Project Proposal

Vision of the Project:
As a group, we want to explore different obstacle avoidance algorithms using TurtleBot4. We want to implement and analyze A*, D*, jump-A*, and DWA (Dynamic Window Approach). After we complete the analysis on these algorithms, we want to begin to combine algorithms and examine performance in different types of environments. We have a few stretch goals as well. We would like to explore more algorithms, such as PRM, and additional features for efficiency, such as potential fields. We will be using built-in ROS2 functions for SLAM in order to focus on creating and optimizing path planning and obstacle avoidance. We will start with a simulation in Gazebo and ROS, but want to apply it to a real-world map of mapped out obstacles. We also want to attempt obstacle avoidance with no prior knowledge about obstacles.

A more detailed description of your project, including final deliverables:
The project is going to focus on implementing different algorithms, followed by a detailed algorithm analysis comparing the ones we test. After comparing the initial algorithm analysis, we plan to compare 1 to 2 stacked algorithms (algorithms that use 2 or more traditional path planning/hybrid algorithms, e.g. A* and DWA combined) –should time allow–and test these in the same environments to compare our stacked/hybrid algorithms to the original algorithms we implemented. We will also conduct a full environment analysis, with the possibility of exploring how these algorithms handle dynamic obstacles. In the first term, we intend to do this through simulations, where we will have a written mini-report summarizing simulation results, discussing strengths and weaknesses per algorithm and including a justification for algorithm(s) chosen for real-world deployment.

These tests will be completed using the TurtleBot in various environments with a start and an end goal that are premapped with some unknown obstacles to show obstacle avoidance and path planning. We will conduct the information and show it in tables sorted by each environment we use. The variables we intend to measure are path distance, speed, obstacle avoidance, and computing time. After conducting the experiments and logging the information, we will analyze our data and complete our project with a comprehensive report that contains the following: background and literature, methodology and implementation details, comparative results and discussion of challenges, limitations, and future work.

Our presentation will break down the algorithms we used (and why), our goal, and use our physical, experiment and simulation data to justify our choices and how it applied to the TurtleBot. It will also demonstrate our process, results of the data, conclusions, and next steps.

Tables like this

Environment 1	Comparison 1	Comparison 2
Algorithm 1		
Algorithm 2		

Environment 2	Comparison 1	Comparison 2
Algorithm 1		
Algorithm 2		

A breakdown of who is likely to be focusing on what parts of the project:
We will split up into groups of two to use gazebo and implement our algorithms into a simulation before working with the robots winter term. We will initially split it up with a group of four and one group of two. The group of four is going to first work on A* because the algorithms D* and jumpA* both require A*. The group of two will be working on DWA in a simulation environment. If anyone finishes their portion early they can help other groups with theirs or work on possibly implementing PRM.

Initial group tasks:

Everyone: Set up Gazebo/ROS
Dexter, Daniel: DWA
Devin, Kat: A* -> D* lite
Raquel, Oliver: A* -> jumpA*
The first group to finish will start looking at PRM or help the other groups.
Determine successful algorithms, decide whether to continue with PRM

Group tasks for combining/testing algorithms:
Dexter, Daniel: combine DWA with A*
Devin, Kat: Combine DWA with D* lite
Raquel, Oliver: Combine DWA with jump A*

As a group, we define metrics and environments for comparative simulation tests
Dexter, Daniel: Test DWA, and hybrid DWA and A* in each environment
Devin, Kat: Test D* lite and hybrid DWA and D* lite
Raquel, Oliver: Test jump A* and hybrid DWA and jump A*

As a group, discuss the initial results and combine them
Dexter, Daniel: Work on optimizing DWA and hybrid DWAA* based on results
Devin, Kat: Work on optimizing D*lite and DWAD* lite based on results
Raquel, Oliver: Work on optimizing DWA and DWA jump A* based on results

Retest on simulation with improved algorithms
Dexter, Daniel: Test DWA, and hybrid DWA and A* in each environment
Devin, Kat: Test D* lite and hybrid DWA and D* lite
Raquel, Oliver: Test jump A* and hybrid DWA and jump A*

Combine Results, and begin drafting preliminary paper
Dexter, Daniel: Write about portion we completed
Devin, Kat: Write about portion we completed
Raquel, Oliver: Write about portion we completed

Depending on time left, begin testing and tweaking with turtle bot
Dexter, Daniel: Test DWA, and hybrid DWA and A* in each environment

Devin, Kat: Test D* lite and hybrid DWA and D* lite
Raquel, Oliver: Test jump A* and hybrid DWA and jump A*

Fall term:

Week	Goals / Tasks	Milestones / Outcomes
Week 5	Finalize simulation environment setup (Gazebo + ROS2). Design obstacle map layouts (static, mixed)	A working ROS–Gazebo environment with maps.
Week 6	Implement A* and D* in simulation (global planners)	A* and D* give feasible paths in simple maps.
Week 7	gazebo	All 4 algorithms running in simple maps.
Week 8	gazebo	Comparative results & initial observations.
Week 9	gazebo	Analysis of performance; weaknesses identified etc.
Week 10	gazebo	Submit mini-report to Chelsey

Winter Break:

Date	Goals / Tasks	
12/1-12/7	Finish implementing DWA	
12/7-12/20	Run comparative tests across basic scenarios (e.g. corridor, obstacle field). Collect metrics (path length, computation time, number of collisions or near-misses).	

Winter term:

Week	Goals / Tasks	Milestones / Outcomes
Week 1-2	Become familiar with the TurtleBot. Select 1–2 algorithms for TurtleBot deployment, merge findings into a mini report, and plan hardware integration.	TurtleBot moves around and can have code implemented into it successfully
Week 2-3	Implement selected algorithm(s) on TurtleBot 4, integrate perception (LiDAR) and localization (SLAM)	TurtleBot follows mapped paths in a controlled space
Week 3-4	Continue implementing selected algorithms onto TurtleBot4	TurtleBot follows mapped paths in a more complex controlled space with obstacles

		(known)
Week 4-5	Test in mapped static obstacle environment; log errors, failures, deviations	TurtleBot navigates successfully in at least one algorithm with unknown obstacles
Week 5-6	Adjust algorithms or settings as necessary and perform more testing	TurtleBot navigates with one or more algorithms successfully in different environments with unknown obstacles
Week 6-7	Test hybrid algorithms and/or dynamic environment (if time allows) in mapped static obstacle environment; log errors, failures, deviations. Begin writing the final paper.	Rough draft paper started while also succeeding in either one hybrid algorithm or one dynamic environment (if time allows).
Week 7-8	Clean code, write paper and work on presentation based on experiments and lit review	CODE FREEZE WEEK 8! Finish presentation and rough draft of paper
Week 8-9	Finalize paper/practice presentation and finish peer evaluations	Present comps and revise paper
Week 9-10	Finalize all material	All material ready for submission

- Code for the turtlebot to apply the hybrid algorithms we decided in the fall
- Test in static environment (no obstacles)
- Test in environment with obstacles
- If works, more tests or try to implement dynamic obstacles/glass
- If completed, test another algorithm or keep stacking more algorithms

Resources needed:

Gazebo
ROS2
GitHub
GitHub Desktop (recommended)
Cheap/free makerspace construction materials.

The first term we will be focusing on the simulator Gazebo. We will use a team GitHub for version control. We will be using ROS2 to eventually code to the robot (most likely) using Python.

In the second term we would like to construct some simple real world environments that are more complex than the basic hallway. For this we would like to use some basic cheap materials from the makerspace such as cardboard and plywood. The makerspace also has some basic sportsballs we could borrow for testing and return.

Literature review:

Recent advances in autonomous navigation have focused on optimizing both global and local path planning algorithms while leveraging improved robotic platforms for implementation and testing. Global planning algorithms such as A* and D* provide efficient overall path generation, with recent research by Xie, Qiang, and Yang (2020) and Choi et al. (2024) demonstrating significant improvements in computational efficiency and dynamic obstacle handling. Local planning algorithms, particularly DWA, excel at real-time obstacle avoidance but require integration with global planners to overcome their limited environmental awareness. Studies by Liu et al. (2021), Missura and Bennewitz (2019), and Kobayashi and Motoi (2022) have explored various modifications and hybrid approaches to enhance DWA's capabilities in complex environments. Emerging research by Wang, Yang, and Li (2022) suggests that AI-enhanced methods like Q-Learning offer promising directions for adaptive path planning, though integration with traditional algorithms like DWA remains largely unexplored. The Robot Operating System, particularly ROS2, has become the standard platform for implementing these algorithms, with Wei et al. (2025) and the ROS2 Real-time Performance Optimization study (2023) demonstrating substantial improvements in modularity, real-time performance, and system reliability. This literature review examines these developments in detail, exploring global path planning algorithms, local path planning approaches, hybrid and AI-enhanced methods, and the ROS2 platform capabilities that enable effective autonomous navigation systems.

Global Pathfinding algorithms: A* and D* Lite:

Recent research has placed emphasis on improving path planning algorithms to achieve both safety and efficiency in autonomous navigation systems. Standard approaches, such as the A* and D* algorithms, are effective for finding optimal paths in dynamic environments, but can struggle with the trade-offs between computational cost and avoiding collisions. Xie, Qiang, and Yang (2020) addressed these limitations. Their improved algorithm introduced a node priority strategy and adjusted heuristic and valuation functions, which helped it to keep a safer distance from obstacles but also reduced unnecessary node expansions. The simulation results showed 14.5% fewer expanded nodes and 12.7% faster computation time, meaning that it's possible to be more efficient but prioritize a safe buffer distance. Although there is typically slightly longer path, algorithms such as D*-Lite's adaptability for real-world applications requiring real-time adjustments and trajectory planning are still proven.

Additionally, Choi et al. (2024) extended the D* algorithm using a Human-Aware Trajectory Optimization approach, meaning using autonomous robots to navigate through walking, moving people. This presented challenges with unpredictable human movement in addition to non-moving obstacle avoidance. By collecting LiDAR and camera-based trajectory data from over 1,000 pedestrians, they found the average human avoidance distance (~1.77 m) and modified D* to decide whether the robot or pedestrian should stop based on sensor-detected behaviors. The modified D* algorithm allowed the robot to maintain its optimal path when pedestrians actively avoided it, improving travel efficiency by 7.5% and speed by 4.1% compared to the original algorithm. Implementing a human-aware modification demonstrated

the potential of D*-based algorithms to adapt to dynamic, semi-structured environments. It's a notable challenge for mobile robotics operating in shared human spaces. Our Turtlebot also uses LiDAR, making this application more important and lets us think about implementing moving obstacles through D*.

These studies exhibit the flexibility and efficiency of the D* algorithm family across different environments. It goes from air route planning to ground-level pedestrian navigation. For our TurtleBot4 project, which focuses on path planning and obstacle avoidance, these findings directly correlate to our goal. Xie et al.'s (2020) improvements emphasize algorithmic optimization for smoother and safer paths, ideal for TurtleBot navigation in indoor settings with lots of obstacles. Meanwhile, Choi et al.'s (2024) human-aware extension aligns with the need for real-time decision-making when the TurtleBot interacts with moving obstacles, such as people or other robots. Implementing similar concepts like the adaptive node expansion, changed heuristics, and behavioral prediction can help our TurtleBot system achieve both computational efficiency and socially compliant movement.

Local Pathfinding algorithms: DWA

Individual path planning strategies are generally divided into global path planning and local path planning. Global path planning takes into account the entire environment to determine a path while local path planning takes into account only the immediate surroundings to make adjustments to the current course. Methods such as A* and D* are considered global path planning algorithms, while DWA is considered a local path planning algorithm. DWA calculates the most optimal next step in terms of robot specifications by taking into account maximum acceleration and rotational acceleration to produce a vector space, translating it into 2-d movement, and calculating the most optimal route given a specified cost function. Although it can incorporate any information into its cost function, DWA itself does not cover global pathfinding strategies as it only refers to surroundings it can reach within the next frame of time. However, any number of methods can be added into the cost function.

Since DWA moves efficiently but cannot process the entire environment, L. Liu et al. worked on DWA combined with Jump-A*. Jump-A* is a global pathfinding algorithm that functions similarly to A* but includes a screening process for points. Instead of reviewing all points, Jump-A* only reviews the next point that leads to the goal until an event occurs (reaching the goal, hitting an obstacle, or necessitating a turn). By reducing the number of points the algorithm considers, Jump-A* reduces computational costs and improves processing speed.

The article (2021) aimed to create an algorithm that performs better than either one individually by taking global efficiency from Jump-A* and local efficiency from DWA. They incorporated the path produced by Jump-A* into DWA's cost function by giving the closest next Jump-A* point to DWA as a local goal rather than the actual target position, allowing DWA to follow the route point by point. However, they (2021) noted that the algorithm has issues with "falling into [a] local area and [being] unable to reach the target position." Despite this limitation, the results still showed a smoother path than Jump-A* alone and improved performance in complex environments with dynamic and static objects.

Beyond combining DWA with global planning algorithms, researchers have explored modifications to DWA's core functionality for dynamic obstacles. Missura and Bennewitz (2019) modified DWA to better account for moving objects, creating a new objective equation that considers temporary target locations rather than velocity. Their algorithm does not dismiss currently untraversable trajectories if the robot believes an object will move. Critically, they used simulations as their assumptions about environmental knowledge were difficult to achieve with current hardware.

Additional Algorithms: Q Learning and PRM

Wang, Yang, and Li (2022) explored Q-Learning for dynamic obstacle avoidance, improving convergence speed by prioritizing higher temporal difference error samples and implementing variable learning rates to prevent bias. Q-learning is a form of reinforcement learning in which a model learns the “best” action to take via trial and error, without using a model (meaning it can be completely unaware of its environment at the beginning of training). While their study applied Q-Learning independently, it demonstrated strong performance in handling dynamic obstacles. The combination of Q-Learning with DWA remains relatively unexplored in the literature and could enable adaptive cost function optimization, where DWA's weights are dynamically adjusted based on observed patterns and experiences.

Comparative studies have also identified PRM (Probabilistic Roadmap Method) as a promising alternative to traditional global planning algorithms. Al-Zubaidi, Ay, and Al-Khafaji (2023) found that PRM produced smoother paths than A* or RRT, though it required more processing time. Wang (2024) further compared multiple algorithms on the ROS platform and determined that PRM performed best overall in path efficiency and stability, suggesting its sampling-based design could be valuable for navigating complex environments. Both Q-Learning integration and PRM implementation represent potential stretch goals for future exploration as we gain familiarity with foundational path planning algorithms like A* and DWA. These alternatives would require additional computational resources and implementation complexity but could offer improved performance in specific scenarios, making them candidates for testing in Gazebo once baseline algorithms are established.

ROS2 Software overview:

Wu et al. (2023) and Tomović (2023) established foundational ROS-based autonomous navigation systems utilizing Gmapping for SLAM-based mapping, AMCL for localization, and integrated path planning algorithms like A* and DWA. These studies confirmed ROS as a reliable platform for linking mapping, localization, and planning functions in both simulated and structured real-world environments. Comparative analyses by Al-Zubaidi, Ay, and Al-Khafaji (2023) and Wang (2024) evaluated multiple path planning algorithms on ROS platforms, with findings showing A* produces the shortest paths with reasonable computation time, while PRM offers superior overall efficiency and stability for complex environments.

Wei et al. (2025) documented the transition from ROS 1 to ROS 2, highlighting the DDS-based decentralized system and Behavior Tree architecture that enables improved modularity,

flexibility, and real-time decision-making. They compared local planning methods, including DWA, TEB, and MPPI, revealing trade-offs between speed, accuracy, and computational cost, while noting emerging trends in AI-assisted navigation and socially aware robotics. The ROS2 Real-time Performance Optimization study (2023) demonstrated significant improvements using the Preempt_RT Linux kernel patch, reducing latency from 6243 µs to 82 µs and ensuring stable performance under heavy processing loads—critical for precise motion control and sensor fusion.

Senczyszyn et al. (2024) emphasized environmental considerations, investigating how factors like rain, dust, and snow affect depth sensor performance and consequently mapping accuracy. These insights underscore the importance of testing navigation algorithms under varied simulated conditions in Gazebo before real-world deployment on platforms like the TurtleBot, ensuring robustness across diverse operating environments.

Literature Summary:

This literature review examined path planning algorithms and ROS2 implementation for autonomous navigation. Global planners like A* and D* have been optimized for efficiency and safety, achieving better travel efficiency. Local planning algorithms, particularly DWA, require integration with global planners to overcome environmental limitations. Hybrid approaches combining DWA with Jump-A* and modified cost functions show promise in dynamic environments. Alternative algorithms including Q-Learning and PRM represent stretch goals for adaptive optimization. ROS2 provides the implementation platform with improved real-time performance (latency reduced from 6243 µs to 82 µs) and comprehensive navigation tools. These findings establish a foundation for testing algorithms in Gazebo simulation before TurtleBot4 deployment.

References

Al-Zubaidi, Z. M., Ay, S., & Al-Khafaji, M. (2023). *A Comparative Study of Various Path Planning Algorithms for Pick-and-Place Robots*. Altınbaş University and University of Technology Production Engineering and Metallurgy Department. https://doi.org/10.21203/rs.3.rs-2808265/v1

C. Wang, X. Yang and H. Li, "Improved Q-Learning Applied to Dynamic Obstacle Avoidance and Path Planning," in *IEEE Access*, vol. 10, pp. 92879-92888, 2022, doi: 10.1109/ACCESS.2022.3203072.

K. Xie, J. Qiang and H. Yang, "Research and Optimization of D-Start Lite Algorithm in Track Planning," in *IEEE Access*, vol. 8, pp. 161920-161928, 2020, doi: 10.1109/ACCESS.2020.3021073.

L. Liu et al., "Global Dynamic Path Planning Fusion Algorithm Combining Jump-A* Algorithm and Dynamic Window Approach," in IEEE Access, vol. 9, pp. 19632-19638, 2021, doi: 10.1109/ACCESS.2021.3052865.

M. Je Choi, S. Jin Park, S. Kim and S. Jae Lee, "Human-Aware Trajectory Optimization for Enhancing D* Algorithm for Autonomous Robot Navigation," in *IEEE Access*, vol. 12, pp. 103237-103250, 2024, doi: 10.1109/ACCESS.2024.3430352

Senczyszyn, S., Pinar, A. J., Salem, M., Donoghue, E., Wills, S., Broestl, M., Webb, A. J., Havens, T. C., & Price, S. R. (2024). *Comparing performance of robot operating system (ROS) mapping algorithms in the presence of degraded or obscured depth sensors.* In *Proc. SPIE 13052, Autonomous Systems: Sensors, Processing, and Security for Ground, Air, Sea, and Space Vehicles and Infrastructure 2024* (130520F). https://doi.org/10.1117/12.3013420

Tomović, A. (2023). *Path Planning Algorithms for the Robot Operating System.* MICS Conference, Saint Cloud State University, MN.

Wang, S. (2024). *Comparative research on path planning algorithms for autonomous mobile robots based on ROS.* In *Proc. SPIE 12969, International Conference on Algorithm, Imaging Processing, and Machine Vision (AIPMV 2023)* (129690Z). https://doi.org/10.1117/12.3014664

Wei, Z., Wang, S., Chen, K., & Wang, F. (2025). ROS-Based Navigation and Obstacle Avoidance: A Study of Architectures, Methods, and Trends. *Sensors (Basel)*, 25(14), 4306. https://doi.org/10.3390/s25144306

Wu, J., Jin, X., Sun, K., Zhou, J., & Zhang, J. (2023). *Design of Autonomous Navigation Robot Based on ROS System.* In *The 3rd International Conference on Electronic Information Technology and Smart Agriculture (ICEITSA 2023)*. ACM. https://doi.org/10.1145/3641343.3641349

ROS2 Real-time Performance Optimization and Evaluation. (2023). [Primary Source Summary].