Comparison of Optimization Techniques for 3D Graph-based SLAM

Doaa M. A.-Latif Mohammed A.-Megeed Salem H. Ramadan Mohamed I. Roushdy
doaa.mahmoud@fcis.asu.edu.eg salem@cis.asu.edu.eg hramadan@eun.eg miroushdy@hotmail.com

Faculty of Computer and Infromation Sciences, Ain Shams University
Abbassia 11566, Cairo, Egypt

Abstract: In this paper we provide an overview of the visual SLAM problem we address the problem of building 3D maps from the data recorded by a Red, Green, Blue, and Depth (RGB-D) sensor. This problem is challenging because it requires the sensor data to be interpreted correctly into the map, minimizing the error while aligning observations to achieve consistency. We also present a comparison of different algorithms used in optimizing a visual graph-based SLAM system on a standard 3D datasets of indoor environments.

Key–Words: Visual SLAM, RGB-D sensor, Graph Optimization

1 Introduction

Simultaneous Localization and Mapping (SLAM) is a well known problem in the computer vision and robotics communities. It refers to the problem of building a map of an unknown environment and at the same time knowing the robot location in this map. SLAM represents the core task of a truly autonomous robot. A wide range of applications such as navigation, object modelling and scene reconstruction depend on the maps generated by the robot.

There has been many approaches to solve the SLAM problem, most of which can be categorized into two main paradigms: filtering and optimization-based approaches [23]. The filtering approaches like extended kalman filter, particle filters and information filters were used widely over the past years due to the fact that the data provided by the robot sensors suffer from noise and inconsistency and these approaches can model different sources of noise and their effects on the measurements, but recently optimization-based approaches have proven to be be more efficient, scalable, and stable than solutions based on filtering algorithms. Optimization-based approach usually uses an underlying graph structure to represent the robot measurements. The graph nodes represent the robot poses and the measurement acquired at this position and the edges represent a spatial constraint relating two robot poses.

Various type of sensors has been used to generate 3D Maps. Classical examples are laser [13][6], stereo cameras [12][16] monocular cameras [11][15][7], and recently RGB-D sensors[4][10][8]. RGB-D sensors provide a trade of between accuracy and complexity. They provide rich visual scenes when compared to laser sensors and at the same time simplify the calibration and rectification processes when compared to monocular or stereo cameras. The most popular RGB-D sensor is the Microsoft Kinect [1]. It has a frame rate of 30Hz and an angular field of view of 57 degrees horizontally and 43 degrees in the vertical axis its depth sensor range is between 1.2m and 3.5m.

This paper presents a comparison of the recent methods for graph optimization in terms of translation, rotation, and trajectory errors. The rest of the paper is organized as follows. An overview of the Graph-based SLAM along with the tested methods are presented in section 2, and the results are detailed in section 3.

2 Graph-based SLAM

The SLAM problem can be represented in a graph based manner. The graph is constructed out of the raw sensor measurements. Each node in the graph represents a robot position and a measurement acquired at that position. An edge between two nodes represents a spatial constraint relating the two robot poses. A constraint usually consists of the relative transformations between the two poses. These transformations are either odometry measurements between sequential robot positions or are determined by aligning the observations acquired at the two

ISBN: 978-960-474-344-5

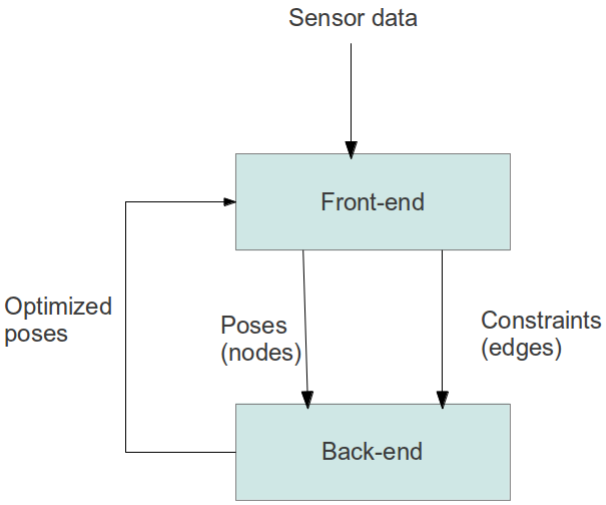

Figure 1: The front-end and back-end of the SLAM process.

robot locations. Once the graph is constructed the optimization process starts to find the configuration of the robot poses that best satisfies the constraints. Thus, in graph-based SLAM the problem is decoupled in two tasks: 1) graph construction in which the graph is built from the raw measurements 2) graph optimization in which the most likely configuration of the poses is determined given the edges of the graph. Fig 1 summarizes the process. The graph construction is usually called front-end and it is heavily sensor dependent, while the second part is called back-end and relies on an abstract representation of the data.

There has been many successful approaches to the visual SLAM problem using the RGB-D sensor. In [8] a hand held approach uses 3D point clouds provided by an RGB-D sensor. The front-end relies on Speeded up robust features (SURF) for feature extraction and matching and then the location is estimated using RANdom SAmple Consensus (RANSAC). The generated map is refined using Hierarchical Optimization for Pose Graphs on Manifolds (HOGMAN). In [10] followed the same path but with some changes, scale invariant feature transform (SIFT) are used to align consecutive data frames then RANSAC and an RGBD-Iterative closest point (ICP) algorithm is used for refining the alignment. This new variant of ICP combines shape and visual information for scan alignment. The SIFT features, verified with RANSAC, act as an initialization for ICP, which reduces the computation time. The Tree-based network optimizer (TORO) was used for global optimization. This approach makes a new addition when compared to the previous ones by using surface elements (SURFELS). The SURFELS reduce the generated map size by a factor of 32 and present a better map quality for viewing, but unfortunately they affect the computation time preventing the whole system from operating in real time. In [4] a graph-based SLAM system is built using the oriented FAST and rotated BRIEF (ORB) as a feature detector and descriptor. The poses were calculated using RANSAC and further refined using the Generalized Iterative Closest point (GICP). Finally global optimization was achieved using the General (Hyper) Graph Optimization g^{2}o.

The approach used to complete this study follows the related work presented above. Fig 2 provides an overview of the system. In the front-end the graph is constructed as the camera moves, new areas are discovered and new poses are added to the graph. When adding a new pose registration is required to align the data together. After a while small errors in registration accumulate resulting in inconsistency in the generated map. This is obvious if the robot visited a previously mapped place, the error will result in presenting the same place twice in the map. Here comes the need for the back-end to adjust the accumulative error and align the complete data sequence.

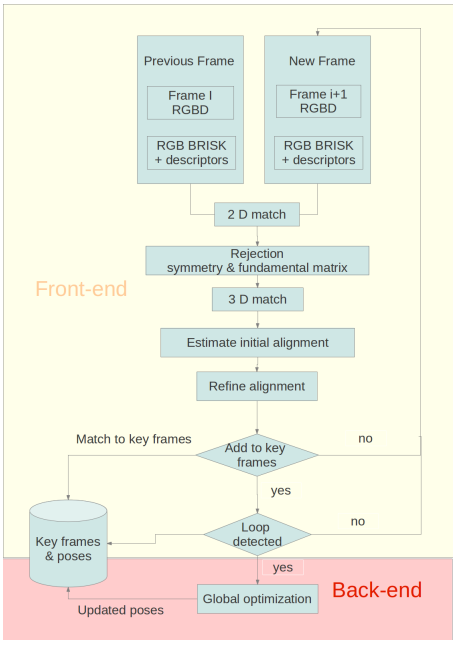

Figure 2: A Block diagram of the working system.

ISBN: 978-960-474-344-5

2.1 SLAM Front-End

The SLAM front-end uses the registration step to align consecutive data frames. The alignment is usually done by estimating an approximate transformation between the consecutive frames and then refining this initial estimate. The approach used in this study is similar to the one presented in [4]. It can be summarized into 3 main steps:

1. Computing the correspondence between successive frames
   (a) Find 2D feature correspondence between RGB Images.
   (b) Reject bad correspondence.
   (c) Transform the 2D features to their equivalent 3D features.
2. Estimate the initial alignment of the frames.
3. Refine the alignment.

We have presented different techniques for feature detection and description and pose refinement and compared them in our earlier work [3] and as a result we found that the best configuration used is the Binary Robust Invariant Scalable Keypoints (BRISK) as a feature detector and descriptor and the generalized iterative closest point (GICP) as a pose refiner.

The loop detection is an important part of the front-end without it the graph will be like a linear chain, loops are represented as edges between nodes that are not temporally adjacent. Once a loop is detected the new correspondence between nodes can be used as an additional constraint in the graph. Fig 3 shows an example. The question now is how to detect the loop in an efficient way. There has been many loop closing techniques presented in the literature and according to [20] they can be classified into 3 main categories:

1. **Map to map** as the name indicates this is used in approaches that depends on building small submaps of the environment, the correspondence between sub-maps is investigated taking into account the visual appearance and the relative position between the sub-maps.
2. **Image to map** the most recent image captured is compared with the built map features looking for correspondence. The pose of the camera is determined relative to a map of point features by finding correspondences between the image and the features in the map.
3. **Image to image** the correspondence is investigated between images of the world being mapped, the most recent image is compared with previously captured images looking for matches.

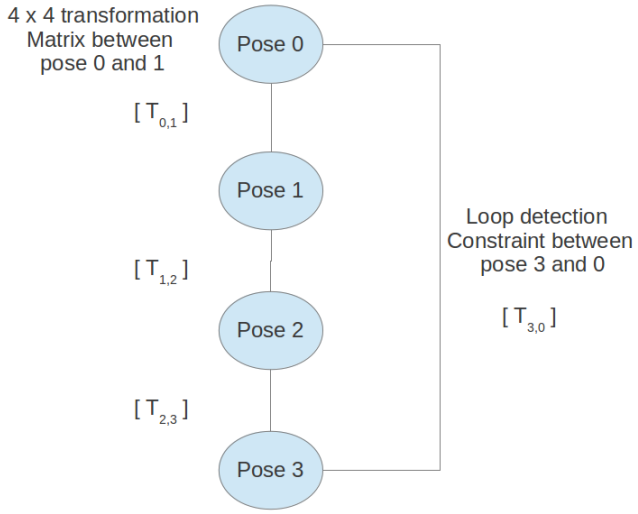

Figure 3: An additional constraint is added to the graph when a loop is detected.

In this study a simple image to image loop detection approach is used. A set of key nodes is stored with the features detected and described in the registration step are kept for each key node. When a new observation is processed its location is examined if it is close to any of the key nodes it is further checked and its features are matched with this close node, if the number of matches passes a certain threshold in our case 40 features then this node is considered as a loop closing node and a constraint represented as an edge is added to the graph connecting the two matched nodes.

2.2 SLAM Back-End

Registration between successive frames is a good method for tracking the robot position over moderate distances. But noise and errors in depth values and in pairwise frame alignment cause the estimated robot pose to drift over time resulting in an erroneous map. The SLAM back-end role is to optimize the map reducing this error by optimizing the underling graph structure provided by the front end, this graph is

composed of n vertices storing the observation at certain poses, and edges representing the neighbour relations between these poses. Global optimization techniques tries to estimate optimally all poses to build a consistent map of the environment. It can be considered as choosing the best solution that minimize an error function from all the feasible solutions.

If we assume that $X = (x_1, ..., x_n)^T$ is a vector describing the robot poses where x_i describe the pose of node i and z_{ij} is the mean and Ω_{ij} is the information matrix of the transformation matrix that aligns the observation at node i and node j together. Let $\hat{z}(x_i, x_j)$ be the estimated measurement at nodes x_i and x_j. The log likelihood l_{ij} of z_{ij} can be calculated as $l_{ij}\alpha[z_{ij} - \hat{z}_{ij}(x_i, x_j)]^T\Omega_{ij}[z_{ij} - \hat{z}_{ij}(x_i, x_j)]$ If we consider $e(x_i, x_j)$ as an error function that computes the difference between the estimated measurement $\hat{z}_{ij}$ and the real measurement z_{ij} such that $e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$. The goal of a maximum likelihood approach is to find the configuration of the nodes x that minimizes the negative log likelihood $F(x)$ of all the observations

$$F(x) = \sum_{<i,j>\in C} \underbrace{e_{ij}^T \Omega_{ij} e_{ij}}_{F_{ij}} \quad (1)$$

where C is the set of pairs of indices for which a constraint (observation) z exists. This means that we tries to find the solution to

$$x^* = \underset{x}{\operatorname{argmin}} F(x) \quad (2)$$

The function value of $F(x)$ at the minimum is not important, what matters is the value of the variable x^* where that minimum occurs, further details about graph optimization in SLAM can be found in [21] [23].

In our study we have compared the performance of 3 different global optimizers:

1. **General (Hyper) Graph Optimization**
   Known also as g^{2}o [9]. It is a C++ framework for performing the optimization of nonlinear least squares problems that can be embedded as a graph or in an hyper-graph.

2. **Georgia Tech Smoothing and Mapping**
   Known also as GTSAM [5]. It is a C++ library based on factor graphs. A factor graph consists of factors connected to variables. The factors represent probabilistic information on the unknown random variables in the estimation problem.

3. **Hierarchical Optimization on Manifolds**
   Known also as HOG-Man [22] it applies Gauss-Newton with sparse Cholesky factorization that considers a manifold representation of the state space to better deal with the camera rotations.

The following open source projects were used to implement different techniques subject to our comparison: The point cloud library (PCL) [14] was used in the transformation estimation and refinement. It is a large scale, open source project for 2D/3D image and point cloud processing, the Open source computer vision library (OpenCV) [2] which has been used in feature detection, description and matching. The different algorithms used in this comparison were tested on the Computer vision and pattern recognition group (CVPR) [18] datasets. These datasets contain the color and depth images of a Microsoft Kinect sensor along with the ground-truth trajectory.

3 Results

All the tests were performed using an Intel Core i7-3610QM CPU @ 2.30GHz $\times$ 8 running a 32 bit Linux 3.2.0. We used four of the CVPR group [18] datasets in our tests: Freiburg2_xyz which contains data for debugging translations where the Kinect was moved along the principal axes in all directions, Freiburg2_desk which captures the details of a typical office scene with two desks, a computer monitor, keyboard, phone, chairs, etc. with the Kinect moving around two tables so that the loop is closed, Freiburg1_room which has been recorded through a whole office environment and is well suited for evaluating how well a SLAM system can cope with loop-closures, and Freiburg2_pioneer_slam which was recorded from a Kinect mounted on top of a Pioneer robot which was joysticked through a maze of tables, containers and other walls, so that several loops have been closed for map building.

We have used the CVPR group evaluation tools to compare the global optimization algorithms. Two error metrics have been used: the absolute trajectory error (ATE), and the relative pose error (RPE). The ATE is useful for measuring the performance of visual SLAM systems. It measures the absolute trajectory error by comparing the difference between the estimated and the groundtruth path after associating them using the timestamps. It also computes the mean, median and the standard deviation of these differences. The RPE is useful for measuring the drift of visual odometry systems. It computes the error in the relative motion between pairs of timestamps.

ISBN: 978-960-474-344-5

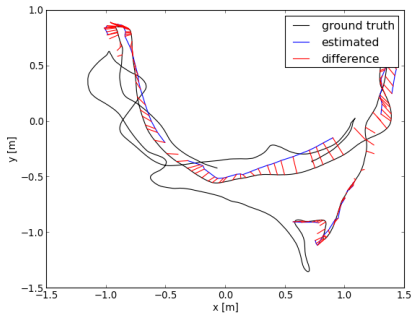

Figure 4: The result trajectory compared to the groundtruth of the Freiburg1_room dataset using g^{2}o.

Table 1: The absolute trajectory error on the Freiburg1_room dataset in meters.

Absolute Trajectory Error	g^{2}o	GTSAM	HOG-Man
rmse	0.079022	0.083895	0.079756
mean	0.068790	0.074727	0.070692
median	0.064971	0.069797	0.065114
std	0.038891	0.038135	0.036927
min	0.009273	0.015323	0.010718
max	0.154001	0.156559	0.163685

Table 2: The relative translation error on the Freiburg1_room dataset in meters.

Relative Translation Error	g^{2}o	GTSAM	HOG-Man
rmse	0.078440	0.089116	0.077349
mean	0.073345	0.081969	0.071589
median	0.076118	0.085064	0.073415
std	0.027809	0.034967	0.029291
min	0.014281	0.015838	0.014209
max	0.147132	0.162746	0.147075

The evaluation results of the Freiburg1_room dataset are presented. The ATE results are detailed in Table 1 and the RPE in Tables 2, 3. Fig. 4 shows the difference between the estimated trajectory and the groundtruth using g^{2}o.

The average error on the tested four datasets is described in Table 4 in which all three produce similar scores but g^{2}o performs slightly better.

4 Conclusion

In this paper a detailed algorithm for building 3D visual Maps using only an RGB-D sensor is presented. We have used various open source libraries to complete our study. All the algorithms were tested on the CVPR datasets. In the future we would like to perform

Table 3: The relative rotation error on the Freiburg1_room dataset in degrees.

Relative Rotation Error	g^{2}o	GTSAM	HOG-Man
rmse	3.773067	4.346548	3.628420
mean	3.398897	3.847612	3.127450
median	0.053414	0.065668	0.042559
std	1.638149	2.021969	1.839698
min	0.884001	0.511624	0.282858
max	7.021277	8.317298	7.151021

Table 4: The average absolute trajectory error and relative pose error on the tested four datasets.

Average rmse	g^{2}o	GTSAM	HOG-Man
Absolute trajectory	0.157364 m	0.158508 m	0.160902 m
Relative translation	0.081775 m	0.109753 m	0.085200 m
Relative rotation	4.279843 deg	2.436107 deg	5.969135 deg

a study on different loop closures techniques to find the best in terms of accuracy and memory consumption, also we would like to investigate the possibility of Building a SLAM system using multiple RGB-D sensors each exploring a part of the environment.

References:

[1] Microsoft kinect.2010. [Online]. Available:http://www.xbox.com/en-US/kinect

[2] Open source computer vision (opencv), November 2012. [Online]. Available:http://opencv.org

[3] Doaa M. A.-Latif, Mohammed A.-Megeed Salem, H. Ramadan, and Mohamed I. Roushdy. Comparison of 3d feature registration techniques for indoor mapping. In *8th IEEE International Conference on Computer Engineering and Systems (ICCES)*, page To appear, 2013.

[4] M. Algaba, JL. Blanco, and J. González. Desarrollo e implementación de un método de generación de mapas 3d usando el sensor kinect. Master's thesis, Higher Technical School of Computer Science Engineering, MAPIR Group, System Engineering and Automation Department, University of Málaga, mar 2012.

[5] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, December 2006.

[6] I. Dryanovski, W. Morris, and Jizhong Xiao. Multi-volume occupancy grids: An efficient

probabilistic 3d mapping model for micro aerial vehicles. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1553–1559, 2010.

[7] Sara Elgayar, Mohammed A.-Megeed Salem, and Mohamed I. Roushdy. Two-level topological mapping and localization based on sift and the wavelet transform. In *2nd International Conference on Circuits, Systems, Communications, Computers and Applications*, Dubrovnik, Croatia, June 25-27 2013.

[8] Nikolas Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard. Real-time 3D visual SLAM with a hand-held RGB-D camera. In *Proceedings of the RGB-D workshop on 3D perception in robotics at the European robotics forum, Vasteras, Sweden*, number c, 2011.

[9] Rainer Kuemmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[10] Peter Henry, Michael Krainin, Evan Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, 2010.

[11] Pierre Lothe, Steve Bourgeois, Fabien Dekeyser, Eric Royer, and Michel Dhome. Monocular SLAM Reconstructions and 3D City Models: Towards a Deep Consistency. *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, pages 201–214, 2010.

[12] Christopher Mei, Gabe Sibley, Mark Cummins, Paul Newman, and Ian Reid. RSLAM: A System for Large-Scale Mapping in Constant-Time Using Stereo. *International Journal of Computer Vision*, 94(2):198–214, June 2010.

[13] William Morris, Ivan Dryanovski, and Jizhong Xiao. 3d indoor mapping for micro-uavs using hybrid range finders and multi-volume occupancy grids. In *RSS 2010 workshop on RGB-D: Advanced Reasoning with Depth Cameras, Zaragoza, Spain*, 2010.

[14] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011.

[15] Mohammed A-Megeed Salem Multi-Stage Localization Given Topological Map for Autonomous Robots. In *The 8th IEEE International Conference on Computer Engineering and Systems (ICCES12)*, November 27-29, 2012, Cairo, Egypt.

[16] David Schleicher, Luis M. Bergasa, Manuel Ocaña, Rafael Barea, and Elena López. Real-time hierarchical stereo Visual SLAM in large-scale environments. *Robotics and Autonomous Systems*, 58(8):991–1002, August 2010.

[17] B Siciliano and O Khatib. *Springer handbook of robotics*. 2008.

[18] Jurgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012.

[19] Sebastian Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[20] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós. A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12):1188–1197, December 2009.

[21] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, Wolfram Burgard. A Tutorial on Graph-Based SLAM. *IEEE Intelligent Transportation Systems Magazine,*, 52(3):199–222, 2010.

[22] Giorgio Grisetti, R Kummerle, Cyrill Stachniss, Udo Frese, and Christoph Hertzberg. Hierarchical optimization on manifolds for online 2d and 3d mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 273–278. IEEE, 2010.

[23] N Sünderhauf. *Robust Optimization for Simultaneous Localization and Mapping*. PhD thesis, Technische Universitat Chemnitz,, 2012.

ISBN: 978-960-474-344-5