Exercise Sheet 6

Topic: Iterative Closes Point, Dense Stereo Reconstruction

Exercise 6.1: Iterative Closest Point Algorithm

In this exercise, you will implement the iterative closest point algorithm to align point clouds and to estimate the camera motion between the RGBD images.

a) Extract the exercise archive to obtain the provided data files. The archive contains RGB and depth images in the data folders **`tum_rgbd/rgb`** and **`tum_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:

$$P1 = (1305031102.175304;\ 1305031102.275326)$$
$$P2 = (1341847980.722988;\ 1341847982.998783)$$

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

$$\boldsymbol{C}_1 = \begin{pmatrix} 517.3 & 0 & 318.6 \\ 0 & 516.5 & 255.3 \\ 0 & 0 & 1 \end{pmatrix} \qquad \boldsymbol{C}_2 = \begin{pmatrix} 535.4 & 0 & 320.1 \\ 0 & 539.2 & 247.6 \\ 0 & 0 & 1 \end{pmatrix}$$

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.

b) Use the provided **`downscale`** function to downsample the images to 80×60 pixels resolution. Convert the images to 3D point clouds.

c) Implement the ICP algorithm by alternating the following two steps:
   1) Establish point correspondences by finding the nearest neighbor for each point in one image within the other image based on the Euclidean distance.
      Hint: For fast nearest neighbor search you can use **`KDTreeSearcher`**.

   2) Find the transformation between the point sets using Arun's method (see lecture).

d) Iterate the ICP steps until convergence, i.e. the relative error falls below a threshold or a maximum number of iterations is reached. Report your results for the 2 image pairs and compare with the results from Ex 3.2.

Exercise 6.2: Dense Stereo Reconstruction

In this exercise, you will implement a basic approach for dense stereo reconstruction.

a) Inspect the code and data provided for this exercise in the materials archive. The data are left and right rectified stereo images from the KITTI dataset, contained in the folders **`kitti/eft`** and **`kitti/right`**. The image file names give the number of the images in the sequence. Corresponding left and right images have the same number. The camera intrinsics are specified in the file **`K.txt`**.

   The provided code contains a file **`main.m`** which implements basic loading of images and camera intrinsics. It also sets up some parameters for the stereo reconstruction. It calls the functions **`calcDisparity`** and **`disparity2PointCloud`** to determine the disparity between images and visualize a 3D point cloud. Function prototypes for these functions are included in the files **`calcDisparity.m`** and **`disparity2PointCloud.m`**.

b) Implement the function **`calcDisparity`** to find the disparity for each pixel in the left image that minimizes the SAD or SSD patch comparison measures (see lecture). Visualize your disparity estimation result for one stereo image pair in the sequence as a color map.

   Hint: use the **`pdist2`** function. The KITTI images are already stereo rectified such that epipolar lines are horizontal on the same rows between the left and right images.

c) Filter outliers by rejecting ambiguous matches. To this end, reset all disparities to zero where there are at least two second best disparity candidates with a matching cost ratio smaller than 1.5. Also reset disparities to zero if the corresponding pixel is found at the border of the right image. Why?

d) Implement the **`disparity2PointCloud`** function. For this, you will need the camera intrinsics and the baseline (the latter is defined in **`main.m`**). Visualize your result for one stereo image pair in the sequence.