Assignment 1

Robot Vision practical

Friedrich Fraundorfer (fraundorfer@icg.tugraz.at)

Submission: Submit the completed assignment (code, output data and report) using the TeachCenter submission system as a single ZIP File. Documents need to be submitted using the PDF format.
Use the following naming scheme for the main source files: “group??_assgn1_task?.cpp”.
Use the following naming scheme for the report: “group??_assgn1_reptask?.pdf”.

Deadline: April 27, 2020, 23:55h
Questions: If you have any questions about the exercises send a mail to chetan.kumar@student.tugraz.at.

Task 1: Camera calibration (10pts)

This task consist of writing a program for computing the individual camera calibration parameters from the provided image sequence of a calibration pattern (**CALIB_DATA.ZIP**). The following parameters need to be estimated and reported for each camera:

- fx, fy, cx, cy
- k1, k2, p1, p2, k3

Deliverables: Document the estimated camera parameters in the report. Document also a quality report of the calibration (overall RMS re-projection error). No file output is required.

Hint: Follow the tutorial at https://docs.opencv.org/master/d4/d94/tutorial_camera_calibration.html. It is not necessary to write totally new code. You can start with the provided example of OpenCV and modify this to your needs. Important, the estimated camera parameters are needed in Task 2.

The physical size of a black (white) patch in the calibration image is 30x30mm and the pattern has 8x5 patch intersection.

Task 2: Stereo calibration (10pts)

This task consists of writing a program for calibrating a stereo setup from the provided images of a calibration pattern (**CALIB_DATA.ZIP**). The following parameters need to be estimated for the stereo setup:

- R … Output rotation matrix between the 1st and the 2nd camera coordinate system
- t … Output translation vector between the coordinate systems of the cameras
- e … RMS of re-projection error (as quality measure)

Deliverables: Document the estimated calibration parameters in the report. No file output is required.

Hint: Follow the example stereo_calib.cpp from the OpenCV installation. It is not necessary to write totally new code. You can start with the provided example of OpenCV and modify this to your needs. Important, the estimated calibration parameters are needed in Task 3.

Task 3: Stereo matching (10pts)

This task consists of writing a program for performing stereo matching. The program should take as input two images (from **STEREO_DATA.ZIP**), intrinsic calibration (from Task 1) and extrinsic calibration (from Task 2) and perform stereo matching. The program should output a disparity image and write out the depth data as colored 3D point cloud in the PLY format. The output should be generated with the SGBM method. Proper parameters for stereo matching should be chosen, such that a meaningful result is obtained.

Deliverables: Put the disparity image, the rectified input images and a screenshot of the visualized point cloud (e..g Meshlab) into the report. Three file outputs are required. The disparity image as PNG, the rectified image pair as PNG and the colored 3D point cloud as PLY.

Hint: Follow the example stereo_match.cpp from the OpenCV installation. It is not necessary to write totally new code. You can start with the provided example of OpenCV and modify this to your needs.

[Optional: Process also the additional stereo pairs from **STEREO_DATA_ADD.ZIP**.]

Task 4: Measuring in-image object dimensions using depth data. (5 pts)

Utilizing the depth data that has been obtained from the previous task, please estimate the dimensions (width, height and length) of the "Stochastic Process and Filtering Theory" book seen in the provided stereo image data. The output can be a number in meters (please specify if you use other units).

Deliverables: The dimensions must be mentioned in the report. Please mark the points you have used to measure the dimensions in both the RGB and depth images.

Other information:

The tasks have to be solved using the OpenCV library. It is advised to use OpenCV 4.2.

Important: Make sure to install opencv-contrib.

https://docs.opencv.org/master/d7/d9f/tutorial_linux_install.html

OpenCV works fine under Ubuntu run in Virtual Box (https://www.virtualbox.org/).

OpenCV can be downloaded from:

https://github.com/opencv

Tutorials can be found at:

https://docs.opencv.org/master/d9/df8/tutorial_root.html