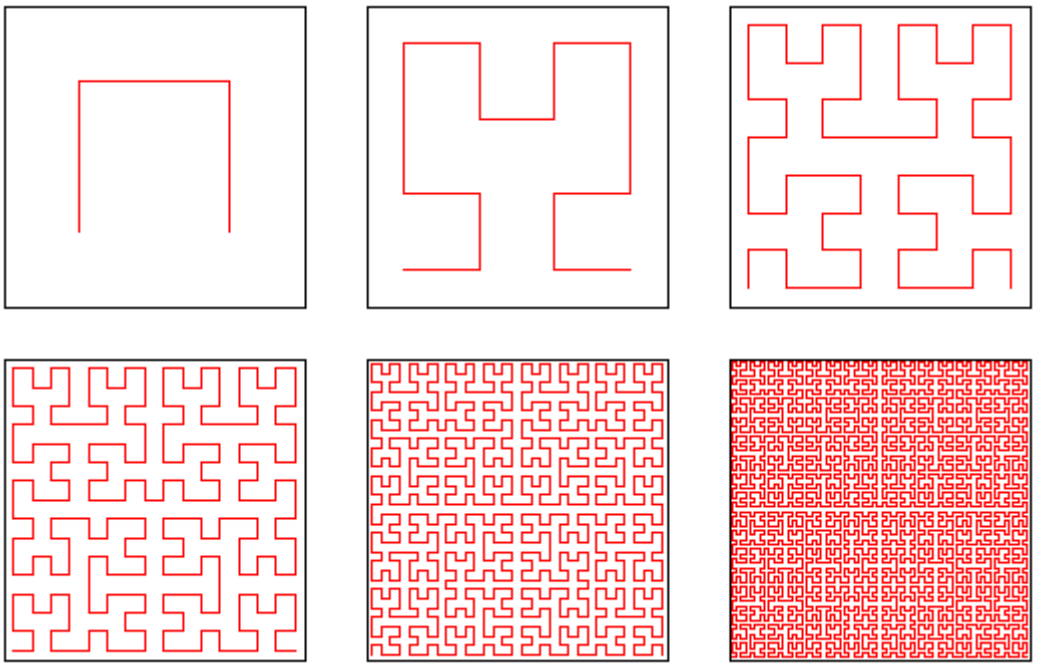
A COMPARATIVE ANALYSIS OF HILBERT CURVE ALGORITHMS IN THE EDUCATIONAL ROBOT CREATE 3

Naeem Sutton
Computer Science Department, Quinsigamond Community College

Ruth De Leon
Computer Science Department, Quinsigamond Community College



Abstract

This present research aims to provide insight to a potentially enhanced method that maximizes the navigational approach of vacuum robots. The primary goal of this project was to determine if using a Hilbert curve algorithm offers optimal room coverage compared to conventional navigational methods commonly used in the market. The application of this work included the implementation of a Hilbert curve algorithm into the Create 3 educational model by iRobot. To achieve this, we utilized a raspberry pi 4 with ROS2 software to control the robot. In addition to this, we implemented a navigational and mapping system utilizing software and tools such as Rviz, nav2 and SLAM in combination with a LiDAR camera. During our research we aimed to test the type of coverage obtained with the developed algorithm in order to identify whether Hilbert curves are optimal in comparison with existing space-filling algorithms. The importance of our research relies in that despite significant technological advancements, contemporary autonomous vacuum robots still exhibit navigation and room coverage flaws in their designs, yielding inconsistent results. Moreover, our project is also focused on advancing in the research about space-filling algorithms as these play important roles in other fields outside of autonomous cleaning robots and it is crucial that further research is done to improve this technology and benefit everyday consumers and organizations.

Table of Contents

CHAPTER 1

Introduction

The primary goal of this project is to assess robot navigation and mapping optimization through the implementation of algorithms, specifically with a focus on a Hilbert space-filling algorithm. To accomplish this, the methodology of this research will be implemented in the *Create 3*® educational robot by *iRobot*, an autonomous vacuum machine engineered to navigate spaces independently. The harward portion of this project includes the use of a *Raspberry Pi 4* chip, as a mediator between the robot and the *Ubuntu* operating system, as well as a LiDAR (Light Detection and Ranging) camera that utilizes lasers and sensors to provide environment information.

The methodology of this project will begin with ensuring complete functionality between the *Create 3* robot, the Raspberry Pi and the rest of components necessary to ensure a smooth testing process. Subsequently, the space-filling algorithm will begin development with the aid of different source code in order to adapt it to the needs of the project. Once the algorithm concludes its scripting phase, the project will move to the testing phase which will consist of two stages: virtual and physical simulations. The focus of this project will be space-filling curves, more specifically, Hilbert curves.

For the timeline of this project, a structured weekly plan has been outlined. Starting the week of January 22nd, weekly meetings are conducted to work in the development of this project with the instructor and peers, which will take place at *WPI* and *QCC*. Towards the finalization of the research, two presentations are

scheduled, one taking place at *WPI* and the second one at *Umass Amherst* for the Umass Undergrad Conference that will take place on April 19th.

While most models in the market already employ advanced mechanisms, plenty of room for innovation remains available. The importance of this project relies on the improvement of existing technologies such as those used in Roombas and similar models. Among the benefits of this project we have that: testing conducted in algorithms such as space-filling curves lay the groundwork for other autonomous systems, not just vacuum robots, specifically in the areas of navigation, coverage and adaptability.

CHAPTER 2

Related Works

In an article by Brian Bennet on CNET, it is noted that "the way a bot navigates around a room will impact not only its cleaning performance but also how long it takes to get the job done" (Bennet, 2021). This conclusion stemmed from research conducted by Brian and his team. They created a test room equipped with a bird's-eye camera and conducted tests on various models of vacuum robots. The methods employed by these devices for mapping and navigation included VSLAM and laser technology, such as lidar cameras.

Brian's research has demonstrated a clear need for improvement among these types of robots, particularly in the methods they employ. Consequently, our project has concentrated on investigating superior implementations for these

devices, aiming to optimize their navigation and mapping capabilities, thereby enhancing their performance as household assistants.

For starters, a space-filling curve (SFC) serves as a method for transforming a multi-dimensional space into a single-dimensional space. Think of it as a path, much like a thread, that travels through every element (like a cell or pixel) in the multi-dimensional space, ensuring that each element is touched precisely once. The concept and creation of a space-filling curve can be traced all the way back to the 20th century; this is the time period where a mathematician named David Hilbert lived. The concept of a SFC was originally used as a way to solve complex problems in mathematical physics. A popular space-filling curve, named the Hilbert curve, was named after him. However, since then, there have been other types of space-filling curves known as Peano curves and Morton curves.

Furthermore, a study on obstacle evasion with space-filling curves [2] recommends using a group of moving robots to explore. Each robot follows a SFC path. This method saves energy, is strong against problems, and guarantees complete coverage in a set amount of time. However, it only works in places without obstacles. Studies on this topic didn't test with obstacles in the environment.

However, in a different paper [3] they come up with a solution to avoiding obstacles using a Hilbert's curve to help a flying robot survey in a closed area. They divide the area into triangles and use the SPC to explore each triangle. Even if there are obstacles around, they've come up with a solution by implementing an algorithm to help the robot avoid it as it goes.

Space-filling curves are versatile tools used for exploration and data gathering, particularly in robotics. With the rise of UAVs, exploration tasks have become more efficient. This paper by Wakode and Sinha, focuses on using the Sierpinski curve for aerial surveying, addressing the need for robust algorithms. The Sierpinski curve offers advantages due to its fractal nature and ability to span both 2D and 3D space. Previous approaches lacked consideration for obstacles or required prior obstacle knowledge. This research presents a strategy for aerial surveying with the Sierpinski curve, including evasive maneuvers for obstacle detection. Finally, this was only accomplished by forming an obstacle avoidance strategy for only one way point being blocked by an obstacle. This leaves the next phase of research at a stagnant place, and future work should aim to introduce multiple objects in an environment and successfully implement object avoidance using a SFC.

Additionally, a study in obstacle-avoidance in Hilbert's curve by Joshi, Bhatt, and Sinha [4] presents a comprehensive exploration of the practical applications of Hilbert's curve(HC), specifically in the robotics, optimization and computer science fields. They dive into the details of utilizing HC for robotic exploration and path planning, they propose a new online strategy that describes the properties of HC to navigate complex environments. The strategy they implemented involved dividing the space into a grid, with each grid cell sized according to resolution of the robot's sensors. By using the HC they devised paths that efficiently navigate around obstacles, ensuring through exploration of the environment while simultaneously

minimizing traversal costs. Furthermore, they created theorems that validate the effectiveness of their approach, these theorems displayed the relationships between blocked nodes and their corresponding positions within the curve. In essence, this research not only contributes to the theoretical understanding of Hilbert's curves but also offers solid background for practical implementation in robotics and path planning.

In another research, Robin Gunning [5] puts to the test different algorithms in a simple robot cleaner that operates using a non-expensive robot cleaner with a single bumper at the front. In his research, Gunning discusses algorithms such as wall following, spiral, random walk and boustrophedon, this one being the main focus of his research. The spiral algorithm gets good coverage in empty rooms, but struggles with obstacles, while the wall following performs well regardless of starting position. Random walk on the other hand avoids getting stuck but may clean already cleaned areas. All combined algorithms perform similarly to random walk but switches between algorithms for faster coverage. Boustrophedon performs best in optimal conditions but struggles with obstacles. At the beginning of his research, Gunning overestimates the Boustrophedon's effectiveness but realizes its limitations. He later concludes that deterministic algorithms like wall following are preferable, specifically in rooms with obstacles. This research provides great insight about how other algorithms perform in autonomous cleaning robots and can serve as a comparison point. While Gunning's investigation highlights deterministic navigation strategies such as spiral and wall following, Hilbert's curve algorithm proposes a novel approach using the spatial efficiency of fractal curves.

CHAPTER 3

Approach and Methodology

3.1 Problem Statement

Throughout the years, the technology around vacuum robots has improved exponentially. However, there still exists room for improvement in the approach that most models utilize, particularly in their navigation systems, mapping capabilities, and general coverage. Low budget models such as the *Eufy Robovac 11S* use a front bumper and cliff sensors to detect when they hit an obstacle. Although more advanced robots, such as the *Roomba j9* by *iRobot,* which implements front-face cameras and modern technologies like *PrecisionVision,* most models in the market still suffer from inconsistent and unpredictable navigation patterns.

Additionally, another problem with current vacuum robot models is that many of them rely on simple maps or basic navigation methods. While some fancy models like the Roomba j9 use advanced mapping tech, lots of cheaper models don't make accurate maps of where they clean. This can make them clean the same spots over and over or miss some areas completely. Also, if things change in the room, like moving furniture, these robots might not clean as well. So, it's important for vacuum robots to have better mapping and navigation abilities to clean well and keep customers happy

3.2 Purpose

The primary objective of this research is to explore innovative strategies for enhancing room coverage in vacuum robots. By investigating these imperfections, this study intends to propose a new approach that addresses the limitations of existing technologies, with the ultimate goal of researching if a Hilbert’s curve algorithm can provide a more consistent navigation pattern. The hypothesis is that the inherent nature of the Hilbert's curve algorithm, designed to efficiently fill spaces, could potentially yield superior and uniform navigation outcomes. The results of our algorithm will be compared with other algorithms such as circular algorithm, zig zag algorithm, and straight line algorithm. These comparisons will show whether or not our algorithm's efficiency is up to par against some of its competitors.

Also, we want to see if using the Hilbert's curve algorithm is practical for vacuum robots. We'll look at things like how hard it is for the robot to compute the algorithm, how well it works in real-time, and if it can handle changes in the environment. And, of course, we'll listen to what people think about using this algorithm in vacuum robots. By testing and analyzing all these aspects, we hope to figure out if using the Hilbert's curve algorithm could really make vacuum robots work better.

3.3 Solutions and Implementation

3.3.1 Preparing the Hardware

Our primary focus was initiating our cleaning assistant, the Create 3 vacuum robot. It serves as a sophisticated tool to streamline cleaning tasks efficiently. At its

core, the robot's operations are managed by a Raspberry Pi 4 with 4 GB of Ram memory, which acts as its brain or central hub for processing instructions and coordinating various tasks.

To enhance its capabilities, we integrated a Rplidar A1M8 camera into the robot's hardware; this camera was directly connected to the Raspberry Pi. This specialized tool possesses a scanning range of 12 meters, enabling the robot to carefully navigate its surroundings with precision. By utilizing this advanced sensor and laser technology, the robot can accurately detect obstacles and plot its course accordingly.

In addition, the Rplidar camera plays a crucial role in ensuring the robot's spatial awareness within its environment. By constantly scanning its surroundings, the camera provides real-time data on the robot's positioning relative to obstacles present in the environment. This information is crucial for the robot to navigate safely and efficiently, avoiding collisions and providing us with a way to optimize its cleaning path.

In essence, the integration of the Raspberry Pi 4 and Rplidar camera equips our cleaning assistant with the necessary intelligence and sensory capabilities to operate autonomously and effectively in various environments. This sophisticated hardware setup lays the foundation for seamless cleaning operations, enhancing efficiency and productivity.

3.3.2 Understanding the Software

To facilitate efficient movement and navigation for our robot, as well as implementation, we leveraged a suite of essential software tools:

- **Ros 2 Humble:** Serves as the foundational framework for our robotic system. It facilitates seamless communication between hardware components and software modules, enabling the integration and coordination of different functionalities within the robot's operating system.

- **Navigation 2:** Provides the robot with crucial spatial awareness, enabling it to understand its surroundings and plan optimal routes. It employs advanced navigation algorithms for autonomous movement, enhancing the robot's ability to navigate safely and efficiently in dynamic environments.

- **SLAM Toolbox:** Enables simultaneous localization and mapping (SLAM), allowing the robot to generate real-time maps of its environment while localizing within the mapped space. In addition, it facilitates accurate localization, essential for navigating unfamiliar ground. Finally, it enhances the robot's ability to adapt to new surroundings and avoid obstacles in real-time.

- **Rviz:** Essential for visualizing the robot's movements and sensor data in a graphical interface. Provides real-time feedback on the robot's position, trajectory, and sensor readings. It also enables debugging and troubleshooting by offering visual insights into the robot's behavior.

The software components mentioned collectively allow our robot to navigate, localize itself and avoid obstacles effectively. By integrating these tools into our project, we enable the robot to operate optimally and efficiently, as well as to adapt to various changing environments.

3.3.3 Teaching with Algorithms

A Python-based algorithm leveraging the Hilbert curve library was developed to generate waypoints for guiding the Create 3 robot during navigation. This algorithm was tailored to the specific characteristics of the Create 3 platform, optimizing coverage efficiency while ensuring smooth and uniform traversal of the environment

```python
def generate_waypoints(self):
    """
    @brief Generates waypoints along a Hilbert curve within the specified cell size.
    """
    # Define the size of the cell (in meters)
    cell_size = 5.5

    # Define the resolution of the Hilbert curve (number of points along one dimension)
    curve_resolution = 10  # Adjust as needed

    # Create a Hilbert curve
    hilbert_curve = HilbertCurve(self.hilbert_curve_order, 2)

    # Generate waypoints along the Hilbert curve within one cell
    for i in range(curve_resolution ** 2):
        # Convert 1D Hilbert index to 2D coordinates within the cell
        x_hilbert, y_hilbert = hilbert_curve.point_from_distance(i)

        # Convert coordinates within the cell to real-world coordinates
        x = (x_hilbert / (curve_resolution - 1)) * cell_size
        y = (y_hilbert / (curve_resolution - 1)) * cell_size

        # Create PoseStamped message for the waypoint
        waypoint = self.create_pose_stamped(x, y, 0.0)
        self.waypoints.append(waypoint)
```

This Python code is designed to assist a robot, such as the Create 3, in generating a path for navigation using a mathematical concept called a Hilbert curve. Here's a breakdown of how the code works:

Firstly, there's a function named ***generate_waypoints*** responsible for managing the waypoint generation process. Within this function, two crucial parameters are defined: cell_size, representing the size of each step the robot takes, and curve_resolution, determining how detailed the path will be.

The code utilizes a library to create a Hilbert curve, a special type of line that efficiently fills up space. This curve serves as the foundation for generating waypoints, ensuring comprehensive coverage of the robot's environment. The library has built in functionality for implementing a hilbert's curve

Next, the code iterates through the Hilbert curve, calculating the 2D coordinates for each waypoint along the path. These coordinates are then converted into real-world positions, enabling the robot to navigate within its physical surroundings. However, we implemented a step variable that limits the amount of points created on the line of our curve allowing our robot to function normally.

Finally, the waypoints are stored in a collection for future reference during navigation tasks, ensuring the robot can follow the planned path effectively. This ensures that the robot never visits a point it did previously and allows it to to complete future waypoints at faster speeds.

For each calculated waypoint, a ***PoseStamped*** message is created. This message contains essential information about the waypoint, such as its position and

orientation, aiding the robot in accurate navigation. These coordinates allow the robot to know its localized position using slam_toolbox functionalities.

```
def create_pose_stamped(self, position_x, position_y, orientation_z):
    """
    @brief Creates a PoseStamped message with the specified position and orientation.

    @param position_x: X-coordinate of the position.
    @param position_y: Y-coordinate of the position.
    @param orientation_z: Z-coordinate of the orientation.
    @return PoseStamped message.
    """
    q_x, q_y, q_z, q_w = tf_transformations.quaternion_from_euler(0.0, 0.0, orientation_z)
    pose = PoseStamped()
    pose.header.frame_id = 'map'
    pose.header.stamp = self.nav.get_clock().now().to_msg()
    pose.pose.position.x = position_x
    pose.pose.position.y = position_y
    pose.pose.position.z = 0.0
    pose.pose.orientation.x = q_x
    pose.pose.orientation.y = q_y
    pose.pose.orientation.z = q_z
    pose.pose.orientation.w = q_w
    return pose
```

This function, create_pose_stamped, is designed to generate a PoseStamped message representing a specific pose with the provided position and orientation. It begins by importing the necessary transformation functions from the tf_transformations module. The position coordinates ***(position_x and position_y***) are directly assigned to the x and y fields of the pose, while the ***orientation_z value*** is used to compute the quaternion representation of the orientation. The orientation is calculated using the quaternion_from_euler function, which converts the provided Euler angles (0.0 for roll and pitch, and orientation_z for yaw) into a quaternion.

The resulting quaternion components ***(q_x, q_y, q_z, and q_w)*** are then assigned to the corresponding fields of the pose's orientation. Additionally, the header of the PoseStamped message is configured with a frame ID of ***'map***' and a

timestamp obtained from the ROS 2 node's clock. Finally, the constructed PoseStamped message is returned, encapsulating the specified pose in the form suitable for ROS 2 message passing.

3.3.4 Putting Everything Together and Testing

The developed algorithm was integrated into the ROS 2 framework, enabling seamless communication and control of the Create 3 robot. Extensive testing was conducted in both simulated and real-world environments to validate the efficacy and robustness of the navigation algorithm.

Furthermore, we brought all our systems together under one roof by crafting a ROS 2 library. Within this library, we designated a specific section tailored for navigation, particularly focused on efficiently clearing a room. Here, we pooled together all the critical settings for Nav2, SLAM Toolbox, our customized scripts, and the various ROS 2 nodes that orchestrate our robot's movements.

This consolidation allowed for a more streamlined approach to development and management. By centralizing our navigation-related components, we simplified the task of integrating different parts of our system. This cohesive setup not only accelerated our workflow but also ensured seamless operation when our robot embarked on room-clearing missions.

Part of our testing included developing simulation environments in Gazebo, a controlled setting for evaluating the algorithm's performance under various

scenarios. To accomplish this, we utilized a special software named *VMWare* to create a virtual machine that could support ROS2 and Gazebo.

Additionally, we conducted extensive testing in real-world environments to assess the algorithm's performance under practical conditions. This involved deploying the Create 3 robot equipped with our integrated algorithm in diverse settings, ranging from spacious living rooms to cluttered office spaces. By observing how the robot navigated through these environments and interacted with obstacles in real-time, we gained valuable insights into the algorithm's adaptability and robustness in dynamic scenarios.

3.3.5 Analyzing the Results

Following the developing of the algorithm, the next results were collected:

Figure 1. Two-dimensional point cloud map of a classroom visualized on RViz.

During our research period, one of the hardest challenges faced was mapping a room, saving the data and subsequently loading the same map into Rviz. The previous image showcases the generated map that was successfully loaded into Rviz. We discovered that one of the main issues was that Rviz needed to be opened before running all the loading commands in order to display the previously recorded map. Additionally, we implemented necessary code in our main script to load the correct map.

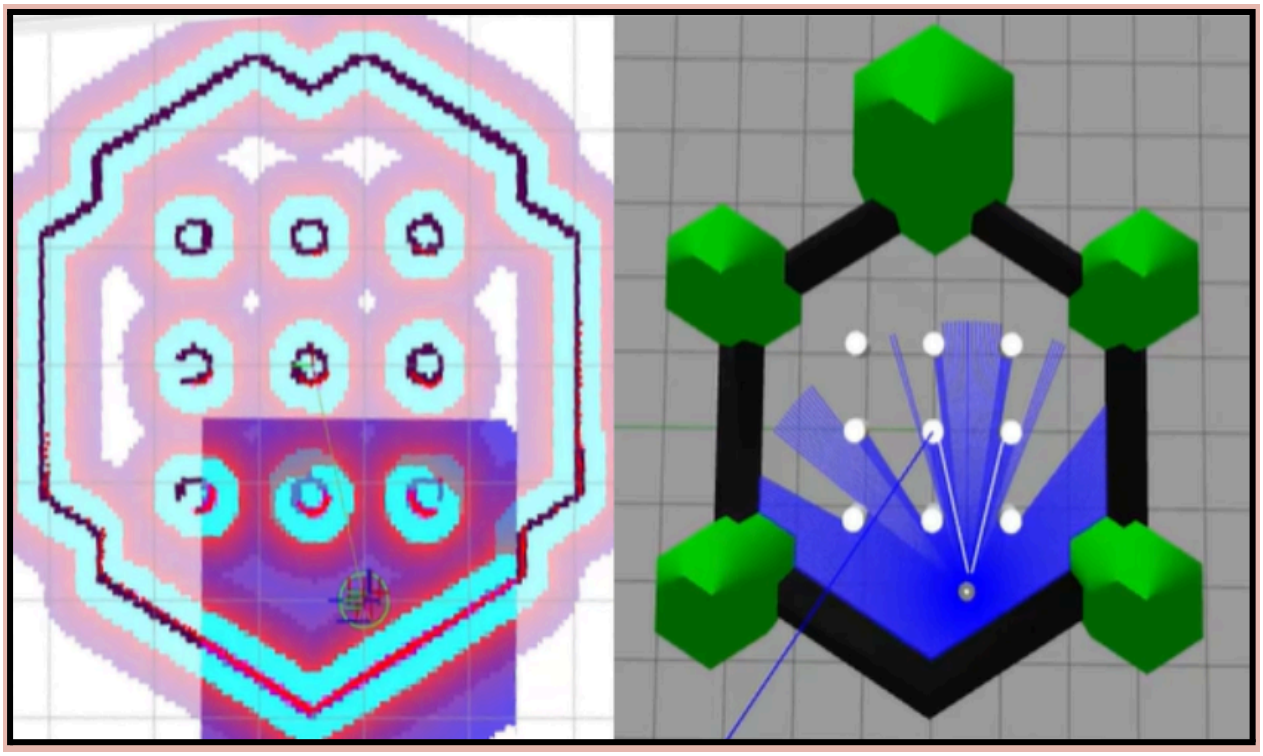


Figure 2. Simulation conducted in Gazebo for experimental evaluation. The left image depicts a visualization of the costmap in RViz, while the right image showcases the simulation environment in Gazebo.

3.3.6 Continuous Improvement

We didn't stop with just one round of testing. We listened to feedback and made improvements to our program to make the robot even better at its job. This involved tweaking algorithms and adjusting parameters to optimize performance. There was a lot of trial and error. This involved troubleshooting coding bugs, implementing Ros2 nodes to communicate with your robot and correcting waypoint plotting using Nav2.

However, despite our best efforts and extensive research, the algorithm implementation did not meet our set expectations. The tests conducted reflected that the curve was working as intended; however, as obstacles emerged, the robot appeared to struggle with updating the path and maintaining the curve continuously. The feedback we received highlighted potential issues stemming from the obstacle avoidance component in our code. Another theory could suggest that the nature of the Hilbert’s curve algorithm conflicts with the software and hardware being used in our research.

While the mathematical simplicity of a Hilbert curve is evident, its recursive nature presents a unique challenge when trying to translate it into robotics. Unlike in mathematical theory, where the curve's construction relies on iterative geometric transformations, a robotic implementation necessitates a more intricate approach, often involving additional computational steps and considerations. Understanding the complexities of the translation process is crucial for developing more efficient robotic systems capable of navigating using Hilbert curves.

In conclusion, developing a Hilbert curve-based robotic navigation system has been quite challenging, but it's also been incredibly fulfilling. Each setback we've encountered has taught us valuable lessons that have pushed us to improve. We're constantly refining our methods and incorporating the latest advancements in technology and research. We're optimistic that with continued effort, Hilbert curves could play a significant role in helping robots navigate complex environments more effectively.

CHAPTER 4

Future Works

In considering the future trajectory for this research, it is crucial to contemplate the technical aspects of implementing the Hilbert’s curve algorithm and the broader implications into the real world. This includes conducting actual experiments to evaluate the algorithm’s efficiency in different environments and scenarios, such as room layouts and obstacle placements.

Furthermore, a more comprehensive approach could involve collaborating with experts in robotics to address present challenges in path planning and refine the algorithm’s performance in relation to vacuum robots. In addition to this, the combination of robotics with other disciplines could potentially open ground for more improvements on the algorithm’s functionality since fields such as artificial intelligence are widely used in various subjects.

Another aspect of our research that could be taken further is the implementation of a 2d planner inside of RviZ that automatically fills up a room with a predetermined pathing. This would improve the algorithm's efficiency by avoiding the concept of plotting hundreds of way points recursively. This improvement would also allow clearing a room to be replicated inside of any environment in very little time.

Overall, a plan for the future should include working on obstacle detection, improving the algorithm performance and the environment, as well as conducting actual testing and evaluations of efficiency such as testing the time and coverage of a room while using the algorithm.

In addition to the previously mentioned suggestions for future exploration, it is imperative to consider the ethical implications and social impact of integrating an advanced algorithm into everyday life and even into other fields of study. As automation and robotics continue to evolve and take over various industries, including household items and tasks like vacuuming, it becomes essential to ensure that these technologies are deployed responsibly.

CHAPTER 5

Conclusions

In the course of the research, we gather insight of different aspects of working with a Hilbert's curve algorithm. As our results showed, one of our major setbacks originated in the obstacle avoidance aspect. We identified limitations regarding space-filling algorithms like the Hilbert curve, particularly in their adaptability to dynamic environments and their ability to navigate multiple rooms or spaces with irregular shapes. In real-world scenarios, where rooms vary in shape and contain numerous obstacles, it becomes imperative for our implementation to navigate such environments seamlessly and maintain a continuous path aligned with the intended Hilbert curve trajectory.

Comparatively, other algorithms such as straight line or circular algorithms are more efficient compared to a hilbert curve algorithm when implementing obstacle detection with robotics. Our algorithm must take in account the order of the curve, dimensionality, resolution, plath planning, and integration with navigation systems. This shows how complex the hilbert curve algorithm is by nature and how much processing must go into plotting waypoints recursively to clear a room.

While our research did not culminate in a fully successful implementation of the algorithm, we encourage further investigation of Hilbert curves in the world of robotics, as this kind of fractal curve provides space-filling properties and continuity in paths that can be beneficial for vacuum robots and other kinds of systems. By addressing the challenges highlighted in our research and further refining the

implementation, we believe that Hilbert curves have the potential to significantly enhance the capabilities of robotic systems in navigating complex environments.

Additionally, Exploring other approaches to addressing the different challenges we faced with the Hilbert Curve algorithm could open up a new avenue for innovation in robotics. For example, putting together machine learning techniques to work hand in hand with a Hilbert Curve algorithm could improve the adaptability of the algorithm in a dynamic environment. Also, making sure to use more advanced hardware with better processing, sensors, motors, and other components could improve the efficiency of the algorithm in navigating complex spaces.

Finally, collaborating with other researchers in fields such as mathematics and robotics could lead to more valuable insight of the Hilbert curve algorithm in robotics. When working on such a complex algorithm, having a collaborative approach for research and development can help overcome challenges more efficiently in a timely manner. In the end, the implementation of hilbert's curve could lead to fascinating discoveries inside of the realm of robotics.

References

1. Bennet, B. (2021, August 27). This is why your Roomba's random patterns actually make perfect sense. *CNET*. https://www.cnet.com/home/kitchen-and-household/this-is-why-your-roombas-random-patterns-actually-make-perfect-sense/
2. Spires, S. V., & Goldsmith, S. T. (1998). Exhaustive Geographic Search with Mobile Robots Along Space-Filling Curves. *Advanced Information Systems Laboratory Sandia National Laboratories.* https://www.osti.gov/servlets/purl/650372
3. Wakode, A. & Sinha A. (2022). Online Evasive Strategy for Aerial Survey using Sierpinski curve. *Indian Institute of Technology Bombay*. https://www.sciencedirect.com/science/article/pii/S240589632300280X
4. Joshi, A. A., & Bhatt, M. C., & Sinha, A. (December 2019). Modification of Hilbert's Space-Filling Curve to Avoid Obstacles: A Robotic Path-Planning Strategy. Retrieved from https://www.researchgate.net/publication/342404990_Modification_of_Hilbert's_Space-Filling_Curve_to_Avoid_Obstacles_A_Robotic_Path-Planning_Strategy
5. Gunning, R. (5 June, 2018). A Performance Comparison of Coverage Algorithms For Simple Robotic Vacuum Cleaners. KTH Royal Institute of Technology. Retrieved from https://www.diva-portal.org/smash/get/diva2:1213970/FULLTEXT02.pdf
6. Jaffer, A. (2003). Hilbert Space-Filling Curves. Retrieved from https://people.csail.mit.edu/jaffer/Geometry/HSFC

7. GeeksforGeeks. (n.d.). Python Hilbert Curve using Turtle. Retrieved from https://www.geeksforgeeks.org/python-hilbert-curve-using-turtle/#
8. Cerveny, J. (n.d.). gilbert2d.py. GitHub. Retrieved from https://github.com/jakubcerveny/gilbert/blob/master/gilbert2d.py
9. Moten, D. (n.d.). hilbert-curve. GitHub. Retrieved from https://github.com/davidmoten/hilbert-curve
10. Sagan, H. (2012). Space-Filling Curves. Springer Science & Business Media. https://books.google.com/books?hl=en&lr=&id=cP_ZBwAAQBAJ&oi=fnd&pg=PA1&ots=sOQoTxiApV&sig=8PXhU4D8f9pu-g9YyUAI30jUIrM#v=onepage&q&f=false