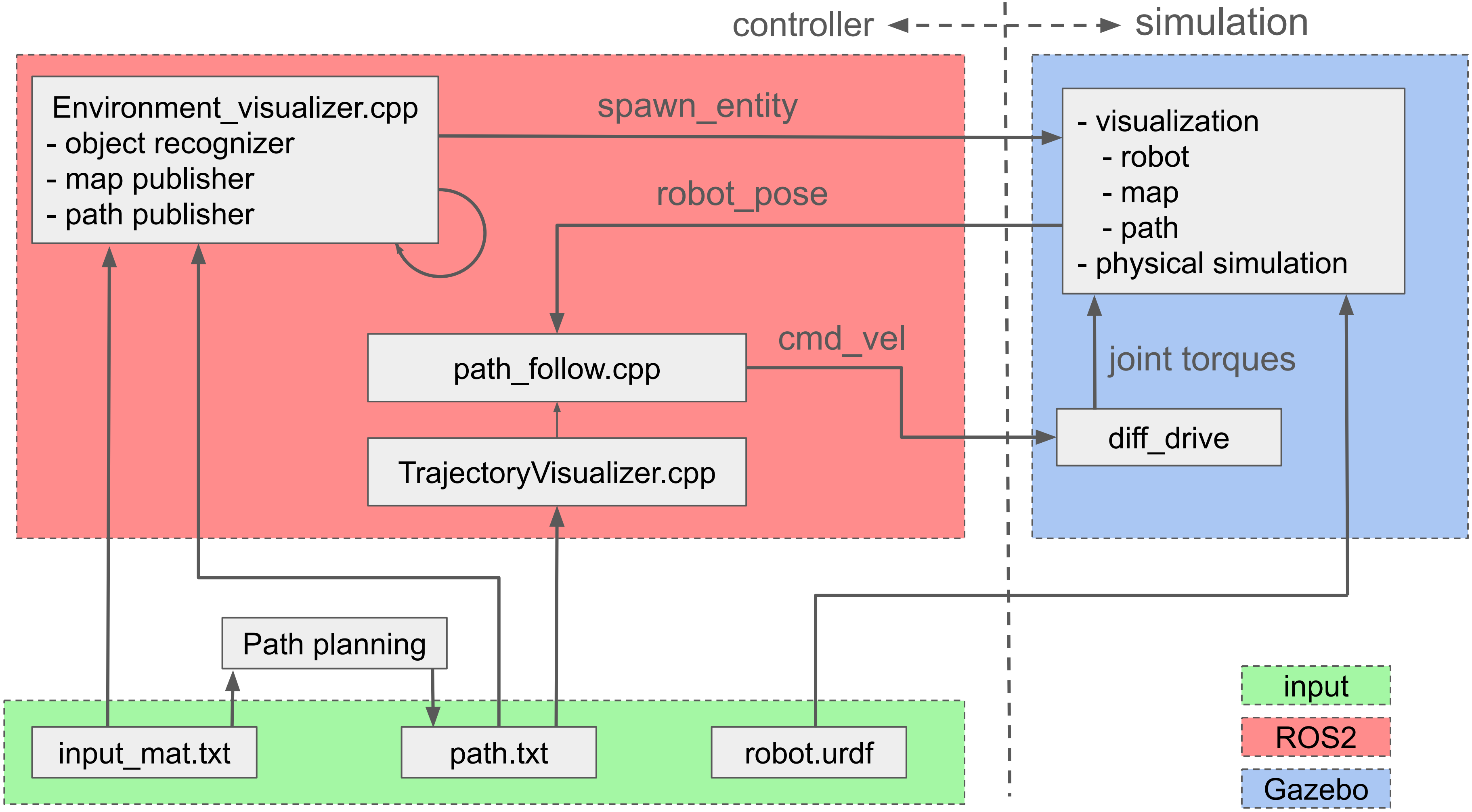
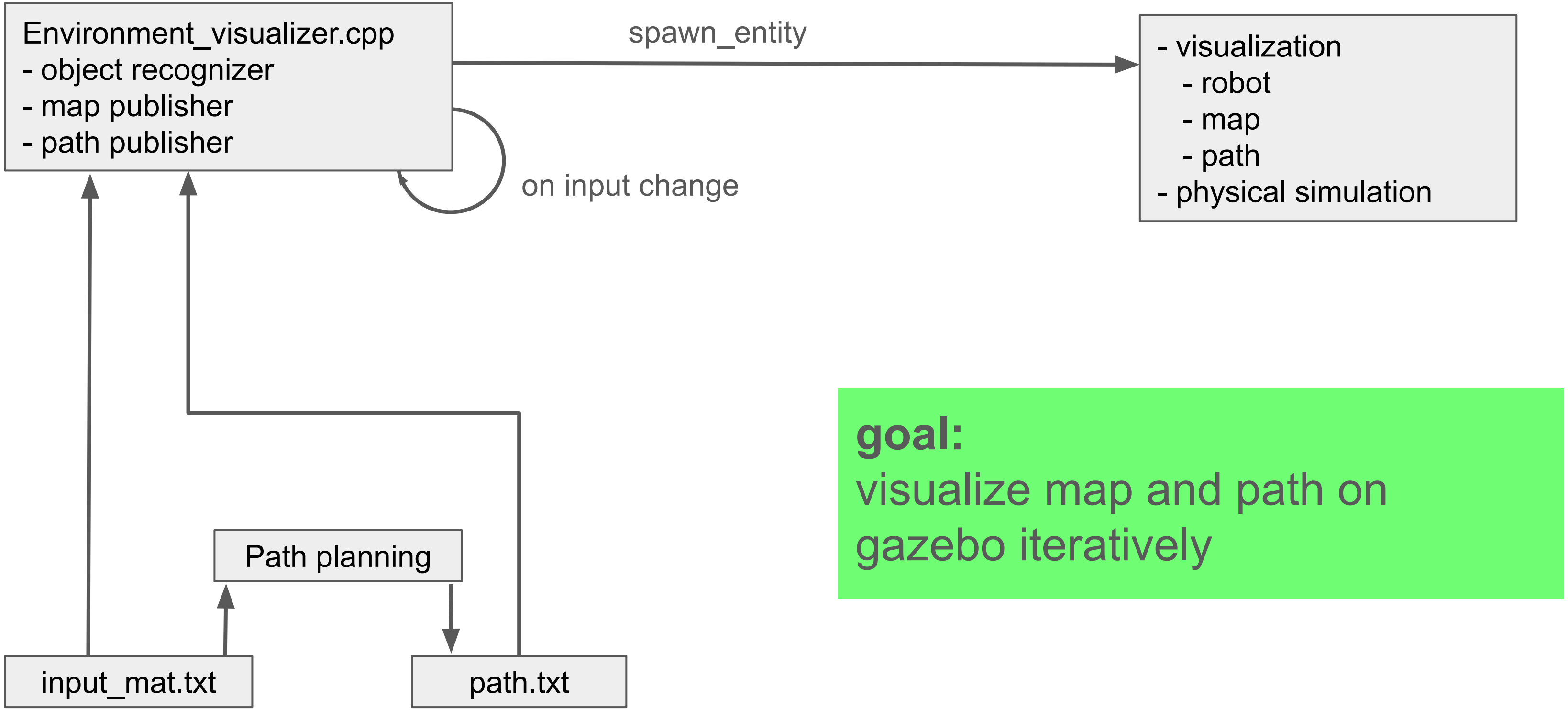
Deambulator Control and Visualization with Ros2 and Gazebo

Robotic Perception and Action Project

Daniele Turrini - 249485
Paolo Golinelli - 247450


controller
simulation
Environment_visualizer.cpp
- object recognizer
- map publisher
- path publisher
spawn_entity
robot_pose
- visualization
- robot
- map
- path
- physical simulation
path_follow.cpp
cmd_vel
joint torques
diff_drive
TrajectoryVisualizer.cpp
Path planning
input_mat.txt
path.txt
robot.urdf
input
ROS2
Gazebo

Map and Path Visualization



Environment_visualizer.cpp
- object recognizer
- map publisher
- path publisher

spawn_entity

- visualization
  - robot
  - map
  - path
- physical simulation

on input change

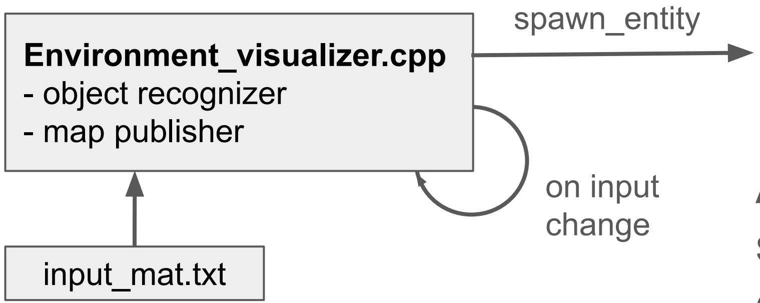
Path planning

input_mat.txt

path.txt

goal:
visualize map and path on gazebo iteratively

Map parsing and object recognition



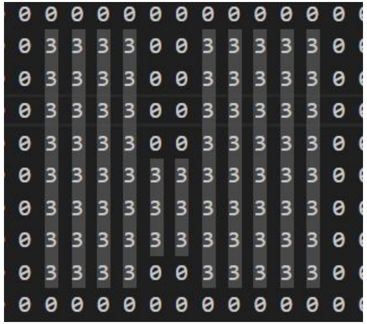
Environment_visualizer.cpp
- object recognizer
- map publisher

input_mat.txt

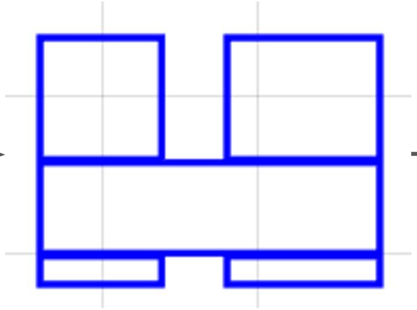
spawn_entity

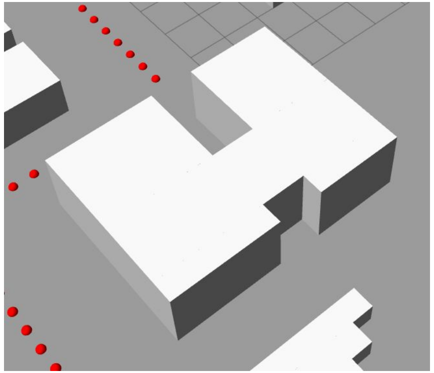
on input change

Algorithm to extract rectangular features of same height, and publish them as URDF objects through spawn_entity on gazebo

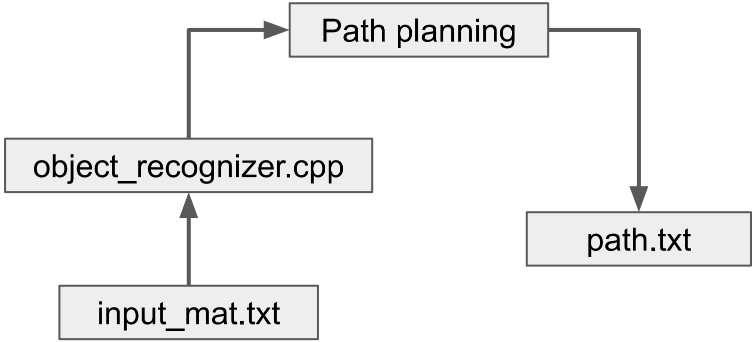


```
0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 3 3 3 3 3 0
0 3 3 3 3 0 0 3 3 3 3 3 0
0 3 3 3 3 0 0 3 3 3 3 3 0
0 3 3 3 3 0 0 3 3 3 3 3 0
0 3 3 3 3 3 3 3 3 3 3 3 0
0 3 3 3 3 3 3 3 3 3 3 3 0
0 3 3 3 3 3 3 3 3 3 3 3 0
0 3 3 3 3 0 0 3 3 3 3 3 0
0 0 0 0 0 0 0 0 0 0 0 0 0
```





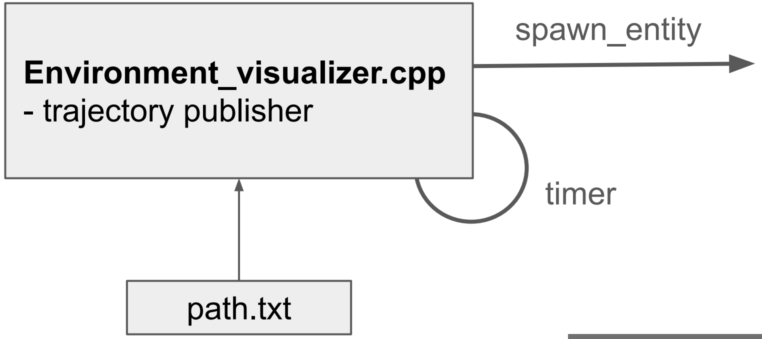
Path planning (?)



path.txt file: list of coords.

x1 y1
x2 y2
x3 y3
x4 y4
...

Path parsing and visualization



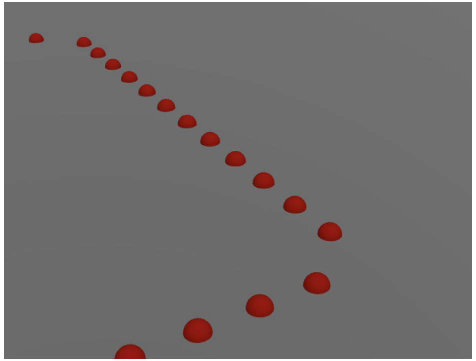
Environment_visualizer.cpp
- trajectory publisher

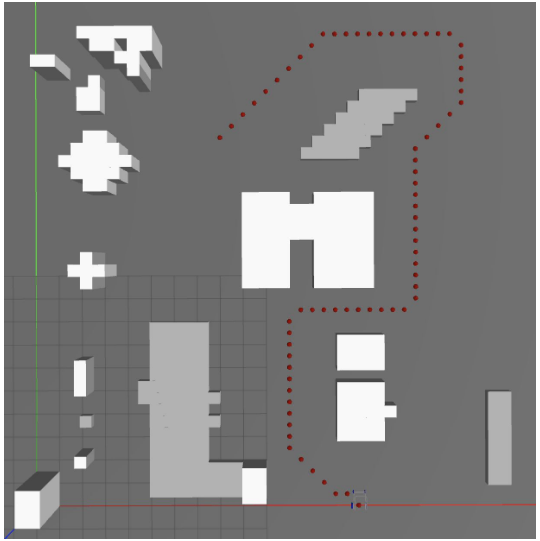
spawn_entity

timer

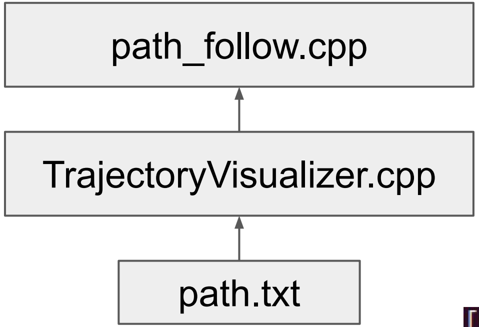
path.txt

Visualization of the path on Gazebo using spherical markers





Path Publisher

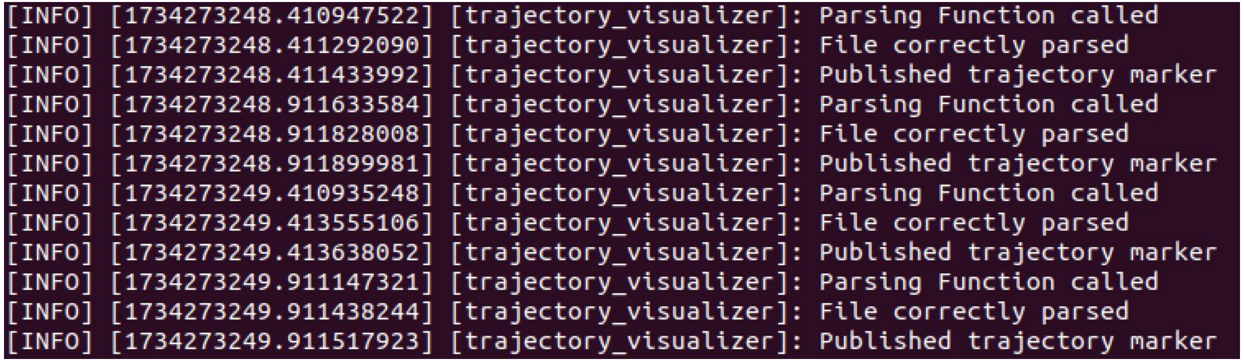


path_follow.cpp

TrajectoryVisualizer.cpp

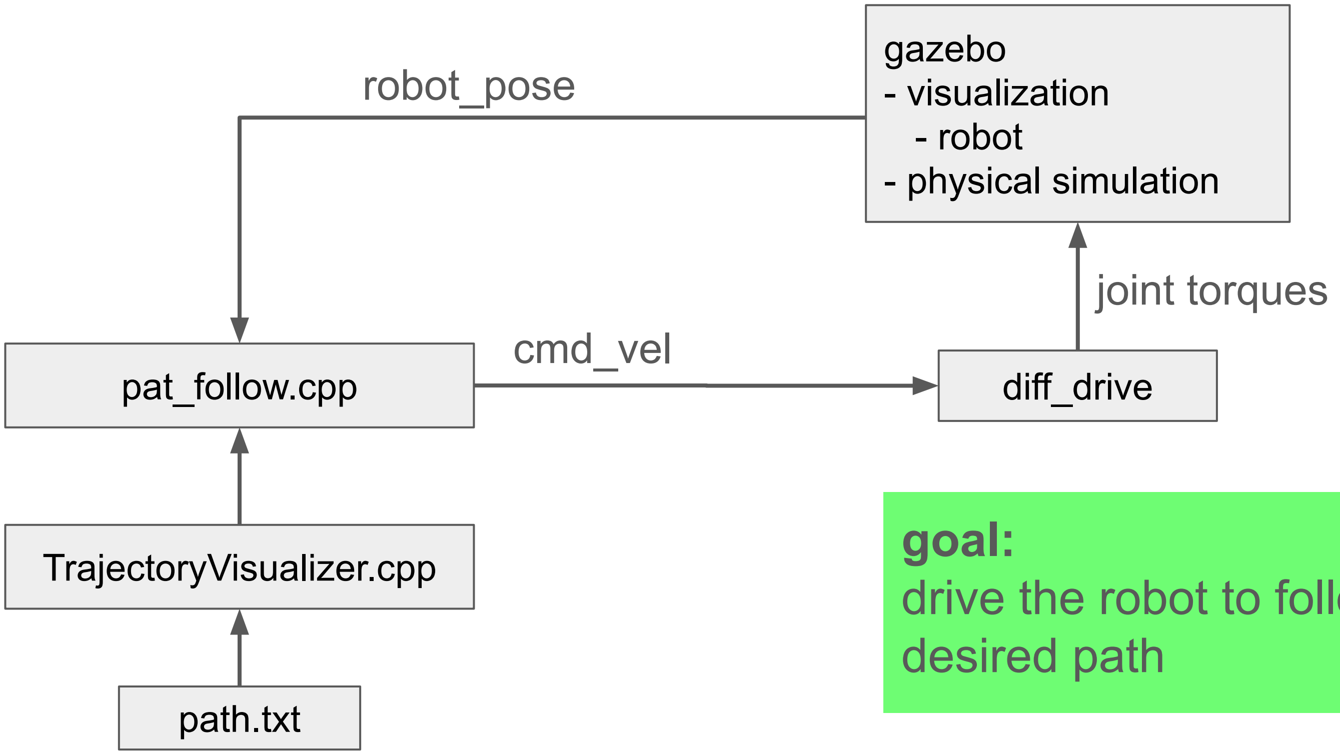
path.txt

Since the control node requires the path in a different format than Gazebo, we need a ros2 node to publish all the points at each time instant.



```
[INFO] [1734273248.410947522] [trajectory_visualizer]: Parsing Function called
[INFO] [1734273248.411292090] [trajectory_visualizer]: File correctly parsed
[INFO] [1734273248.411433992] [trajectory_visualizer]: Published trajectory marker
[INFO] [1734273248.911633584] [trajectory_visualizer]: Parsing Function called
[INFO] [1734273248.911828008] [trajectory_visualizer]: File correctly parsed
[INFO] [1734273248.911899981] [trajectory_visualizer]: Published trajectory marker
[INFO] [1734273249.410935248] [trajectory_visualizer]: Parsing Function called
[INFO] [1734273249.413555106] [trajectory_visualizer]: File correctly parsed
[INFO] [1734273249.413638052] [trajectory_visualizer]: Published trajectory marker
[INFO] [1734273249.911147321] [trajectory_visualizer]: Parsing Function called
[INFO] [1734273249.911438244] [trajectory_visualizer]: File correctly parsed
[INFO] [1734273249.911517923] [trajectory_visualizer]: Published trajectory marker
```

Path Control



gazebo
- visualization
    - robot
- physical simulation

robot_pose

joint torques

pat_follow.cpp

cmd_vel

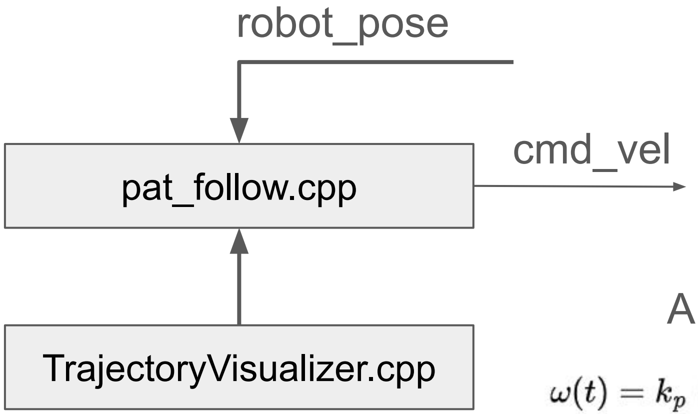
diff_drive

TrajectoryVisualizer.cpp

path.txt

goal:
drive the robot to follow the desired path

Stanley heuristic control method



robot_pose

pat_follow.cpp

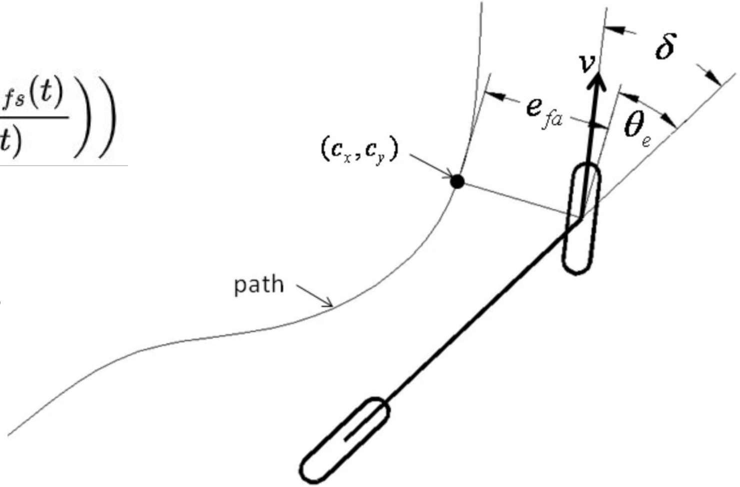
cmd_vel

TrajectoryVisualizer.cpp

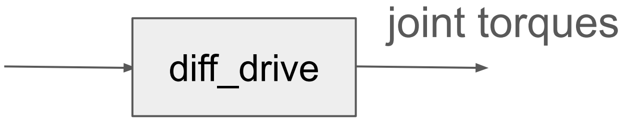
Inside the path_follow.cpp we have the control scheme which takes the reference path and the current robot pose to compute the command velocities using stanley method.

Angular velocity:

$$\omega(t) = k_p \left(\theta_e(t) + \arctan\left(\frac{k_e \cdot e_{fs}(t)}{v(t)} \right) \right)$$



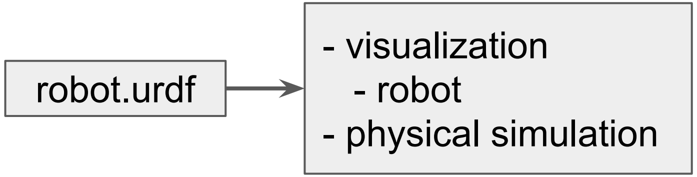
cmd_vel:
- Linear velocity
- Angular velocity



diff_drive

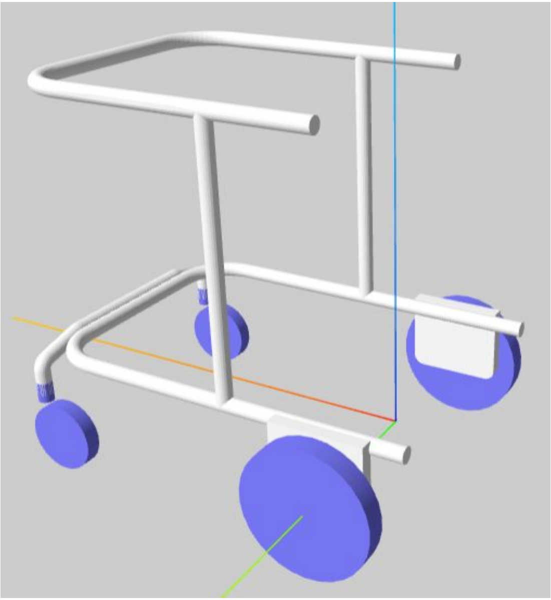
joint torques

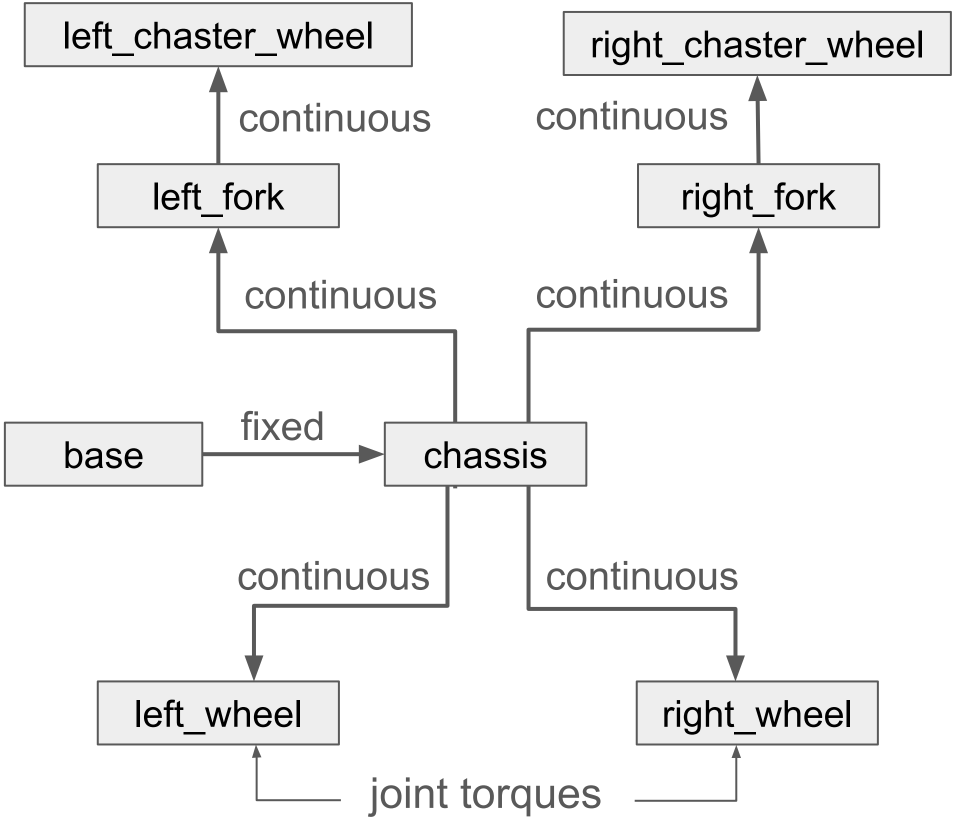
Deambulator URDF



robot.urdf

- visualization
  - robot
- physical simulation





left_chaster_wheel

right_chaster_wheel

continuous

continuous

left_fork

right_fork

continuous

continuous

base

fixed

chassis

continuous

continuous

left_wheel

right_wheel

joint torques