Paper Seminar Presentation

An Evaluation of the RGB-D SLAM System

Endres, Felix & Hess, Jurgen & Engelhard, Nikolas & Sturm, Jurgen & Cremers, Daniel & Burgard, Wolfram.

Project 15: 3D Reconstruction of Infants' Cranial Shape Using Mobile Devices

Tara Tang
Biomedical Engineering
ttang14@jhu.edu

Table of Contents

1. Project Summary

Infant cranial deformities such as deformational plagiocephaly/brachycephaly (DPB) and craniosynostosis are becoming a pediatric epidemic. Pediatricians currently do not have a reliable and accurate method for early detection of these deformities, before they require surgery and expensive treatments.

Our project goal is to create a software pipeline that can reconstruct an accurate 3D model of a baby's head from depth information. Our general approach is to take RGB and depth images of a baby's head from several different viewpoints. The registration transformation between each of these frames can be found, allowing us to combine all of the viewpoints into a unified 3D model.

2. Paper Overview

The paper I have chosen to review is titled "An Evaluation of the RGB-D SLAM System," by Endres, et al., published in 2012. It describes a novel approach to the SLAM problem that combines RGB and depth (D) information. Their algorithm employs feature descriptors and a pose graph (with global optimization) to register different frames to each other. They use three feature descriptors: SIFT, SURF, and ORB.

In general, this paper is extremely relevant to our project. It outlines a basic approach that we have been loosely trying to follow and adapt to our own project. We have frequently referenced this paper throughout the semester.

3. Problem Background

The field of computer vision and 3D reconstruction is very commonly faced with a fundamental chicken-and-egg problem: a camera's pose must be known in order to generate a model of the world/environment. However, the world model must be known in order to localize the camera's pose in any meaningful way. The solution to this problem is referred to as SLAM, which stands for Simultaneous Localization And Mapping. In other words, both the camera pose and world model must be estimated simultaneously.

Sensors like the Microsoft Kinect, which were quite new when this paper was published in 2012, have the ability to capture both color (RGB) images and a corresponding dense depth (D) map at a high frame rate and accuracy. Integrating these two modes of information offers a novel approach to SLAM.

4. Approach Overview

The paper provides a simple schematic of their overall approach (Fig.1).

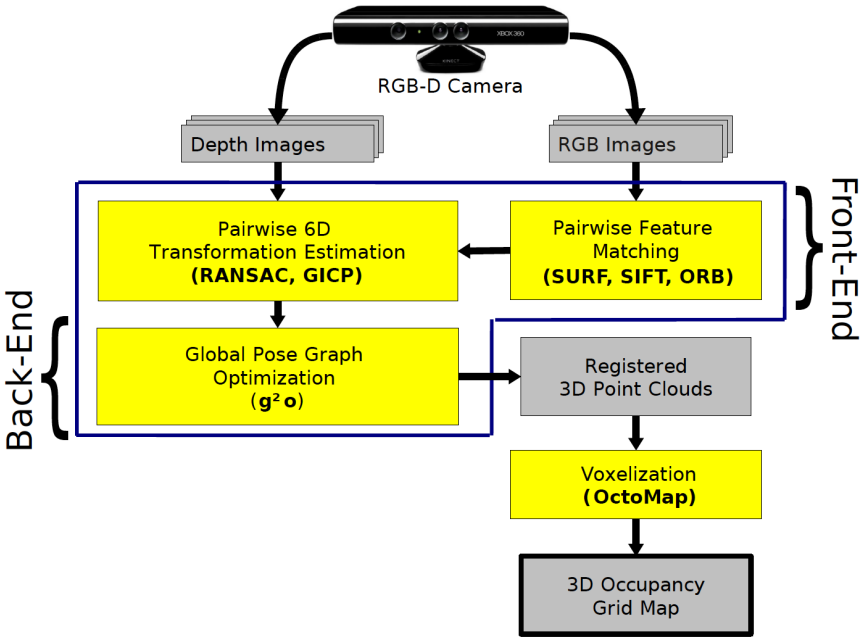

Fig.1: Schematic of Overall Approach

The approach can be broken down into six steps, detailed below.

4.1. Feature Detection and Mapping

In the first step, a feature detection and matching algorithm is implemented using OpenCV. The paper chooses to compare the performances of three feature detection algorithms: SIFT, SURF, and ORB.

SIFT (Scale Invariant Feature Transform) is a benchmark feature detection algorithm first published by Lowe in 1999 [2]. Significantly, it is invariant to scale, unlike previous feature detectors like the Harris corner detector or the Canny edge detector. Most currently competitive feature detection methods are derived from SIFT.

SURF (Speeded Up Robust Features) is a faster, more robust version of SIFT [3]. ORB (Oriented FAST and Rotated BRIEF) was, at the time of publication, a very new feature detection algorithm designed to be a faster alternative to both SIFT and SURF. ORB uses FAST (Features from Accelerated Segment Test) feature detection and BRIEF (Binary Robust Independent Elementary Features) feature descriptors.

Any of these feature detection algorithms can be used to detect features (also known as keypoints) in the RGB images. Then, corresponding keypoints in two RGB images can be matched based on how similar their keypoint descriptors are. The matching process is achieved either with brute force or FLANN (Fast Library of Approximate Nearest Neighbors), but this comparison is not the focus of this paper.

4.2. Project to 3D Point Clouds

Once feature keypoints have been located in the 2D RGB images, they can be projected to 3D space using the depth measurement at the center of the corresponding keypoint. This projection is a very common computer vision operation that requires knowledge of the depth camera's intrinsic parameters. These parameters describe how a camera's coordinates relate to the coordinates in the image [5].

If we disregard skew (as is commonly done), there are four intrinsic parameters: focal length in *x* and *y* (f_x, f_y) and the optical center coordinates (c_x, c_y). We denote a 2D image pixel's coordinates as (u, v). The *z*-coordinate in 3D space is the depth measurement (sometimes scaled). Thus, we can solve for the *x*- and *y*-coordinates in 3D space using the following equations (Fig.2):

$$\begin{aligned} \text{focal length} &= (f_x, f_y) \\ \text{optical centers} &= (c_x, c_y) \\ \text{image pixel} &= (u, v) \\ \text{depth} &= z \\ x &= \frac{(u - c_x)z}{f_x} \\ y &= \frac{(v - c_y)z}{f_y} \end{aligned}$$

Fig.2: Project 2D image coordinates to 3D points using depth and camera intrinsics parameters.

4.3. Estimate Transformations

In general, the detection of visual features is prone to false positives, inconsistencies with the corresponding depth image (due to the distinction between the RGB and depth cameras), projection errors at the borders of the image, etc. Thus, the paper authors chose to cope with noisy data using RANSAC, which is short for Random Sample Consensus [6].

In the RANSAC pipeline, three pairs of feature matches are randomly chosen and the rigid transformation between the two frames is calculated. Then, inliers are counted based on the user-defined requirement that the registered points are less than 3 centimeters apart. The rigid transformation is refined by recalculating with inliers only. This procedure is done repeatedly, and the final transformation is that with the most inliers.

Endres et al. chose to do pairwise registration, meaning that every image was registered with every other image. However, they found that this soon became too

computationally intensive, so they chose a subset of 20 frames to register to. This subset consisted of the 3 most recent frames, plus a uniformly sampled group of all other previous images.

4.4. Pose Graph

The results of Step 3 are all stored in a pose graph, where each node represents a frame and the edges between nodes represent the transformations between frames. If a frame could not be matched to any previous frames, the authors describe two possible choices.

If the application allows for a fragmented map such that the final result has at least two distinct, disconnected sections, then the unmatched frame can simply be added to the pose graph without any edges. Endres et al. did not want a fragmented map, so they chose to store the unmatched frame at the same pose as the previous frame but with a high level of uncertainty. Their reasoning for this was that it would better facilitate evaluation against other approaches.

4.5. Global Optimization

The final pose graph then underwent global optimization and edge pruning. A globally consistent camera trajectory was achieved with the g^{2}o optimization algorithm [7], which employs the following cost function (Fig.3):

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j\rangle \in \mathcal{C}} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^T \mathbf{\Omega}_{ij} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$$

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \mathbf{F}(\mathbf{x})$$

Fig.3: Cost function for g^{2}o global pose graph optimization.

While Endres et al. explicitly provided this cost function, they did not go into much detail about its implementation. $\mathbf{x}$ represents all of the frames, while $\mathbf{z}$ and $\mathbf{\Omega}$ represent the mean and information matrix for some error constraint. $\mathbf{e}$ describes the error between $\mathbf{x}_i$ and $\mathbf{x}_j$ according to the constraint $\mathbf{z}_{ij}$, such that $\mathbf{e} = \mathbf{0}$ when $\mathbf{x}_i$ and $\mathbf{x}_j$ are perfectly aligned under the given transformation.

As part of optimization, edges with a high cost (above some threshold) were pruned. Loop closures (e.g. nodes where the camera revisits a previously seen location) greatly facilitate this global optimization to diminish the error that accumulates as the camera moves.

4.6. Voxel Occupancy Grid

Once all of the point clouds have been registered to a common coordinate system, they must be combined in some way. Simply adding them together is large, highly redundant, and computationally expensive. Thus, Endres et al. chose to represent the world model as a 3D occupancy grid map using the OctoMap framework [8]. OctoMap is an octree-based mapping framework that uses probabilistic occupancy estimation to better cope with error and noise. It also explicitly represents free space and unmapped areas, unlike a simple point cloud or mesh representation. This helps facilitate other computer vision tasks, such as robot navigation.

5. Results and Evaluation

An RGB-D benchmark dataset was used for evaluation. This dataset contained several Kinect sequences and synchronized ground-truth data.

First, SIFT (with GPU) and FLANN matching were used on the simplest benchmark dataset (named FR1). The average camera velocity ranged from 0.06 m/s to 0.43 m/s, with angular velocity ranging from 9 deg/s to 42 deg/s. The average translational RMSE (root-mean-square error) was 9.7 cm, and the average rotational RMSE was 3.95º. Each image was processed in an average of 0.35 seconds.

Next, they compared the three feature detectors. SIFTGPU and SURF performed similarly, with average translational RMSEs of 0.097 m and 0.098 m, and average rotational RMSEs of 3.95º and 3.39º. ORB performed relatively poorly, failing 2 of the 9 cases. ORB's average RMSEs were 0.215 m and 7.75º.

Next, they evaluated runtimes. ORB was faster than both SIFT and SURF by one order of magnitude (0.018 s vs. 0.19 s and 0.34 s). Additionally, FLANN performed feature matching faster than brute force by a factor of 2 (0.203 s vs. 0.386 s). Endres et al. very briefly mentioned pose graph optimization, only stating that it was extremely fast. The ratio of graph optimization time to total runtime was consistently below 6%. They also stated that for "reliable estimates," global optimization was not required for every step.

6. Pros, Cons, and Project Relevance

This paper is extremely detailed and concise. It is written well, making it easy to understand and follow. Additionally, all of the code is open source for evaluation and comparison.

However, I had some unanswered questions. Firstly, the paper did not describe how the camera pose transformation is calculated, which is a fundamental part of the algorithm's functionality. Secondly, It was unclear why they could not simply discard a frame that could not be matched to any previous frames, instead of incorporating it into

the pose graph with a high uncertainty. Thirdly, the details for loop closure optimization and how exactly the error is diminished when a loop is closed were not discussed at all. Lastly, the statement about “reliable estimates” not requiring global optimization was brief and unclear. How does one determine when an estimate is reliable and how often global optimization should be used?

Overall, this paper is a great introduction to the SLAM problem and how one might solve it using RGB-D data. Although the implementation is not particularly translatable to our project, the approach is a very good basis for our project. Their ideas for reducing pose graph error and selecting a subset of 20 frames for pairwise registration are potentially relevant to our implementation. Lastly, their comparison of feature detectors/matchers in both runtime and performance can help us optimize our own pipeline.

7. References

[1] Endres, Felix & Hess, Jurgen & Engelhard, Nikolas & Sturm, Jurgen & Cremers, Daniel & Burgard, Wolfram. (2012). An evaluation of the RGB-D SLAM system. Proceedings - IEEE International Conference on Robotics and Automation. 1691-1696. 10.1109/ICRA.2012.6225199.

[2] D. G. Lowe, "Object recognition from local scale-invariant features," Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 1150-1157 vol.2, doi: 10.1109/ICCV.1999.790410.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool (2008). Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding, Volume 110, Issue 3. 346-359, ISSN 1077-3142, https://doi.org/10.1016/j.cviu.2007.09.014.

[4] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, Barcelona, Spain, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544.

[5] Szeliski, Richard. (2011). *Computer Vision: Algorithms and Applications* (1st ed.). London: Springer.

[6] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24, 6 (June 1981), 381–395. doi: https://doi.org/10.1145/358669.358692

[7] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard, "G2o: A general framework for graph optimization," 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 3607-3613, doi: 10.1109/ICRA.2011.5979949.

[8] K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation, 2010.