Harsh Agrawal

Harshag37@gmail.com

23 September 2021

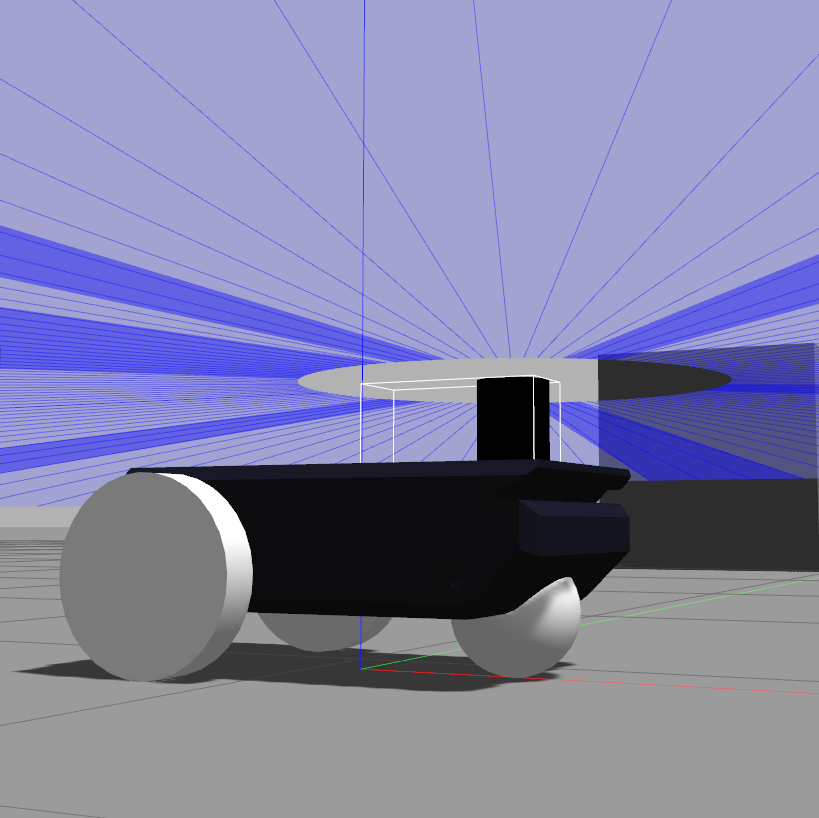
Path Planning

**Plan a path from point A to point B in a workspace consisting of obstacles.**

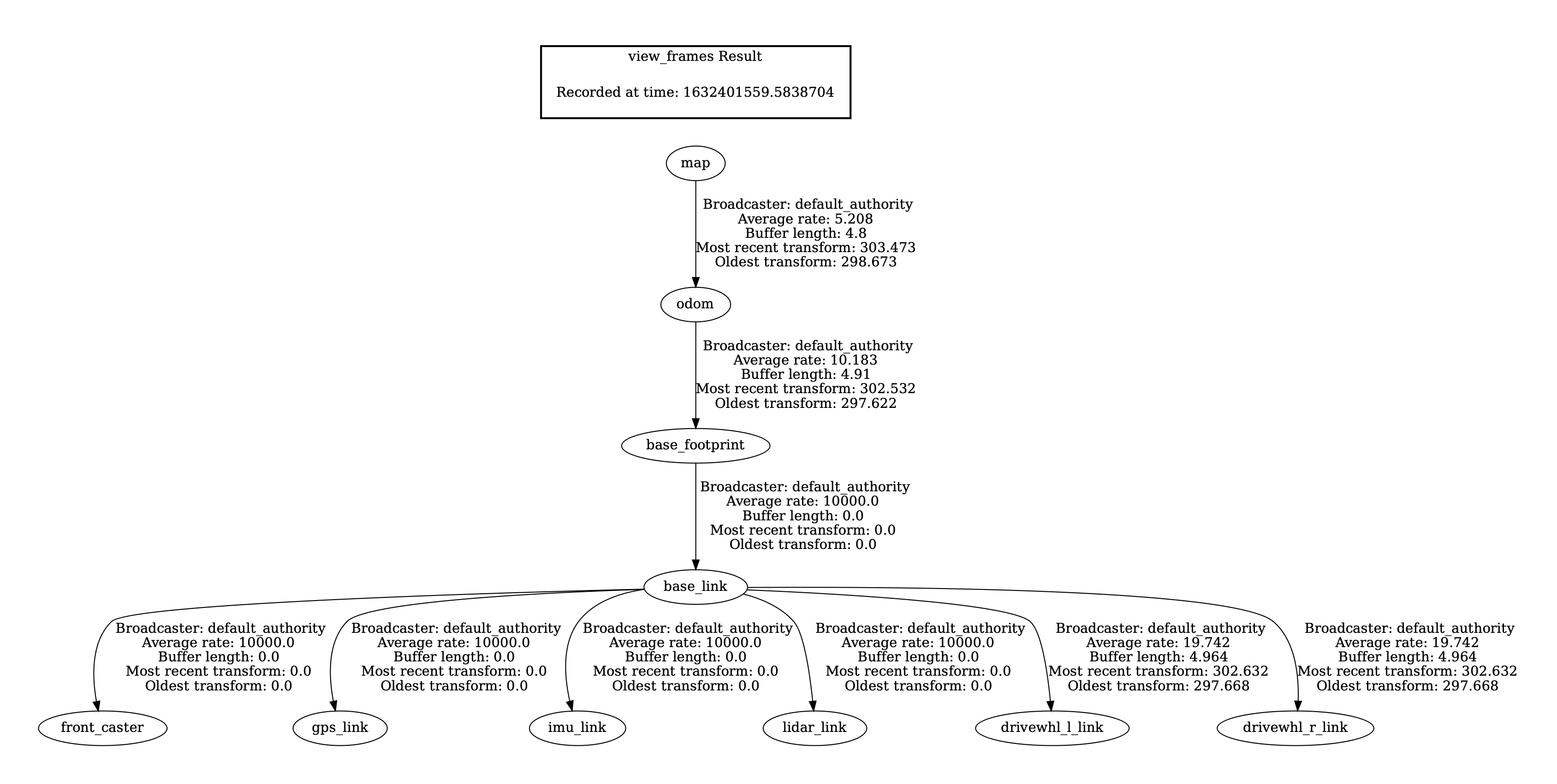
* + You can use any existing code but need to explain everything that is happening with the algorithm
  + You can use GPS/SLAM for localisation
  + You can use IR/LIDAR/Vision for collision avoidance
  + You can use environment but the path between point A to point B should have at least one obstacles. More the better
  + You can use Turtlebot or any other robot as your vehicle for the simulation with or without gazebo. However, please use some form of visualisation.

**Software used**

* + ROS2 ( FOXY )
  + GAZEBO
  + RVIZ
  + UBUNTU 20.0.4 (OS)

**Model**

**Type of drive:** Differential Drive

**Coordinate TransformImage**

**Sensors Integrated**

* Odometery sensor
  + Rotary Encoders: Two encoders are mounted perpendicular to each other such that they are always in contact with the ground to obtain the distance travelled

along the X and Y axis of the robot to localize itself.

* + - IMU: IMU Sensor is used to localise the robot and find its orientation with respect to the reference axis.
* Lidar

Lidar, which stands for **Light Detection and Ranging**, is a remote sensing method that uses light in the form of a pulsed laser to measure ranges (variable distances) to the Earth.

**Localisation**

**Extended Kalman Filter**

We have configured the **robot\_localization** package to use an [Extended Kalman Filter](https://drive.google.com/file/d/1EFq2nOVUQM-h1T22hoi5af-KNosuN41C/view) (ekf\_node) to fuse the data from sensor inputs. These sensor inputs come from the IMU Gazebo plugin and the differential drive Gazebo plugin that are defined in our SDF file.

**Mapping**

We have used [nav2\_map\_server](https://github.com/ros-planning/navigation2/tree/main/nav2_map_server) of the [navigation2](https://github.com/ros-planning/navigation2) package. Map Server provides maps to the rest of the Nav2 system using both topic and service interfaces.

**SLAM**

We have used [slam\_toolbox](https://github.com/SteveMacenski/slam_toolbox) package for for creating dynamic 2D costmap

This includes:

* Ordinary point-and-shoot 2D SLAM mobile robotics folks expect (start, map, save pgm file) with some nice built in utilities like saving maps
* Continuing to refine, remap, or continue mapping a saved (serialised) pose-graph at any time
* An optimization-based localization mode built on the pose-graph. Optionally run localization mode without a prior map for "lidar odometry" mode with local loop closures
* Synchronous and asynchronous modes of mapping

**Path Planning**

We have used [nav2\_bt\_navigator](https://github.com/ros-planning/navigation2/tree/main/nav2_bt_navigator) for planning and following a generated path. The BT Navigator (Behaviour Tree Navigator) module implements the NavigateTo Pose task interface. It is a Behaviour Tree-based implementation of navigation that is intended to allow for flexibility in the navigation task and provide a way to easily specify complex robot behaviours.Image