Robotics & Internet-of-Things

ROS

Udemy Course
Robot Operating System (Part II)
Navigation and SLAM

Prof. Anis Koubaa

Navigation Stack Tuning

https://www.udemy.com/user/anis-koubaa/

Robotics & Internet-of-Things

ROS

UDEMY COURSE
ROBOT OPERATING SYSTEM (PART II) NAVIGATION AND SLAM

PROF. ANIS KOUBAA

Setting Max/Min Velocity and Acceleration

https://www.udemy.com/user/anis-koubaa/

Where to Update Velocity and Acceleration Info

- The parameters of the velocity and acceleration are provided in the yaml configuration file of the local planner
- For Turtlebot3, the parameters configuration files are located in this folder /opt/ros/{ros_version}/share/turtlebot3_navigation/param
- Open any file related to local planner and you identify the velocity and acceleration parameters for the Turtlebot3 robot. The question is how to determine/estimate these values?

How to Obtain the Maximum Velocity

- Use a Joystick
- **For Translation Velocity:** move the robot in a straight line until speed becomes constant, then echo the odom topic and record the maximum speed value
- **For Rotational Velocity:** move the robot 360 degrees until it reaches a constant speed, then echo the odom topic and record the maximum speed value

How to Obtain the Maximum Acceleration

- Try to get it from motors manual (if available)
- We can use the timestamp in the odometry message to estimate the acceleration from the velocity
- accel_translation = $V_{max}/t_{V_{max}}$
- accel_rotation = $W_{max}/t_{W_{max}}$

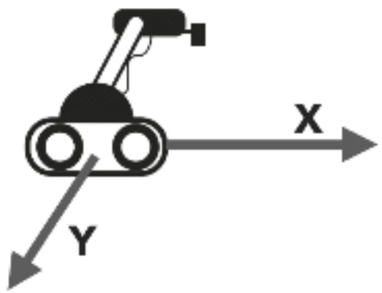
```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
  float64[36] covariance
geometry_msgs/TwistWithCovariance twist
  geometry_msgs/Twist twist
    geometry_msgs/Vector3 linear
      float64 x
      float64 y
      float64 z
    geometry_msgs/Vector3 angular
      float64 x
      float64 y
      float64 z
```

How to Obtain the Minimum Values?

- Set the minimum translational velocity to a negative value
  - Robots will backoff when it gets stuck
- Set the minimum rotation velocity to a negative value
  - Robot can rotate in both directions
- These parameters are used by the local planner when the robot gets stuck.

Velocities in X and Y Directions

- The velocity in X-direction should be the same as the translational velocity
- The velocity in Y-direction should be zero for non holonomic robots



Robotics &
Internet-of-Things

ROS

Udemy Course
Robot Operating System (Part II)
Navigation and SLAM

Prof. Anis Koubaa

Global Path Planner Tuning

https://www.udemy.com/user/anis-koubaa/

Built-In Global Path Planner in ROS

- The **global path planner** is responsible for finding a global obstacle-free path from initial location to the goal location using the environment map
- Global path planner must adhere to the **nav_core::BaseGlobalPlanner** interface.

BUILT-IN GLOBAL PATH PLANNER IN ROS

Public Member Functions | Protected Member Functions | List of all members

nav_core::BaseGlobalPlanner Class Reference abstract

Provides an interface for global planners used in navigation. All global planners written as plugins for the navigation stack must adhere to this interface. More...

```
#include <base_global_planner.h>
```

Public Member Functions

virtual void	**initialize** (std::string name, costmap_2d::Costmap2DROS *costmap_ros)=0 Initialization function for the **BaseGlobalPlanner**. More...
virtual bool	**makePlan** (const geometry_msgs::PoseStamped &start, const geometry_msgs::PoseStamped &goal, std::vector< geometry_msgs::PoseStamped > &plan)=0 Given a goal pose in the world, compute a plan. More...
virtual bool	**makePlan** (const geometry_msgs::PoseStamped &start, const geometry_msgs::PoseStamped &goal, std::vector< geometry_msgs::PoseStamped > &plan, double &cost) Given a goal pose in the world, compute a plan. More...
virtual	**~BaseGlobalPlanner** () Virtual destructor for the interface. More...

http://www.ros.org/doc/api/nav_core/html/classnav__core_1_1BaseGlobalPlanner.html

BUILT-IN GLOBAL PATH PLANNER IN ROS


ROS.org

About | Support | Discussion Forum | Service Status | Q&A answers.ros.org

Search: Submit

Documentation Browse Software News Download

navigation/ Tutorials/ Writing A Global Path Planner As Plugin in ROS

Note: This tutorial was developed for ROS Hydro version. However, it is expected to also work with Groovy (not tested). For any suggestions/comments about this tutorial, please send an email to Anis Koubaa (akoubaa@coins-lab.org). More information about Global Planner for ROS can be found in the iroboapp project page. A mirror of this tutorial is also available in this link.. This tutorial assumes that you have completed the previous tutorials: ROS tutorials, Turtlebot.

Please ask about problems and questions regarding this tutorial on answers.ros.org. Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Writing A Global Path Planner As Plugin in ROS

Description: In this tutorial, I will present the steps for writing and using a global path planner in ROS. The first thing to know is that to add a new global path planner to ROS, the new path planner must adhere to the `nav_core::BaseGlobalPlanner` C++ interface defined in nav_core package. Once the global path planner is written, it must be added as a plugin to ROS so that it can be used by the move_base package. In this tutorial, I will provide all the steps starting from writing the path planner class until deploying it as a plugin. I will use Turtlebot as an example of robot to deploy the new path planner. For a tutorial that shows how to integrate a real GA planner as ROS plugin, refer to Adding Genetic Algorithm Global Path Planner As Plugin in ROS. A video tutorial is also available in this link

Keywords: Global Planner, Navigation Stack, Turtlebot, nav_core, Costmap

Tutorial Level: INTERMEDIATE

Contents

Wiki: Distributions, ROS/Installation, ROS/Tutorials, RecentChanges, Robots/TurtleBot, Writing A G...ugin in ROS, navigation/Tutorials

Page: Edit (Text), Edit (GUI), Info, Unsubscribe, Remove Link, Attachments, More Actions:

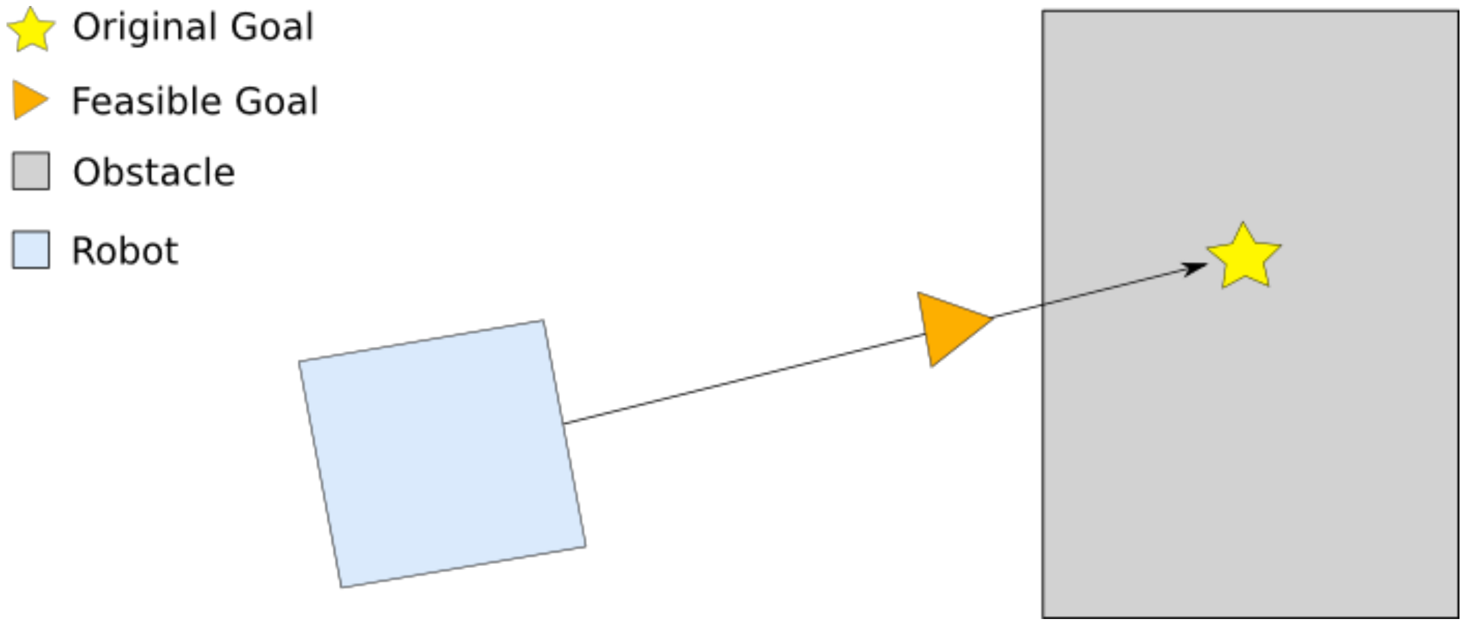
User: AnisKoubaa, Settings, Logout

http://wiki.ros.org/navigation/Tutorials/Writing%20A%20Global%20Path%20Planner%20As%20Plugin%20in%20ROS

Built-In Global Path Planner in ROS

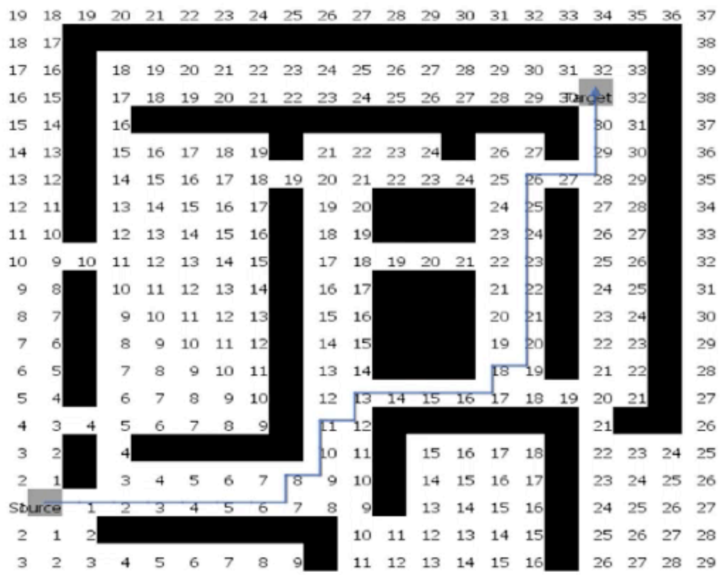
- There are three built-in global path planners in ROS:
  - **carrot_planner:** simple global planner that takes a user-specified goal point and attempts to move the robot as close to it as possible, even when that goal point is in an obstacle.
  - **navfn:** uses the Dijkstra's algorithm to find the global path between any two locations.
  - **global_planner:** is a replacement of navfn and is more flexible and has more options.

Carrot Planner



NAVFN AND GLOBAL_PLANNER

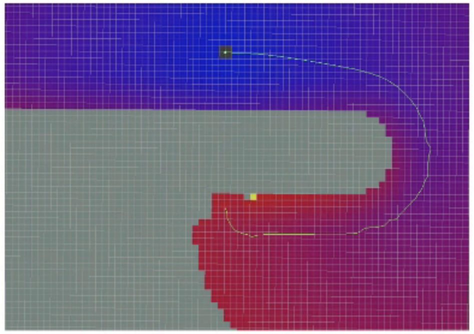
- navfn uses Dijkstra's algorithm to find a global path
- global_planner is a flexible replacement of navfn
  - Support of A*
  - Can use a grid path



GLOBAL_PLANNER: EXAMPLE OF DIFFERENT PARAMETERS

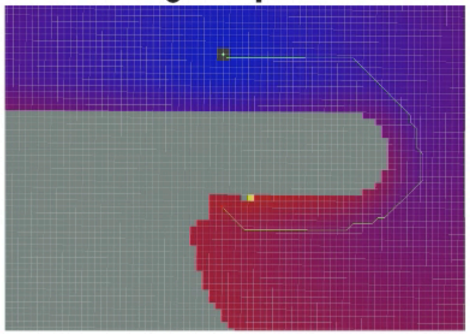
http://wiki.ros.org/global_planner

Standard Behavior



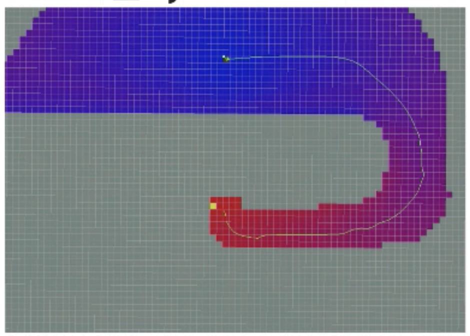
Grid Path

use_grid_path=True



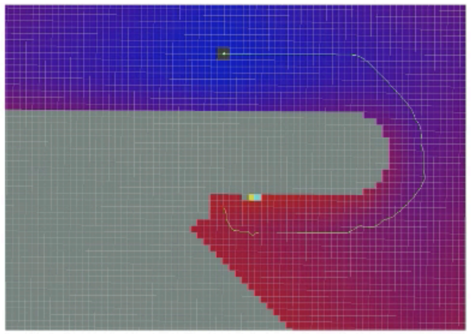
A* Path

use_dijkstra=False



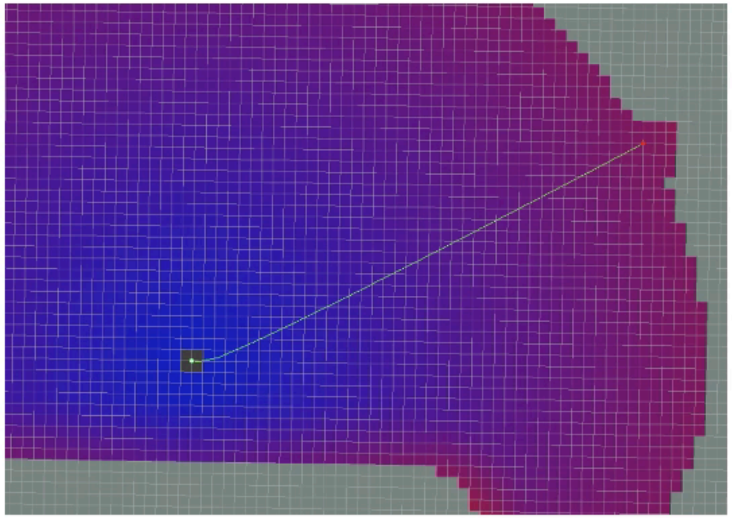
Simple Potential Calculation

use_quadratic=False

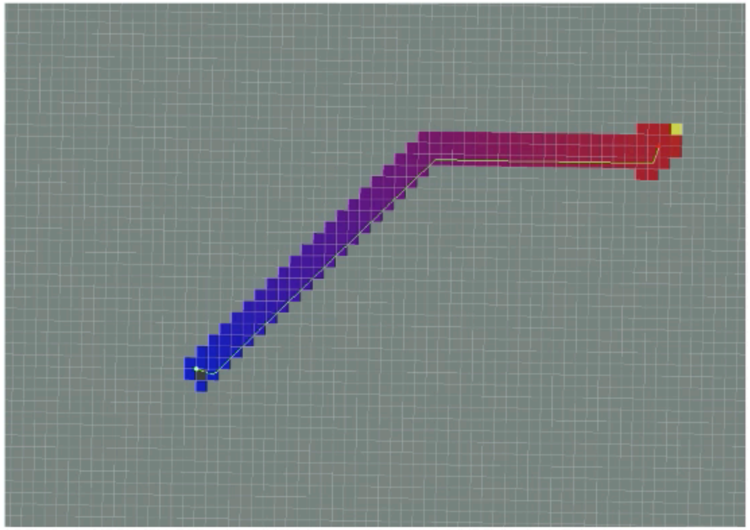


GLOBAL_PLANNER: EXAMPLE OF DIFFERENT PARAMETERS

Dijkstra algorithm



A*



Robotics &
Internet-of-Things

ROS

Udemy Course
Robot Operating System
(Part II)
Navigation and SLAM

Prof. Anis Koubaa

Local Path Planner Tuning

https://www.udemy.com/user/anis-koubaa/

Built-In Local Path Planner in ROS

- The **local path planner** is responsible execution the static path determined by the global path planner while avoiding dynamic obstacle that might come into the path using the robot's sensors.
- Global path planner must adhere to the **nav_core::BaseLocalPlanner** interface.

Built-In Local Path Planner in ROS

Public Member Functions | Protected Member Functions

nav_core::BaseLocalPlanner Class Reference

Provides an interface for local planners used in navigation. All local planners written as plugins for the navigation stack must adhere to this interface.
More...

```
#include <base_local_planner.h>
```

List of all members.

Public Member Functions

virtual bool	**computeVelocityCommands** (geometry_msgs::Twist &cmd_vel)=0 Given the current position, orientation, and velocity of the robot, compute velocity commands to send to the base.
virtual void	**initialize** (std::string name, tf::TransformListener *tf, costmap_2d::Costmap2DROS *costmap_ros)=0 Constructs the local planner.
virtual bool	**isGoalReached** ()=0 Check if the goal pose has been achieved by the local planner.
virtual bool	**setPlan** (const std::vector< geometry_msgs::PoseStamped > &plan)=0 Set the plan that the local planner is following.
virtual	**~BaseLocalPlanner** () Virtual destructor for the interface.

Protected Member Functions

	BaseLocalPlanner ()

Robotics &
Internet-of-Things

ROS

Udemy Course
Robot Operating System (Part II)
Navigation and SLAM

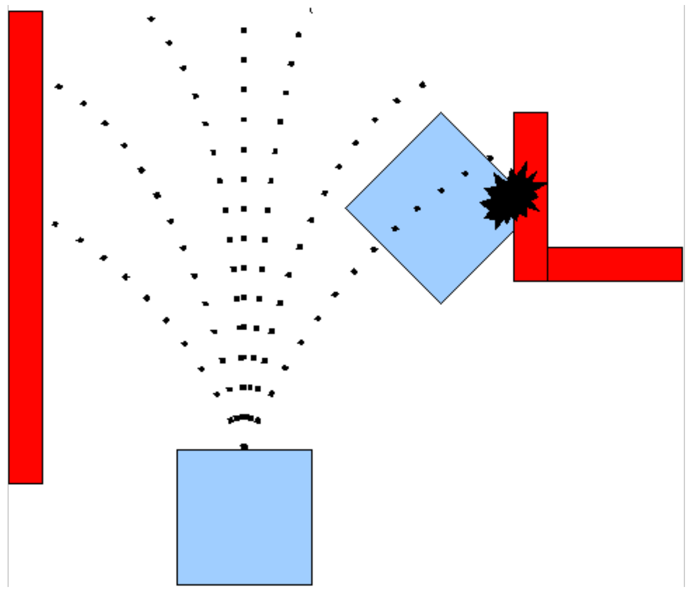
Prof. Anis Koubaa

Local Path Planning
Dynamic Window Approach

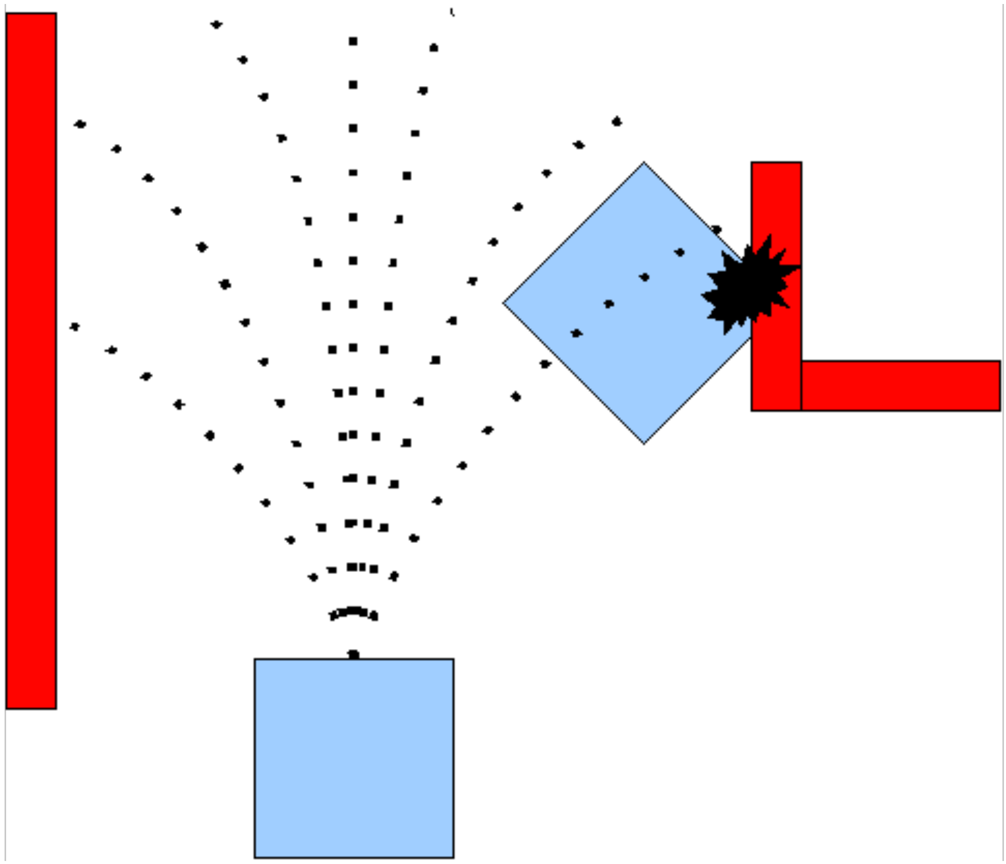
https://www.udemy.com/user/anis-koubaa/

DWA ALGORITHM

- Discretely sample in the robot's control space (dx,dy,dtheta)
- For each sampled velocity, perform forward simulation from the robot's current state to predict what would happen if the sampled velocity were applied for some (short) period of time.
- Evaluate (score) each trajectory resulting from the forward simulation, using a metric that incorporates characteristics such as: proximity to obstacles, proximity to the goal, proximity to the global path, and speed. Discard illegal trajectories (those that collide with obstacles).
- Pick the highest-scoring trajectory and send the associated velocity to the mobile base.
- Rinse and repeat.



DYNAMIC WINDOW APPROACH



Robotics & Internet-of-Things

ROS

Udemy Course
Robot Operating System (Part II)
Navigation and SLAM

Prof. Anis Koubaa

Tuning the Simulation Parameters of DWA

https://www.udemy.com/user/anis-koubaa/

ROS COURSE PART II: NAVIGATION

DWA ALGORITHM

- This DWA planner depends on the local costmap which provides obstacle information
- It is important to well tune the parameters

DWA Parameters

- **simulation time:** time allowed for the robot to move with the sampled velocities
- takes the velocity samples in robot's control space, and examine the circular trajectories represented by those velocity samples, and finally eliminate bad velocities
- High simulation times (>=5) lead to heavier computation, but get longer paths
- Low simulation times (<=2)
  - limited performance, especially when the robot needs to pass a narrow doorway, or gap between furnitures, because there is insufficient time to obtain the optimal trajectory that actually goes through the narrow passway

DWA Parameters

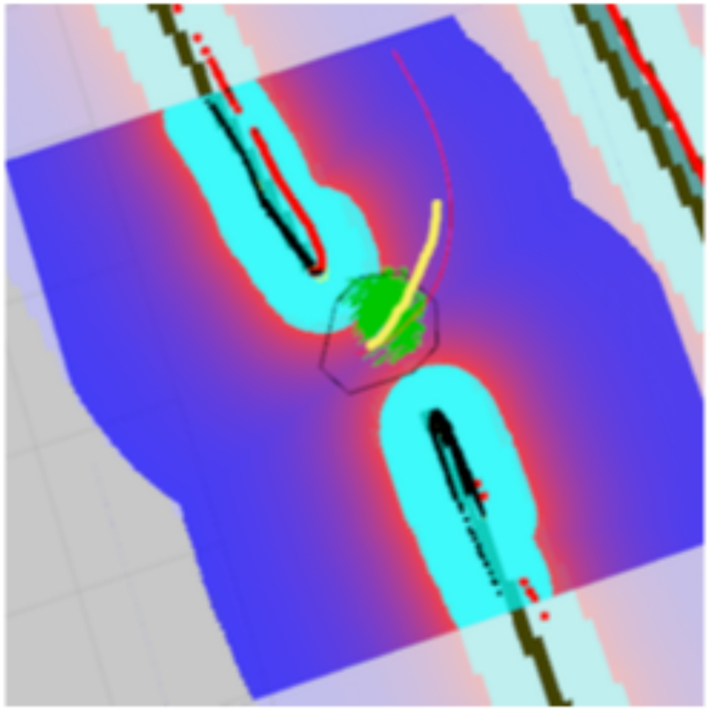


Figure 11: sim_time = 1.5

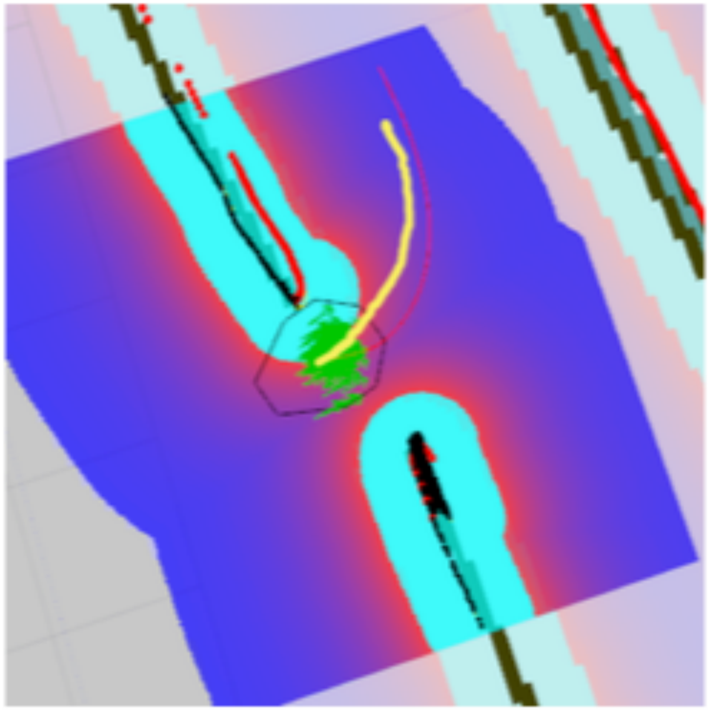


Figure 12: sim_time = 4.0

Robotics & Internet-of-Things

ROS

Udemy Course
Robot Operating System (Part II)
Navigation and SLAM

Prof. Anis Koubaa

Trajectory Scoring

https://www.udemy.com/user/anis-koubaa/

TRAJECTORY SCORING

$$
\begin{aligned}
\mathit{cost} &= \texttt{path\_distance\_bias} * (\text{distance}(m) \text{ to path from the endpoint of the trajectory}) \\
&+ \texttt{goal\_distance\_bias} * (\text{distance}(m) \text{ to local goal from the endpoint of the trajectory}) \\
&+ \texttt{occdist\_scale} * (\text{maximum obstacle cost along the trajectory in obstacle cost (0-254)})
\end{aligned}
$$

DWA Parameters

- Velocities Samples
  - vx sample, vy sample determine how many translational velocity samples to take in x, y direction
- vth sample controls the number of rotational velocities samples
- vth samples to be higher than translational velocity samples, because turning is generally a more complicated condition than moving straight ahead
- vx_sample = 20 and vth_sample = 40

DWA Parameters

- Simulation granularity
- sim granularity is the step size to take between points on a trajectory.
- It basically means how frequent should the points on this trajectory be examined (test if they intersect with any obstacle or not).
- A lower value means higher frequency, which requires more computation power. The default value of 0.025 is generally enough for turtlebot-sized mobile base.