Autonomous Navigation System For Indoor Delivery Robots

Literature Review

By

Syed Muhammad Hussain CS-24	(sh06892@st.habib.edu.pk)
Syed Muhammad Daniyal CS-24	(sz06880@st.habib.edu.pk)
Muhammad Ahmed Atif CS-24	(ma06413@st.habib.edu.pk)
Laiba Ahmed EE-24	(la06855@st.habib.edu.pk)

November 28, 2023

In partial fulfillment of the requirement for

Bachelor of Science

Computer Science

Dhanani School of Science and Engineering

Habib University

Fall 2023

1 Introduction

This interdisciplinary project aims to develop an autonomous delivery system inspired by the Industry 4.0 paradigm, with a focus on SLAM (Simultaneous Localization and Mapping), Navigation, and Voice Bot technology. The project envisions a future where society functions with minimal human intervention, enhancing operational efficiency and safety. The project seeks to create a versatile autonomous mobile robot platform adaptable to various sectors, including industrial logistics, healthcare, unmanned missions, and security operations. For its final year milestone, the project focuses on the development of a self-driving delivery robot for indoor environments, notably the Habib University cafeteria. The robot will integrate SLAM to create a map of its surroundings and localize itself within the map. This will allow the robot to navigate safely and efficiently, even in dynamic and cluttered environments. The robot will also use navigation algorithms to plan and execute paths to its destination, avoiding obstacles and other hazards. An interactive food ordering system with a voice bot interface will enhance the user experience and streamline the ordering process. The voice bot will allow users to place orders using natural language, making the system more accessible and user-friendly.

2 Purpose Literature Review

The purpose of doing an extensive literature review is to gauge what work have already been done in the field of Mobile Robotics. The field mobile robots is as old as mid 20th century, hence there is alot of existing work already done to be expected and that which can greatly benefit this project. Doing the literature review ensures that we do not have to implement the desired algorithms from scratch, which can save us a lot of time and resources.

The following literature review focuses on the navigation aspect of mobile robotics. The literature review is divided into 3 parts, the first part is going to focus on different Global Planners used in Nav2 stack for ROS2, the second will go over different Local Planners used in Nav2 stack and lastly this document will breifly discuss about path smoothers available in Nav2 stack. One more reason to do literature review is to get reference for comparative analysis on different algorithm and which one would be better suited for this project.

3 Global Planners

The navigation system component in charge of creating a high-level plan or path to direct a mobile robot from its current location to a predetermined goal point within the environment is known as the global path planner. The global path planner takes into account the environment's general structure and functions at a higher level than local planners.

The global path planner's main function is to offer a high-level, long-term strategy that directs the robot through its surroundings, making sure it avoids obstacles and effectively accomplishes its objective. The local route planner, which functions at a lower level and takes the robot's immediate surroundings into account, is in charge of handling the finer points of obstacle avoidance and real-time changes.

3.1 Navigation Function Planner (NavFn)

A navigation function is based on A* path finding algorithm or Dijkstra's expansion. It assume 2D holonomic mobile robot. It was developed by Eitan Marder-Eppstein and Kurt Konolige.

Dijkstra Algorithm

A widely used technique in computer science and mobile robotics for figuring out the shortest path between nodes in a grid network is called Dijkstra's algorithm, after the Dutch computer scientist Edsger Dijkstra. The goal of the technique is to discover the most efficient path from a defined source node to every other node in the network, which works on graphs having weighted edges that reflect costs or distances. Initially, it establishes approximate distances, utilizes a priority queue to investigate nodes progressively more apart, and keeps adjusting distance estimations till the best routes are found. Because it can find the shortest pathways in a range of situations, Dijkstra's method is essential in applications like network routing and optimization. While Dijkstra's method can be effective in a complicated environment, it is less effective when the path to the destination cell is straightforward, such as a straight line.[1]

A* Algorithm

The A* method is a well-known pathfinding technique that is widely used in mobile robot navigation to discover the best route across a two-dimensional grid or map from a starting point to an objective. Combining the advantages of heuristic techniques and Dijkstra's algorithm, A* effectively explores the search space while taking the projected cost from the beginning to the end. It is particularly useful for exploring complicated surroundings because it uses a heuristic function, usually the Manhattan distance or the Euclidean distance, to direct the search towards the objective.The A algorithm reduces the number of cells visited by using a heuristic function that indicates the direction to the goal cell[2]. A* manages the open set of nodes using a priority queue, choosing the most promising node by adding the heuristic estimate and the total cost to reach that node.

Comparitive Analysis between the Dijkstra and A*

Map type	Grid scale	Minimum time/s			Maximum time consumption/s			Average time/s		
		RDJ	DJ	A*	RDJ	DJ	A*	RDJ	DJ	A*
Random	25 ×25	0.0041	0.2528	0.0629	0.0096	0.3165	0.2003	0.0061	0.2574	0.0656
Maze	25 ×25	0.0041	0.1207	0.0844	0.0078	0.1480	0.0939	0.0065	0.1296	0.0865
Warehouse	25 ×25	0.0043	0.1355	0.0942	0.0100	0.1461	0.1052	0.0067	0.1369	0.0966
Random	50 ×50	0.0146	1.1023	0.4090	0.0272	1.2411	0.4587	0.0200	1.1498	0.4306
Maze	50 ×50	0.0156	0.6759	0.6924	0.0943	0.6921	0.7101	0.0218	0.6790	0.6961
Warehouse	50 ×50	0.0121	0.5849	0.6745	0.0199	0.5977	0.7176	0.0181	0.5880	0.6795
Random	100 ×100	0.0509	4.8763	0.5808	0.1266	5.9406	0.6459	0.0566	5.2011	0.6048
Maze	100 ×100	0.0487	2.5128	1.7310	0.0654	2.5470	1.7479	0.0547	2.5222	1.7367
Warehouse	100 ×100	0.0405	2.4214	2.7325	0.0506	2.7174	2.9317	0.0427	2.4365	2.7469

Figure 1: Running time comparison of DJ and A* algorithm [3]

The Figure 1 shows that the for smaller problem sets the A* performs much better on average(initially with a difference in order of 10). However, as the grid size is increases the Dijstra

algorithm overtakes the A* and performs much better than A* algorith.

3.2 SmacPlannerHybrid

The SmacPlannerHybrid is another powerful global planner used in navigation. It is based on Hybrid A* algorithm which is a variant of classic A* algorithm discussed above.

Hybrid A* is created especially for mobile robot navigation in continuous state spaces is the Hybrid A* algorithm. Hybrid A* incorporates vehicle dynamics to overcome the difficulties of driving across continuous settings, in contrast to classic A*, which runs on discrete grids. It uses a hybrid system model to anticipate the robot's state and trajectory in a continuous space, and it makes use of a discretized lattice graph to describe possible pathways. By combining discrete and continuous approaches, Hybrid A* can provide more practical and realistic pathways for mobile robots while taking their dynamics and kinematics into consideration.

To ensure the kinematic feasibility of the calculated trajectories, the initial phase employs a heuristic search in continuous coordinates. The second step produces a path that is at least locally optimal and typically reaches the global optimum as well, using conjugate gradient (CG) descent to increase the quality of the solution locally.

The primary benefit of hybrid-state A* is evident during movements in confined places, where discretization errors reach a crucial level. [4] However, oOut of all the planners, the Smac Hybrid A-Star planner experienced the most collisions. Its performance might be deemed subpar when compared to the other planners, as it demonstrated longer navigation times and went larger distances on average. [5].

3.3 SmacPlannerLattice

Similar to SmacHybridPlanner, SmacPlannerLattice is another global planner used in Nav2 stack. SmacPlannerLattic was implemented by Steve Macenski[6] and is based on State Lattice Planner. A set of practical paths is formed by discretizing the robot's state space into a lattice structure using the State Lattice Planner mobile robot navigation method. Taking into account the robot's kinematic restrictions and dynamic limits, this lattice shows a range of potential routes it might follow. The robot's present state is explored by the algorithm to identify an optimum or nearly optimal path to a destination state. For non-holonomic robots, state lattice planners are especially useful since they can handle their complicated kinematics and guarantee that the trajectories that are created are practical in real-world situations. State Lattice Planners can be used for a variety of mobile robot navigation applications since this method strikes a compromise between computing efficiency and completeness.

Experimental Analysis

In experiment the State Lattice Planner is compared against Grid Planner in terms of time, memory consumption and the quality of resulting plan. The grid should perform better than the lattice in this obstacle field for all query classes, according to results obtained when obstacles are present. Interestingly, the lattice outperforms basic grid search when there are no barriers.[7]

The state lattice algorithm's primary benefit is the ability to choose numerous target endpoints as endpoints. It creates several reference trajectories in this manner, selecting the optimal one based on the predetermined cost function.[8]

3.4 ThetaStarPlanner

Another version of A* that is used in robot navigation is Theta-Star-Planner. It's an any-angle path planning variant of A* that spreads information along grid edges without tying pathways to grid edges. It integrates the concepts of A* on grids and A* on visibility graphs. On grids, its pathways are somewhat slower than A* but only marginally longer than actual shortest paths. It is identical to A* except that, when it updates the g-value and parent of an unexpanded visible neighbor s of vertex s in procedure Update Vertex, it considers two paths instead of only the one path considered by A*.[9]

Experimental Analysis

In experiment[9] comparison between Basic Theta* and AP Theta* to A* on grids, A* PS, FD* and A* on visibility graphs with respect to path length, number of vertex expansions, runtime (measured in seconds) and number of heading changes:

		FD*	Basic Theta*	AP Theta*	A* on Visibility Graphs (true shortest path)	A* on Grids	A* PS
100×100	Game Maps	40.04	39.98	40.05	39.96	41.77	40.02
	Random Grids 0%	114.49	114.33	114.33	114.33	120.31	114.33
	Random Grids 5%	114.15	113.94	113.94	113.83	119.76	114.71
	Random Grids 10%	114.74	114.51	114.51	114.32	119.99	115.46
	Random Grids 20%	115.20	114.93	114.95	114.69	120.31	116.16
	Random Grids 30%	115.45	115.22	115.25	114.96	120.41	116.69
500×500	Game Maps	223.64	223.30	224.40	N/A	233.66	223.70
	Random Grids 0%	576.19	575.41	575.41	N/A	604.80	575.41
	Random Grids 5%	568.63	567.30	567.34	N/A	596.45	573.46
	Random Grids 10%	576.23	574.57	574.63	N/A	603.51	581.03
	Random Grids 20%	580.19	578.41	578.51	N/A	604.93	585.62
	Random Grids 30%	581.73	580.18	580.35	N/A	606.38	588.98

Figure 2: Path length

		FD*	Basic Theta*	AP Theta*	A* on Visibility Graphs (true shortest path)	A* on Grids	A* PS
100×100	Game Maps	247.07	228.45	226.42	68.23	197.19	315.08
	Random Grids 0%	592.74	240.42	139.53	1.00	99.00	1997.29
	Random Grids 5%	760.17	430.06	361.17	35.35	111.96	1974.27
	Random Grids 10%	880.21	591.31	520.91	106.23	169.98	1936.56
	Random Grids 20%	1175.42	851.79	813.14	357.33	386.41	2040.10
	Random Grids 30%	1443.44	1113.40	1089.96	659.36	620.18	2153.28
500×500	Game Maps	6846.62	6176.37	6220.58	N/A	5580.32	9673.88
	Random Grids 0%	11468.11	2603.40	663.34	N/A	499.00	49686.47
	Random Grids 5%	15804.81	7450.85	5917.25	N/A	755.66	49355.41
	Random Grids 10%	19874.62	11886.95	10405.34	N/A	2203.83	50924.01
	Random Grids 20%	26640.83	18621.61	17698.75	N/A	6777.15	50358.66
	Random Grids 30%	34313.28	25744.57	25224.92	N/A	14641.36	53732.82

Figure 3: Number of vertex expansions

		FD*	Basic Theta*	AP Theta*	A* on Visibility Graphs (true shortest path)	A* on Grids	A* PS
100×100	Game Maps	0.0111	0.0060	0.0084	0.4792	0.0048	0.0052
	Random Grids 0%	0.0229	0.0073	0.0068	0.0061	0.0053	0.0208
	Random Grids 5%	0.0275	0.0090	0.0111	0.0766	0.0040	0.0206
	Random Grids 10%	0.0305	0.0111	0.0145	0.3427	0.0048	0.0204
	Random Grids 20%	0.0367	0.0150	0.0208	1.7136	0.0084	0.0222
	Random Grids 30%	0.0429	0.0183	0.0263	3.7622	0.0119	0.0240
500×500	Game Maps	0.1925	0.1166	0.1628	N/A	0.0767	0.1252
	Random Grids 0%	0.3628	0.1000	0.0234	N/A	0.0122	0.6270
	Random Grids 5%	0.4514	0.1680	0.1962	N/A	0.0176	0.6394
	Random Grids 10%	0.5608	0.2669	0.3334	N/A	0.0573	0.6717
	Random Grids 20%	0.6992	0.3724	0.5350	N/A	0.1543	0.6852
	Random Grids 30%	0.8562	0.5079	0.7291	N/A	0.3238	0.7355

Figure 4: Runtime

		FD*	Basic Theta*	AP Theta*	A* on Visibility Graphs (true shortest path)	A* on Grids	A* PS
100×100	Game Maps	247.07	228.45	226.42	68.23	197.19	315.08
	Random Grids 0%	592.74	240.42	139.53	1.00	99.00	1997.29
	Random Grids 5%	760.17	430.06	361.17	35.35	111.96	1974.27
	Random Grids 10%	880.21	591.31	520.91	106.23	169.98	1936.56
	Random Grids 20%	1175.42	851.79	813.14	357.33	386.41	2040.10
	Random Grids 30%	1443.44	1113.40	1089.96	659.36	620.18	2153.28
500×500	Game Maps	6846.62	6176.37	6220.58	N/A	5580.32	9673.88
	Random Grids 0%	11468.11	2603.40	663.34	N/A	499.00	49686.47
	Random Grids 5%	15804.81	7450.85	5917.25	N/A	755.66	49355.41
	Random Grids 10%	19874.62	11886.95	10405.34	N/A	2203.83	50924.01
	Random Grids 20%	26640.83	18621.61	17698.75	N/A	6777.15	50358.66
	Random Grids 30%	34313.28	25744.57	25224.92	N/A	14641.36	53732.82

Figure 5: Number of heading changes

From these result following conclusion were made:

- In terms of path length, Basic Theta* finds true shortest paths more often than FD* and A* PS.
- In terms of number of vertex expansion, Basic Theta* is 3rd best.
- In terms of runtime, Basic Theta* 2nd best runtime.
- In terms of number of heading change, Basic Theta* is 3rd best.

4 Local Controllers/Planner

In mobile robot navigation, local planners/controllers handle the problem of real-time obstacle avoidance and complicated environment navigation by concentrating on the robot's immediate surroundings. Local planners handle the fine-grained minutiae of mobility and obstacle avoidance, in contrast to global planners, which create high-level pathways from the start to the objective. Reactive methods—like potential fields—are popular in local planning because they allow the robot to react in real-time to its surroundings and change course to prevent collisions. Another method that takes into account the dynamics and limits of the robot to produce short-term plans is model predictive control. In tandem with global planners, local planners modify the robot's trajectory in response to unexpected impediments or dynamic shifts in the surroundings.

4.1 Dynamic Window Approach (DWA)

In mobile robotics, the Dynamic Window Approach (DWA) is a local navigation method that allows for real-time path planning and obstacle avoidance. Using its kinematic restrictions and sensor data, this technique dynamically computes a "window" of possible velocities and rotational speeds for the robot. The program generates a collection of possible routes by taking into account the robot's dynamic capabilities, its present condition, and the world around it. Subsequently, DWA assesses these paths by employing a cost function that considers the distance to barriers, objective alignment, and motion smoothness. The robot is guided through its immediate surroundings by the necessary control instructions once the trajectory with the lowest cost within the dynamically possible window is selected.

Experimental Analysis

In the experiment[10] RHINO robot is being used. The RHINO was using DWA as its local planner. Two setting were selected for this experiment:

- **Straight motion through corridor**: The average speed in this experiment was about 72 cm/sec, even if RHINO slows down to 55 cm/sec before crossing the obstacle. When feasible, RHINO drives in straight lines after navigating the obstruction.
- **Fast Motion through Cluttered Environment**: With a minimum speed of less than 20 cm/sec, all people in the corridor may be easily avoided at a maximum speed of 95 cm/sec. and 65 cm/sec on average.

4.2 TEB Controller

The robot's trajectory is locally optimized using the underlying technique, known as Timed Elastic Band, in terms of trajectory execution time, obstacle separation, and kinodynamic compliance. In the context of model predictive control, the TEB allows for the planning of time-optimal trajectories by combining the states, control inputs, and time intervals into a combined trajectory representation[11]. In a few rounds, the TEB technique produces a trajectory that is close to the analytical time optimum trajectory restrictions during runtime.

Technically speaking, Time Elastic Bands (TEB) alter the original global plan by generating a series of intermediate vehicle positions. It needs the vehicle's maximum speed and acceleration, the obstacles' safety distance, and the vehicle's geometric, kinematic, and dynamic limitations. When the vehicle is driving, this arrangement creates a sequence of directives for steering angle and speed (V), which are necessary to reach the intermediate waypoints.[11]

Experimental Analysis

The experiment[11], the TEB algorithm is compared to Time Optimal Model Predicitve Control (TOMPC) that constitutes a state-of-the-art MPC extension to time-optimal point-to-point control. The performance and potential of the TEB algorithm is analyzed on the control of two nonlinear systems:

- **Van-der-Pol Oscillator:**

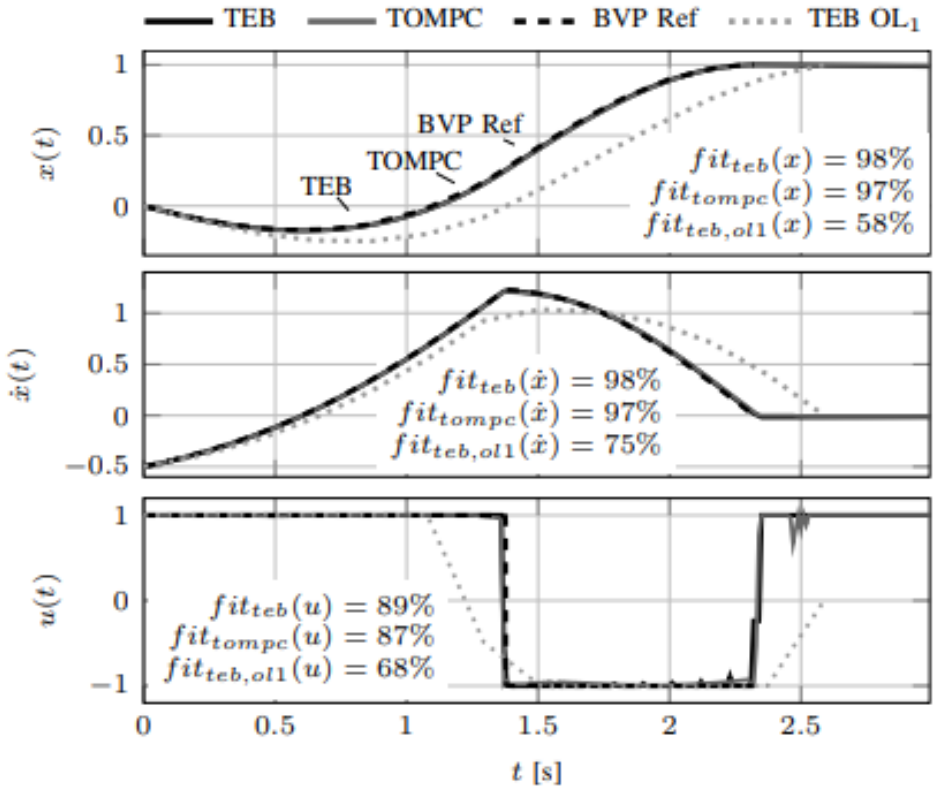

Figure 6: Closed-loop control of the Van-der-Pol oscillator

- **Free-Space Rocket System:**

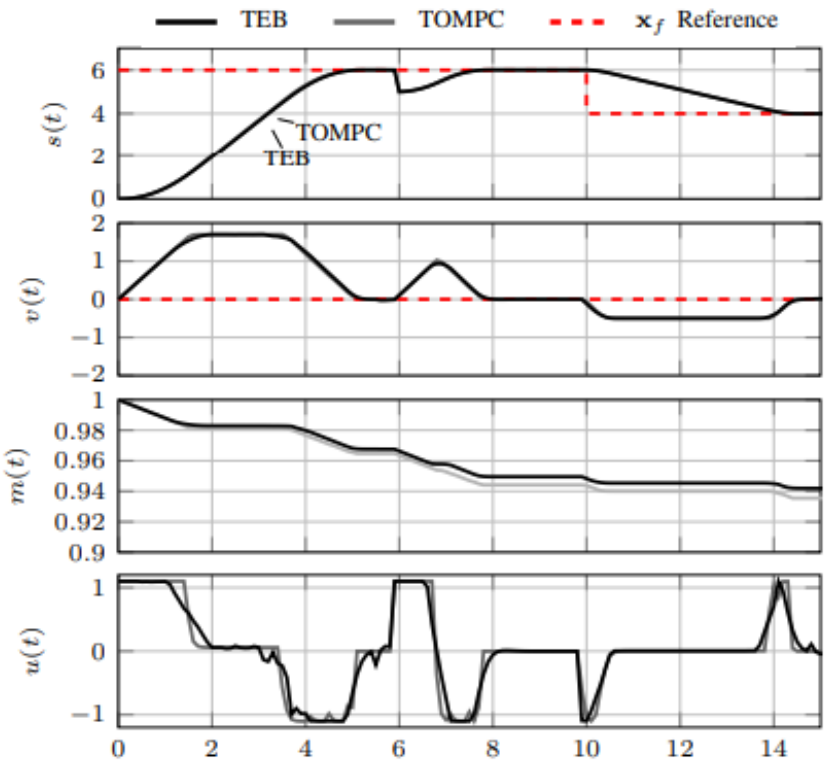

Figure 7: Closed-loop control of the free-space rocket system

Results

- The computational efficiency of the TEB method is demonstrated by a comparison with the most recent methods.
- The TEB can produce a workable approximation of time-optimal control within a few repetitions, compared to a state-of-the-art method.
- This research presents the performance of TEBs extended to Time-Optimal MPC tasks on two nonlinear systems.

4.3 Regulated Pure Pursuit Controller

Regulated Pure Pursuit is a mobile robot navigation algorithm that improves performance by adding further regulatory mechanisms to the Pure Pursuit method's simplicity. In Pure Pursuit, the robot is guided along a predetermined course, usually based on a lookahead distance, towards a target location. Refinements are introduced in Regulated Pure Pursuit to alleviate oscillations

and overshooting. In order to provide smoother control, it combines lookahead adaptation with speed regulation. This method modifies the lookahead distance dependent on the robot's velocity. Regulated Pure Pursuit, which combines regulatory features with Pure Pursuit, is ideally suited for mobile robot navigation applications. It alleviates some of the drawbacks of standard Pure Pursuit while offering a balance between simplicity and adaptability in a variety of environments.

4.4 MPPI Controller

An improved method of mobile robot navigation called Model Predictive Path Integral (MPPI) control combines path integral control with model predictive control approaches. Taking into account the restrictions and dynamic model of the robot, MPPI formulates the control issue as an optimization assignment. It uses an approach based on stochastic sampling to generate a collection of control trajectories and assess how well they function in the face of uncertainty. Path integration is then used to integrate these trajectories, and the probability distribution that results directs the choice of control inputs. The robot can navigate constantly changing situations because to MPPI's adaptable architecture, which can adjust to changing conditions. Mobile robot navigation in complex and unpredictable situations may be made more efficient and reliable with MPPI control, which combines probabilistic sampling with predictive modeling.

Algorithm 1 Model Predictive Path Integral Control based on Model Sampling

Require: T, X_{init}
  $t \leftarrow 0$
  while $t < N$ or $C(t)$ converges **do**
    $S(t) \leftarrow (S_t, ..., S_{t+T-1})$
    $\Sigma_{prior}, \mu_{prior}, \lambda, \nu \leftarrow GMM(S(t))$
    Update Σ, μ via Equation 5
    Update model $F_{x,u}$ via Equation 7, 8

    Roll-out the trajectories $C(t)$ with the dynamic model $F_{x,u}$
    Compute the weights $\omega(t)$ with $C(t)$ via Equation 11
    $U_t = \sum_{i=0}^{N-1} \omega(t)_i C(t)_i$
    $t \leftarrow t + 1$
  end while

Figure 8: Algorithm Pseudo-code [**?**]

Experiment Analysis

We gave the MPPI algorithm the task of navigating the AutoRally platform around a roughly ellipsoid track:

- Maximum outer diameter of 30 meters
- A uniform 3 meter wide driving surface.
- Test speeds: 8 m/s, 6m/s

The following equation was used to formulate cost:

$$2.5\left(V_{des}-V\right)^2+50.0h(p_x,p_y)^2+40.0C+10.0\|\mathbf{u}\|^2$$

Figure 9: Cost Formulation [14]

Results:

- **For 6m/s:**The controller is able to swiftly re-converge on a trajectory and steer the vehicle back toward the track's center in spite of these disruptions.
- **For 8m/s:** Only at this large difference in both quality and quantity between this setting and the 6 m/s level can the vehicle be suited this amount of track.

Performance Metric	6 m/s target	8 m/s target
Average Speed	5.67 m/s	6.15 m/s
Top Speed	7.75 m/s	7.98 m/s
Average Lap Time	11.26s	10.04s
Maximum Side-Slip Angle	43.99°	64.83°

Figure 10: Performance metrics for 6 m/s and 8 m/s test runs.[14]

5 Path Smoothers

In mobile robot navigation, techniques called path smoothers are used to improve and streamline the trajectories produced by route planners or path-following controllers. In addition to preventing collisions, these smoothers also seek to create pathways with less abrupt curves and more seamless transitions. For path smoothing, a variety of approaches can be used, such as curve-fitting techniques, optimization algorithms, and spline interpolation. The main goal is to make the robot's navigational pathways more intuitive and natural-looking, hence increasing its efficiency and safety. When mobile robots are working in dynamic or restricted situations, path smoothers come in very handy since a smooth trajectory is necessary for accurate and efficient motion planning. We will breifly discuss 2 of the path smoother that are used in Nav2 stack:

1. Constrained Smoother
2. Savitzky-Golay Smoother

5.1 Savitzky-Golay Smoother

Savitzky-Golay Smoother is based on Savitzky–Golay filter is a signal processing method for smoothing noisy data or signals that is frequently utilized in mobile robot navigation. When handling sensor measurements impacted by random fluctuations or disturbances, this filter is very helpful. It works by fitting a polynomial to small subsets of data points, and then estimating the smoothed values using the polynomial's coefficients. The Savitzky-Golay filter can be used to minimize noise and fluctuations in sensor data, such as location or distance measurements, in the context of mobile robot navigation. Smoothed data helps improve the robot's localization and perception systems' accuracy, which leads to more steady and dependable navigation in real-world settings.

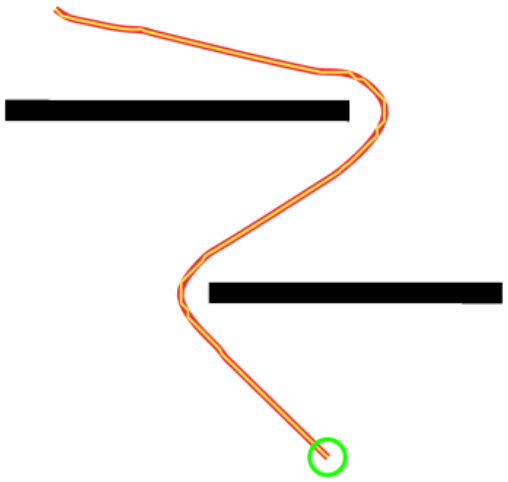

Figure 11: NavFn path (yellow) smoothed by Savitzky-Golay Smoother (red).[17]

Applications

- It was first used to find the location of maxima and minima in experimental data curves.[15]
- It is often used in resolution enhancement in spectroscopy.[16]

5.2 Constrained Smoother

An algorithm created to improve and optimize robot trajectories while abiding by certain limitations is known as a constrained smoother in mobile robot navigation. The smoothed path must not only be collision-free but also adhere to the physical limits of the robot. These constraints may include limitations on acceleration, velocity, or other dynamic elements. Constrained smoothers are critical components for mobile robots that must operate in situations where dynamic limitations must be strictly adhered to, including confined places or situations requiring smooth and controlled mobility. These algorithms help to provide courses that are not only more natural and devoid of obstacles, but also in line with the robot's physical capabilities. This makes navigation in intricate and dynamic environments safer and more efficient.

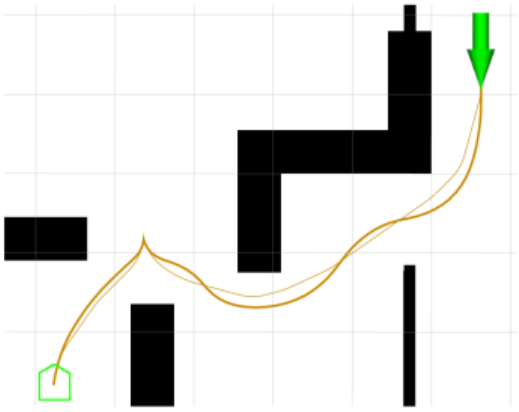

Figure 12: Smac Hybrid A* path smoothed by Constrained Smoother. Non-smoothed path depicted in narrow line, smoothed - in bold.[17]

Comparitive Analysis

	Time (ms)	Length (cm)	Ave. Cost	Max Cost	Smoothness	Ave. Turning Radius (m)
Unsmoothed Path	–	1150.47	12.26	129.46	76.45	0.71
Simple Smoother	0.90	1115.39	18.89	132.89	58.79	3.07
Constrained Smoother	20.82	1149.37	6.88	103.98	87.85	2.59
Savitzky–Golay Smoother	0.06	1146.75	12.18	127.76	76.19	2.29

Figure 13: Path Smoother Comparision[17]

References

[1] Ben-Ari, M., Mondada, F. (2018). Mapping-Based Navigation. In: Elements of Robotics. Springer, Cham. https://doi.org/10.1007/978-3-319-62533-1_10

[2] Xiang Liu and Daoxiong Gong, "A comparative study of A-star algorithms for search and rescue in perfect maze," 2011 International Conference on Electric Information and Control Engineering, Wuhan, 2011, pp. 24-27, doi: 10.1109/ICEICE.2011.5777723.

[3] Xueyan Li 2021 J. Phys.: Conf. Ser. 2083 042034

[4] Dolgov, Dmitri & Thrun, Sebastian & Montemerlo, Michael & Diebel, James. (2008). Practical Search Techniques in Path Planning for Autonomous Driving. AAAI Workshop - Technical Report.

[5] Recchiuto, CT, Sgorbissa, A. Post-disaster assessment with unmanned aerial vehicles: A survey on practical implementations and research approaches. J Field Robotics. 2018; 35: 459–490. https://doi.org/10.1002/rob.21756

[6] https://navigation.ros.org/plugins/index.html

[7] Pivtoraiko, Mihail. "Title of the Document" Carnegie Mellon University Robotics Institute, 2007, https://www.ri.cmu.edu/pub_files/pub4/pivtoraiko_mihail_2007_1/pivtoraiko_mihail_2007_1.pdf.

[8] Pivtoraiko, Mihail. "Generating Near Minimal Spanning Control Sets for Constrained Motion Planning in Discrete State Spaces" Carnegie Mellon University Robotics Institute, 2005, https://www.ri.cmu.edu/pub_files/pub4/pivtoraiko_mihail_2005_1/pivtoraiko_mihail_2005_1.pdf.

[9] Kenny, Daniel KF, et al. "Theta*: Any-Angle Path Planning on Grids." arXiv, https://arxiv.org/pdf/1401.3843.pdf.

[10] Fox, Dieter & Burgard, Wolfram & Thrun, Sebastian. (1997). The Dynamic Window Approach to Collision Avoidance. Robotics & Automation Magazine, IEEE. 4. 23 - 33. 10.1109/100.580977.

[11] Rosmann, C., Hoffmann, F., & Bertram, T. (2015). Timed-Elastic-Bands for time-optimal point-to-point nonlinear model predictive control. 2015 European Control Conference (ECC). doi:10.1109/ecc.2015.7331052

[12] Marín, Pablo & Hussein, Ahmed & Martín Gómez, David & de la Escalera, Arturo. (2018). Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles. Journal of Advanced Transportation. 2018. 1-10. 10.1155/2018/6392697.

[13] Wu, W., Chen, Z., & Zhao, H. (2019). Model Predictive Path Integral Control based on Model Sampling. 2019 2nd International Conference of Intelligent Robotic and Control Engineering (IRCE). doi:10.1109/irce.2019.00017

[14] Williams, Grady & Drews, Paul & Goldfain, Brian & Rehg, James & Theodorou, Evangelos. (2016). Aggressive driving with model predictive path integral control. 1433-1440. 10.1109/ICRA.2016.7487277.

[15] Savitzky, Abraham (1989). "A Historic Collaboration". Analytical Chemistry. 61 (15): 921A–3A. doi:10.1021/ac00190a744.

[16] Giese, Arthur T.; French, C. Stacey (1955). "The Analysis of Overlapping Spectral Absorption Bands by Derivative Spectrophotometry". Appl. Spectrosc. 9 (2): 78–96. Bibcode:1955ApSpe...9...78G. doi:10.1366/000370255774634089. S2CID 97784067.

[17] S. Macenski, T. Moore, D. Lu, A. Merzlyakov, M. Ferguson, "From the Desks of ROS Maintainers: A Survey of Modern & Capable Mobile Robotics Algorithms in the Robot Operating System 2", Robotics and Autonomous Systems 2023.