Visual Navigation for Flying Robots
D. Cremers, J. Sturm, J. Engel, C. Kerl
Summer Term 2013

Computer Vision Group
Department of Informatics
Technical University of Munich

Sheet 1

Topic: Motion Models and Robot Odometry

Submission deadline: Tue, 30.04.2013, 23:59
Hand-in via email to visnav2013@vision.in.tum.de

General Notice

To be admitted to the final exam, every student has to submit solutions to all three exercise sheets. All exercises should be done in teams of two to three students. If you have not yet done so, please register yourself together with your team members on the team list in room 02.05.14.

Introduction

The goal of this exercise is to familiarize yourself with the concept of the motion model of a quadrocopter. In the theoretical part of the exercise, you will have to define the state space and motion model of an AR.Drone quadrocopter. In the practical part, you will have to apply these models to compute the odometry and plot the trajectory taken by the robot.

Exercise 1:

In this exercise, we will model the locomotion of a quadrocopter. We consider the quadrocopter as a rigid body floating in 3D space. Note that there are different representations of 3D poses. We recommend to you to keep it simple (minimal) here, but alternative ways are possible (and absolutely valid). The quadrocopter uses its down-facing camera and a ultrasound distance sensor to determine its motion, which provides us with the forward and lateral speed and the distance to the ground. Furthermore, the inertial measurement unit (consisting of an 3-axis accelerometer and a 3-axis gyroscope) provides the orientation of the quadrocopter. By integrating these measurements, it is possible to estimate the (absolute) position of the quadrocopter. This process is also called odometry, and the task of this exercise is to derive the odometry model for the AR.Drone.

(a) What is the state space of the quadrocopter? (recommendation: use a state space with six dimensions)

(b) Define the odometry vector of the (AR.Drone) quadrocopter. (hint: the odometry observation vector of the AR.Drone has six dimensions).

(c) Specify the odometry model f, i.e., the function that computes the current state $\mathbf{x}_t$ from the previous state $\mathbf{x}_{t-1}$ and the odometry observation $\mathbf{u}_t$:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t) \tag{1}$$

Exercise 2:

In this exercise, we will implement the above model in a ROS C++ node and evaluate it on several bag files. Both the bag files and a stub of the ROS node are available from the course website.

(a) Download the three bag files provided for this sheet from `http://vision.in.tum.de/teaching/ss2013/visnav2013`

(b) Replay the bag files and inspect the topics. (`rosbag play`, `rostopic list`, `rostopic echo`). Which topics are there? What data does the `/cmd_vel` and the `/ardrone/navdata` topics contain?

(c) Use `rxplot` to visualize vx and vy from `/cmd_vel` and from `/ardrone/navdata`. What is the relation between `/cmd_vel/linear/x` and `/ardrone/navdata/vx`? Estimate (roughly) the delay between steering commands and the onset of the motion.

(d) Visualize the camera images using `RVIZ`.

(e) Download the ROS node stub from GitHub into your ROS workspace, e.g. `~/fuerte_workspace`, using

```
$ cd ~/fuerte_workspace
$ git clone git://github.com/tum-vision/
   visnav2013_exercise.git
```

(f) Extend the `onNavdata` method in `src/main.cpp` to print the x and y velocities, yaw angle, and height to the screen. In which coordinate system are these values given? What are their units?

(g) Implement the pose update in the `onNavdata` method according to your answers from Exercise 1.

(h) The provided code publishes the trajectory taken by the quadrocopter using `visualization_msgs/MarkerArray` messages. Visualize these messages in `RVIZ`. Take a screen shot showing the trajectory of the `square.bag` sequence and attach it to your report.

(i) Compute the distance traveled by the quadrocopter for the `flight_manual.bag` sequence and the mean height for `flight_z.bag`.

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 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 `visnav2013@vision.in.tum.de`.