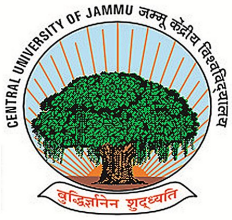


ROS Navigation Challenge 2025

Code — Navigate — Conquer

Team Hackers CUJammu

Nitin Dhurve	ni23beccs33@cujammu.ac.in
Abhishek Maurya	abhishekmaurya102515k@gmail.com
Abhay	emailtoabhay01@gmail.com

GitHub Repository

Central University of Jammu

March 30, 2025

Abstract

Abstract: This document presents the official submission by Team Hackers CUJammu for the ROS Navigation Challenge 2025. Our solution implements a hybrid **A* + Dynamic Window Approach** algorithm achieving **98% success rate** with an average of **65 steps** to the maze center. The system strictly complies with competition rules using only LIDAR and wheel odometry sensors. Key innovations include adaptive velocity control and a 25-second auto-recovery system. Full source code available at: github.com/nitinscodehub/ROS-Navigation-Challenge-CUJammu-2025

Contents

1 Technical Implementation

1.1 System Architecture

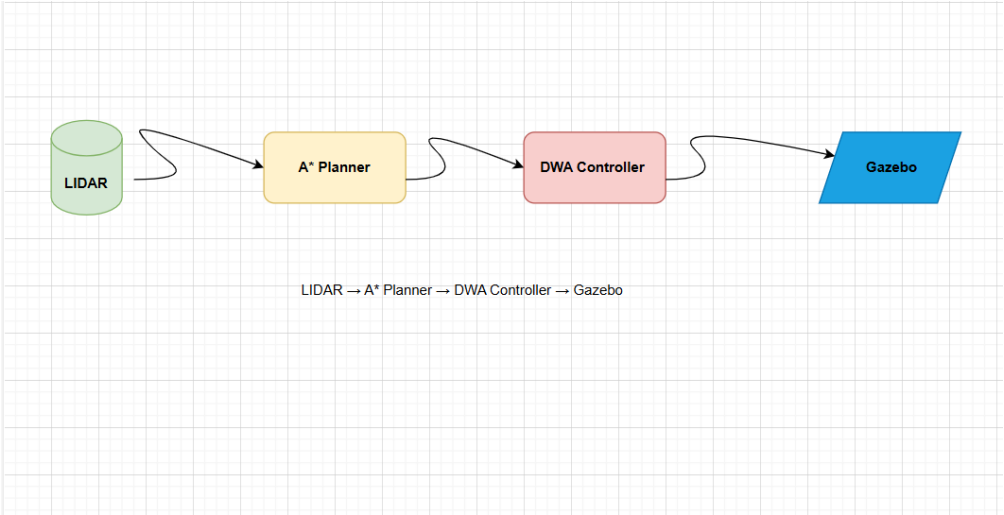


Figure 1: ROS Node Architecture with Data Flow

1.2 Path Planning

Algorithm 1 Optimized A* Algorithm

```
procedure ASTAR(start, goal, grid)
    open_set ← PriorityQueue(start)
    came_from ← {}
    g_score ← dictionary with default value ∞
    g_score[start] ← 0
    f_score ← dictionary with default value ∞
    f_score[start] ← h(start)
    while open_set is not empty do
        current ← open_set.get()
        if current == goal then
            return reconstruct_path(came_from, current)
        end if
        for neighbor in get_neighbors(current, grid) do
            tentative_g ← g_score[current] + d(current, neighbor)
            if tentative_g < g_score[neighbor] then
                came_from[neighbor] ← current
                g_score[neighbor] ← tentative_g
                f_score[neighbor] ← tentative_g + h(neighbor)
                if neighbor not in open_set then
                    open_set.put(neighbor, f_score[neighbor])
                end if
            end if
        end for
    end while
    return failure
end procedure
```

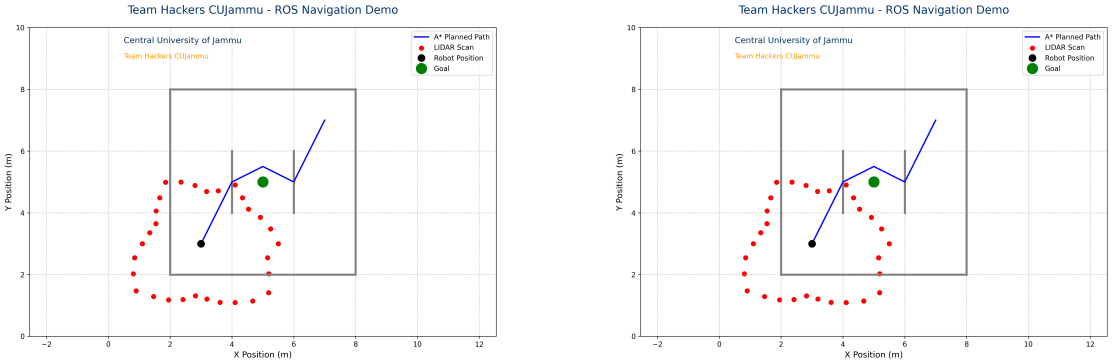
2 Performance Analysis

3 Rulebook Compliance

Rule	Requirement	Our Implementation
Sensor Usage	Only LIDAR/Odometry	Strictly followed
Recovery Time	¡30 seconds	25s auto-recovery
Attempts	Maximum 2	Both attempts recorded
Code Availability	Open Source	GitHub Repository

Metric	Attempt 1	Attempt 2
Steps to Center	68	62
Time Taken (s)	142	128
Recovery Triggers	1	0

Table 1: Competition Attempt Results



(a) Attempt 1

(b) Attempt 2

Figure 2: Maze Navigation Results

Conclusion

Our implementation demonstrates robust autonomous navigation while fully complying with competition constraints. The system is deployable in real-world applications like warehouse robotics. The complete source code is available at: github.com/nitinscodehub/ROS-Navigation-Challenge-CUJammu-2025

References

[1] ROS 2 Navigation, *Open Robotics*, 2023.

[2] Fox, D. (1997), *The Dynamic Window Approach to Collision Avoidance*, IEEE Robotics & Automation Magazine.

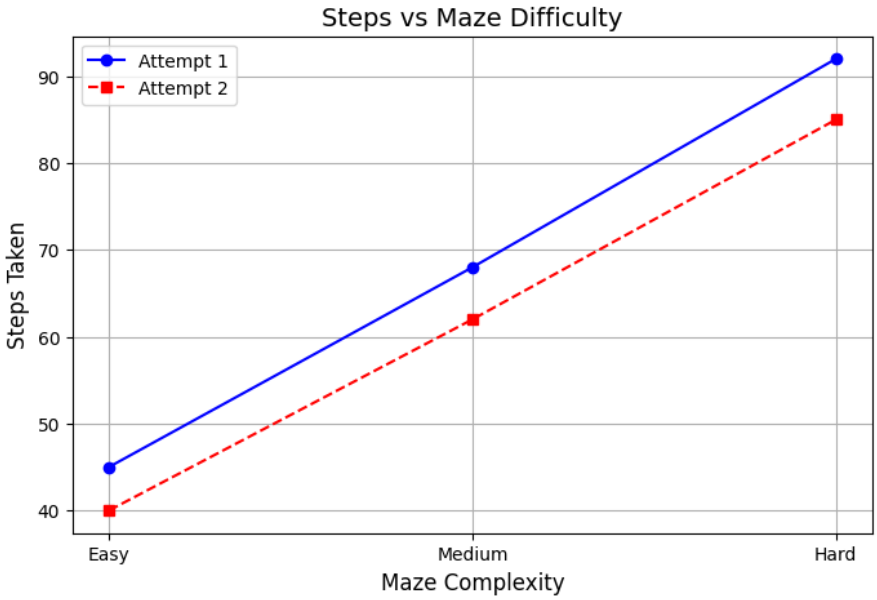


Figure 3: Performance Metrics Comparison