Risk-Weighted Global Planning in ROS2 Navigation2 Using Costmap-Based Penalty Accumulation

Preeti S. Powar

VIT Chennai

TABLE OF CONTENTS

1. Abstract
2. Introduction
3. System Architecture
    3.1 Overview of the Navigation System
    3.2 Software Components
    3.3 Navigation Pipeline
    3.4 Integration of the Custom Planner
4. Methodology
    4.1 Risk-Aware Planning Concept
    4.2 Candidate Path Generation
    4.3 Costmap Sampling
    4.4 Risk Metric Definition
    4.5 Path Scoring Function
    4.6 Path Selection
5. Implementation
    5.1 Planner Plugin Structure
    5.2 Costmap Integration
    5.3 Path Generation and Evaluation
    5.4 Planner Configuration
    5.6 Simulation Environment
6. Experimental Setup
    6.1 Simulation Environment
    6.2 Navigation Setup
7. Results and Observations
    7.1 Open Space Navigation
    7.2 Obstacle Blocking the Direct Path
    7.3 Navigation in a Dense Obstacle Region
    7.4 Directional Path Selection
    8. Discussion

1. Abstract

Mobile robot navigation requires path planning algorithms that enable robots to reach their goals while maintaining safe distances from obstacles. Many traditional global planners primarily optimize path length, which can produce trajectories that pass close to obstacles and increase the risk of collision in the presence of localization uncertainty or control errors. This project presents a risk-aware global planning approach implemented within the ROS2 Navigation2 (Nav2) framework. The proposed method evaluates multiple candidate trajectories between the start and goal positions and selects the path with the lowest accumulated risk. Risk is computed using inflation cost values from the navigation costmap, which represent proximity to obstacles. For each candidate trajectory, costmap values are sampled along the path and aggregated into a risk score, which is combined with a penalty term for path deviation to determine the final path selection. The planner was implemented as a custom Nav2 global planner plugin in C++ and integrated into the Nav2 navigation pipeline. The system was tested in simulation using the TurtleBot3 Burger robot in Gazebo and visualized in RViz. Experimental scenarios demonstrate that the planner prefers trajectories that maintain greater clearance from obstacles while still successfully reaching navigation goals. The results indicate that incorporating costmap-based risk evaluation can improve navigation safety compared to shortest-path planning strategies.

2. Introduction

Autonomous mobile robots rely on path planning algorithms to navigate safely through their environment while reaching specified goal locations. A core component of modern robot navigation systems is the global planner, which computes a collision-free path from the robot's current position to a target goal. In many robotic navigation frameworks, including the ROS2 Navigation2 (Nav2) stack, global planners typically optimize for the shortest path or lowest traversal cost. While these approaches are efficient, shortest paths often pass very close to obstacles, increasing the likelihood of collisions in real-world conditions where localization uncertainty, sensor noise, or control errors may occur.

To address this limitation, navigation strategies that explicitly consider safety or risk have been explored. In grid-based navigation systems, environmental risk can be approximated using costmaps, which represent the environment as a grid where each cell contains a cost value indicating the presence of obstacles or proximity to them. In particular, the inflation layer in the Nav2 costmap assigns higher costs to cells closer to obstacles, effectively encoding obstacle proximity information that can be used for safer path planning.

This project proposes a **risk-aware global planning approach** that utilizes costmap inflation values to guide navigation decisions. Instead of computing a single shortest path, the planner generates multiple candidate trajectories between the start and goal positions. Each candidate trajectory is evaluated by sampling costmap values along the path and accumulating a risk score that reflects the path's proximity to obstacles. The planner then selects the trajectory with the

lowest combined risk and deviation penalty, thereby prioritizing safer navigation routes even if they are slightly longer than the shortest path.

The proposed planner is implemented as a custom global planner plugin for the ROS2 Navigation2 framework, using C++ and the `nav2_core::GlobalPlanner` interface. The planner integrates directly with the Nav2 navigation pipeline and operates using the same costmap representation used by other Nav2 components. The system is evaluated in a simulated environment using the TurtleBot3 Burger robot in Gazebo, with visualization and goal control provided through RViz.

The main contributions of this project are:

1. The design and implementation of a **risk-aware global planner** integrated into the ROS2 Navigation2 stack.
2. A **costmap-based risk evaluation method** that accumulates obstacle proximity penalties along candidate trajectories.
3. A demonstration of **safer navigation behavior in simulated environments**, where the planner selects paths that maintain greater distance from obstacles compared to shortest-path approaches.

By incorporating risk evaluation into the global planning stage, this work demonstrates how navigation behavior can be improved to produce safer trajectories while maintaining compatibility with existing ROS2 navigation frameworks.

3. System Architecture

3.1 Overview of the Navigation System

The navigation system used in this project is based on the **ROS2 Navigation2 (Nav2) framework**, which provides a modular architecture for autonomous robot navigation. The system consists of several interconnected components responsible for environment representation, path planning, motion control, and visualization.The overall navigation pipeline begins when a navigation goal is provided by the user. The global planner computes a path from the robot's current position to the goal location. This path is then followed by a controller that generates velocity commands for the robot. Throughout this process, costmaps are used to represent the environment and ensure obstacle avoidance.The proposed **risk-aware planner** replaces the default global planner in the Nav2 stack and computes navigation paths based on accumulated risk derived from costmap inflation values.

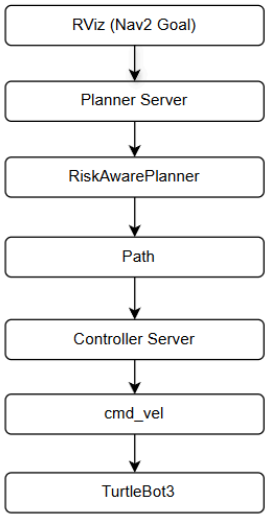


Fig. 3.1 System Architecture Diagram

3.2 Software Components

The project was implemented using several software tools and frameworks commonly used in robotics research and development.

Component	Role
ROS2 Humble	Robotics middleware for communication between system components
Navigation2 (Nav2)	Autonomous navigation framework
Gazebo	Physics-based simulation environment
RViz	Visualization and interaction interface
TurtleBot3 Burger	Differential drive robot model used in simulation
C++	Programming language used to implement the planner plugin

ROS2 enables communication between different modules through topics, services, and actions. The Navigation2 stack provides the infrastructure required for robot navigation, including global planning, local control, costmap management, and recovery behaviors.

3.3 Navigation Pipeline

The navigation process follows a structured pipeline within the Nav2 framework.

1. **Goal Input** – A navigation goal is specified through RViz using the Nav2 Goal tool.
2. **Global Planning** – The custom risk-aware planner computes a path from the start position to the goal.
3. **Path Execution** – The controller server generates velocity commands to follow the planned path.
4. **Costmap Updates** – The costmap continuously updates obstacle information from the environment.
5. **Robot Motion** – The robot executes velocity commands to reach the goal.

The interaction between these components ensures that the robot can navigate through the environment while avoiding obstacles.

3.4 Integration of the Custom Planner

The risk-aware planner was implemented as a plugin compatible with the `nav2_core::GlobalPlanner` interface. This allows the planner to be loaded dynamically by the Nav2 planner server.

The planner is configured through the Nav2 parameter file using the plugin configuration:

```
planner_server:

 ros__parameters:
```

```
  planner_plugins: ["GridBased"]

GridBased:

 plugin: "risk_aware_planner/RiskAwarePlanner"
```

Once loaded, the planner replaces the default global planner and generates navigation paths using the risk-aware evaluation algorithm.

4. Methodology

4.1 Risk-Aware Planning Concept

Traditional global planners in robot navigation typically compute paths that minimize travel distance or traversal cost. While these approaches are efficient, the resulting paths may pass very close to obstacles, which increases the likelihood of collisions in environments with localization errors, sensor noise, or imperfect control.

To improve navigation safety, this project introduces a **risk-aware planning strategy** that evaluates the safety of candidate trajectories based on their proximity to obstacles. Instead of generating a single shortest path, the planner produces multiple candidate trajectories between the start and goal positions. Each candidate trajectory is evaluated using costmap inflation values that represent the distance from nearby obstacles.

By accumulating the inflation costs along each candidate path, the planner estimates the overall risk associated with the trajectory. The path with the lowest combined risk and deviation penalty is selected as the final navigation path.

4.2 Candidate Path Generation

The planner begins by computing the straight-line direction between the start position and the goal location. Instead of following only this direct path, the planner generates several alternative trajectories using **lateral offsets** relative to the straight-line direction.

Each offset produces a slightly shifted trajectory that allows the robot to explore alternative routes around obstacles.

Example offsets used for candidate path generation include:

offsets = {-0.6 m, -0.3 m, 0.0 m, 0.3 m, 0.6 m}

These offsets generate a set of candidate paths distributed around the direct line connecting the start and goal positions.

Each candidate trajectory is discretized into a series of sample points along the path.

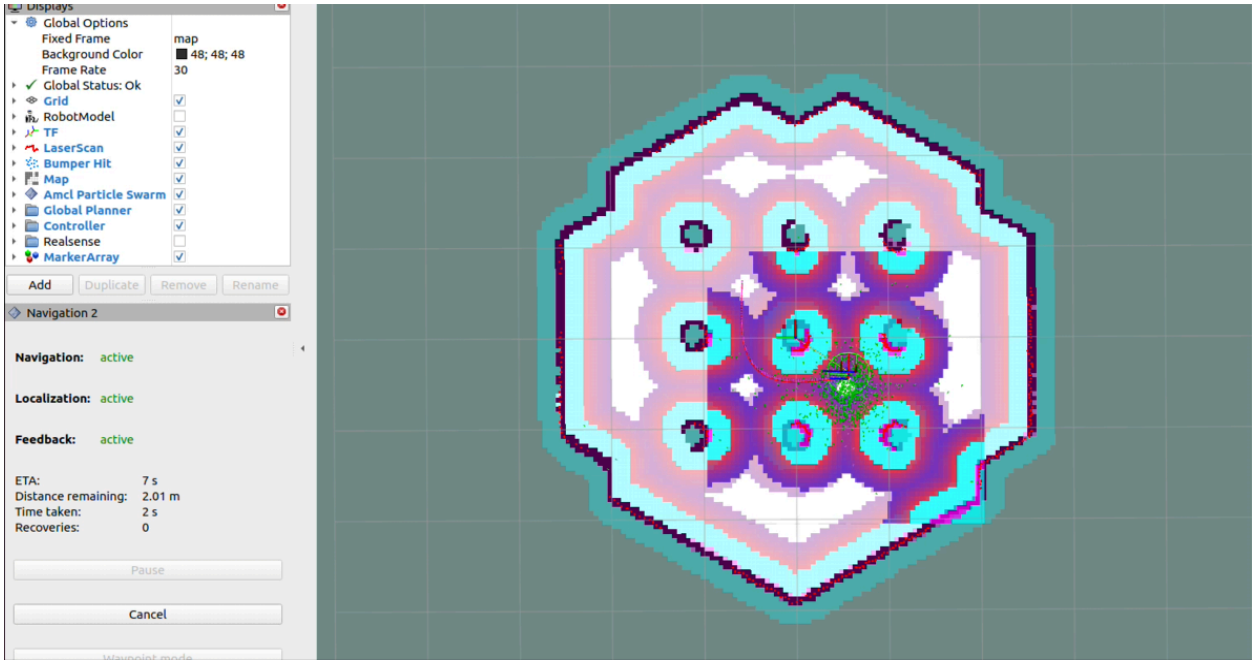


Fig 4.1 Example candidate trajectories generated using lateral offsets

4.3 Costmap Sampling

To evaluate the safety of each candidate trajectory, the planner samples the navigation costmap at each point along the path.

For each sampled point, the planner performs two operations:

1. Convert the world coordinates of the point to costmap grid indices.

   ```
   costmap_->worldToMap(wx, wy, mx, my)
   ```

2. Retrieve the cost value associated with that grid cell.

```
costmap_->getCost(mx, my)
```

The costmap assigns higher values to cells closer to obstacles through the inflation layer. As a result, paths that pass near obstacles accumulate larger cost values during evaluation.

4.4 Risk Metric Definition

The risk associated with a candidate trajectory is computed by accumulating the squared costmap values along the path.

$$Risk = \sum_{i=1}^{N} (cost_i)^2$$

Where:

- $cost_i$ is the costmap value at sample point iii
- N is the number of sampled points along the trajectory

Squaring the cost values increases the penalty for paths that approach obstacles closely, ensuring that trajectories near obstacles are strongly discouraged.

4.5 Path Scoring Function

In addition to accumulated risk, the planner also considers the deviation of each candidate path from the straight-line direction.

The final score for a candidate trajectory is computed as:

$$Score = Risk + \lambda|offset|$$

Where:

- **Risk** represents the accumulated costmap risk along the trajectory
- **offset** represents the lateral displacement from the straight path
- **λ (lambda)** is a weighting factor that penalizes large deviations

This scoring function ensures that the planner balances **safety and path efficiency**, preferring safer trajectories while avoiding unnecessarily large detours.

4.6 Path Selection

After evaluating all candidate trajectories, the planner compares their final scores and selects the trajectory with the lowest score.

The selected path is then converted into a ROS navigation path message (`nav_msgs::msg::Path`) and returned to the Navigation2 planner server. The controller server subsequently follows this path to move the robot toward the goal location.

4.7 Algorithm Workflow

The overall planning procedure executed within the `createPlan()` function can be summarized as follows:

1. Receive start and goal positions.

2. Compute the straight-line direction between the two points.
3. Generate multiple candidate trajectories using lateral offsets.
4. Sample points along each candidate path.
5. Convert sample points to costmap coordinates.
6. Retrieve costmap inflation values.
7. Accumulate risk values along each trajectory.
8. Compute the final score for each candidate path.
9. Select the trajectory with the lowest score.
10. Return the selected path to the Nav2 navigation system.

5. Implementation

5.1 Planner Plugin Structure

The proposed risk-aware planner was implemented as a **custom global planner plugin** compatible with the ROS2 Navigation2 (Nav2) framework. Nav2 allows developers to extend navigation functionality by implementing plugins that follow predefined interfaces.

The planner implements the `nav2_core::GlobalPlanner` interface, which defines the functions required for integration with the Nav2 planner server.

The key functions implemented in the plugin include:

Function	Description
configure()	Initializes the planner and loads required parameters
activate()	Activates the planner when navigation starts
deactivate()	Deactivates the planner when navigation stops
cleanup()	Releases allocated resources
createPlan()	Computes a navigation path between start and goal

Among these functions, the `createPlan()` method contains the main path planning logic, including candidate trajectory generation and risk evaluation.

5.2 Costmap Integration

The planner uses the Nav2 costmap to evaluate obstacle proximity during path planning. The costmap provides a grid-based representation of the environment, where each cell contains a cost value representing obstacle information.

The planner accesses the costmap through the costmap interface provided by Nav2. For each sampled point along a candidate trajectory, the planner converts the world coordinates into costmap grid coordinates and retrieves the corresponding cell cost.

The conversion from world coordinates to costmap indices is performed using:

costmap_->worldToMap(wx, wy, mx, my)

After the grid indices are obtained, the cost value associated with the cell is retrieved using:

costmap_->getCost(mx, my)

These cost values represent the inflation costs generated by the obstacle inflation layer, which assigns higher costs to cells closer to obstacles.

5.3 Path Generation and Evaluation

During the execution of the `createPlan()` function, the planner generates multiple candidate trajectories between the start and goal positions. Each trajectory is sampled at regular intervals to evaluate its risk.

For every sampled point along the trajectory, the planner retrieves the corresponding costmap value and accumulates the squared cost to compute the total risk of the trajectory.

The planner also tracks the maximum cost encountered along the path, which can be used for debugging and visualization.

After evaluating all candidate trajectories, the planner computes the final score for each path and selects the trajectory with the lowest score.

5.4 Planner Configuration

The planner is loaded into the Nav2 navigation pipeline through the planner server configuration file. The plugin is specified in the Nav2 parameter file used when launching the navigation stack.

An example configuration is shown below:

```
planner_server:
 ros__parameters:
   planner_plugins: ["GridBased"]

GridBased:
 plugin: "risk_aware_planner/RiskAwarePlanner"
```

This configuration instructs the planner server to load the custom risk-aware planner instead of the default global planner.

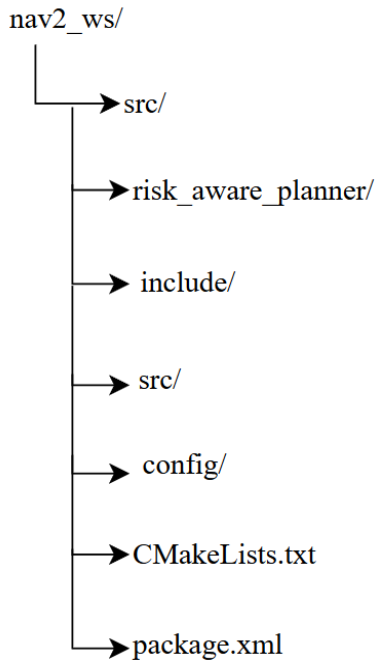
5.5 Workspace Structure

The planner was developed within a ROS2 workspace named `nav2_ws`. The workspace follows the standard ROS2 package structure.

The main package created for this project is:

risk_aware_planner

The workspace directory structure is organized as follows:



The source files inside the `src` directory contain the implementation of the planner plugin, while the `config` directory contains the Nav2 parameter configuration file used to load the planner.

The workspace was built using the ROS2 build tool `colcon`.

5.6 Simulation Environment

The planner was tested in a simulated environment using the **TurtleBot3 Burger** robot model in the Gazebo simulator. The simulation environment allows the robot to navigate within a predefined world containing obstacles.

The Navigation2 stack was launched together with the custom planner, and RViz was used for visualization and interaction. Goals were provided through the **Nav2 Goal** tool in RViz, allowing the robot to navigate autonomously while the planner generated risk-aware paths.

6. Experimental Setup

6.1 Simulation Environment

All experiments were conducted in a simulated environment using the **Gazebo simulator** with the **TurtleBot3 Burger** robot model. Gazebo provides a physics-based environment that allows testing navigation algorithms without requiring physical hardware.

The navigation system was built using the **ROS2 Humble** distribution and the **Navigation2 (Nav2) framework**, which provides a modular architecture for autonomous robot navigation.

Visualization and interaction with the navigation system were performed using **RViz**, which allows users to observe costmaps, planned paths, and robot motion.

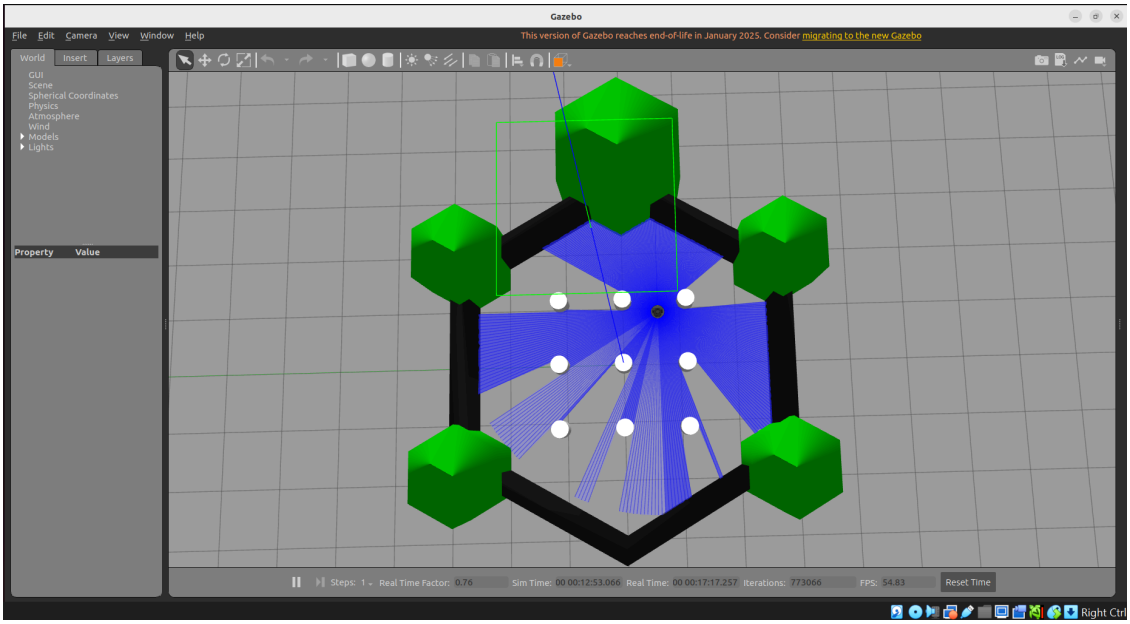


Fig 6.1 Gazebo simulator with TurtleBot3 Burger robot model

The system configuration used in the experiments is summarized in Table 1.

Component	**Specification**
Operating System	Ubuntu 22.04
ROS Distribution	ROS2 Humble
Navigation Framework	Navigation2 (Nav2)
Robot Model	TurtleBot3 Burger

Simulator	Gazebo
Visualization Tool	RViz
Programming Language	C++

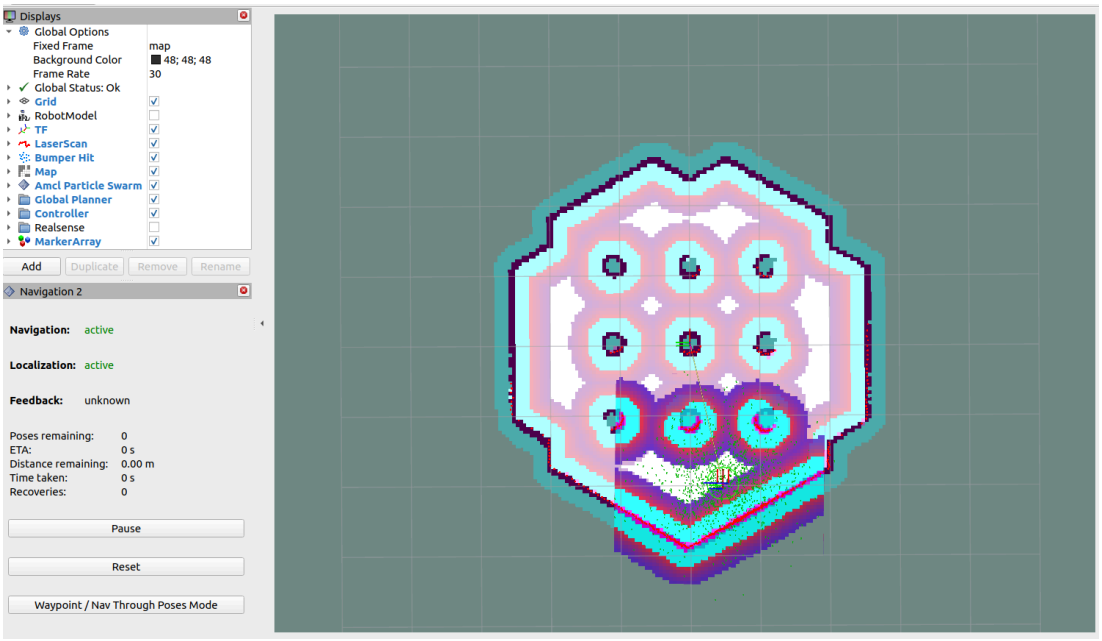


Fig. 6.2 RViz visualization of the robot and costmap.

6.2 Navigation Setup

The navigation system was launched using the TurtleBot3 Gazebo world and the Nav2 navigation stack. The custom planner plugin was integrated into the Nav2 planner server through the navigation parameter configuration file.

The navigation stack was started using the following command:

```
ros2 launch turtlebot3_navigation2 navigation2.launch.py \

use_sim_time:=True \

params_file:=~/nav2_ws/src/risk_aware_planner/config/nav2_params
.yaml
```

Once the navigation system was running, the robot pose was initialized in RViz using the **2D Pose Estimate** tool. Navigation goals were then provided using the **Nav2 Goal** tool.

6.3 Navigation Procedure

Each experiment followed a consistent procedure to evaluate the behavior of the risk-aware planner.

The experimental procedure consisted of the following steps:

1. Launch the Gazebo simulation environment.
2. Start the Navigation2 stack with the custom planner plugin.
3. Initialize the robot pose using the 2D Pose Estimate tool in RViz.
4. Specify a goal location using the Nav2 Goal tool.
5. Observe the generated global path and robot navigation behavior.
6. Record the selected path and robot motion for analysis.

This procedure allowed evaluation of the planner under different environmental configurations and obstacle layouts.

6.4 Evaluation Scenarios

To evaluate the behavior of the risk-aware planner, several navigation scenarios were designed within the simulation environment. Each scenario tested the planner's ability to select safe navigation paths under different environmental conditions.

The experimental scenarios are summarized in Table 2.

Scenario	Description	Purpose
Open Space Navigation	Robot navigates in an obstacle-free area	Verify basic path planning behavior
Obstacle Blocking Path	Obstacle placed between robot and goal	Evaluate obstacle avoidance
Dense Obstacle Region	Multiple obstacles forming narrow passages	Test path safety in complex environments
Directional Path Choice	Two alternative routes available	Evaluate planner decision-making

These scenarios were designed to observe how the planner balances path efficiency and safety when selecting navigation trajectories.

7. Results and Observations

7.1 Open Space Navigation

In the first experiment, the robot and the goal location were placed in an environment without nearby obstacles. Since the costmap contained mostly free cells with low cost values, the accumulated risk for all candidate trajectories was nearly identical.As a result, the planner selected the trajectory with the smallest deviation from the straight-line direction. The robot followed a direct path toward the goal, which corresponds to the shortest path.This experiment confirms that the risk-aware planner behaves similarly to traditional shortest-path planners when the environment contains no obstacles.

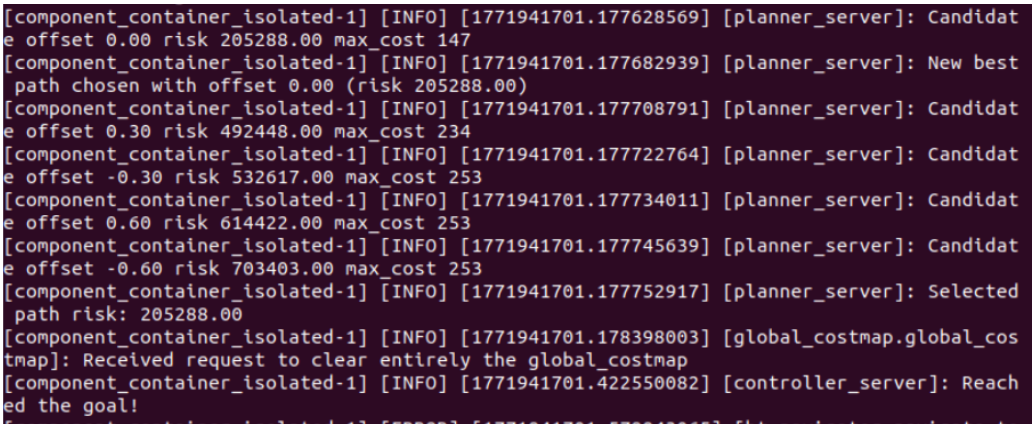


Fig. 7.1 Scenario 1 – Open Space Navigation

7.2 Obstacle Blocking the Direct Path

In the second scenario, an obstacle was positioned between the robot and the goal location. The direct path between the start and goal passed through a region with high inflation cost values near the obstacle.

When evaluating candidate trajectories, the planner detected higher accumulated risk for paths that passed close to the obstacle. As a result, the planner selected an alternative trajectory that deviated slightly from the direct path while maintaining greater clearance from the obstacle.

Although the selected path was slightly longer than the shortest path, the robot maintained a safer distance from the obstacle during navigation.

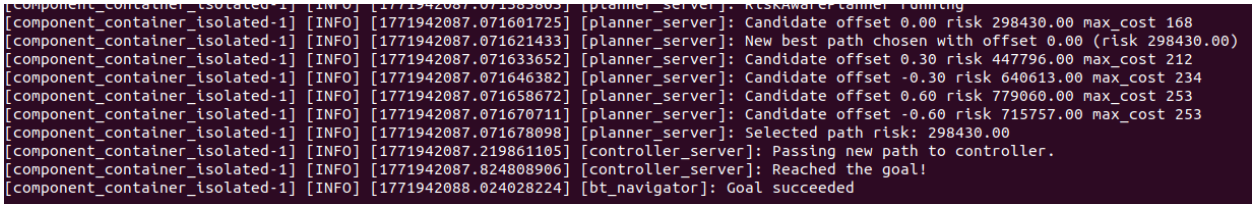


Fig.7.2 Scenario 2 – Obstacle Blocking Path

7.3 Navigation in a Dense Obstacle Region

The third scenario involved navigating through an environment containing multiple obstacles arranged to create narrow corridors. In this case, several candidate trajectories intersected regions with elevated costmap values due to obstacle proximity.

The planner evaluated the accumulated risk along each trajectory and selected the path that passed through the region with the lowest overall costmap risk. This resulted in the robot navigating through a corridor that provided greater clearance from surrounding obstacles.

This experiment demonstrates that the planner is capable of identifying safer routes even in complex environments with multiple obstacles.

```
[component_container_isolated-1] [INFO] [1771942973.065719624] [planner_server]: Candidate offset 0.00 risk 0.00 max_cost 0
[component_container_isolated-1] [INFO] [1771942973.065734149] [planner_server]: New best path chosen with offset 0.00 (risk 0.00)
[component_container_isolated-1] [INFO] [1771942973.065747039] [planner_server]: Candidate offset 0.30 risk 189033.00 max_cost 181
[component_container_isolated-1] [INFO] [1771942973.065759800] [planner_server]: Candidate offset -0.30 risk 0.00 max_cost 0
[component_container_isolated-1] [INFO] [1771942973.065770415] [planner_server]: Candidate offset 0.60 risk 479812.00 max_cost 253
[component_container_isolated-1] [INFO] [1771942973.065781261] [planner_server]: Candidate offset -0.60 risk 183404.00 max_cost 220
[component_container_isolated-1] [INFO] [1771942973.065788509] [planner_server]: Selected path risk: 0.00
[component_container_isolated-1] [INFO] [1771942973.215363513] [controller_server]: Passing new path to controller.
[component_container_isolated-1] [INFO] [1771942973.225753907] [controller_server]: Reached the goal!
[component_container_isolated-1] [INFO] [1771942973.456789392] [bt_navigator]: Goal succeeded
```

Fig. 7.3 Scenario 3 – Dense Obstacles

7.4 Directional Path Selection

In the final scenario, the robot was placed in a position where two possible routes were available to reach the goal location. One route passed closer to obstacles, while the other route provided greater clearance.

The planner evaluated candidate trajectories corresponding to both directions. The path that passed closer to obstacles accumulated higher inflation costs, resulting in a higher risk score. Consequently, the planner selected the alternative route with lower accumulated risk.

This behavior illustrates the planner's ability to evaluate multiple navigation options and select the safer trajectory.

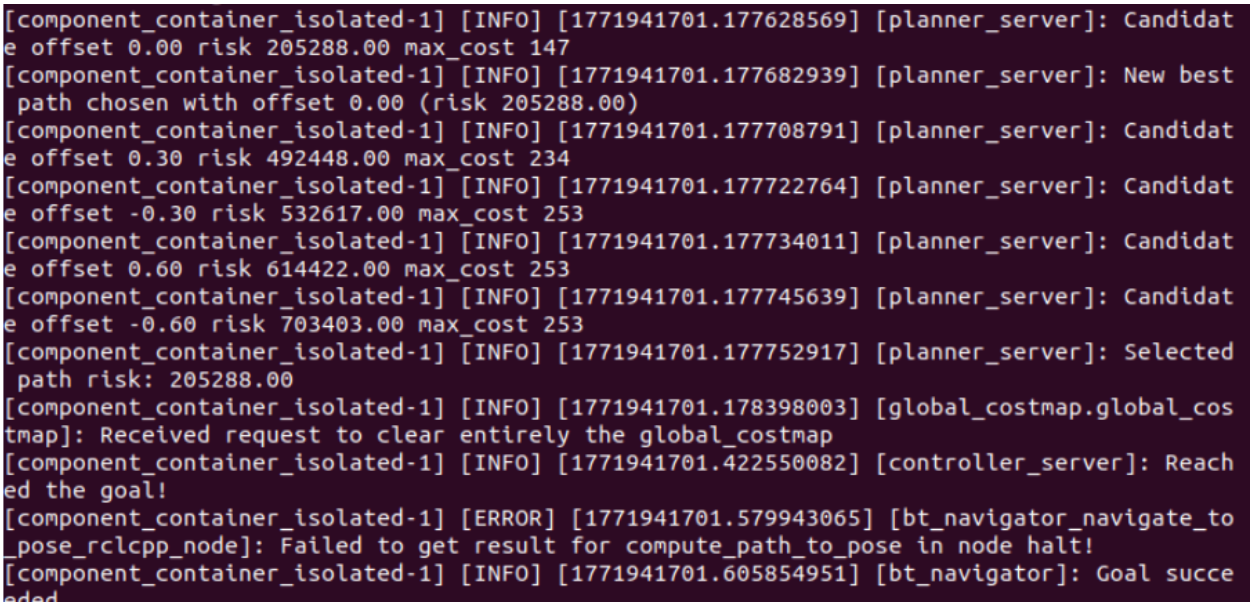


Fig. 7.4 Scenario 4 – Directional Path Decision

8. Discussion

The experimental results demonstrate that incorporating a risk-based evaluation into the global planning stage can improve the safety of navigation paths generated by the robot. By evaluating multiple candidate trajectories and accumulating costmap inflation values along each path, the proposed planner is able to prefer trajectories that maintain greater clearance from obstacles rather than strictly minimizing path length.

In scenarios where the environment contained no obstacles, the planner selected a trajectory close to the straight-line path between the start and goal locations. This behavior indicates that the risk-aware planner does not introduce unnecessary deviations in open environments, and its

behavior remains consistent with traditional shortest-path planners when obstacle proximity is not a concern.

When obstacles were placed along the direct path to the goal, the planner successfully identified alternative trajectories with lower accumulated risk. Paths that passed close to obstacles accumulated higher costmap inflation values, which increased the calculated risk score. As a result, the planner selected trajectories that deviated slightly from the shortest path but maintained safer distances from obstacles. This behavior highlights the planner's ability to prioritize safety over minimal travel distance.

In environments containing multiple obstacles, the planner evaluated several candidate paths and selected trajectories that navigated through regions with lower overall costmap risk. This demonstrates that the planner can identify safer corridors within cluttered environments and avoid areas where obstacle proximity would increase navigation risk.

The directional path selection experiment further illustrates the planner's ability to evaluate multiple navigation options. When two possible routes were available, the planner compared the accumulated risk associated with each path and selected the route that provided greater obstacle clearance. This capability enables the planner to make informed decisions in environments with multiple possible navigation routes.

Although the risk-aware planner improves safety by avoiding areas with high inflation costs, this approach may produce paths that are slightly longer than the shortest possible route. This represents a trade-off between path efficiency and navigation safety. However, in many

real-world robotic applications, maintaining a safe distance from obstacles is preferable to minimizing path length, particularly in environments where uncertainty or sensor noise may affect robot localization.

Overall, the results suggest that costmap-based risk evaluation is an effective method for improving navigation safety while remaining compatible with existing ROS2 Navigation2 infrastructure. The proposed approach provides a practical enhancement to traditional global planning strategies by incorporating obstacle proximity into the path selection process.

9. Limitations

Although the proposed risk-aware planner improves navigation safety by avoiding high-cost regions in the costmap, several limitations remain in the current implementation.

First, the planner evaluates risk based solely on static costmap inflation values. This means that the planner assumes a static environment and does not explicitly account for dynamic obstacles such as moving people or other robots. In real-world environments, dynamic obstacle prediction may be required to maintain safe navigation.

Second, the candidate trajectory generation approach uses a fixed set of lateral offsets. While this method is computationally efficient and simple to implement, it limits the range of possible paths that the planner can evaluate. More advanced planning algorithms could explore a larger set of candidate trajectories to improve optimality.

Another limitation is that the planner evaluates safety using costmap values but does not explicitly incorporate robot dynamics or motion constraints. The controller server is responsible for executing the path, but the planner itself does not consider factors such as turning radius or dynamic feasibility.

Finally, the evaluation in this project was conducted entirely in simulation. While simulation provides a controlled environment for testing navigation algorithms, additional testing on real robotic platforms would be required to validate the planner's performance under real-world conditions.

10. Conclusion

This project presented a risk-aware global planning approach implemented within the ROS2 Navigation2 framework. The proposed planner evaluates multiple candidate trajectories between the start and goal positions and selects the path with the lowest accumulated risk based on costmap inflation values.

The planner was implemented as a custom Nav2 plugin using the `nav2_core::GlobalPlanner` interface and integrated into the Navigation2 planning pipeline. The algorithm generates candidate trajectories using lateral offsets, samples costmap values along each path, and computes a risk-based score that balances obstacle proximity and path deviation.

Experimental scenarios conducted in the TurtleBot3 simulation environment demonstrated that the planner consistently selects safer navigation paths that maintain greater clearance from obstacles. In open environments, the planner behaves similarly to traditional shortest-path planners, while in obstacle-rich environments it prioritizes safer trajectories.

The results indicate that incorporating costmap-based risk evaluation into the global planning stage can improve navigation safety while remaining compatible with existing ROS2 navigation frameworks.

11. Future Work

Several improvements could be explored in future work to extend the capabilities of the proposed planner.

One possible extension is the incorporation of dynamic obstacle awareness. Integrating sensor data or prediction models for moving obstacles would allow the planner to adjust trajectories in real time and improve safety in dynamic environments.

Another potential improvement is the development of more advanced candidate trajectory generation methods. Instead of using fixed lateral offsets, sampling-based or optimization-based planning techniques could generate a larger set of possible paths and improve path optimality.

Future work could also explore adaptive weighting of the risk function. Dynamically adjusting the balance between path length and safety based on environmental conditions could improve navigation performance in different scenarios.

Finally, the planner could be tested on a real mobile robot platform to evaluate its performance under real-world sensing and control conditions.