Proceeding of the 2nd
RSI/ISM International Conference on Robotics and Mechatronics
October 15-17, 2014, Tehran, Iran

An Online Implementation of Robust RGB-D SLAM

M. A. Athari
Faculty of Electrical Engineering
K.N. Toosi University of Technology
Tehran, Iran 16315-1355
miralireza.athari@ee.kntu.ac.ir

H. D. Taghirad *Senior Member, IEEE*
Faculty of Electrical Engineering
K.N. Toosi University of Technology
Tehran, Iran 16315-1355
taghirad@kntu.ac.ir

***Abstract*—This paper presents an online robust RGB-D SLAM algorithm which uses an improved switchable constraints robust pose graph slam alongside with radial variance based hash function as the loop detector. The switchable constraints robust back-end is improved by initialization of its weights according to information matrix of the loops and is validated using real world datasets. The radial variance based hash function is combined with an online image to map comparison to improve accuracy of loop detection. The whole algorithm is implemented on K. N. Toosi University mobile robot with a Microsoft Kinect camera as the RGB-D sensor and the whole algorithm is validated using this robot, while the map of the environment is generated in an online fashion.**

I. Introduction

During the recent years, RGB-D cameras have been used vastly for odometry estimation of indoor environments in SLAM for mobile robots. These cameras provide depth information of the environment, like stereo cameras, while they are usually much cheaper than the latter. The main problem of RGB-D cameras is limited range of depth estimation which can cause uncertainty in depth information of indoor environments and can result in a failure of a RGB-D SLAM algorithm.

RGB-D SLAM algorithm consist of two front-end and back-end sections. The front-end section receives RGB-D data and extracts motion estimation as well as detecting the loops, while the back-end has to deal with front-end output to optimize the pose estimations and build the map. Detecting and closing correct loops reduces the motion estimation errors in the map but a false positive loop can result in failure of the RGB-D SLAM algorithm. Preventing from detection of false loops can be accomplished by two approaches. The first approach is to design a loop detection algorithm in front-end with minimum false positive detection [1], [2]. The second approach is using a robust back-end which can detect and remove false loops from optimization step [3], [4], [5]. Trying to avoid detecting false loops in front-end section may lead to many undetected true positive loops in indoor environments. Furthermore, even the most accurate loop detection methods can not guarantee $\%100$ accuracy. Robust loop closing methods in the back-end section extend the optimizer with the ability of detecting and removing or disabling false positive loops detected by the front-end. These robust back-end methods may be used along side an accurate front-end loop detector, while it will result in a slow SLAM performance.

For online implementation of RGB-D SLAM algorithm we need a fast and robust back-end. The main disadvantage of RRR robust loop closing algorithm [5], is its slow performance compared to that of other algorithms. This drawback makes this method unusable for online implementations. Among two other popular robust loop closing back-ends introduced in the literature, the switchable constraints algorithms has better accuracy and speed compared to that of max-mixture algorithm [9]. However, this algorithm does not perform well in some real world datasets. In this paper we use a robust back-end which is based on switchable constraints algorithm [3], and extend it by adding useful information of the front-end loop detector. It is shown in this paper that by using this information the results of this algorithm is significantly improved for real world datasets. We use this method alongside with iSAM2 graph optimizer [8], to have an online framework. By this means, in the front-end we use fast odometry from vision (FOVIS) library for motion estimation [6]. For detection of the loops in the front-end section we use an algorithm based on radial variance hash function [7], which can suitably detect the loops in small amount of time compared to that of other detectors. Finally the proposed RGB-D SLAM algorithm is implemented on the mobile robot of K. N. Toosi University and the real time implementation results are discussed in details.

II. Backgrounds

In front-end section of a SLAM algorithm, a visual odometry algorithm first extracts the odometry constraints and then the loop detector searches for any loops and finds the loop constraints. Furthermore, the pose graph will be sent to back-end for nonlinear optimization of this graph. In online implementations, this procedure concurrent with every frame of RGB-D camera capture. Since the accuracy of each section will specify the total accuracy of the iSLAM algorithm, in this section we will review the details of these sections that are being used in the SLAM algorithm.

A. FOVIS

FOVIS is a visual odometry system for odometry estimation that is used in both RGB-D and stereo cameras. FOVIS extracts FAST features from RGB images and by matching

978-1-4799-6743-8/14/$31.00 ©2014 IEEE

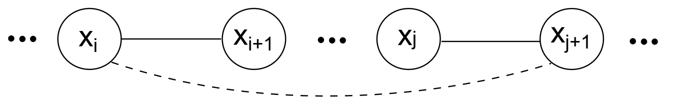

Fig. 1. A simple pose graph demonstration

them and finding inlier matches, it uses absolute orientation method for motion estimation. FOVIS uses depth information by minimizing the reprojection error and refines the estimated motion. Because of FAST features, FOVIS can perform very fast and also provides suitable accuracy.

B. Radial Variance Based Hash Functions

Hash functions may be used for comparing images. Some hash functions provide a good robustness against image distortions like rotation, resize, etc. The radial variance based hash function introduced by Lefebvre *et al* is one of the robust hash functions which may be used for comparing images in short time [7]. The radial variance vector is defined as

$$R[\alpha] = \frac{\sum_{(x,y)\in\Gamma[\alpha]} I^2(x,y)}{\#\Gamma[\alpha]} - \left(\frac{\sum_{(x,y)\in\Gamma[\alpha]} I(x,y)}{\#\Gamma[\alpha]}\right)^2 \quad (1)$$

in which, α being an angle changing from $0°$ to $179°$ and $I(x,y)$ denotes the intensity of (x,y) pixel. Using this function on an image will result in an array with size of 180 describing the image. Furthermore, a DCT function is used on the array to compute the radial variance hash of the above equation and the first forty coefficient will be considered as radial variance hash of the image. For comparison of radial variance based hashes, peak of cross correlation (PCC) is used as a popular criterion.

C. Pose Graph Slam

A pose graph is a graph which only consists of robot poses, and therefore, there is no landmark depicted in this graph. Robust loop closing methods are based on pose graph SLAM formulation. A simple demonstration of pose graph slam is shown in Figure 1 in which each circle shows a robot pose, solid lines denote odometry constraints and dashed line denotes a loop constraint. Given a set of odometry constraints $u_{i,i+1}$ between pose x_i and pose x_{i+1} is we have

$$x_{i+1} = f(x_i, u_{i,i+1}) + w_{i,i+1} \quad (2)$$

in which, $w_{i,i+1}$ is a zero mean Gaussian error term and f is a nonlinear function which implements motion model of the robot. for the loop constraints between pose x_i and x_j we have

$$x_j = f(x_i, u_{i,j}) + v_{i,j} \quad (3)$$

in which, $v_{i,j}$ is the zero mean Gaussian error term. For the whole graph, the conditional probability over all variables may be written as

$$P(X|U) \propto \Pi_{i,i+1} P\left(x_{i+1}|x_i, u_{i,i+1}\right) \cdot \Pi_{i,j}\left(x_j|x_i, u_{i,j}\right) \quad (4)$$

in which,$X = \{x_i\}$ and $U = \{u_{i,i+1}, u_{i,j}\}$. In SLAM we seek for maximum of posterior robot poses, X^*. If it is assumed that the probabilities are Gaussian, then the problem may be written as a nonlinear least squares optimization:

$$\begin{aligned} X^* =& \underset{x}{argmax} P(X|U) = \underset{x}{argmin} - logP(X|U) \\ =& \underset{x}{argmin} \sum_{i,i+1} ||f(x_i, u_{i,i+1}) - x_{i+1}||^2_{\sum_{i,i+1}} \\ &+ \sum_{i,j} ||f(x_i, u_{i,j}) - x_j||^2_{\Lambda_{i,j}} \end{aligned} \quad (5)$$

with $||a-b||^2_{\sum}$ being the Mahalanobis distance. Solving this nonlinear least squares problem will lead to a maximum posterior of robot pose.

D. Switchable Constraints Robust Pose Graph SLAM

The main idea of this algorithm is not to keep the topology of graph fixed, therefore, it can be changed during the optimization. In this method a binary weight $w_{i,j}$ is assigned to each set of loop constraints. As a binary weight cannot be used with specific nonlinear optimization algorithms, a continuous switch variable $s_{i,j} \in \mathbb{R}$ is define for each weight. It is not necessary to have function that maps the continues space to discrete values for weights, and therefore, even functions like sigmoid may be used as functions to map switch variables to continuous space of $(0,1)$ by:

$$w_{i,j} = \Psi(s_{i,j}) : \mathbb{R} \to (0,1) \quad (6)$$

These switch variables need an initial value which is considered such that the initial weights to be near one. Therefore, it is assumed that all loops are correct in the beginning of the optimization. The nonlinear least squares problem with the presence of these switch variables may be written as

$$\begin{aligned} X^*, S^* = \underset{x}{argmin} & \sum_{i,i+1} ||f(x_i, u_{i,i+1}) - x_{i+1}||^2_{\sum_{i,i+1}} \\ &+ \sum_{i,j} ||sig(s_{i,j} f(x_i, u_{i,j}) - x_j||^2_{\Lambda_{i,j}} \\ &+ \sum_{i,j} ||\lambda_{i,j} - s_{i,j}||^2_{\Xi_{i,j}} \end{aligned} \quad (7)$$

in which, the final term has been added to avoid all switch variables to move toward amounts that result in near zero weights, and all the loops disabled. In this term $\lambda_{i,j}$ is initial value of the switch variable.

III. Switchable Constraints With Weight Initialization

In switchable constraints algorithm, the initial weights are always assumed to be one. This assumption can greate3y affect the convergence of the optimization. As the optimization problem in switchable constraints algorithm is an augmented one, the initial weights become more important on the rate of convergence. In front-end section after detection of a loop we have to use a motion estimator to extract loop constraints. Most motion estimation methods are based on a least square solver, and therefore, they can provide an information matrix (inverse of covariance matrix) of the estimation error. This matrix provides detailed information about the certainty of estimation, and therefore, it can be used to initialize the weight values of the switchable constraints algorithm.

The diagonal elements of the information matrix are the most important elements and in most cases the effect of cross terms are almost negligible. In this paper it is proposed to use the diagonal elements for initialization of switchable constraints weights. The switchable constraints weights are scalars and between 0 and 1. Hence, we define the initial values of these weights as

$$w_{i,j}(0) = \frac{\text{tr}(R_{i,j})}{N} \tag{8}$$

in which, N is dimension of information matrix which is 3 in 2D environments and is 6 in 3D environments. $R_{i,j}$ is information matrix related to the loop detected between pose i and pose j. According to this equation, each $w_{i,j}$ will be initialized with average of variances of estimated constraint elements. This value is a number between 0 and 1 and can describe the certainty of estimated motion between pose x_i and x_j. Using this initialization can lead to a more robust optimization results compared to that on conventional switchable constraints algorithm. This claim is verified in the reported experimental results.

IV. Radial Variance Based Hash Function Loop Detector

The next contribution of this paper is the proposal of a loop detection algorithm based on radial variance hash functions. This algorithm only uses RGB images of RGB-D cameras for comparing images and detecting the loops. Using PCC[1] criterion for loop detection using these hash function needs a threshold to be specified for deciding whether two compared images are indicating a loop or not. In indoor environments, an unjustified threshold may either result in detection of many false positive loops or may result in loss of many true positive loops. As existence of many false positive loop can affect the accuracy of robust back-end, we have proposed a method based on the idea of [10] to reduce the impact of threshold on accuracy. By this means, instead of comparing each image related to a robot pose to other images of other poses we will compare each image to the small amount of map that is built till the current robot pose. By using this procedure we will have an online implementable algorithm which increases the accuracy of radial variance based hash loop detector.

Figure 2 shows a graphical demonstration of distance matrix of the proposed method. In this case we have compared the current pose of the robot to three past poses. If there exist a path consisting of 4 comparison with PCC of more than a specific threshold, the current position will be accepted as a loop. By using this method false positive loops, that usually occur as a result of the randomness characteristic of hash functions, will be rejected and also impact of the threshold on loop detection will be decreased. In Figure 2 each pixel shows a comparison between two images of the robot two poses. A black pixel shows a comparison with PCC of 1 and a white pixel shows a PCC of 0, hence, darker pixels mean higher PCC. Figure 2.a shows an accepted loop as there is a path indicating a loop and Figure 2.b shows a rejected loop as there is no such path. In indoor environments we can not search for a path within many previous poses because the loop may only exist in few images. It is later shown by precision-recall curves that using this method can significantly increase the performance of loop detection based on radial variance hash function.

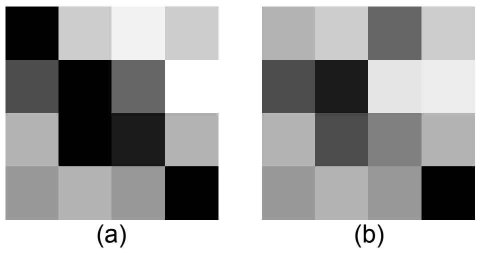

Fig. 2. Comparing image to map for loop detection: (a) shows an accepted loop and (b) shows a rejected one.

V. Experimental Results

All back-end experiments have been run using iSAM2 [11], which is an incremental graph optimizer to enable us for online implementation. The benchmark system is a laptop with a core i7 cpu with Ubuntu 12.04 LTS operating system.

A. Switchable Constraints With Weight Initialization Results

For this part we have also implemented max-mixture method in iSAM2 framework for sake of comparison in real-world datasets. According to results reported in [7], two real-world 2D datasets Bovisa06 and Bovisa04 reported in [12], the switchable constraints algorithm does not perform well on the latter. Furthermore it provides worse results than that of max-mixture algorithm in g2o [13], framework.

Table I shows the results of the implementation of Switchable Constraints (SC), Max-Mixture (MM) and the proposed initialized SC (iSC) for both mentioned datasets in iSAM2 frame work. For each dataset there is two column in this table. The first column shows RMSE according to ground truth of the dataset, while the second column shows the computation time needed for performing the optimization of the whole dataset using iSAM2 framework within online implementation. Comparing the first and the second rows, it is clearly seen that MM algorithm has less error for Bovisa06 dataset compared to SC algorithm while for dataset Bovisa04 the converse is observed. The time that is needed for optimization of the whole datasets in iSAM2 framework is slightly more for SC algorithm compared to MM algorithm, but the difference is around 10 percent of the whole time, which is tolerable. The error behavior is the same in what reported in [7] for the g2o framework, as well.

The third row shows the proposed initialized SC results and as we can see in both datasets the RMSE has decreased

TABLE I
MM, SC & iSC RESULTS FOR BOVISA06 & BOVISA04

	Bovisa04		Bovisa06	
	RMSE(m)	time(s)	RMSE(m)	time(s)
MM	32.2081	2939.99	24.0652	2693.12
SC	28.5109	3387.02	31.2925	2832.18
iSC	22.6847	3385.26	23.0868	2832.94

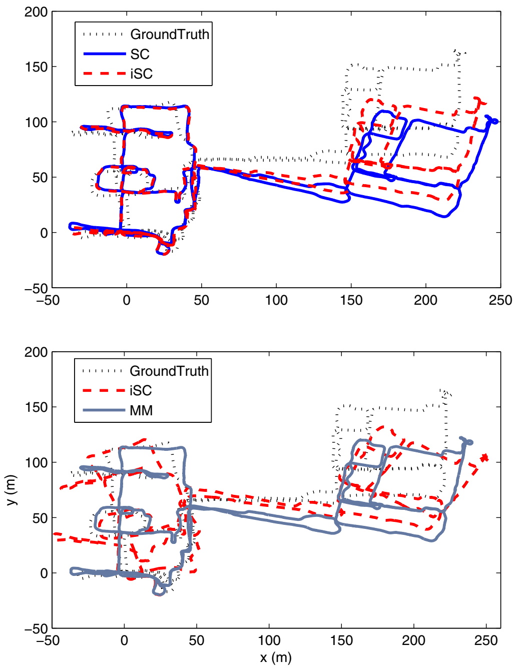

Fig. 3. Bovisa06 : Dotted line is ground truth of dataset, solid line is SC result and the dashed line is iSC result (top). Dotted line is ground truth of dataset, solid line is MM result and the dashed line is iSC result (bottom).

compared to that of SC algorithm. For Bovisa06, the RMSE is even less than MM algorithm and for both datasets, iSC algorithm provides better RMSE compared to that of other two algorithms. As it is shown in second column of each dataset, the time that is needed for optimization did not noticeably change when we added weight initialization. This is due to the fact that in iSAM2 framework the optimization is run only one iteration for each pose added to the graph.

Figure 3 illustrates the results compared to ground truth for Bovisa06. In the top figure, the dashed line, which denote the result of iSC algorithm, is closer to ground truth than that of the solid line, which represents the results of the SC algorithm. Also in the bottom Figure, the MM algorithm result which is shown in solid line is far from the ground truth of the left building but it is closer to the ground truth of the right building compared to that of iSC. Furthermore, the global numerical results shown in Table I, verifies that that iSC has better overall results for this dataset. According to the numerical and graphical results, adjusting initial weights with respect to the information matrix, will result in improvement of switchable constraints robust algorithm error.

TABLE II
LOOP DETECTION RUN TIME COMPARISON

	RVB (image to map)	RVB (image to image)	FAB-MAP2.0
Run Time	8.43s	7.36s	54.45s

B. Loop Detection Using Radial Variance Based Hash Function

Radial variance based hash function can detect loops in a small amount of time. To see this, we have compared the run time of this function with FAB-MAP2.0 algorithm [1], which is considered as a gold standard in loop detection of SLAM robots. For this comparison we have generated a RGB-D dataset in K. N. Toosi University using a Microsoft Kinect camera. This dataset is collected with a mobile robot while the Kinect camera was mounted on it. Table II shows the run time of FAB-MAP2.0 and radial variance based hash function loop detector. For implementation of radial variance based hash we have use pHash library [14]. This table shows that radial variance based hash function performs significantly faster than FAB-MAP2.0, whether we use an image to image or an image to map comparison.

For accuracy comparison, the precision-recall curves of the proposed algorithm and the FAB-MAP2.0 are given in Figure 4. As we can see in the left curve, using image to map comparison (dashed line) leads to a better accuracy compared to that of image to image comparison (solid line), for radial variance based hash function loop detector. Also both results have better accuracy compared to the right curve which shows precision-recall of FAB-MAP for K. N. Toosi-Avril RGB-D dataset. As the right curve shows, the accuracy of FAB-MAP falls below 50% for recall of below 10% but in the left curve both lines still keep 100% accuracy for more than 10% of recall. This is mainly because of the fact that the FAB-MAP2.0 algorithm is very sensitive to any overlap between the images. In indoor environments the method that FAB-MAP2.0 has implemented to avoid overlapping the images is not working well for Microsoft Kinect images and thus its accuracy reduces very fast. However, our proposed algorithm is not sensitive to overlaps and in presence of overlaps it will not loose true positive loops.

C. Implementation On K. N. Toosi-Avril Robot

By using FOVIS for visual odometry and radial variance based hash function for loop detection we have a complete front-end. We have attached this front-end to proposed back-end and made an online robust RGB-D SLAM system. We implemented this system on K. N. Toosi-Avril mobile robot which is a four wheel robot and we've used the overhead camera of a small size league to locate real position of the robot and make a 2D ground truth. We have implemented the

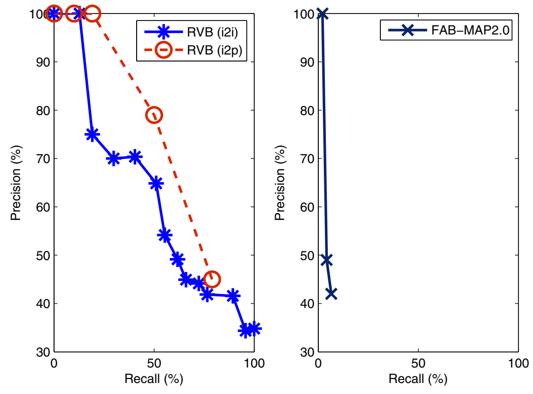

Fig. 4. Precision-Recall curve for radial variance based hash function algorithm (RVB) with both image to image (i2i) and image to map (i2p) comparison in left curve and FAB-MAP2.0 algorithm in the right curve for K. N. Toosi-Avril RGB-D dataset.

whole system in the following five steps and computed the error of robot pose (RMSE) at each step:

1) Using only FOVIS for visual odometry
2) Using FOVIS and RVB hash function loop detector with simple threshold of 0.85 and iSAM2 as the back-end
3) Using FOVIS and RVB hash function loop detector with image to map comparison and iSAM2 as the back-end
4) Using FOVIS and RVB hash function loop detector with image to map comparison and SC robust back-end
5) Using FOVIS and RVB hash function loop detector with image to map comparison and iSC robust back-end

The results of each step is shown in Table III. In the second step where we use the proposed loop detection algorithm with only a simple threshold, we see that the RMSE goes up and gets near twice as that of a pure visual odometry. This shows that existence of false positive loops can result in failure of the iSAM2 which is the back-end used in here. In the next step by using image to map comparison idea, the error decreases significantly and the computational time slightly increases, which shows the importance of rejecting false positive loops. In the fourth step we use the SC robust back-end, which is implemented in iSAM2. The results shows that there still exist some false positive loops, which are successfully reduced their effects. Also as size of the dataset is not very large, the amount of time that is increased is very limited. In the last step, the whole proposed SLAM system is implemented and the results shows that the final RMSE is the minimum among all others while the runtime is well suited for an online SLAM implementation. The whole dataset consists of 3300 RGB-D frames with total time of 200 seconds, which means that we can have an average frame rate of around 16 frames per second.

Figure 5 shows graphical results of the 5 implemented steps. In this figure only the X-Y view of the robot trajectory is shown since the robot moves on a flat surface. In the sub figure (a) the first step is shown, as we see FOVIS can not provide a good estimation of real robot trajectory (dashed line) alone. In

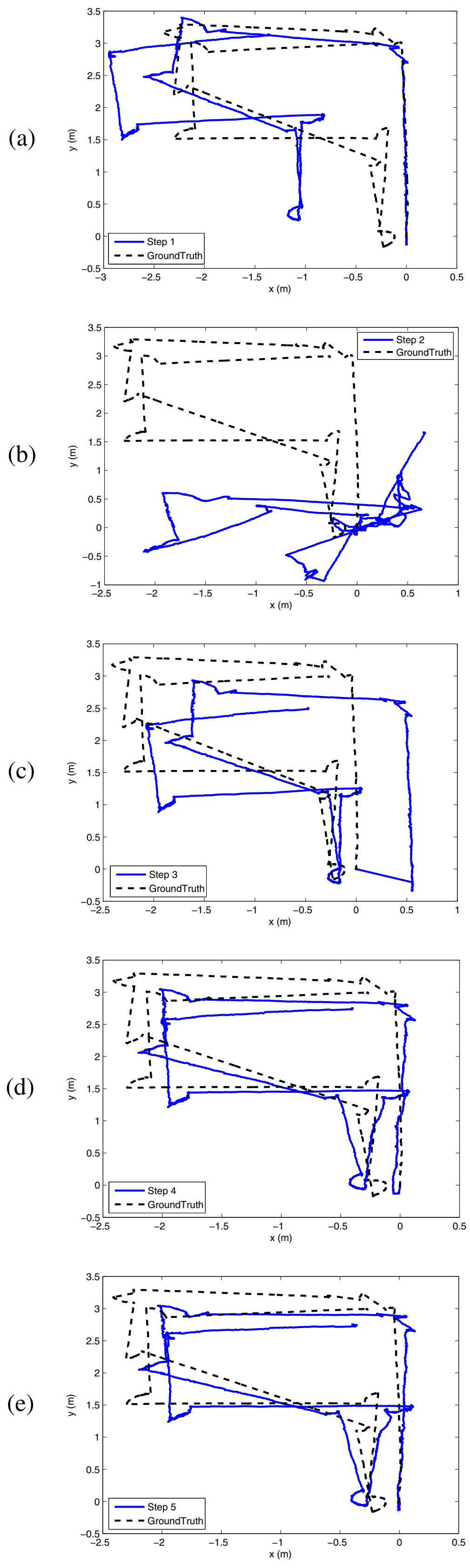

Fig. 5. Graphical results of the whole proposed SLAM algorithm from step 1 (a) to step 5 (e).

TABLE III
RESULTS OF IMPLEMENTATION ON K. N. TOOSI AVRIL ROBOT

	Steps				
	1	2	3	4	5
RMSE (m)	0.353	0.716	0.324	0.121	0.115
Time (s)	112.06	191.16	199.26	199.83	200.07

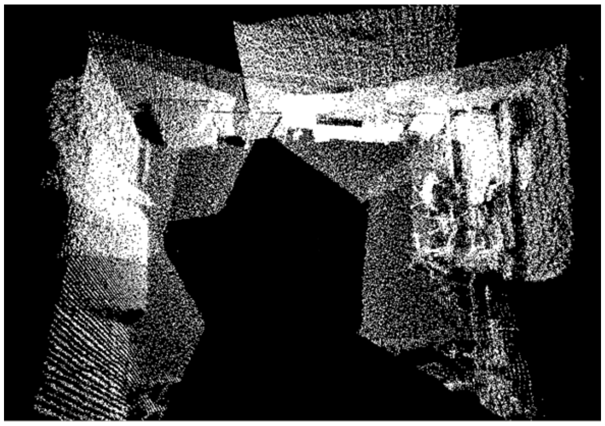
Fig. 6. Graphical results of the whole proposed SLAM algorithm from step 1 (a) to step 5 (e).

the next step (b), the result of proposed loop detection method with simple threshold is illustrated, in which the structure of the robot trajectory is completely destroyed and the algorithm has failed. In the next step (c), the comparison of image to map has resulted in a better estimation but the estimation is still affected by some false positive loops. After using an SC robust back-end in step (d), the whole estimated trajectory gets very close to the ground truth and in the final step there is a little improvement in some areas where the trajectory of robot starts.

At the end we make a point cloud representation map of the environment that the robot has seen. For this implementation PCL [15], the library for working with point clouds is used. By having the exact pose of the robot we only need to put points cloud of each pose together to have the map of environment. This map is show in Figure 6, which is generated without any point registration with frame selection rate of 100. As it is seen in this figure, the structure of the room and it contents is correctly placed in the map. The other wall of the room was very far and not in range of Kinect camera, therefore, it is not mapped.

VI. CONCLUSIONS

This paper proposed an online robust RGB-D SLAM algorithm which has been evaluated from many different aspects. It is shown that using a loop detector that barely detects any false positive loop may result in loss of true positive loops and consequently produces an unacceptable map. Since robust back-end methods may be used to minimize the effects of false positive loops it is not necessary to ensure zero false positive loop detection in front-end section. Furthermore, it is shown that using front-end information to initialize the robust back-end optimizer will improve the result of optimization. This improvement is evaluated for switchable constraints robust pose graph SLAM algorithm in real world datasets. Furthermore, state-of the-art algorithms such as the one that use a radial variance based hash function as loop detector may be extended to image to map comparison, to perform suitably well in terms of loop detection with low computational complexity. The proposed algorithm is implemented on K. N. Toosi mobile robot in a step by step implementation hierarchy, by which the importance of adding each step to the algorithm is elaborated. Graphical and numerical results are reported for each step of the extended algorithm, by which it is verified that the proposed algorithm works suitably well with RGB-D data from Kinect camera. Furthermore, it is shown that the required execution time needed for each step is such that the algorithm is promising for implementation in real time with current graphical processing unit capabilities.

REFERENCES

[1] M. Cummins and P. Newman, Appearance-only SLAM at large scale with FAB-MAP 2.0, The International Journal of Robotics Research , vol. 30, no. 9, August 2011.

[2] Gálvez-López, Dorian and Tardos, Juan D, "Bags of Binary Words for Fast Place Recognition in Image Sequences", *IEEE Transactions on Robotics*, 2012.

[3] Niko Sunderhauf and Peter Protzel, Switchable Constraints for RobustPose Graph SLAM,*In Proc. of IEEE International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, 2012.

[4] Edwin Olson and Pratik Agarwal, Inference on Networks of Mixturesfor Robust Robot Mapping, *Int. J. of Robotics Research (IJRR)*, RSS Special Issue, 2013

[5] Yasir Latif, Csar Cadena and Jos Neira, Robust Loop Closing Over time for Pose Graph SLAM, *The International Journal of Robotics Research* , vol. 32, no. 14, December 2013

[6] Huang, Albert S., Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. "Visual odometry and mapping for autonomous flight using an RGB-D camera." *In Int. Symposium on Robotics Research (ISRR)*, (Flagstaff, Arizona, USA). 2011.

[7] Lefebvre, F., Macq, B., and Legat, J.D., RASh: RAdon Soft Hash algorithm, *In Proceedings of the European Signal Processing Conference (EUSIPCO)*, vol. I, Sept. 2002.

[8] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert, iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree, *Intl. J. of Robotics Research*, IJRR, vol. 31, Feb. 2012.

[9] Niko Sunderhauf and Peter Protzel, Switchable constraints vs. max-mixture models vs. RRR - A comparison of three approaches to robust pose graph SLAM, *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013

[10] M. Milford and G. Wyeth, "SeqSLAM: Visual Route-Based Navigation for Sunny Summer Days and Stormy Winter Nights, *IEEE International Conference on Robotics and Automation (ICRA)*, Saint Paul MN, 2012

[11] D. Boley and R. Maier, "Robust Real-Time Visual Odometry for Dense RGB-D Mapping", *in Third SIAM Conference on Applied Linear Algebra*, Madison, WI, 1988, pp. A20.

[12] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti, and P. Taddei, awseeds groundtruth collection systems for indoor self-localization and mapping, *Autonomous Robots*, 27(4):353371, 2009.

[13] Kuemmerle, Rainer, 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.

[14] Zauner, Christoph: Implementation and Benchmarking of Perceptual Image Hash Functions. Master's thesis, Upper Austria University of Applied Sciences, Hagenberg Campus, 2010.

[15] PointClouds official website: http://www.pointclouds.org.