Robust Map Optimization using Dynamic Covariance Scaling

Pratik Agarwal, Gian Diego Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard

***Abstract*—Developing the perfect SLAM front-end that produces graphs which are free of outliers is generally impossible due to perceptual aliasing. Therefore, optimization back-ends need to be able to deal with outliers resulting from an imperfect front-end. In this paper, we introduce dynamic covariance scaling, a novel approach for effective optimization of constraint networks under the presence of outliers. The key idea is to use a robust function that generalizes classical gating and dynamically rejects outliers without compromising convergence speed. We implemented and thoroughly evaluated our method on publicly available datasets. Compared to recently published state-of-the-art methods, we obtain a substantial speed up without increasing the number of variables in the optimization process. Our method can be easily integrated in almost any SLAM back-end.**

I. Introduction

Building maps with mobile robots is a key prerequisite for several robotics applications. As a result, a large variety of SLAM approaches have been presented in the robotics community over the last decades [1], [2], [3], [4]. One intuitive way of formulating the SLAM problem is to use a graph. The nodes in this graph, represent the poses of the robot at different points in time and the edges model constraints between these poses. The edges are obtained from observations of the environment or from motion carried out by the robot. Once such a graph is constructed, the map can be computed by optimization techniques. The solution is the configuration of the nodes that is best explained by the measurements.

Most approaches assume that the constraints are affected by noise but no outliers (false positives) are present, i.e., there are no constraints that identify actually different places as being the same one. This corresponds to the assumption of having a perfect SLAM front-end. In traditional methods, a single error in the data association often leads to inconsistent maps. Generating outlier-free graphs in the front-end, however, is very challenging, especially in environments showing self-similar structures [5], [6], [7]. Thus, having the capability to identify and to reject wrong data associations is essential for robustly building large scale maps without user intervention. Recent work on graph-based SLAM addressed the issue and there are now methods that can handle a large number of outliers [8], [9], [10].

The contribution of this paper is a novel approach, namely Dynamic Covariance scaling (DCS) which deals with outliers, while at the same time avoiding an increase in execution time (see Fig 1). Our work stems from the analysis of a recently introduced robust back-end based on switchable constraints (SC) [8] and uses a robust function that generalizes classical gating by dynamically scaling the covariance. Compared to state-of-the-art approaches in robust SLAM back-ends, our strategy has a reduced computational overhead and typically has better convergence. The proposed function shares common grounds with existing robust M-estimators. Furthermore, our method can be integrated easily into existing graph-based SLAM systems and does not require significant changes in the code. In the experimental section, we thoroughly evaluate our approach and show that our formulation outperforms alternative methods in terms of convergence speed.

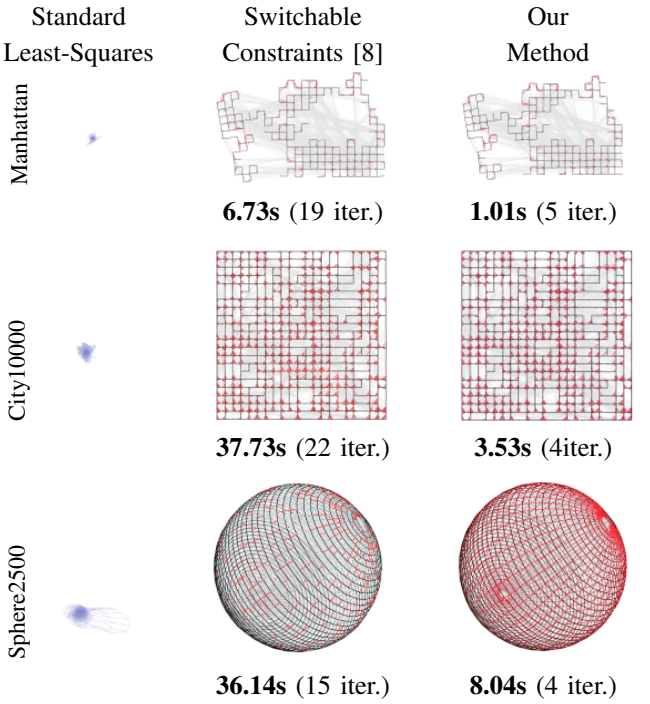


Fig. 1. Graph optimization on standard datasets with 1,000 outliers. Standard least square (left) fails while the switchable constraints method (center) as well as our approach (right) provides comparable results. Our method shows substantially faster convergence of up to a factor of 10. Colored lines depict accepted loop closures.

II. Related Work

Various SLAM approaches have been presented in the past. Lu and Milios [11] were the first to refine a map by globally optimizing the system of equations to reduce the error

All authors are with the University of Freiburg, Institue of Computer Science, 79110 Freiburg, Germany. This work has been partially supported by BMBF, contract number 13EZ1129B-iView and EC under contract numbers ERC-267686-LifeNav and FP7-600890-ROVINA

introduced by constraints. Subsequently, Gutmann and Konolige [12] proposed a system for constructing the graphs and for detecting loop closures incrementally. Since then, many approaches for minimizing the error in the constraint network have been proposed, including relaxation methods [13], [3], stochastic gradient descent and its variants [2], [14], smoothing techniques [4] and hierarchical techniques [15], [1].

The techniques presented above allow for Gaussian errors in the constraints of the pose-graphs, i.e., noisy constraints, but they cannot handle outliers, i.e., wrong loop closing constraints between physically different locations. Although SLAM front-ends (loop generation and loop validation) improved over the last years [7], [6], [5], it is not realistic to assume that the generated pose-graphs are free of outliers.

Hence researchers recently started using the back-end slam optimizer to identify outliers. For example, Sünderhauf and Protzel [8] proposed a technique that is able to switch off potential outlier constraints. The function controlling this switching behavior is computed within the SLAM back-end. Olson and Agarwal [9] recently presented a method that can deal with multi-modal constraints, by introducing a max operator. Their approach approximates the sum of Gaussian model by the currently most promising Gaussian. This allows for dealing with multi-modal constraints and rejecting outliers while maintaining computational efficiency. Latif *et al.* [10] proposed RRR, which handles outliers by finding the maximum set of clustered edges, consistent with each other. The key difference of RRR to the previously described two approaches [8], [9] is that RRR rejects false edges while the other two always keep rejected edges with a low weight.

Our approach is similar to [8] since we also keep rejected constraints with a small probability, but it is more principled and leads to faster convergence.

III. Optimization with Switchable Constraints

The graph-based approach to solve the SLAM problem is to minimize the error function:

$$X^* = \operatorname*{argmin}_X \sum_i \| f(x_i, x_{i+1}) - z_{i,i+1} \|^2_{\Sigma_i} + \sum_{ij} \underbrace{\| (f(x_i, x_j) - z_{ij}) \|^2_{\Lambda_{ij}}}_{\chi^2_{l_{ij}}} \quad (1)$$

where x_i represents the pose of the robot at time i, z_{ij} is the transformation between two poses obtained by odometry or sensor measurements and $f(x_i, x_j)$ is the function that predicts the measurement between x_i and x_j. The covariances of the odometry and sensor measurements are Σ and Λ. Thus, whereas the first term in Eq. 1 represents the incremental constraints that result from odometry constraints, the second term refers to loop closing constraints.

Recently, Sünderhauf and Protzel [8] published an effective method for optimizing graphs in the presence of outliers. Their basic idea is to introduce switching variables $s_{ij} \in [0, 1]$ that can disable potential outlier constraints. Since our method builds upon their insights, we provide a short review of this technique. They minimize:

$$X^*, S^* = \operatorname*{argmin}_{X,S} \sum_i \| f(x_i, x_{i+1}) - z_{i,i+1} \|^2_{\Sigma_i} + \sum_{ij} \| \Psi(s_{ij})(f(x_i, x_j) - z_{ij}) \|^2_{\Lambda_{ij}} + \sum_{ij} \| 1 - s_{ij} \|^2_{\Xi_{ij}} \quad (2)$$

which can be interpreted as three sums over different χ^2 errors, i.e., the one of the incremental constraints, the loop closing constraints and the switch priors. Here, $\Psi(s_{ij}) \in [0, 1]$ is a scaling function that determines the weight of a constraint given s_{ij} and a switching prior Ξ_{ij}. Sünderhauf and Protzel propose a joint optimization of X and S. As a result of this, for each loop closing constraint, an additional variable s_{ij} has to be considered by the optimizer. The addition of the switch variable increases computation cost of each iteration and in this case increases the problem complexity and thus, potentially decreases the convergence speed.

A. Analysis of the Switchable Constraints Error Function

The function $\Psi(\cdot)$ can be interpreted as a scaling factor in the information matrix associated to a constraint, since

$$\begin{aligned} &\| \Psi(s_{ij})(f(x_i, x_j) - z_{ij}) \|^2_{\Lambda_{ij}} \\ &= \Psi(s_{ij})(f(x_i, x_j) - z_{ij})^\top \Lambda_{ij}^{-1} \Psi(s_{ij})(f(x_i, x_j) - z_{ij}) \quad (3) \\ &= (f(x_i, x_j) - z_{ij})^\top \Lambda_{ij}^{-1} \Psi(s_{ij})^2 (f(x_i, x_j) - z_{ij}) \quad (4) \\ &= \| f(x_i, x_j) - z_{ij} \|^2_{\Psi(s_{ij})^{-2}\Lambda_{ij}} \quad (5) \end{aligned}$$

Sünderhauf and Protzel [8] suggest to set $\Psi(s_{ij}) = s_{ij}$ within the interval $[0, 1]$ to obtain the best results. To simplify the following derivations, we directly replace $\Psi(s_{ij})$ by s_{ij}.

IV. Dynamically scaled covariance kernel

The disadvantage of SC is the need of additional variables s_{ij}, one for each constraint subjected to be an outlier. In the remainder of this paper, we will show how to circumvent the use of the switching variables inside the optimizer. This leads to a significantly faster convergence, meanwhile obtaining comparable robustness. We next introduce the main contribution of this work, namely a technique that provides an analytical solution to the scaling factors. This does not only simplify the optimization problem by greatly reducing the number of variables, but it also creates an easier to optimize cost surface as shown empirically in Section V.

In the following, we analyze the behavior of the error function defined in Eq. 2. In particular, we investigate how the switching variables influence the χ^2 at the local minima. Without the loss of generality, let us consider the edge between two nodes k and l. We can split the error function into two blocks, the first one considers all edges except kl and the

second block only the edge kl:

$$\begin{aligned} X^*, S^* = \underset{X,S}{\operatorname{argmin}} & \sum_i \| f(x_i, x_{i+1}) - z_{i,i+1} \|^2_{\Sigma_i} \\ & + \sum_{ij \neq kl} \| s_{ij}(f(x_i, x_j) - z_{ij}) \|^2_{\Lambda_{ij}} \\ & + \sum_{ij \neq kl} \| 1 - s_{ij} \|^2_{\Xi_{ij}} \\ & + \| s_{kl}(f(x_k, x_l) - z_{kl}) \|^2_{\Lambda_{kl}} + \| 1 - s_{kl} \|^2_{\Xi_{kl}} \\ = \underset{X,S}{\operatorname{argmin}} & \underbrace{g(X_{ij \neq kl}, S_{ij \neq kl}) + \underbrace{s^2_{kl}\chi^2_{l_{kl}} + (1 - s_{kl})^2\Phi}_{h(s,\chi^2)}}_{b} \end{aligned} \tag{6}$$

with $\Phi := \Xi_{ij}^{-1}$. In Eq. 6, the term $g(\cdot)$ represents the error for all edges, including odometry, except of the edge kl.

Once the optimizer converges, the partial derivatives with respect to *all* variables $\in \{X, S\}$ are zero. Hence the derivative with respect to s_{kl} must be 0. Taking the partial derivative of Eq. 6 with respect to s_{kl}, we obtain for a generic constraint (indices are omitted for notational simplicity):

$$\nabla b = \begin{bmatrix} \vdots \\ \frac{\partial b}{\partial s} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ 2s\chi_l^2 - 2(1-s)\Phi \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ 0 \\ \vdots \end{bmatrix} \tag{7}$$

Solving for s, in terms of Φ and χ_l^2, leads to

$$\begin{aligned} 2s\chi_l^2 - 2(1-s)\Phi &= 0 \\ s\left(\chi_l^2 + \Phi\right) &= \Phi \\ s &= \frac{\Phi}{\chi_l^2 + \Phi}. \end{aligned} \tag{8}$$

Substituting s from Eq. 8 in $h(\cdot)$, we obtain:

$$\hat{h} = \frac{\Phi^2\chi_l^2}{\left(\chi_l^2 + \Phi\right)^2} + \Phi - \frac{2\Phi^2}{\chi_l^2 + \Phi} + \frac{\Phi^3}{\left(\chi_l^2 + \Phi\right)^2} \tag{9}$$

The function $\hat{h}(\cdot)$ represents the projection of $h(\cdot)$ on the manifold where the gradient is 0. Finding the maxima of this function is equivalent to obtain an upper bound on χ_l^2 for all possible solutions computed by the optimizer. Lets analyze the derivative of Eq. 9:

$$\frac{d\hat{h}}{d\chi_l} = \frac{2\chi_l\Phi^2}{\left(\Phi + \chi_l^2\right)^2} \tag{10}$$

As can be seen from Eq. 10, the derivative is 0 when $\chi_l = 0$. Thus, we evaluate the function $\hat{h}$ at $\pm\infty$ and 0:

$$\lim_{\chi_l \to \pm\infty} \hat{h} = 0 + \Phi - 0 + 0 = \Phi \tag{11}$$

$$\chi_l = 0 \Rightarrow \hat{h} = 0 + \Phi - 2\Phi + \Phi = 0 \tag{12}$$

Thus, $h(\cdot) \leq \Phi$ for every solution computed by the optimizer. This can be generalized to all switch variables and hence $h(\cdot) \leq \Phi$ for every constraint. By using Φ as an upper bound for all robust edges, we obtain:

$$\begin{aligned} (1-s)^2\Phi + s^2\chi_l^2 &\leq \Phi \\ \Phi + s^2\Phi - 2s\Phi + s^2\chi_l^2 &\leq \Phi \\ s^2(\Phi + \chi_l^2) + s(-2\Phi) + (\Phi - \Phi) &\leq 0 \\ s(s(\Phi + \chi_l^2) - 2\Phi) \leq 0. \end{aligned} \tag{13}$$

The solution to this inequality is given as

$$0 \leq s \leq \frac{2\Phi}{\Phi + \chi_l^2}. \tag{14}$$

In theory, one could choose any value for s within that interval. We choose the value that minimize $h(\cdot)$ within that interval, while not exceeding 1. It is given by

$$s = \min\left(1, \frac{2\Phi}{\Phi + \chi_l^2}\right). \tag{15}$$

In sum, we have a closed form solution for computing the scaling factor s individually for each loop closing constraint. It depends on χ_l^2, which is the original error term for each loop closing constraint. This formulation dynamically scales the information matrix of each non-incremental edge by s^2 given by Eq. 15 and thus by a factor that considers the magnitude of the current error. A gradient always exists in the direction of an edge and gradually increases in the presence of more mutually consistent constraints. The cost surface is always quadratic but the magnitude of the gradient is scaled according to s, which depends on the current error (χ_l^2) and Φ.

It should be noted that this result can be integrated in basically any graph-based SLAM system with minimal efforts. Once s_{ij} is computed based on the error of the corresponding constraint, the residuals can be scaled with s_{ij} or the information matrix be scaled with s_{ij}^2 depending on implementation of the SLAM back-end.

A. Relations to Robust Estimation Theory

The analytical solution of s^2 derived above also shows its relation to iterative re-weighted least squares (IRLS) and robust M-Estimation. Similar to IRLS, we scale the covariance to modify the residuals iteratively. In fact the Geman-McClure [16] M-Estimator has a similar weight function:

$$w(x) = \frac{1}{\left(1 + x^2\right)^2}, \tag{16}$$

Under the special condition of $\Phi = 1$ and not forcing $s \leq 1$, both scaling functions are similar. DCS allows changing the values of Φ and prevents over-fitting by having $w(x) \leq 1$. Computing exact values of Φ as well as comparison with Geman-McClure is subject of future work.

V. Experimental Evaluation

We implemented the method described above and conducted a large set of evaluations comparing it to switching constraints (SC) [8]. We have used the g2o framework with Gauss-Newton optimization steps for all our experiments [17].

TABLE I
DATASETS USED IN OUR EXPERIMENTS.

Dataset	# Poses & Landmarks	# Correct Loop Constraints	# Outliers (max.)
ManhattanOlson	3,500	2,099	5,000
ManhattanG2O	3,500	2,099	5,000
City10000	10,000	10,688	5,000
Intel Research	943	894	5,000
Sphere2500	2,500	2,450	5,000
CityTrees10000	10,100	4,443	1,000
Victoria Park	7,120	3,640	1,000
Bicocca	43,116	767	516
Lincoln Labs	6,357	2,334	3,754

A. Datasets and Outliers

To support comparisons, we used publicly available datasets, namely the Manhattan3500, Intel Research Lab, City10000, Sphere2500, CityTrees10000, and Victoria Park datasets. For Manhattan3500, we considered the two different initialization procedures provided by Olson [2] and g2o [17]. The Intel Research Lab dataset is available in the g2o package [17] and the City10000, CityTrees10000, Sphere2500 datasets as well as the Victoria Park dataset were released with the iSAM package [4]. We also evaluated additional large-scale datasets such as the 36 loops of the Lincoln Lab and the five loops of the Bicocca multi-session experiment initially evaluated with RRR [10]. Tab. I lists the properties of the different datasets as well as the maximum number of outliers present in the corrupted versions. For landmark datasets, the loop constraints are pose-landmark edges.

The corrupted versions of the data sets contain both, real and simulated outliers. For simulated outliers, we used four different approaches to generate them namely "random", "local", "random grouped", and "local grouped" as described in [8]. Random outliers connect any two randomly sampled nodes in the graph. Local outliers connect random nodes that are in the vicinity of each other. For the grouped outliers, we create clusters of 10 mutually consistent outliers. We believe that randomly grouped outliers are the most realistic form of outliers as such constraints are similar to systematic errors generated due to perceptual aliasing by a front-end. The outliers are generated using the script provided in the Vertigo package [18]. For landmark datasets such as Victoria Park and CityTrees10000, we added wrong loop closures between random pairs of nodes and landmarks.

For the Bicocca and Lincoln multi-session datasets, we used the processed datasets provided by Latif et al. [10] in which loop closures are generated using a place recognition system subjected to perceptual aliasing. The Bicocca dataset uses a bag of word-based front-end while the Lincoln Lab dataset was created with a GIST-based front-end.

For all evaluations, unless otherwise stated, $\Phi = 1$, since this is suggested value according to [8].

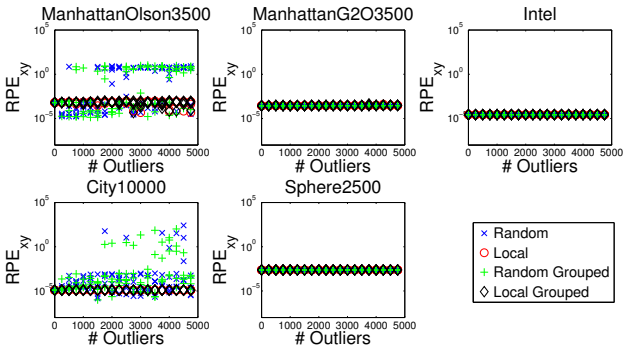


Fig. 2. Scatter plots showing the error depending on the number and type of outliers for DCS. ManhattanG2O, Intel, and Sphere2500 converge to the correct solution even with 5,000 outliers while City10000 and ManhattanOlson always convergence in the case of local outliers. City10000 converges to the correct solution for up to 1,500 outliers which are not local. ManhattanOlson is more sensitive to non-local outliers.

B. Robust against Simulated and Real Outliers

To show the robustness against outliers we evaluated DCS on both simulated and real outliers. First, we evaluated DCS on datasets with up to 5,000 simulated outliers. In total, we evaluated 400 graphs per dataset—100 for each of the four outlier generation strategy. Scatter plots of the resulting reprojection error (RPE) after convergence are shown in Fig. 2. As can be seen, for the ManhattanG2O, Intel and Sphere2500 datasets, DCS always converges to the correct solution. For ManhattanOlson and City10000, DCS converges in all the local outlier cases but is sensitive to the non-local outliers. City10000 fails to converge to the correct solution in some non-local cases with more than 1500 outliers. Even when ManhattanOlson does not converge, the RPE is always less than 10 and appears somewhat constant. This case is further analyzed in Section V-G.

Compared to the other datasets evaluated in this paper, the next two datasets contain outliers created from a front-end due to place recognition errors. The goal of these experiments is to evaluate the performance of DCS with an error prone front-end. We use the data-sets evaluated in [10] and thus also provide an informal comparison to it. Fig. 3 depicts the optimization results of DCS on the Biccoca and Lincoln Lab datasets.

DCS takes 0.79 s (3 iterations) to optimize the Lincoln Lab dataset and 1.56 s (16 iterations) to optimize the Bicocca dataset. For the Bicocca dataset, we achieved the best result with $\Phi = 5$. By visual inspection, we can see that our solution is close to the reported correct structure in [10]. Compared to RRR, which reports a timing of 314 s for the Bicocca dataset, DCS takes only 1.56 s and thus is around two orders of magnitude faster. SC does not find the correct solution in the standard settings and requires an additional Huber robust kernel which takes 5.24 s to find the solution [10].

C. Timing analysis

The next set of experiments are conducted to analyze the timing performance of DCS and to support the claim

TABLE II
OPTIMIZATION TIME NEEDED BY SC AND DCS IN THE PRESENCE OF 1000 TO 5000 OUTLIERS WITH RANDOM(R), LOCAL(L), RANDOM-GROUPED(RG) AND LOCAL-GROUPED(LG) OUTLIER GENERATION STRATEGIES.

Dataset	1000 R, L, RG, LG	2000 R, L, RG, LG	3000 R, L, RG, LG	4000 R, L, RG, LG	5000 R, L, RG, LG
ManG2O					
SC	4.70, 1.91, 3.11, 1.55	8.17, 2.93, 4.46, 2.85	10.11, 3.45, 11.89, 5.11	11.21, 2.80, 11.17, 3.32	24.53, 3.14, 15.33, 4.67
DCS	**2.09, 0.86, 1.41, 0.88**	**3.83, 1.07, 2.80, 1.00**	**5.47, 1.25, 4.24, 1.17**	**7.62, 1.44, 6.27, 1.38**	**9.29, 1.69, 8.42, 1.59**
ManOlson					
SC	14.53, 2.21, 10.65, 2.21	18.96, 2.80, 15.45, 2.71	39.34, 3.41, 39.94, 3.27	53.29, 4.69, 36.71, 4.54	67.44, 5.33, 61.16, 5.09
DCS	**4.62, 1.08, 3.40, 1.07**	**6.57, 1.35, 3.23, 1.27**	**26.21, 1.57, 20.21, 1.46**	**29.24, 1.84, 26.46, 1.71**	**16.80, 2.03, 14.00, 1.93**
Intel					
SC	0.54, 0.42, 0.51, 0.39	1.20, 0.94, 1.18, 0.94	1.60, 1.22, 1.61, 1.20	2.00, 1.52, 2.01, 1.50	2.37, 1.78, 2.44, 1.74
DCS	**0.34, 0.22, 0.31, 0.21**	**0.52, 0.31, 0.52, 0.34**	**0.69, 0.45, 0.71, 0.42**	**0.85, 0.53, 0.85, 0.50**	**1.00, 0.58, 1.08, 0.58**
City10000					
SC	47.61, 30.06, 41.11, 29.86	108.2, 33.84, 79.50, 33.52	212.8, 41.14, 134.9, 39.04	285.7, 43.82, 207.1, 40.70	389.9, 49.98, 446.5, 49.92
DCS	**10.09, 3.98, 7.88, 3.93**	**36.94, 4.80, 15.74, 4.53**	**51.60, 5.95, 34.02, 5.65**	**218.8, 6.92, 50.09, 6.44**	**262.9, 8.04, 393.2, 7.37**
Sphere2500					
SC	53.83, 11.09, 48.26, 10.62	115.5, 14.88, 108.9, 16.03	240.1, 24.10, 170.3, 18.55	218.7, 30.78, 230.2, 57.22	310.7, 67.53, 281.8, 63.37
DCS	**19.52, 7.83, 16.84, 7.51**	**42.52, 9.22, 38.39, 9.02**	**50.58, 10.40, 50.32, 9.94**	**66.51, 11.31, 69.39, 11.36**	**90.12, 12.35, 97.07, 11.97**

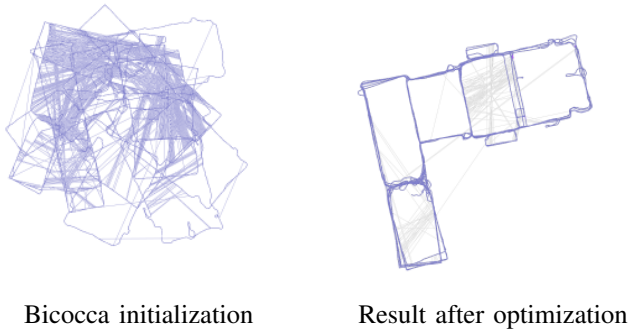


Fig. 3. Qualitative evaluation on 5 sessions of Bicocca (top) and 36 loops of Lincoln Labs (bottom) datasets that contain outliers generated by the vision system. Latif et al. [10] report a that RRR solves Bicocca in 314 s whereas DCS requires only 1.56 s to obtain the solution.

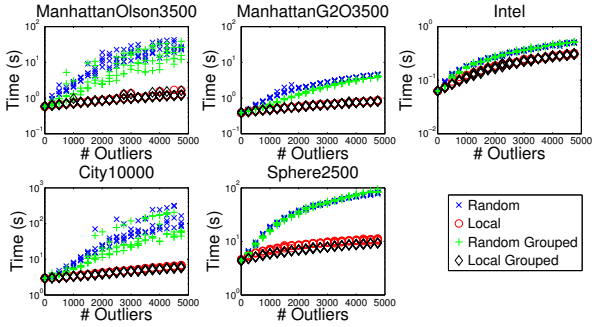


Fig. 4. Scatter plots showing the runtime depending on the number and type of outliers for DCS. DCS does not add significant overheads to the sparse matrix factorization algorithms compared to standard least squares. Local outliers create less fill-in inside and hence requires less time.

that our method converges faster than SC. We first show in Fig. 4 that the optimization time required by DCS depends only on the number of constraints and the outlier generation criteria. Importantly, DCS does not increase the optimization time significantly compared to the standard least squares. The required pptimization time shows a larger variance for random outliers in ManhattanOlson compared to ManhattanG2O as the former starts with a worse initialization.

Tab. II compares the time required by DCS and SC to converge in presence of outliers. This table compares the total time taken for both these algorithms to reach the optimal solution. As can be seen from the table, DCS is faster than SC in all cases. The increase in convergence speed is most noticeable in City10000 dataset. The optimization process for both methods were stopped when the change in χ^2 was less than the set threshold. In the next section we show reduction of χ^2 and RPE is significantly faster and smoother for DCS compared SC.

D. Convergence Properties

The experiments in this section analyze the convergence behavior of DCS and SC in the presence of 1000 randomly grouped errors, as this is the most difficult and realistic scenario. Fig. 5 plots the evolution of the RPE (top row) and the χ^2 error (bottom row) during optimization for SC (blue) and DCS (green). As can be seen from these plots, within at most 6 iterations, DCS converges while SC typically

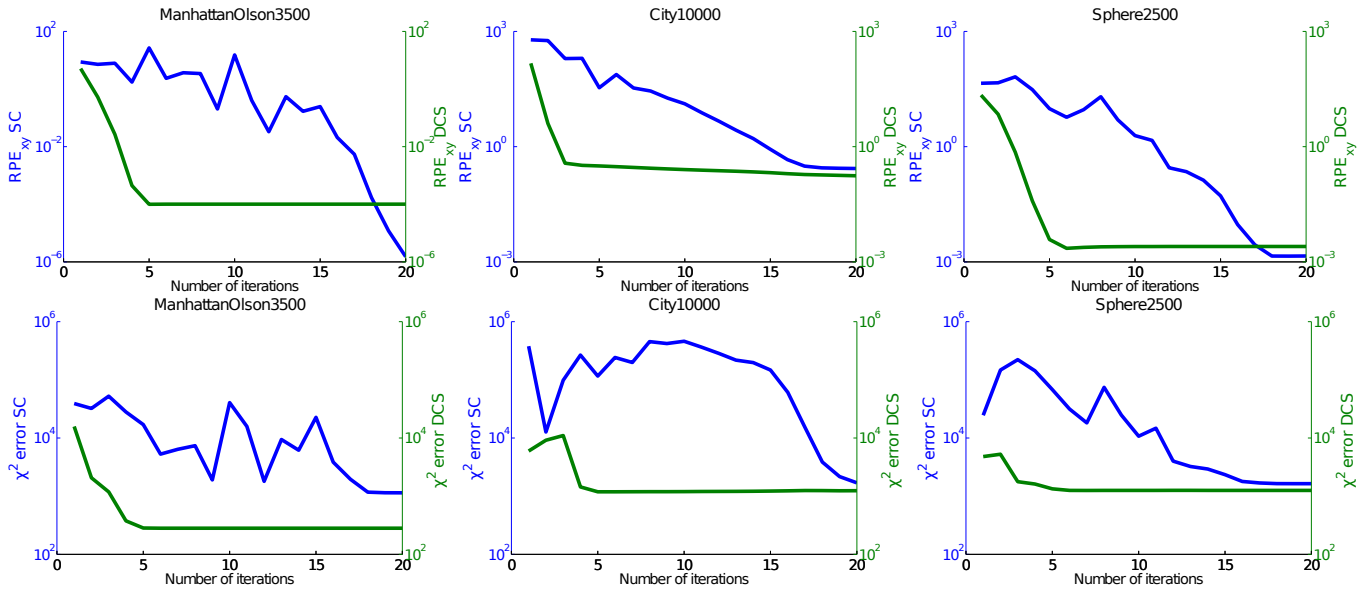


Fig. 5. The figure plots RPE (top row) and χ^2 error (bottom row) for 20 iterations for SC and DCS. While DCS converges within 6 iterations or less, SC needs between 15 and 20 iterations to converge. The shapes of the plots for SC reveal a frequent increase of RPE and χ^2 error which tend to indicate that there are more local minima in the SC formulation compared to DCS.

needs between 15 and 20 iterations to converge. The shapes of the plots for SC reveal a frequent increase of the RPE as well as χ^2 error. We believe this may be indicative of the fact that the Gauss-Newton quadratic approximation of the cost functions for the new optimization problem with additional switch variables in SC is not completely accurate in the neighborhood of evaluation.

For our method, the evolution of χ^2 and RPE is smooth and almost monotonous. The plots illustrate that DCS requires a smaller number of iterations and offers a faster convergence while at the same time being robust to outliers. This is also apparent from the video submitted with the paper, available at http://www.informatik.uni-freiburg.de/%7Eagarwal/videos/icra13/DCS.mp4. Note that the absolute χ^2 values for SC and DCS have to be interpreted differently since SC introduces extra switch prior constraints contributing to the overall error.

E. Parameter Sensitivity

To analyze the sensitivity of DCS and SC with respect to the parameter Φ, we analyzed several values of Φ in the range of 0.01 and 10. Both methods are evaluated on the standard datasets adding 1,000 outliers using random, local random, grouped, and local grouped outliers. Once chosen, the outliers are not changed for different values of Φ. Fig. 6 shows the RPE for varying values of Φ. In general, we found that the sensitivity of DCS and SC is similar for ManhattanG2O, Intel and City10000 datasets. Small values of $\Phi < 0.1$ lead to larger RPE. The RPE however is small, around 10^{-5}, for a wide range of values $0.1 < \Phi < 10$.

For the Sphere2500 dataset, both DCS and SC do not converge for $\Phi < 0.1$. The convergence improves with increasing Φ but DCS fails to converge for $\Phi > 5$ in the presence of local grouped outliers. For the ManhattanOlson dataset, DCS and SC converge for values $0.1 < \Phi \leq 1$ in all cases while both approaches appear to be sensitive to non-local errors. This may be explained by the structure of this dataset since it can be devided into three parts which are connected by very a few constraints only (see left part of ground truth configuration in Fig. 8 and the colored parts). In summary, DCS appears to be more sensitive to the value of Φ in the case of the Sphere2500 dataset but for all other datasets the sensitivity on Φ is comparable for both approaches. Importantly, DCS and SC show a wide range of values for Φ and we thus fixed $\Phi = 1$ as also suggested in [8].

F. Dynamically Scaled Covariances Approach on Landmark-Based Graphs

So far, we evaluated our method only on datasets that contain pose-to-pose constraints, i.e., that directly relate pairs of poses of the robot with each other. Our method also works for landmark-based SLAM. Most landmark-based SLAM systems provide pose-feature range-bearing constraints and pose-to-pose constraints only for odometry. Operating on pose-feature constraints is more challenging for outlier rejection since there is no reliable constraints such as odometry between the feature nodes. In the previous evaluated pose graphs, every node is constrained by two odometry edges which are not subjected to being an outlier. For landmark datasets all constraints to a feature node are potential outliers and hence create large number of local minima solutions.

For landmark datasets we corrupt the outlier free Victoria Park and the CityTrees10000 dataset with up to $1,000$ random outlier constrains. The outliers are random measurements from a robot pose to a landmark. Fig. 7 shows the initialization for these two datasets and the RPE with an increasing number of outliers. As can be seen from the plots, in these two datasets,

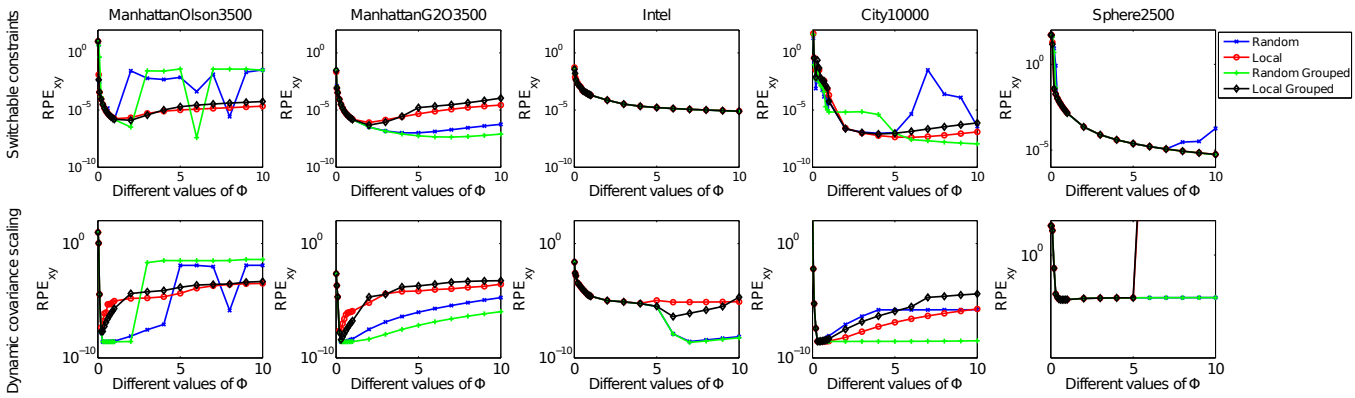
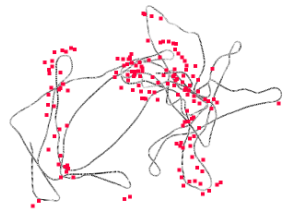
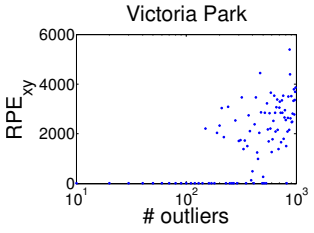


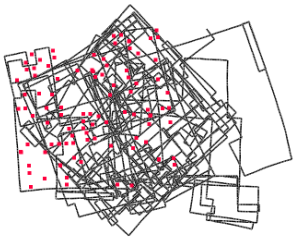
Fig. 6. Robustness of SC (top) and DCS (bottom) with respect to the parameter $\Phi \in [0.01, 10]$ in presence of 1,000 outliers.



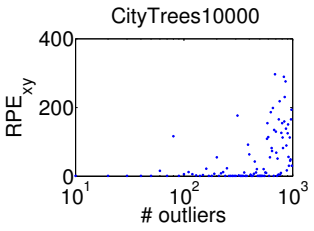
(a) Victoria Park initialization



(b) RPE error vs Num of outliers



(c) CityTrees10000 initialization



(d) RPE error vs Num of outliers

Fig. 7. Resulting RPE for Victoria Park and cityTrees10000 dataset in the presence of a varying number of outliers. Although the initialization is far from the global minimum, DCS is able to converge to the correct solution for small number of outliers.

DCS is robust up to around 100 outliers and the robustness decreases as the outliers are increased thereafter. The fact that DCS is still able to optimize the Victoria Park dataset from the initialization shown for 100 random outliers is strong evidence that the method can be used in synergy with existing front-ends validation techniques for landmark based system to improve robustness.

G. Degenerate case

Investigations of the failure cases in ManhattanOlson found is Section V-B reveal an interesting behavior. We analyses two specific failure cases, one with 501 and one with 4,751 random outliers. After converging, both solutions appear to have similar configuration, even though the second case is subjected to roughly ten times more outliers, shown in Fig. 8. They are locally consistent and appear to have converged to a similar local minima. The scaling values of each false positive edge is shown in the plots in Fig. 8. The problem here is that three parts of the graph are only sparsely connected (see Fig. 8-left). By adding non-local and mutual consistent outliers, there exists configurations in which the system cannot determine all outliers correctly. SC shows a similar issue with ManhattanOlson, which the authors solved by introducing an additional robust Huber kernel at the expense of an even slower convergence [8].

The parking garage dataset is a difficult real world dataset compared to all the previous datasets. This is mainly because of the sparse nature of loop closures. Each deck of the parking garage is connected by two odometry chains. SC had reported degenerate behavior with this dataset [8]. It argued that since only a small number of constraints connect the decks robust methods were not able to outperform non-robust methods.

DCS is able to reject outliers even in this dataset. We also added mutually consistent constraints between decks at multiple levels and compared both methods with standard parameters as shown in fig 9. We believe DCS is able to reject outliers as the gradients of odometry edges and correct loop edges outweigh those provided by the outliers.

VI. Conclusion

In this paper, we introduced dynamic covariance scaling (DCS), an elegant and principled method to cope with outliers in graph-based SLAM systems. We showed that DCS generalizes the switchable constraint method of Sünderhauf and Protzel, while introducing a substantially lower computational overhead. This is achieved by analyzing the behavior of the error function and deriving an analytical solution for computing the weighting factors. We implemented and thoroughly evaluated our approach. We supported our claims with extensive experiments and comparisons to state-of-the-art methods on publicly available datasets. The results show a comparable robustness to outliers as well as in accuracy but with a convergence rate that is substantially faster. The authors have released the source code of the approach presented in this paper with the latest version of g2o.

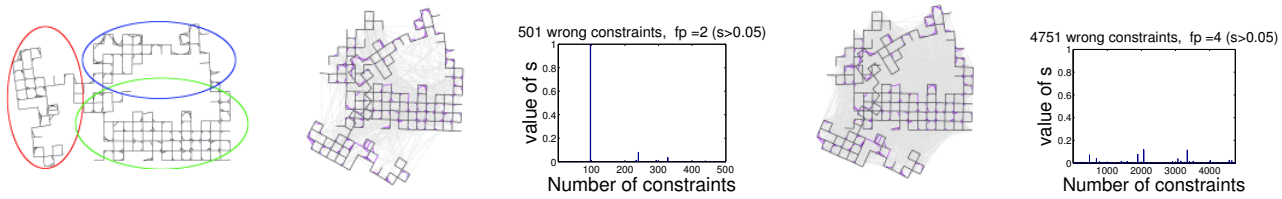


Fig. 8. Left: Ground truth configuration for Manhattan3500. The dataset reveals three sparsely connected region illustrated by the colored ellipses. The other four images are designed to illustrate the two failure cases, obtained for 501 and 4,751 random outliers, in the ManhattanOlson dataset. The images show the local minima maps in both situations together with the scaling values for the false positive constraints. The plots show that even if our method fails to converge to the optimal solution, the number of false positives accepted by the system is small, evident by a small scaling factor. With 501 outliers only two constraints have a scale value of more than 0.05 and with 4,751 outliers only four outliers have a scale value more than 0.05.

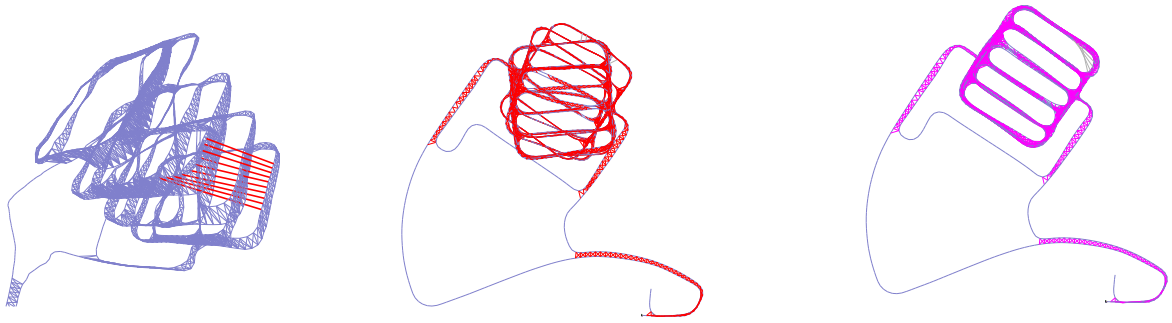


Fig. 9. Parking garage dataset with sparse connection. (Left) The original datasets with wrong loop closures connecting different decks in red. Note: the z-axis is scaled up to clearly show the wrong edges. (Center) SC returns the wrong solution while DCS rejects the outliers(right). This figure shows DCS being able to reject outliers even in the challenging case of datasets with minimal graph connectivity.

VII. Acknowledgement

We thank E. Olson for the manhattanWorld dataset, Y. Latif for providing the processed Bicocca and Lincoln Lab datasets, E. Nebot and A. Ranganathan for original and processed Victoria Park datasets, M. Kaess for City10000, CityTrees10000, and Sphere2500 datasets, N. Sünderhauf for the Vertigo package and his open source implementation. We also thank three anonymous reviewers for providing some insightful comments to help make this manuscript better.

References

[1] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *Proc. of the IEEE Int. Conf. on Rob. & Aut. (ICRA)*, 2010.

[2] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proc. of the IEEE Int. Conf. on Rob. & Aut. (ICRA)*, 2006.

[3] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Trans. on Rob.*, vol. 21, no. 2, 2005.

[4] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association," in *Proc. of the IEEE Int. Conf. on Rob. & Aut. (ICRA)*, 2007.

[5] G. D. Tipaldi, M. Braun, and K. O. Arras, "FLIRT: Interest regions for 2D range data with applications to robot navigation," in *Proc. of the Int. Symp. on Exp. Rob. (ISER)*, 2010.

[6] E. Olson, "Recognizing places using spectrally clustered local matches," *Robotics and Autonomous Systems*, 2009.

[7] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Int. Journal of Robotics Research*, vol. 27, no. 6, 2008.

[8] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph slam," in *Proc. of the IEEE/RSJ Int. Conf. on Intel. Rob. and Sys. (IROS)*, 2012.

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

[10] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time," *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[11] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Aut. Rob.*, vol. 4, 1997.

[12] J.-S. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proc. of the IEEE Int. Symp. on Comput. Intell. in Rob. and Aut. (CIRA)*, 1999.

[13] A. Howard, M. Matarić, and G. Sukhatme, "Relaxation on a mesh: a formalism for generalized localization," in *Proc. of the IEEE/RSJ Int. Conf. on Intel. Rob. and Sys. (IROS)*, 2001.

[14] G. Grisetti, C. Stachniss, and W. Burgard, "Non-linear constraint network optimization for efficient map learning," *IEEE Tran. on Intel. Transp. Sys.*, 2009.

[15] M. Bosse, P. M. Newman, J. J. Leonard, and S. Teller, "An ATLAS framework for scalable mapping," in *Proc. of the IEEE Int. Conf. on Rob. & Aut. (ICRA)*, 2003.

[16] Z. Zhang, "Parameter estimation techniques: A tutorial with application to conic fitting," *Image and vision Computing*, vol. 15, no. 1, pp. 59–76, 1997.

[17] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Rob. & Aut. (ICRA)*, 2011.

[18] N. Sünderhauf, "Vertigo: Versatile extensions for robust inference using graphical models," 2012. [Online]. Available: http://openslam.org/vertigo.html