Exercise Sheet 3

Topic: Keypoint Detection and Matching, Direct Visual Odometry

Exercise 3.1: Keypoint Detection and Matching for Motion Estimation

In this exercise, you will implement and analyze 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,

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

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.
Set **MinContrast** = 0.08 for FAST and use default settings for SURF. Compute BRISK and SURF descriptors for the detected keypoints using the **extractFeatures** function and match the keypoints once using only SSD/Hamming distance and once using the Lowe's distance ratio with a threshold of 0.6.
Hint: Make use of the parameter **MaxRatio** of the function **matchFeature**.
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 (also standard setting). 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? 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).
Hint: Implementations of the eight point algorithm and for 3D point reconstruction are provided.

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 two 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

$$P_1 = (1305031102.175304;\ 1305031102.275326)$$
$$P_2 = (1341847980.722988;\ 1341847982.998783)$$

The corresponding camera intrinsics matrices C_1 and C_2 of the RGB image pairs are:

$$C_1 = \begin{pmatrix} 517.3 & 0 & 318.6 \\ 0 & 516.5 & 255.3 \\ 0 & 0 & 1 \end{pmatrix}, \quad C_2 = \begin{pmatrix} 535.4 & 0 & 320.1 \\ 0 & 539.2 & 247.6 \\ 0 & 0 & 1 \end{pmatrix}$$

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 the 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

$$\xi_1 = (-0.0018, 0.0065, 0.0369, -0.0287, -0.0184, -0.0004)$$

For pair P_2 it should be close to

$$\xi_2 = (0.2979, -0.0106, 0.0452, -0.0041, -0.0993, -0.0421)$$

Compare your results with the ground truth.

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 w.r.t. accuracy and run-time in comparison to numeric differentiation?

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

$$\|r\|_\delta = \begin{cases} 0.5r^2 & \text{for } |r| \leq \delta \\ \delta(|r| - 0.5\delta) & \text{otherwise} \end{cases}$$

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.