Vision-Based Navigation
D. Cremers, J. Stückler, V. Usenko, J. Engel
Summer Term 2015

Computer Vision Group
Department of Informatics
Technical University of Munich

Exercise Sheet 2

Topic: Visual Motion Estimation Using Keypoints

Submission deadline: Monday, 04.05.2015, 12:00 am
Hand-in via email to visnav15@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 Indigo on Ubuntu 14.04 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 a visual motion estimation method for RGB-D images using keypoints.

Exercise 1:

Download the following bag file for use in this exercise:

- `https://vision.in.tum.de/rgbd/dataset/freiburg2/rgbd_dataset_freiburg2_desk.bag`

(a) Write a C++ node that reads in the RGB and depth images from the bag file.

(b) Use the ORB keypoint detector and descriptor implemented in OpenCV (`http://docs.opencv.org/modules/features2d/doc/feature_detection_and_description.html`) to detect keypoints in the RGB images. Also compute the ORB descriptor for each keypoint. Draw the keypoints using the OpenCV drawKeypoints method.

(c) Match the ORB keypoints in subsequent frames $(t, t+1)$ using brute force matching implemented in OpenCV. Note that the ORB descriptor is binary,

thus use the Hamming distance measure. Draw the matches using the OpenCV drawMatches method.

(d) Determine the fundamental matrix from the keypoint matches using the 8-point RANSAC algorithm implemented in OpenCV. The function also retrieves the set of inliers. Visualize the inlier keypoints. What are your observations?

(e) Estimate the camera motion (3D rotation and translation) from the essential matrix (determined from the fundamental matrix).

(f) Triangulate the 3D points using the found camera motion using the OpenCV method 'triangulatePoints'. Rescale the camera motion such that the triangulated mean depth of the 3D points in frame t matches the measured mean depth of the keypoints in the corresponding depth image of frame t. Display the triangulated points and their measured 3D position (from 2D pixel coordinate and depth) in RViz, also draw lines between corresponding pairs of triangulated and measured point. Determine the mean deviation in meters of the triangulated from the measured points.

(g) Compute the rescaled camera motion between all pairs of successive frames in the dataset.

(h) Compare your motion estimates with the ground truth using the evaluate_rpe and evaluate_ate tools of the RGB-D benchmark dataset. For the evaluate_rpe tool, use the unit of 'frames' and a fixed difference of 1 frame to calculate the relative poses. What is the RMSE error in RPE and ATE of your method? Plot the ground truth trajectory and your result using the evaluate_ate tool. Also plot the RPE per frame using the evaluate_rpe tool. Does the accuracy increase if you drop frames for the processing? Evaluate ATE and RPE for different number of frame drops (1,2,4,8,16,32).

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 `visnav15@vision.in.tum.de`.