TU Graz

Camera Drones Lecture 2023-24
Jesús Pestana <jesus.pestana@pro2future.at>
Friedrich Fraundorfer <fraundorfer@icg.tugraz.at>

Practical Part 4 – Pre-calculation of trajectories for flight experiment

ROS related tasks

- Use a synthetically generated octomap of the Dronespace with the trajectory planner and smoother from the previous practicals (2 & 3 respectively).
- ROS bag recording of the planned trajectories (for the flight experiment)
- (Optional) Test different parameter settings for the trajectory planner and smoother.

Task Description

This is intended to be a very short practical exercise. You will use your software developed in previous practicals (2 & 3) to provide pre-calculated trajectories (examples shown in Fig. 1) for a flight experiment. You will use a synthetically generated octomap of the Dronespace (provided along with this practical) for this purpose. A description and pictures (Fig. 2) of the Dronespace can be found in the appendix at the end of this document.

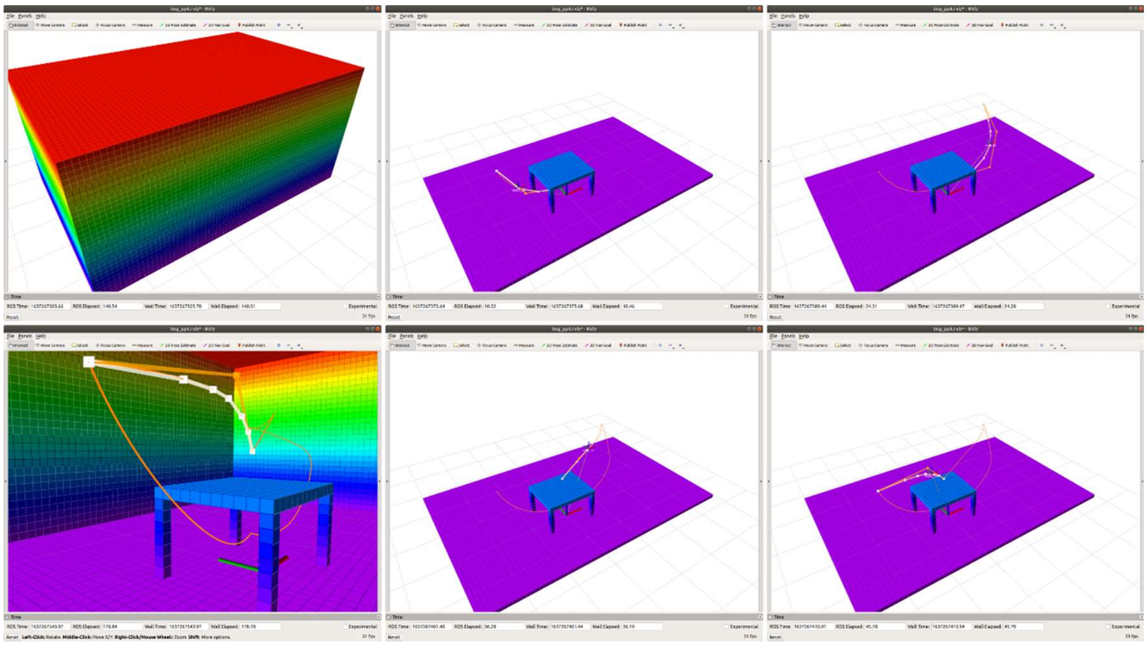

Fig. 1. Synthetically generated obstacle-map (octomap) of the Dronespace with examples of planned trajectories required for this practical: (left) obstacle-map used for trajectory planning, and (center and right) obstacle-map used to be displayed in 3D using ROS RViz. Regarding the trajectories: trajectory planned using the RRTStar before (orange with square markers) and after (white with markers) applying BSpline path smoothing; the (orange) trajectories show the fully smoothed - quintic spline - trajectory.

The synthetically generated octomap corresponds to setting up the Dronespace completely empty and placing in it only one table, which will be used as an obstacle. As shown in Fig. 1, we are using different octomaps for the trajectory planning step and for displaying results in 3D in ROS RViz, their filenames are respectively: ‘dronespace_synthetic.bt’ and ‘dronespace_synthetic_display.bt’.
Note: instead of using 2 maps, this could also be achieved using the additional explored/unexplored free-space information contained in the octomap, but this feature would need to be coded properly in the trajectory planner and the distance-map, adding unnecessary burden to this practical.

During the flight experiment, the drone will execute the following flight schedule:

1. Take-off (landed position approximately directly below waypoint 0)
2. Fly to waypoint 0
3. Fly to waypoint 1 (below table)
4. Fly to waypoint 2 (opposite end of the Dronespace)
5. Fly to waypoint 3 (above table)
6. Fly to waypoint 4 (== waypoint 0)
7. Land

Your task is to provide us with obstacle-free smooth trajectories like the ones shown in Fig. 1. As explained below, you will provide 1 rosbag and 1 image per trajectory (trajectory from WP0 to WP1, trajectory from WP1 to WP2, and so on). For this task, utilize the following waypoints:

- WP0= {x: -1.75, y: 0.00, z: 1.50}
- WP1= {x: -0.03, y: 0.13, z: 0.35}
- WP2= {x: +2.25, y: 1.50, z: 1.50}
- WP3= {x: -0.03, y: 0.13, z: 1.10}
- WP4= WP0

The utilized ROS architecture for the flight experiment is the following:

- mocap_node (ROS Node 1): publishes the drone pose measurements obtained by the motion tracking system.
- autopilot_interface (Node 2): publishes the Inertial Measurement Unit (IMU) data measured by the autopilot board, that is: gyroscope (rotation velocity [rad/s] in body coordinates) and accelerometer (acceleration minus gravity [m/s^2] in body coordinates) data. It also re-directs the low-level flight commands to the autopilot board, that is: thrust [%], yaw heading rotation speed [rad/s], and pitch and roll angles [rad].
- state_estimator (Node 3): subscribes to the pose measurements and the IMU data and fuses these data sources to estimate (and publish) the drone state (position, speed and acceleration; rotation and rotation speed). The drone state estimates are calculated at the same timestamps as the IMU measurements, therefore they can be published at IMU rate (regardless of the rate of the pose measurements).
- trajectory_planner (Node 4): plans a trajectory from the current to a desired drone position.
- rosbag_play (Node 4): in our case, the trajectory_planner is substituted by playing back rosbag recordings that contain the pre-calculated trajectories (from and to waypoints).
- navigation_controller (Node 5): calculates low-level flight commands for the autopilot board. This calculation requires: (1) state references - either a trajectory under execution or a desired pose {position, heading} - and (2) state feedback - the current drone state estimate. Note: The reference trajectory is sampled in time, with respect to the time at which its execution was initialized. This means that the trajectory is a function of time (quintic spline on 't'), for which we also know the derivatives (polynomial derivatives), that we evaluate at a given time value in order to obtain drone state references (position, speed and acceleration, and yaw heading references) for the navigation controller. For this we use the relevant code provided by the authors of the trajectory smoother that we selected for the practical 3 [R3].

- (optional) task_scheduler (Node 6): this node commands the navigation_controller, the trajectory_planner and the autopilot_interface with simple instructions such as: start/stop drone propellers, take-off, calculate trajectory to waypoint, execute trajectory, hover, hover to waypoint and land. Each task has a clear completion criteria. By initiliazing and monitoring the tasks sequentially, the scheduler makes the drone follow the flight schedule automatically.
  Note: for instance, the completion criteria of a trajectory execution task is monitored in time (elapsed time greater than trajectory duration) and in position (distance to goal point).

Task steps

(step 1) Configure your {trajectory planner + smoother} ROS architecture from the practicals 2 & 3 to use the provided obstacle maps: ‘dronespace_synthetic.bt’ and ‘dronespace_synthetic_display.bt’ (for, respectively, the trajectory planning and the visualization in ROS RViz, see Fig. 1).

Test that your setup is working properly.

[optional] (step 2) Test different parameter settings for the trajectory planner and smoother. At first, our main objective, as this would be the first time that we test your trajectory planner, is flight safety. Therefore, we care about: (planner settings) distance to obstacles and (smoother settings) flying at reasonable accelerations and speeds. Later, it would be interesting to test the system further, in order to identify further points to improve the drone flight (for instance, test for better parameters for the trajectory smoother).

For the trajectories shown in Fig. 1, the following parameters were used:

- Trajectory simplification:
  - snapToVertex parameter is set to 0.1, when calling the function: simplifier.shortcutPath(…)
  - The following functions were not used for the path shortening: simplifier.smoothBSpline(…), simplifier.perturbPath(…)
  - Trajectory smoother (mav_trajectory_generation) soft constraints: (velocity) max_v: 2.0, (acceleration) max_a: 0.5, (jerk) max_j: 0.75, (snap) max_s: 1.50

(step 3) Take a screenshot and record a rosbag of the obtained trajectory per waypoint pair (example: trajectory from WP0 to WP1) with 1 msg per topic of the following ROS topics:

```
/path_planner/planned_trajectory
/path_planner/planned_trajectory_raw
/trajectory_generator/smooth_trajectory
/trajectory_generator/smooth_trajectory_markers
/trajectory_visualization/trajectory_markers
/trajectory_visualization_raw/trajectory_markers_raw
```

See the file ‘trajectory_wp1_to_wp3.bag’ for an example. You can inspect the contents of a rosbag file using the terminal command ‘rosbag info <bagfile_name>’.

(step 4) Upload the rosbags and screenshots of the trajectories, with the corresponding information to identify your ‘team’ of the Camera Drones Lecture, to the teach-center. A video with the playback of one of your generated trajectories is also welcome.

Appendix: Dronespace Description

The Dronespace laboratory of the Institute of Computer Graphics and Vision (ICG) (TU Graz) is a small space equipped with a motion tracking system, which main purpose is (1) the acquisition of ground-truth data, (2) the realization of experiments in the areas of Robotics (mainly small drone flights), Computer Vision and Augmented Reality and (3) supporting the experimental part of some lectures at the TU Graz like the Camera Drones Lecture (this lecture). In the use of this laboratory as a small drone flight-test area, the usable space measures approximately 7x4.5x3m (LxWxH).

A.1 Motion tracking system – Optitrack – Flex 13 Cameras

Our motion tracking system consists of 20 Flex 13 cameras (Optitrack) [R1], 4 synch routers and a desktop-PC (WIN_10 operating system) which runs the Motive (Optitrack) sofware. The synch routers ensure the images acquired by the cameras are time-synchronized; and they are connected to the WIN_10 PC through USB-2 ports. An additional desktop PC (Ubuntu 18.04 operating system) is used to run a software to stream the data from the motion tracking system to the robots in the room by means of the Robot Operating System (ROS) middleware software [R2].

The Flex 13 cameras emit infrared (IR) light (see the ring of LEDs around the Flex 13 camera lenses), which is reflected by spherical IR markers. These reflective markers can be detected, and its location estimated (at 120 Hz) inside the motion tracking volume when they are correctly detected by at least 3 cameras. Objects (named "Assets" in the Motive software) with a fixed constellation of IR Markers can be defined in the "Motive" Software, and their pose – translation and rotation – is then estimated by the software at 120 Hz. Note: The "Motive" software can also track a person and its limbs positions by using IR-Markers.

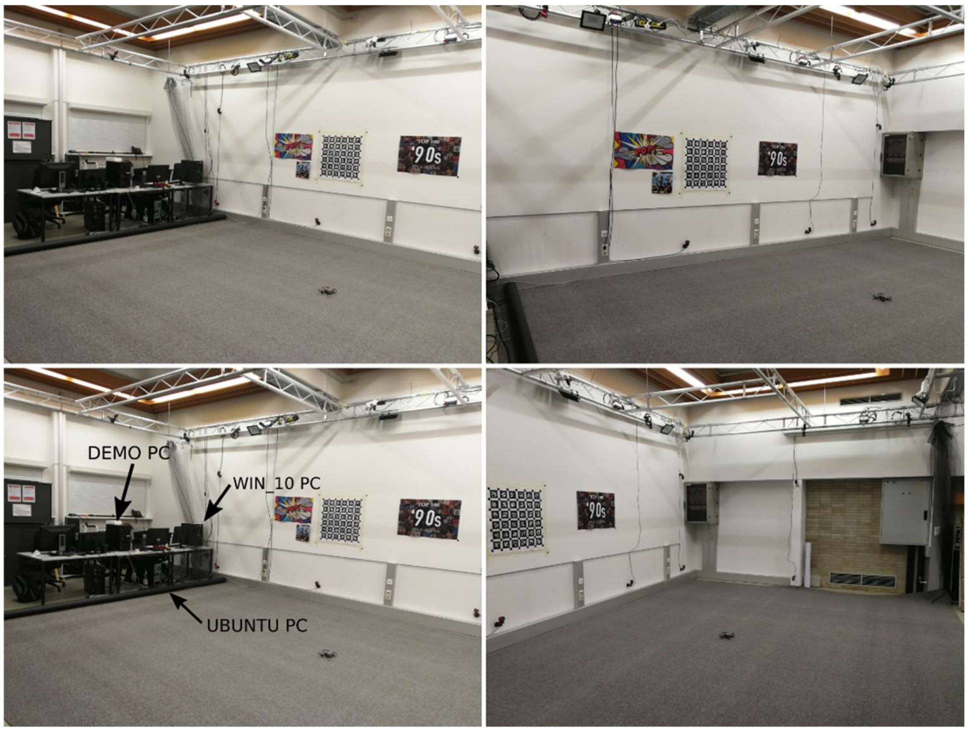

Fig. 2. Pictures of the Dronespace of the ICG - TU Graz.

A.2 Images of the Dronespace

The following images, see Fig. 2, show the Dronespace. As it can be seen most of the cameras are arranged on a metallic structure above the tracking volume. There are some cameras mounted along the walls at a middle height and near the ground. A safety net can be attached to the border of the flying area, which is very useful for working with students and during public demonstrations. The desktop-PCs that are connected to the motion tracking system are located behind the safety net.

Tips

The visualization in RViz and the print-out of data are very useful to be able to evaluate the quality of the obtained results.

Take screenshots of the visualization and write a small report for yourself.

Grading Scheme, assignment 4 score: 10 out of 60 points

The submission needs to include a small report – PowerPoint or PDF slides – about your work on this assignment and a bagfile with your trajectories. No source-code needs to be submitted for this assignment.

Most of the score obtained for this assignment (75%) is decided from the following content:

- Image(s) showing your trajectories (similar to those in Fig. 1)
- All 4 smooth trajectories, 4 shortened paths and 4 raw traj. planner paths are provided.
- (Alternatively) Only 4 shortened paths and/or 4 raw traj. planner paths are provided.

To obtain the rest of your score (25%):

- Provide videos (screen recordings) with the playback of your 4 smooth trajectories, which demonstrate that the trajectories are smooth and with no issues.

Software packages

See previous practicals.

Videos

References

[R1] Optitrack – Flex 13 camera: https://optitrack.com/cameras/flex-13/

[R2] Robot Operating System (ROS): https://www.ros.org/

[R3] Sampling the reference trajectory in time: https://github.com/ethz-asl/mav_trajectory_generation#sampling-trajectories

Otherwise, see previous practicals.