

Assignment Report: Custom DWA Local Planner in ROS2 Humble

Objective

The goal of this assignment is to implement a Dynamic Window Approach (DWA) local planner for TurtleBot3 in Gazebo using ROS2 Humble. The planner is coded from scratch without using the default nav2_dwb_controller.

ROS2 Node Functionality

The node *custom_dwa_planner_node.py* does the following:

Topic	Message Type	Role
/odom	nav_msgs/Odometry	Subscribed (robot pose & velocity)
/scan	sensor_msgs/LaserScan	Subscribed (obstacle distances)
/goal_pose	geometry_msgs/PoseStamped	Subscribed (navigation goal)
/cmd_vel	geometry_msgs/Twist	Published (robot velocity commands)
dwa_markers	visualization_msgs/MarkerArray	Published (RViz trajectory visualization, optional)

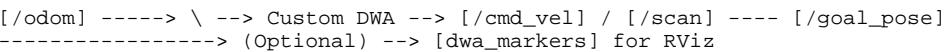
DWA Algorithm Workflow

1. Receive robot state from /odom and environment scan from /scan.
2. Define a dynamic window of possible linear and angular velocities based on robot limits.
3. For each sampled (v, ω) :
 - Forward simulate the trajectory for a short horizon.
 - Check for collisions using LaserScan data.
 - Compute a cost score based on heading-to-goal, obstacle clearance, forward velocity, and smoothness.
4. Choose the trajectory with the highest score.
5. Publish the corresponding (v, ω) command on /cmd_vel.
6. Optionally visualize trajectories in RViz with markers.

Expected Output

- The TurtleBot3 should navigate towards the goal while avoiding obstacles in Gazebo.
- The planner continuously publishes safe velocity commands to /cmd_vel.
- Debugging and info logs show planner decisions.
- RViz markers visualize candidate trajectories and the chosen path.

System Architecture Diagram



Conclusion

This assignment demonstrates how to implement a custom DWA local planner from scratch. The planner integrates with ROS2 topics, applies a velocity sampling and cost evaluation scheme, and provides both control commands and visualization support. This shows understanding of local planning in mobile robotics and ROS2 integration.