

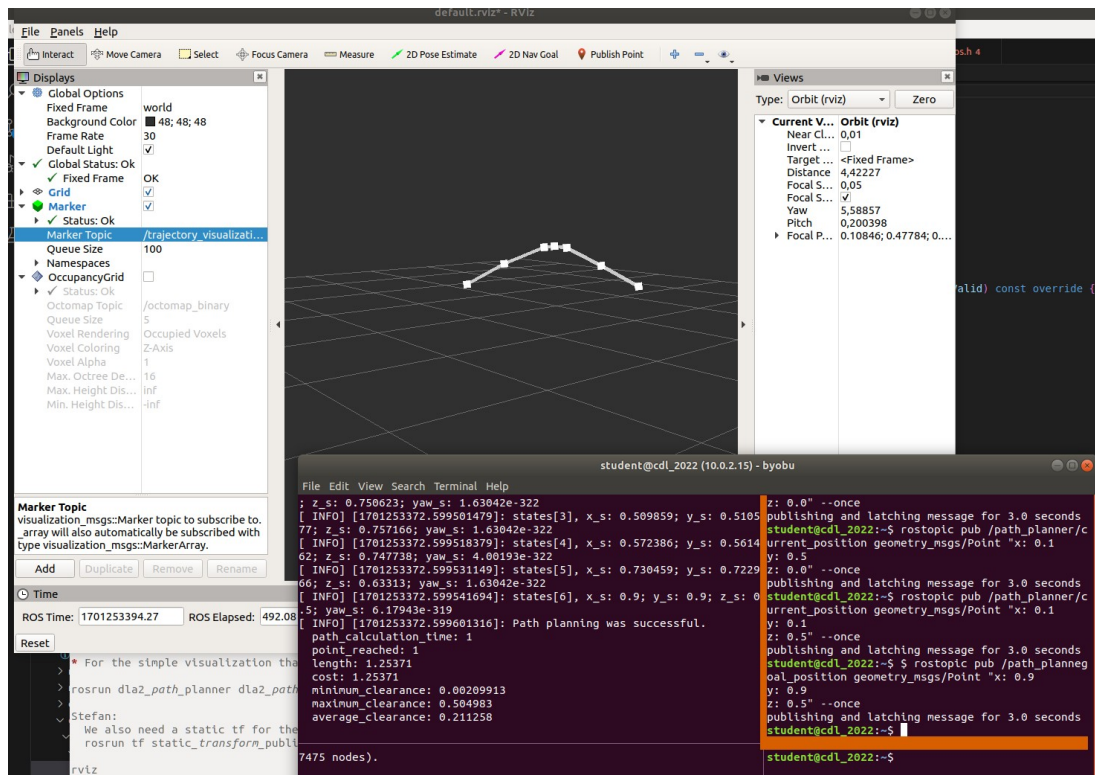
Assignment 2 - Report

Gruppe 23 - Florian Werkl, Stefan Schörkmeier

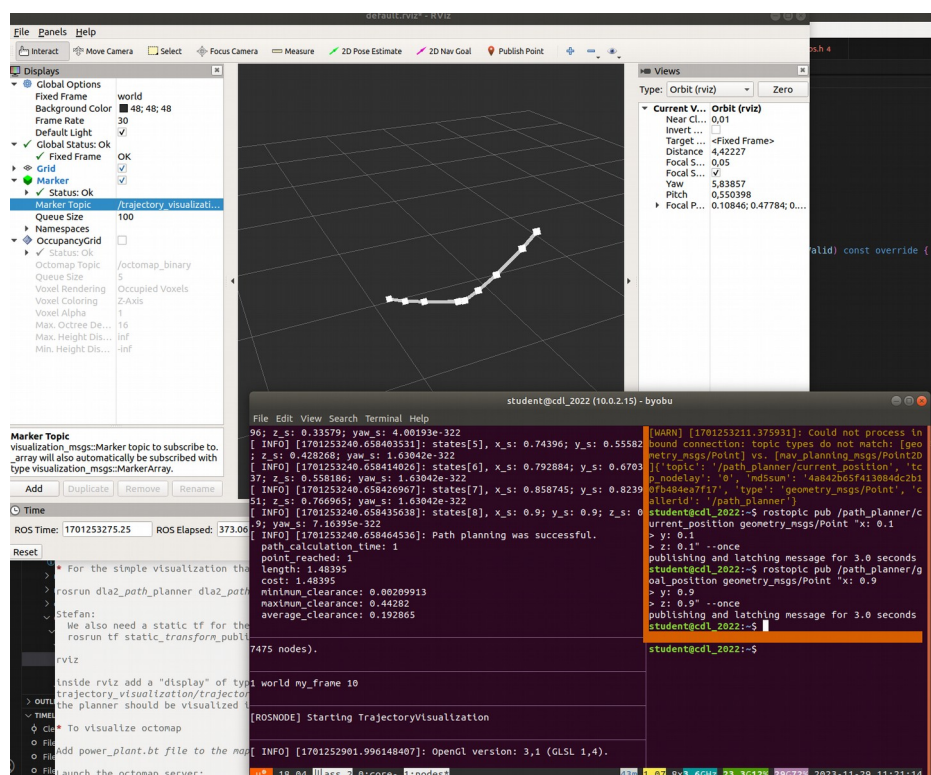
Image(s) showing your successful “first tests” using the OMPL library (for instance, using Fig. 1-left as obstacle map).

Unfortunately, we have too late read this task. So we only have the 3D case. We hope that’s enough to show our experiments – obviously we first tried out the 2D case as suggested :)

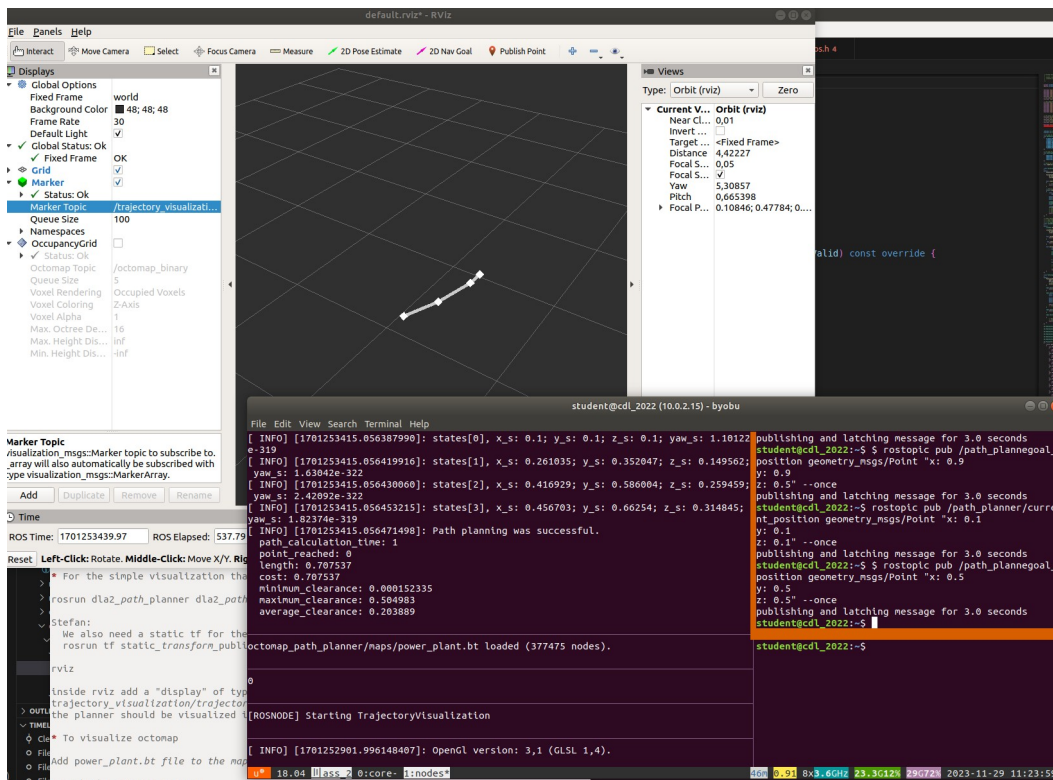
Here we can see an experiment where it comes as close to 2D as it gets. By setting the current and goal position values of z to 0.5 we only have movement on the x and y axis aka we only move on the 2D plane:



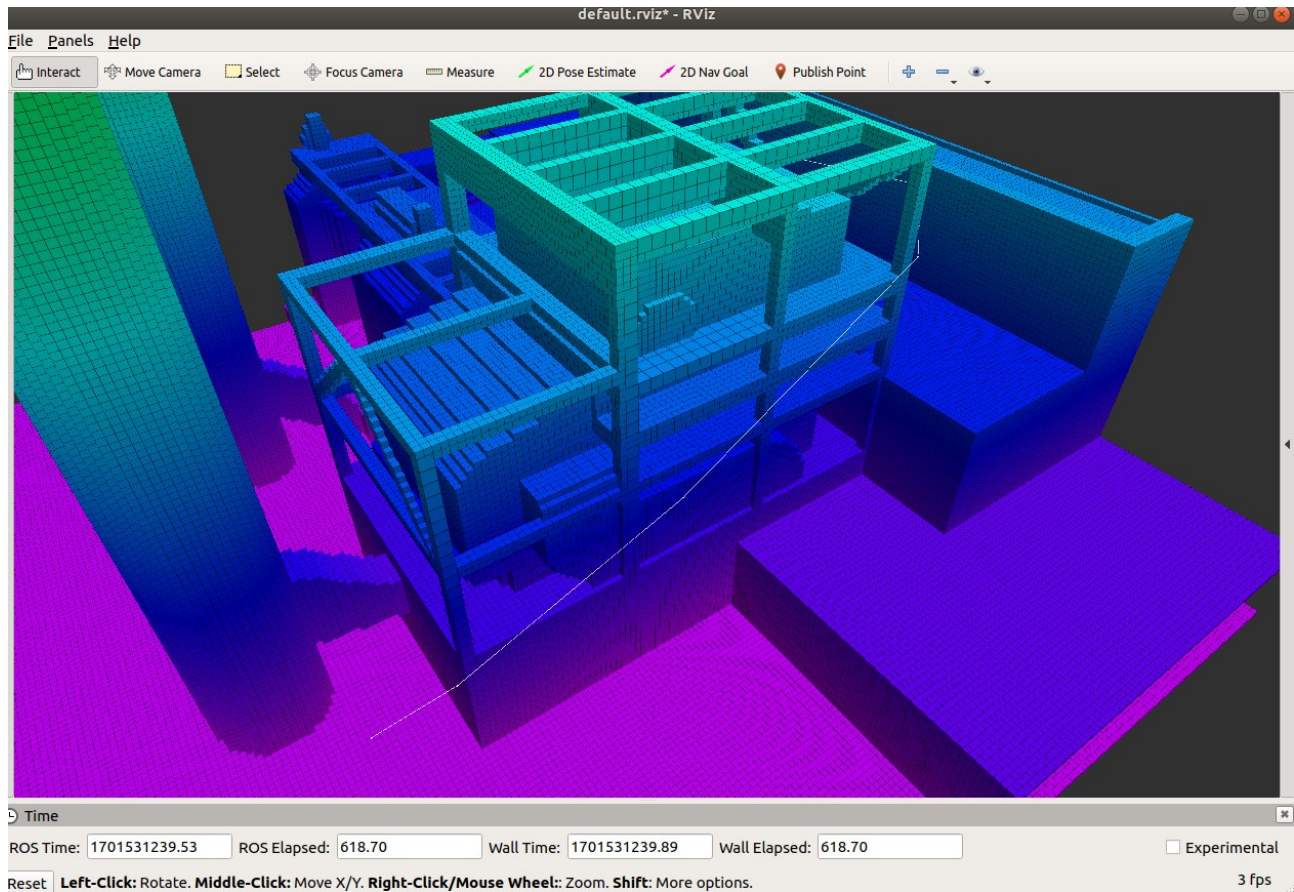
Then we moved in an 3D manner with the default locations of the README in you pathplanner dla file:



After that we did a little edge case, where we wanted a path right in the middle of the sphere (remember the sphere center is at 0.5, 0.5, 0.5 with an radius of 0.25). Its hard to see in that image, but the path only comes as close to the object as possible, but has not infered with the sphere.



Image(s) of your generated obstacle-free trajectories in the power_plant.bt octomap (Fig. 2).
 We got following path using the same coordinates as on Fig. 2 (from the point [0, 0, 3] to the point [10, -27, 15]) with length objective. As we can see the planned path snugs really close to the power plant (Do you believe as well that the drone will crash here? xD)



As we can see in the terminal, the path planning was successful and we ended up on the given point.

```

student@cdl_2022 (10.0.2.15) - byobu
File Edit View Search Terminal Help
tes in tree. Final solution cost 40.594
RRTstar found a solution of length 40.594 with an optimization objective value of 40.594
[ INFO ] [1701531200.394507850]: states[0], x_s: 0; y_s: 0; z_s: 3; yaw_s: 1.63042e-322
[ INFO ] [1701531200.394571509]: states[1], x_s: -1.2516; y_s: -3.09095; z_s: 5.0603; yaw_s: 1.63042e-322
[ INFO ] [1701531200.394584774]: states[2], x_s: -2.61127; y_s: -11.7181; z_s: 10.3549; yaw_s: 1.63042e-322
[ INFO ] [1701531200.394593150]: states[3], x_s: -2.06771; y_s: -22.5899; z_s: 15.9629; yaw_s: 1.63042e-322
[ INFO ] [1701531200.394603179]: states[4], x_s: 1.70258; y_s: -27.0028; z_s: 16.3152; yaw_s: 4.00193e-322
[ INFO ] [1701531200.394618237]: states[5], x_s: 10; y_s: -27; z_s: 15; yaw_s: 3.40345e-318
[ INFO ] [1701531200.394645839]: Path planning was successful.
  path_calculation_time: 15000ms
  point_reached: 1
  length: 40.594
  cost: 40.594
  minimum_clearance: 2.34808
  maximum_clearance: 32.2577
  average_clearance: 19.2212

process[octomap_server-1]: started with pid [31784]
[ INFO ] [1701530620.768841195]: Publishing latched (single publish will take longer, all topics are prepared)
[ WARN ] [1701530620.779128513]: Nothing to publish, octree is empty
[ INFO ] [1701530620.808960226]: Octomap file /home/student/camera-drones/catkin_ws/src/octomap_path_planner/maps/power_plant.bt loaded (377475 nodes).

0

[ROSNODE] Starting TrajectoryVisualization

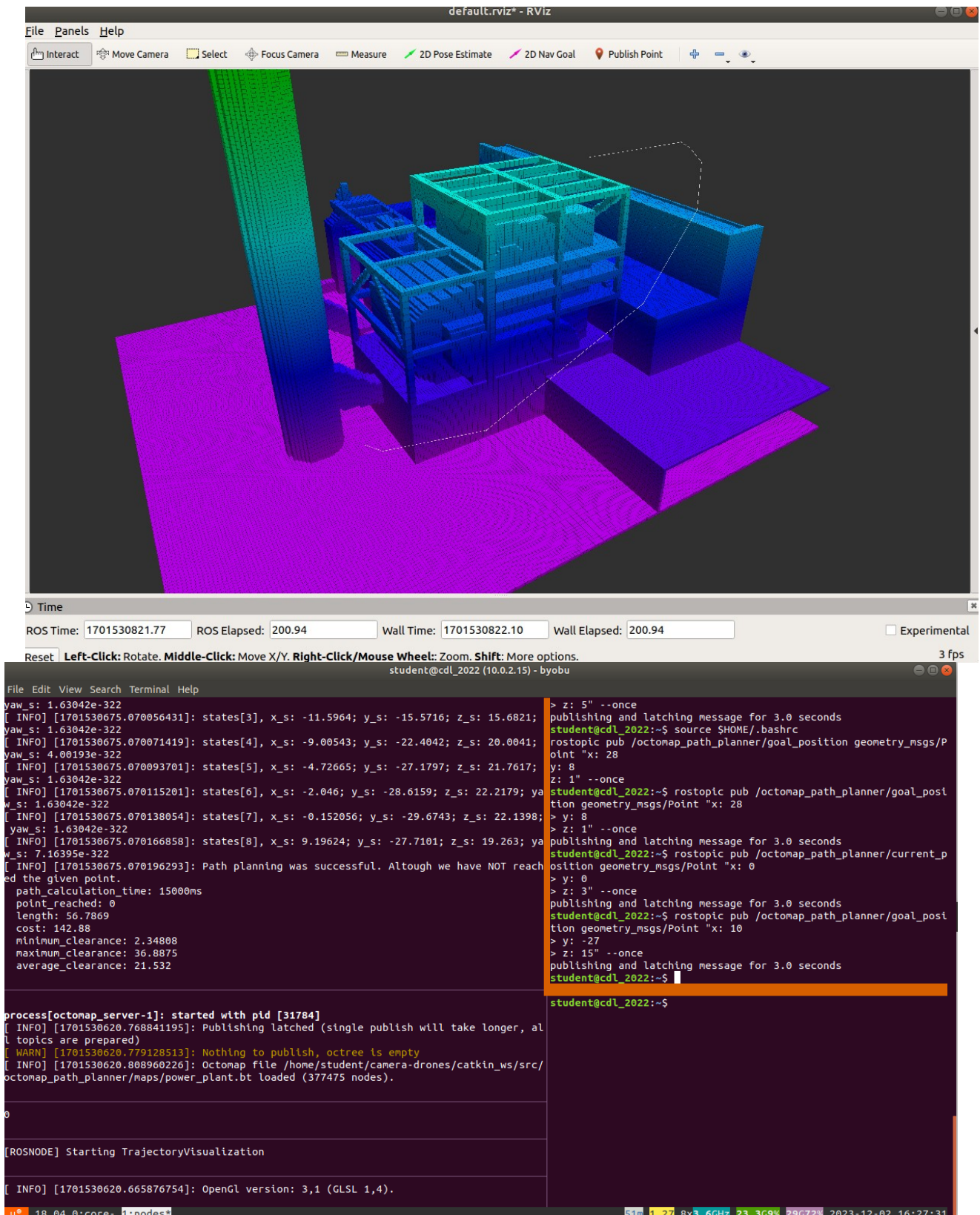
[ INFO ] [1701530620.665876754]: OpenGL version: 3.1 (GLSL 1.4).

u* 18.04 0:core- 1:nodes*
1h1m 2.74 8x3.6GHz 23.3G13% 29G72% 2023-12-02 16:37:05

```

Same as previous, but of trajectories showing good clearance (distance to any obstacles).

We got following path using the same coordinates as on Fig. 2 (from the point [0, 0, 3] to the point [10, -27, 15]) with clearance and length objective. As we can see in the terminal output the path planner was successful but has not reached the given point. That might be, because our clearance doesn't allow to get as close to the power plant. **QUESTION:** should the path look exactly as yours or is it ok this way? :)



The image displays a screenshot of the RViz (Robot Visualization) interface, showing a 3D environment with a path and a terminal window with ROS logs.

RViz Interface:

- Top Panel:** Contains a toolbar with icons for Interact, Move Camera, Select, Focus Camera, Measure, 2D Pose Estimate, 2D Nav Goal, and Publish Point.
- Main View:** A 3D visualization of a robot environment. A green path is shown, starting from a point and moving through the environment. The environment includes a large blue structure (possibly a power plant) and a smaller blue structure (possibly a robot or obstacle).
- Bottom Panel:** A status bar showing Time, ROS Time, ROS Elapsed, Wall Time, Wall Elapsed, and Experimental checkbox. It also includes a Reset button and a 3 fps display.

Terminal Window:

```
File Edit View Search Terminal Help
student@cdl_2022 (10.0.2.15) - byobu

yaw_s: 1.63042e-322
[ INFO] [1701530675.070056431]: states[3], x_s: -11.5964; y_s: -15.5716; z_s: 15.6821;
yaw_s: 1.63042e-322
[ INFO] [1701530675.070071419]: states[4], x_s: -9.00543; y_s: -22.4042; z_s: 20.0041;
yaw_s: 4.00193e-322
[ INFO] [1701530675.070093701]: states[5], x_s: -4.72665; y_s: -27.1797; z_s: 21.7617;
yaw_s: 1.63042e-322
[ INFO] [1701530675.070115201]: states[6], x_s: -2.046; y_s: -28.6159; z_s: 22.2179; ya
w_s: 1.63042e-322
[ INFO] [1701530675.070138054]: states[7], x_s: -0.152056; y_s: -29.6743; z_s: 22.1398;
yaw_s: 1.63042e-322
[ INFO] [1701530675.070166858]: states[8], x_s: 9.19624; y_s: -27.7101; z_s: 19.263; ya
w_s: 7.16395e-322
[ INFO] [1701530675.070196293]: Path planning was successful. Although we have NOT reach
ed the given point.
path_calculation_time: 15000ms
point_reached: 0
length: 56.7869
cost: 142.88
minimum_clearance: 2.34808
maximum_clearance: 36.8875
average_clearance: 21.532

> z: 5" --once
publishing and latching message for 3.0 seconds
student@cdl_2022:~$ source $HOME/.bashrc
rostopic pub /octomap_path_planner/goal_position geometry_msgs/P
oint "x: 28
y: 8
z: 1" --once
student@cdl_2022:~$ rostopic pub /octomap_path_planner/goal_posi
tion geometry_msgs/Point "x: 28
y: 8
z: 1" --once
publishing and latching message for 3.0 seconds
student@cdl_2022:~$ rostopic pub /octomap_path_planner/current_p
osition geometry_msgs/Point "x: 0
y: 0
z: 3" --once
publishing and latching message for 3.0 seconds
student@cdl_2022:~$ rostopic pub /octomap_path_planner/goal_posi
tion geometry_msgs/Point "x: 10
y: -27
z: 15" --once
publishing and latching message for 3.0 seconds
student@cdl_2022:~$

student@cdl_2022:~$

process[octomap_server-1]: started with pid [31784]
[ INFO] [1701530620.768841195]: Publishing latched (single publish will take longer, al
l topics are prepared)
[ WARN] [1701530620.779128513]: Nothing to publish. octree is empty
[ INFO] [1701530620.808960226]: Octomap file /home/student/camera-drones/catkin_ws/src/
octomap_path_planner/maps/power_plant.bt loaded (377475 nodes).

[ROSNODE] Starting TrajectoryVisualization

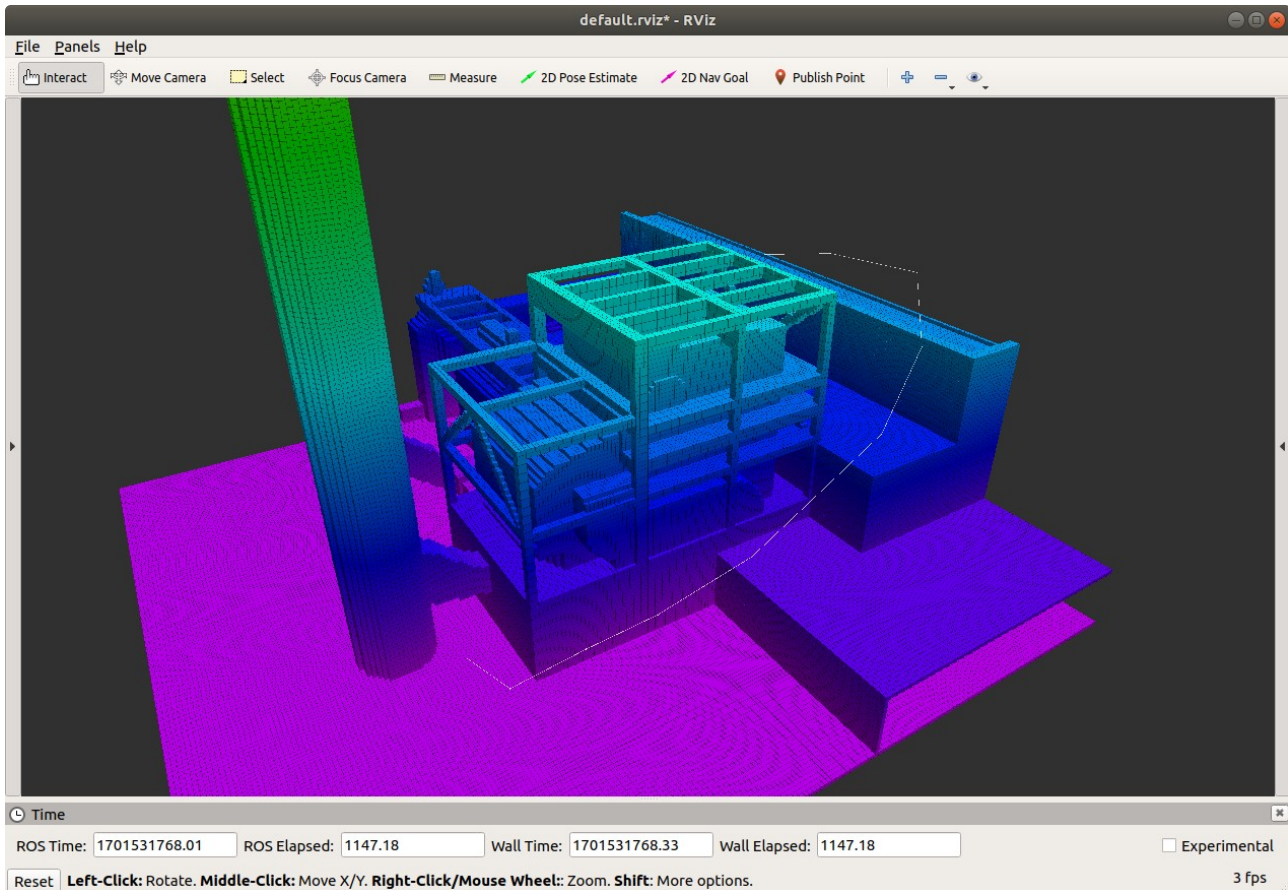
[ INFO] [1701530620.665876754]: OpenGL version: 3.1 (GLSL 1.4).

0* .18.04 0:core- 1:nodes* 51% 1.27 8x3.6GHz 23.369% 29G72% 2023-12-02 16:27:31
```


Same as previous, but of trajectories that are optimized to avoid unnecessary movement in the altitude direction, preferring longer horizontal paths instead.

We got following path using the same coordinates as on Fig. 2 (from the point [0, 0, 3] to the point [10, -27, 15]) with clearance and length objective and an optimization to avoid unnecessary movements in the altitude.

It is quite hard to see on the image, but have an look in the middle (on the purple step) there you can see that it is much more flatter then in the image of the previous part.



```
student@cdl_2022 (10.0.2.15) - byobu
File Edit View Search Terminal Help
yaw_s: 1.63042e-322
[ INFO] [1701531635.647370978]: states[4], x_s: -11.0274; y_s: -10.1096; z_s: 12.746; y
aw_s: 4.00193e-322
[ INFO] [1701531635.647387328]: states[5], x_s: -10.3843; y_s: -18.1682; z_s: 16.3104;
yaw_s: 1.63042e-322
[ INFO] [1701531635.647400724]: states[6], x_s: -7.21932; y_s: -23.9916; z_s: 18.3115;
yaw_s: 1.63042e-322
[ INFO] [1701531635.647413007]: states[7], x_s: -2.65945; y_s: -28.2259; z_s: 20.3497;
yaw_s: 1.63042e-322
[ INFO] [1701531635.647428576]: states[8], x_s: 4.74903; y_s: -28.5842; z_s: 19.5385; y
aw_s: 7.16395e-322
[ INFO] [1701531635.647440929]: states[9], x_s: 10.8107; y_s: -27.3722; z_s: 18.1076; y
aw_s: 3.38736e-319
[ INFO] [1701531635.647472328]: Path planning was successful. Although we have NOT reach
ed the given point.
  path_calculation time: 15001ms
  point_reached: 0
  length: 56.2293
  cost: 150.035
  minimum_clearance: 2.34808
  maximum_clearance: 34.8095
  average_clearance: 21.9641
> z: 15" --once
publishing and latching message for 3.0 seconds
student@cdl_2022:~$ rostopic pub /octomap_path_planner/current_p
osition geometry_msgs/Point "x: 0
y: 0
z: 3" --once
publishing and latching message for 3.0 seconds
student@cdl_2022:~$ rostopic pub /octomap_path_planner/goal_posi
tion geometry_msgs/Point "x: 10
y: -27
z: 15" --once
publishing and latching message for 3.0 seconds
student@cdl_2022:~$ rostopic pub /octomap_path_planner/goal_posi
tion geometry_msgs/Point "x: 10
y: -27
z: 15" --once
publishing and latching message for 3.0 seconds
student@cdl_2022:~$

process[octomap_server-1]: started with pid [31784]
[ INFO] [1701530620.768841195]: Publishing latched (single publish will take longer, al
l topics are prepared)
[ WARN] [1701530620.779128513]: Nothing to publish, octree is empty
[ INFO] [1701530620.808960226]: Octomap file /home/student/camera-drones/catkin_ws/src/
octomap_path_planner/maps/power_plant.bt loaded (377475 nodes).

0

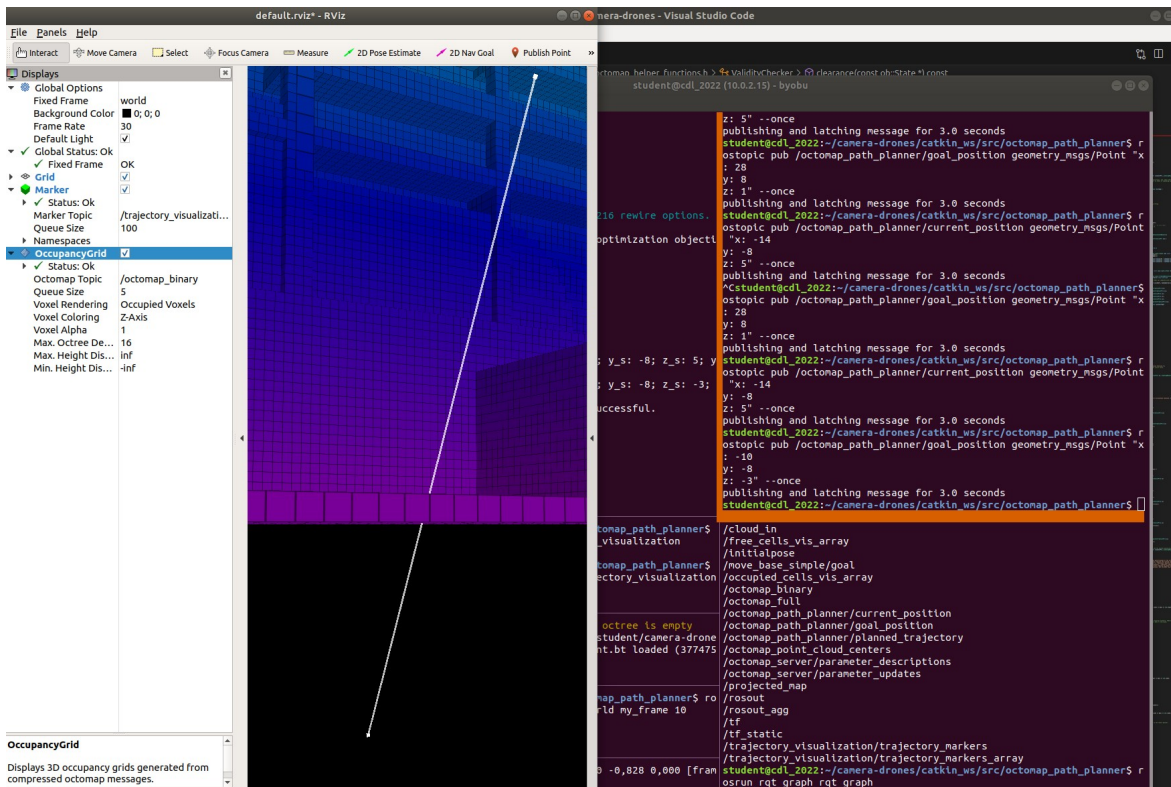
[ROSNODE] Starting TrajectoryVisualization

[ INFO] [1701530620.665876754]: OpenGL version: 3.1 (GLSL 1.4).

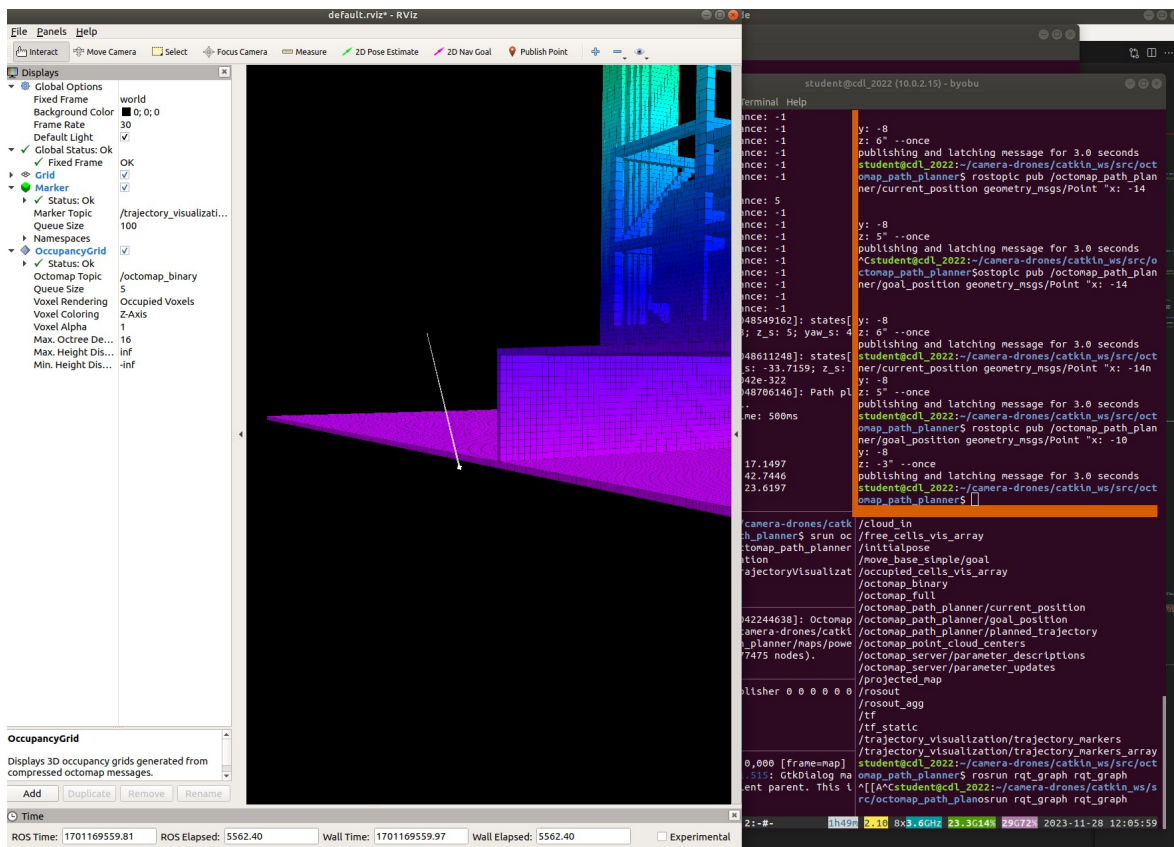
u* 18.04 0:core- 1:nodes 1hr 1.71 8x3.6GHz 23.36G13% 29G72% 2023-12-02 16:42:35
```

Find a test case that causes your trajectory planner to generate a trajectory that goes through a wall. To avoid this buggy behavior, for instance, define a MotionValidator for your trajectory planner, which overrides the default implementation and uses the ray-casting functionality of the Octomap library to determine that a trajectory segment is obstacle-free. For ease of this task we allowed that z is negative and set a small runtime (0.1)

As we can see in this image with no raycast the first path went directly through the floor plane:



While with an activated raycast we ended up on a random-ish position (but without going through an obstacle)



Find out how to make the OMPL library stop the trajectory search process, other than using a fixed time allotment for this task. Report on the advantages of your alternate solution.

Here we simply used an max number of iterations:

```
ob::PlannerStatus solved = ob::PlannerStatus::UNKNOWN;
// One of those two variables has to be set.
if (runTime > 0) {
    solved = optimizingPlanner->solve(runTime);
}
else if (optimizingPlannerMaxIterations > 0){
    solved = optimizingPlanner->solve(ompl::base::IterationTerminationCondition(optimizingPlannerMaxIterations));
}
```

The advantage of using iterations instead of time is, that on the one hand it makes our algorithm independent of the environment/hardware (eg runtime of 10 seconds on a much stronger machine will most likely lead to a different solution then on a weaker machine) and on the other hand it should be quite predictable. With fixed time we could always have fluctuations (eg different amount of processes running, ...)