Team 15 Cars – Autonomous Driving

Introduction to ROS Summer Semester 23

Prof. Dr. Markus Ryll

Autonomous Aerial Systems

TUM School of Engineering and Design

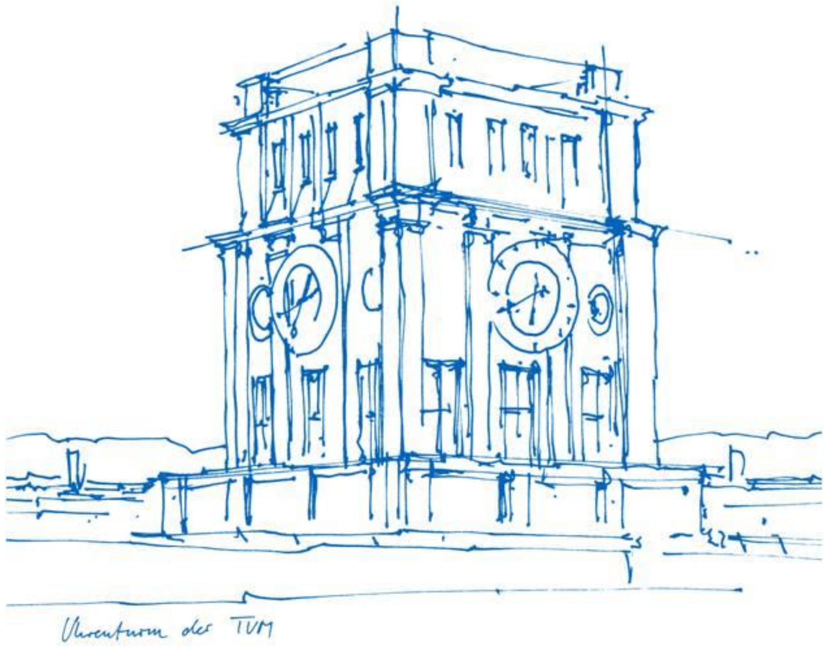
Jididu Li

Peishi Liu

Upendra Bevinamara Arun

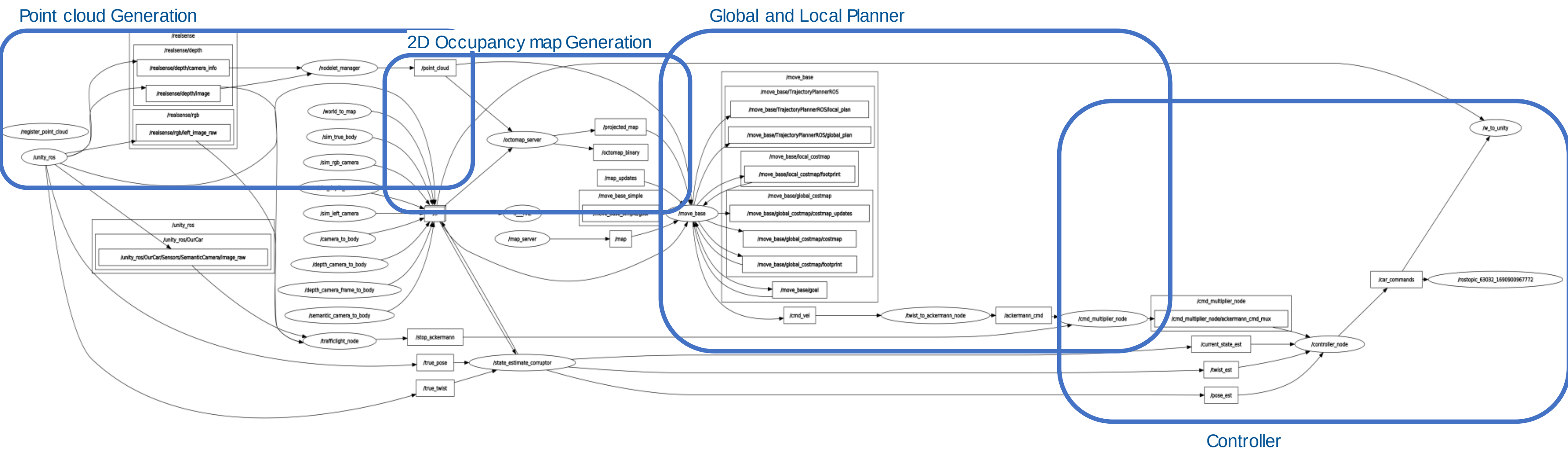
Xing Hu

Yuan Zhao



Contents

Overview of the pipeline

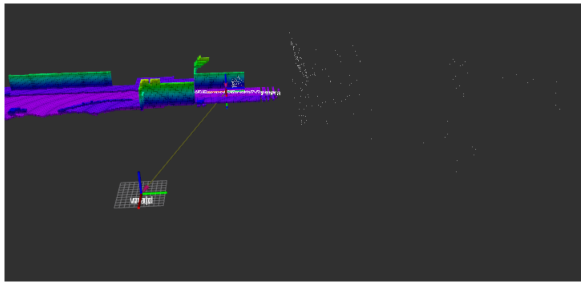


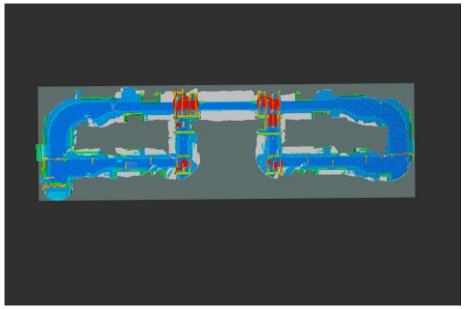
Perception: Point cloud and Occupancy map

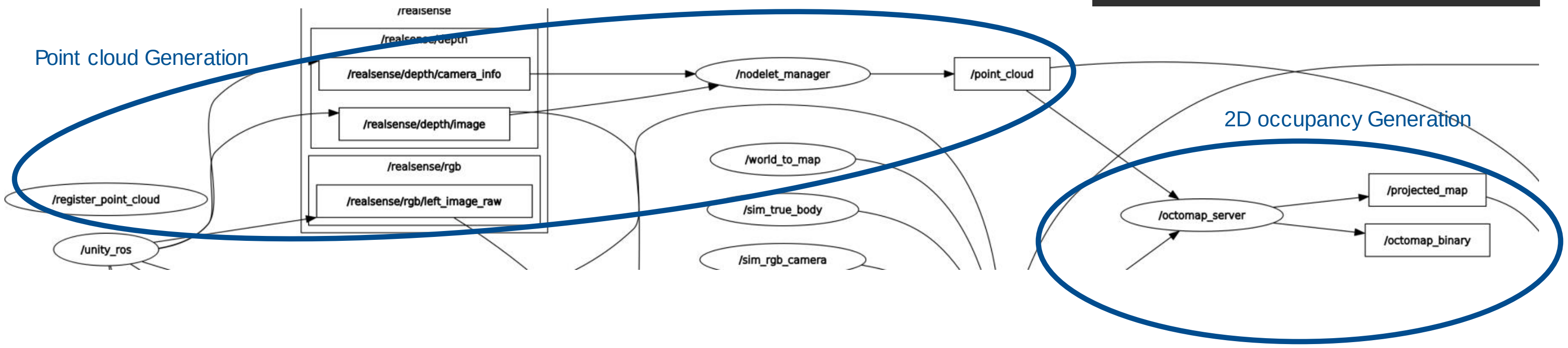
- **Initial step:**
- Transformation setup using tf2_ros package to get the sensor readings in the right directions.

- **Point cloud**
  - Input: Depth image feed and depth camera info
  - Output: Point cloud

- **Occupancy map**
  - Input: Point cloud
  - Setup to filter ground plane, sensor model max range and hit/miss
  - Output: 3D Voxel grid and 2D Occupancy map







Path planning: Global Planner

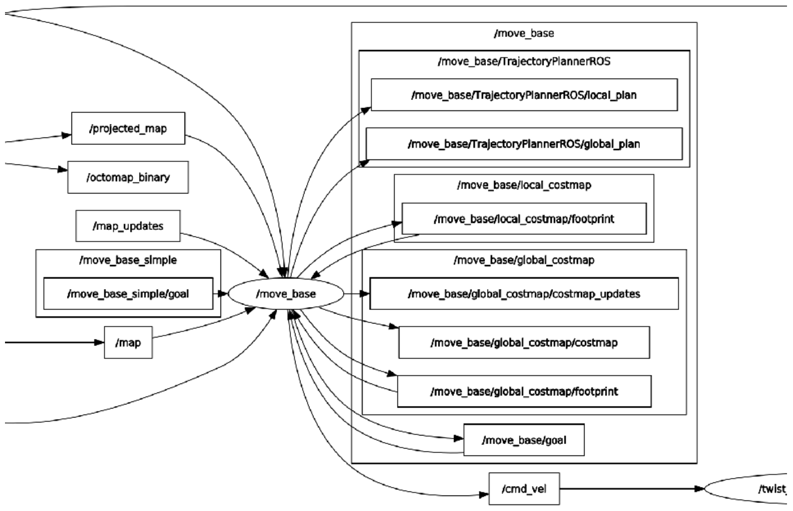
Backbone: Move Base package

Global Planner:
- Hard coded 20 waypoints for a smoother path
  - 2 points per turn (8 turns x 2 points = 16)
  - 2 points on the long straight
  - 2 points for start and goal position
  - Goal publisher node publishes these points to the move base package

Setup the Global planner and costmaps

Global Planner parameters:
- A* algorithm

Global and local cost map parameters:
- Obstacle and robot footprint setup



/move_base
/move_base/TrajectoryPlannerROS
/move_base/TrajectoryPlannerROS/local_plan
/move_base/TrajectoryPlannerROS/global_plan
/projected_map
/octomap_binary
/map_updates
/move_base/local_costmap
/move_base/local_costmap/footprint
/move_base_simple
/move_base_simple/goal
/move_base
/move_base/global_costmap
/move_base/global_costmap/costmap_updates
/map
/move_base/global_costmap/costmap
/move_base/global_costmap/footprint
/move_base/goal
/cmd_vel
/twist

Trajectory planning: Local planner

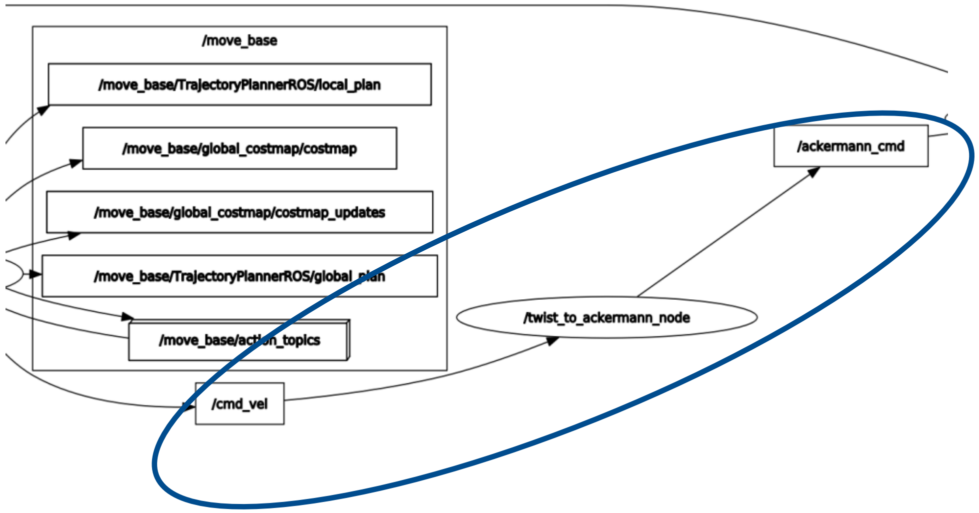
Desired Velocity and Steering angle for a front-wheel-steering vehicle

Used package:

- teb_local_planner
  1. time-optimal
  2. maximum velocities
  3. accelerations

Node:

- Twist_to_ackermann_node
  1. Messages with linear velocity and steering angle

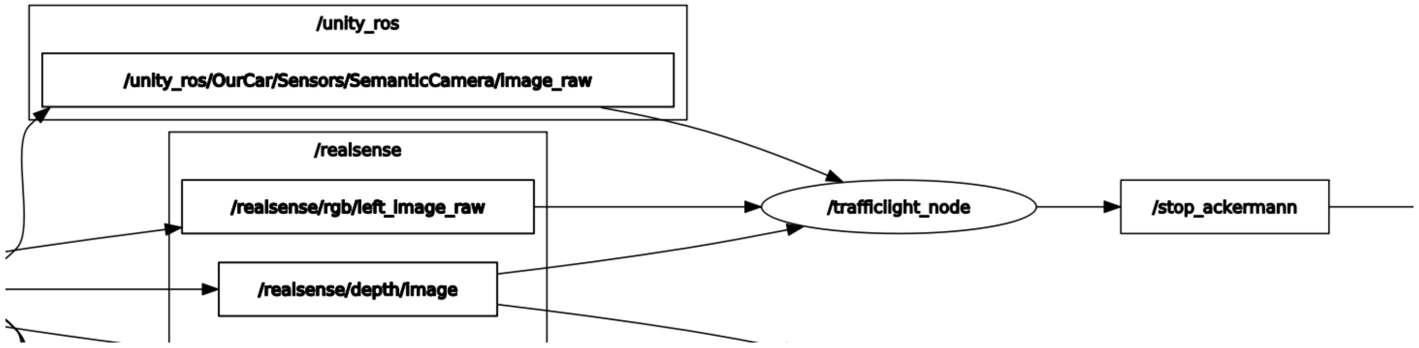

/move_base
/move_base/TrajectoryPlannerROS/local_plan
/move_base/global_costmap/costmap
/move_base/global_costmap/costmap_updates
/move_base/TrajectoryPlannerROS/global_plan
/move_base/action_topics
/cmd_vel
/twist_to_ackermann_node
/ackermann_cmd

Perception: Traffic light detection

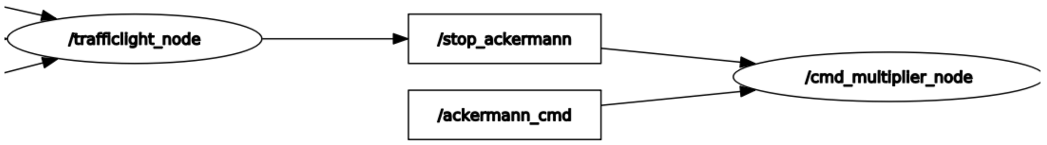
Multiple sensors used for traffic light detection
Specific image area would be processed
road constriction convention

- **Input**
  - Semantic camera: detection of traffic light
  - RGB camera: estimate current signal
  - Depth camera: calculate the distance

- **Output**
  - Ackermann msg with factors content.


/unity_ros
/unity_ros/OurCar/Sensors/SemanticCamera/image_raw
/realsense
/realsense/rgb/left_image_raw
/realsense/depth/image
/trafficlight_node
/stop_ackermann

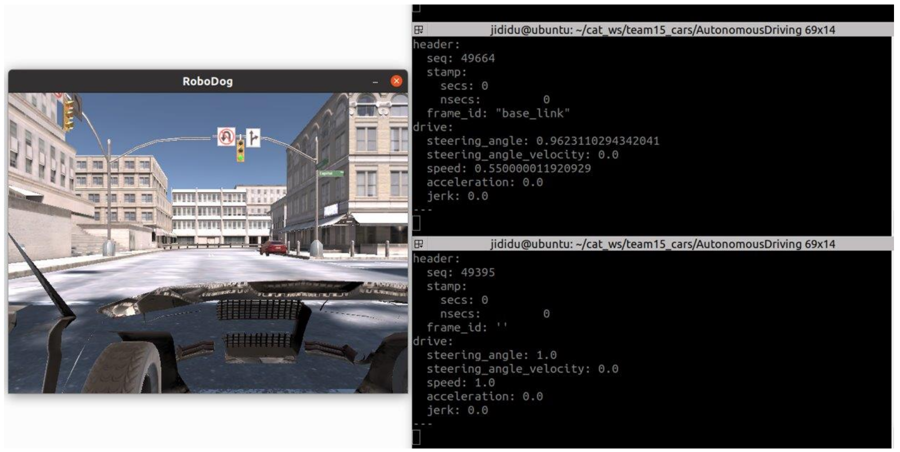
Trajectory planning: Command Fusion



Node:

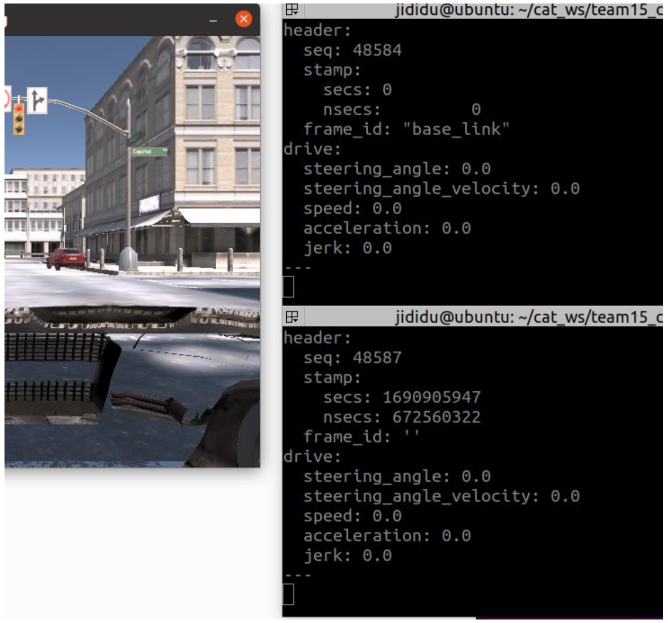
- /cmd_multiplier_node
  - Multiply corresponding value in /ackermann_cmd with factors

  (currently 1 for green light and 0 for red light)



```
jididu@ubuntu: ~/cat_ws/team15_cars/AutonomousDriving 69x14
header:
  seq: 49664
  stamp:
    secs: 0
    nsecs:         0
  frame_id: "base_link"
drive:
  steering_angle: 0.9623110294342041
  steering_angle_velocity: 0.0
  speed: 0.550000011920929
  acceleration: 0.0
  jerk: 0.0
---
```

```
jididu@ubuntu: ~/cat_ws/team15_cars/AutonomousDriving 69x14
header:
  seq: 49395
  stamp:
    secs: 0
    nsecs:         0
  frame_id: ''
drive:
  steering_angle: 1.0
  steering_angle_velocity: 0.0
  speed: 1.0
  acceleration: 0.0
  jerk: 0.0
---
```



```
jididu@ubuntu: ~/cat_ws/team15_c
header:
  seq: 48584
  stamp:
    secs: 0
    nsecs:         0
  frame_id: "base_link"
drive:
  steering_angle: 0.0
  steering_angle_velocity: 0.0
  speed: 0.0
  acceleration: 0.0
  jerk: 0.0
---
```

```
jididu@ubuntu: ~/cat_ws/team15_c
header:
  seq: 48587
  stamp:
    secs: 1690905947
    nsecs: 672560322
  frame_id: ''
drive:
  steering_angle: 0.0
  steering_angle_velocity: 0.0
  speed: 0.0
  acceleration: 0.0
  jerk: 0.0
---
```

Controller: Controller and Drive

Calculate the control value for the velocity and turning angle via two PID controller. Passing the calculated value to the Actuators, make sure our vehicle fellow the traffic light.

Node:

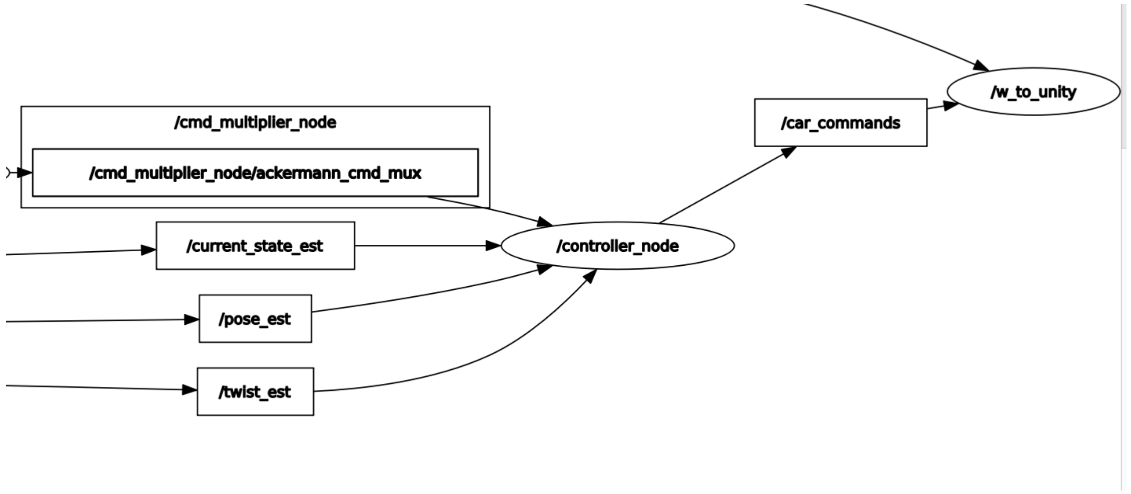
- Controller_node

Controller:

- PID controller for velocity and turning angle
- Integration of traffic light into control policy
- Publish control value to Actuators

Differently :

- Can not generate cost map(works for others).
- Command value are always zero.



Result

- Detects traffic lights and stops at Red and vehicle moves when Green.
- Avoids obstacles.
- Drives around as fast as possible.
  - Can not receive vel and acc signal(only local)

References

1. Point cloud from Depth Image: http://wiki.ros.org/depth_image_proc
2. Point cloud to Voxel Grid and Octomap: http://wiki.ros.org/octomap_server
3. Move base: http://wiki.ros.org/move_base
4. Teb local planner: http://wiki.ros.org/teb_local_planner
5. Command fusion with mux: http://wiki.ros.org/twist_mux
6. Image processing for traffic lights: https://opencv.org/
7. Communication between ROS and Open CV: http://wiki.ros.org/cv_bridge

Thank you for your attention