Indoor 3D Mapping with RGB-D Camera

Yuwei Wang
yuwwang@seas.upenn.edu

***Abstract*—The goal of this project is to implement a full RGB-D camera-based SLAM system from the ground up. The implementation consists of mainly two parts - a front end that detects key point features, builds up a pose graph for camera poses and checks for loop closure; and a back-end that optimizes the pose graph resulting in a globally consistent map. The system is tested on the TUM SLAM benchmark dataset and results have shown that the system is both effective and robust in the mapping task. Code is available on Github**

I. Background

RGB-D cameras are attractive sensing systems that capture RGB images along with per-pixel depth information. They have the advantage of providing the scale information typically missing with monocular cameras, thus allowing us to construct dense point cloud of the environment. Similar to occupancy grid maps, it can be used for a good range of tasks in robotics like navigation and manipulation. Most mapping systems contain two main components - the spatial alignment of consecutive data frames; second, the detection of loop closures and optimization for a globally consistent structure. While 3D point clouds like LiDAR scans are quite well suited for frame-to-frame alignment using techniques like ICP scan matching [1], they do not take into account useful information embedded in images. Additionally, ICP-like algorithms require a good initial guess to avoid being stuck at local minimum. Color cameras, on the other hand, capture rich visual information that are well suited for feature tracking and loop closure detection, with the downside being not able to provide dense 3D information. In that regards, the RGB-D camera combines the strength of both sensors mentioned above. The obvious limitation is that it is only suitable for indoor tasks due to its limited range and sensitivity to the lighting condition, but for indoor environment mapping and reconstruction tasks it is an ideal sensor that allows capture of reasonably accurate mid-resolution depth and appearance information at high data rates.

Visual SLAM approaches simultaneously compute the robot's motion and the map using camera sensors. In contrast with liDAR-based SLAM, camera-based SLAM systems typically extract sparse keypoints from the camera images. Visual feature points have the advantage of being more distinctive than typical geometric structures, which makes data association easy with the use of feature descriptors. Common key point feature descriptors are SIFT [2] and ORB [3]. SIFT is widely considered as the gold standard in computer vision with highly desired properties such as scale and lighting invariance. ORB features, though less accurate than SIFT, has the attractive advantage of being highly speed and memory efficient, thus suitable for real-time applications [4].

Graph-based SLAM is becoming the most popular approach to SLAM problems over the recent years. To deal with the inherent uncertainty introduced in the sensor measuring process, the back-end of the SLAM system constructs a pose graph that represents spatial constraints between camera poses and points in the environment. The most likely trajectory is obtained by minimizing the inconsistency between estimates and measurements of the poses. With the known trajectory, we can project the sensor data into a common coordinate frame [5].

II. METHODOLOGY

A. Feature Extraction

The RGB camera data allows us to extract sparse visual features such SIFT, which can be used to associated with the same physical points observed in other frames. The depth camera gives the depth measurement that allows us to localize the feature points in 3D. The descriptors associated with the keypoints provide a way to measure similarity. To match a pair of the, one simply computes their distance in the descriptor space. For SIFT Euclidean distance is used. The matching procedure is as follows:

1) For a pair of RGB frames, extract SIFT key points and compute a descriptor (a vector of length 128) associated with each key point.
2) For each key point in frame 1, search for its two nearest neighbors(in descriptor space) in frame 2. A FLANN-based kd-tree is used for efficient search.
3) Use Lowe's ratio to filter out some bad matches. Specifically, keep a match only if the distance to its closest neighbor is much larger than to its second neighbor. In other words, only keeps a match when there is only one possible match in frame 2 to the query point in frame 1. This eliminates most of the ambiguous matches, thus

acting as an important first step in the outlier rejection process.

B. Frame-to-frame Transformation Estimation

The major task involved in building up the pose graph for SLAM is forming spatial constraint between a given pair of data frames. Given two data frames, we need to estimate the relative 3D transformation between the two in order to form an edge that can be added to the pose graph. The SIFT key point extracted in the previous steps give us the 3D correspondences in two frames which are needed for computing the transformation. The problem is formulated as follows:

- Given two point sets with known correspondences $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_N\} \quad P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_M\}$, estimate the rigid body transformation (R, t) that minimizes the sum of squared errors:

$$E(R, \boldsymbol{t}) = \sum_{(i,j)\in\mathcal{C}} \left\|\boldsymbol{q}_i - R\boldsymbol{p}_j - \boldsymbol{t}\right\|^2 \tag{1}$$

This is a well understood problem and can be solved in closed form with Singular Value Decomposition(SVD).The solution is given as follows:

1) First compute center of mass for each point set.

$$\boldsymbol{\mu}_Q = \frac{1}{|C|}\sum_{(i,j)\in\mathcal{C}} \boldsymbol{q}_i \quad \boldsymbol{\mu}_P = \frac{1}{|C|}\sum_{(i,j)\in\mathcal{C}} \boldsymbol{p}_j \tag{2}$$

2) Substract the corresponding center of mass from every point in the set.

$$\begin{aligned} Q' &= \left\{\boldsymbol{q}_i - \boldsymbol{\mu}_Q\right\} = \{\boldsymbol{q}'_i\} \\ P' &= \left\{\boldsymbol{p}_j - \boldsymbol{\mu}_P\right\} = \left\{\boldsymbol{p}'_j\right\} \end{aligned} \tag{3}$$

3) Compute the cross-covariance matrix of the shifted sets, and decompose the matrix using SVD.

$$W = \sum_{(i,j)\in C} \boldsymbol{q}'_i \boldsymbol{p}'^{\top}_j = UDV^{\top} \tag{4}$$

4) Finally, the transformation from one frame to the other is given by

$$\begin{aligned} R &= UV^{\top} \\ \boldsymbol{t} &= \boldsymbol{\mu}_Q - R\boldsymbol{\mu}_P \end{aligned} \tag{5}$$

C. 3 Point Estimation with RANSAC

Outliers are still likely to be present and a few badcorrespondences could heavily skew the transformation estimation. Therefore Random Sample Consensus (RANSAC) is incorporated to reject outliers. The above transformation estimation method is very attractive in the sense of RANSAC since only a minimumof 3 points is needed to compute one hypothesis for the transformation. This greatly reduces the number of sampling iteration required by RANSAC. The rule of thumb in determining number of iteration is derived from a probability perspective using:

$$k = \frac{\log(1-p)}{\log\left(1-w^n\right)} \tag{6}$$

where n is the number of samples needed by the model to compute a single hypothesis, w is the probability of choosing an inlier each time a single point is selected from the set, and p is the desired probability that the RANSAC algorithm provides a useful result computed with inliers only. Since n is only 3 in our case, the number of iteration needed to be 99% certain is less than 20, which greatly reduces the computation load commonly introduced by RANSAC. The full RANSAC procedure goes as follows:

1) Randomly sample 3 pairs of corresponding 3D key points and compute the relative transformation as a single hypothesis using the SVD method.
2) To evaluate this particular hypothesis, for each pair in all remaining corresponding pairs, apply the estimated transformation to the point in frame 1 and compute an error between the transformed point and its corresponding point in frame 2.
3) Count one correspondence as an inlier if the error is smaller than a threshold value. Record the total number of inliers for this hypothesis.
4) Iterate until maximum iteration set for RANSAC is reached. Select the hypothesis with the most inliers.

To obtain a statistically more stable solution, recompute the transformation with SVD using all inliers. The result will be used as the final estimated transformation between the two frames. Fig 1 shows keypoint matches before and after RANSAC. Fig 2 shows an example of aligning two frames' point clouds using the estimated transformation.

D. Refinement with ICP

So far we have aligned a given pair of frames using sparse feature points. Considering the fact that we have dense point clouds here, the Iterative Closest Point algorithm may be used to further refine the alignment by taking all the points in to account. ICP has been shown to be effective when the two clouds are already nearly aligned. Therefore we can use solution from the SVD method as an initial guess to ICP, and perform ICP to minimize the sum of squared distance between all point correspondences. The drawback of ICP is a significant slowdown in speed. During evaluation, no significant improvement was observed by applying ICP at the cost of processing time.

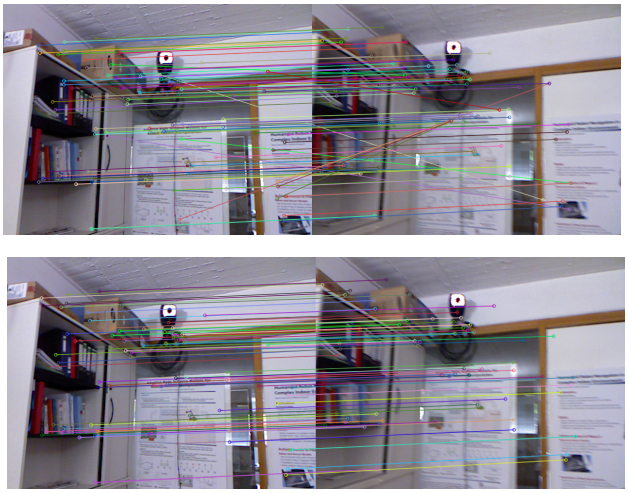


Fig. 1: Top: SIFT matches based only on distance in descriptor space. Bottom: filtered matches after applying RANSAC

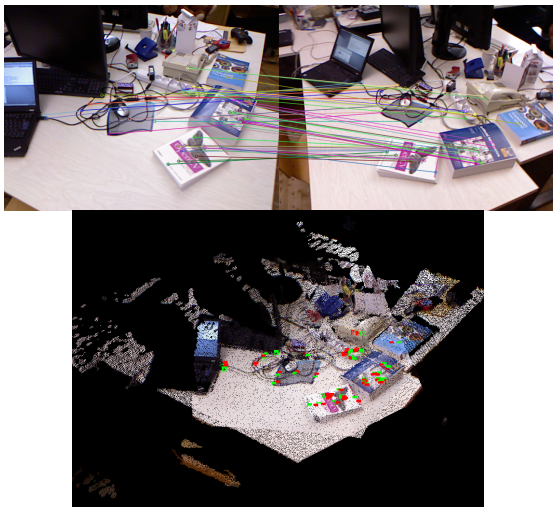


Fig. 2: Aligned point clouds of the two frames on top. Green dots are keypoints detected in image 1, red dots are keypoints detected in image 2

E. Further Filtering

The SVD method with RANSAC is sufficiently robust in outlier rejection in most cases. However, there are still corner cases in which it may fail. Fig 3 shows an example of such a case where both frames have similar scene with object with similar textures. During loop closure detection, pose constraint formed by linking two completely different locations in the environment like this can heavily distort the pose graph and thus the resulting map as well.

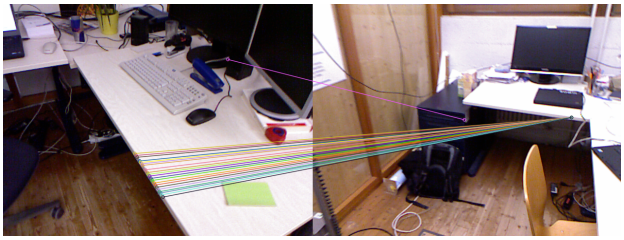


Fig. 3: A case where RANSAC falsely matches the edge of a desk to that of another similar desk in the enviroment. Such cases are effectively rejected by the mean distance-to-centroid method

To tackle this, we can exploit the fact that geometric property of a point set should be preserved after a rigid transformation. In other words, the mean distance to the point set's centroid should be roughly the same in both frames. The following simple metric implements this idea:

- Compute the point set's centroid and mean distance from each point to the centroid for both frames.
- Only if $0.9 < \frac{mean\ distance\ in\ frame1}{mean\ distance\ in\ frame2} < 1.1$, geometry of the point set is considered as being preserved and count the two frames as a match.

By doing so, the systems becomes highly robust in rejecting wrong matches especially in comparing non-neighboring frames, which is crucial to successful loop closure detection.

F. Pose Graph Optimization

Pose graph optimization is the gold standard today for most state-of-the-art SLAM systems. A pose graph consists of vertices and edges. Simply speaking, a vertex represents a pose at a given time in the trajectory, and an edge represents a constraint between any pair of poses. The edge usually is formed by a measurement or estimation of the relative transformation between the two poses as described earlier. The pairwise transformation estimates between sensor poses, as computed by the frontend, form sequential edges of a pose graph. However, estimation errors accumulate over time and leads to global inconsistency. To compute a globally consistent trajectory, the pose graph is optimized by minimizing the following Nonlinear Least Squares error:

$$\sum_{\langle i,j\rangle \in \mathcal{C}} \mathbf{e}\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}\right)^{\top} \mathbf{\Omega}_{ij} \mathbf{e}\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}\right) \tag{7}$$

where

$$\mathbf{e}\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}\right) = \mathrm{t2v}\left(\mathbf{Z}_{ij}^{-1}\left(\mathbf{X}_i^{-1}\mathbf{X}_j\right)\right) \tag{8}$$

$\mathbf{X} = \left(\mathbf{x}_1^{\top}, \ldots, \mathbf{x}_n^{\top}\right)^{\top}$ are the camera poses, $\boldsymbol{z}_{ij}$ represents a measurement constraint between any two poses, and $\mathbf{\Omega}_{ij}$ is the information matrix which is simply the inverse of the

measurement covariance. Note that uppercase $\mathbf{X}$ and $\mathbf{Z}$ are simply the homogeneous transformation representation of their lowercase pose vector.

At runtime, when a new pose(frame) comes in, add it to the pose graph as a new vertex, and add an edge to its immediate predecessor as well as to a few other preceding frames to which a transformation can be estimated. These edges are sequentially edges. In constrast, non-neighboring edges can also be formed in the form of loop closure detection which will be explained in the next section. The pose graph is optimized using the g2o framework [6], which iteratively solves the above Nonlinear Least Squares problem with Levenburg-Marquart method.

G. Loop Closure with Keyframes

Loop closure is an essential part to most SLAM systems in order to achieve global consistency. It works by forming edges/constraints between non-neighboring poses when revisiting known parts of the map and thus diminishing the accumulated error. Loop closures are represented as constraints between frames that are not temporally adjacent.

Searching through all previous frames to find a potential match to the current frame would be computationally demanding, therefore sampling a few "keyframes" at each timestep is a good workaround. Keyframes are a subset of all previous frames, and they can be determined based on visual overlap, adapting the density of keyframes to camera motion and local appearance. Each time a new frame comes in, we align it to the most recent keyframe using the same RANSAC procedure described earlier and see if there are enough inlier corresponding points between the two frames.

At each time step, randomly sample a few keyframes from all previously recorded keyframes, and perform the RANSAC procedure to match the current frame to the sampled keyframes. If a match is successful, add an edge between the two to the pose graph.

III. Results

A. Local Map

The system is evaluated on the publicly available TUM benchmark RGB-D dataset for SLAM [7]. The data used in this experiment consists of a video sequence of RGB and depth frames recorded in a feature-rich room. Fig 4 shows a constructed map for a local region in the environment. As can be seen, incremental pairwise alignment can be fairly accurate for creating local small maps even without graph optimization, so long as the sequence remains short.

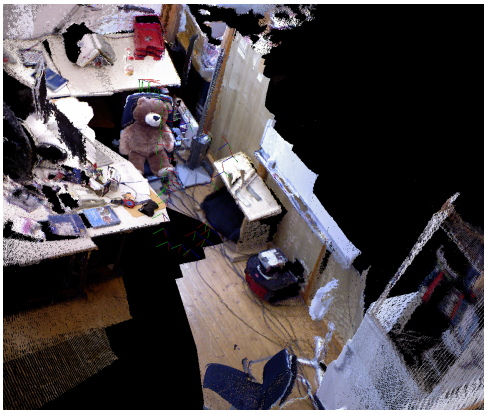
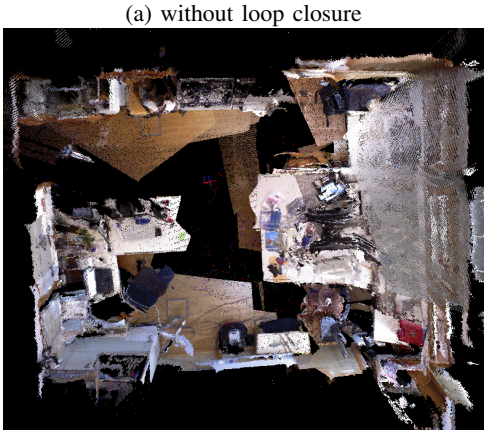


Fig. 4: Map constructed for a local region. Camera trajectory is also shown in the figure

B. Global Map

Global map is constructed by running SLAM on the full data sequence. To prevent memory runout, the map is downsampled using a voxel map with grid size of 1cm. Fig 5 shows a comparison between map built with and without loop closure. Fig 6 shows the full map viewed from a few different angles.

(a) without loop closure



(b) with loop closure

Fig. 5: Top-down view camparison

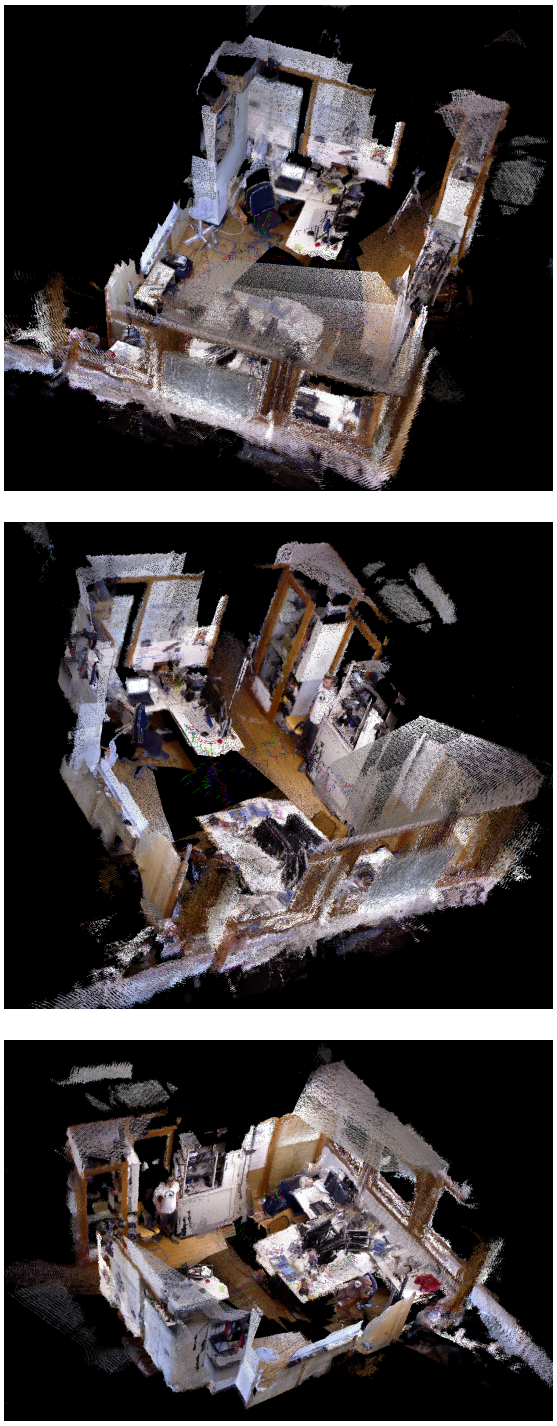


Fig. 6: Full constructed map

IV. Conclusions

A full RGB-D SLAM system is implemented in this project with SIFT feature matching, pairwise pose estimation with RANSAC and loop closure with graph optimization. The systems has demonstrated good effectiveness and robustness in efficiently building a consistent map in a indoor setting. Future work includes the following:

- experimenting with ORB features to achieve real-time performance
- include a relocalization mechanism to recover from a localization failue or frame loss.
- Implement more efficient search for loop closure frames by leveraging uncertainty information in the pose graph.

References

[1] P. J. Besl and H. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.

[2] D. Lowe, "Discriminative image features from scale-invariant keypoints," *s. International Journal of Computer Vision*, 2004.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," *Proc. of the IEEE Intl. Conf. on Computer Vision*, 2011.

[4] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments," *Experimental Robotics*, 2011.

[5] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3d mapping with an rgb-d camera," *IEEE TRANSACTIONS ON ROBOTICS*, 2012.

[6] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2011.

[7] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," *Proc. of the International Conference on Intelligent Robot Systems*, 2012.