available. Going further, the RRT# algorithm described in [6] builds on the ideas in RRT* by introducing cost propagation, in which information on new minimum-cost paths is shared with nodes in the same neighborhood. This change solved some of the problems that arose in the original rewiring logic from RRT* and further improved the optimality of solutions. This thesis uses RRT# as the motion planning in simulation trials, but does not propose any modifications or improvements to the algorithm itself.

There are many avenues available to improve shortest-path algorithms, but this thesis focuses on using geometry to reduce the exploration space. This is done in [32] by reducing the search space of Dijkstra's algorithm using different geometric containers. Pre-processed paths

are implicitly encapsulated in a geometric container to speed up the shortest-path computation, and these different geometric container types are compared and tested. Moreover, additional algorithms are proposed that handle a dynamic version of this problem in which the edge weighting and the containers may change or update.

Some work applies these containers to SBMP algorithms like RRT* as ellipsoids to improve their computation speed and asymptotic optimality. The C-FOREST algorithm in [33] is a framework that parallelizes single-query SBMP algorithms. Each tree communicates the current best path to other trees to achieve superlinear speedup for shortest path planning. When a solution is found, an ellipse is defined based on the length of this solution, prunes any existing nodes outside of it, and becomes the new boundary for drawn samples. Informed RRT*, proposed in [34], improves on the optimality in RRT* by directly sampling from an ellipsoidal subset once an initial solution is found. With the initial and goal states as the foci, the ellipsoidal subset is directly sampled to increase the probability of minimizing the path length in $\mathbb{R}^n$, although this method of "informed sampling" fails in non-Euclidean spaces. Ellipsoids are computed such that the eccentricity is equivalent to the ratio of the theoretical minimum cost to the cost of the current best solution. Batch Informed Trees (BIT*) is proposed in [35] and builds on Informed RRT* by introducing graph-search techniques and searching incrementally. BIT* draws samples in batches using an ellipsoidal heuristic, like in Informed RRT*. The process is iterative, and new batches are generated within the ellipsoid every time a new best solution is found. The research in [36] uses Informed RRT* as a basis for analyzing and avoiding the limitations that come from using Euclidean distance as an admissible heuristic. By sampling n-dimensional L^2 informed sets in its proposed method, it improves the speed of solution improvement while avoiding some of the issues from the minimum path-length curse of dimensionality. Similar to all the afore-

mentioned algorithms, Sampling-SEE is a sampling method that also fully leverages information from the current best path to reduce the space being explored [37]. This method combines path pruning, path merging, and biased reject sampling with the informed ellipsoid from Informed RRT* to derive an improved biasing ratio. This improved ratio is a dynamic biasing ratio that is path-informed, such that the convergence rate does not rely solely on the theoretical minimum cost.

2.4 Path reconstruction

Path reconstruction in the context of this thesis can refer to both restructuring and reproducing a path. With a unique approach to task planning, [38] abstracts skills into an abstract grid planning domain. Abstracting these skills builds a library of them over time and reduces the required computation to plan complicated tasks. In a graphical representation of this, the abstract skills from the library can be manipulated to satisfy given constraints. The framework from this research is extended in [39] such that past planning experience is used to derive more abstract skills. Reusable and generalizable abstract skills are extracted from past successful planning instances such that they can be adapted to future planning efforts. By being able to extract abstract skills for reuse, paths from past experience can be used and reconstructed to solve similar tasks in the future. Although it is not designed around SBMP, the research in [40] accelerates path planning in video games by abstracting environments into subregion graphs. The subregions in this graph contain waypoints, which can be adapted to an agent's dynamics. Planning within subregions using the abstract waypoints accelerates path planning computation while maintaining similar path lengths.

Also in the domain of skill learning and trajectory replication, [41] proposes a new "learning from demonstrations" algorithm based on kernelized movement primitives (KMP) to replicate trajectories. Given a task (such as replicating a specific trajectory), the algorithm learns the optimized hyperparameters necessary for the kernel to successfully reproduce a trajectory. In the same vein of reproducing trajectories, the work in [42] approaches the problem such that trajectories are encoded in a hidden Markov model. This model is then able to generate a reproduced trajectory by identifying "key points" in the original trajectory and interpolating them in a specific manner. With this sequence, agents can learn and reproduce a generalized version of complex trajectories.

For the domain of industrial robotics, the methods proposed in [43] combine a trajectory reproduction method with formation control to improve the stability and efficiency of industrial multi-agent systems. Obstacles and agents in the environment have artificial forces applied to them such that trajectory reproduction is achieved by tweaking the potential field parameters. Simplifying complex environments and cooperation through forces improved the accuracy of trajectory reproduction for industrial agents.

Regarding the alternative interpretation of path reconstruction, the path reconstruction method in [44] works to improve the initial path from a sampling-based motion planner. Voronoi tessellation is applied to the environment based on the given path and used to update the nodes, generating a graph. Then Dijkstra's algorithm is used to search the graph for the best path. In this manner, the path quality is improved to help avoid collisions and make the path more efficient.

2.5 Knowledge sharing

Knowledge sharing refers to sharing observed information and data between agents to supplement navigation. Regarding collision avoidance for mobile robots, the work in [45] uses a central server to collect data from connected agents and use it to predict collisions such that affected agents can be reset and redirected by the server to avoid them. The centralized approach allows agents to communicate indirectly by aggregating their data and using it to avoid predicted collisions.

Knowledge sharing can also be more indirect or abstract. In [46], robots and agents are viewed and judged by their "affordances" (capabilities and interactive properties) rather than their physical properties. These affordances, assigned to both agents and objects, are matched to prevent an agent from interacting with objects that it cannot handle. By sharing observed affordances, agents become capable of reassigning tasks earlier and avoiding unnecessary interactions.

In [47], agents in the same environment form a "sensor network" to facilitate communication and information sharing. Each agent has an internal map of the environment based on its own observations. If an agent observes any change to the environment compared to its internal map, the information is communicated to other agents so their map can be remotely updated as well. Taking into account the relevance of new updates, agents can update their internal maps without having to navigate to those areas themselves, improving the efficiency with which agents complete their tasks. The work in [48] makes considerable improvements by addressing the correspondence problem present in information sharing. The improved method accounts for heterogeneous agents and maps, as well as integrating considerations for uncertainty and dynamic

obstacles.

2.6 Mapping and localization

Both localization and mapping also take part in this thesis. Due to the accessibility, low cost, and orientation of QR codes, they can be reliably used to localize autonomous agents, as shown by some work in 2.2. A very direct approach is taken in [49], where coordinates, unit vectors, and QR code size are all stored as plain text in a QR code. This work had a few hardware issues that bottlenecked the precision and autonomy of this method. With this in mind, the experiments in [50] further test the use of QR codes as localizing landmarks, determining the practicality and effectiveness of QR-based self-localization for mobile robots. The results consider the method to be effective, but more processing power would be needed to handle larger QR codes. The work in [51] also uses QR codes for localization, but is more reliant on coordinate frame transformations and robot movement. The QR code only contains its coordinates, and the robot aligns itself to the QR code, allowing it to compute its position relative to it. Instead of using QR codes, [52] relies on a passive RFID floor to reduce localization error. Monitoring and comparing read times from nearby RFID tags on the floor improves the accuracy of the robot's localized pose.

Using QR codes or RFID tags helps with lower-cost localization, but some methods do not rely on any tags. In [53], robots are equipped with stereo vision sensors to build 3-D maps of their surroundings and localize themselves. A team of agents uses the proposed statistical sensor model to regularly share their observations and collectively build a probabilistic 3-D map. Agents are designed to prioritize high entropy areas to more efficiently map the entire region. Research

in [54] goes more in-depth into 3-D models and proposes a method for creating mapping models that allow seamless navigation across different types of environments. The unified 3-D space-based navigation model helps put indoor and outdoor map models within the same standards so that seamless path computation can take place between any space. Alternatively, any preliminary mapping involved with vision-based localization is avoided in [55]. Instead, localization is determined from a building information model, and the research proposes a method to bridge the gap between the model and real photos. With the proposed algorithm, the camera determines its pose from the information it derives from the building information model.

Using a more traditional vision-based system, a mapping method designed for mobile robots in office environments is proposed by [56]. In this proposal, agents combine the visual identification of ceiling light fixtures with robotic kinematics to localize themselves and navigate the indoor environment. In [57], inter-robot communication and landmarks are used as part of multi-robot SLAM to solve the problem of properly aligning maps from different robots. By communicating the relative pose to neighboring robots, agents can properly merge the independent maps together. Additionally, the inclusion of landmarks on bots helps identify overlaps and further improves map alignment.

2.7 Closely related work

There are a few pieces of research that combine these niches and align closely with the work in this thesis. A good example of this is the work in [58], where different teams of autonomous agents use RFID tags as a basis for coordination and navigation. The complex problems of task assignment and multi-agent path planning are solved by splitting the work into two phases.

During an initial local exploration, agents demonstrate stigmergy by placing RFID tags that build a graph agents can use for planning. With the environment generally mapped out via a graph, a globally monitored exploration in which agents are placed in the most effective and convenient locations by the monitoring process.

Similarly, a two-phase approach involving stigmergy with RFID tags is adopted in [26]. First, a team of agents places RFID tags in the environment containing navigational maps, where each tag is associated with one predetermined goal. Using this established RFID floor, future robots that already have a goal in mind can use the corresponding tag to navigate safely to their destination. No additional information is required beyond the RFID tag, thereby avoiding the complex problems of localization and mapping. The RFID floor utilizes stigmergy to share information, allowing future agents to replicate a previously defined path.

Instead of using physical means like RFID tags or QR codes, the proposed approach in [59] shares binary matrices representing the roads in an area, where obstacles are zeroes and roads are ones. Using a binary matrix approach reduces the volume of traditional mapping data while still maintaining its qualitative value in identifying obstacles and paths available to a vehicle. This approach helps a car find a short, feasible path to its goal without involving the more complex problems of comprehensive mapping.

The methods in [60] are similar to the thesis in that they involve incorporating QR codes as nodes into A*-based path planning. By treating QR codes as special nodes and building both a reward function and a heuristic to promote navigating through them, agents can verify and correct errors from dead reckoning, improving the accuracy of an agent's navigation. The QR codes themselves help correct an agent's pose, while the position of the QR code implicitly ensures some degree of navigational safety.

Chapter 3: Overview

We are motivated by a scenario in which two or more robots must asynchronously navigate an unknown environment to complete a task. It seems reasonable if the first robot in a new area could leave a message that would enable a second robot to benefit from the work and experience of the first. If agents are unable to communicate this information directly, it also seems reasonable to use physical codes to store information, which leads to the concept of stigmergy in motion planning and multi-agent systems.

While the idea of incorporating physical codes into motion planning can be done in numerous ways, we took the approach of splitting it into "phases" to emphasize and experiment with the asynchronous communication. Across two different agents, there are three primary phases: preliminary planning, post-processing, and re-planning. However, before discussing the research and experiment in detail, it is important to present the prerequisite nomenclature and problem definition in this work.

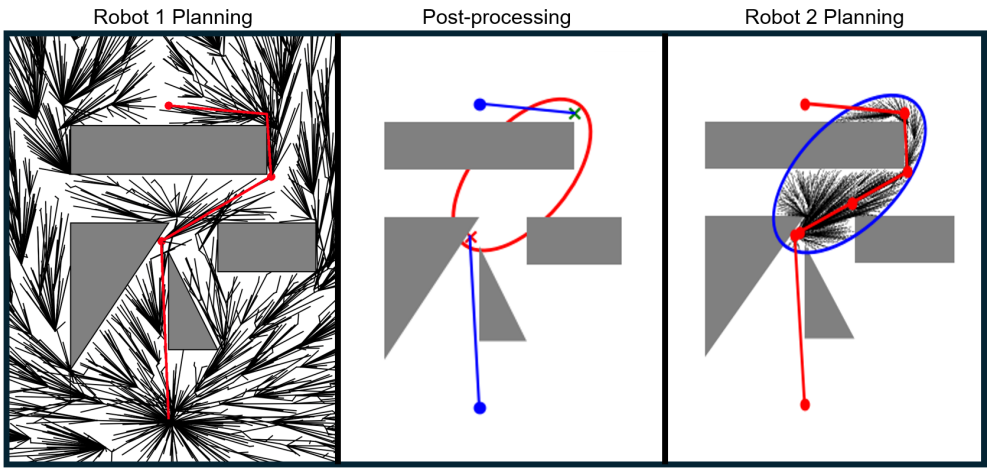


Figure 3.1: This visual overview visualizes the end result of each phase. The first robot (on the left) generates a feasible path, which then gets post-processed. In the post-processing (center box), the first robot prunes the path to eliminate nodes. If the path is too long to fit into a specific QR code, an ellipse is computed to eliminate more nodes from the path and encapsulate the area they were in. Finally, a second robot decodes this information and replans solely in the ellipsoid, if one exists.

3.1 Nomenclature

This section serves as a reference for all the unique variables and notation used in the following chapters and gives basic explanations of what they are. The primary focus is to define the variables unique to this research, but all variables and notation will be discussed as they come up throughout each section, too.

Variable	**Definition**	**Variable**	Definition
E	Set of edges	x_s	Start location
V	Set of nodes	x_g	Goal location
D	Ellipsoid	x_{rand}	Random sample
W	Workspace	x_Q	QR code location
C	Configuration space	f_i	Focus i of the ellipsoid
X	Space as a subset of W	P	Planned path
X_{Free}	Open space within X	P'	Sub-path from a path P
X_G	Goal region	$\bar{P}'$	Sub-path from a path P'
O	Obstacle space within X	$s_{i,j}$	Sum of the path lengths between nodes x_i and x_j
Q	Data encoded in QR code	ΔX	Degree of resolution
$\mathcal{G}$	Graph structure	$a,\ b,\ c,\ e$	Ellipsoid parameters
$\mathcal{T}$	Search Tree	θ	Ellipsoid orientation angle

3.2 Definition of the multi-agent stigmergy problem

We now formally define the problem that we are solving and thus state the assumptions, constraints, and desired solution.

3.2.1 Assumptions and Constraints

Here, we outline a number of assumptions that are used in the remainder of this thesis, as well as a number of constraints that we consider when solving the multi-agent stigmergy problem.

Each generated environment contains static 2-D polygonal obstacles that can overlap. In addition, the start and goal points remain constant, and each environment's obstacles are assumed to be completely constant between phases. In terms of experiment design, QR codes are assumed to be placed at the center of the goal region, and each robot operates within the same localization and coordinate system. Finally, we constrain each agent to be considered holonomic, such that $C \subseteq \mathbb{R}^2$ and the agent only considers its coordinates when planning. These paths may also be used directly for robots that can turn in place.

We also apply certain constraints to simplify the problem. Regarding the agent(s), any state must be within the free space to be considered valid. Regarding QR code restrictions, a maximum of 25 characters is enforced, although this constraint is flexible and depends on the QR code version being used. The 25-character limit corresponds to a version 1 QR code with low error correction. Additionally, while robots may apply multiple ellipsoids to a path, only one ellipsoid is used in the experiments.

3.2.2 Formal problem definition

With these preliminary assumptions and constraints in mind, the problem being studied is now defined. Let the workspace W be a space such that $W \subseteq \mathbb{R}^2$ and, consequently, the configuration space C is defined as $C \subseteq W$. All agents start at the point x_s in an unknown environment and must solve for a feasible path to the goal region X_G, which is centered at the point x_g. In the first phase, an agent identifies a feasible path P and then processes it into a shorter path P'. Then, an ellipsoid D is greedily placed over the path P' such that the characters satisfy the character limit when put into Q. With Q computed, a QR code is generated and placed

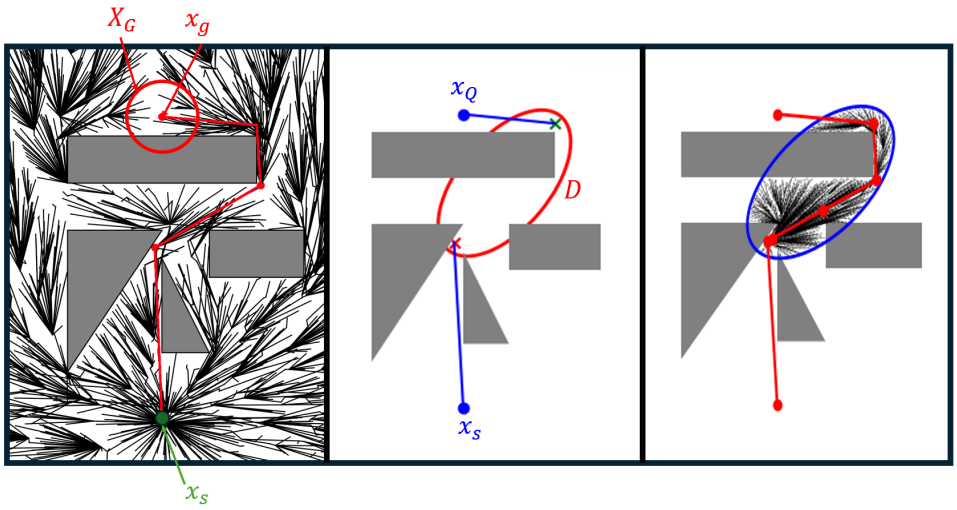


Figure 3.2: Labeled version of figure 3.1 that shows where each variable in the problem is located. On the left, the first robot generates a path P. Path P is modified and turned into a path P', which can contain an ellipsoid D (shown in the center). This path P' and ellipsoid D is used by a second robot (on the right) to complete a new path between x_s and x_g similar to the original P.

at or near enough to the location x_g that it can be sensed from the new starting location x_Q. Given the same x_s, x_g, and environment, an agent must parse the data within Q and find a feasible path connecting x_s and x_g. If there is one or more ellipsoids within the data from Q, the agent must also find a feasible path that connects the foci within it to identify a piecewise continuous path.

Chapter 4: Approach

This chapter provides a preliminary explanation of the methods proposed in this thesis. We approach the stigmergic motion planning by splitting the work across two different agents. The first agent to explore an area performs its own preliminary motion planning in the environment, without relying on any additional information from past agents. Based on its results, the first agent processes this information to fit it into a QR code Q that will be physically printed. In this process, both path pruning and ellipsoids are applied to save memory when storing data in Q. Any future agents that encounter Q will decode its information and attempt to utilize it to expedite the motion planning process and reduce their search area. The following sections give a brief introduction to the work done in each phase of the problem.

4.1 Preliminary motion planning

In what has been defined as the first phase, an agent uses SBMP to navigate an unknown environment and reach some arbitrary goal region X_G. Algorithm 1 contains high-level pseudocode of the RRT# algorithm to support an understanding of what motion planning looks like functionally. Once the planning is complete, the agent takes time to process the raw path P into a modified path P'. Pruning P into P' based on line of sight is a preliminary step toward satisfying constraints on Q before trying to use an ellipsoid.

Algorithm 1 RRT# Main Body

procedure RRT#($x_{\text{init}}, X_{\text{goal}}, X$)
  $V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$
  $\mathcal{G} \leftarrow (V, E)$
  for $k = 1 \rightarrow N$ **do**
    $x_{\text{rand}} \leftarrow \texttt{Sample}(k)$
    $\mathcal{G} \leftarrow \texttt{Extend}(\mathcal{G}, x_{\text{rand}})$
    $\texttt{Replan}(\mathcal{G})$
  end for
  $(V, E) \leftarrow \mathcal{G}$; $E' \leftarrow \emptyset$
  for all $x \in V$ **do**
    $E' \leftarrow E' \cup \{(\texttt{Parent}(x), x)\}$
  end for
  return $T = (V, E')$
end procedure

As previously acknowledged, this work relies on SBMP algorithms to plan paths between the start and goal. Any SBMP generally works with the algorithms proposed in this work, but RRT# is used as the primary motion planning algorithm for the experiment. Any alterations or additions to these algorithms are later defined in (section). While this work uses only RRT#, theoretically, any form of algorithmic motion planning can be used in its place, whether it is sampling-based or not. However, for reasons later demonstrated in the next chapter, C and W must remain Euclidean.

4.2 Post-processing

The second phase focuses on the process of formatting and encoding motion planning data into Q. This phase involves encoding the compacted data in a QR code to be printed at x_Q. The storage limits of a QR code (Q) are based on its version, error correction level, and the data type of the information stored in it. To reduce the data to satisfy the constraints on Q, we propose using an ellipsoid to greedily expand over a sub-path $\bar{P}'$ and take its place. Once the storage limit

is satisfied according to whatever encoding standard is determined, Q is generated and inserted into a QR code. The details and code involved in this process are later explained in 5.3.

4.2.1 Ellipsoid construction

Part of the aim of this work is to use geometry as a tool to encapsulate valuable motion planning data so it can be shared more easily in restricted or communication-denied environments. We chose to use an ellipse due to its unique properties, which are detailed in the next chapter. The main idea is that ellipsoids can be constructed and sized to be proportional to the length of a given path. With this in mind, we can use them to encapsulate a specific sub-path $\bar{P}'$ such that the amount of data required to detail the path is reduced. It is important to note that if a given path immediately satisfies the data limit, no ellipse is computed and encoded into Q.

In this work, we take a greedy approach to computing and expanding the ellipsoid over a sub-path $\bar{P}'$. If we use nodes along the path as potential foci, it is necessary to have some metric(s) to differentiate nodes and judge them. In the greedy ellipse expansion, nodes are judged by data collected from an occupancy grid that tracks every sample location within a set resolution ΔX and counts how often it results in a collision. A node is therefore differentiated and 'scored' by the sampled collision rate in its vicinity. The node with the most collisions is used as the starting point, based on the assumption that replanning should be focused on the difficult regions rather than the easier ones.

Two neighboring nodes from P' are selected as the starting foci for the ellipsoid. Then, the algorithm iteratively compares the two neighboring path nodes outside of those current foci until the constraints on Q are satisfied. When the two outside nodes are compared, the node with

more recorded collisions near it is selected as the next focus candidate. In this way, the ellipsoid is greedily expanded along the path, encapsulating nodes that get internalized during the process.

Selecting the focus for the ellipsoid is only one part of the ellipsoid construction. To ensure that the ellipsoid is sized proportionally, at least one of the parameters must be defined along with the foci. Each loop continuously sums the path length of the segments that are within the ellipsoid, determining the value $s_{i,j}$. The value $s_{i,j}$ is then used to proportionally determine the length of the major axis and therefore the size of the ellipse. Because the ellipsoid is constructed around $\bar{P}'$, the agent knows that a solution must exist and will be found given enough time.

4.2.2 QR codes

Many different fiducial markers are used regularly, but this work is based on the use of QR codes due to their unique orientation and flexibility in storing data. QR codes can come in 40 different sizes, storing up to 152 bits in the smallest size and up to 23,648 bits in the largest. Every QR code also uses either low (L), medium (M), quartile (Q), or high (H) error correction. This error correction is the Reed-Solomon code and is added to the original data upon generation. These levels of error correction correspond to being able to recover 7, 15, 25, and 30 percent of the data, respectively. In addition to the variety of sizes and error correction levels, data can be stored in a QR code in four different forms: numeric, alphanumeric, binary, or kanji. Each version of a QR code has a specific limit of how many bits of data it can hold (depending on the level of error correction, too). The algorithms and experiments in this work store data as an alphanumeric string, but all types are viable and may have a use case. However, each type must be briefly covered for the sake of background knowledge and clarity.

For consistency, consider a version 1 QR code with L error correction. For this specific QR code, there can be no more than 41 numeric characters, 25 alphanumeric characters, 17 binary characters, and 10 kanji characters. For the numeric type, any number containing only digits 0-9 is encoded within the QR code. In this example, any number no more than 41 digits long would be satisfactory. The alphanumeric character type can be likened to a 'char' or 'string' type in programming contexts. The string '5 5 0 9 7 0 15 20 0 20 25' contains alphanumeric characters (whitespace included) that satisfy the derived 25-character limit. Letters, numbers, and special characters are all valid with this type and have the most symbolic capabilities. Next, the binary type contains raw binary data. This binary data can come from files, images, and more. Binary data in this example would have to be no more than 17 bytes. Finally, Japanese kanji characters are also a valid type to encode within a QR code. Although kanji is generally considered the least efficient in regard to data consumption, the characters can be more complex than the other three modes.

For simplicity in standardizing the data format, this work uses the alphanumeric character type. Using whitespace delimiters allows us to easily organize data values and simplify the future parsing process. Although you may be able to get more raw data from the other types, more complex encoding methods would be required. The chosen format and its reasoning are discussed in more detail in the next chapter.

4.3 Using stored data

For the final phase, future agents read the provided data Q, parse the encoded information, and reconstruct the original path. If an ellipsoid is encoded in Q, the agent will re-plan within the

ellipsoid itself, and any other segments outside of that are kept as is. These parts are then stitched together to create a complete path connecting the start and goal, if one exists. It is necessary to note that if the path in Q satisfies the constraints on Q as it is, that path is automatically used and no extra planning takes place.

Before using the data from Q, the data must be parsed and decoded. We solve this problem by encoding the path data in Q in a standardized format. Considering that the smallest version of the QR code is used in the experiment, we first convert the coordinates of all the path nodes to integers to save characters. Since we also only use coordinates in C and operate in a Euclidean space, the data is written as a string and follows a specific order. If we assume that a node x_i in the path is a 2-D coordinate and all terms contain integers, data in Q uses the following format:

$$\text{“}x_s\ s_{0,1}\ x_1\ s_{1,2}\ x_2\ s_{2,j}\ \text{...}\text{”}$$

Beginning with the start location x_s, the data alternates between a path node and the corresponding term $s_{i,j}$ in between. We nominally set every $s_{i,j}$ as zero if they are neither of the foci. If there is a non-zero $s_{i,j}$ in Q, the x_i coordinates before and after it are assumed to be f_1 and f_2, respectively. In this manner, we can decode which two nodes are the foci and what the total path length was between them, allowing the desired D to be reconstructed. Also, due to the assumption $x_Q = x_g$, the goal location can be omitted, as it is implied whenever the QR code is read. Using this standardized format and delimiting with spaces makes it simple to parse the motion planning variables related to $\bar{P}'$ and D.

When an ellipsoid is extracted from Q, the agent begins planning from x_g and attempts to reconstruct a path back to the original x_s. First, the path nodes that are not identified as f_1 or f_2 are saved for the final path. The agent then uses SBMP to plan a path between f_1 and f_2.

Using some of the properties of ellipsoids, the agent can sample points directly from the ellipsoid, avoiding much of X_{Free} and prioritizing the area around the sub-path.

Chapter 5: Processing motion planning information

This chapter discusses the properties and definitions supporting the proposed algorithms in greater detail, demonstrating how they work together. Ellipsoids are used to replace segments of a feasible path and reduce the storage size enough for it to fit within a QR code. Certain properties of an ellipsoid are leveraged to maintain some implicit information about the path it represents. Additionally, a modified version of an occupancy grid is used to identify difficult planning areas and aid in foci selection when computing an ellipsoid.

5.1 Ellipsoid properties

There are a few key properties of a 2-D ellipsoid that this work relies on. This section defines an ellipsoid and discusses the properties used to construct them and leverage them for motion planning.

5.1.1 Ellipsoid definitions and properties

Definition 1. *A 2-D ellipsoid D is defined as the set of all points where the sum of the distances to each foci remains constant.*

$$D = \{q \in \mathbb{R}^2 | \ \|q - f_1\| + \|q - f_2\| \leq 2a\} \tag{5.1}$$

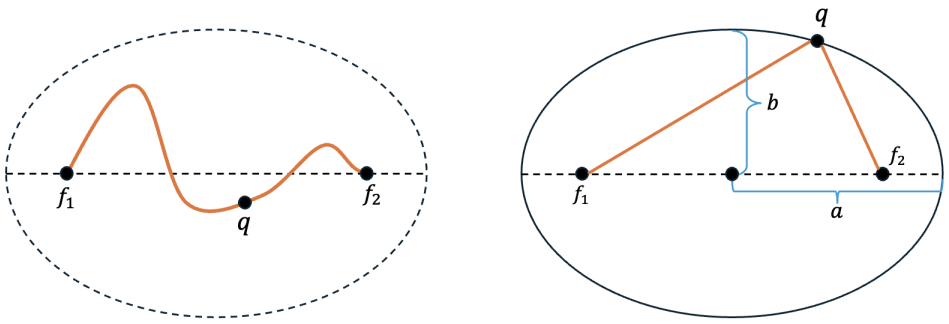


Figure 5.1: Construction of an ellipsoid from an arbitrary path

Looking at 5.1, we can visualize this locus of points. In this definition, D represents the ellipsoid, f_1 and f_2 are the foci, and q is a point in the locus. This definition is the crux of how the ellipsoid will be used and computed. Ellipsoids have multiple unique properties that will be exploited to preserve the existence of a path under the constraints of the Q.

To construct an ellipsoid that covers a trajectory, only the foci and one ellipsoid parameter (a, b, c, e) must be known to define an ellipsoid. If we assume f_1, f_2, and the semi-major axis a are known, we can derive the rest of the ellipsoid parameters b, c, e, and θ, where θ is the angle between the semi-major axis and the positive x-axis.

For any arbitrary 2-D ellipsoid, it is necessary to find the center point (h, k). With the foci already known, point (h, k) is simply the midpoint between f_1 and f_2:

$$h = \frac{1}{2}(f_{1_x} + f_{2_x}) \tag{5.2}$$

$$k = \frac{1}{2}(f_{1_y} + f_{2_y}) \tag{5.3}$$

The parameter c, defined as the distance from the center point to a focus, can now be derived

using the center point (h, k):

$$\sqrt{(f_{2_x} - h)^2 + (f_{2_y} - k)} \tag{5.4}$$

Consequently, the semi-minor axis b is calculated using a and c via the Pythagorean theorem:

$$b = \sqrt{|a^2 + c^2|} \tag{5.5}$$

Similarly, the eccentricity can be calculated using a and b:

$$e = \frac{\sqrt{|a^2 - b^2|}}{a} = \frac{c}{a} \tag{5.6}$$

Finally, the tilt θ can be computed using an inverse tangent, accounting for the quadrant:

$$\theta = \arctan(\frac{f_{2_y} - f_{1_y}}{f_{2_x} - f_{1_x}}) \tag{5.7}$$

However, simply defining the parameters is not enough. An ellipsoid cannot be guaranteed to completely contain a path if it is not sized proportionally. We propose correlating a to the summed path length of the path that is being covered. If we construct the ellipse such that its a is scaled to be half of the summed path length $s_{i,j}$, the original path is guaranteed to be completely within the ellipse within the defined precision ΔX.

Lemma 5.1.1. *According to a given precision ε (numerical or rounding error), an ellipsoid D is guaranteed to contain a sub-path $\bar{P}$ if $s = 2a$.*

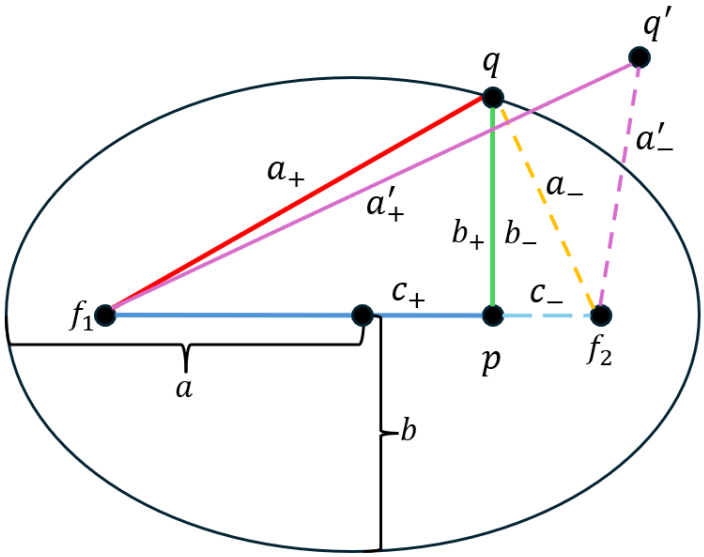


Figure 5.2: Ellipsoid formed from an arbitrary path with a point q on the boundary

Proof. Let $\bar{P}$ be a nontrivial path between the foci f_1 and f_2 in an ellipsoid D, and s be the total path length of $\bar{P}$. Assigning the relationship $s \equiv 2a$, assume an arbitrary point q along $\bar{P}$ is positioned to be along the boundary of the ellipsoid D.

If we split the resulting triangle between q, f_1, and f_2 into two right triangles (see figure 5.2), we develop the following equations:

$$\ell = \|f_2 - f_1\| = c_+ + c_- \tag{5.8}$$

Additionally, we know by inspection that $b_+ = b_-$. We can further define a_+ and a_-:

$$a_+ = \|q - f_1\| \tag{5.9}$$

$$a_- = \|q - f_2\| \tag{5.10}$$

Substituting this definition into the ellipsoid equation (eqn. 5.1), we know that the sum of a_+ and a_- must remain constant and be equal to $2a$, where a is the semi-major axis.

$$a_+ + a_- = 2a \tag{5.11}$$

Recall that the boundary of a 2-D ellipsoid can be defined by the set of q that satisfy this relationship. As such, if we assign the semi-major axis to be equivalent to the length of $\bar{P}$, we can guarantee that any point on the path P will remain within the constructed ellipsoid D

$$a = \frac{a_+ + a_-}{2} = \frac{s}{2} \tag{5.12}$$

The same relationship can be used to determine whether an arbitrary point q' lies inside the boundary of D or not. Using equation 5.9, if a point q' is inside the boundary, it will satisfy this triangle inequality that defines the ellipsoid boundary:

$$a'_+ + a'_- \leq s = 2a \tag{5.13}$$

If this is not satisfied, the point q' violates the above inequality and therefore lies outside of the boundary that defines D. □

The rounding error can be propagated to identify the effects of a numerical error ε in s and on the resulting ellipsoid. Consider an ellipsoid with foci separated by the distance $2c$. Assume that $s > 2c$ so the ellipsoid is proper and b is positive. Using an ellipsoid defined by $s = 2a$, its parameters are defined as

$$s = 2a \Rightarrow a = \frac{s}{2},\ b = \sqrt{a^2 - c^2},\ e = \frac{c}{a}.$$

To propagate an error in s, suppose s is rounded to $\tilde{s} = s + \varepsilon$ where $|\varepsilon| \leq \Delta$ is the rounding error. We can use first-order differential propagation on the ellipsoid parameters to identify the error in its area.

Treat ε as small and expand the above equations to first order. Starting with the semi-major axis value a,

$$a = \frac{s}{2} \Rightarrow \tilde{a} = a + \frac{\varepsilon}{2},\ \delta a = \frac{\varepsilon}{2}.$$

For semi-minor b,

$$db = \frac{a}{b} da \Rightarrow \delta b \approx \frac{a}{b} \delta a = \frac{a}{b} \cdot \frac{\varepsilon}{2} = \frac{a}{2b} \varepsilon.$$

Equivalently, using $a = s/2$,

$$\delta b \approx \frac{s}{4b}\,\varepsilon, \quad \frac{\delta b}{b} \approx \frac{a}{2(a^2 - c^2)} \varepsilon = \frac{s}{s^2 - 4c^2} \varepsilon.$$

Then, with the eccentricity $e = \frac{c}{a}$ where c is fixed,

$$de = -\frac{c}{a^2} da = -\frac{e}{a} da \Rightarrow \delta e \approx -\frac{e}{a} \cdot \frac{\varepsilon}{2} = -\frac{e}{2a} \varepsilon = \frac{c}{2a^2} \varepsilon,$$

And putting this in the relative form, it becomes

$$\frac{\delta e}{e} \approx -\frac{\delta a}{a} = -\frac{\varepsilon}{s}.$$

Using the relative errors of a and b, the relative error for the ellipsoid area is approximated.

$$\frac{\delta A}{A} = \frac{\delta a}{a} + \frac{\delta b}{b} \approx \frac{\varepsilon}{s} + \frac{s}{s^2 - 4c^2}\varepsilon = \varepsilon(\frac{1}{s} + \frac{s}{s^2 - 4c^2}).$$

Rounding s affected the ellipsoid's area A and its parameters a, b, and e, while the orientation and location of the ellipsoid are unaffected. Both b and A are especially sensitive, increasing significantly as $s \to 2c$.

5.1.2 Sampling ellipsoids directly

In addition to the ability of an ellipsoid to encapsulate a path, it can be sampled directly instead of relying on rejection sampling. This section discusses the ellipsoidal sampling methods used in this work. An ellipsoidal sampling method is employed by the robot(s) reading Q, as points are sampled directly from the decoded ellipsoid instead of the entire environment.

We take the approach of using polar coordinates to uniformly sample a unit disk (or a unit ball, for higher dimensions). An affine transformation is then applied to the unit disk, transforming it into the defined ellipsoid D. This approach quickly and uniformly samples an ellipsoid based on its parameters ($a,\ b,\ \theta$, etc.).

Algorithm 2 shows this exact procedure. When the polar coordinates are sampled, the unit radius is square-rooted to make the unit-disk area uniform. Then, the unit disk sample is transformed via the input ellipsoid center point (h, k) and its axes parameters a and b, resulting in a uniform sample from the ellipsoid.

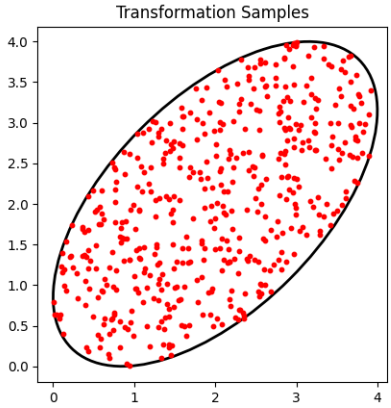


Figure 5.3: Example of drawing direct samples using algorithm 2. Samples are drawn from a unit-disk, which is then transformed into an area-uniform ellipse.

Algorithm 2 Sampling from an Ellipsoid

Require: Ellipse object D with center (h, k) and parameters a, b, θ_D

function `SampleEllipse`(D)
Sample $r_i \sim \sqrt{\text{Uniform}(0, 1)}$
Sample $\theta_i \sim \text{Uniform}(0, 2\pi)$
$x_i \leftarrow h + a \cdot r_i \cos(\theta_i) \cos(\theta_D) \; - \; b \cdot r_i \sin(\theta_i) \sin(\theta_D)$
$y_i \leftarrow k + a \cdot r_i \cos(\theta_i) \sin(\theta_D) \; + \; b \cdot r_i \sin(\theta_i) \cos(\theta_D)$
$x_{rand} \leftarrow (x_i, y_i)$
return x_{rand}

A similar approach, shown in algorithm **??**, uses the implicit Cartesian form of an ellipse and applies Cholesky factorization [11]. The Cholesky factorization maps the points sampled from the unit disk to the ellipsoid being sampled. In cases where the ellipsoids are defined analytically instead of by their parameters, it may be more applicable to define an ellipse by its general implicit equation form:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F \leq 0 \tag{5.14}$$

The matrix form Δ of the ellipsoid is constructed using the coefficients from this implicit

form. Then, Σ is computed using Δ to map the unit disk to the subject ellipsoid. This matrix is then factorized using Cholesky decomposition. Eigen decomposition can be used in case of any small numerical errors or asymmetries in the matrix.

It must be noted that both methods work and scale with higher dimensions, although only a two-dimensional version is used in the experiments in this thesis. Additionally, these methods are generally restricted to Euclidean spaces, particularly due to the construction of ellipsoids that relies on Euclidean distance. The work in [11] provides a more detailed examination of this topic, comparing and improving existing methods for generating uniform samples within multi-dimensional ellipsoids.

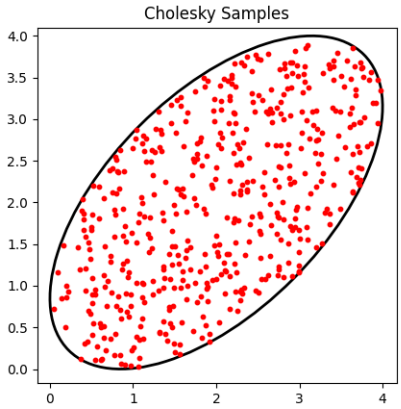


Figure 5.4: Example of drawing direct samples using algorithm **??**. Cholesky decomposition is used to map the unit-disk to the desired ellipse. A sample is randomly drawn from the unit disk and mapped to the ellipse.

5.2 Occupancy grids

This section discusses how occupancy grids are applied to ellipsoid construction. An important aspect involved in using an ellipsoid to encapsulate a path segment is being able to judge and prioritize the importance of nodes. This experiment approaches this by using a modified

discrete occupancy grid.

The general philosophy of our approach is to prioritize areas in the environment where there may be more obstacles. Prioritizing the 'difficult' regions around the planned path means more time can be dedicated to searching them in the future, when other robots plan in the same space. Additionally, using an ellipsoid to encapsulate these prioritized regions uses less memory than the original path, while also retaining some of the inherent informational value.

Instead of using a traditional binary occupancy grid, a modified version is used that tracks the success rate of samples within the environment, using a predetermined level of discretization and precision. Each space in the occupancy grid keeps count of how many samples fall within that cell, as well as how many of them resulted in a collision. During motion planning, every new sample is counted by the nearest cell. Using some occupancy grid M, the corresponding cell $(i,\ j)$ is updated every iteration, as demonstrated in algorithm 3. This update can be easily integrated into a motion planning algorithm.

Algorithm 3 Example Occupancy Grid Cell Update

$M[i][j] \leftarrow M[i][j] + 1$
$M.samples\ \ +=\ 1$
if collision **then**
    $M.total_collisions\ \ +=\ 1$
    $M[i][j].collisions\ \ +=\ 1$
end if

Nodes can then be scored and judged using this modified occupancy grid structure. When the final path from the motion planning phase is saved, this occupancy grid is used to correlate grid spaces to nearby path nodes and quantify a basic heuristic for the collision rate for each node. This heuristic is then applied to organize and prioritize which nodes from the path should be encapsulated by the ellipsoid.

5.3 Generating the QR code

The next step after generating the occupancy grid is to use the information to start processing the given path and prepare it for encoding in Q. This section presents the logic and methods used to process the given path, with the ultimate goal of encoding a feasible path solution into the environment. This post-processing phase involves pruning the original path, scoring path nodes, and expanding an ellipsoid D.

Algorithm 4 Generating the QR string Q

Require: Path P, obstacles O, character limit K

function `GenerateQR`(P, O)
Convert P coordinates to integers ▷ Depends on precision level for Q data
$P' \leftarrow$ PRUNEPATH(P, O)
Remove x_g from P'
$K_{qr}, Q \leftarrow$ CHECKQR(P', f_1_idx, f_2_idx, K)
$K_{qr} \leftarrow$ length(P')
scores $\leftarrow$ SCORE(P', M, δ)
if $K_{qr} > K$ **then** ▷ Q_{test} length exceeds K
    $\bar{P}', D \leftarrow$ GREEDYELL(P, $scores$, K)
    Build Q using builder
    return Q
else if $\ell \leq K$ **then** ▷ P' within K
    Build Q directly from P'
    return Q
end if

The function GENERATEQR in algorithm 4 presents the methods used to compute the data string stored in Q. According to a predetermined precision level, the given P undergoes path pruning based on line-of-sight to shorten the number of nodes in the path and create a newly modified path P'. Due to the QR code being placed at the original goal node x_g, the goal node is also removed during path pruning to free up potential space in Q.

With the path pruned, the encoded version of the pruned path P' is tested to see if it fits Q.

If it is too long, the GREEDYELL is called to iteratively construct an ellipsoid D and $\bar{P}'$ until it can successfully be encoded into Q. Once its function is successful, $\bar{P}'$ and D are encoded into Q for future robots to use. The following subsections discuss the smaller steps and subroutines found in the GENERATEQR function.

5.3.1 Node scoring

Before encoding the data into Q, some metric or score is needed to judge the points and filter out less valuable nodes. This section presents the method used to compute the scoring metric. Nodes within the input path have their corresponding sampling collision rate computed. Algorithm 5 demonstrates this proposed scoring method.

Algorithm 5 Node scoring

Require: Path P, occupancy grid M, steering radius δ

function `Score`(P, M, δ)
Initialize empty list $scores$
$T \leftarrow M.total_collisions$
for each node v in P **do**
  Convert x, y to integers ▷ Depends on precision level for Q data
  $score \leftarrow 0$
  for $dx \leftarrow -\delta$ to δ **do** ▷ Sum all the collisions from surrounding cells
    for $dy \leftarrow -\delta$ to δ **do**
      $nx \leftarrow x + dx$, $ny \leftarrow y + dy$
      if $x_{min} \leq nx < x_{max}$ and $y_{min} \leq ny < y_{max}$ **then**
        $score \leftarrow score + M[nx][ny].collisions$
      end if
    end for
  end for
  Append $\frac{score}{T}$ to $scores$
end for
return $scores$

In this algorithm, each node in the path sums the total number of collisions from each cell within δ, provided by the occupancy grid M. After each node is assigned a score, the values get

normalized by the total number of samples in the entire environment. The scores are then used to select the foci f_1 and f_2 for the ellipsoid D and expand it as needed.

5.3.2 Path pruning

The next part of the feasible path post-processing is filtering out path nodes based on their collision rates. This section presents the steps taken to manipulate a given path to fit the constraints on a QR code Q. The primary methods for achieving this goal are path pruning and ellipsoids.

Path pruning is applied before ellipsoid construction to naively simplify the path as a first pass toward satisfying the constraints on Q. In the context of sharing motion planning information between agents, it stands that it is generally better to share an explicit path whenever possible rather than an incomplete version with an ellipsoid. Path pruning based on line-of-sight increases the odds of this happening.

Algorithm 6 Path pruning

Require: Path P, obstacle space O

```
1: function PrunePath(P, D)
2: if length(P) ≤ 2 then
3:     return P
4: end if
5: Initialize P′ with the first node of P
6: i ← 0
7: while i < length(P) − 1 do
8:     j ← length(P) − 1
9:     while j > i + 1 do
10:        L ← segment from path[i] to path[j]
11:        if L does not intersect any obstacle in O then
12:            break inner loop
13:        end if
14:        j ← j − 1
15:    end while
16:    Append P[j] to P′
17:    i ← j
18: end while
19: return P′
```

If path pruning reduces the number of nodes in P enough such that it fits in Q, the construction of an ellipsoid can be avoided entirely, saving valuable computation time and resources. Algorithm 6 demonstrates how path pruning is applied to the given path P. Starting from the first node, the algorithm iteratively tries to safely extend each node in P as far as possible while avoiding collisions. This results in a new path P' that is used to populate Q.

5.3.3 Ellipsoid Construction

The foci are then greedily expanded until the stored data can satisfy the limit on Q. Algorithm 7 judges the nodes in P' as candidates for the foci of an ellipsoid D. The algorithm begins by temporarily assigning foci f_1 to be the node with the highest collision rate. Similarly, f_2 is iteratively assigned as one of the path nodes adjacent to f_1 based on the collision rate. Every

node in P' is a potential focus candidate, and the foci f_1 and f_2 iteratively move outward to each adjacent path node until enough nodes are internally removed and the storage limit of Q is satisfied. The node chosen as the next new focus is based on whichever node has the higher collision rate.

Algorithm 7 Greedy expansion of an ellipsoid D

Require: Path P', $scores$, character limit K

function `GreedyEll`(P', $scores$, K)
if P' or $scores$ empty **then**
    return None
end if
if length(P') ¡ 2 **then**
    return None
end if
max_index ← index of maximum score in path
neighbors ← valid neighbors of max_index
if neighbors empty **then**
    return None
end if
best_neighbor ← neighbor with highest score
(f_1_idx, f_2_idx) ← sorted pair (max_index, best_neighbor)
while CHECKQR(P', f_1_idx, f_2_idx, s, K) exceeds K **do**
    if indices invalid **then**
        return None
    end if
    candidates ← expand left of f_1_idx or right of f_2_idx
    if candidates empty **then**
        break
    end if
    best ← candidate with highest score
    if best $< f1_idx$ **then**
        f_1_idx ← best
    else
        f_2_idx ← best
    end if
end while
if indices invalid **then**
    return None
end if
$s \leftarrow \|f_2 - f_1\|$ ▷ s kept to chosen precision level
$D \leftarrow$ ELLIPSOID(f_1, f_2, s)
return D

Using the foci as the medium, a temporary ellipsoid is expanded to cover the given path and encapsulate nodes, resulting in a new path $\bar{P}'$. As a result, the path $\bar{P}'$ is guaranteed to fit within Q based on the selected encoding method. Moreover, the data in Q can then be used to

recreate a feasible path. When another robot decodes the information from Q, it can exploit it to get a head start on planning a feasible path through the same path planning problem without any additional prior knowledge of the environment.

Chapter 6: Stigmergic motion planning

This chapter serves to explain how the stigmergic features previously generated are leveraged to improve planning and navigation for future agents reading the data. Data from the original processing phase is parsed based on an established standard, referenced in 4.3. The data from Q is decoded according to the chosen encoding format, giving an agent three possible planning cases. If no ellipsoid is present, the agent will use the decoded path without any extra planning. If there is an ellipsoid encoded, path planning is carried out within that ellipsoid, and the result is stitched along with the other coordinates. Otherwise, the agent will motion plan without any extra information.

6.1 Parsing data

Stigmergy in this work is represented through a QR code Q and is how agents can use the environment to store and share data with other agents. For it to be sufficient, agents must agree on some standard or method of both encoding and decoding information. To facilitate the proof of concept, we use a simpler approach to standardizing the shared data. This section explains the structure and assumptions used to encode data into Q.

Introduced in 4.3, data is encoded into Q as a whitespace-delimited string, with variables taking a specific order. Any data in Q begins with a coordinate x_s and is followed by a series

of modified tuples. This pattern continues until the predetermined QR code character limit is reached. Additionally, note that x_g is omitted from the path stored in Q because x_Q is always placed at the goal x_g. Therefore, Q always ends with the coordinates of a path node, or some path length variable $s_{i,g}$. Assuming that x_i is a 2-D coordinate somewhere in a path P, the data string Q follows this format:

$$\text{“}x_s\ s_{0,1}\ x_1\ s_{1,2}\ x_2\ s_{2,j}\ \ldots\text{”}$$

Algorithm 8 presents the method used to generate these Q strings in practice. Tuples in the form (x_i, y_i, $s_{i,i+1}$) are derived from each of the path nodes and contain its coordinates and a path length to the next node in the list. Because the robot's C-space is assumed as two-dimensional in this work, only the x and y coordinates are placed in the tuple. Given a list of tuples from a path, the tuples are flattened and formatted into a whitespace-delimited string.

This algorithm splits the process into multiple parts and builds the string in order, starting with the start node x_s. All the path nodes preceding f_1 are then added to Q. If f_1 and f_2 are defined, they are added next, with s, the summed path length between them, being placed between the two respective coordinates. If x_g happens to be assigned as f_2, it is still excluded from Q, and the Q will end with the corresponding s value that will define the chosen ellipsoid. If there are any additional nodes after f_2 and space permits it, tuples after f_2 are added to Q as well. By default, the path length variable s in each tuple is set to zero. If any value is non-zero, it is implied that the two coordinates around it are the foci f_1 and f_2. While this means fewer nodes fit in Q overall, including them makes identifying and extracting the proper coordinates for each node much easier. Algorithm 8 is used in 9, which is used by the algorithms in 5.3 to build a temporary Q and test its validity iteratively.

Algorithm 8 QR code builder

Require: Path P', indices f_1_idx, f_2_idx, parameter s_value, character limit K

function `BuildQRString`(P', f_1_idx, f_2_idx, s, K)
$P' \setminus x_g$
Clamp f_1_idx, f_2_idx to valid range and ensure $f_1_idx \leq f_2_idx$
Initialize empty list Q
▷ 1) Pre-f_1 tuples
for $i \leftarrow 0$ to $f_1_idx - 1$ **do**
    tuple_$P'[i] \leftarrow (x_i, y_i, 0)$
    if Q with new tuple exceeds K **then**
        return (None, $K + 1$, None)
    end if
    Append tuple_$P'[i]$ to Q
end for
▷ 2) f_1 tuple
$f_1 \leftarrow P'[f_1_idx]$
tuple_$f_1 \leftarrow (f_1.x,\ f_1.y,\ \text{int}(s))$
if Q with tuple_f_1 exceeds K **then**
    return (None, K+1, None)
end if
Append tuple_f_1 to Q
▷ 3) f_2 tuple unless $f_2 = x_g$
$f_2 \leftarrow P'[f_2_idx]$
if $f_2 \neq x_g$ **then**
    tuple_$f_2 \leftarrow (f_2.x,\ f_2.y)$
    if Q with tuple_f_2 exceeds K **then**
        return (None, $K + 1$, None)
    end if
    Append tuple_f_2 to Q
end if
▷ 4) Post-f_2 tuples
for $i \leftarrow f_2_idx + 1$ to length(P')-1 **do**
    tuple_$P'[i] \leftarrow (x_i,\ y_i,\ 0)$
    if Q with tuple_$P'[i]$ exceeds K **then**
        return (None, $K + 1$, None)
    end if
    Append tuple_$P'[i]$ to Q
end for
▷ Output represented path
$\bar{P}' \leftarrow P'[start : f_1_idx]\ +\ P'[f_2_idx : end]\ +\ x_g$
$K_{qr} \leftarrow$ length of string representation of Q ▷ K_{qr} is the actual length of the string
return (Q, K_{qr}, $\bar{P}'$)

It must be noted that whitespace counts toward the character count and that the precision for each value can be determined by the user. The values present in the experiments are converted to be integers, prioritizing the quantity of data points over their precision, especially with such a small character limit in place. Larger QR codes can be used if higher precision is needed.

With the builder in place, algorithm 9 is used in 7 to check what Q would be based on the current iteration, and whether the result is valid or not. Using the builder, algorithm 9 computes the cumulative path length s based on the given foci. The ellipsoid-defining tuple $(f_1,\ f_2\ s)$ is passed to algorithm 8 to build the corresponding Q. The checking function then passes the resulting Q back to algorithm 7, which uses it to determine how to proceed with processing the original path.

Algorithm 9 Check the length of Q

Require: Path P', indices $f1_idx$, $f2_idx$, character limit K

function `CheckQR`(P', f_1_idx, f_2_idx, K)
$P \setminus x_g$
Clamp $f_1_idx,\ f_2_idx$ to $[0,\ \text{length}(P') - 1]$
if $f_1_idx > f_2_idx$ **then**
    swap $f1_idx,\ f2_idx$ ▷ Maintain proper order
end if
▷ Compute cumulative path length s between f_1 and f_2
$s \leftarrow 0$
for $i \leftarrow f_1_idx$ to $f_2_idx - 1$ **do**
    $p_1 \leftarrow P'[i]$
    $p_2 \leftarrow P'[i+1]$
    $s \leftarrow s + \textsc{Dist}(p_1, p_2)$
end for
▷ Attempt QR payload build
$(Q,\ K_{qr}) \leftarrow \textsc{BuildQRString}(P',\ f_1_idx,\ f_2_idx,\ s,\ K)$
if Q is empty **then**
    return $(K+1,\ [\,])$
else
    return $(K_{qr},\ Q)$
end if

Three general cases come from this encoding process. Depending on the existence and placement of a non-zero s, there may be a full path without an ellipse, a path with an ellipse, or an incomplete path in case the agent never reached the goal. Each case is determined by the decoding phase, which is handled by future agents trying to use Q to rebuild a feasible path for the same problem.

6.2 Motion planning within an ellipse

Once all the data is properly parsed from Q, the agent tries to find a feasible path between x_s and x_Q using the encoded information provided. This section covers the flow of how the data from Q is handled and how agents change their motion planning based on it. No significant alterations are made to the motion planning algorithm (RRT#) itself. The primary modification is drawing samples from the ellipsoid directly, rather than rejection sampling or sampling the entire space. If an ellipsoid is used and an agent plans a path within it, the solution is stitched together with the rest of the path nodes given by Q.

Assuming there is a feasible path encoded within Q, there are two prevalent cases. If all s values in Q are zero, it signals that no ellipsoid is encoded and the coordinates. Otherwise, there is a non-zero s present in Q that defines an ellipsoid. In both cases, the encoded path may also be an incomplete solution. In the case of an incomplete solution, the agent assumes no solution exists.

The advantage of using a simple and common method of encoding data is that unpacking it is generally easy. Algorithm 10 presents the method used to decode Q and unpack the variables it contains. After splitting the values based on the delimiter, the algorithm iteratively unpacks each

tuple $(x,\ y,\ s)$. Each iteration, the algorithm checks for a non-zero s. If one exists, it is known that the tuple it comes from is f_1 and that an ellipsoid D is defined. Additionally, it is known that the following tuple is f_2. If more tuples exist in Q after f_2, they are unpacked in the same manner as before. Otherwise, if Q ends with the f_1 tuple, it is assumed that the goal location x_g was designated as f_2.

Algorithm 10 Unpacking the values in Q

Require: Encoded QR string Q, goal location x_Q

function `DecodeQ`(Q, x_Q)
Initialize empty list P
Initialize D, f_1, f_2, s
$i \leftarrow 0$
while $i <$ length(Q) **do**
    if $i + 2 <$ length(Q) **then** ▷ At least a tuple available
        $(x, y, s) \leftarrow (Q[i], Q[i+1], Q[i+2])$
        if $s \neq 0$ and $D =$ None **then**
            $f_1 \leftarrow (x, y)$, $s \leftarrow s$
            Append (x, y) to P
            if $i + 4 <$ length(Q) **then**
                $f_2 \leftarrow (Q[i+3], Q[i+4])$
                Append f_2 to P
                $i \leftarrow i + 5$
            else
                $f_2 \leftarrow goal$
                $i \leftarrow i + 3$
            end if
            continue
        else
            Append node (x, y) to P ▷ regular tuple with $s = 0$
            $i \leftarrow i + 3$
            continue
        end if
    else if $i + 1 <$ length(Q) **then** ▷ Q ends with only coordinates
        Append $(Q[i], Q[i+1])$ to P
        $i \leftarrow i + 2$
        continue
    else
        break
    end if
end while
Append goal node x_Q to P ▷ $x_Q = x_g$
if $f_1 \neq$ None and $f_2 \neq$ None **then**
    $D \leftarrow$ `Ellipse`(f_1, f_2, s)
end if
return P, D

In the absence of an ellipsoid D, the approach is for the agent to adopt the encoded path and avoid any extra planning. Under the assumptions outlined in section 3.2, the decoded path is

guaranteed to be a successful feasible path as long as the final segment is free of collisions. Alternatively, given an ellipse derived from the Q, the planner breaks the unpacked path into segments and plans within the ellipsoid D. Using the ellipsoid, the agent plans a sub-path between f_1 and f_2 and generates samples within D using algorithm 2. If a feasible path is found that connects the two points, a full feasible solution is stitched together based on where the foci are located in the index order of the nodes from Q. Although D being defined means a solution exists in this case, the planner is still not guaranteed to find a solution, possibly due to a lack of planning time or precision.

Chapter 7: Experiments

We run multiple simulation trials to test the effectiveness of the proposed algorithms and the overall concept. All of the functions, scripts, and testing are done in Python. In this experiment, each trial features the same set of starting parameters and proceeds through all three phases. First, a standard implementation of RRT# is run for a set time limit, searching for a feasible path in each of the generated environments. Following the first phase of RRT#, post-processing begins. The raw path is pruned via line-of-sight, and greedy ellipsoid expansion is applied to it to replace path segments in order to meet the data requirements of Q. Regardless of the result, Q is generated and input to RRT# again, with the caveat that planning occurs only within an ellipsoid, if it exists. The rest of the path is assumed to be the path segments read from Q, with no additional path planning. The following sections in this chapter provide an overview of the simulation experiment setup and process.

7.1 Experiment set-up

All simulated trials were run using Python 3 on an i7-13700KF processor and 64 GB of RAM. All motion planning trials use the same set of randomly generated 2-D environments that contain random polygonal obstacles. The table below displays the parameters used in the experiment trials. Additionally, figure 7.1 provides a visualization of the volume of data run in

this experiment.

Table 7.1: Experimental parameters

Parameter	**Value/Range**	**Details**
Algorithm	RRT#	Motion planning algorithm used
Trials	20	# of trials run in total
Environments	100	# of environments tested in each trial
Run time	5 sec	Max algorithm run time
C	50 by 50	Dimensions of configuration space
x_s	(5, 5)	Starting coordinates
x_g	(45, 45)	Goal coordinates
δ	2, 4, 6, 8, 10	Maximum steering radius
γ_{RRT}	70	RRT# ball constant
$\|Q\|$	25	Character limit for data stored in QR code

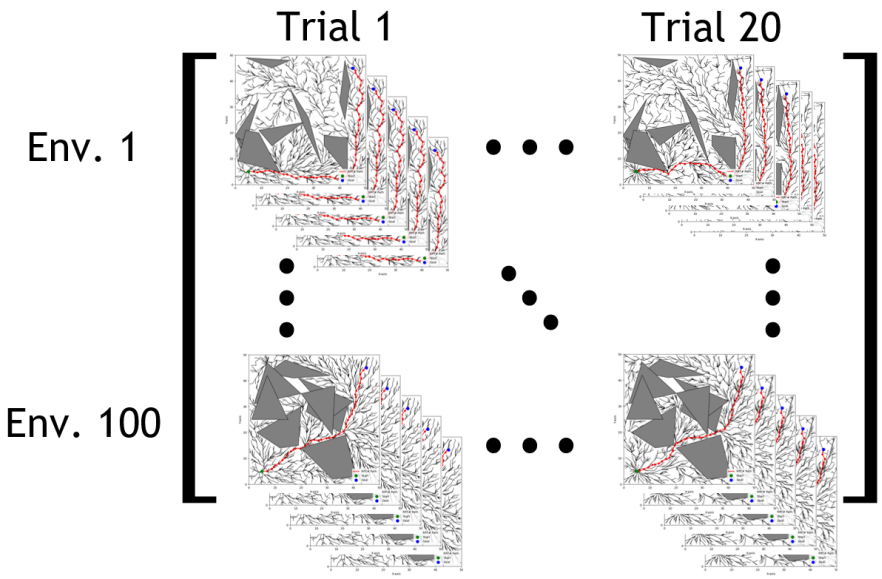


Figure 7.1: Visualization of the total collection of data run in this experiment.

7.2 Parameter selection

The trials are conducted using a 2-D holonomic translation robot modeled as a point, with five different maximum steering radii. Distances are computed with a 2-D Euclidean distance function. There are 20 overall simulation trials in total, each using the same set of 100 environments. The five chosen steering radii are also tested for every environment. A static start and goal are used for each simulation to ensure non-trivial solutions.

Stopping criteria for path planning algorithms such as RRT# can be implemented by limiting either the number of nodes or the run time, but we chose run time as the stopping criterion, taking into account the asymptotic optimality of RRT#. Moreover, the factor of time is more significant to us than the number of nodes generated during planning. In this experiment, a maximum of 5 seconds is set for all path planning.

In addition to stopping criteria, the constant γ_{RRT} was carefully chosen to balance nearest-neighbor operations. This constant affects the rate at which the neighborhood radius shrinks when doing nearest neighbor computation. Based on the steering radii and the testing environment parameters, a γ_{RRT} of 70 is used.

The maximum steering radius is varied during the experiment so that paths with a varying number of nodes can be tested. Given a smaller steering radius, feasible path solutions contain more nodes than instances with larger steering radii, given the same amount of planning time. Having more nodes in the path means more memory is required to store them, and the ellipsoid is meant to help alleviate this aspect. Therefore, varying the steering radii helps further evaluate ellipsoids as a concept for reducing memory requirements without losing valuable information regarding the environment and past feasible solutions.

Chapter 8: Results and discussion

This chapter reviews the results of the simulation experiments and discusses the data and insights gained from them. The proposed stigmergy-based method for communicating path planning information shows marginal improvement over cases without it. Path planning with prior data on the environment reduces both solution time and overall computation time when trying to find a feasible solution. The following results suggest that spending time upfront to process paths and share them for future agents saves future agents valuable time and resources when attempting to find a feasible solution in the same scenario.

8.1 Computation time comparison

When comparing the time it takes to discover a feasible solution between the two robots, the size of the search area and the sampling method may have a significant impact. Common intuition in motion planning suggests that, all things equal, narrowing down the search area helps provide a feasible solution more quickly, even if the path is not within the optimal homotopy class. This is assuming that a solution exists within the area, which is a prerequisite for the methods tested with the second robot. Although ellipsoids are not generated in every case, they are the tool that is used to test and confirm this intuition.

From the trials, both the feasible solution time and the overall runtime for the feasible

solutions are measured and compared in table 8.1. For the second phase, the solution time of a second robot without any help from the first robot is compared to that of a robot that uses data from robot 1. Additionally, the overall runtime is compared for robot 1 and robot 2. As mentioned in section 7.2, the motion planning was run for 5 seconds. To make comparisons fair, only the feasible solution time is included, and the leftover time spent optimizing the path is subtracted from the overall computation times.

In table 8.1, it is evident that the feasible solution time average improves significantly when prior information is used. The mean of the solution time across all trials for robot 2 with Q shows improvement by about 40% over the robots that do not use Q, with a similar distribution. Additionally, the robot using Q shows a 35% improvement in the average runtime compared to robot 1. Adopting existing feasible paths as a basis and reducing the search area means feasible solutions are found faster on average. Even if a path is pared down and manipulated to fit in a highly constrictive QR code, it maintains enough informational value and integrity to help future robots facing the same problem.

Parameter	**Mean (s)**	**Standard Deviation (s)**
Solution Time (Robot 1)	0.1806	0.3286
Solution Time (R2 w/o Q)	0.1936	0.3527
Solution Time (Robot 2)	0.1150	0.6145
Robot 1 Time	0.1865	0.2819
Robot 2 Time	0.1216	0.5749

Table 8.1: Statistics for feasible solution time and processing time

The improvement with robot 2 using Q is further evident when comparing figures 8.4, 8.5, and 8.6. In these plots, the time axis is logarithmic, and the number of successful solutions remains very similar to the original. However, more successful solutions are found sooner by the

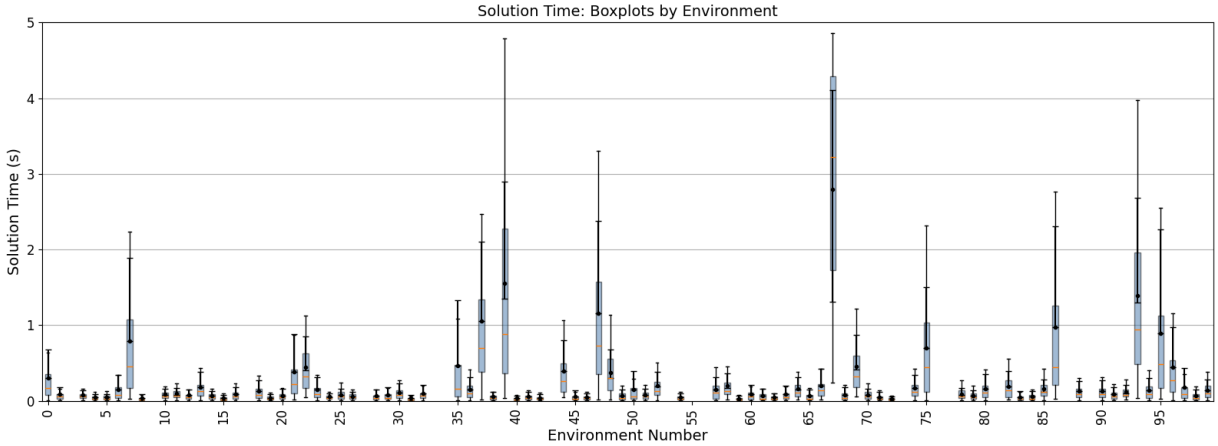


Figure 8.1: Per-environment boxplots of the solution time data for robot 1.

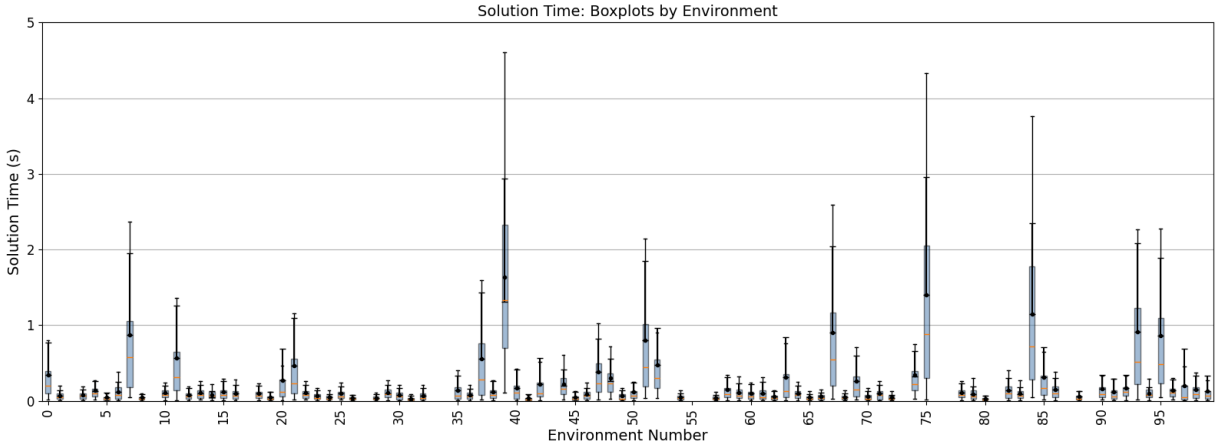


Figure 8.2: Per-environment boxplots of the solution time data for the robot 2 without assistance from robot 1.

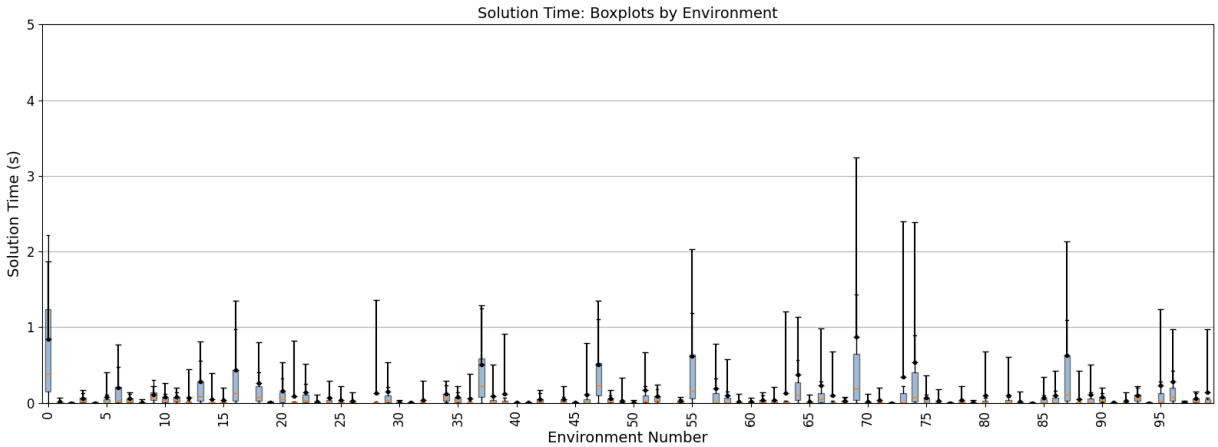


Figure 8.3: Per-environment boxplots of the solution time data for the robot 2 using Q.

robot using Q than by the robot without Q at the same time marker. In this sense, the second robot using Q gets a head start in finding a feasible path.

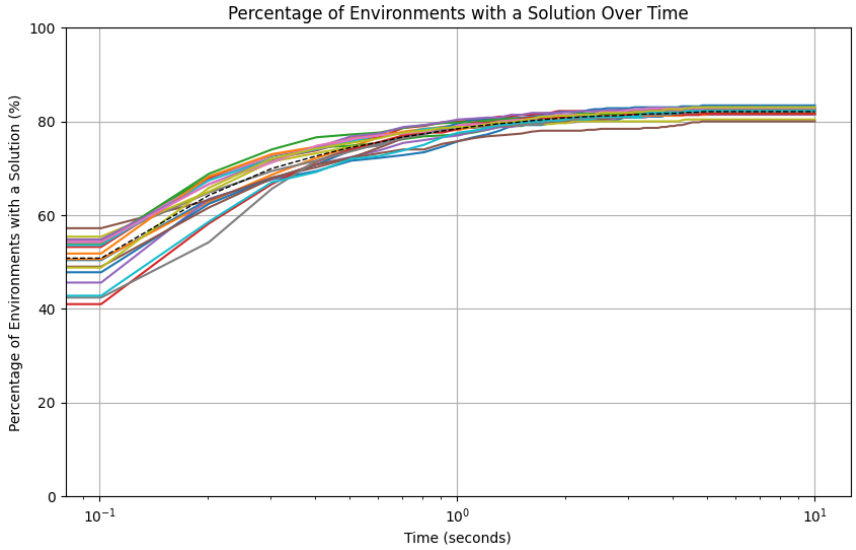


Figure 8.4: Percentage of successful solutions over time (robot 1)

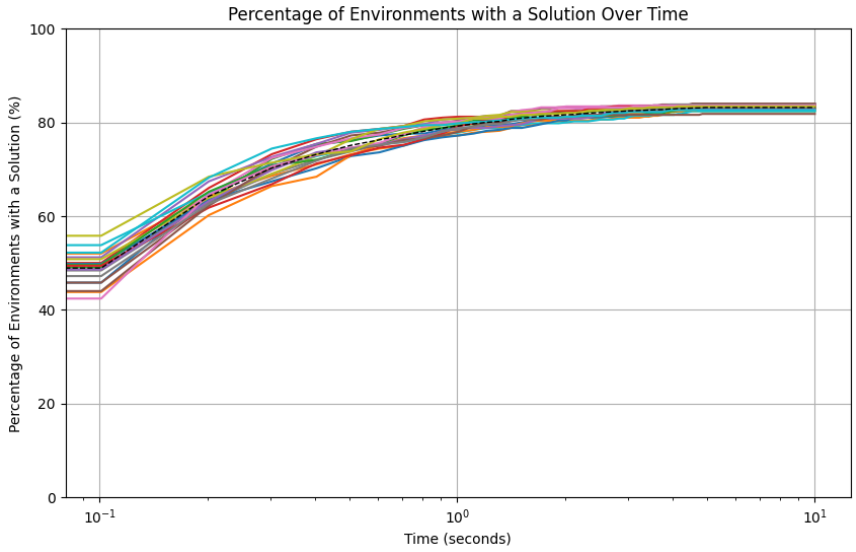


Figure 8.5: Percentage of successful solutions over time (robot 2 without Q)

Despite improvement in the feasible solution times, the overall computation times compared in table 8.1 remain similar. The 'Robot 1 Time' accounts for the feasible solution time and ellipse placement. The 'Robot 2 Time' accounts for the QR decoding, feasible solution time

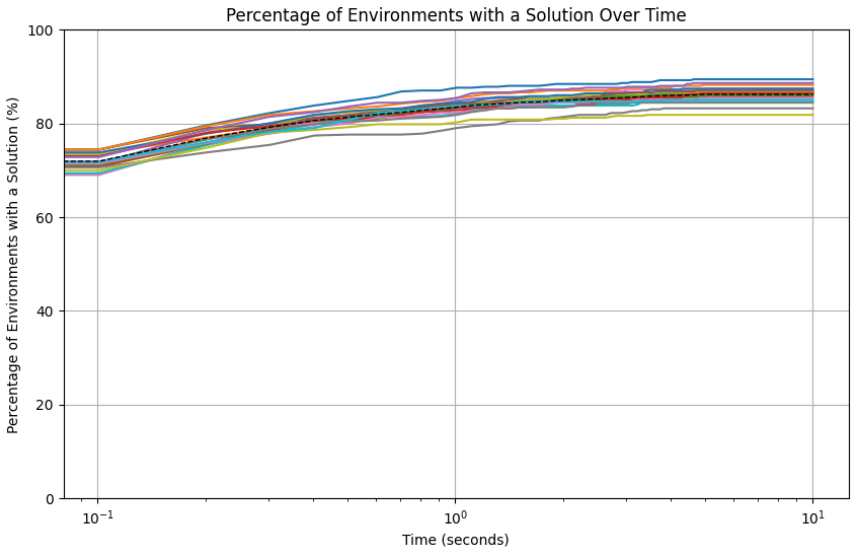


Figure 8.6: Percentage of successful solutions over time (using Q)

(if there is any replanning), and path stitching. Unlike the feasible solution time comparison, the overall computation time from robot 2 remains almost equal to robot 1, albeit with a wider distribution.

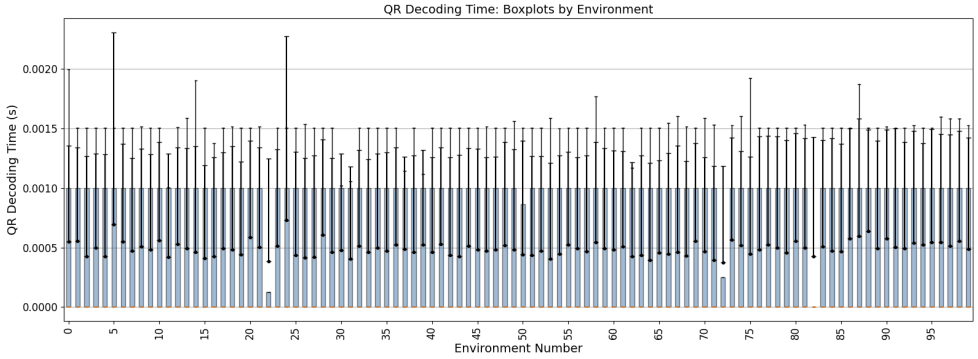


Figure 8.7: Mean and standard deviation of QR decoding time in each environment

Another valuable insight is found in figure 8.7. This plot presents a box plot for the runtime of the QR decoding algorithm per environment. Across each environment, the runtime is nearly constant, with the mean in each environment hovering around half a millisecond. With the error bars, nearly every environment stays below 1.5 ms, with a few environments containing error bar

upper bounds that reach 2 ms and beyond. Regardless of the size of the ellipsoids, paths, and data contained within Q, the decoding time stays roughly constant throughout, only depending on the size of Q.

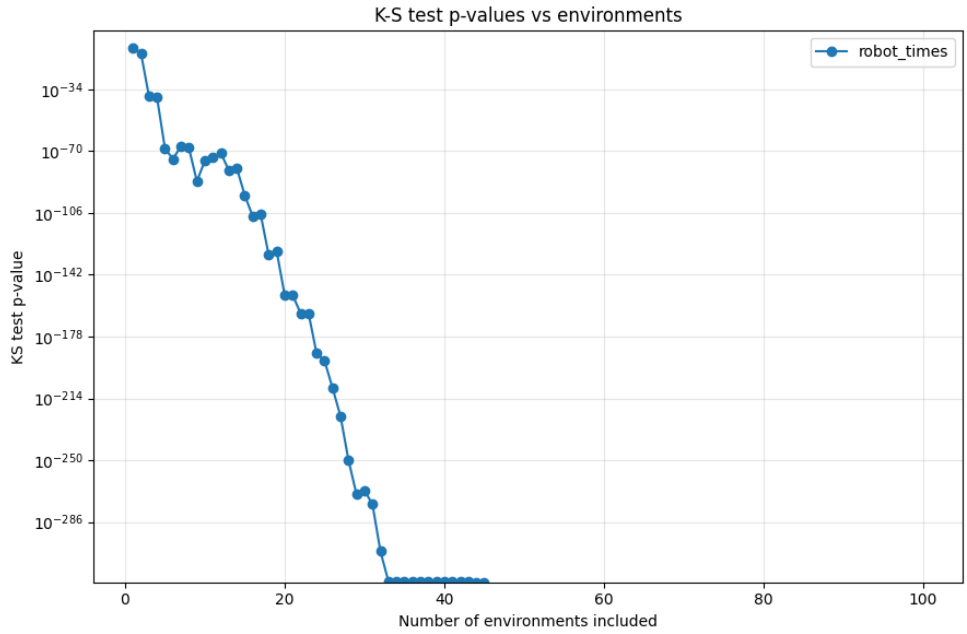


Figure 8.8: Plot showing the p-value from a K-S test as additional environments are added into the data set. Each K-S test uses solution time data between robot 1 and robot 2.

In addition to plotting the computation time statistics, we run a two-sample K-S test comparing the solution time and computation time results from both robots. A generalized gamma distribution is used due to runtimes inherently being non-negative. Figures 8.8, 8.9, and 8.10 show the progressive p-value as each environment data set is added one at a time. Due to the volume of the data set and corresponding numerical errors, the p-values are plotted progressively in this manner.

Each environment corresponds to a set of 100 data points per robot. With the first environment by itself, the p-value begins below the 0.05 mark. As the number of environments is increased, the p-value for both the solution times and the computation times trends downward un-

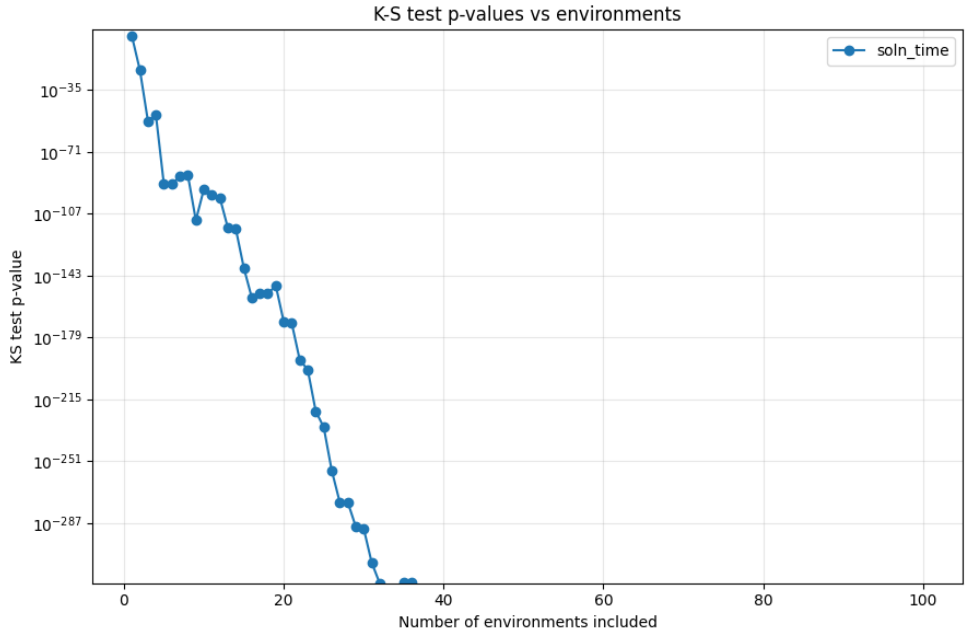


Figure 8.9: Plot showing the p-value from a K-S test as additional environments are added into the data set. The K-S test is performed using the solution time data from trials with and without Q.

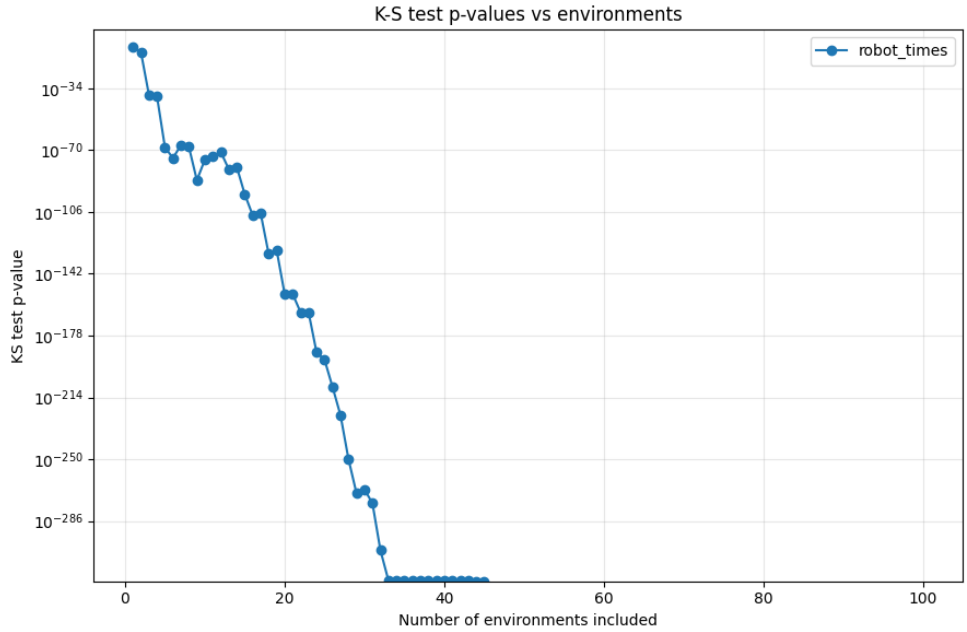


Figure 8.10: Plot showing the p-value from a K-S test as additional environments are added into the data set. The overall computation times from the first robot are compared to the second robot in each K-S test.

til it gets past 40 environments. Beyond the range of 40 to 50 environments, the data set becomes too large, and the p-value collapses to zero. Seeing as the p-value never rises above the 0.05 line,

we can conclude that the data sets from robot 1 and robot 2 come from different distributions.

Overall, receiving and implementing prior knowledge reliably helps reduce the time required to discover a feasible solution to a motion planning problem. However, doing this requires some extra processing and conditional decision-making that can cause future robots to require longer runtimes compared to a robot planning without any prior information, at least in simpler contexts. Based on this data, using geometry like ellipsoids to reduce the search area has been shown to improve feasible solution times, but may lose its advantage in simpler motion planning problems where the curse of dimensionality is less prevalent.

8.2 Path length results

The goal of the second robot in the experiment is to use the data from Q to recreate a path between the same end points as the original robot. The second robot is expected to generate a path similar to or better than that of the first robot. At a high level, the second robot was successful in doing this, even given the short planning time and the rounding error in ellipsoids.

Looking at the overall averages across the full data set for robot 2, table 8.2 shows that the mean path length for the trials using Q improves the mean by 7 percent, albeit with a slightly higher standard deviation. In the cases where the robot plans inside the given ellipsoid, it has the opportunity to dedicate more time toward improving the original path of the first robot, instead of starting from scratch. So even if it is minimal, the second robot is expected to improve the original path on average.

The per-environment box plots of the path lengths for the first and second robots in figures 8.11, 8.12, and 8.13 also support the conclusion that the second robot was able to improve upon

Parameter	Mean (units)	Standard Deviation (units)
Path Length (Robot 1)	65.9377	9.3811
Path Length (R2 w/o Q)	65.9826	10.0699
Path Length (Robot 2)	61.1405	12.2327

Table 8.2: Statistics for feasible solution path length

the first. The set of box plots both have similar shapes, but the box plots corresponding to the second robot remain more consistently under the 80 units mark. However, some environments have much wider interquartile ranges for the second robot, supporting what is shown for the standard deviation comparison in table 8.2. Taking the rounding errors into account, there are some cases where the second robot fails to find a path in the ellipsoid in time, which also lowers the mean and standard deviation in some cases.

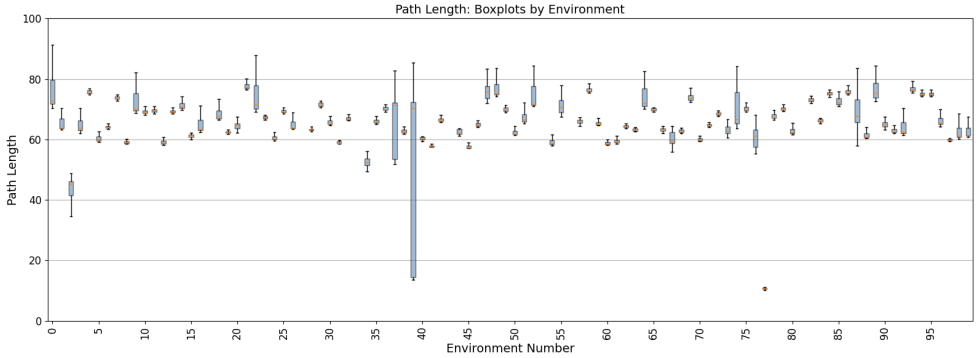


Figure 8.11: Path length mean and standard deviation for each environment (robot 1)

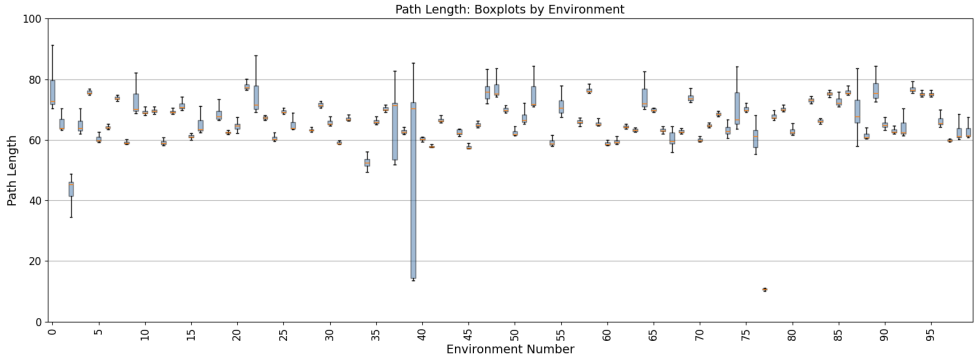


Figure 8.12: Path length mean and standard deviation for each environment (robot 2 without Q)

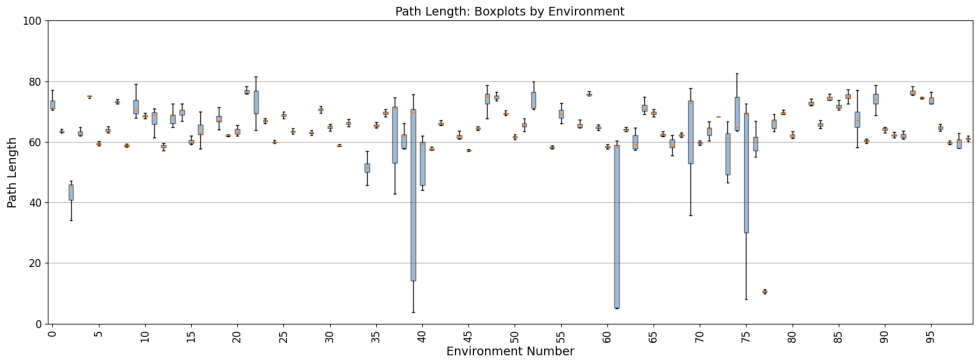


Figure 8.13: Path length mean and standard deviation for each environment (robot 2 with Q)

Each robot in this experiment was given only five seconds to search the environment and plan a path. Even in this short time, the second robot was able to improve the paths found by the first robot. It is reasonable to conclude that, given more time and a more precise Q, the second robot can consistently replicate the original path and make it shorter, even for more complex trajectories.

8.3 Ellipsoid results

The ellipsoids generated during the trials achieve the qualitative goal of reducing the search area to improve the feasible solution time when data is restricted. Figure 8.14 shows that the majority of ellipsoids have an area that is less than 50% of the original environment, and an overwhelming majority are less than 75%. With that said, there are a few outliers that are larger than the original environment, due to the path being longer. This shows that even while sizing the ellipsoid to fit a path, it reduces the search area significantly, a majority of the time.

If we expand this histogram by adding an axis for the feasible solution time, we see an even faster drop off as the solution time increases in figure 8.15. Most ellipsoids across all sizes see a solution time of less than a second. Many of the smaller ellipsoids have trials that go beyond the

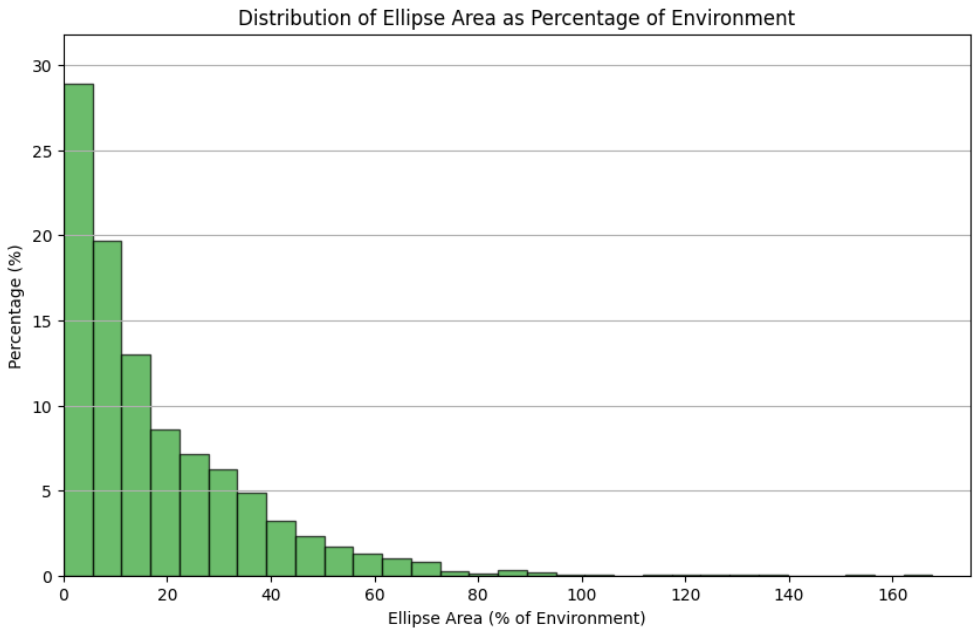


Figure 8.14: Histogram of ellipse areas as a percentage of the environment bounds

two-second mark and reach up to the 5-second cap. This doesn't happen for the ellipsoids under 100%, and the reason can be attributed to both the rounding error discussed in 5.1.1 and the small sample size. As mentioned with the figure 8.14, very few trials generated an ellipsoid larger than the environment's area. Across all environments and trials, there are only nine outlier ellipsoids that are larger than the original environment.

To view the ellipsoid areas from the perspective of each environment, figure 8.16 presents boxplots for each environment, where applicable. We can draw insights from this plot, similar to those from the histograms. First, most environments have interquartile ranges that sit entirely below 1000 $units^2$, and even more have at least a median value below this mark. Recalling that the environment's area is nominally 2500 $units^2$, only five environments have a maximum value that comes close to this value, with the rest being comfortably below the 2000 $units^2$ mark.

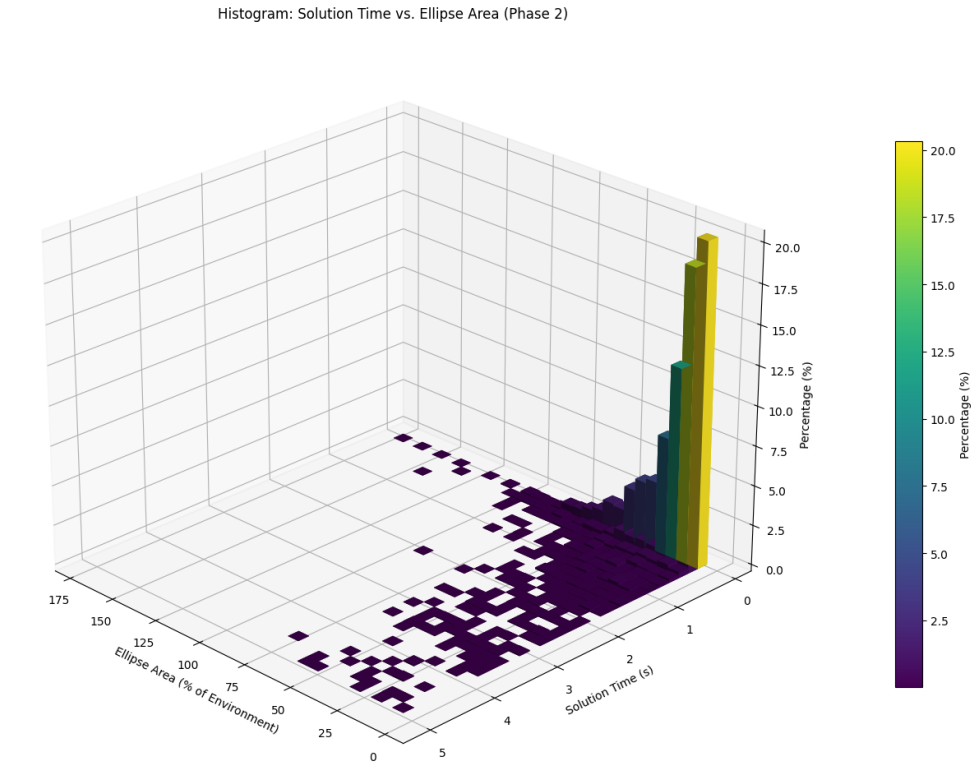


Figure 8.15: 3D Histogram of ellipsoid area and solution time

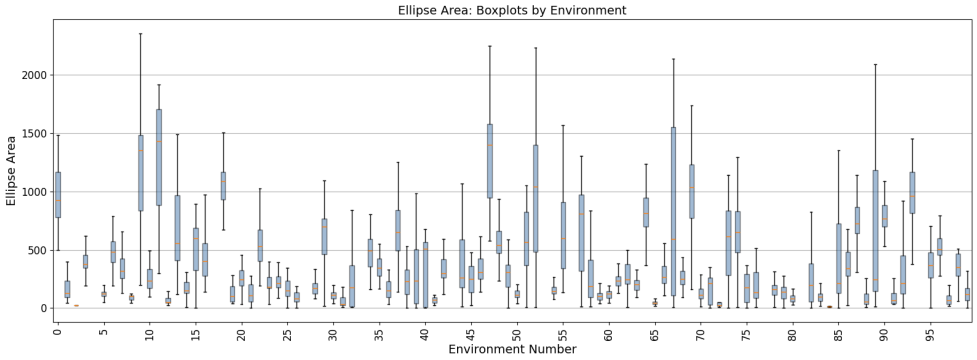


Figure 8.16: Mean and standard deviation of ellipsoid area in each environment

8.4 Memory usage results

Another key measure to determine the method's effectiveness is how well it saves on memory. Looking at the data regarding the original paths in figure 8.17 and 8.18, the majority of paths

take up less than 1 kB of memory, although this far surpasses the storage limit of the smaller QR code versions. The interquartile range is from 268 to 593 bytes with a median of 373 bytes. The second robot that does not use Q has the same results as robot 1, as shown in figure 8.19. For reference, the median value nearly fits in a version 12 QR code, which can hold 370 bytes when using low error correction.

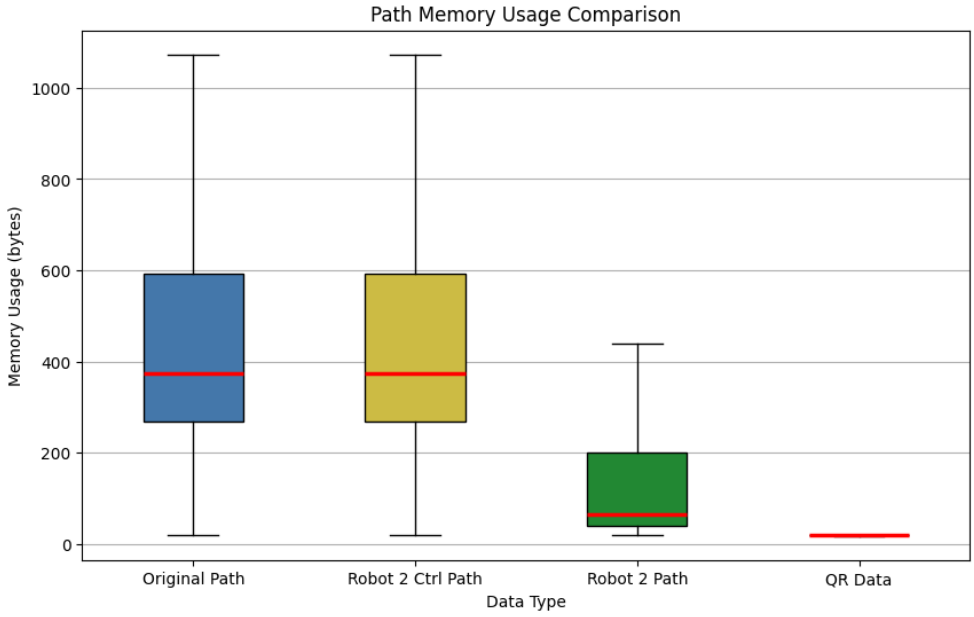


Figure 8.17: Box plot showing memory usage, comparing the robot 1 path to Q and the robot 2 path.

In contrast, the primary distribution of paths from the second robot in 8.20 uses much less space. Its interquartile range is between 40 and 200 bytes and has a median of 66 bytes. The full box plot distribution for the second robot also falls below 500 bytes, which is less than half of that in figure 8.18. Moreover, the 3rd quartile for the second robot is at 200 bytes, compared to 593 bytes for the first robot. Comparing the medians, the memory usage of the paths used by the second robot improves in memory usage by 82 percent. This improvement is due to both the path pruning and numerical rounding that are done by robot 1, as well as the second robot finding shorter paths on average. In terms of a QR code, a median of 66 bytes fits in a version 4 QR code.

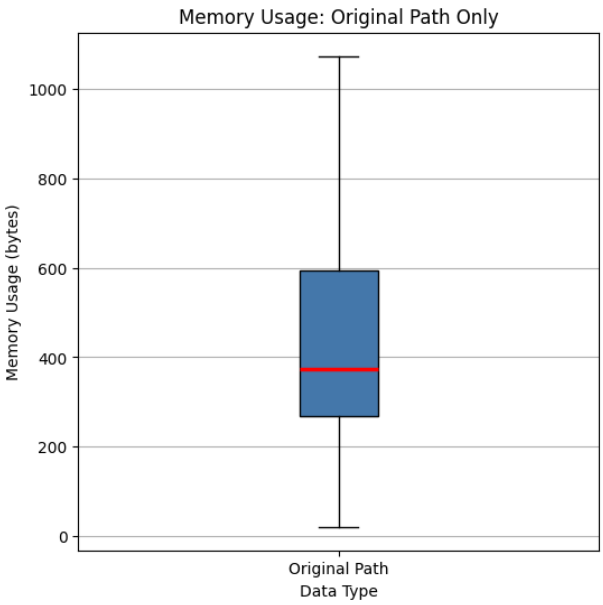


Figure 8.18: Box plot showing memory usage of the robot 1 path data

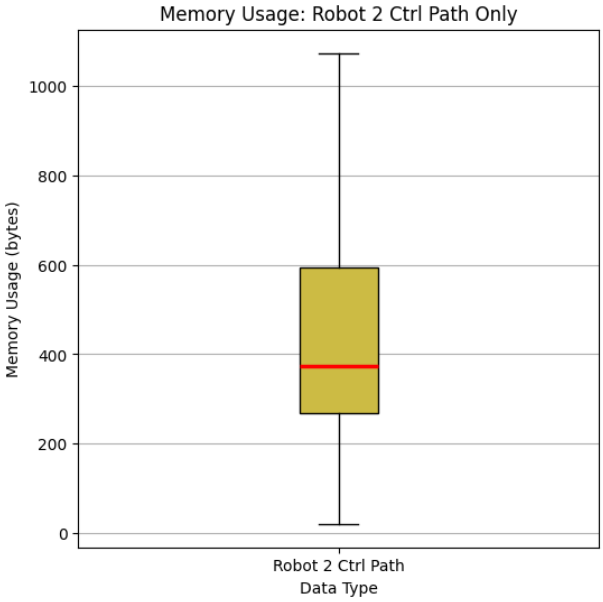


Figure 8.19: Box plot showing memory usage of the robot 2 path data (without Q)

In addition to the paths, the box plot for the encoded data Q is shown in figure 8.21. While all the data satisfies the prescribed character limit of 25 bytes, it is noticeable that some storage remains unused. The median of the data set sits at 20 bytes, leaving a few empty bytes. Additionally, its interquartile range remains between just 19 and 20, and all the data can fit inside a

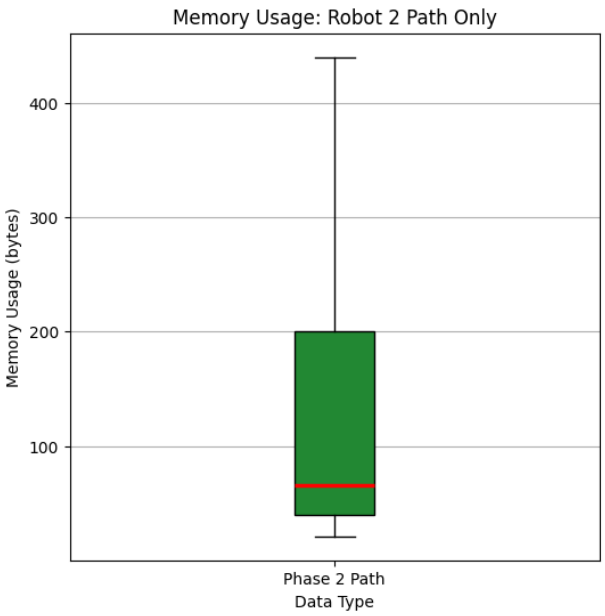


Figure 8.20: Box plot showing memory usage of the robot 2 path data

version 1 QR code, which is the smallest standard QR code size. Part of this phenomenon is due to the chosen encoding method. Given a more sophisticated encoding method, the full data limit could be taken advantage of and accounted for. Further ideas for improvement are discussed in the future work in chapter 9.

Viewing the comparison through the histogram in figure 8.22, there are additional interesting observations. In the paths from robot 1, a few groups emerged. First, the robot 1 bin is large and presumably comprises paths that failed to reach the goal. Next, two large groups remain in the region between 250 and 750 bytes and 1 and 1500 bytes, showing a bimodal distribution. However, this near-bimodal phenomenon disappears with the robot 2 paths. For robot 2, more paths take up less memory. The count then declines steadily as the memory usage increases, unlike the robot 1 data, which presents bimodal distributions.

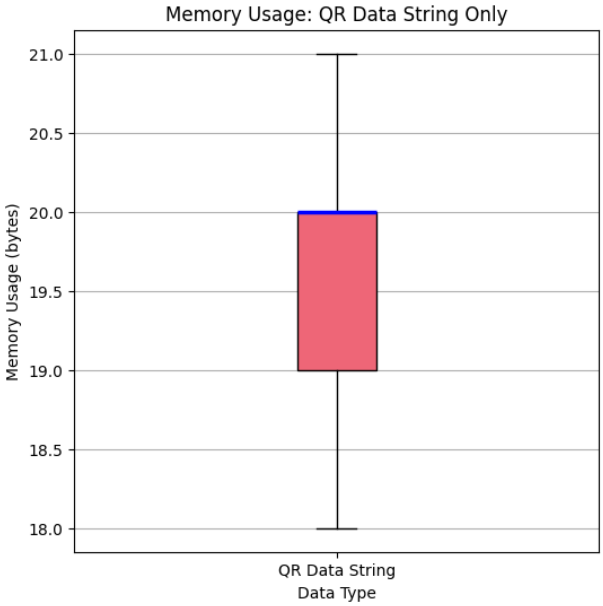


Figure 8.21: Box plot showing memory usage of Q data

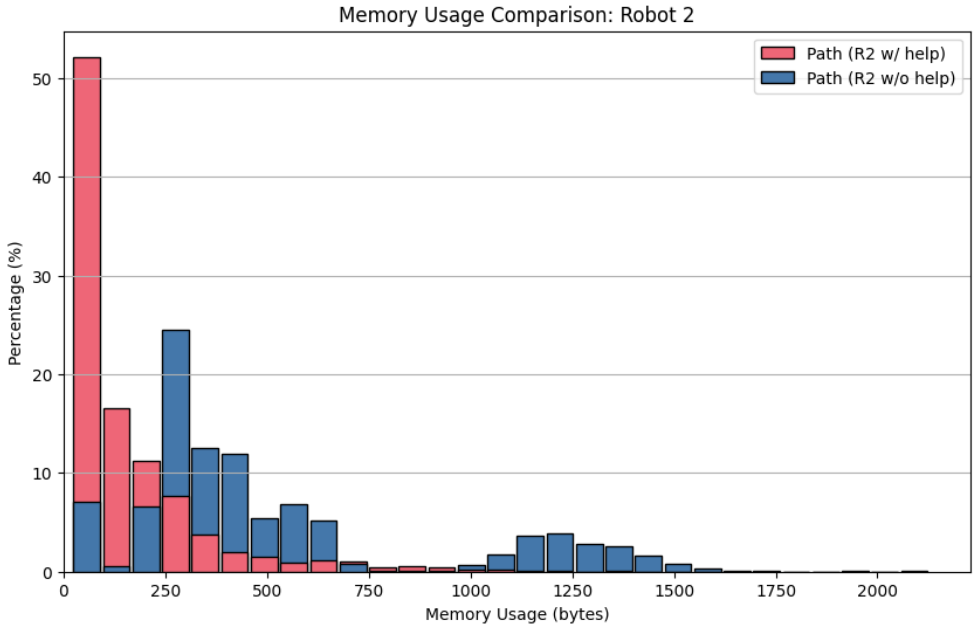


Figure 8.22: Histogram comparing memory usage between the paths from robot 1 and robot 2.

Chapter 9: Future Work

This thesis focuses on developing concepts derived from the intersection of stigmergy and geometry to improve asynchronous motion planning between autonomous systems. Due to the novelty of this niche intersection, the work presented in this thesis is just one of many avenues to approach the topic. We believe that having the ability to pare down autonomous motion planning data and share it asynchronously through the environment is very valuable and extends to mapping, exploration, and teaming applications as well, not just path reconstruction.

In any practical use, a dynamical system will rarely be holonomic or as simple as the dynamics used in this research. More advanced systems may operate within larger C-spaces, some of which may not be Euclidean, which is currently a requirement for the work presented. As such, broadening our research to include more conventional platforms (car-like systems, quadrotors, etc.) is an immediate avenue for future work. The way ellipsoids are used in this research can only be generalized to higher-dimensional Euclidean spaces. However, depending on what C-space a system is working in, the proposed methods would be ineffective if they are non-Euclidean. Being able to identify an approach that includes non-Euclidean spaces opens up the possibility for this concept to work well with other motion planning methods like graph searches, Markov Decision Processes, and more.

In addition to expanding in terms of dynamics, improvements can be made to the pre-

sented methods regarding judgment and optimization. More specifically, the greedy ellipse expansion method differentiates and qualifies nodes based on information about collisions, but there is potential to use other metrics as well. Given additional metrics to qualify nodes, future work includes researching ways to optimize the placement of ellipsoids relative to pre-defined parameters. This manifests in using multiple ellipsoids or using more complex ellipse construction (Steiner ellipse). Moreover, these improvements have the potential to extend to whole search trees and graphs. If more work can be done to develop more sophisticated and compact data encoding much more data can fit into a single physical code.

As discussed in the introduction, this thesis focuses on just one potential application out of many. There are alternative routes and applications beyond the scope of this thesis that can be studied in the future as well. Although ellipsoids are used in this thesis, there is potential to use more advanced computational geometry, such as bounding boxes, meshes, and more. Going a step further, it seems logical to experiment with the use of stigmergy in mapping and coverage applications. Sharing information about what areas have already been explored would allow autonomous agents to avoid redundant work.

Chapter 10: Conclusions

This thesis presented a stigmergy-based approach to multi-agent motion planning, combining sampling-based algorithms with geometric containers to enable asynchronous knowledge sharing without direct communication. By using path pruning and ellipsoidal containers, feasible paths are compactly represented and stored in QR codes that can be physically deployed into the environment. Subsequent agents decode this information and reconstruct partial trajectories, constraining their search space to a given ellipsoid. This results in faster planning and reduced memory requirements.

Simulation results confirm that reusing compact path information from past experiences marginally decreases the feasible solution time while maintaining comparable path quality. Memory usage is also significantly reduced when both path pruning and ellipsoids are used, validating the effectiveness of geometric abstraction as a storage strategy with fiducial markers. The findings demonstrate that even minimal prior knowledge, when structured appropriately, can meaningfully bias motion planning toward promising regions of the planning space.

This work highlights the viability of geometry-based stigmergy as an alternative to conventional wireless communication for autonomous collaboration. Encoding trajectory data directly into the environment creates persistent, lightweight knowledge artifacts that future agents can exploit, promoting scalability and resilience in communication-restricted domains. By reducing

reliance on centralized infrastructure, the presented methods support more flexible deployment of multi-robot teams in unpredictable or difficult settings.

While the experiments focused on simple holonomic dynamics and two-dimensional Euclidean spaces, the approach establishes a foundation for future extensions to higher-dimensional or non-Euclidean configuration spaces, more sophisticated encoding schemes, and broader applications such as mapping, exploration, and coverage. Ultimately, this research contributes to the concept of robust, self-organizing multi-agent systems capable of leveraging shared experiences from environmental markers to accelerate decision-making and reduce redundant computation and overhead.

Bibliography

[1] Marco Dorigo, Eric Bonabeau, and Guy Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871, 2000.

[2] Guy Theraulaz and Eric Bonabeau. A brief history of stigmergy. *Artificial Life*, 5(2):97–116, 1999.

[3] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[4] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[5] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[6] Oktay Arslan and Panagiotis Tsiotras. Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *2013 IEEE International Conference on Robotics and Automation*, pages 2421–2428, 2013.

[7] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., USA, 1999.

[8] Francis Heylighen. Stigmergy as a universal coordination mechanism i: Definition and components. *Cognitive Systems Research*, 38:4–13, 2016. Special Issue of Cognitive Systems Research – Human-Human Stigmergy.

[9] Marco Dorigo, Gianni Di Caro, and Luca M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[10] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014.

[11] Hongyan Sun and M. Farooq. Note on the generation of random points uniformly distributed in hyper-ellipsoids. In *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002. (IEEE Cat.No.02EX5997)*, volume 1, pages 489–496 vol.1, 2002.

[12] Huijuan Zhang, Chengning Zhang, Wei Yang, and Chin-Yin Chen. Localization and navigation using qr code for mobile robot in indoor environment. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2501–2506, 2015.

[13] Pamarthi Ravi Teja and A A Nippun Kumaar. Qr code based path planning for warehouse management robot. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1239–1244, 2018.

[14] Sy-Hung Bach, Phan-Bui Khoi, and Soo-Yeong Yi. Application of qr code for localization and navigation of indoor mobile robot. *IEEE Access*, 11:28384–28390, 2023.

[15] Matevž Ogrinc, Robert Modic, Andraž Simčič, Bojan Blažica, and Drago Torkar. Passive tag-based indoor navigation and localization system using qr codes: A proof of concept. In *2024 7th International Balkan Conference on Communications and Networking (BalkanCom)*, pages 163–167, 2024.

[16] Yumeng Wang, Qiang Zhou, Honglin Li, and Haosong Yue. Technology of mobile robot localization assisted byqrcodes. In *2024 4th International Conference on Industrial Automation, Robotics and Control Engineering (IARCE)*, pages 123–126, 2024.

[17] Xuewei Cao, Yiping Yang, Tao Lu, Lixin Fang, and Jixiang Zhang. A fast pose estimation method based on new qr code for location of indoor mobile robot. In *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 257–262, 2019.

[18] Tae-Won Kang, Yong-Sik Choi, and Jin-Woo Jung. Estimation of relative position of drone using fixed size qr code. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 442–447, 2022.

[19] Yuanfeng Li, Qi Zhang, Haipeng Yao, Ran Gao, Xiangjun Xin, and F. Richard Yu. Stigmergy and hierarchical learning for routing optimization in multi-domain collaborative satellite networks. *IEEE Journal on Selected Areas in Communications*, 42(5):1188–1203, 2024.

[20] Sichuang Ren, Xiye Guo, Xuan Li, Xiaotian Ma, and Yang Li. A stigmergic optimization method for distributed collaboration in satellite swarm. In *2023 IEEE 16th International Conference on Electronic Measurement & Instruments (ICEMI)*, pages 439–445, 2023.

[21] Xu Cheng, Rong Jiang, Hongrui Sang, Gang Li, and Bin He. Trace pheromone-based energy-efficient uav dynamic coverage using deep reinforcement learning. *IEEE Transactions on Cognitive Communications and Networking*, 10(3):1063–1074, 2024.

[22] Vladislav Bolshakov, Alexander Alfimtsev, Sergey Sakulin, and Nikita Bykov. Deep reinforcement ant colony optimization for swarm learning. In *Advances in Neural Computation, Machine Learning, and Cognitive Research V: Selected Papers from the XXIII International Conference on Neuroinformatics, October 18-22, 2021, Moscow, Russia*, pages 9–15. Springer, 2022.

[23] Xing Xu, Rongpeng Li, Zhifeng Zhao, and Honggang Zhang. Stigmergic independent reinforcement learning for multiagent collaboration. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):4285–4299, 2022.

[24] Kun Chen, Rongpeng Li, Zhifeng Zhao, and Honggang Zhang. The implementation of asynchronous advantage actor-critic with stigmergy in network-assisted multi-agent system. In *2020 International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1082–1087, 2020.

[25] Carlo Pinciroli, Adam Lee-Brown, and Giovanni Beltrame. A tuple space for data sharing in robot swarms. In *9th EAI International Conference on Bio-inspired Information and Communications Technologies (BICT 2015)*. European Union Digital Library, May 2016.

[26] Ali Abdul Khaliq and Alessandro Saffiotti. Stigmergy at work: Planning and navigation for a service robot on an rfid floor. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1085–1092, 2015.

[27] Sujith Suresh, P.R. Anand, and D. Sahaya Lenin. A novel method for indoor navigation using qr codes. *International Journal of Applied Engineering Research*, 10(77):2015, 2015.

[28] Ying Zhuang, Yuhao Kang, Lina Huang, and Zhixiang Fang. A geocoding framework for indoor navigation based on the qr code. In *2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)*, pages 1–4, 2018.

[29] Jinjin Yan, Jinwoo (Brian) Lee, Sisi Zlatanova, Abdoulaye A. Diakité, and Hyun Kim. Navigation network derivation for qr code-based indoor pedestrian path planning. *Transactions in GIS*, 26(3):1240–1255, 2022.

[30] Karthik Soma, Vivek Shankar Vardharajan, Heiko Hamann, and Giovanni Beltrame. Congestion and scalability in robot swarms: A study on collective decision making. In *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 199–206, 2023.

[31] Sebastian Schmid, Daniel Schraudner, and Andreas Harth. Performance comparison of simple reflex agents using stigmergy with model-based agents in self-organizing transportation. In *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pages 93–98, 2021.

[32] Dorothea Wagner, Thomas Willhalm, and Christos Zaroliagis. Geometric containers for efficient shortest-path computation. *ACM J. Exp. Algorithmics*, 10:1.3–es, December 2005.

[33] Michael Otte and Nikolaus Correll. C-forest: Parallel shortest path planning with superlinear speedup. *IEEE Transactions on Robotics*, 29(3):798–806, 2013.

[34] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ international conference on intelligent robots and systems*, pages 2997–3004. IEEE, 2014.

[35] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, 2015.

[36] Jonathan D. Gammell, Timothy D. Barfoot, and Siddhartha S. Srinivasa. Informed sampling for asymptotically optimal path planning. *IEEE Transactions on Robotics*, 34(4):966–984, 2018.

[37] Zhuping Wang, Yunsong Li, Hao Zhang, Chun Liu, and Qijun Chen. Sampling-based optimal motion planning with smart exploration and exploitation. *IEEE/ASME Transactions on Mechatronics*, 25(5):2376–2386, 2020.

[38] Khen Elimelech, Lydia E Kavraki, and Moshe Y Vardi. Efficient task planning using abstract skills and dynamic road map matching. In *The International Symposium of Robotics Research*, pages 487–503. Springer, 2022.

[39] Khen Elimelech, Lydia E Kavraki, and Moshe Y Vardi. Extracting generalizable skills from a single plan execution using abstraction-critical state detection. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

[40] Nicholas Mario Wardhana, Henry Johan, and Hock Soon Seah. Subregion graph: A path planning acceleration structure for characters with various motion types in very large environments. *Computational Visual Media*, 1(2):105–118, 2015.

[41] Andong Liu, Shuwen Zhan, Zhehao Jin, and Wen-An Zhang. A variable impedance skill learning algorithm based on kernelized movement primitives. *IEEE Transactions on Industrial Electronics*, 71(1):870–879, 2024.

[42] Aleksandar Vakanski, Iraj Mantegh, Andrew Irish, and Farrokh Janabi-Sharifi. Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1039–1052, 2012.

[43] Weidong Cheng. A study of formation control and trajectory replication algorithms for unmanned systems of industrial robots. In *2024 IEEE 7th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 7, pages 691–695, 2024.

[44] Byungjae Park, Jinwoo Choi, and Wan Kyun Chung. Path reconstruction method for sampling based planners. In *2010 IEEE Workshop on Advanced Robotics and its Social Impacts*, pages 81–86, 2010.

[45] Young-In Choi, Jae-Hoon Cho, and Yong-Tae Kim. Collision avoidance algorithm of mobile robots at grid map intersection point. *International Journal of Fuzzy Logic and Intelligent Systems*, 20(2):96–104, 2020.

[46] Chang'an Yi, Huaqing Min, and Ronghua Luo. Affordance matching from the shared information in multi-robot. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 66–71, 2015.

[47] Abhijeet Ravankar, Ankit A. Ravankar, Yukinori Kobayashi, and Takanori Emaru. Symbiotic navigation in multi-robot systems with remote obstacle knowledge sharing. *Sensors*, 17(7), 2017.

[48] Abhijeet Ravankar, Ankit A Ravankar, Yohei Hoshino, and Yukinori Kobayashi. On sharing spatial data with uncertainty integration amongst multiple robots having different maps. *applied sciences*, 9(13):2753, 2019.

[49] Hiroyuki Kobayashi. A new proposal for self-localization of mobile robot by self-contained 2d barcode landmark. In *2012 Proceedings of SICE Annual Conference (SICE)*, pages 2080–2083, 2012.

[50] Marta Rostkowska and Michał Topolski. On the application of qr codes for robust self-localization of mobile robots in various application scenarios. In *Progress in Automation, Robotics and Measuring Techniques: Volume 2 Robotics*, pages 243–252. Springer, 2015.

[51] V Kroumov and K Okuyama. Localisation and position correction for mobile robot using artificial visual landmarks. *International Journal of Advanced Mechatronic Systems*, 4(2):112–119, 2012.

[52] Sunhong Park and Shuji Hashimoto. Indoor localization for autonomous mobile robot based on passive rfid. In *2008 IEEE International Conference on Robotics and Biomimetics*, pages 1856–1861, 2009.

[53] Rui Rocha, Jorge Dias, and Adriano Carvalho. Cooperative multi-robot systems:: A study of vision-based 3-d mapping using information theory. *Robotics and Autonomous Systems*, 53(3-4):282–311, 2005.

[54] Jinjin Yan, Sisi Zlatanova, and Abdoulaye Diakité. A unified 3d space-based navigation model for seamless navigation in indoor and outdoor. *International Journal of Digital Earth*, 14(8):985–1003, 2021.

[55] Junjie Chen, Shuai Li, Donghai Liu, and Weisheng Lu. Indoor camera pose estimation via style-transfer 3d models. *Computer-Aided Civil and Infrastructure Engineering*, 37(3):335–353, 2022.

[56] Paulo Alves, Hugo Costelha, and Carlos Neves. Localization and navigation of a mobile robot in an office-like environment. In *2013 13th International Conference on Autonomous Robot Systems*, pages 1–6, 2013.

[57] Xun S Zhou and Stergios I Roumeliotis. Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In *2006 IEEE/RSJ international conference on intelligent robots and systems*, pages 1785–1792. IEEE, 2006.

[58] V. A. Ziparo, A. Kleiner, B. Nebel, and D. Nardi. Rfid-based exploration for large robot teams. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4606–4613, 2007.

[59] Árpád Barsi, Ádám Nyerges, Vivien Potó, Szilveszter Siroki, Viktor Tihanyi, and Márton Virt. Offline path planning of automated vehicles for slow speed maneuvering. In *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 000319–000324, 2018.

[60] Zhenwu Zhou, Wenjie Yu, Ping Li, and Hanbing Yan. Research on mobile robot path planning based on a* algorithm with specific node priority. In *2023 5th International Conference on Robotics, Intelligent Control and Artificial Intelligence (RICAI)*, pages 7–11, 2023.