Robust Graph SLAM Back-ends: A Comparative Analysis

Yasir Latif, César Cadena and José Neira

Abstract— **In this work, we provide an in-depth analysis of several recent robust Simultaneous Localization And Mapping (SLAM) back-end techniques that aim to recover the correct graph estimate in the presence of outliers in loop closure constraints. We present a benchmark dataset for evaluation of such methods by augmenting the KITTI Vision Benchmark with ground truth as well as generated loop closure hypotheses and present a detailed analysis of recently proposed robust SLAM methods using this benchmark. We also look into how these methods achieve the desired robustness and what are the implications for the SLAM problem. We discuss the issues involved in using the output of these robust back-ends for tasks such as path planning and how they can be addressed. The problem of robustness needs to be addressed adequately in order to have a complete and reliable solution to the SLAM problem.**

I. Introduction

The graph based formulation of the SLAM problem poses it as a non-linear least squares optimization problem and iteratively finds the Maximum A Posteriori (MAP) estimate that explains all the observations. While this formulation itself was proposed a while back [9], efficient solutions have emerged only recently. In a pose graph, robot poses are modeled as nodes in the graph and edges represent constraints between the poses. Sequential constraints are introduced by an odometry system that estimates the incremental change in the robot pose, while a place recognition system introduces non-sequential constraints. Given these constraints, an optimization back-end finds the most likely position of the nodes.Majority of pose graph optimizers assume Gaussian noise in the constraints and several works have been proposed to cater for non-Gaussian noise, e.g. using robust cost functions (Huber function) [6], robust optimization methods [13], or explicitly handling non-Gaussian distributions [14]. While these approaches are excellent for handling unmodeled errors in the constraints, such as wheel slippage, in the presence of false positives in the place recognition system these approaches cannot prevent the estimate from getting corrupted, especially when there are multiple persistent spurious constraints.

There is a fundamental difference between edges generated by odometry and those generated by place recognition: odometry constraints, by definition, are topologically correct (even though they might be metrically inaccurate) while the same is not true for loop closure constraints, since perceptual aliasing may cause the place recognition system to incorrectly associate two topologically unrelated places, severely corrupting the map estimate as a result. Recently, many proposals have been made to deal with the problem of inconsistent constraints. A more traditional way is to robustify the front-end place recognition system, in an effort to reduce the number of inconsistent constraints that corrupt the back-end. In this regard, Olson [10] proposed a hypothesis verification method for loop closure constraints using graph partitioning based on spectral clustering, though this method fails to distinguish between topologically inconsistent loop and loops that have large drift. At the moment front-end algorithms alone cannot guarantee 100% accuracy. On the other hand, the back-end has more information about the problem and can contribute to a more informed decision about the validity of loop closures. In recent literature, several methods have been proposed to "robustify" the back-end against possible false positive loop closures [16], [11], [1], [8]. A comparison of these techniques on various real and synthetic datasets can be found in [17]. This work is more focused on real world scenarios in which there are a lot of false positives and very few (or no) true positives.

This research has been partially funded by the Dirección General de Investigación of Spain under project DPI2012-36070 and by DGA-FSE (group T04), and supported by Australian Research Council grant DP130104413

Yasir Latif and José Neira are with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain. {ylatif, jneira}@unizar.es. César Cadena is with the Department of Computer Science, University of Adelaide, Australia. cesar.cadena@adelaide.edu.au

II. Pose Graph Formulation

In the graph based formulation for SLAM, the so-called "Graph-SLAM", robot poses are modeled as nodes in the graph nodes and constraints as edges between the nodes. Under the Gaussian assumption, the sensor noise is modeled using the covariance (or equivalently, the information) matrix. Let $\mathbf{x} = (x_1 \dots x_n)^T$ be a vector of parameters that describe the configuration of the nodes and ω_{ij} and Ω_{ij} be the mean and the information matrix of the observation of node j from node i. Given the state $\mathbf{x}$, let $f_{ij}(\mathbf{x})$ be a function that calculates the perfect observation according to the current state. The residual r_{ij} can then be calculated as:

$$r_{ij}(\mathbf{x}) = \omega_{ij} - f_{ij}(\mathbf{x}) \tag{1}$$

Constraints can either be introduced by odometry which are sequential constraints $(j = i+1)$, or from a place recognition system, which are non-sequential. The amount of error introduced by each constraint, weighed by its information, can be calculated as:

$$d_{ij}(\mathbf{x})^2 = r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) \tag{2}$$

and therefore the overall error, assuming all the constraints to be independent, is given by:

$$D^2(\mathbf{x}) = \sum_{(i,j)\in\mathcal{G}} d_{ij}(\mathbf{x})^2 = \sum_{(i,j)\in\mathcal{G}} r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) \tag{3}$$

where $d_{ij}(\mathbf{x})^2$ is residual on the edge connecting nodes i and j in the graph $\mathcal{G}$. The solution to graph-SLAM problem is to find a state $\mathbf{x}^*$ that minimizes the overall error.

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{G}} r_{ij}(\mathbf{x})^T \Omega_{ij} r_{ij}(\mathbf{x}) \tag{4}$$

The above can more compactly be written as

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{G}} \|r_{ij}(\mathbf{x})\|^2_{\Sigma_{ij}} \tag{5}$$

where $\Sigma_{ij} = \Omega_{ij}^{-1}$ is the corresponding covariance matrix for the given information matrix.

III. Overview of Robust Back-end Methods

In this section we present a short overview of recent methods proposed for solving the robust SLAM problem.

A. Switchable Constraints

Switchable Constraints (SC) [16], [18] poses the robust back-end problem as a regularization problem. For every loop closure that is introduced in the graph, an additional variable, termed a "switch", is attached to the loop closure. This switch controls the contributed residual error of the corresponding loop closures towards the non-linear optimization problem. In this formulation, the optimizer has to find the optimal configuration of poses ($\mathbf{x}^*$) as well as the switch variables ($\mathbf{s}^*$). This can be written as:

$$\mathbf{x}^*, \mathbf{s}^* = \underset{\mathbf{x},\mathbf{s}}{\operatorname{argmin}} \sum_{j=i+1} \|r_{ij}(\mathbf{x})\|^2_{\Sigma_{ij}} + \sum_{j\neq i+1} \|s_{ij} r_{ij}(\mathbf{x})\|^2_{\Sigma_{ij}} + \sum_{j\neq i+1} \|\gamma_{ij} - s_{ij}\|^2_{\Gamma_{ij}} \tag{6}$$

where γ_{ij} is the switch prior (set to 1) and Γ_{ij} is the corresponding covariance for the switch prior, sequential constraints are represented by $j = i+1$ and loop closures as $j \neq i+1$. The switch variable s_{ij} is allowed to vary between 0 and 1. Initially, γ_{ij} is set equal to s_{ij} representing the belief that all loop closures are correct. The optimizer can move the switch values (s_{ij}) to increase the error in the third term, if it decreases the error in the second. This implements the regularizer, which penalizes loop closures with large errors. The effect of switch variables can be interpreted in two ways, as a robust function that scales the residual by the switch variable (s_{ij}) or alternately, scales the information matrix with the square of the switch variable (s_{ij}^2). For a small switch value (≈ 0) the residual becomes zero and therefore it does not contribute towards the estimation problem. At the same time, from an information matrix point of view, the information matrix becomes too uninformative (close to zero) and the corresponding constraint is not considered in the estimation problem.

For this method, the optimizer has to find an optimal state that depends on the poses in the graph as well as the switch variable. SC therefore solves a problem that is larger than the original problem in the number of unknowns. The difference in size depends on the number of loop closures. The only tunable parameter is Γ_{ij}, representing the covariance of the switch priors.

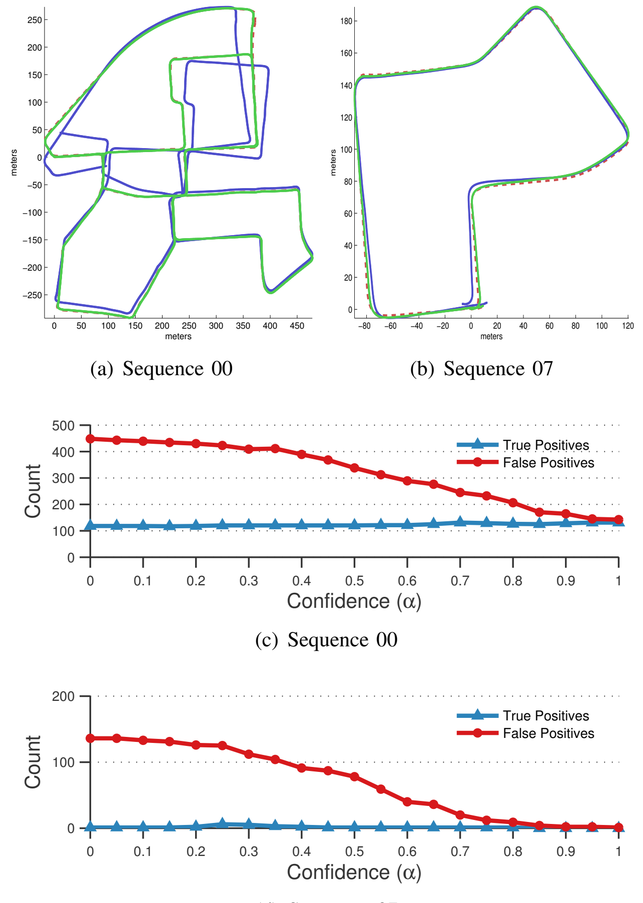


Fig. 1. Example of sequences in the dataset: **(a-b):** Ground truth (dashed), visual odometry (blue), optimized trajectory with all correct loop closure (green). **(c-d):** Number of true positives and false positives as the confidence parameter (α) is varied from zero to one.

B. Dynamic Covariance Scaling

Dynamic Covariance Scaling (DCS) [1] proposes a closed form solution to the regularized robust back-end problem by extending SC. Rather than introducing a new switch variable and letting the solver find the optimal values for it, DCS calculates the switch values using:

$$s_{ij} = min\left(1, \frac{2\Phi_{ij}}{\Phi_{ij} + \chi^2_{ij}}\right) \tag{7}$$

where $\Phi_{ij} = \Gamma_{ij}^{-1}$ and χ^2_{ij} is the current normalized residual as given in (2).

DCS implements a robust cost function (m-estimator) that can be applied to any term in the graph SLAM formulation. Furthermore, unlike SC, the problem size does not change. The only parameter that needs to be defined is Φ_{ij} which is inverse of the parameter Γ_{ij} used in SC.

C. Max Mixtures

Max Mixtures (MM) [11] takes a similar approach to the problem of robust back-ends as SC and DCS, but rather than dynamically scaling the information matrix like SC and DCS, MM attaches to each loop closing constraint a predefined large covariance matrix. This represents a uniform distribution over space representing the null hypotheses; an incorrect loop closure. The distribution (original or uniform) which best explains the loop closure is then selected and used in the optimization process. The simplicity of the algorithm comes from the observation that rather than using mixtures of Gaussians, which do not fit well into the framework of non-linear least square optimization, the problem can be represented by a max-mixture of Gaussian distribution, in which the most likely Gaussian distribution is selected from the mixture. In case of loop closure verification, a second Gaussian distribution with the same mean and a very large variance is associated to loop closure. This second distribution is specified by two parameters; a weight (w) and a scale (s) which turns the original distribution $N(\mu,\Sigma)$ into a weighted and scaled version $wN(\mu,s\Sigma)$.

D. Realizing, Reversing, Recovering

Realizing, Reversing, Recovering (RRR) [8] is a consensus based algorithm which checks for loops that lead to successful convergence of the graph-SLAM problem. RRR divides loop closures into clusters based on topological similarity and then tries to find the largest subset of clusters than are consistent among themselves as well as with the underlying odometry. Consistency is considered in the chi-squared (χ^2) sense. The algorithm first carries out consistency checks for each cluster in order to weed out incorrect links within it, followed by an intra-cluster consistency check. RRR is different from the previous algorithms as it explicitly requires convergence of the graph in order to verify the validity of loop closures. In contrast with SC and DCS, in RRR and MM loop closure decisions are not modeled as continuous variables but as discrete yes/no decisions that need to be made.

IV. Comparative Analysis of Robust SLAM Back-ends

In this section, we first present an enhanced benchmark dataset and use it to investigate the performance of the aforementioned methods. We present comparative results for trajectory error, precision and recall. We also look at how a particular formulation affects the SLAM back-end and what are the implications of using the decisions made by these methods for higher level tasks such as path planning.

A. Benchmark Dataset

The data used in this work comes from the KITTI vision benchmark suite [4] which has been acquired using various sensors mounted on top of a moving vehicle. Visual odometry is calculated using grayscale stereo image pairs working at 10Hz using the open-source library libviso2 [5]. The benchmark provides 21 sequences for visual odometry evaluations but ground truth is provided for only the first 10, among which only 6 close at least one loop. We have selected these 6 along with one sequence without any loops to calculation of stereo visual odometry. Potential loop closures are identified from the right image of each stereo pairs using DLoopDetector library [3] at 2Hz. The library uses a minimum confidence parameter (α) as a threshold for accepting place recognition decision. This parameter is varied from 0.00 (accept on little evidence) to 1.00 (accept only when there is great evidence) with increments of 0.05, generating 21 sets of loop closures hypotheses for every visual odometry sequence. This allows us to evaluate the effect of varying number of outliers in loop closure hypotheses. Ground truth loop closures were manually annotated using visual inspection of images in conjunction with the provided ground truth trajectory. Transformations between the loop closure nodes have also been calculated using libviso2. All the 7×21 datasets, generated as a results of these processing steps, have been made public on the authors' website[1].

The KITTI Vision Benchmark also provides per-pose ground truth for all the sequences used in this work. Relative Pose Error (RPE) is used to obtain an estimate of the noise statistics. RPE is defined as $t_{ij,GT}\ominus t_{ij,Odom}$ for each corresponding transformation t_{ij} in the Ground Truth (GT) and odometry (odom). The covariance matrix obtained from RPE for all transformations was used as the noise estimate for each transformation in the pose graph, including the loop closures. Two of seven datasets, along with number of true and false positive loop closures, are shown in Fig. 1. Due to space limitations, results are presented here for the two sequences shown in Fig. 1. Complete results for all the seven sequences can be found on the authors' website[2].

B. ATE, Precision and Recall

In the first set of experiments, we compare the aforementioned methods on the two sequences, namely sequence 00 and sequence 07 from the KITTI vision benchmark. Since the datasets come from a moving vehicle in an urban environment, loop closure algorithms suffer greatly from perceptual aliasing as many of the buildings and roads look similar. In the first sequence the vehicle revisits its path at several different locations and is able to close many loops correctly. In the second sequence, the vehicle moves along a circuit that just closes the loop at the end of the trajectory. The number of false positives in this case greatly exceeds the number of true positives.

Each of the robust back-end methods was executed with default parameters and then with a set of varying parameters to assess the effect of these parameters on the performance of the algorithm. The performance metric used for evaluation are Precision, Recall, and Absolute Trajectory Error (ATE) [12], which is the mean of squared differences in location of the estimated and ground truth poses, after they have been aligned to a common frame of reference.

In order to visualize the accuracy of the final trajectory, each ATE plot contains the ATE of odometry, which is the error of the visual odometry against the ground truth, and the ATE of trajectory optimized with all correct loop closures, which is the lowest achievable error. If an algorithm rejects all the loop closures, the ATE ends up being the same as that

[1]`http://webdiis.unizar.es/~ylatif`

[2]`http://webdiis.unizar.es/~ylatif/IROS.html`

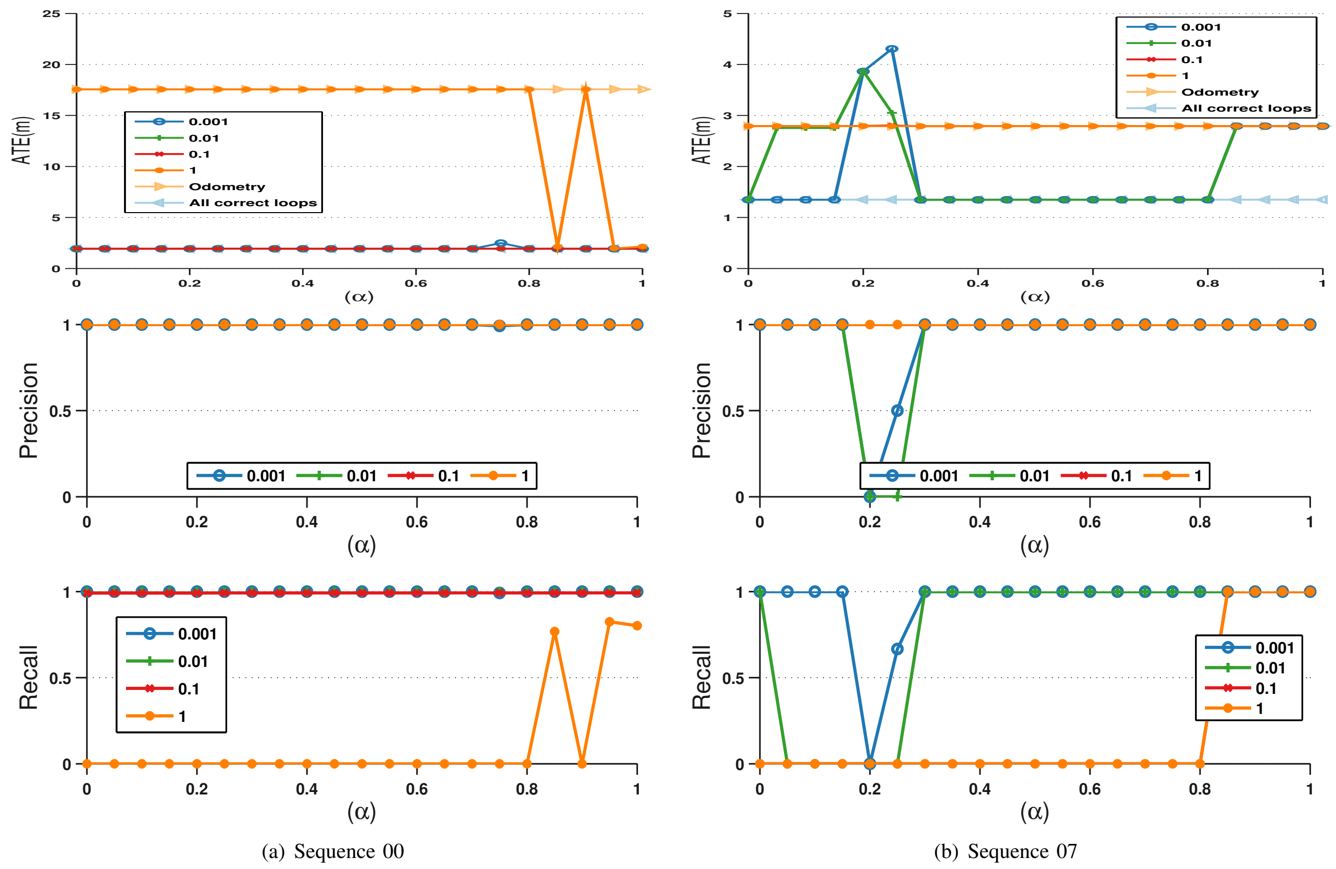


Fig. 2. Results for SC on Sequence 00 and 07: Each line represents the value of the switch covariance used for the experiment. **(Top)**: ATE **(Middle)**: Precision **(Bottom)**: Recall

of the odometry. If it accepts false positives, which deforms the final estimate, ATE is greater than the odometry. Correct loop closures lower the ATE and if all loop closures are accepted, the lower limit of ATE will be achieved. Next we present the performance evaluation of each method.

1) Switchable Constraints (SC): SC has a single tuning parameter (s) representing the covariance of the switch prior. The authors suggest that in most cases the default value of $s = 1$ should be used. We conduct four experiments, decreasing the value of the parameter by an order of magnitude for each successive experiment. The results are given in Fig. 2. For calculating precision and recall, loop closures are considered accepted if the switch value is greater that 0.5 and rejected otherwise.

For Sequence 00, the default parameter (1.0) rejects all the proposed loop closures resulting in zero recall in most of the experiments. For the next values of switch covariance (0.1,0.01,0.001) the algorithm works with full precision and recall, leading to a trajectory with the lowest possible trajectory error.

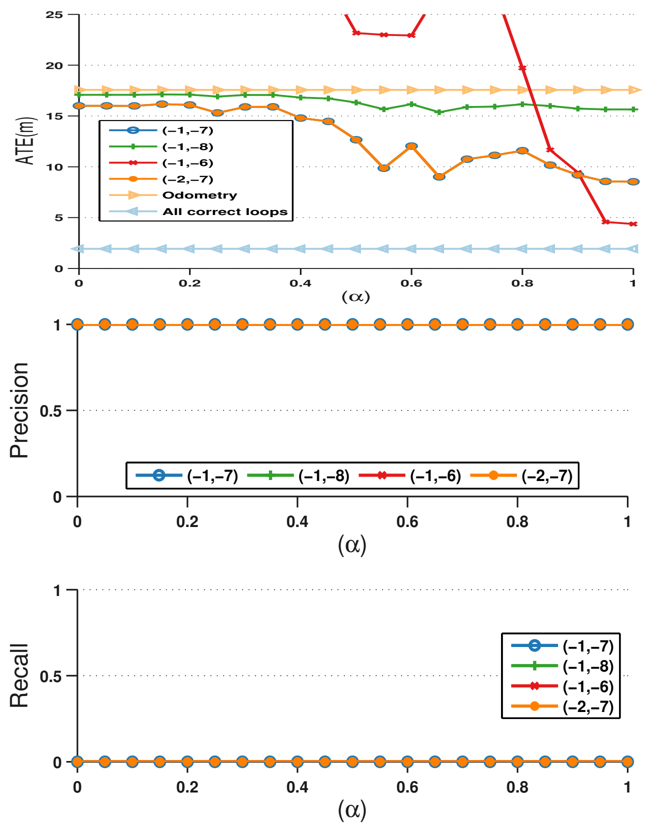
For Sequence 07, where there are a very few loop closures, the default parameter still rejects all the loop closures. In should be noted that for alpha $>$ 0.8, there are no true positive loop closures suggested by the front-end place recognition system. Similar behavior is observed for $s = 0.1$. For values of switch covariance (0.01, 0.001), some false positives loop closures are accepted, leading to a higher trajectory error. This can be seen in Fig. 2(b) where the recall as well as precision falls to zero.

There are two things that need to be noted. Firstly, to a very large extent parameters need to be tuned in order to make the algorithm work for different trajectories. Secondly, for the same value of the parameter, the output is dependent on the distribution of loop closures i.e. the same parameters would not work for a given trajectory in the presence of different incorrect loop closures.

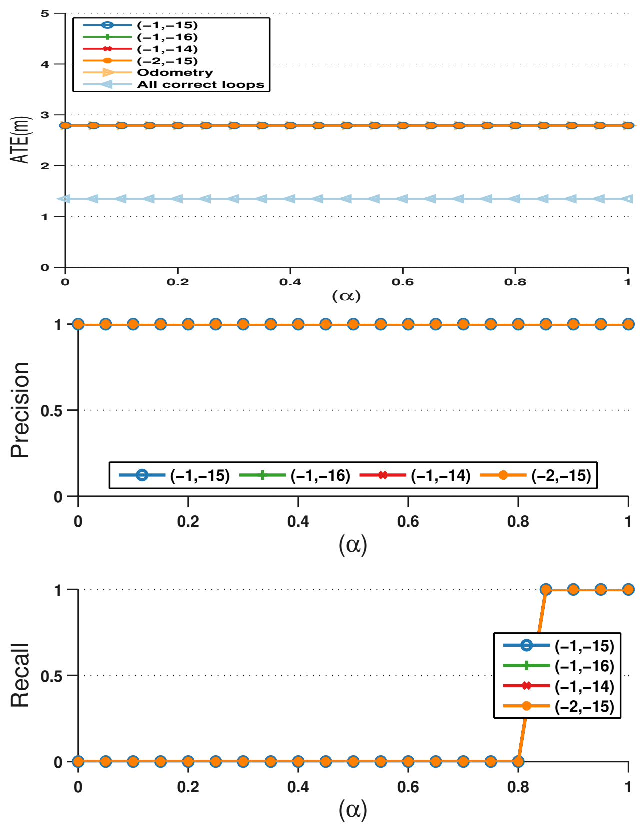
2) Max-Mixtures (MM): Max-mixtures has two tunable parameters, the weight (w) and scale (s), that are used to create the weighted and scaled version of the original distribution. Unlike DCS, MM and RRR, this requires a parameter search on a 2D space. For this experiment, parameters were searched in the range $10^{-5} < s < 10^{-1}$ and $10^{-15} < w < 10^{-1}$ and the parameters providing the least ATE for each sequence were selected. These are the base values for the experiment. In order to evaluate the sensitivity of MM to tuning parameters, three further experiment with neighboring parameters $(s, w/10)$, $(s, 10w)$ and $(s/10, w)$ we also carried out. The results are given in Fig. 3.

For Sequence 00, the base parameters are w = 10^{-1} and s = 10^{-7}. We could not find any parameters that would allow MM to differentiate between correct and incorrect loop closures. The results presented here are the smallest scale (10^{-15}) and the largest weight (10^{-1}).

Our experiments show that MM is more sensitive to



(a) Sequence 00



(b) Sequence 07

Fig. 3. Result for MM on Sequence 00 and Sequence 07: Each line represents the log_{10} of weight (w) and scale(s) parameters used for the experiment. **(Top)**: ATE **(Middle)**: Precision **(Bottom)**: Recall

change in scaling parameter but less sensitive to change in weighting parameter, as can be seen in Fig. 3(a). The two experiments with different weights but same scale result in same ATE while moving the scale parameter leads to greater ATE.

One thing to note is that while the ATE is different from the trajectory error of just odometry, the recall is still zero. For MM, a loop closure is counted as being accepted when it becomes more likely compared to the null hypothesis. In the experiments shown, initially some of the correct loop closures are more likely but as the optimization goes on, all null hypothesis are accepted instead. Initially the trajectory error decreases but stops short of the minimum achievable ATE when the null hypotheses become more likely and result in the rejection of all loop closures.

3) Dynamic Covariance Scaling (DCS): DCS is formulated as a robust kernel and therefore the only tunable parameter is the kernel width (w). We conduct experiments with DCS using parameters equivalent to SC. The default value for w is 1. We use three other values w = (10,100,1000) for each sequence to carry out additional experiments. The results are given in Fig. 4.

For Sequence 00, the default parameters lead to everything being rejected. For the next value of the parameter (10), DCS performs better than SC. This can be attributed to the better convergence properties of DCS. For higher values (100,1000) DCS correctly identifies all the correct loop closures, exhibiting full precision and recall.

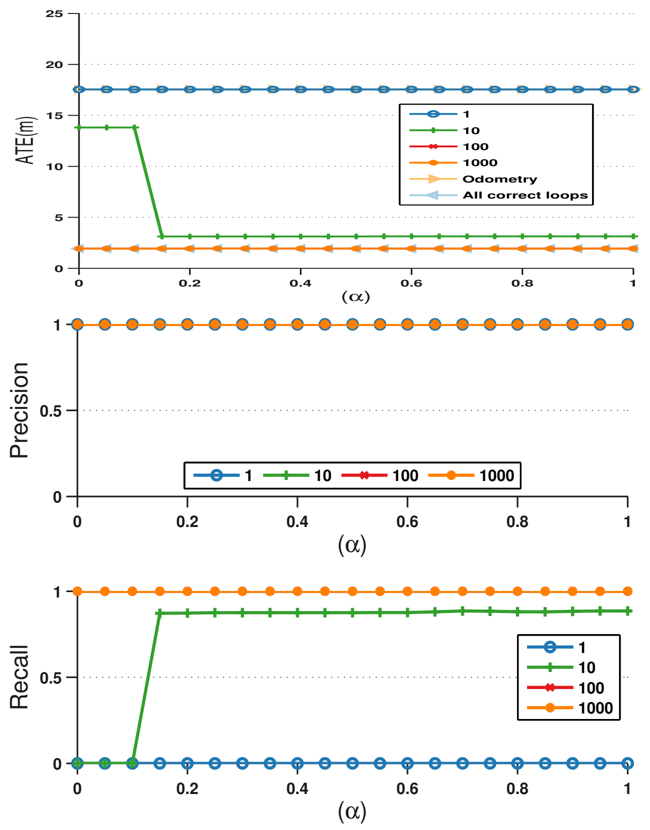
For Sequence 07, the behaviour of DCS is very similar to SC. Default parameter again leads to rejection of everything. Loop closures are accepted only when w=1000 (the highest parameter) and in that case DCS accepts false positives.

While DCS exhibits better convergence properties than SC, it suffers from the same problem of parameter tuning. In Fig. 4(b) the value that allows us to make some correct decision, is orders of magnitudes greater than the default parameter suggested.

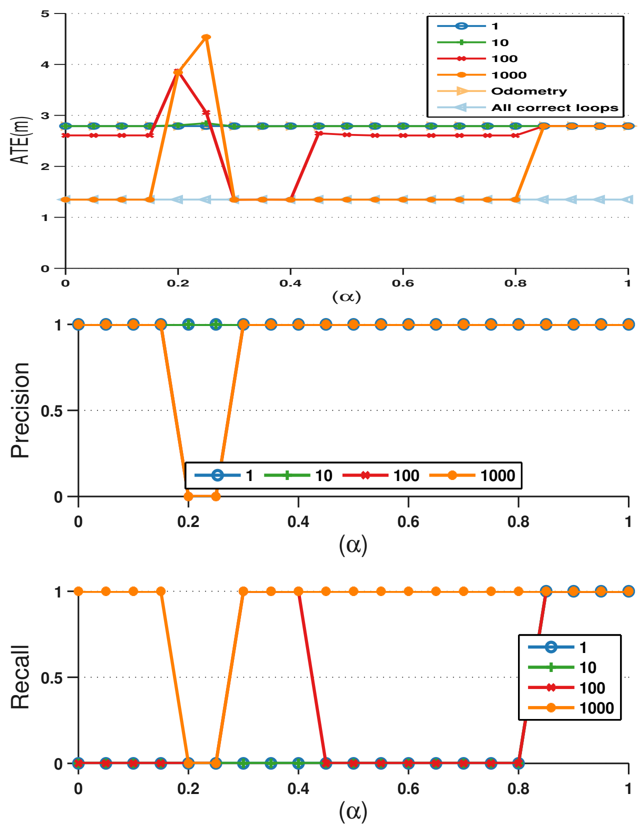
4) RRR: RRR has a single parameter, that controls how clusters are formed, the clustering threshold (t_g). We suggest using a threshold of 10 seconds. In order to investigate sensitivity to tuning parameters, we use t_g = (1,5,20) in addition to the default value of 10. The results are presented in Fig. 5.

For Sequence 00, RRR works with full precision and considerable recall, resulting in a map estimate that is better than all the previous methods. For different values of t_g, the method works with full precision except when the value is the highest (20). In that case some false positives are accepted because clusters arising from different locations become a single cluster.

For Sequence 07, RRR successfully detects the very few correct loop closures. In comparison, all other algorithms either rejected all loop closures or accepted all correct ones, RRR selects a portion of the loop closures about which it



(a) Sequence 00



(b) Sequence 07

Fig. 4. Result for DCS on Sequence 00 and Sequence 07: Each line represents the kernel width (w) used for the experiment.(**Top**): ATE (**Middle**): Precision (**Bottom**): Recall

is most certain. This results in a lower recall at times but compared to previous methods, it detects the correct loop closures over a varying range values of the tuning parameter.

C. Problem Representation and Sparsity

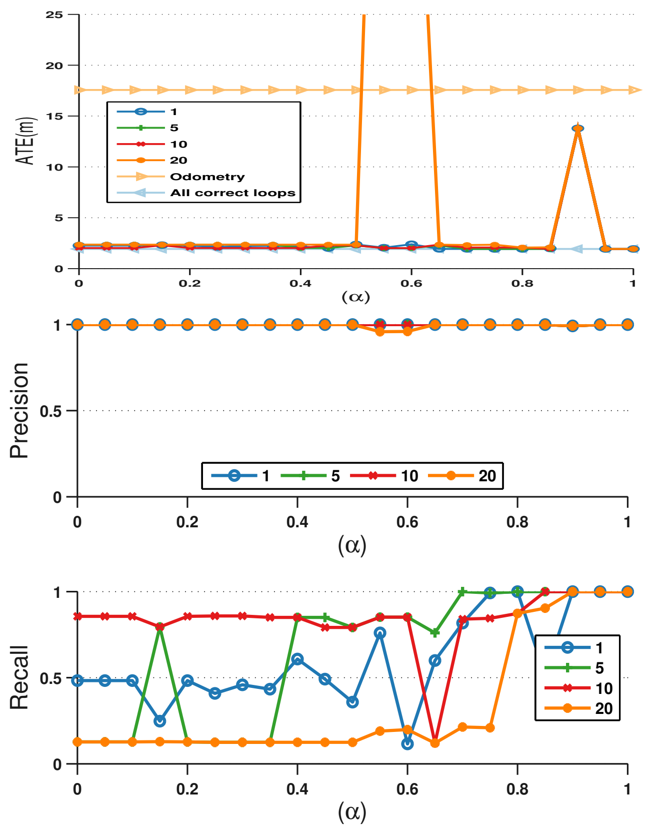
As mentioned before, the robust methods considered can be divided into two categories: 1) methods that make binary decision: MM and RRR and 2) methods that provide a continuous value $\in (0,1)$: SC and DCS. While in principle the validity of a loop closure is a binary decision, the implication of making a non-binary decision are deeper, especially considering an incremental pose-graph SLAM problem since efficient solvers take advantage of the sparsity of the problem. MM while making a binary decision, represents the rejected loop closures with small information matrices, in effect making the corresponding blocks in the information matrix non-zero. In an incremental setting, such as Bayes Trees in iSAM2, this leads to solving for a greater number of variables than are actually needed to be solved. SC has the additional overhead of solving for all the switch variables as well, as they are a part of state being optimized. In order to illustrate this, we use the city10000 dataset available with g2o. It consists of $10,000$ poses, 10688 loop closures to which an additional 900 false loop closures are added. The difference between how the algorithms maintain information matrices becomes highly noticeable for such a large dataset. The information matrix along with the upper triangular decomposition using COLAMD for the output of each algorithm is shown in Fig. 6. In general, the greater the fill-in, the less sparse the problem and hence the more difficult to solve.

As discussed earlier, SC causes the most fill-in as the switch variable are a part of the state being estimated. DCS relieves this problem by converting the switch variables into a robust-kernel, but near zero non-zero blocks representing rejected loop closures are still maintained in the information matrix. The same is true for MM as information regarding rejected hypotheses is still maintained. RRR makes a binary decision and removes the information belonging to incorrect loop closure from the pose graph, resulting in the least fill-in.

D. Navigation

The aim of solving a SLAM problem is to create a map that can then be used to carry out some higher level tasks such as navigation and planning. In this context, loop closures provide information about the graph topology and traversibility. Methods that do not make a binary decision about the validity of loop closures do not get completely rid of topologically inconsistent paths in the graph. When used for path planning, some algorithms may select these non-existent paths and try to navigate using them, which may lead to failures in the navigation task.

In order to illustrate this, we use the open source FaMuS algorithm [2] that plans minimum uncertainty paths given a pose graph. We use the output of Sequence 00 with alpha = 1.00 for DCS with w = 1000 and try to plan a


(a) Sequence 00

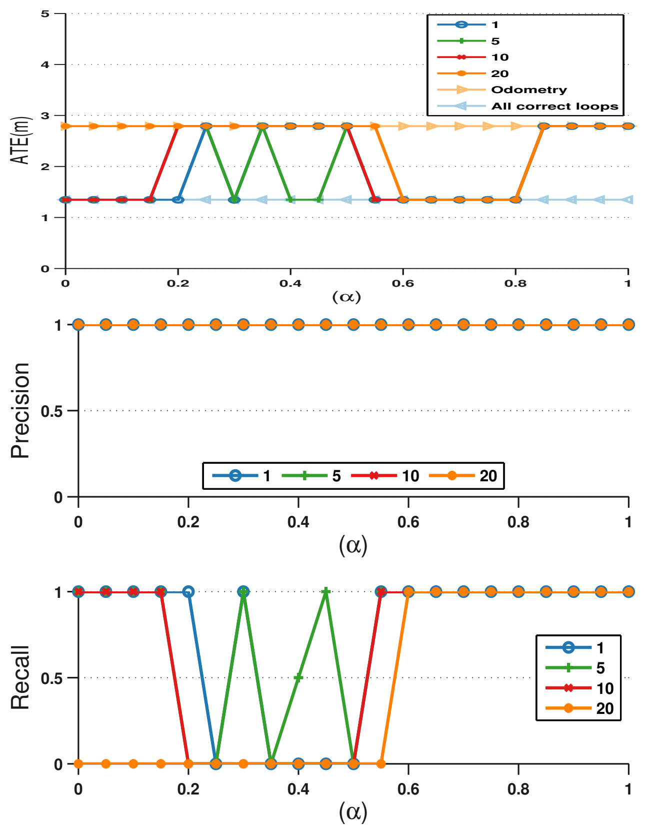
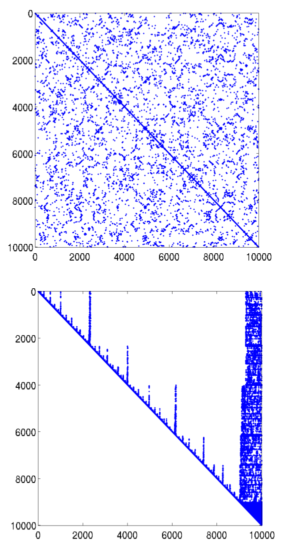
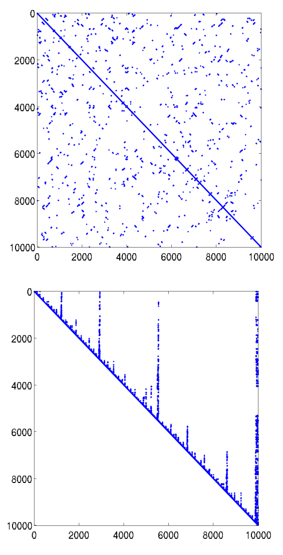

(b) Sequence 07

Fig. 5. Result for RRR on Sequence 00 and Sequence 07: Each line represents clustering threshold in seconds used for the experiment.**(First Row)**: ATE **(Second Row)**: Precision **(Bottom Row)**: Recall


(a) DCS/MM


(b) RRR

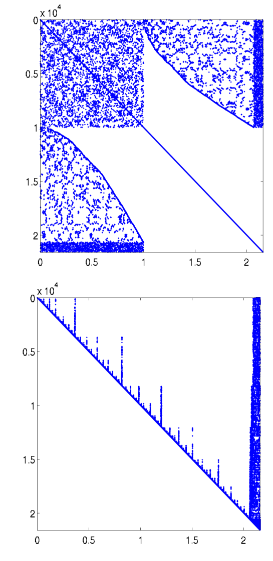
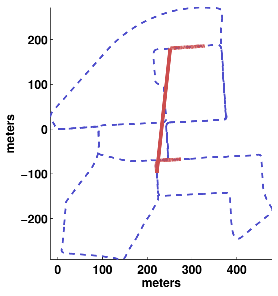

(c) SC

Fig. 6. City10,000 dataset **Top**: Information matrix maintained by each algorithm. **Bottom**: The corresponding fill in using COLAMD for upper triangular matrix **R**

path between two random nodes. The map output by DCS correctly identifies and optimizes with all the correct loop closures. For this experiment, the square of scaling parameter in (7) is used to scale the covariance matrix for each loop closure. The result of the planned path from one part of the trajectory to another is shown in Fig. 7.

The planning algorithm assumes that all the links provided are navigable. It chooses a path which has been rejected by DCS (it has very small information) and tries to plan a path using this non-existent link in the graph. A planning algorithm will face the same problem when it tries to use the output of SC.

One way to address this problem is to threshold the loop closures based on the scaling parameter before using the


(a) DCS

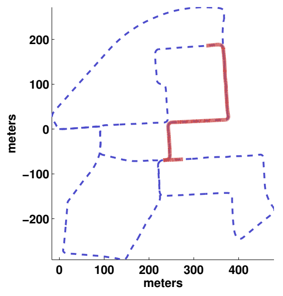

(b) DCS - Thresholded / RRR

Fig. 7. Path Planning on the Output of DCS: Sequence 00, alpha=1.00, w=1000. **(a)**: Using output of DCS directly leads to inclusion of rejected links in the path **(b)**: Tresholding and only using accepted links gives correct path.

graph for path planning. Although this will resolve the above issue by removing links from the graph which have been rejected, such a decision might cause two problems: firstly, false positive loops accepted at the moment of decision will become a permanent part of the graph; secondly, in an incremental setting, decisions may need to be reversed in light of new evidence. Fixing decisions by thresholding takes this ability away from the algorithm to reconsider past decisions in light of new evidence.

V. Conclusions

In this work we have reviewed the current state of the art robust SLAM back-end methods. We have introduced an enhancement to an existing dataset, which has been extended with loop closure verification in mind, based on real data. We compare the performance of SC, MM, DCS and RRR under varying conditions for different trajectories and assess their sensitivity to tunable parameters. We have also shown that it is not a good idea for such methods to make non-binary decisions about the validity of loop closures as it may cause problems for applications trying to use the pose graph to execute a higher level task.

SC and DCS formulate the problem as a robust m-estimator which has a breaking-point of zero, that is, even a single outlier can cause unbounded error in the estimate [15]. MM makes the assumption that the we already know what the contaminating distribution is, which is a very strong (and often impractical) assumption because outliers in loop closings do no necessarily follow a known distribution. RRR, on the other hand, is robust but a false positive accepted can lead to a great error in the estimated map, as it has the complete effect on optimization as no robust functions are used. In terms of execution time, all methods take a negligible amount of time compared to the time of the full experiment. They can be arranged in increasing order (from fastest to slowest) of execution speed as: DCS, MM, SC and RRR.

This work also highlights the need for parameter tuning that some of the algorithms need and how much their performance is affected by them. DCS, MM and SC belong to the same family of algorithms that try to either scale the covariance matrix or provide a fixed "guess" for its value. RRR on the other hand carries out a series of statistical tests which works even when we have to find the metaphorical needle in the haystack.

The ideal solution to the robust SLAM problem is to design an algorithm that can reason along time, and be able to reverse decisions in light of new evidence. If methods such as DCS and SC accept some false positives, this would lead them to reject any conflicting evidence that might arrive in future because the optimizer has already reached a local minima even though it is an incorrect one due to the accepted false positive. On the other hand, consensus based methods, such as iRRR [7], have been shown to have the ability to reconsider past decisions in an incremental setting. While these methods do not lessen the need for finding better front-end place recognition algorithms, they can certainly complement place recognition methods in achieving better map estimates.

References

[1] P. Agarwal, G. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust map optimization using dynamic covariance scaling. In *Proc. IEEE Int. Conf. Robotics and Automation*, Karlsruhe, Germany, 2013.

[2] H. Carrillo, Y. Latif, J. Neira, and J.A. Castellanos. Fast minimum uncertainty search on a graph map representation. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2012.

[3] D. Galvez-Lopez and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013.

[5] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*, 2011.

[6] P.J. Huber. Robust regression: asymptotics, conjectures and monte carlo. *The Annals of Statistics*, 1(5):799–821, 1973.

[7] Y. Latif, C. Cadena, and J. Neira. Realizing, Reversing, Recovering: Incremental Robust Loop Closing over time using the iRRR algorithm. In *Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, October 2012.

[8] Y. Latif, C. Cadena, and J. Neira. Robust loop closing over time for pose graph SLAM. *The International Journal of Robotics Research*, 32(14):1611–1626, 2013.

[9] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[10] E. Olson. Recognizing places using spectrally clustered local matches. *Robotics and Autonomous Systems*, 57(12):1157–1172, December 2009.

[11] E. Olson and P. Agarwal. Inference on networks of mixtures for robust robot mapping. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[12] RAWSEEDS. Robotics advancement through Webpublishing of sensorial and elaborated extensive data sets (project FP6-IST-045144), 2009. http://www.rawseeds.org/rs/datasets.

[13] D.M. Rosen, M. Kaess, and J.J. Leonard. An incremental trust-region method for robust online sparse least-squares estimation. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, St. Paul, MN, May 2012.

[14] D.M. Rosen, M. Kaess, and J.J. Leonard. Robust incremental online inference over sparse factor graphs: Beyond the Gaussian case. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Karlsruhe, Germany, May 2013.

[15] P. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):pp. 871–880, 1984.

[16] N. Sünderhauf and P. Protzel. Switchable Constraints for Robust Pose Graph SLAM. In *Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, 2012.

[17] N. Sünderhauf and P. Protzel. Switchable Constraints vs. Max-Mixture models vs. RRR–a comparison of three approaches to robust pose graph SLAM. In *Proc. IEEE Int. Conf. Robotics and Automation*, Karlsruhe, Germany, 2012.

[18] N. Sünderhauf and P. Protzel. Towards a robust back-end for pose graph slam. In *Proc. IEEE Int. Conf. Robotics and Automation*, 2012.