Vision-Based Navigation
J. Stückler, V. Usenko, L. von Stumberg
Winter Semester 2017/2018

Computer Vision Group
Department of Informatics
Technical University of Munich

Exercise Sheet 3

Topic: Robot Control

Submission deadline: Tuesday, 21.11.2017, 23:59
Hand-in via email to visnav_ws2017@vision.in.tum.de

General Notice

The exercises should be done in teams of two to three students. Each student in a team must be able to present the solution to the tutors during the exercise sessions on a lab PC in room 02.05.014. The presentations and solutions will be graded and will count for the final grade of the lab course. If you have not yet done so, please register yourself together with your team members on the team list in room 02.05.14.

We will use ROS Kinetic on Ubuntu 16.04 in this lab course. It is already installed on the lab computers. If you want to use your own laptop, you will need to install these versions of Ubuntu and ROS. Please read the ROS and OpenCV documentation for further reference.

Introduction

The goal of this exercise is to acquire practical experience with controlling a flying robot in simulator. You will write a PID controller to make the robot hover on spot and resist external disturbances.

Exercise 1 (12 points):

Download the code sample for this exercise provided on the course website:

- `https://vision.cs.tum.edu/teaching/ws2017/visnav_ws2017/slides`

To get you started, it contains a flying robot simulator and a solution skeleton code.

In this exercise you will implement a position PID controller for the flying robot assuming the ground truth pose of the robot's center of mass is available.

(a) Get familiar with simulator and skeleton code of the exercise. Launch the simulator by running the following command:

```
roslaunch euroc_simulation_server ex3_setting1.launch
```

Launch the exercise solution in a different terminal:

```
roslaunch ex3_solution ex3.launch
```

Verify that launching solution unpauses the simulator and it starts publishing messages. What messages does the simulator publish? What are the frequencies of these messages?

(b) Inspect uav_controller.hpp class in the solution source code. In the constructor this class initializes some constants, such as gravity, robot mass, noise characteristics of the sensor. After that it initializes publishers and subscribers and unpauses the simulator. The groundTruthPoseCallback function receives the ground truth pose from the simulator, computes velocity and stores pose and velocity in the class variables. The getPoseAndVelocity function returns current estimate of pose and velocity. This function will be used later to switch easily between ground truth measurements and filter output.

(c) Fill the computeDesiredForce function with the code to compute desired force using the PID controller for hovering at (0,0,1) with zero velocity. Desired force can be computed using the following formula:

$$\ddot{x} = k_p(x_d - x) + k_d(\dot{x_d} - \dot{x}) + k_i \int (x_d - x)dt,$$

where x_d and $\dot{x_d}$ are desired position and velocity, x and $\dot{x}$ are current position and velocity and k_p, k_d, k_i are proportional, differential and integral gains of the PID controller.

(d) The simulator provides a low-level controller for the robot that expects commands consisting of desired roll, pitch angles, thrust value and yaw rate, and executes a motor controller to maintain these desired values. Fill the computeCommandFromForce function with the code to compute desired roll, pitch angles and thrust. For this exercise you can set the yaw rate to zero. To obtain roll, pitch and thrust you can use the following equations:

$$\begin{aligned}
\phi_d &= \frac{1}{g}(\ddot{x}_1 \sin\psi - \ddot{x}_2 \cos\psi), \\
\theta_d &= \frac{1}{g}(\ddot{x}_1 \cos\psi + \ddot{x}_2 \sin\psi), \\
T_d &= \ddot{x}_3 + mg,
\end{aligned}$$

where ϕ_d is the desired roll angle, θ_d the desired pitch angle, T_d the desired thrust and ψ the current yaw angle.

(e) Write a sendControlSignal function that will obtain a current pose and velocity estimate from getPoseAndVelocity function, compute desired force with computeDesiredForce function, transform it into the message with computeCommandFromForce and publish it.

(f) Publish the control message with the rate of the most high frequency sensor. You can, for example, call the sendControlSignal in the end of IMU callback.

(g) Test the controller in different settings (ex3_setting1.launch, ex3_setting2.launch, ex3_setting3.launch). In the first setting no external forces are applied to the robot, in setting 2 at 40 s of the simulation a constant wind starts blowing and in setting 3 the wind gust blows just for several seconds and then stops. Tune the PID controller such that the robot maintains a stable flight in all those settings.

Submission instructions

A complete submission consists both of a PDF file with the solutions/answers to the questions on the exercise sheet and a ZIP file containing the source code that you used to solve the given problems. Note all names and matriculation numbers of your team members in the PDF file. Make sure that your ZIP file contains all files necessary to compile and run your code, but it should not contain any build files or binaries. Please submit your solution via email to `visnav_ws2017@vision.in.tum.de`.