CS284: Simultaneous Localization and Mapping

Homework 1, Spring Semester 2019

- Contact: lkneip@shanghaitech.edu.cn
- TAs: hulan@shanghaitech.edu.cn, yuph@shanghaitech.edu.cn
- For general questions not directly related to the homework material, please first use Piazza.
- Please also consider the office hours of both the TAs and the professor.
- Submission deadline: 23:59, 31^{st} of March 2019, 50% penalty for up to 1 day delay.
- **Please reread our policies on academic integrity. It is a very serious matter, and any violations will be prosecuted to the fullest extend.**

Homework 1 consists of **implementing the ICP algorithm** and applying it to an open-source dataset used within the SLAM community for automatic algorithm benchmarking. The dataset is recorded by a Kinect camera, but only the depth images from which 3D point cloud measurements can be extracted are needed. Please download the dataset fr2/xyz from the following link:

https://vision.in.tum.de/data/datasets/rgbd-dataset/download#freiburg2_xyz

Download the sequence as a .tgz file. the depth images of the sequence are contained in the sub-directory *depth*.

Your task is to implement the ICP algorithm in order to perform simple registration between successive depth frames. The obtained relative pose estimations may then be concatenated in order to recompute the entire camera trajectory (see the simplified SLAM algorithm introduced during the initial lectures). The pose of the initial frame is to be set to identity rotation and zero translation, and you need to use only every 20th frame in the sequence (i.e. frame 1, 21, 41, etc.) in order to ensure sufficient displacements between successive frames and limit computational burden. The detailed steps are as follows:

- Convert the depth images to point clouds. The camera matrix is

$$\mathbf{K} = \begin{bmatrix} 525 & 0 & 319.5 \\ 0 & 525 & 239.5 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

  Using this camera matrix, each image point $\mathbf{x} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$ (homogeneous form) may be transformed into a 3D point using the equation

$$\mathbf{X} = d \cdot \mathbf{K}^{-1}\mathbf{x}, \tag{2}$$

  where d is the depth of the point (coordinate of the point along the principal axis). This depth is notably given by the value of the depth image at that particular image point.

Important: The depth images are 16-bit grayscale images. Because they can only store integer values, the value in the depth image is to be divided by 5000 in order to return the actual depth in metric unit. For convenience, a function to open and read the depth images and produce an Eigen Array with metric depth values is provided. It also checks for valid depth measurements, and sets the depth in the Array to -1 if the reading is invalid (Kinect cameras only measure depth between about 0.5 and 5 m).

- Filter the depth images and sub-sample them in order to speed up the computation. What could be a straightforward way to speed up the subsampling given that the points are coming in structured form (i.e. registered in a matrix where each element corresponds to 1 pixel).
- Implement the correspondence search. What could be a straightforward way to speed up the correspondence search given that the points are coming in structured form (i.e. registered in a matrix where each element corresponds to 1 pixel). Perform outlier filtering after each correspondence establishment.
- Calculate the rotation and translation that relates the two point clouds using Arun's method. Start with the point-to-point error.
- Iterate over the above steps each time taking the already identified relative transformation as an initial transformation to be applied before the correspondence search.
- Implement a simple surface normal extraction method, and apply it to the depth images. Use it in the ICP objective function to formulate the point-to-plane error.
- Propose suitable algorithm termination criteria. Verify the correctness of your result by comparing the recovered trajectory to the ground truth poses provided on the webpage.
- **Bonus question**: Knowing that the Kinect sensor also returns color for the various points, can you think of a method to improve the correspondence search.

Submission: Please submit a PDF report (not more than 3 pages!) along with the code to the TA email addresses before the deadline (put professor in CC), then arrange for a time with the TAs to demonstrate your code. **Your code and results should be the same with what you submitted in the email**. Your report must include:

- A description of your implementation. This should cover a basic description of the algorithm. It should also include the physical structure of the program (i.e. file description, dependencies), and instructions on how to compile and use it (i.e. how to interpret the result).
- Your observations on the performance of your ICP algorithm, both in terms of efficiency and in terms of accuracy (you may compare your results against the ones provided on the web page). Please compare the differences between the "point-to-point" and the "point-to-plane" metric, and list them in the report.

Please use "SLAM HW1 – Your name" as the email-header when sending the email. Please pack all the files into a zip file, the structure should be:

[Your name] folder
– Your name.pdf
– [code] folder