Construction & Robotics

RESEARCH DRIVEN PROJECT VOL. 7

Research Paper / SS 2024

Sigrid Brell-Cokcan
Thomas Adams

AUTHORS

Xiangcheng Li
Yufei Hu
Zeynep Sertkaya
Jiani Zhu Xingying
Li Zonghai Yang
Alperen Balcin
Merve Yanik
Damla Yigit
Nalan Yilmazarslan

construction robotics
www.cr.rwth-aachen.de

international M.Sc.programme

www.cr.rwth-aachen.de
cr@ip.rwth-aachen.de

Editors:
Univ.-Prof. Dr.-techn. Sigrid Brell-Cokcan
Dr.-Ing. Thomas Adams

RWTH Aachen University
Chair of Individualized Production
Campus-Boulevard 30
52074 Aachen
Germany
www.ip.rwth-aachen.de

Layout:
David Lukert B.Sc.

Original cover:
Lukas Kirner M.Sc.

Funded by Production Engineering - Grant of RWTH Aachen University

The content of the book was created for teaching purposes as part of the Research Driven Projects course.

RWTH Aachen University
Templergraben 55
52066 Aachen

First published 2024

Available via the institutional repository of RWTH Aachen University:
DOI: 10.18154/RWTH-2024-09253

ZEYNEP SERTKAYA, JIANI ZHU

Spatiotemporal Multi Agent Path Planning

ABSTRACT

This study investigates the use of spatiotemporal path planning to enhance autonomous navigation in dynamic environments. The spatiotemporal path planning methodology proposes an approach for multiple mobile robot navigation based on predicting agent's near-future trajectory. The predicted trajectories are converted into a time series of cost maps which are used to guide the agent to a desired point in space and avoid where collision in future might occur.

The primary research question focused on whether spatiotemporal path planning could improve navigation efficiency and safety. Initial tests showed that with the algorithms to incorporate the probability distribution of displacement, the system demonstrated a 36.75% reduction in motion time and improved collision avoidance capabilities. These results confirm the effectiveness of spatiotemporal path planning in optimizing autonomous movement in complex scenarios.

The project's findings are consistent with similar research, such as the work by Sato et al. (2023), which also demonstrated improved performance using trajectory prediction and spatiotemporal planning in environments with pedestrians. Future work will focus on incorporating advanced machine learning models, scaling the system for more complex environments, and conducting real-world tests to further validate the approach. Overall, this research shows that spatiotemporal path planning is a viable strategy for enhancing the efficiency and safety of autonomous systems, making it applicable to various fields including robotics, urban transportation, and industrial automation.

Keywords: multi-robot pathplanning; robot learning; dynamic environment; mobile robot

DOI: 10.18154/RWTH-2024-09256

1. Introduction

Future developments of robotics require robots to coexist in a shared environment and operate with each other. Multi-agent systems explore the challenge to control the communication, coordination, and collaboration between multiple robots operating together in complex and dynamic environments. With the increased number of agents, or non-stationary obstacles, such as other robots or moving objects, the dimensionality of the problem and the complexity of computation for obstacle avoidance grows exponentially. Our main objective is to outline a methodology for mapping displacement predictions with time layer based on agent observations. This allows us to operate on a gridworld with the added time dimension, and track dynamic obstacles in possible near-future states [1]. Therefore, in this research, the term spatiotemporal path planning refers to the integration of spatial and temporal data to represent dynamic positions of agents and obstacles to map all near-future states and plan the optimal path for the agent.

We commence with the research questions

- How can spatial and temporal interpolation with Hermite cubic splines
- How are displacement predictions effectively converted into a time series of cost maps to navigate an agent in a dynamic environment?'

We built our research and this paper, on three foundations: (i) literature review and current developments, (ii) creating a research methodology, (iii) prototyping and conducting quantitative analysis with a test setup to experiment the proposed approach. Thus, the following result is derived: converting displacement predictions of dynamics obstacles into a time series of cost maps significantly reduces computation payload for agent's path plan and yield improved robustness in the navigation.

2. Literature review

Research on path planning methodologies for robots operating in dynamic environments has explored various solutions [1–5]. There are studies that propose methods for navigating environments with single or multiple dynamic obstacles, single or multiple agents, and finding optimal shortest path while considering various constraints. It must be remembered that the scenario proposed in each research is unique and therefore requires a specific solution.

2.1 Approaches for Dynamic Environments

Traditional motion planning algorithms in dynamic environments, such as A* search or dynamic window, often rely on currently observed information [6] and complex optimization problems [7] that struggle to effectively offer a solution when the number of dynamic agents increases [8], and their stationary model-based approach requires the path planning to be recalculated from scratch in the case that a deviation occurs in the environment. For example, the dynamic window approach usually assumes that all obstacles observed in the present remain stationary; consequently the agent may try to move into areas that will be occupied by a dynamic obstacle in the future. In this case, the agent would have to freeze or make a sharp turn, and restart fresh to reach the goal position. In order to realize smooth motion in a multi agent environment, future trajectories of obstacles must be considered [9].

2.2 Learning Based and Prediction Based Methods

Recent research has explored learning- and prediction-based methods to address the limitations of model-based planning frameworks in the complex and real-world scenarios [10]. Liu et al. [11] propose a decentralized structural recurrent neural network (DS-RNN) that stores the time-series data as a hidden state vector in the RNN [11] and used as the input to the controller. Dugas

et al. [12] proposes an end-to-end path planner using LiDAR data or local maps as inputs and outputs velocity commands. They represent the LiDAR data of robot's local environment as vectors in space by combining auto-encoders and transformers, to be able to transform the non-numerical data into numerical data, which is then used as the input to the controller supported with deep reinforcement learning (DLR). However, implementing a controller that handles all state probabilities takes significant computation and training, and in addition, the DLR method only determines the next action, and is therefore short-term.

2.3 Spatial and Temporal Analysis Implemented Methods

Time Enhanced A* search [13] is a multi-robot path planning using future occupied states, and it allows each agent to recognize the area it will occupy at each point by determining search priorities of the agents. While this approach provides a long-term planning strategy in the field of multi-agents, as the number of agents and missions increases, the prioritization of path search requires greater computation, resulting in suboptimal robustness.

D* (Dynamic A*) is an advanced path planning algorithm that extends the A* search algorithm to handle dynamic changes in the environment efficiently. It recalculates paths in real time when obstacles are encountered or when the environment changes [14–16], making it well-suited for applications where the map or terrain is not fully known in advance or is subject to changes. D* considers spatial aspects by finding the shortest path to the goal and adapting that path when obstacles appear. However, while D* can react to changes in real time, it primarily focuses on spatial adjustments rather than incorporating a predictive model that anticipates future states of the environment [2, 17]. In contrast, spatiotemporal path planning integrates both spatial and temporal analysis by predicting the future positions of obstacles [18] and dynamically adjusting the path based on those predictions. This allows spatiotemporal methods to proactively plan for potential future collisions rather than just reacting to present obstacles, offering a more comprehensive approach to navigation in highly dynamic environments.

Our research builds on existing research by trajectory prediction of dynamic obstacles, however, diverges with the implementation of Hermite cubic spline interpolation in spatial and temporal layers.

3. Research methods

Our method exploits the agent's observations to generate probability distributions for displacement of obstacles, which is then mapped as displacement predictions. In the initial step, the agent's position is estimated in gridworld coordinates from sensory input. After localization of the agent, obstacle observations in robot's coordinate system are transformed to gridworld coordinates with the added parameters: time t, obstacle ID h, position (x,y). From this point onward, the time value t is the representation of future time intervals with $t = 0$ being the current time. To predict trajectories of obstacles in future time intervals, we employ a combination of sparse graph convolution network (SGCN) and Hermite cubic spline interpolation (HCSI) in spatial and temporal dimensions.

3.1 Step 1: Finding Future Positions of Obstacles with Sparse Graph Convolution Network (SGCN)

The network focuses on capturing the interactions among obstacles by modeling them as a sparse graph where each node represents an obstacle [19], and edges represent interactions. The network generates multiple displacement probability distributions for each obstacle. These distributions are then mapped as discrete point estimates which represent future positions of obstacles at future time intervals. This step estimates where each obstacle will be located at a certain time step.

3.2 Step 2: Creating Trajectories by Interpolating Between Points with Hermite Cubic Spline

This step models the motion of obstacles by generating knots between two consecutive points of positions, and creating smooth curves of motion. After SGCN provides discrete point estimates for predicted future positions of obstacles, we must interpolate the discrete points into continuous curves for smooth transitions. To achieve this, first we define consecutive points and their first derivatives.

3.2.1 Defining Consecutive Points and Their First Derivatives

The sequence of predicted positions for each obstacle, which are denoted as $(x_0, y_0), (x_1, y_1), \ldots (x_n, y_n)$ where n represents the time step, are used to calculate the first derivative (or the tangents) at each point. This information gives us the direction of the movement.

3.2.2 Constructing the Hermite Cubic Spline

For each segment between consecutive points in the predicted paths generated by SGCN, Hermite Cubic Spline Interpolation (HCSI) is employed.

Hermite splines use four basis functions H_1, H_2, H_3, H_4 to interpolate between points.

$$H_1(t) = 2t^3 - 3t^2 + 1$$
$$H_2(t) = -2t^3 + 3t^2$$
$$H_3(t) = t^3 - 2t^2 + t$$
$$H_4 = t^3 - t^2$$

Using these basis functions, we compute the position $P(t)$ on the spline between P_i and P_{i+1} so:

$$P(t) = H_1(t)P_i + H_2(t)Pi + 1 + H_3(t)T_i + H_4(t)Ti + 1$$

where *t* varies from 0 to 1 for each segment. P(t) provides the curve between P_i and P_{i+1} and ensures the paths are continuous and there aren't abrupt or sudden changes in obstacle trajectories. Overall, the Hermite Cubic Spline *H(t)* for a segment between P_0 and P_1 with tangents T_0 and T_1 is given as:

$$H(t) = (2t^3 - 3t^2 + 1)P_0 + (t^3 - 2t^2 + t)T_0 + (-2t^3 + 3t^2)P_1 + (t^3 - t^2)T_1$$

3.3 Step 3: Spatiotemporal Mapping

3.3.1 Creating the Spatial Layer as Gridworld

Initialized with a 2D grid environment where each cell represents a specific area in the *x-y* plane and holds information about obstacle presence, uncertainties, etc. The resolution of this grid (i.e., the size of each cell) is chosen based on the agent's accuracy and the density of the obstacles.

3.3.2 Addition of the Temporal Layer

The 2D spatial grid is extended into a third dimension in *z*-direction which represents the layers of time. Each layer in z-direction corresponds to a specific time step. For instance, if predictions are made every 0.1 seconds, each layer would represent the state of the environment at that specific future time (*t=0, t=0.1, t=0.2*, ...). The maximum number of layers (or the depth of the grid) is determined by the prediction horizon (e.g., up to 2 seconds into the future).

3.3.3 Populating the Grid with Obstacle Position Predictions (Combining Spatial and Temporal Data with the Gridworld)

In the previous steps, we have already achieved these:

- Using the Sparse Graph Convolution Network (SGCN), predicted future positions of obstacles are generated. These predictions are made at discrete time intervals and provide spatial coordinates for each obstacle at each future time step.
- HCSI (Hermite Cubic Spline Interpolation) is applied to these

discrete points to smooth the trajectories and create continuous paths. The smoothed paths are then used to update the grid.

After continuous trajectories are generated with HCSI, each path segment is assigned to the corresponding grid map based on its predicted time frame. Each time step has its own grid layer, and the occupancy probabilities from smoothed paths are projected into these layers. For example, predictions for 1 second into the future would update the grid for *t=1*.

3.4 Step 4: Mapping Cell Occupancy Probability

For each time step, the predicted pedestrian positions are translated into probability distributions. This can be visualized as a heatmap where each grid cell is assigned a probability value indicating the likelihood of a pedestrian being present in that cell. The occupancy probability is higher near the predicted pedestrian positions and gradually decreases with distance. This distribution accounts for potential deviations in pedestrian movement, providing a buffer against sudden changes in direction.

Obstacle distribution O_t at time *t* in the gridworld coordinates is formulated as the sum of the O_{th}s for each obstacle with the equation below:

$$O_t = \sigma_1 \sum_{h \in H} O_{th} \quad (1)$$

$$O_{th} = \sigma_2 \sum_{s \in S} O_{pth}\,(x - x_{hs}, y - y_{hs}) \quad (2)$$

$$(x_{hs}, y_{hs}) \sim P_{(t-1)h}$$

Table 1: Experimental Settings

Number of dynamic obstacles	2
Agent	Mobile Robot
Microcontroller	Raspberry Pi 3
Motion Capture	MotionAnalysis Eagle x 18, Parallax 28015
Agent Motion	Stop, Straight, Autonomous Navigation

3.5 Step 5: Creating a Cost Map for Path Planning 3

Each cell in the spatiotemporal grid is assigned a cost value based on its occupancy probability. Occupied cells are assigned the highest cost (avoided by the path planner), caution cells are assigned intermediate costs (areas with lower probability of obstacle presence) and free cells are assigned the lower cost (preferred by the path planner). Uncertainties are treated as no-go buffer zones. The cells are defined by three cost $c(l)$ types as shown below:

$$c(l) = \begin{cases} C_0 \; if \; Occupied\big(d \le r_p + r_b\big) \\ C_c \; if \; Caution\big(r_p + r_b \le \mathrm{d} \le \mathrm{r_p} + r_b + b\big) \\ C_f \; if \; Free\big(r_p + r_b + b \le d\big) \end{cases}$$

3.6 Step 6: Searching Through Spatiotemporal Layers

The agent searches through these spatiotemporal layers to find a path from its current position to the goal. STP4 considers the costs across both spatial and temporal dimensions, ensuring that paths avoid future obstacle-occupied spaces. The path planner evaluates paths that extend into the future, checking each time layer for potential obstacles. This enables the robot to preemptively avoid areas that will be occupied by obstacles.

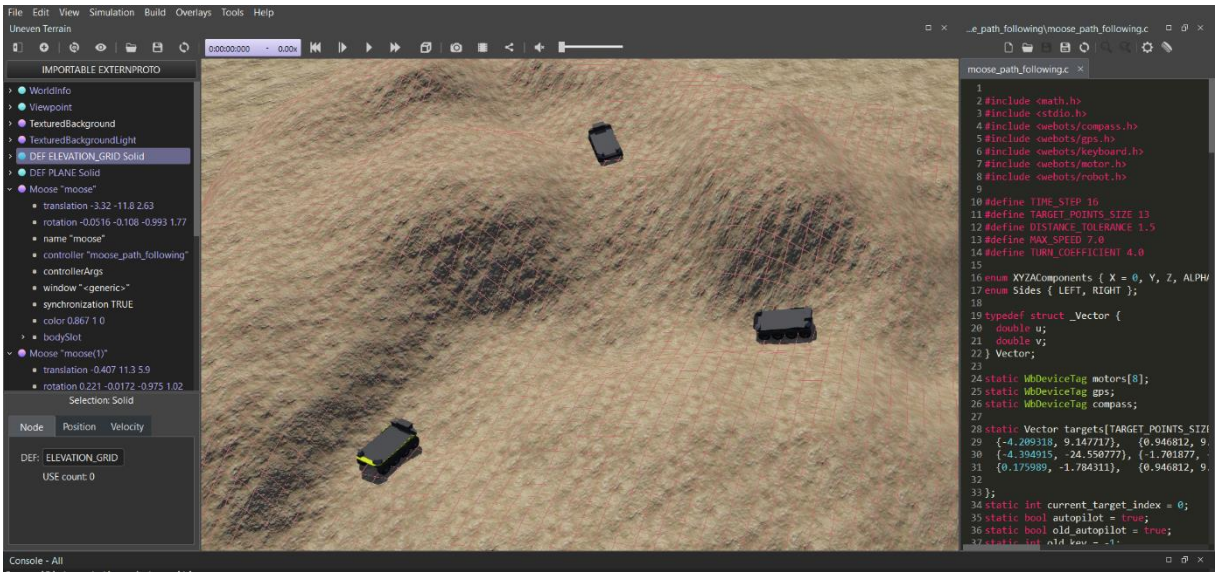

Figure 1: Simulation environment

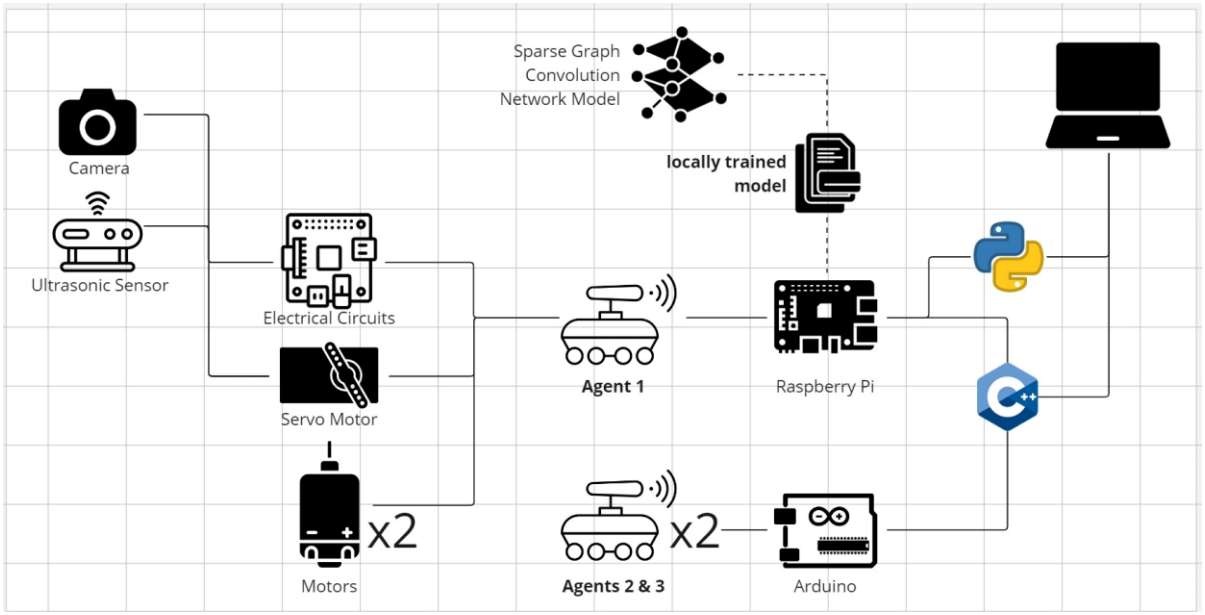

Figure 2: System Architecture

The prototyping process combines hardware and software components. Our system consists of three mobile robots and Raspberry Pi. The main algorithm runs on the Raspberry Pi on the main robot. The experimental settings are summarized in Table 1.

3.7 Concept development

The proposed experiment focuses on three mobile robots. The goal is to navigate the agent with a path plan which is updated in the presence of two dynamic constraints. The test was first conducted in a simulation environment and was designed in a mission area with three mobile robots, with one of them as the main agent and the other two as dynamic constraints. The mission area was confined to 65x65 fields in width and depth. Each individual field is 50x50 centimeter in size. Only the start and end goal were known to the agent.

3.8 Prototype development

The system architecture, as shown on Figure 2, integrates hardware components, microcontroller units, and software setup. A camera and ultrasonic sensor provide real-time environmental data, which is processed

through electrical circuits and servo motors to control the movement of multiple robotic agents. Raspberry Pi serves as the central processing unit, running locally trained models based on Sparse Graph Convolutional Networks to interpret sensor data and make path planning decisions using Python and C++ programming languages. Additional robotic agents are controlled via an Arduino microcontroller, demonstrating a coordinated multi-agent setup.

3.8.1 Hardware Assembly

The base structure of the robot includes a metallic chassis with mounting brackets for securing components. The main agent's chassis pieces were assembled, and Raspberry Pi was connected to the electrical circuit board. LiDAR, ultrasonic, and greyscale sensors were mounted on the robot. While the main agent is controller by Raspberry Pi, the secondary robots (that were used as dynamic constraints) were controlled by CyberPi and Arduino.

Raspberry Pi as central processor is mounted on the robot's chassis connected to the electrical circuit board that controls the electrical connections for motor, servo, camera, and so on. Raspberry Pi also handles sensor data processing and control signals for navigation and motion. Two direct current (DC) motors produce mechanical force, while three servo motors control rotation and angular movement, and control of the camera. Servos provide the robot with the ability to adjust its sensors dynamically based on environmental feedback.

The microcontroller board must be connected to a power supply unit, in the project a battery pack was used. The necessary circuitry includes various connections for interfacing sensors and actuators. Mounted at the front of the robot, the camera is used for visual input, capturing images or video for object detection, mapping and localization tasks. The camera is crucial for interpreting the environment and making informed path-planning decisions. The ultrasonic sensor is positioned beneath the camera, and is used for distance measurement, providing real-time feedback about obstacles or changes in the environment. It sends out ultrasonic waves and measures the time taken for the echo to return, allowing the robot to detect nearby objects and avoid collisions.

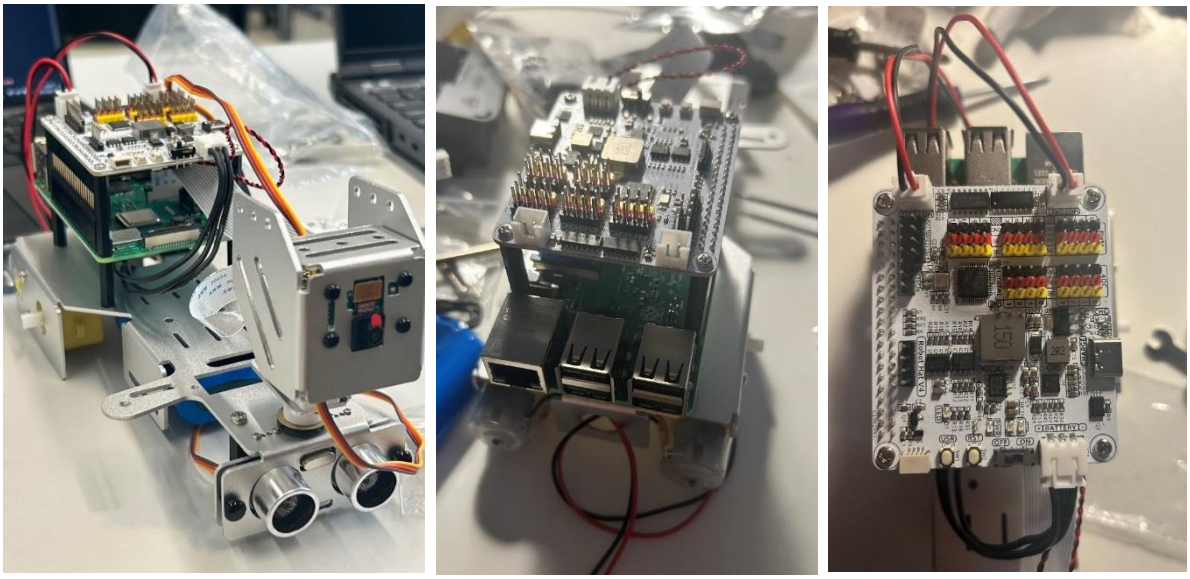

Figure 3: Robot chassis, camera, ultrasonic sensor, raspberry pi connection, electrical board connections

Figure 4: Raspberry Pi controls the primary robot

Figure 5: CyberPi controls the secondary robots

After installing the Raspbian OS on the Raspberry Pi, the camera module is connected to the Raspberry Pi's camera interface port. The ultrasonic sensor is attached to the GPIO pins, ensuring that the power, ground, and data lines are correctly wired. The Raspberry Pi will handle the main computational tasks, including image processing and running the trained model.

Arduino was connected to the Raspberry Pi via a USB connection for serial communication. Servo motors were wired to the Arduino's PWM-capable pins for control. DC motors were connected to motor drivers, which are then interfaced with the Arduino for movement control of the agents with motor control signals.

3.8.2 Software Setup

On the Raspberry Pi, install the necessary Python libraries as given in Table 3 below using pip, such as OpenCV for image processing, NumPy for data handling, and PyTorch for running the machine learning model.

3.9 Prototype demonstration

The test concept involves demonstrating the interaction between two robotic agents, Agent-1 and Agent-2, as they navigate towards a designated goal point. The objective is to evaluate how Agent-1 adjusts its speed and trajectory in response to the movement of Agent-2, which moves at a set motor speed. The test aims to observe whether Agent-1 can effectively use a spatiotemporal path planning algorithm to predict and avoid potential collisions based on the probability distribution of displacement. By analyzing the behavior of the agents under different conditions, the experiment seeks to refine and validate the effectiveness of the path planning and collision avoidance strategies employed. The experiment aimed to improve real-time responsiveness and ensure more accurate trajectory predictions.

4. Scientific results and discussion

4.1 Results and discussion

In the initial test, Agent-2 is put in motor motion of 60RPM. As demonstrated in Figure-6 that, it was observed that Agent-1 attempted to adjust its speed based on the probability distribution of displacement, however, did not manage to bridge the distance in time, and collided with Agent-2.

After adjustments in the algorithm and the mathematical calculations, as seen in Figure-7, the speed adjustment of Agent-1 was improved and the robot bridged the distance between start and terminal points, resulting the test positively.

Figure-10 shows the comparison of motion time and the kinematic profiles between regular motion and path planner motion. The top row shows the position over time, with the path planner achieving reduced motion time to 1.63s compared to regular motion of 2.57s.

Table 2: Hardware Components

Component	Amount
Robot Chassis Pieces	14
LiDAR Sensor	1
Ultrasonic Sensor	1
Greyscale Sensor	1
Raspberry Pi	1
Arduino	2
Servo Motors	3
DC Motors	2
Electrical Circuit	1
Wires	

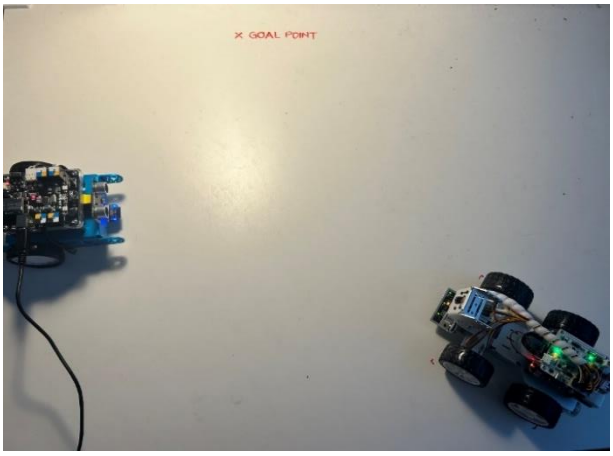

Figure 6: Testing (Agent-2 left, Agent -1 right)

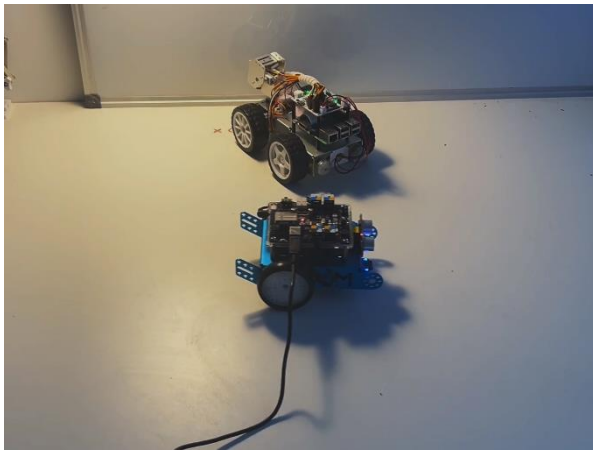

Figure 7: Positive result of the test

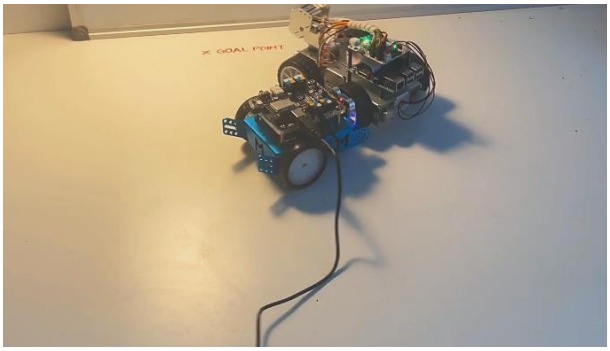

Figure 8: Due to faults in probability distribution, Agent-1's displacement was cut off before completion due to incorrect increase in its velocity and collision with Agent-2

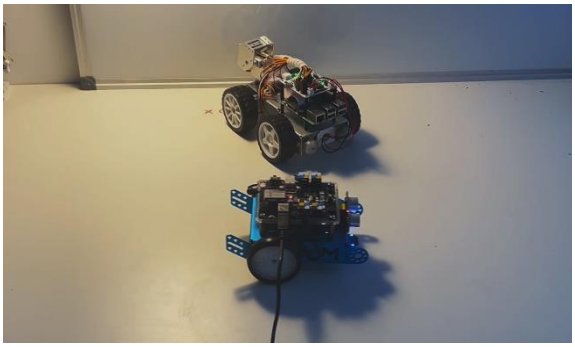

Figure 9: The probability distribution was corrected and Agent-1 managed to cross the distance in time to avoid collision with Agent-2

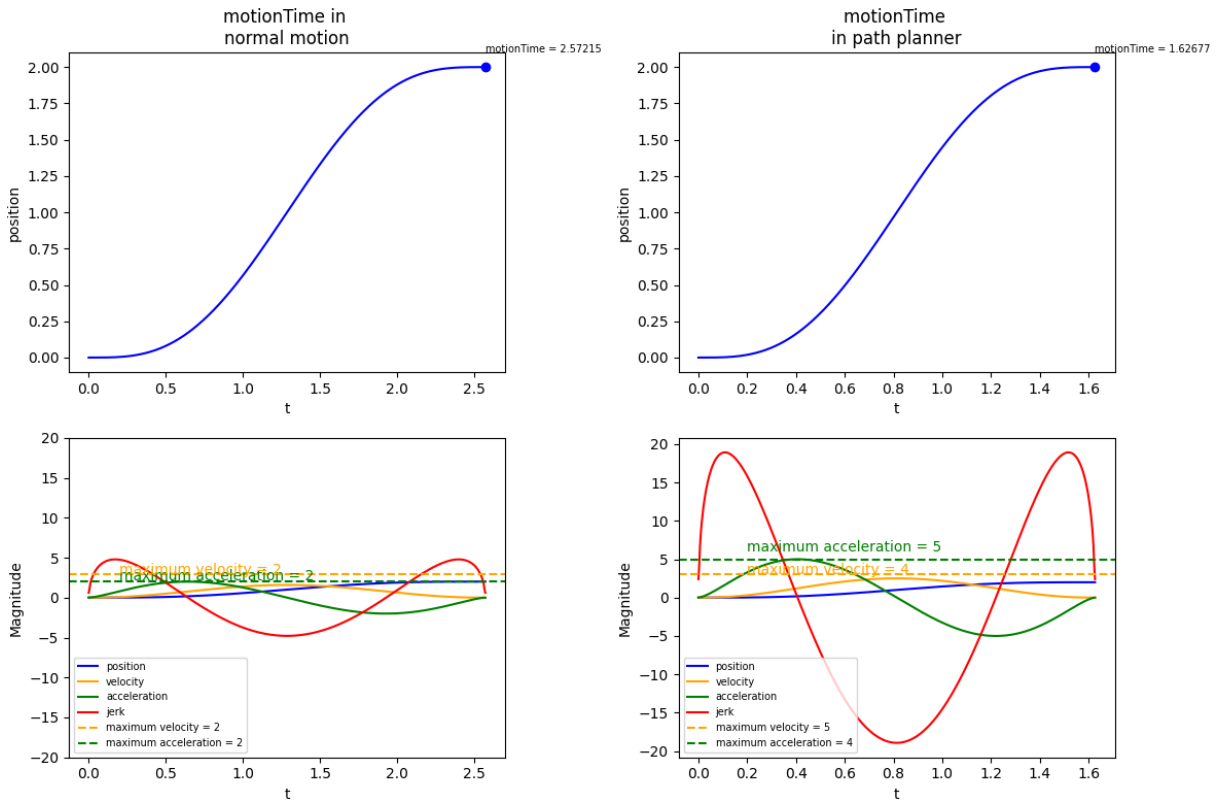

Figure 10: Comparison of motion time(top row) and kinematic profiles(bottom row) between regular motion(left side) and path planner motion(right side)

The bottom row displays position, velocity, acceleration, and jerk, indicating that the path planner allows higher velocity and acceleration limits, leading to faster completion. The results of the experiment show the difference between regular motion and the motion improved with the path planner. Over the course of 20 trials, the spatiotemporal path planning model achieved a reduction in motion time by 36.754% as calculated so:

$$Motion\ Time\ Improvement = \frac{2.57215 - 1.62677}{2.57215} \times 100$$

$$\frac{0.94538}{2.57215} \times 100 = 36.754466$$

In the given scenario and the specific positions of the agent and the constraint, the spatiotemporal path planned motion reached the terminal position in a shorter time frame. This indicates that the path planner can significantly reduce motion time by permitting higher velocity and acceleration within specified constraints, thereby optimizing the overall efficiency of the motion while still adhering to safety and performance limits.

The finding shows that optimizing path planning with the spatiotemporal method improves navigation efficiency and can adjust to real-time changes in dynamic constraint trajectories.

The application of path planning in environments with high dynamicity, such as spaces with non-stationary constraints or other robots, enable a mobile robot to navigate smoothly in dense crowds, avoiding abrupt movements and optimizing travel paths. our approach demonstrated that allowing higher velocity and acceleration constraints through path planning can lead to significant time savings without compromising safety or stability. These findings have direct implications for autonomous systems in urban mobility, robotics, and other applications where efficient and reliable navigation is critical.

The research supports that broader application of spatiotemporal path planning techniques, particularly in scenarios anticipating future positions of moving entities (e.g., public spaces, various active robots in the environment, or other forms of dynamic constraints) were helpful for optimizing

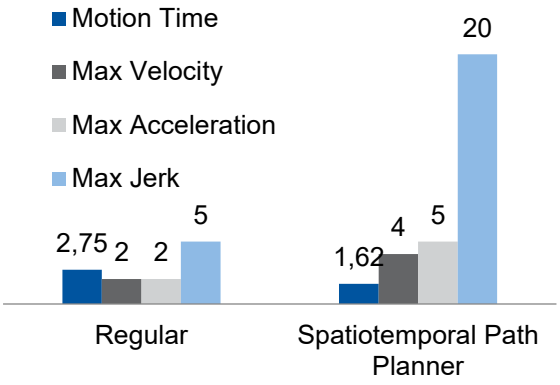

Figure 11 Comparison of motion time and kinematic profiles

navigation paths. By combining predictive models with path planning, as demonstrated in both studies, autonomous systems can achieve higher efficiency, reduce potential collision risks, and maintain smooth operation in complex environments. Future research may further explore integrating these methods to handle even more diverse and unpredictable scenarios, enhancing the robustness and adaptability of navigation systems.

5. Discussion

The spatiotemporal path planning project demonstrated the effectiveness of integrating advanced sensing, real-time data processing, and adaptive control strategies in autonomous robotic navigation. Through a series of tests and experiments involving multiple agents, the project successfully showcased the potential of combining hardware components like cameras, ultrasonic sensors, Raspberry Pi, and Arduino microcontrollers with sophisticated algorithms to achieve reliable and efficient navigation in dynamic environments.

The primary research question addressed in this project was: *Can* spatiotemporal *path planning techniques significantly improve the efficiency and safety of autonomous navigation in dynamic environments?* Based on the experimental results and analysis, the answer is positive. The use of spatiotemporal path planning, incorporating real-time adjustments to speed and trajectory based on environmental changes, led to a notable reduction in motion time and improved collision avoidance capabilities. These improvements confirm that spatiotemporal path planning can optimize the movement of autonomous agents, enhancing both efficiency and safety.

One of the critical findings of this project was the significant improvement in motion time and collision avoidance achieved by using a path planning algorithm that leverages spatiotemporal data. The comparison between normal motion and path-planned motion revealed that incorporating higher velocity and acceleration constraints led to a reduction in motion time by approximately 36.75%. These results validate the effectiveness of real-time path planning and prediction models in optimizing the performance of autonomous systems.

The project also highlighted the importance of fine-tuning the mathematical models and algorithms that control agent behavior. Initial tests showed that, without precise adjustments, Agent-1 struggled to avoid collisions with Agent-2, underscoring the necessity for accurate prediction and timely speed adjustments. Subsequent refinements to the algorithm improved the system's ability to predict movements based on the probability distribution of displacement, allowing Agent-1 to adjust its speed more effectively and avoid collisions. This iterative approach to algorithm development was crucial in enhancing the system's robustness and reliability.

Spatiotemporal path planning techniques could be applied in urban settings for autonomous vehicles, where predicting the movements of pedestrians and other vehicles is critical for safety and efficiency. In industrial automation, such systems could optimize the movement of robotic arms or mobile robots in manufacturing environments, reducing downtime and improving productivity.

Despite the successes, the project also faced several challenges. One primary challenge was the accurate prediction of dynamic obstacles, especially in scenarios involving sudden changes in speed or direction. While the improved algorithm showed better performance, there is still room for enhancing the prediction accuracy, particularly in highly dynamic or unpredictable environments. Future work could focus on integrating machine learning models that can learn from past experiences and adapt to new situations more effectively. Techniques such as deep reinforcement learning and neural

network-based prediction models could provide more robust solutions.

Recommendations for Future Work

Future research should focus on the following areas to enhance the capabilities and applicability of spatiotemporal path planning:

1. **Integration of Machine Learning Models**: Incorporating machine learning techniques, such as deep reinforcement learning or neural networks, could improve the system's ability to predict and respond to complex, dynamic environments. These models can learn from past experiences and improve their accuracy over time.
2. **Scalability**: Expanding the system to handle multiple agents in more complex environments is essential for practical applications. Future work should explore the coordination of multiple autonomous agents, ensuring they can navigate efficiently without collisions in crowded or unpredictable settings.
3. **Real-world Testing**: While the current project focused on controlled experimental setups, future research should include testing in real-world scenarios, such as urban environments with pedestrians and vehicles. This would provide valuable insights into the system's robustness and adaptability under real-world conditions.
4. **Improvement of Sensor Integration**: Enhancing the quality and integration of sensors, such as using more advanced LiDAR or depth cameras, can provide better environmental perception, improving the accuracy of path planning and collision avoidance.
5. **Long-term Adaptation and Learning**: Developing algorithms that allow the system to adapt and learn over the long term would be beneficial. This could involve the continuous updating of path planning strategies based on the evolving behavior of obstacles and agents within the environment.

Another area for future research involves scaling the system to handle more complex environments and multiple agents. As demonstrated in the research by Sato et al. (2023), handling dense crowds requires sophisticated coordination among multiple agents. Extending the current project to include more agents and more complex interactions could provide valuable insights into the scalability and flexibility of the proposed path planning techniques.

6. Conclusion

In conclusion, this project demonstrated the potential of spatiotemporal path planning in enhancing the performance and safety of autonomous robotic systems. By integrating real-time sensing, adaptive control, and predictive modeling, the system was able to navigate dynamically changing environments effectively. The findings provide a solid foundation for further research and development in autonomous navigation, with the potential to impact various industries and applications. Continued advancements in this field will likely lead to more intelligent, efficient, and safe autonomous systems, capable of operating in increasingly complex and dynamic real-world scenarios.

The knowledge gained from this project lays a foundation for further exploration and development in autonomous navigation. By addressing the identified areas for improvement and expanding the scope of testing, future research can build upon these findings to create more intelligent, responsive, and adaptable autonomous systems. As the technology continues to advance, the potential for safe and efficient autonomous navigation in increasingly complex real-world scenarios will become a reality, contributing to advancements in automation and the broader application of autonomous technologies.

7. References

[1] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, and A. Cuesta-Infante, "Mobile Robot Path Planning Using a QAPF Learning Algorithm for Known and Unknown Environments," *IEEE Access*, vol. 10, pp. 84648–84663, 2022, doi: 10.1109/access.2022.3197628.

[2] I. Maurovic, M. Seder, K. Lenac, and I. Petrovic, "Path Planning for Active SLAM Based on the D* Algorithm With Negative Edge Weights," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 8, pp. 1321–1331, 2018, doi: 10.1109/tsmc.2017.2668603.

[3] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997, doi: 10.1109/100.580977.

[4] Y. Sato, Y. Sasaki, and H. Takemura, "STP4: spatio temporal path planning based on pedestrian trajectory prediction in dense crowds," *PeerJ. Computer science*, early access. doi: 10.7717/peerj-cs.1641.

[5] X. Deng, R. Li, L. Zhao, K. Wang, and X. Gui, "Multi-obstacle path planning and optimization for mobile robot," *Expert Systems with Applications*, vol. 183, p. 115445, 2021, doi: 10.1016/j.eswa.2021.115445.

[6] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[7] Stachniss C. &Burgard W.(2002). An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS).

[8] T. Kulvicius, S. Herzog, M. Tamosiunaite, and F. Worgotter, "Finding Optimal Paths Using Networks Without Learning-Unifying Classical Approaches," *IEEE transactions on neural networks and learning systems*, early access. doi: 10.1109/TNNLS.2021.3089023.

[9] L. Yang, L. Fu, P. Li, J. Mao, N. Guo, and L. Du, "LF-ACO: an effective formation path planning for multi-mobile robot," *Mathematical Biosciences and Engineering*, vol. 19, no. 1, pp. 225–252, 2022, doi: 10.3934/mbe.2022012.

[10] S. Daftry *et al.,* "MLNav: Learning to Safely Navigate on Martian Terrains," Sep. 2022. [Online]. Available: https://arxiv.org/pdf/2203.04563

[11] Liu, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," p. 3517, 2021.

[12] D. Dugas, J. Nieto, R. Siegwart, and J. J. Chung, "NavRep: Unsupervised Representations for Reinforcement Learning of Robot Navigation in Dynamic Human Environments," Aug. 2020. [Online]. Available: https://arxiv.org/pdf/2012.04406

[13] J. Santos, P. Costa, L. F. Rocha, A. P. Moreira, and G. Veiga, "Time enhanced A*: Towards the development of a new approach for Multi-Robot Coordination," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, Seville, Spain, 2015, pp. 3314–3319, doi: 10.1109/ICIT.2015.7125589.

[14] S. K. Pattnaik, D. Mishra, and S. Panda, "A comparative study of meta-heuristics for local path planning of a mobile robot," *Engineering Optimization*, vol. 54, no. 1, pp. 134–152, 2021, doi: 10.1080/0305215x.2020.1858074.

[15] Y. Quan, H. Ouyang, C. Zhang, S. Li, and L.-Q. Gao, "Mobile Robot Dynamic Path Planning Based on Self-Adaptive Harmony Search Algorithm and Morphin Algorithm," *IEEE Access*, vol. 9, pp. 102758–102769, 2021, doi: 10.1109/access.2021.3098706.

[16] *Stentz A.(1995). The Focussed D*algorithm for real-time replanning. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

[17] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968, doi: 10.1109/TSSC.1968.300136.

[18] K. Mangalam, Y. An, H. Girase, and J. Malik, "From Goals, Waypoints & Paths To Long Term Human Trajectory Forecasting," in 2021, doi: 10.1109/ICCV48922.2021.01495.

[19] Shi, "SGCN: sparse graph convolution network for pedestrian trajectory prediction," p. 8994, 2021.