2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Robust Pose-Graph Loop-Closures with Expectation-Maximization

Gim Hee Lee[1], Friedrich Fraundorfer[2], and Marc Pollefeys[1]

[1]Computer Vision and Geometry Lab, Department of Computer Science, ETH Zürich
[2]Remote Sensing Technology, Faculty of Civil Engineering and Surveying, Technische Universität München
glee@student.ethz.ch, friedrich.fraundorfer@tum.de, marc.pollefeys@inf.ethz.ch

***Abstract*— In this paper, we model the robust loop-closure pose-graph SLAM problem as a Bayesian network and show that it can be solved with the Classification Expectation-Maximization (EM) algorithm. In particular, we express our robust pose-graph SLAM as a Bayesian network where the robot poses and constraints are latent and observed variables. An additional set of latent variables is introduced as weights for the loop-constraints. We show that the weights can be chosen as the Cauchy function, which are iteratively computed from the errors between the predicted robot poses and observed loop-closure constraints in the Expectation step, and used to weigh the cost functions from the pose-graph loop-closure constraints in the Maximization step. As a result, outlier loop-closure constraints are assigned low weights and exert less influences in the pose-graph optimization within the EM iterations. To prevent the EM algorithm from getting stuck at local minima, we perform the EM algorithm multiple times where the loop constraints with very low weights are removed after each EM process. This is repeated until there are no more changes to the weights. We show proofs of the conceptual similarity between our EM algorithm and the M-Estimator. Specifically, we show that the weight function in our EM algorithm is equivalent to the robust residual function in the M-Estimator. We verify our proposed algorithm with experimental results from multiple simulated and real-world datasets, and comparisons with other existing works.**

I. Introduction

The focus of many existing works [1], [2] on the back-end pose-graph SLAM problem is on improving the efficiency of the optimization algorithms. Most of these optimization algorithms assumed that the constraints provided by the front-end are free from errors, and would fail if this assumption was violated. For most cases, these errors are from erroneous loop-closure constraints. Erroneous loop-closure constraints are the result of wrong place recognitions by the appearance or vocabulary-tree based approaches and this problem is aggravated in environments with highly repetitive scenes. Despite the numerous efforts [3], [4] to improve the accuracy of the front-end recognition, none of these algorithms is totally free from false positives. The task of identifying erroneous loop-closure constraints is always left to the front-end, and it is only until the recent two years that several works [5]–[9] demonstrated the ability robustly detect and disable erroneous loop-closure constraints within the back-end optimization process.

In this paper, we propose a robust pose-graph SLAM optimization algorithm based on the Classification EM algorithm [10] to robustly detect and minimize the influences from the erroneous loop-closure constraints within the optimization process. In particular, we express our robust pose-graph SLAM as a Bayesian network where the robot poses and constraints are latent and observed variables. An additional set of latent variables is introduced as weights for the loop-constraints. We show that the weights can be chosen as the Cauchy function, which are iteratively computed from the errors between the predicted robot poses and observed loop-closure constraints in the Expectation step, and used to weigh the cost functions from the pose-graph loop-closure constraints in the Maximization step. As a result, outlier loop-closure constraints are assigned low weights and exert less influences in the pose-graph optimization within the EM iterations. To prevent the EM algorithm from getting stuck at local minima, we perform the EM process multiple times where the loop constraints with very low weights are removed after each EM process. This is repeated until there are no more changes to the weights. We show proofs of the conceptual similarity between our EM algorithm and the M-Estimator [11]. Specifically, we show that the weight function in our EM algorithm is equivalent to the robust residual function in the M-Estimator. We verify our proposed algorithm with experimental results from multiple simulated and real-world datasets, and comparisons with other existing works. An example of the results from our algorithm is

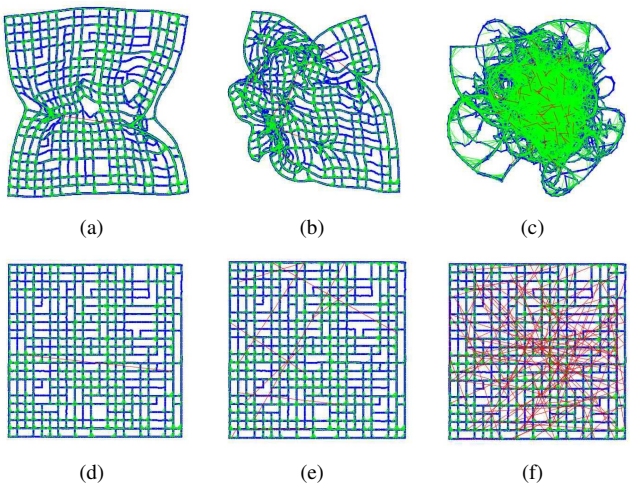


Fig. 1. (a)-(c) City1000 dataset randomly corrupted with 1, 10 and 100 outlier loop-closure constraints (red lines) leading to wrong convergence with standard pose-graph SLAM. (d)-(f) Our EM algorithm detects all the outliers and converges to the correct solution. Green and blue lines are the correct loop-closure and odometry constraints.

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

shown in Figure 1.

II. Related Works

Back-end optimizers are usually standard non-linear least squares methods that try to minimize the sum-of-squares residuals. The rapid gain of the quadratic error function in both ends means that any outlier that is present would assert a strong influence on the optimizer and lead to a wrong solution. A popular approach to reduce the effect of outliers within the optimizer is the M-Estimator Huber robust cost function [11]. The Huber function replaces the quadratic error function with another error function which is quadratic in the vicinity of zero but increases linearly when the error is above a certain threshold known as the Huber kernel width. The linear gain of errors above the Huber kernel width helps to reduce the influence of outliers within the optimizer. The Huber function is implemented as an option within the state-of-the-art SLAM solver g2o [2]. However, it has been shown in [5] that the Huber function is only capable of reducing the effects of outliers and not removing them completely in pose-graph SLAM thus causing pose-graphs with erroneous loop constraints to converge to wrong solutions (see Section IV-A for our explanations on Huber robust cost).

In [5], Sünderhauf $et\ al.$ introduced the so-called switch variables to each loop constraints. The switch variable is used as a parameter in the weight for the cost function of the loop-closure constraint and lies within the range of $[0, 1]$. The switch variable for an erroneous loop constraint would result in a low weight for the cost function of the loop-closure constraint thus reducing or removing the effect of the wrong loop constraint within the optimization. The weighting function is chosen arbitrarily as a sigmoid function. In a further work [6], Sünderhauf $et\ al.$ suggested experimentally that a linear function is a better choice than the sigmoid function. A penalty cost is introduced for each switch variable to prevent a trivial solution of zero weight for all loop constraints and the penalty term was chosen empirically. Standard non-linear least squares optimizer such as the Levenberg-Marquardt algorithm is used to jointly optimize both the pose-graph SLAM and switch variables. In a more recent work [9], Agarwal $et\ al.$ showed that the joint optimization of the pose-graph SLAM and the switch variable is equivalent to iteratively re-evaluating the switch variable with the Geman-McClure function. In contrast, we show that solving the problem with Classification EM allows us to naturally select the weight as a function of the error between the robot poses and observed loop-constraints, in particular the Cauchy function, thus avoiding the needs to arbitrarily assign the weighting function and penalty term. We also show in Section V-A.1 that our algorithm outperforms [5].

In [7], Latif $et\ al.$ proposed the RRR algorithm which detects and removes wrong loop-closure edges by evaluating the "goodness-of-fit" from the non-linear least squares pose-graph optimization using the χ^2 test. The loop-closure edges are segmented into clusters according to its spatial arrangements. Intra-cluster consistency check is done by multiple pose-graph optimizations with respect to each single cluster while disabling the rest. Individual loop-closure edge that does not pass the χ^2 for individual edge is removed. Similarly, a cluster is removed if it does not pass the χ^2 test for a cluster. The algorithm tests for inter-cluster consistency after the intra-consistency checks. Clusters are further grouped into subsets of clusters and multiple pose-graph optimizations are carried out to test for joint consistency of each subset of clusters. Subsets of clusters which are found to be jointly consistent from the χ^2 test for subsets of clusters and passing a threshold test are grouped as $goodSet$ while those found to be jointly inconsistent are grouped as $rejectSet$. The joint consistency checks are repeated for the remaining subsets of clusters which passed the joint consistency check but failed the threshold test. The final subsets of clusters in the $goodSet$ are all the correct loop-closure edges. Extensive experimental results showed the reliability of the RRR algorithm. However, the need to perform multiple pose-graph optimizations during the intra- and inter-consistency checks makes the algorithm slow. In addition, the algorithm makes fix assignments of the wrong loop-closure edges and there is no chance of re-verifying them further in the intra- and inter-consistency checks.

The Max-Mixture model was proposed by Olson $et\ al.$ in [8] as a replacement to the Corrupted Gaussian model (Gaussian Mixture model) as the robust cost function. The Max-Mixture model consists of a front-end loop-closure and null hypotheses. The front-end loop-closure hypothesis represents the distribution of the inlier loop-closure constraints and the null hypothesis represents the distribution of the outlier loop-constraints. Each loop-closure constraint is verified against the hypotheses iteratively within the optimization loops and the weight associated with the most likely hypothesis is used to scale the Jacobian, residual and information matrix from that loop-closure constraint. In other words, the hypothesis testing acts as an "selector" to the weighting of the loop-closure constraint. The Max-Mixture model requires the specification of the covariance of the outlier distribution for the null hypothesis which is difficult to quantify since the outlier distribution is usually unknown. It was also shown in [8] that the algorithm fails when the outlier to inlier ratio becomes too high. We show examples of this failure case in Section V-A.2 and verify experimentally that our proposed algorithm has a higher tolerance to high outlier to inlier ratio.

III. Robust SLAM with EM

Our robust back-end pose-graph SLAM can be represented by the Bayesian network shown in Figure 2. $X = [x_1, x_2, ..., x_n]^T$ are the robot poses and $Z = [z_{12}, ..., z_{i,i+1}, z_{i,j}]^T, j \neq i+1$ are the odometry constraints $z_{i,i+1}$, and loop-closure constraints $z_{i,j}$. For brevity, we will drop the notation $j \neq i+1$ in the rest of the equations and assume that the indices $\{i, j\}$ always comes with this condition. The variables $W = [..., w_{i,j}],\ w \in [0, 1]$ are the weights to the loop-constraints $z_{i,j}$. The values in W determines the weight of the loop-constraints in the optimization. X and W are latent variables and Z is an observed variable. Formally,

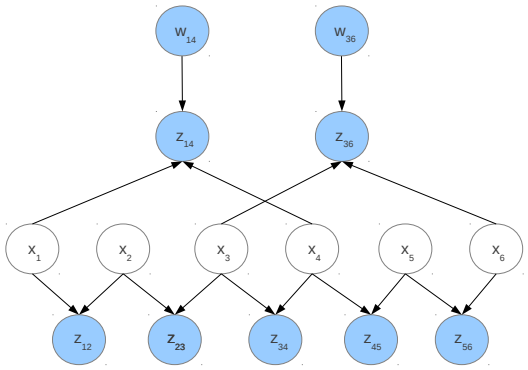


Fig. 2. Bayesian network representing the pose-graph problem. $w_{i,j}$ is the weight for the loop constraint $z_{i,j}$ in the optimization.

the problem of pose-graph optimization involves finding the Maximum a Posterior (MAP) solution of $p(X|Z)$ and this means marginalizing out the latent variable W as given in Equation (1).

$$p(X|Z) = \int_W p(X, W|Z)\ dW \tag{1}$$

One possible way to find the MAP solution to $p(X|Z)$ is to use the EM algorithm. The EM algorithm is an iterative algorithm that iterates between the Expectation and Maximization steps. In the Expectation step, the current X^k is used to find the posterior distribution of the latent variable W given by Equation (2a). This posterior distribution is then used to find the next X_{k+1} by maximizing the complete log likelihood given by Equation (2b) in the Maximization step.

$$p(W|X^k, Z) \tag{2a}$$

$$X^{k+1} = \underset{X}{\operatorname{argmax}} \int_W p(W|X^k, Z) \ln p(X|W, Z)\ dW \tag{2b}$$

The main drawback of the EM solution is the computation of $p(W|X^k, Z)$ grows exponentially with the size of W and becomes intractable. As suggested by [12], an alternative solution would be to use the Classification EM algorithm [10]. In the Expectation step given by Equation (3a), we compute W^{k+1} by maximizing $p(W|X^k, Z)$ instead of evaluating $p(W|X^k, Z)$ explicitly. With W^{k+1} known from the Expectation step, X^{k+1} can be found by maximizing the log-likelihood $\ln p(X|W^{k+1}, Z)$ as given by Equation (3b) in the Maximization step. The Maximization step can also be written into a minimization problem by appending a negative sign to the log-likelihood and this turns the problem into the usual pose-graph SLAM optimization.

$$W^{k+1} = \underset{W}{\operatorname{argmax}}\ p(W|X^k, Z) \tag{3a}$$

$$\begin{aligned} X^{k+1} &= \underset{X}{\operatorname{argmax}}\ \ln p(X|W^{k+1}, Z) \\ &= -\underset{X}{\operatorname{argmin}}\ \ln p(X|W^{k+1}, Z) \end{aligned} \tag{3b}$$

A. Expectation Step

The task in Equation (3a) is to find the correct values of $w_{i,j} \in [0, 1]$ given the current estimate of the robot poses X^k and observations Z. Applying Bayes rule on $p(W|X^k, Z)$ and assuming that all observations Z are independent, we have the following relation:

$$\begin{aligned} p(W|X^k, Z) &\propto p(Z|W, X^k) = \prod_{ij} p(z_{i,j}|w_{i,j}, x_i^k, x_j^k) \\ &\propto \prod_{ij} exp\{-w_{i,j}(h(x_i^k, x_j^k) - z_{i,j})^T Q_{i,j}^{-1}(h(x_i^k, x_j^k) - z_{i,j})\} \end{aligned} \tag{4}$$

where $Q_{i,j}$ is the error covariance of the loop constraint observation $z_{i,j}$ and $h(.)$ is the observation model. Putting Equation 4 back into Equation 3a and taking the negative log-likelihood, the Expectation step becomes

$$W^{k+1} = \underset{W}{\operatorname{argmin}} \sum_{ij} w_{i,j} ||h(x_i^k, x_j^k) - z_{i,j}||^2_{Q_{i,j}} \tag{5}$$

However, a trivial solution of $W = 0$ exists for the minimization of Equation 5. In order to circumvent this problem, we introduce a penalty term $-C^2(\ln w_{i,j} - w_{i,j})$ to Equation 5, which penalizes the cost as $w_{i,j}$ goes to 0. This turns Equation 5 into

$$\begin{aligned} W^{k+1} = \underset{W}{\operatorname{argmin}} \sum_{ij} & w_{i,j} ||h(x_i^k, x_j^k) - z_{i,j}||^2_{Q_{i,j}} \\ & -C^2(\ln w_{i,j} - w_{i,j}) \end{aligned} \tag{6}$$

where C is a constant. It is important to note that we do not chose the penalty term arbitrarily, but it is chosen such that the cost function in Equation 6 becomes convex with the values for $w_{i,j}$ that gives the minimal cost bounded between the range of $[0, 1]$ as the Mahalanobis distance $||h(x_i^k, x_j^k) - z_{i,j}||^2_{Q_{i,j}}$ changes. Differentiating Equation 6 and setting it to 0 gives us

$$w_{i,j}^{k+1} = \frac{C^2}{C^2 + ||h(x_i^k, x_j^k) - z_{i,j}||^2_{Q_{i,j}}} \tag{7}$$

which is the Cauchy function where C is a constant that corresponds to the half maximum at $w_{i,j}^{k+1} = 0.5$. Figure 3 shows the Cauchy weighting function at $C = 0.01$m. The value of C determines the range of the Mahalanobis distances to be considered as inliers. It becomes apparent from the Cauchy function that the weight gradually decreases from a maximum value of 1 to 0 with increasing error and this means that the loop-constraints are down-weighted gradually as the errors increase. We shall see from the results in Section V that this smooths out loop-constraints which are correct but corrupted with noises. In addition, the near-zero values at both ends of the Cauchy function serve the purpose of suppressing the bad effects from the outlier loop-constraints.

B. Maximization Step

From the Bayesian Network shown in Figure 2, we have the following relation:

$$\begin{aligned} p(X|W^{k+1}, Z) \propto \\ \underbrace{\prod_i p(z_{i,i+1}|x_i, x_{i+1})}_{Odometry\ Constraints} \underbrace{\prod_{ij} p(z_{i,j}|w_{i,j}^{k+1}, x_i, x_j)}_{Loop\ Constraints} \end{aligned} \tag{8}$$

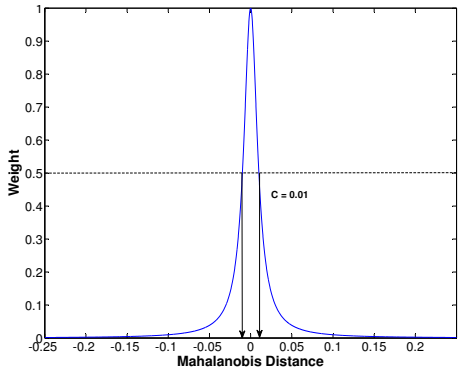


Fig. 3. The Cauchy weighting function with $C = 0.01$.

where $p(z_{i,i+1}|x_i, x_{i+1})$ and $p(z_{i,j}|w_{i,j}^{k+1}, x_i, x_j)$ are the odometry and loop constraints. Assuming that X and Z are random variables that follow the Gaussian distribution, we can write

$$p(z_{i,i+1}|x_i, x_{i+1}) \propto exp(-\Delta z_{i,i+1}^T P_{i,i+1}^{-1} \Delta z_{i,i+1}) \quad (9a)$$

$$p(z_{i,j}|w_{i,j}, x_i, x_j) \propto exp(-w_{i,j}^{k+1} \Delta z_{i,j}^T Q_{i,j}^{-1} \Delta z_{i,j}) \quad (9b)$$

where $P_{i,i+1}$ is the error covariance of the odometry observations $z_{i,i+1}$. $w_{i,j}^{k+1}$ is the weight variable found from the Expectation step. It acts as a scaling factor to the information matrix $Q_{i,j}^{-1}$ of the loop constraint since it takes a value ranging from $[0, 1]$. Therefore, a lower weight implies lower influence of the loop-constraint. $\Delta z_{i,i+1}$ and $\Delta z_{i,j}$ are defined in Equation (10) and these are the error terms between the observed relative pose z and the predicted relative pose from the observation model $h(.)$.

$$\Delta z_{i,i+1} = h(x_i, x_{i+1}) - z_{i,i+1} \quad (10a)$$

$$\Delta z_{i,j} = h(x_i, x_j) - z_{i,j} \quad (10b)$$

Putting Equations (8), (9) and (10) into Equation (3b), we obtain

$$X^{k+1} = \underset{X}{\operatorname{argmin}} \sum_i ||h(x_i, x_{i+1}) - z_{i,i+1}||^2_{P_{i,i+1}} + \sum_{ij} w_{i,j}^{k+1} ||h(x_i, x_j) - z_{i,j}||^2_{Q_{i,j}} \quad (11)$$

We solve for X^{k+1} as a standard non-linear least squares problem given by

$$J^T \Lambda J \delta = -J^T \Lambda \epsilon \quad (12a)$$

$$X^{k+1} = X^k + \delta \quad (12b)$$

where J is a Jacobian matrix that consists of the Jacobian of $h(x_i, x_{i+1})$ and $h(x_i, x_j)$. ϵ is the error term given by $\Delta z_{i,i+1}$ and $\Delta z_{i,j}$. Λ is the weight for the non-linear least squares and it is a diagonal matrix formed by the information matrices $P_{i,i+1}^{-1}$ and $\hat{Q}_{i,j}^{-1} = w_{i,j}^{k+1} Q_{i,j}^{-1}$. It is important to note that, unlike standard non-linear least squares, we do not solve Equation 12 iteratively until convergence. Instead, we solve Equation 12a and update Equation 12b only once within the Maximization step. As mentioned in [12], [13], making an one step update in the Maximization step prevents the EM optimization from getting stuck at local minima and improves the convergence rate.

C. Summary

Algorithm 1 Pose-graph SLAM with Robust Loop-Closure

Require: Poses X, observations Z, loop vertex pairs $\mathcal{I}$
Ensure: Corrected poses X, Weight variables W.

1: $W' = 0$; $W = 1$; $\mathcal{I}' = \mathcal{I}$;
2: **while** $|W - W'| > \nu$ **do**
3: $W' = W$;
4: // Classification EM iterations
5: $k = 0$; $X^k = X$;
6: **while** $\mid \delta \mid > \eta$ **do**
7: // Expectation Step
8: **for all** loop vertex pairs $\{i, j\} \in \mathcal{I}'$ **do**
9: Compute $w_{i,j}^{k+1}$ with Equation 7;
10: **end for**
11: // Maximization Step, $\{i, j\} \in \mathcal{I}'$
12: Form Λ with $P_{i,i+1}^{-1}$ and $\hat{Q}_{i,j}^{-1} = w_{i,j}^{k+1} Q_{i,j}^{-1}$;
13: Compute J and ϵ with X^k and Z;
14: Solve for δ in $J^T \Lambda J \delta = -J^T \Lambda \epsilon$;
15: Update $X^{k+1} = X^k + \delta$;
16: $k = k + 1$;
17: **end while**
18: // Remove loop constraints with low weight
19: **for all** loop vertex pairs $\{i, j\} \in \mathcal{I}$ **do**
20: Compute w_{ij}^k with Equation 7;
21: **if** $w_{ij}^k < \omega$ **then**
22: Remove loop vertex pair $\{i, j\}$ from $\mathcal{I}'$;
23: **end if**
24: **end for**
25: $X = X^k$; $W = W^k$;
26: **end while**
27: **return** X, W;

Algorithm 1 shows the pseudo code of our pose-graph SLAM with robust loop-closure. Lines 8 to 10 are the Expectation step where the weights $w_{i,j}^{k+1}$ are computed based on the current pose X^k. It is important to note that the weights $w_{i,j}^{k+1}$ are not kept fixed but are update within the EM iterations based on the current pose X^k. Lines 12 to 16 are the Maximization step where the pose update X^{k+1} is computed from the weights $w_{i,j}^{k+1}$ and current pose X^k. Intuitively, our algorithm re-evaluates the weights $w_{i,j}^{k+1}$ based on the current poses X^k and observations Z, and scales the updated poses X^{k+1} according to these weights at every iteration. The scaling is done in Line 12 where the loop-closure information matrices $Q_{i,j}^{-1}$ are scaled with the weights $w_{i,j}^{k+1}$. We repeat the EM process until there are no more changes to the weights in Line 2. Lines 19 to 24 check and remove loop constraints with very low weights, i.e. high confidence to be outliers from the next EM process. The repeated EM process and removal of loop constraints with very low weights help to prevent local minima where the EM algorithm terminates before all outlier loop constraints are removed.

IV. Relation with M-Estimators

Our proposed algorithm can also be viewed as solving an iteratively re-weighted least-squares problem where the weights are re-evaluated from the error between the robot poses and observed loop-constraints $\Delta z = h(.) - z$ at every iteration. We shall see that our algorithm which is based on EM turns out to be conceptually similar to the M-Estimators. The M-Estimators reduces the effect of outliers by replacing the usual squared error Δz^2 term with a robust cost function $\rho(\Delta z)$. In the context of a robust pose-graph optimization, we have the following

$$X^{k+1} = \underset{X}{\operatorname{argmin}} \sum_{i} \Delta z_{i,i+1}^2 + \sum_{ij} \rho(\Delta z_{i,j}) \tag{13}$$

where the first term is the odometry constraint and the second term is the loop-constraints. Let us look at the second term where the solution to the minimization is given by solving for X after differentiating the second term and setting it zero.

$$\sum_{ij} w(\Delta z_{i,j}) \Delta z_{i,j} \frac{\partial(\Delta z_{i,j})}{\partial X} = 0 \tag{14}$$

where $w(\Delta z_{i,j})\Delta z_{i,j}$ is obtained from differentiating the robust cost function $\rho(\Delta z)$ and multiplying it by $\Delta z_{i,j}$. It can be immediately observed that solving Equation 14 is the same as optimizing the following

$$\underset{X}{\operatorname{argmin}} \sum_{ij} w(\Delta z_{i,j}) \Delta z_{i,j}^2 \tag{15}$$

Putting the results from Equation 15 back into Equation 13, we get

$$X^{k+1} = \underset{X}{\operatorname{argmin}} \sum_{i} \Delta z_{i,i+1}^2 + \sum_{ij} w(\Delta z_{i,j}) \Delta z_{i,j}^2 \tag{16}$$

which is also an iteratively re-weighted least-squares where $w(\Delta z_{i,j})$ is the weight evaluated from the error $\varepsilon_{i,j}$ at every iteration similar to our method given in Equation 11. This means that Lines 5 to 17 of Algorithm 1 can also be implemented as an M-Estimator with $\rho(\Delta z) = \int w(\Delta z)\Delta z \; d(\Delta z)$, where $w(\Delta z)$ is given by the Cauchy function in Equation 7.

A. Why Huber Robust Cost Fails?

The obvious question that follows after proving that our algorithm based on EM is conceptually similar to the M-Estimators is why does the commonly used robust Huber cost function fails for pose-graph optimization with outliers while our proposed algorithm works? The answer is in the choice of the weight function. Figure 4 shows a plot of the Huber weight function with a kernel width $C = 0.01m$. We see that the Huber weight function assigns a considerable weight to the loop-constraints with high errors on both ends of the plot. This means that the outlier loop-constraints are still exerting considerable influences from the Huber weight on the pose-graph optimization thus rendering it to failure.

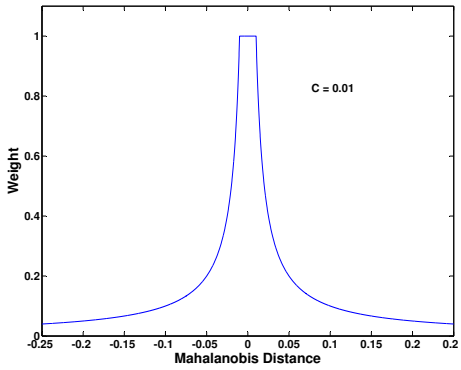


Fig. 4. The Huber weighting function with $C = 0.01$.

In comparison, the Cauchy weight function proposed in this paper assigns near-zero weight thus suppressing outliers with high errors.

V. Results and Evaluations

We implement our algorithm with the Google Ceres solver [14] and evaluate it with multiple simulated and real-world datasets. Here, simulated datasets refer to datasets where the outlier loop-constraints are simulated and real-world datasets refer to datasets with loop-constraints obtained from real robots and sensors. $C = 1.0m$ for all datasets except for the ParkingGarage, Carpark01 and Carpark02 datasets where $C = 0.01m$.

A. Simulated Datasets

TABLE I
Simulated Datasets Used for Evaluations

Dataset	2D/3D	Vertices	Loop-Edges
City10000	2D	10000	10688
Intel	2D	943	894
Manhattan3500 (Olson)	2D	3500	2099
Manhattan3500 (g2o)	2D	3500	2099
Sphere2500	3D	2500	2450
ParkingGarage	3D	1661	4615

Similar to [6], we make the evaluations of our robust pose-graph algorithm with six different Open-Source pose-graph datasets shown in Table I. We obtained the first five datasets from Vertigo SLAM [1] and the ParkingGarage dataset from g2o. The Intel and ParkingGarage datasets were collected from real robots while the remaining datasets were simulated. We simulate outliers loop constraints to corrupt the datasets based on the four different policies mentioned in [6] - (a) Random Constraints, (b) Local Constraints, (d) Randomly Grouped Constraints and (e) Locally Grouped Constraints. The simulation of the outliers are done using the script which is provided by Vertigo SLAM. We compare our algorithm with previous approaches - (a) Switchable Constraints [6] and (b) Max-Mixture model [8] based on the Open-Source implementations Vertigo SLAM. Note that we classify a loop-closure constraint as outlier if the error is larger than a given threshold after optimization.

[1] http://openslam.org/vertigo.html

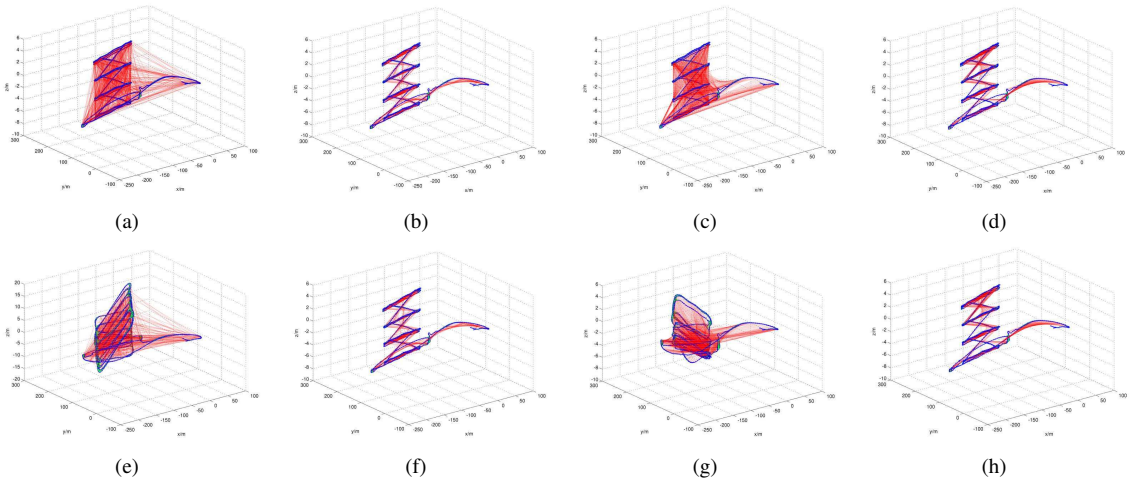


Fig. 5. Results from the ParkingGarage dataset comparing our method (top row) with the Switchable Constraints method (bottom row). (a)-(e) 1000 Random Constraints, (b)-(f) 1000 Local Constraints, (c)-(g) 50 sets of 20 Randomly Grouped Constraints, and (d)-(h) 50 sets of 20 Locally Grouped Constraints.

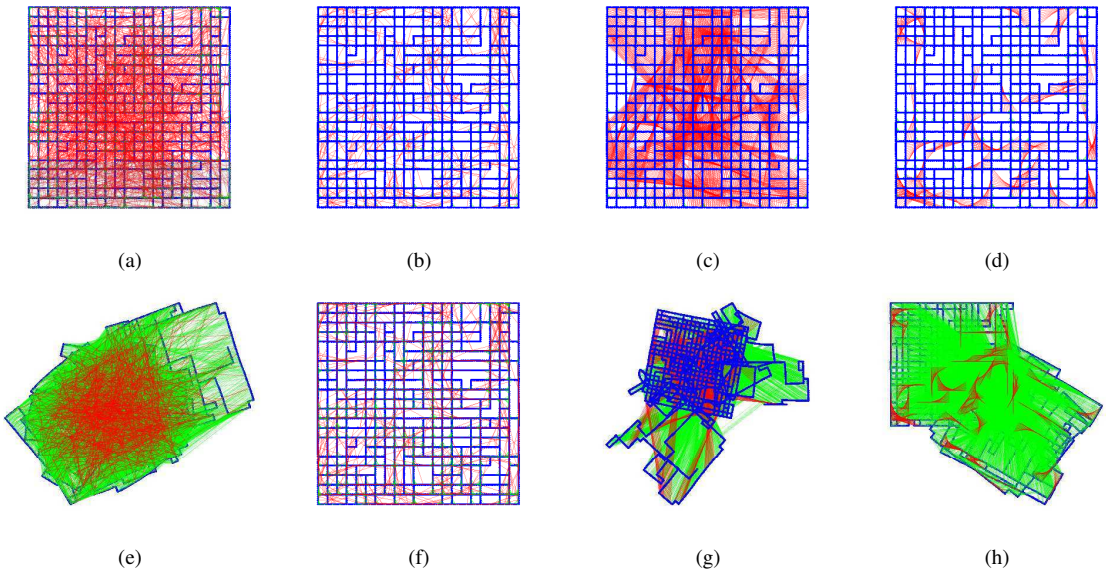
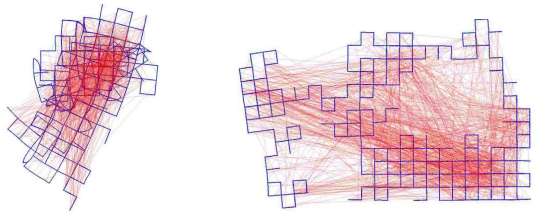


Fig. 6. Results from the City10000 dataset comparing our method (top row) with the Max-Mixture model (bottom row). (a)-(e) 1000 Random Constraints, (b)-(f) 1000 Local Constraints, (c)-(g) 50 sets of 20 Randomly Grouped Constraints, and (d)-(h) 50 sets of 20 Locally Grouped Constraints.

1) Comparison with Switchable Constraints: It was shown in [6] that the Switchable Constraints method fails with the ParkingGarage dataset. We re-create the results with the Vertigo SLAM for comparisons. Figure 5(a) and 5(e) shows the results from our algorithm and Switchable Constraints method. The dataset is corrupted with 1000 Random Constraints shown as red lines. The correct loop-closure constraints are shown as green lines. Note that the z-axis in the figures are scaled up to have a clearer view of the results. We can see from Figure 5(a) that the pose-graph converges to the correct solution with our method. All the outlier loop constraints are correctly identified. The pose-graph however fails to converge to a correct solution with the Switchable Constraints method in Figure 5(e). In particular, the failure occurs at the connections between different levels of the parking garage where there are more wrong than correct loop constraints. Similar failure for the Switchable Constraints method is observed in Figure 5(g) when the dataset is corrupted with 50 sets of 20 Randomly Grouped Constraints. In contrast, Figure 5(c) shows the correct convergence of the pose-graph with our algorithm. Results from Figure 5(b) and 5(f), and Figure 5(d) and 5(h) show that both our algorithm and the Switchable Constraints method work well for 1000 Local Constraints and 50 sets of 20 Locally Grouped Constraints outliers. The reason for the success of our algorithm on the ParkingGarage dataset is because our method iteratively discriminates against the outliers by the weights computed from the errors as the pose-graph converges. In comparison, the Switchable Constraints method optimizes for the switch values within the non-linear least squares without taking the errors into account and this proves to be detrimental at the connections between different levels of the parking garage where correct loop-closure constraints are scarce.

2) Comparison with Max-Mixture Model: It was mentioned in [8] that the success of the Max-Mixture model depends on the outliers to inliers ratio. The Max-Mixture model fails when this ratio becomes too high. Figure 6(a) and 6(e) show the results from the City10000 dataset corrupted with 1000 Random Constraints with our algorithm and Max-Mixture model. The red lines are the outlier loop constraints and green lines are the correct loop constraints. Figure 6(a) shows a correct convergence of the pose-graph with our method. All the outliers are correctly detected. Figure 6(e) shows the wrong convergence of the pose-graph with the Max-Mixture model. Failures to converge to the correct solutions are also observed for the Max-Mixture model with the City10000 dataset corrupted 50 sets of 20 Randomly Grouped Constraints and Locally Grouped Constraints in Figures 6(g) and 6(h). The Max-Mixture model converges to the correct solution with the City10000 dataset corrupted with Local Constraints shown in Figure 6(f). In contrast, our algorithm correctly detects all the outliers and converged to the correct pose-graph with outliers generated from all the four different policies. The results from this comparison suggest that our algorithm is able to suppress the influence from the outliers better than the Max-Mixture model even when the outliers to inliers ratio is high.

3) Relative Pose Metric Errors: We run our algorithm on all the datasets shown in Table I corrupted with simulated outliers. For each dataset, we generate 10 times each with 1000 outliers based on the four different policies, i.e. 6 x 10 x 4 corrupted datasets with 1000 outliers each. We compare the results from our algorithm with the results from the outlier-free datasets since we do not have the ground truth. The relative pose metric error from [15] is computed. Our algorithm had successfully detected the outliers and converged to the correct solutions for all the datasets with 100% success rate. The maximum relative pose metric error from all results is 3.84×10^{-5}. Figure 7(a) shows an example of the Manhattan3500 (Olson) dataset corrupted with 1000 Random Constraint outliers stuck at a local minima after the first EM process. Figure 7(b) shows the correct convergence after the third EM process.

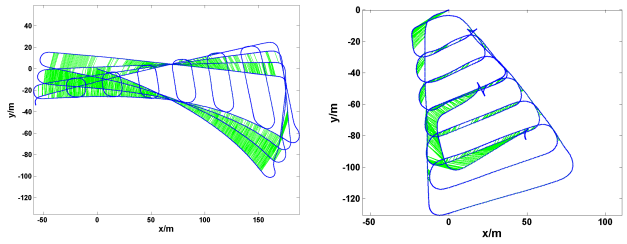


(a) (b)

Fig. 7. Manhattan3500 (Olson) dataset corrupted with 1000 Random Constraint outliers. (a) Local minima after the first EM process with 5 outlier constraints wrongly classified. (b) Correct convergence after the third EM process.

B. Real-World Datasets

We also make evaluations of our algorithm on two real-world datasets - Carpark01 and Carpark02. These datasets were collected with cameras mounted on a car which was driven approximately 3.5km and 1km around two different parking garages. The pose-graphs for both datasets are formed using the wheel odometry readings. The loop-closures constraints are computed with a vocabulary-tree based [16] place recognizer and geometric verification. The Carpark01 dataset consists of 3997 vertices and 1352 loop-closure edges. The Carpark02 dataset consists of 2180 vertices and 642 loop-closure edges. Both datasets are equipped with the GPS/INS readings as the ground truth. Figures 8(a) and 8(b) show the pose-graphs of the two datasets before loop-closure. In contrast to the simulated datasets, the real-world datasets possess very few outlier loop-constraints in grossly wrong locations because the place recognizer and geometric verification are usually capable of identifying and removing these outliers. Instead, we observed that the main source of error comes from the uncertainties in the estimation of the relative poses for the loop-constraints. Figures 9(a) and 10(a) show the pose-graphs from the two datasets after non-robust loop-closures. It can be seen that in addition to a few outliers that caused huge errors in the pose-graph, the uncertainties in the estimation of the relative poses for the loop-constraints caused small kinks to appear in the pose-graph. Figures 9(b) and 10(b) show the results from the Max-Mixture model [8]. Figures 9(c) and 10(c) show the results from the Switchable Constraints [6]. The plots clearly show that both methods detected the outliers but are unable to smooth out the uncertainties that are present in the loop-constraints. In comparison, Figures 9(d) and 10(d) show that our algorithm is able to handle the uncertainties in the estimation of the relative poses for the loop-constraints because the pose-graphs after loop-closure clearly align with the GPS/INS ground truth for both datasets.



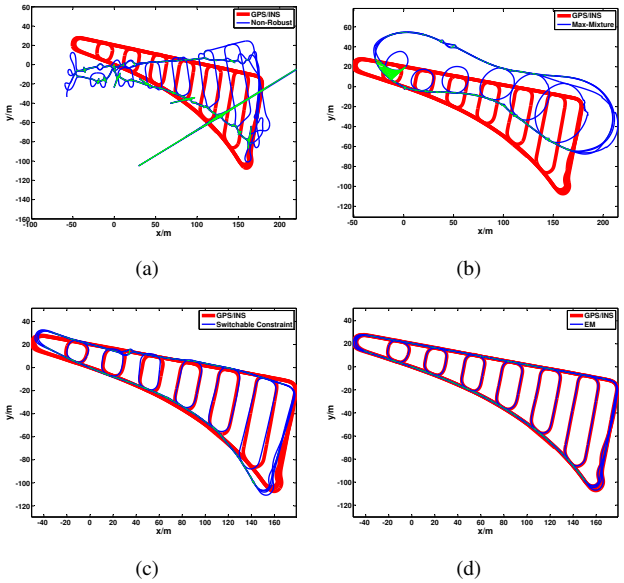
(a) (b)

Fig. 8. Pose-graph before loop-closure. Loop-constraints are shown in green. (a) Carpark01 dataset. (b) Carpark02 dataset.

C. Runtime Performance

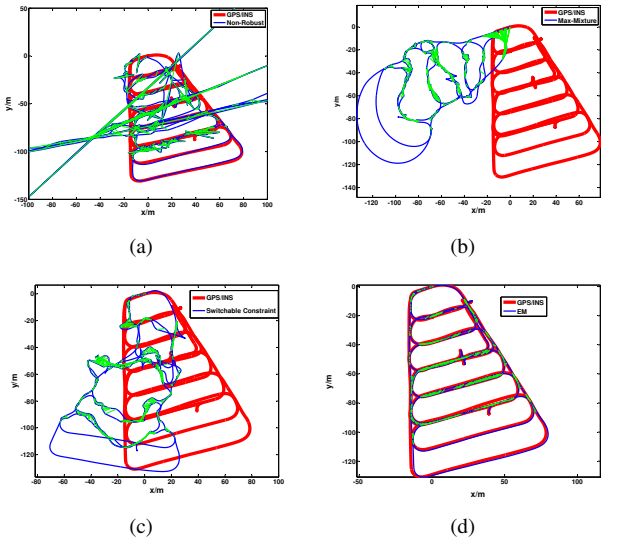
TABLE II
CONVERGENCE TIME COMPARISONS

Dataset	Runtime (secs) Non-Robust	EM
City10000	2.2701	2.2926
Intel	0.0813	0.0819
Manhattan3500 (Olson)	0.9900	1.0047
Manhattan3500 (g2o)	0.6662	0.6675
Sphere2500	16.6994	17.1384
ParkingGarage	4.43296	4.49716



(a) (b) (c) (d)

Fig. 9. Results from Carpark01 dataset. (a) Pose-graph after non-robust loop-closure. (b) Pose-graph after Max-Mixture model loop-closure. (c) Pose-graph after Switchable Constraints loop-closure. (d) Pose-graph after loop-closure with our algorithm.



(a) (b) (c) (d)

Fig. 10. Results from Carpark02 dataset. (a) Pose-graph after non-robust loop-closure. (b) Pose-graph after Max-Mixture model loop-closure. (c) Pose-graph after Switchable Constraints loop-closure. (d) Pose-graph after loop-closure with our algorithm.

Table II shows the time taken for each dataset to converge to the correct solution with the non-robust pose-graph optimization and our algorithm. Both non-robust pose-graph optimization and our algorithm are implemented with Google Ceres solver [14]. The tests were run on a Intel Core2 Quad CPU @ 2.40GHz x 4 computer and the datasets were not corrupted with any outlier. It is important to note that our algorithm still computes the weights in the Expectation steps because it does not have any prior information that none of the loop constraint is an outlier. Our algorithm is able to correctly identify that all loop-closure constraints are correct and converges to the correct solution for all the datasets. We observe from Table II that our algorithm incurs a very slight increase in runtime as compared to the non-robust pose-graph optimizations.

VI. Conclusion

We showed in this paper that the robust pose-graph loop-closure problem can be modeled with the Bayesian network and solved with the EM algorithm. We further proved that our robust pose-graph loop-closure algorithm with EM is conceptually similar to the M-Estimator and our choice of the Cauchy function has a better capability in suppressing the effects from outliers than the commonly used Huber cost function. We showed that our algorithm performed better than the existing algorithms - (a) Switchable Constraints and (b) Max-Mixture Model by showing results from both simulated and large-scale real-world datasets.

VII. Acknowledgement

This work is supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #269916 (v-charge) and 4DVideo ERC Starting Grant Nr. 210806.

References

[1] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *International Conference on Robotics and Automation (ICRA)*, 2006, pp. 2262–2269.

[2] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *International Conference on Robotics and Automation (ICRA)*, May 2011.

[3] D. Galvez-Lopez and J. D. Tardos, "Real-time loop detection with bags of binary words," in *Intelligent Robots and Systems (IROS)*, Sept 2011, pp. 51–58.

[4] M. Cummins and P. Newman, "Appearance-only slam at large scale with fab-map 2.0," *International Journal of Robotics Research (IJRR)*, vol. 30, no. 9, pp. 1100–1123, Aug 2011.

[5] N. Sünderhauf and P. Protzel, "Towards a robust back-end for pose graph slam," in *International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1254–1261.

[6] ——, "Switchable constraints for robust pose graph slam," in *Intelligent Robots and Systems (IROS)*, 2012, pp. 1879–1884.

[7] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time," in *Robotics Science and Systems (RSS)*, July 2012.

[8] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," in *Robotics Science and Systems (RSS)*, July 2012.

[9] A. Pratik, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *International Conference on Robotics and Automation (ICRA)*, 2013.

[10] G. Celeux and G. Govaert, "A classification EM algorithm for clustering and two stochastic versions," *Computational Statistics and Data Analysis*, vol. 14, no. 3, pp. 315–332, Oct 1992.

[11] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[12] C. Bibby and I. Reid, "Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association," in *Robotics Science and Systems (RSS)*, 2007.

[13] R. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*. Kluwer Academic Publishers, 1998, pp. 355–368.

[14] S. Agarwal and K. Mierle, "Google ceres solver." [Online]. Available: http://code.google.com/p/ceres-solver/

[15] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of slam algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, Nov. 2009.

[16] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 2161–2168.