(REST-MODE) Robotic Environment Space Technician - Mobile Operator Designing Ecosystem

Konstantine Batonisashvili

Abstract— **When thinking about path planning, our minds immediately jump to possible strategies for reaching the goal from the start. What would we do if our goal is not static and moves around? More importantly, what would we do if our goal is to create the environment instead of navigating it? Configuration spaces are a common method of transcribing a zone for experimental purposes in a measurable, quantitative way. We see examples of this not just in research, but in sports, kitchens, martial arts, and more. Our research presents a valid solution of laying down a configuration space based on an offline planner which converts digital environments into real-world coordinates. We utilize a novel Depth-First Search and Breadth-First Search path-planning algorithm to achieve 100% coverage without significant backtracking. Our results highlight the feasibility of translating virtual configurations into physical setups, laying the foundation for scalable, user-friendly automation in diverse industries.**

I. INTRODUCTION

Configuration space identification and mapping is a subject that has long been studied in the robotics field. With over half a century of research, the topic could be considered as *solved*, however with each new location, a new set of challenges and concerns need to be addressed. The RASTIC lab at Boston University has a dedicated area for motion planning nicknamed *The Arena* which uses masking tape for work space detection. This masking tape must be removed and applied manually every time a new project is tested. Without a modular environment, this task becomes tedious and prone to human error as manual measurements are required for each setup. Our idea extends this further towards a more generalized audience. We propose a solution for any organization, sport venue, or setup crew, in setting up a workspace field autonomously and with minimal effort.

We utilize state-of-the-art mapping techniques within a modified mobile robot that processes a user defined environment, and maps it in the physical world autonomously. The robot begins by translating a provided configuration space mapping into a traceable environment. Once processing is complete, the robot autonomously traces the environment through a hybrid depth-first search (DFS) and breadth-first search (BFS) algorithm. With included modifications or attachments to the robot base, any type of configuration is re-creatable in the physical world. Thanks to the highly modular nature of this proposition, the base of the robot is interchangeable, attachments are interchangeable, and the configuration space is also interchangeable.

This work was made possible through the contributions of Sophia Compagni (BU MS 2025) in her assistance with attachment designs. Haroon Muhammed (BU MS 2025) and Dwarakesh Rajesh (BU MS 2025) for assistance in procuring a Rosmaster x3.

In this report, we provide historical context on configuration space design and creation, similar works in tangential fields, and currently available solutions on the market for configuration mapping. We discuss our approach to creating the ecosystem which replicates configurations through an overview of our hardware and software components. We break down the details on platform selection, software integration, and evaluate our environment planning strategies. It is important to acknowledge shortcomings and limitations of your proposal, and this is further deliberated at the end of the report.

A. Prior Work

Looking through a historic lens, creating a configuration space for robots to localize themselves has been researched for some 50 years now. The integration of a designated workspace for a robot, the very fundamental ideals, are brought up by papers such as [9] and [15]. These early pioneers set the precedent for all work that is being done in our modern day. Over the years, these techniques were further enhanced through various hardware and software improvements. For example, the incorporation of lidar within robotics advanced localization to new grounds. [7] shines a light on the DARPA 2007 challenge, where lidar was used for lane marker detection and achieving incredible results within the competition. With technological improvements, environmental mapping became less computationally intensive and more feasible to implement.

The combination of lidar and depth cam has been a common source of data input for many years now. With the ability to measure ground-truth distances with lidar [10], [12] and accurate short-range navigation with depth-cameras [11], [14], there is ample information for localization and mapping. Conceptually closest to our work is [13], [17], where both papers focused on incorporating ROS-based SLAM on in-house mobile units. Since these papers only go as far as mapping and confirming the surrounding area, our approach involves offline path planning to ensure full path completion, and uses SLAM as localization.

Our offline path planner can be compared to [18], which proposes a software solution for route planning based on BFS and DFS. The main difference being their binary start and goal variables compared to our multiple goals which at times may be disjointed and disconnected. We see the shortcomings of utilizing just DFS in [4], where a maze with a loop is able to break the algorithms search. Our approach creates alternative solutions which take into consideration the digital world dimensions, along with the physical. Our early variants of DFS resulted in similar problems of entrapment. The inclusion of BFS eased the continuity error and ensured completion. We discuss this further in the path planning section.

While coverage path planning resembles our overall objective of environment mapping, the subtle nuance of *what* is being mapped calls for vastly different implementation. There are a number of works that are similar to ours such as [3], which discusses coverage planning in a 2D environment focused on coverage time optimality. This research focuses again on the free space within the environment as opposed to the bounds that are set by the configuration itself. Similar coverage techniques are used in broader fields as seen in [8] and [2], taking path planning into unstructured environments or using cellular decomposition in ocean studies. Our work takes these ideas and flips them in its entirety, focusing on the creation of the constraints.

B. Currently Available Solutions

As far as we are aware, there is currently no product in development that performs all of the tasks proposed by our system. The goal of our robot is to sweep an area and lay down the required lines for any type of arena, such as a court, field, or rink. While the physical models were not tested, we created designs for the robot to have interchangeable attachments for the user to decide what type of markings are preferred, such as tape, sand, paint, or marker. The product currently on the market that performs similar, but not identical, tasks is the Turf Tank [16]. The Turf Tank is an autonomous robot that can paint the lines on any given sports field. It is marketed as an easy-to-use, efficient robot that requires only paint insertion and to press go. There are limitations on its capabilities, as it can only paint fields or turf, and the paint cannot be interchanged with another type of applicator.

Robots that share a common theme of sweeping an entire area are the iRobot Roomba and the Phaneuf Ice Resurfacer. The Roomba is an autonomous vacuum cleaner that is required to sweep an entire room, but instead of laying something down such as paint, its main goal is to clean the entire surface and pick up any dirt, dust, and grime as it goes [6]. The Phaneuf Ice Resurfacer has a similar goal, however it is used on a hockey rink. The goal is to act as an autonomous zamboni, which cleans the ice periodically leaving a fresh layer of ice for players and skaters [5]. The Resurfacer shares a similar goal to our system in the sense that it is sweeping the area of a sports arena, however it is not focusing on just the lines or laying any type of marking down, its sole purpose is to clean the ice.

C. Our Contributions

1) Hardware: Our hardware for this project is chosen with a specific mindset of being interchangeable based on the required necessity. Platform considerations were made for the use of an AgileX Limo, and the Raspberry Pi Mouse, but our chosen model was a Yahboom Rosmaster X3 because of its readiness with ROS, built-in sensors, and availability. These models and many others are compatible with our systems through adjustments to the ROS environment.

Our attachments come in a wide variety and are adjustable based on the environment. Since our proposition is targeted to a broad range of audience, a simple design which attaches to any robot platform provides scalability and ease-of-use. Due to the involved nature of our proposal, only 3 attachments were designed for basic application purposes. The first is designed for a marker slot, which supports any permanent or dry-erase markers to be pressed down against the floor. The second style is a tape dispenser for setting down a configuration space commonly seen in robot vision and path planning environments. Finally, the third design is a rotary grain or liquid dispenser which can be used where the environment needs to be physically modified. We operate under the limitations of item degradation, where the attachments might not function properly if the items have suffered substantial wear and tear.

2) Software: Software handling becomes difficult when different languages are required to collaborate and transmit data. We have our pipeline separated into environment translation, path planning, and communication. Our offline planner takes in raw image data with the desired configuration map. Image conversion is completed through scikit-image, an open-source image processing library in python. Route planning is simulated on the processed image through a hybrid DFS-BFS algorithm. The digitally generated path is forwarded to a physical coordinate space conversion algorithm and then translated into ROS. Through ROS, a set of commands are given to the motors, and the robot begins to autonomously traverse the path in the physical world. The core concerns within the pipeline would be the breakdown of communication. The main assumption is the input of a valid configuration map. Each component within the pipeline relies on one another, without an acceptable image or layout, the system would not function from the very first step.

II. Detailed Overview

A. Platform

With the ability to have an interchangeable platform, the main focus in our robot base was the management and configuration of the ROS 2 environment. Most modern day robots operate on this architecture because of the available customization in communication and command handling. Our Rosmaster x3 is equipped with an NVIDIA Jetson Nano, which serves as the central processing unit for interfacing with and managing data from all hardware and software components. Currently, the Jetson Nano operates on Ubuntu 18.04, a version that does not support any ROS 2 distributions and is significantly outdated. To address compatibility issues and ensure the robot remains future-proof, we utilize Docker. Docker encapsulates applications and their dependencies into standardized units called containers, ensuring consistency and reliability across various environments.

To facilitate seamless development and deployment, we have created a custom Docker image that includes our ROS 2 workspace along with all necessary dependencies. This tailored image streamlines the setup process and guarantees that all components work harmoniously together. This Docker image is deployed using a Docker Compose script, which mounts all of the robot's USB ports and displays them to the container. This configuration provides the container with direct access to the motor driver and sensors, enabling seamless interaction between the software and hardware components. The Docker Compose setup ensures that all necessary hardware interfaces are available to the ROS 2 environment within the container, promoting efficient and reliable operation of the robot.

Control for the Robot falls at the end of our software pipeline once the physical space is measured and path coordinates are provided. Translating the path into motion is dictated by PID controls to the mecanum wheels on the Rosmaster. The robot is equipped with four mecanum wheels, each governed by a PID (Proportional-Integral-Derivative) controller to ensure precise motion control. Specifically, there are three PID controllers responsible for generating velocity commands along the x, y, and yaw axes. Each controller is individually tuned to achieve the desired performance and behavior.

B. Attachments

The purpose of the robot attachments is to be able to have flexibility in which type of marking is desired to be laid down. In order to appeal to a variety of audiences, there are three different attachments that have the ability to lay down tape, marker, grains, and liquids. All of the add-ons have been designed in SolidWorks and scaled to fit the Yahboom Rosmaster X3. While the design is specific to this model of autonomous robots, the advantage of a simple clasp leaves room for adjustments.

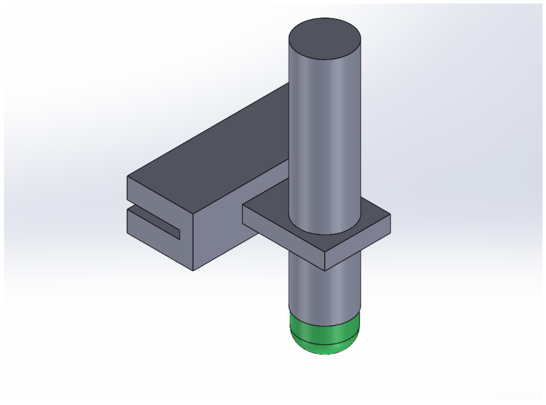


Fig. 1: Marker attachment for the Yahboom Rosmaster X3.

The first attachment is a simple marker holder that would contain a linear actuator to help in the motion of pushing the marker to the ground, keeping it in place when it is drawing, and lift the marker back up in areas where markings are not required, such as a turning point or an area where lines are not necessary. A simple design is shown in Figure 1.

The clasp is made to snap-fit onto the center platform of the Rosmaster, which has a thickness of 2.5 mm, and the marker holder is designed to hold a marker with the diameter of a sharpie, 15.5 mm. A linear actuator would be attached to the top of the marker for movement. The advantages of this attachment is its ease of use, simple replacement, and low cost to make. The wear and tear on the marker tip, along with frequent ink depletion, serve as system limitations which would need to be addressed in the future.

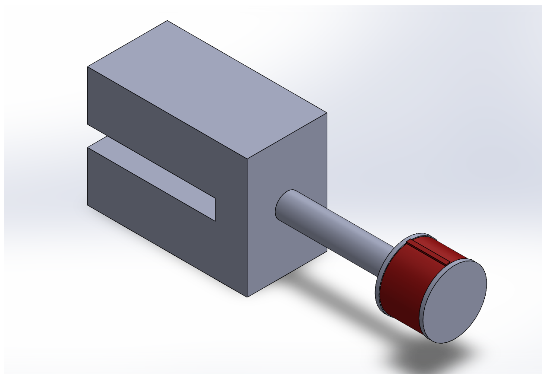


Fig. 2: Tape dispenser for the Yaboom Rosmaster X3.

The second attachment as seen in figure 2 was inspired by the tape that is found in RASTIC, which is used as guidance and boundaries for the limos robots. It has the same clasp snap fit as the first attachment, but instead has a tape dispenser attached.

The tape dispenser would be longer lasting, as the user could attach large rolls of tape that can successfully lay down the required lines without being replaced. The issue of cutting the tape at corners and areas where it is not necessary adds the requirement for some sort of sharp edge, which makes the attachment less user-friendly.

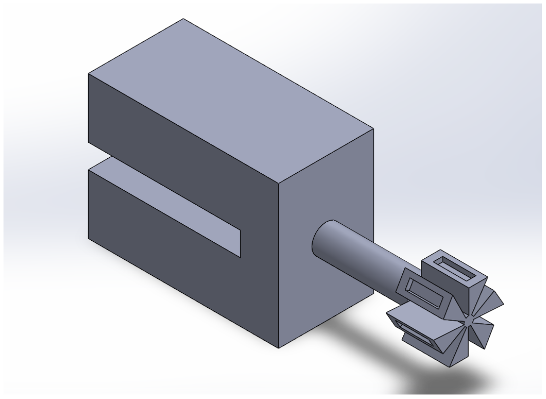


Fig. 3: Grain and liquid dispenser for the Yahboom Rosmaster X3.

The third attachment is promising due to it's simplicity. Our add-on contains a waterwheel in which each section of the wheel has a barrel that can hold the desired type of material. For example, for a sand volleyball court, it can hold sand to continuously lay down lines, or for a turf, it can hold paint.

Some concerns with this attachment is the need to stop it from spinning when moving from spot to spot, and how much material it can actually hold. The size of the barrels would determine the replacement frequency. This device would need an adjustable motor which determines how quickly the material is disbursed from the barrels.

The benefit of having multiple attachments is the ability to use our robot on any given arena or location, depending on user application. Each design ensures a sturdy fit to the main robot, not offering harsh resistance if a replacement is necessary. The downfall of all mechanical systems lie in their respective concerns of failure and wear, overall putting our robots level of autonomy up for debate.

C. *Environmental Translation*

This aspect of the software pipeline focuses on translating the provided environment map into traceable binary paths. A valid pre-configured map must be created for this translation to work properly. There are no size restrictions and any configuration, whether personally created or utilized from other sources, is parse-asble in our ecosystem. An experimental application was developed in Unity3D to act as a one-stop-shop software solution for map uploading and processing. The application utilizes several open-source image processing libraries such as Scikit-image, OpenCV, and SciPy.

Unity3D was chosen specifically due to their robust documentation, python integration, and ease of simulation. This lightweight program can be seen in figure 4, which shows us the steps it goes through to complete environmental translation. First, an image is chosen by providing the appropriate file path and clicking the preview button. Once the image is identified and loaded into the program, clicking submit runs through the skeletonization process and prepares the environment for path planning. The simulate button will then perform the hybrid DFS-BFS algorithm and produce a valid path that is exported as an X-Y coordinate tuple.

Image processing is the key step which allows our software to understand what environment we are working with and all possible segments which need to be traced. Within Scikit-image, we heavily rely on their skeletonize package, which reduces an image down to binary ones and zeros. When choosing an image or an environment to work with, we must ensure that it follows appropriate guidelines to be compatible.

1) The image must to be in a PNG, JPG, JPEG format.
2) The image must have a white background, and a configuration space denoted in black line segments (You may have noticed how during processing, the colors actually get inverted).
3) The image must have feasible pathways that can be recreated in the physical world.

The skeletonization process is complete once the entire image is reduced down to single-line binary code. The middle image in figure 4 shows the skeleton of a simple box with a diagonal inside of it. The key takeaway is the reduction in depth size of the line segments. In a way, the skeletonization process operates as a control for route accuracy. The depth

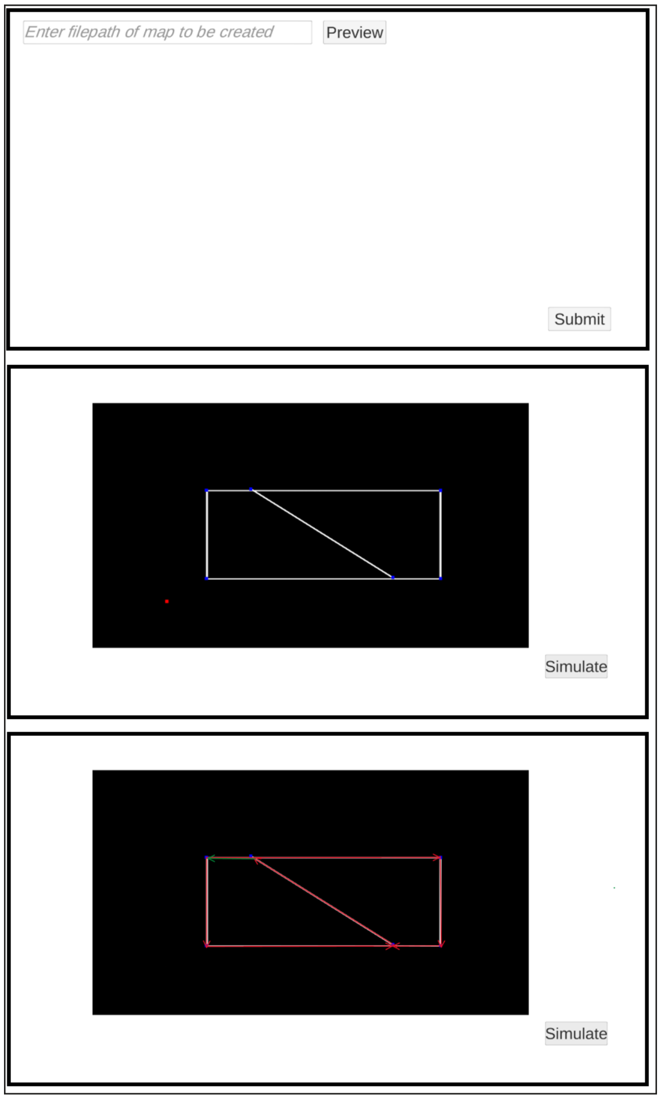


Fig. 4: Unity3D Software which uploads the image, submits environmental translation, and simulates the path planning.

reduction ensures an upper and lower boundary of the original configuration is not breached.

This software is designed to accommodate robots of various sizes and maps with different layouts. The convenience of this tool is seen in its simplicity, where it eliminates the need for prior robotics expertise, allowing a layperson to execute path planning tasks effectively. However, with such adaptability and scalability come complexities. The current framework does not work well with dynamic environments and this may lead to erroneous image processing due to unaccounted parameters.

An additional input must be given after path planning is complete. For proper environmental conversion, the user must provide the dimensions of the space they are working with. Real-world scaling is performed through a simple real-to-image scaling factor.

Equation 1 shows how the path gets converted into real coordinates, where real x and real y are the arena measurements mentioned previously. Image X and image y are determined by the inherent pixel count of the generated configuration space. This scaling is then applied to the path that was generated by our path planning algorithm and finally as a route to follow for our Rosmaster. While this leaves room for user error where they might create incorrect configurations, it is rooted in prioritizing scalability, as this design would support any uploaded image.

$$S_x = \frac{Real\ x}{Image\ x} \qquad S_y = \frac{Real\ y}{Image\ y} \tag{1}$$

D. Path Planning

Since our task does not have an *end goal* in the typical understanding of path planning, the solution that follows is also atypical. If we are to picture the entire configuration space as a goal, we may be left with unvisited locations and local minima traps such as dead ends. The solution in its entirety takes inspiration from several different methodologies. Our unique approach implements a depth-first search algorithm which has a breadth-first search extension when no available neighbors are reported for DFS. This resolves the issue of unvisited locations and ensures every single point in the configuration space is traced.

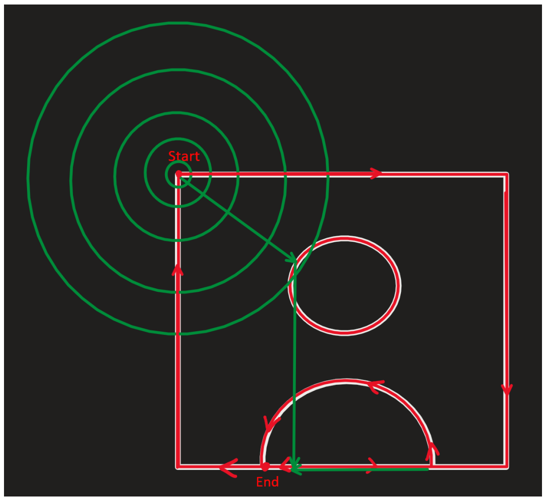


Fig. 5: Example of a potential environment traced by the hybrid DFS-BFS algorithm. The red arrows show the pathways of the DFS algorithm, followed by the green arrows which indicate jumps made by BFS.

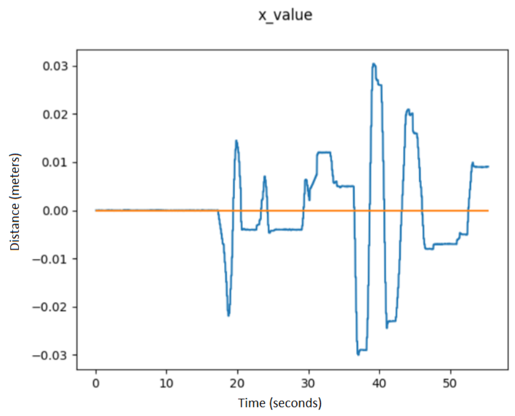
Figure 5 presents an example of how one might visualize our hybrid DFS-BFS algorithm. It starts at the top-left corner of the skeletonized image and begins looking for direct neighbors in an 8-connected neighborhood around the initial point. The neighbors are searched from top-left to bottom-right row-wise. If one is found, that is the destination for our path to follow as can be seen by the red arrows. If there are no direct neighbors and the DFS part reaches an end, the BFS algorithm takes over and begins searching for unvisited points on the skeletonized image based on Euclidean distance. Once BFS discovers the nearest unvisited point, a jump is made

Algorithm 1 Path Planning with Hybrid DFS-BFS Algorithm

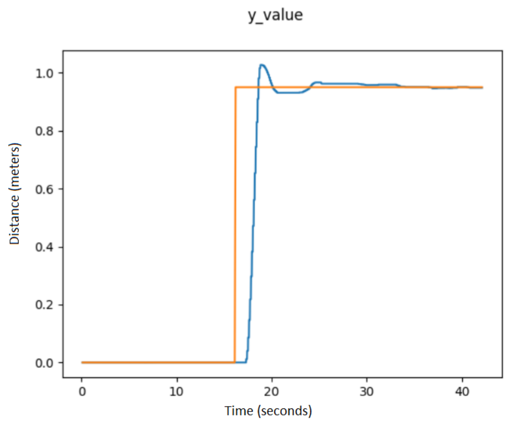
```
Require: Skeleton map $skeleton\_map$
Ensure: Unvisited points $unvisited\_points = \emptyset$
 1: Initialize all $skeleton\_points$ ▷ Find all non-zero pixels
 2: $current\_point \leftarrow skeleton\_points[0]$
 3: $path \leftarrow [current\_point]$
 4: while $skeleton\_points \neq \emptyset$ do
 5:     $visited \leftarrow visited \cup \{current\_point\}$
 6:     $skeleton\_map[current\_point] \leftarrow \emptyset$   ▷ Mark traced
 7:     $neighbors \leftarrow$ Find neighbors of $current\_point$
 8:     if $neighbors \neq \emptyset$ then
 9:         $next\_point \leftarrow neighbors[0]$   ▷ (DFS)
10:     else
11:         $unvisited\_points \leftarrow$ Find remaining non-zero pixels
12:         if $unvisited = \emptyset$ then
13:             break   ▷ All segments are traced
14:         end if
15:         $bfs\_distance \leftarrow$ All distances from $current\_point$ to all $unvisited\_points$
16:         $nearest\_idx \leftarrow$ Index of closest $unvisited\_points$
17:         $next\_point \leftarrow unvisited\_points[nearest\_idx]$ ▷ Jump to nearest unvisited point (BFS)
18:     end if
19:     Traverse from $current\_point$ to $next\_point$
20:     $current\_point \leftarrow next\_point$
21:     Append $current\_point$ to $path$
22: end while
23: Return $path$
```

towards it and then the standard DFS algorithm kicks in once more. Figure 5 shows these jumps marked with green arrows from the final DFS point to the nearest unvisited point.

The complete logic breakdown can be further seen in algorithm 1. This specific strategy was chosen as a way to prevent DFS backtracking and ensure that every single unvisited point on the configuration map has been traced. Optimizations are indeed possible and are entirely dependent on preference and environment. The order of neighbor selection can be modified which would result in drastically different path planning. Certain sports fields would benefit from a clockwise search for example, whereas some personalized configuration spaces would call for a unique pattern in neighbor selection. The modularity of our algorithm is by design, and it can be further accessed in the cloud storage below. We provided our code for public access, with examples of various sports fields with attached simulations, and a fun example of an entire maze being traced [1]. It is noteworthy to mention that the hybrid algorithm in the cloud demonstrations have the skeletonization component attached, therefore the only requirement is an available environment for tracing.



(a) Graph showing x_value change over time where orange is the desired x coordinate, and blue is the actual x coordinate.



(b) Graph showing y_value change over time where orange is the desired y coordinate, and blue is the actual y coordinate.

Fig. 6: Partial successs in operating Rosmasters individual wheels to achieve basic path planning.

III. Results

While path planning was not fully implemented on our physical robot, digital simulations yielded positive results. Our Rosmaster implementation was held back because of fine motor control and time constraints, however basic movement and path tracing was captured. Figures 6a and 6b show the attempt at path following where we see changes in actual x and y coordinates over time compared to their desired location. An unexpected issue presented itself in the form of oscillations that we were not able to resolve.

Digital results showed incredible promise in effective implementation. Four example configurations were used to test path planning. Three environments were representations of sports fields in basketball, soccer, and hockey. The last environment was a custom created maze and was tested to ensure every possible segment is traced regardless of environment complexity.

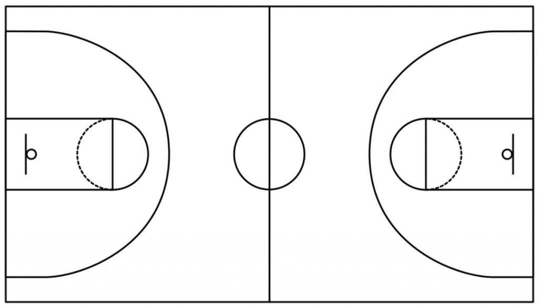


Fig. 7: Basketball field configuration space used as template for tracing.

Figure 7 was used as an example template for environment translation. Having a white background with black lines, valid traceable segments, and in jpg format, the image satisfied the requirements for upload. Figure 8 shows the path in accordance with our DFS-BFS algorithm. The red tracing DFS algorithm falls within our expectations and as soon as no more neighbors are available, the green BFS algorithm shows us the next jump point. Route completion was based on coverage of all segments and these results are consistent among all tested environments, achieving 100% completion on even more complex templates like the maze.

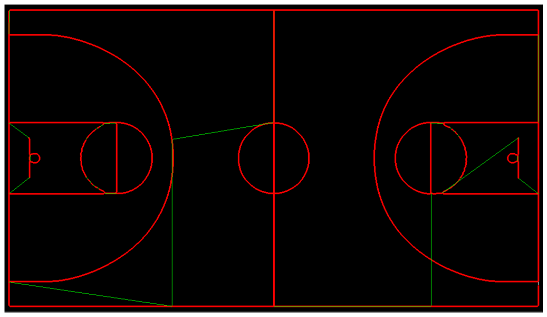


Fig. 8: Completed path planning on Basketball template.

IV. System Limitations

Our proposition has certain restrictions and limitations that must be taken into consideration to ensure successful deployment. With some mentions previously, there are assumptions that we have made for the physical and digital pipelines. Sim-to-real challenges are faced all across academia in research. Transferring your simulation which operates under perfect conditions into the real world always comes with unexpected hiccups with hardware limitations. The robot base or the platform utilized must either be purchased, or acquired through other ways. The platform that was utilized for this project contained several built-in sensors which assisted with localization, however not all sensors are created equal and must be thoroughly tested during system setup. Because the main objective of our proposition is the setup and creation of the configuration space, our most important assumption is environmental obstruction, or the lack thereof. If the space where the robots are operating is littered with obstacles, or the weather conditions are not favorable, critical failure is more likely to occur.

We must also note setbacks in our software. Our goal was to present a solution for full path completion, however incorrect environment builds would be a major setback. Our ecosystem would not operate properly without a functional environment map, which also accounts for correct dimensions and paths. If route planning is processed properly and the robot begins its traversal, the sensors must provide accurate data for localization. It is vital for the robot to have access to the ground truth, or a way to calculate it to ensure it stays on the proper path.

V. Future Work

Future work may expand on this project through full physical implementation or route optimizations within path planning. Our Rosmaster was unable to stabilize its path planning either from bad sensor data or communication pipeline obstructions. With a solid baseline to build from, sim-to-real feasibility would depend on adjustments to the ROS environment, accurate data capture, and processing.

Optimal path planning for full coverage has several decades of research behind it, but extending this idea to the configuration space boundaries as opposed to the free space within has not. Our proposition captures all possible segments through a neighbor connected DFS-BFS search, and it can be optimized for each unique environment. Optimality can be defined through least backtracking, shortest distance, quickest completion, or any other defined metric. Any of these directions are possible for our algorithm in potential future work.

VI. Conclusion

In this paper, we proposed an in-depth, out-of-the-box, ready-to-go solution to a market which is yet to be looked at. The REST MODE ecosystem consists of several parts, which when joined together, provide the end-user with a robust, modular, and autonomous configuration creating robot. Our digital code and offline planner are included for demonstrations and personal use. This proposal may have some limitations, however with such a large scope it is critical to acknowledge all of them. Special thanks are extended to Sophia Compagni for assistance in attachment design. To Haroon Muhammed and Dwarakesh Rajesh, for providing the Rosmaster x3.

References

[1] K. Batonisashvili. Rest-mode. `https://drive.google.com/drive/folders/1lUfqdj1cR4IC9LQd7mNP7YrzA50-Zzpl?usp=sharing`. Accessed: Dec. 15, 2024.

[2] G. Chen, Y. Shen, X. Duan, J. Wan, T. Yan, and B. He. Coverage path planning based on improved exact cellular decomposition method in ocean survey. In *Global Oceans 2020: Singapore – U.S. Gulf Coast*, pages 1–4, 2020.

[3] S. Gajjar, J. Bhadani, P. Dutta, and N. Rastogi. Complete coverage path planning algorithm for known 2d environment. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 963–967, 2017.

[4] A. S. Hidayatullah, A. N. Jati, and C. Setianingsih. Realization of depth first search algorithm on line maze solver robot. In *2017 International Conference on Control, Electronics, Renewable Energy and Communications (ICCREC)*, pages 247–251, 2017.

[5] P. International. Phaneuf ice resurfacer, 2024.

[6] iRobot. Guide to imprint smart maps, 2024.

[7] S. Kammel and B. Pitzer. Lidar-based lane marker detection and mapping. In *2008 IEEE Intelligent Vehicles Symposium*, pages 1137–1142, 2008.

[8] Y. Liu, X. Lin, and S. Zhu. Combined coverage path planning for autonomous cleaning robots in unstructured environments. In *2008 7th World Congress on Intelligent Control and Automation*, pages 8271–8276, 2008.

[9] Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983.

[10] G. Lu. Slam based on camera-2d lidar fusion. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16818–16825, 2024.

[11] D. Maier, A. Hornung, and M. Bennewitz. Real-time navigation in 3d environments based on depth camera data. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 692–697, 2012.

[12] M. A. Markom, A. H. Adom, E. S. M. M. Tan, S. A. A. Shukor, N. A. Rahim, and A. Y. M. Shakaff. A mapping mobile robot using rp lidar scanner. In *2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pages 87–92, 2015.

[13] P. Phueakthong and J. Varagul. A development of mobile robot based on ros2 for navigation application. In *2021 International Electronics Symposium (IES)*, pages 517–520, 2021.

[14] K. Qian, X. Ma, F. Fang, and H. Yang. 3d environmental mapping of mobile robot using a low-cost depth camera. In *2013 IEEE International Conference on Mechatronics and Automation*, pages 507–512, 2013.

[15] W. E. Red and H.-V. Truong-Cao. The configuration space approach to robot path planning. In *1984 American Control Conference*, pages 288–297, 1984.

[16] T. Tank. Turf tank two, 2024.

[17] Y. K. Tee and Y. C. Han. Lidar-based 2d slam for mobile robot in an indoor environment: A review. In *2021 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*, pages 1–7, 2021.

[18] E. Žunić, A. Djedović, and B. Žunić. Software solution for optimal planning of sales persons work based on depth-first search and breadth-first search algorithms. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1248–1253, 2016.