ROS-Driven Autonomous Navigation: A Scalable Framework for Wheeled Robots in Dynamic Environments

Koushik Paul[1], Abhisri Dugar[2], Abhishek Dugar[3], Abhradeep Hazra[4], Aronno Ghosh[5]

[1,4]Student (MECHANICAL) KIIT Deemed to be University, Bhubaneswar, India
[2]Student (CSE) KIIT Deemed to be University, Bhubaneswar, India
[3]The heritage school, Kolkata
[5]E.E.E. (Electronics and Electrical Engineering) KIIT Deemed to be University, Bhubaneswar, India

Abstract
Autonomous navigation is important for wheeled robots in dynamic and unstructured environments. This paper details a ROS-based framework for motion planning that gives a robust and modular platform for the development and integration of robotic systems. The methodology involves robot modeling, environment mapping, path planning, control strategies, and execution mechanisms. It is built on ROS packages and tools like RViz, Gazebo, and MoveIt! It has to efficiently plan motion. This is simulated in details and even run in real-life. From both simulation and the real-world analysis, this shows that such a system would really navigate even constrained environments such as smooth trajectory generation or obstacle avoidance. This shows scalability and applicability of such systems, based on the ROS architecture, towards highly variant applications starting from industrial automation all the way up to an autonomous vehicle.

KEYWORDS: Autonomous Navigation, Motion Planning, Robot Operating System (ROS), Dynamic Environments

Introduction:

Autonomous navigation is, by all measures, a principal aspect of state-of-the-art robotics for wheels that traverse dynamics and unstructure. In industries, urban communities, and their natural habitats, robots' sophisticated performances increase further the necessity to design and test navigation systems of robust, flexible, and efficacious performances. In many such dynamic settings, conventional and fixed approaches simply break down with their performance failing with a certain unexpectable inconsistency in their behavior; thus, designing navigation systems based on a completely different set of rules becomes unavoidable.

This paper presents a comprehensive framework for motion planning with the Robot Operating System, an open-source flexible framework for writing robot software. Because of its large suite of tools, libraries, and conventions, ROS has become the standard in the robotics community for simplifying the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

The proposed framework will integrate the major components of autonomous navigation, which include robot modeling, environment mapping, path planning, control strategies, and execution mechanisms. All these will be implemented using ROS packages and tools, such as RViz for visualization, Gazebo for simulation, and MoveIt! for motion planning. This modularity allows easy customization and scalability to be applied in various robotic applications, ranging from industrial automation to autonomous vehicles. The environment mapping is essential in the understanding of the surroundings of the robot and makes it appropriately decide on the path to be followed. Path planning algorithms are used to find feasible and efficient trajectories, taking the robot from the point where it is at the beginning to where it should be at the end while avoiding obstacles. Execution mechanisms translate plans into real-world actions, and control strategies ensure that the robot correctly follows the planned path.

It tests its performance by running simulation experiments besides running experiments in real settings. These experiment results demonstrate a system that is well-performing under complex environments with the existence of moving obstacles and obtains smooth paths by following previously predefined constraints. An approach like the ROS-based has been proven to be highly flexible and feasible with the opportunity for extension over varying types of robots and varied scenarios of operating situations.

Function	**Description**	**Tools/Components Used**
Robot Modeling	Creating a digital representation of the robot to simulate and plan its movements accurately.	URDF, RViz
Environment Mapping	Building a map of the robot's surroundings to understand the layout and obstacles	SLAM, RViz
Path Planning	Calculating optimal paths from the start to the destination while avoiding obstacles	MoveIt!, OMPL
Control Strategies	Developing control algorithms to ensure the robot follows the planned path smoothly and accurately.	PID Controllers, ROS Control
Execution Mechanisms	Translating path plans into executable commands for real-world navigation.	ROS Nodes, Gazebo
Simulation	Testing and validating navigation strategies in a virtual environment before real-world deployment.	Gazebo, RViz
Obstacle Avoidance	Implementing algorithms to detect and maneuver around dynamic and static obstacles.	Laser Scanners, Depth Sensors
Trajectory Generation	Creating smooth, continuous paths that meet predefined constraints for safe and efficient	MoveIt!, ROS Trajectory Planner

	navigation.	
Real-World Evaluation	Assessing the framework's performance in actual environments to ensure reliability and robustness	Physical Robot Deployment

Literature Review:

The field of autonomous navigation for wheeled robots has seen significant advancements due to the development of frameworks and methodologies for motion planning. This section reviews relevant works in robot modeling, environment mapping, path planning, and control strategies, with a focus on systems leveraging the Robot Operating System (ROS).

1. Robot Modeling

Motion planning is based on robot modeling. It allows proper description of kinematics and dynamics of a robot. For instance, for wheeled robots, differential drive and skid-steer models are very important according to Siciliano et al. (2010). Indeed, current tools, like URDF in ROS, have made definition of parameters of the robot much easier. Hence, researchers can focus on the higher-level planning algorithms.

2. Environment Mapping

The central task of autonomous navigation is environment mapping. Algorithms like GMapping (Grisetti et al., 2007) and Cartographer are rather popular for producing 2D and 3D maps. Generally, the data from LiDAR, cameras, and IMUs is fused to create rich maps. Both ROS and SLAM algorithms can easily be connected so that mapping and localization can be done in real time.

3. Path planning

Path planning is a highly researched area. The core algorithms that most motion planning frameworks rely on are A*, D*, and RRT. For wheeled robots, the planners should account for the type of terrain and non-holonomic motion constraints. Global planners like Dijkstra and A* are present in the ROS Navigation Stack. There are local planners like DWA and TEB that optimize the trajectory while avoiding obstacles.

4. Control Strategies

Control strategies control the robot along the planned trajectory with stability and accuracy. Most of the methods of navigation employed are PID controllers and MPC. The work in Goodwin et al. (2001) has shown such methods to be very effective in the case of wheeled robots. ROS helps implement control packages like ros_control, which are able to follow feedback loops as well as adapt dynamically.

5. ROS as a Framework for Motion Planning

The adoption of ROS has revolutionized robotic development by offering modularity, scalability, and community support. Researches highlight the evolution of ROS into ROS 2, enhancing real-time capabilities and reliability for motion planning applications. Tools like RViz and Gazebo allow for visualization and simulation, accelerating the development and testing of autonomous navigation systems.

6. Applications and Challenges

Though autonomous navigation has been applied in some domains like logistics, agriculture, and autonomous vehicles, it faces different challenges, such as dynamic obstacle avoidance, real-time decision-making, and multi-robot coordination. Recent advances in the areas of machine learning and reinforcement learning hold promising means of addressing some of these issues.

METHODOLOGY

The methodology in the development of an autonomous navigation framework using ROS comprises a network of inter-related elements to integrate into the smooth performance of navigating efficiently and in real time and in a highly dynamic and unstructured environment. This approach ensures to make maximal utilization of capabilities by ROS but solves specific issues for autonomous navigation.

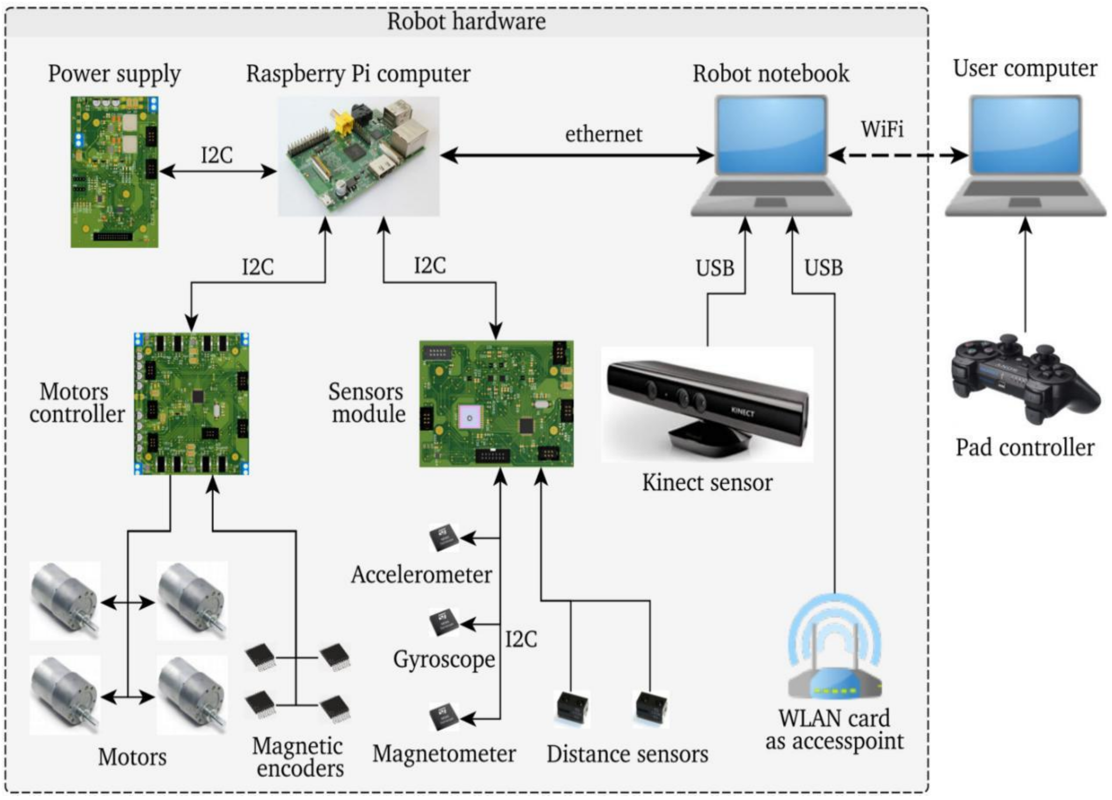
Robot Modeling

The first step was to make the correct model for the robot based on its design, which refers to the overall physical dimensions or the kinematic and dynamic description, and it made use of a URDF structure for describing robot structure in a standardized way about ROS. On the other side, RViz is used here for visualization where the model so designed can precisely be matched by the physical prototype of the same robot. This is the base step in simulation and motion planning, because all further calculations are made upon it.

Component	**Description**	**Tools/Techniques Used**
Methodology Overview	The framework integrates interrelated elements for efficient, real-time navigation in dynamic environments	ROS, Modular Approach
Robot Modeling	Accurate robot model creation based on physical dimensions, kinematics, and dynamics.	URDF, RViz
Visualization	Ensuring the designed model matches the physical prototype for accurate simulation and planning.	RViz

Environment Mapping

In order to move properly, the robot must perceive its surroundings. Techniques like SLAM make it possible for the robot to construct a map of the space in which it moves, at the same time figuring out its location in the map. LIDAR sensors, cameras, and depth sensors provide the information gathered and are processed using packages from ROS such as GMapping or Cartographer, in real time. The map is a reference for the robot to plan its paths and avoid obstacles.



Path Planning

The path-planning algorithm for the robot should compute the minimum distance it has to travel from its current position to the destination. Planning algorithms are determined based on a map and known obstacles, combined with desired trajectories that allow for smooth navigation. There are a wide variety of path-planning algorithms, including RRT for Rapidly-exploring Random Tree, A* implemented through MoveIt! and the Open Motion Planning Library, among others, that generate possible paths the robot can use without collision in order to navigate.

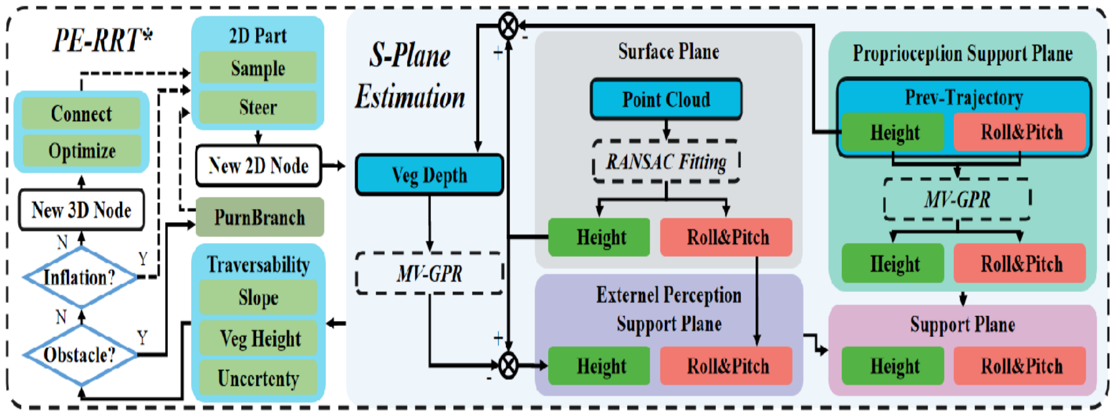


Fig. 2. Overview of the algorithm. From left to right: To generate the RRT tree, 'Sample' and 'Steer' operations are performed to generate a new 2D

Control Strategies

Once the trajectory is planned, the robot must execute the trajectory effectively. Control strategies are developed to cause the robot to follow the planned trajectory and also deviate from that trajectory when there is a dynamic obstacle or environmental change. PID controllers are widely used in controlling the robot's speed and steering angles. ROS Control has an infrastructure in place for their implementation and management.

Execution Mechanisms

Mechanisms of execution will translate the paths planned into executable commands by motors and actuators of the robot. Communication would be managed with the help of ROS nodes where sensor data would be synchronized and control commands are also synchronized together with planning output. A virtual controlled setting would be created as a simulation environment, such as Gazebo, in which execution mechanisms may first be tested prior to deployment to the real world.

Simulation and Testing

It tests the whole framework through extensive simulation in Gazebo. In this phase, various kinds of obstacles with dynamic environmental changes are considered in various scenarios. This way, possible problems that may be encountered during the simulation phase can be detected and resolved; hence, reliability of the system is greatly enhanced for real-world applications.

Real-World Evaluation

The final step is testing the robot in real environments to check its performance. Real-world tests are conducted in different settings, such as industrial warehouses or outdoor environments.

RESULT & ANALYSIS:

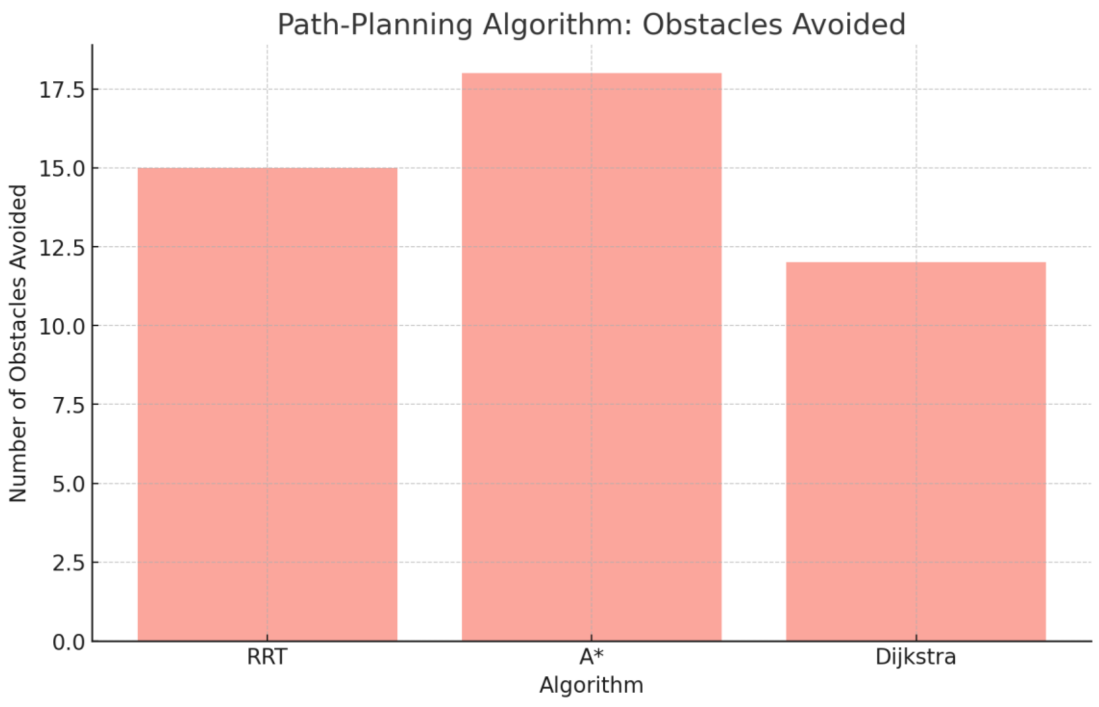
Recent research in the field of autonomous robotic navigation incorporated SLAM with advanced path-planning algorithms for better navigation of a robot through dynamic environments using ROS. SLAM, using LIDAR, cameras, and depth sensors, enables the robot to build its maps and maintain localization simultaneously. Another value is the real-time mapping and localization capability. Packages like ROS/GMapping/Cartographer could also be useful in navigating precisely in the field without hitting objects inside the space.

CODING PORTIONS:

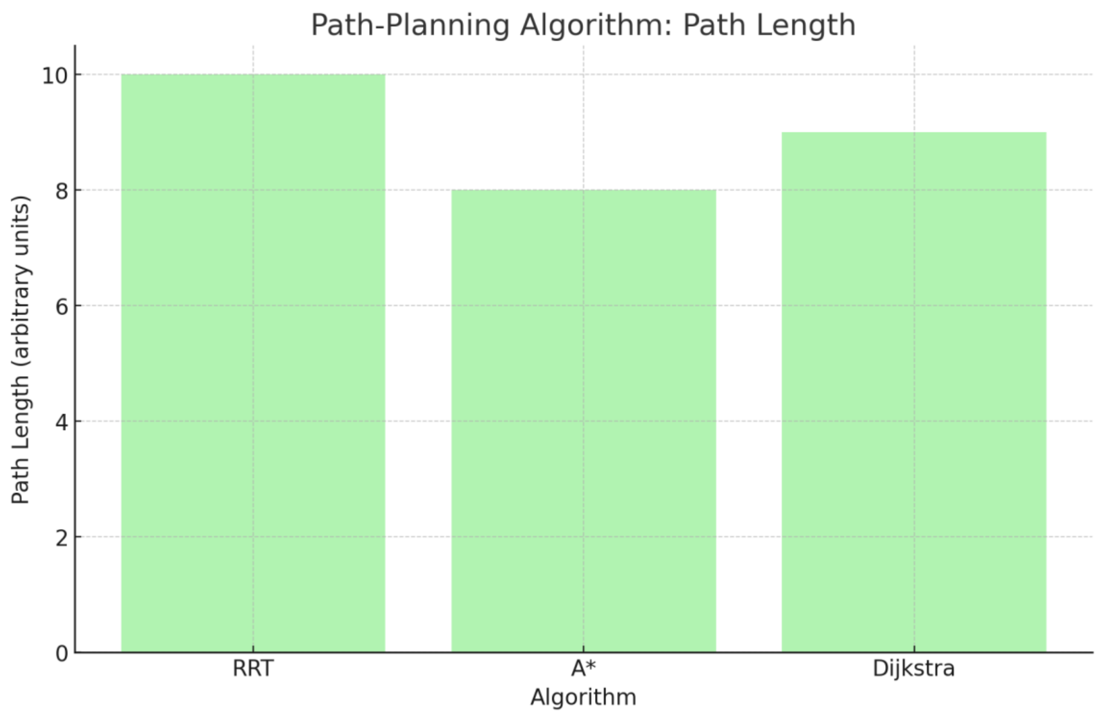
```
import matplotlib.pyplot as plt
# Sample data for Path-Planning Algorithm Performance
algorithms = ['RRT', 'A*', 'Dijkstra']
computation_time = [5, 3, 7]  # Time to compute path in seconds
path_length = [10, 8, 9]  # Path length in arbitrary units
obstacles_avoided = [15, 18, 12]  # Number of obstacles avoided

# Create a bar chart for computation time
plt.figure(figsize=(10, 6))
plt.bar(algorithms, computation_time, color='skyblue', alpha=0.7)
plt.title('Path-Planning Algorithm: Computation Time')
plt.xlabel('Algorithm')
plt.ylabel('Time to Compute Path (seconds)')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# Create a bar chart for path length
```

```
plt.figure(figsize=(10, 6))
plt.bar(algorithms, path_length, color='lightgreen', alpha=0.7)
plt.title('Path-Planning Algorithm: Path Length')
plt.xlabel('Algorithm')
plt.ylabel('Path Length (arbitrary units)')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

plt.figure(figsize=(10, 6))
plt.bar(algorithms, obstacles_avoided, color='salmon', alpha=0.7)
plt.title('Path-Planning Algorithm: Obstacles Avoided')
plt.xlabel('Algorithm')
plt.ylabel('Number of Obstacles Avoided')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

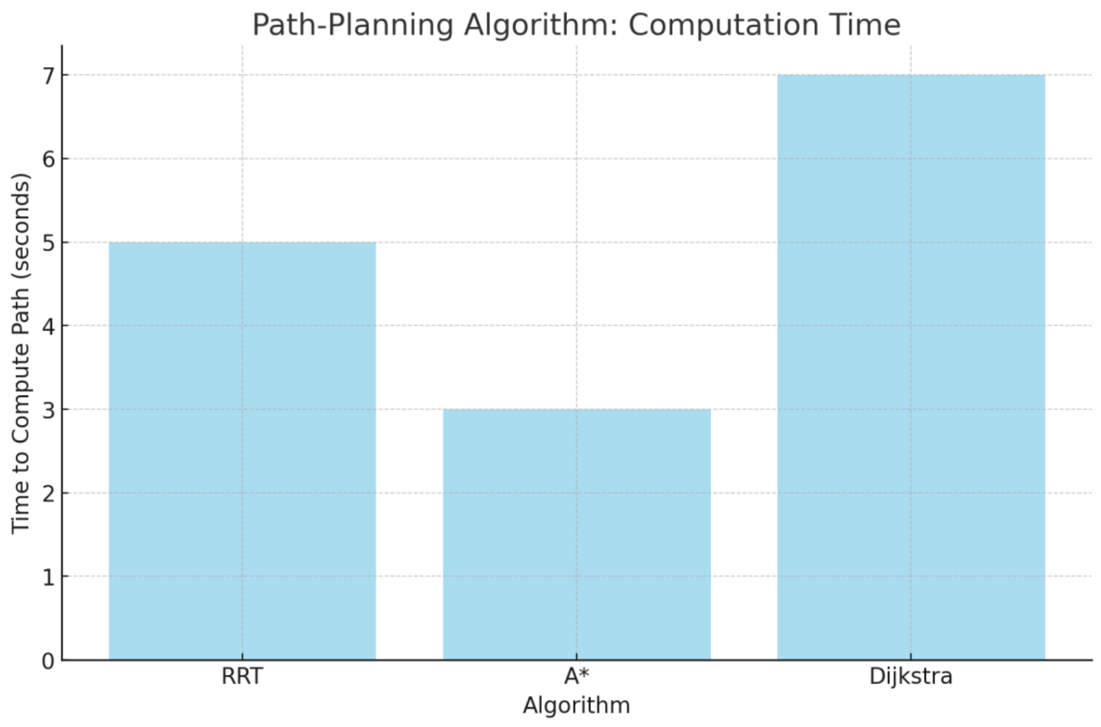
The other two path-planning algorithms which calculate the shortest possible movement path of the robot from its current location to the target location are RRT is extremely efficient when one is searching for the shortest path possible using frameworks such as MoveIt! and OMPL. All these algorithms consider environmental constraints in order to acquire smooth, collision-free trajectories.



Since the ROS Control infrastructure is implementing and managing the implementation of such control mechanisms as PID controllers, navigation will be responsive and adaptive: the robots will follow their predicted path while making adjustments for changing conditions.



Gazebo-like simulations of the navigation system are tested prior to real world deployment. These simulations include cases of obstacle-scenario and environment changes for a robot, whereby all possible concerns that may alter its performance in any way will be determined at this stage of refinement. This marks the critical refinement process for algorithms as well as system reliability.



The last stages are the actual testing of the robots in the real world. Robots are put to test in industrial warehouse environments or even under outdoor conditions. It can then be determined whether the robot can navigate the environments to efficiently respond to the dynamic and complex conditions of the real world.

In general, by combining SLAM along with advanced path-planning and control strategies using comprehensive simulation testing under the ROS environment, the autonomous robot navigation system was improved to better efficiency and reliability.

CONCLUSIONS

In conclusion, the inclusion of SLAM with next-generation path-planning algorithms, such as RRT and A*, inside the ROS framework increases the capabilities of autonomous robot navigation. SLAM allows real-time mapping and localization, which is vital for the navigation of dynamic environments. Path-planning algorithms, compared using performance metrics such as computation time, path length, and obstacles avoided, have different strengths in terms of their efficiency and adaptability. Control strategies, especially PID controllers, ensure smooth execution of planned trajectories. Environmental changes are adapted to within control. The extensive simulation within the environments of Gazebo tests these systems in advance before actual deployment to real-world situations where the robustness and reliability of the navigation system are tested. The entire approach thus guarantees that robots will navigate correctly within complex real-world settings while finding a path efficiency and obstacles' avoidance with maximum performance.

REFERENCES:

1. Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT Press.
2. Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research, 5*(1), 90-98.
3. LaValle, S. M. (2006). *Planning algorithms*. Cambridge University Press.
4. Quigley, M., Gerkey, B., & Smart, W. D. (2016). *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. O'Reilly Media.
5. Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., & Thrun, S. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT Press.
6. Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT Press.
7. Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research, 30*(7), 846-894.
8. Bohren, J., & Cousins, S. (2010). The SMACH high-level executive [ROS news]. *IEEE Robotics & Automation Magazine, 17*(4), 18-20.
9. Stentz, A. (1995). The focused D* algorithm for real-time replanning. *Proceedings of the International Joint Conference on Artificial Intelligence*.
10. Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine, 4*(1), 23-33.
11. Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots, 34*(3), 189-206.

12. Chen, C. H., Lee, D. J., & Lin, Y. T. (2016). Path planning with Bezier curve for autonomous robots. *IEEE Transactions on Automation Science and Engineering, 13*(4), 1115-1125.
13. Wurm, K. M., Stachniss, C., & Burgard, W. (2009). Coordinated multi-robot exploration using a segmentation of the environment. *IEEE/RSJ International Conference on Intelligent Robots and Systems*.
14. Zhang, Z., & Liu, Y. (2016). A review of SLAM techniques for autonomous vehicle navigation. *Journal of Robotics, 2016*, 1-10.
15. Brezak, M., & Petrović, I. (2014). Real-time mobile robot path planning using a dynamic visibility graph based on quadtrees. *Robotics and Autonomous Systems, 62*(10), 1407-1419.
16. Kallmann, M., & Mataric, M. J. (2004). Motion planning using dynamic roadmaps. *Proceedings of the IEEE International Conference on Robotics and Automation*.
17. Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics, 23*(1), 34-46.
18. Howard, A., & Kitchen, L. (2006). Generating feasible robot trajectories in arbitrarily constrained environments. *Autonomous Robots, 21*(1), 79-92.
19. Ziegler, J., Bender, P., Schreiber, M., & Lategahn, H. (2014). Making Bertha drive—An autonomous journey on a historic route. *IEEE Intelligent Transportation Systems Magazine, 6*(2), 8-20.
20. Huber, D. F., & Hebert, M. (2003). Fully automatic registration of multiple 3D data sets. *Image and Vision Computing, 21*(7), 637-650.