16-833: Robot Localization and Mapping, Spring 2021

Homework #4
Dense SLAM with Point-based Fusion

Due: Wednesday April 21, 11:59pm, 2021

Your homework should be submitted as **a typeset PDF file** with **a folder of Python code (and/or output results)**. Please fill in the 'TODO' sections in the provided python skeleton. If you have questions, please post them on Piazza or come to office hours. Do not post solutions or codes on Piazza. Discussion is allowed, but each student must write and submit his or her own solution. Note that you should list the name and Andrew ID of each student you have discussed with in the first page of your PDF file. You have a total of 3 late days, use them wisely. Good luck and have fun!

1 Overview

In this homework you will be implementing a 3D dense SLAM system using point-based fusion.

2 Iterative Closest Point (ICP) (50 points)

In general, as its name indicates, ICP runs iteratively. Here is the general flow:

1. Setup data correspondences between source and target point clouds given the current pose $[R^t \mid t^t]$ (usually initialized with identity).
2. Linearize the loss function at $[R^t \mid t^t]$ and solve for incremental update $[\delta R \mid \delta t]$.
3. Update by left multiplying the incremental transformation $[R^{t+1} \mid t^{t+1}] = [\delta R \mid \delta t][R^t \mid t^t]$.
4. Go back to step 1 until convergence.

You will be working on `icp.py` in this section.

2.1 Projective data association

Question (5 points): Suppose you have projected a point p to a vertex map and obtained the u, v coordinate with a depth d. The vertex map's height and width are H and W. Write down the conditions u, v, d must satisfy to setup a valid correspondence. Also implement the `TODO : first filter` section in function `find_projective_correspondence`.

The conditions are as following:
$0 \leq u < W$
$0 \leq v < H$
$0 \leq d$

After obtaining the correspondences q from the vertex map and the corresponding normal n_q in the normal map, you will need to additionally filter them by a distance threshold, so that $|p - q| < d_{thr}$.

Question (5 points): Why is this step necessary? Implement this filter in `TODO : second filter` section in function `find_projective_correspondence`.

This step is necessary because in the projection and first filter, we are using a quick heuristic and considering only the 2D mapping of the point and ignoring the depth from the camera center. Now with this shortlist of points, we calculate the distance between the two points to make sure they are close enough to be true correspondences. (In other words, when ignoring depth an entire line of points are mapped to the same 2D pixel, but obviously they are not all valid correspondences.)

2.2 Linearization

Question (15 points): Now reorganize the parameters and rewrite $r_i(\delta R, \delta t)$ in the form of

$$r_i(\alpha, \beta, \gamma, t_x, t_y, t_z) = A_i \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ t_x \\ t_y \\ t_z \end{bmatrix} + b_i,$$

where A_i is a 1×6 matrix and b_i is a scalar.

Note: You don't have to explicitly write down all the entries, and feel free to introduce intermediate variables. A cross-product operator may be useful here:

$$[w]_\times = \begin{bmatrix} 0 & -w_2 & w_1 \\ w_2 & 0 & -w_0 \\ -w_1 & w_0 & 0 \end{bmatrix}, w = [w_0, w_1, w_2]^\top \in \mathbb{R}^3.$$

After reorganizing, we get the following:
$A_i = n_{q_i}^T[-[p_i']_\times | I_3]$

$b_i = n_{q_i}^T(p_i' - q_i)$

2.3 Optimization

The aforementioned step in fact does the linearization. Now suppose we have collected n associated pairs, we move on to optimize

$$\sum_{i=1^n} r_i^2(\alpha, \beta, \gamma, t_x, t_y, t_z) = \sum_{i=1}^{n} \left\| A_i \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ t_x \\ t_y \\ t_z \end{bmatrix} + b_i \right\|^2. \tag{1}$$

Question (15 points): Write down the linear system that provides a closed form solution of $\alpha, \beta, \gamma, t_x, t_y, t_z$ in terms of A_i and b_i. You may either choose a QR formulation by expanding a matrix and filling in rows (resulting in a $n \times 6$ linear system), or a LU formulation by summing up n matrices (resulting in a 6×6 system). Implement `build_linear_system` and the corresponding `solve` with your derivation.

The linear system's A and b are vertically stacked A_i's and b_i's for the n pairs.

Question (10 points): Report your visualization before and after ICP with the default source and target (frame 10 and 50). Then, choose another more challenging source and target by yourself (e.g., frame 10 and 100) and report the visualization. Analyize the reason for its failure or success.

Note: Press P to take a screenshot.

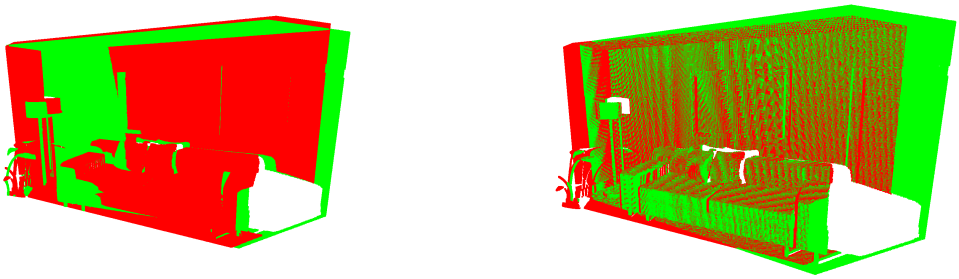

Figure 1: Point clouds before and after registration for frames 10 and 50

Figure 2: Point clouds before and after registration for frames 10 and 100

For frames 10 and 100, the icp algorithm implemented was not able to find the correct R and t. The reason for this is that we made the small-angle assumption and transformed R and t into δR and δt. Our initial guess of R and t being the identity matrix and no translation is off by too much such that the algorithm fails.

3 Point-based Fusion (40 points)

3.1 Filter

Similar to projective data association, we need a transformation. In a typical SLAM system, this is obtained from accumulative ICP. In this section, however, we assume ground truth transformations are known. The transformation $T_c^w = [R_c^w \mid t_c^w]$ is from the input **c**amera frame's coordinate system to the **w**orlds's coordinate system.

With the available transformation, you need to perform filtering similar to projective data association. The only difference is that you will need to add a normal angle constraint to be more strict with filtering.

Question (5 points): Implement `filter_pass1`, `filter_pass2` to obtain mask arrays before merging and adding input points.

See Code.

3.2 Merge

The merge operation updates existing points in the map by calculating a weight average on the desired properties.

Question (15 points): Given $p \in \mathbb{R}^3$ in the map coordinate system with a weight w and its corresponding point $q \in \mathbb{R}^3$ (read from the vertex map) in the frame's coordinate system with a weight 1, write down the weight average of the positions in terms of p, q, R_c^w, t_c^w, w. Similarly, write down the weight average of normals in terms of n_p, n_q, R_c^w, w. Implement the corresponding part in `merge`. Note a normalization of normals is required after weight average.

$p \leftarrow \frac{wp + R_c^w q + t_c^w}{w+1}$

$n_p \leftarrow \frac{wn_p + R_c^w n_q}{w+1}$
and then
$n_p \leftarrow \frac{n_p}{\|n_p\|}$

3.3 Addition

The projective data association may cover a big portion of the input vertex map, but still there are uncovered regions. These points have to be added to the map.

Question (10 points): Implement the corresponding part in `add`. You will need to select the unassociated points and concatenate the properties to the existing map.

See Code.

3.4 Results

Now run `python fusion.py /path/to/dataset`. The system will take the ground truth poses and produce the resulting image after fusing 200 frames.

Question (10 points): Report your visualization with a normal map, and the final number of points in the map. Estimate the compression ratio by comparing the number of points you obtain and the number of points if you naively concatenate all the input. Note to speed up we use a downsample factor 2 for all the input.

Note: Press N to visualize the normal map.

Figure 3: Fusion with ground truth poses with a normal map.

The number of points in the map after running 200 frames is 1,362,143. This is in comparison to the naive implementation where no merging or down-sampling occurs, equating to 15,360,000 points. Dividing the two, we only used around 8.87% of the space otherwise required.

4 The dense SLAM system (20 points + 10 points)

Now we put together both ICP and fusion code in `main.py`. We have called the relevant functions implemented by you for your convenience.

Question (5 points): Which is the source and which is the target, for ICP between the map and the input RGBD-frame? Can we swap their roles, why or why not?

For ICP, the map is used as the source and the next RGBD frame is the target.

Map points are projected onto the target frame, trimmed, and correspondences found. Interestingly, the map can be interpreted as using an RGBD camera that has identity intrinsic and extrinsic matrices to view all the points at the same time. By this analogy, the roles of the target can the source can be swapped for ICP, however, after projecting the target frame's points onto the global map frame, trimming is more complex, and if not done well the A and b matrices will be much larger than necessary, resulting in slow optimization solves.

Run `python main.py /path/to/dataset`. By default, 200 frames will be tracked and mapped.

Question (15 points): Report your visualization. Also, record the estimated trajectory and compare against the ground truth. Drift is acceptable in the case, to the extend shown in Fig. 5.

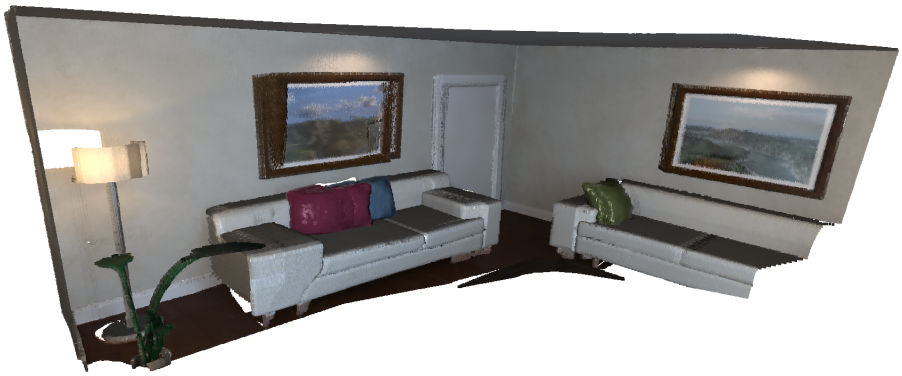

Figure 4: Fusion with poses estimated from frame-to-model ICP

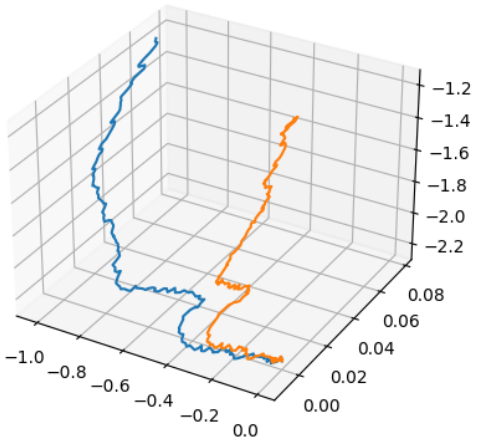

Figure 5: Drift Estimates

Bonus (10 points): Can you try to reduce the drift? There could be several methods, from improving the RGBD odometry to improving the fusion by rejecting outliers and eliminating dormant points (i.e., not associated to any points for a certain period). Report your updated visualization and trajectories.

5 Code Submission Rules

Upload all of your python code in a folder. Include all your visualizations in your PDF file. Do not upload the dataset when you submit.