Robotic 3D Vision
Prof. Dr. Jörg Stückler, Rui Wang
Winter Semester 2017/2018

Computer Vision Group
Department of Informatics
Technical University of Munich

# Exercise Sheet 3

Topics: Keypoint Detection and Matching, Direct Visual Odometry

Submission deadline: Wednesday, 13.12.2017, 23:59
Hand-in via email to rob3dvis-ws17@vision.in.tum.de

### General Notice

All exercises can be done in teams of up to three students. Please hand-in your solution before the submission deadline, indicating names and matriculation numbers of your team members. Teams are encouraged to present their submitted solution during the exercise sessions.

### Exercise 3.1: Keypoint Detection and Matching for Motion Estimation

In this exercise, you will implement and analyse keypoint detection and matching algorithms to determine point correspondences for indirect motion estimation between pairs of images.

(a) Extract the image files `0005.png` and `0007.png` from folder data/fountain in the exercise archive. The intrinsic camera calibration parameters for both images are provided in the file
`camera_calibration.txt` as the camera intrinsics matrix,

$$\mathbf{C} = \begin{pmatrix} 2759.48 & 0 & 1520.69 \\ 0 & 2764.16 & 1006.81 \\ 0 & 0 & 1 \end{pmatrix}. \tag{1}$$

(b) Find corresponding point pairs using keypoint detection, description and matching. Detect FAST and SURF keypoints using the functions `detectFASTFeatures` and `detectSURFFeatures` functions in matlab. Use the default settings for both keypoint detectors. Compute BRISK and SURF descriptors for the detected keypoints using the `extractFeatures` function and match the keypoints using the second best ratio distance from the lecture. For computing descriptors at SURF keypoints, you should take the scale and rotation of the keypoints into account using the standard settings of SURF features. For FAST you should use the scale of the detector as descriptor scale. Visualize your matching results for the detector-descriptor combinations FAST-BRISK, FAST-SURF, SURF-BRISK, SURF-SURF.

(c) Implement RANSAC to find the 2D-to-2D motion estimate using the eight-point algorithm from the lecture for a success probability of 0.99%, an outlier ratio of 0.2% and SURF detector and descriptor. What is the number of required iterations $N$ for RANSAC with this setting? Determine inliers using a threshold of 10 pixels on the reprojection error. What is the number of inliers and the average reprojection error of the final inlier set? What is the run-time of your algorithm? Compare your results for different detector-descriptor combinations (FAST-BRISK, FAST-SURF, SURF-BRISK, SURF-SURF), outlier ratios (0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 0.9%), and reprojection error thresholds (1, 2, 5, 10, 20, 50 pixels).

**Exercise 3.2: Direct Image Alignment**

In this exercise, you will implement direct image alignment of RGB-D images to estimate the camera motion between the images.

(a) Extract the exercise archive to obtain the provided data files. The archive contains RGB and depth images in the data folders rgbd/rgb and rgbd/depth. The file names of the images specify the recording timestamps in seconds. In the following, associate the RGB with depth images by the closest timestamp. The file formats are described here: https://vision.in.tum.de/data/datasets/rgbd-dataset/file_formats

The RGB image timestamps of each subsequent pair are

$$\begin{aligned} P_1 &= (1305031102.175304, 1305031102.275326) \\ P_2 &= (1341847980.722988, 1341847982.998783) \end{aligned} \tag{2}$$

The corresponding camera intrinsics matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ of the RGB image pairs are:

$$\mathbf{C}_1 = \begin{pmatrix} 517.3 & 0 & 318.6 \\ 0 & 516.5 & 255.3 \\ 0 & 0 & 1 \end{pmatrix}, \ \mathbf{C}_2 = \begin{pmatrix} 535.4 & 0 & 320.1 \\ 0 & 539.2 & 247.6 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

Note: Convert the RGB images to floating point grayscale images before processing them. The depth images represent depth values by 16-bit integer values and need to be scaled by a factor of 1/5000 to obtain metric depth. Convert the depth images to floating point metric values before further processing them.

Use the provided `downscale` function to downsample the images 4 times with sampling factor 2. Display the images in original resolution and their downsampled versions.

(b) Implement the `se3Exp` and `se3Log` functions in the respective scripts. Hint: matlab provides an implementation of the matrix exponential and logarithm.

(c) Implement the `calcResidual` function in the respective script such that it determines the photometric residual between the two images on the original resolution. The function should return the residuals in a vector. Display the residual image.

(d) Use the `calcResidual` function to implement numeric differentiation of the direct image alignment residuals for a left-multiplied pose increment. Use twist coordinates to represent pose and apply small pose increments ($10^{-6}$) on each twist coordinate individually to determine the numeric derivatives. Implement the function in the `deriveResidualsNumeric.m` script.

(e) Implement the Gauss-Newton step in the `doAlignment.m` script and a suitable stopping criterion for the Gauss-Newton iterations. Finalize the script to determine the relative camera motion for the two image pairs $P_1$ and $P_2$ through direct image alignment using the numeric derivative of the residuals. For pair $P_1$, the resulting pose should be close to

$$\boldsymbol{\xi}_1 = (-0.0018, 0.0065, 0.0369, -0.0287, -0.0184, -0.0004)^\top \tag{4}$$

For pair $P_2$ it should be close to

$$\boldsymbol{\xi}_2 = (0.2979, -0.0106, 0.0452, -0.0041, -0.0993, -0.0421)^\top . \tag{5}$$

Compare your results with the ground truth and discuss your results.

(f) Implement the analytic derivative of the residuals for the left-multiplied pose increment (in script `deriveResidualsAnalytic.m`). Use twist coordinates again to represent the pose. Run Gauss-Newton iterations with analytic derivatives using the `doAlignment.m` script. What are your observations wrt. accuracy and run-time in comparison to numeric differentiation?

(g) Use iteratively reweighted least squares to implement the Huber norm

$$\|r\|_\delta = \begin{cases} \frac{1}{2}\|r\|_2^2 & , \text{if } \|r\|_2 \leq \delta \\ \delta\left(\|r\|_1 - \frac{1}{2}\delta\right) & , \text{otherwise} \end{cases} \tag{6}$$

on the residuals. First derive the weights for the Huber norm in each iteration. Test the weighting on the defective RGB-D image pair in the data archive and compare the result with the unweighted Gauss-Newton method when using analytic derivatives. Hint: Find weights for the $l_2$ residuals such that Gauss-Newton corresponds to optimizing an error function with the Huber-norm on the residuals.

**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 `rob3dvis-ws17@vision.in.tum.de`.