Published in Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA), 2012. DOI: http://dx.doi.org/10.1109/ICRA.2012.6224709

Towards a Robust Back-End for Pose Graph SLAM

Niko Sünderhauf and Peter Protzel

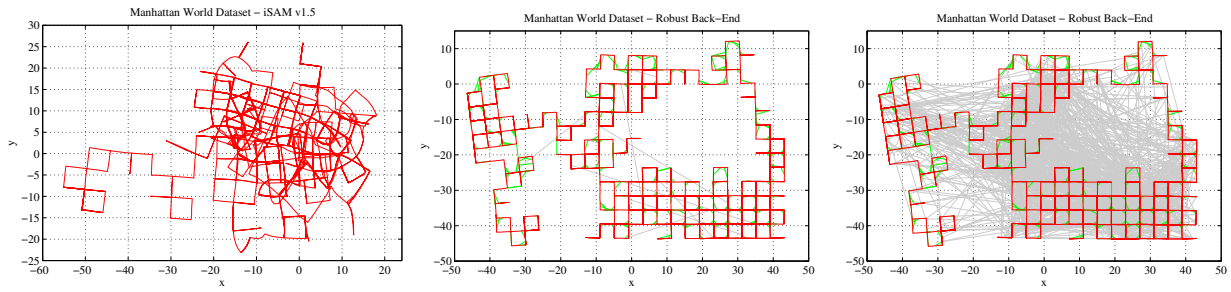


Fig. 1: Exemplary results of the proposed robust SLAM back-end on the synthetic Manhattan world dataset [26] that contains 3500 poses and 2099 loop closures. We corrupted the dataset by introducing 10 additional wrong loop closures that might have been produced due to data association errors (e.g. failed place recognition) in the SLAM front-end. Current back-ends like iSAM 1.5 (left figure) are not able to converge to a correct solution. Our robust solution (middle) correctly discards the wrong loop closure candidates (visible as grey links, the green links are correct loop closure constraints) and converges to a correct solution. Right figure: Despite 1000 wrong loop closures (visible in grey), our proposed method is robust enough to identify all of them and converge towards a correct solution. Notice that the 1000 additional wrong loop closures correspond to 47% of the correct loop closure constraints in the dataset.

***Abstract*— Current state of the art solutions of the SLAM problem are based on efficient sparse optimization techniques and represent the problem as probabilistic constraint graphs. For example in pose graphs the nodes represent poses and the edges between them express spatial information (e.g. obtained from odometry) and information on loop closures. The task of constructing the graph is delegated to a front-end that has access to the available sensor information. The optimizer, the so called back-end of the system, relies heavily on the topological correctness of the graph structure and is not robust against misplaced constraint edges. Especially edges representing false positive loop closures will lead to the divergence of current solvers.**

We propose a novel formulation that allows the back-end to change parts of the topological structure of the graph during the optimization process. The back-end can thereby discard loop closures and converge towards correct solutions even in the presence of false positive loop closures. This largely increases the overall robustness of the SLAM system and closes a gap between the sensor-driven front-end and the back-end optimizers. We demonstrate the approach and present results both on large scale synthetic and real-world datasets.

I. INTRODUCTION

For many years, filter-based methods have dominated the SLAM literature. Although optimization-based approaches that solve the *full*-SLAM problem are known to the community since the work of Lu and Milios in 1997 [19], they have only recently begun to become more popular as efficient algorithms for solving the underlying optimization problems are now available. Prominent examples have been the pose graphs of Olson [26], TreeMap [7], TORO [10], $\sqrt{\text{SAM}}$ [6], iSAM [14] and very recently Sparse Pose Adjustment [16], iSAM2 [13], and g2o [18].

In contrast to filter-based methods (like EKF-SLAM, FastSLAM, etc., see [30] for an introduction to these methods), optimization-based approaches build upon efficient algorithms for nonlinear optimization that exploit the sparsity inherent in the SLAM problem. This way, large-scale SLAM problems containing several 10k variables (poses, landmarks) and constraints (observations, loop closings) can be solved in a matter of seconds on standard hardware.

All of the optimization-based approaches mentioned above express the SLAM problem using a graph structure. For pose-only SLAM problems, i.e. when landmarks are not explicitly modelled and are not part of the SLAM problem, the nodes in these graphs represent the unknown robot poses. The edges express probabilistic constraints between the poses and can contain odometry information or represent loop closures.

A graph representation of this kind is generally referred to as *pose graph* [15] [26] [18]. It is built by the so called *front-end* of the SLAM system that has access to the available sensor information. The *back-end* contains the optimizer that solves the nonlinear least squares optimization problem expressed by the graph.

Since least squares optimization methods are in general not robust against outliers, the back-end has to rely on the front-end to construct a topologically correct graph. If that graph representation is ill-defined, the optimization is likely to fail, diverge and produce defective solutions. This typically occurs when the front-end inserts erroneous loop closure constraint edges due to errors in the underlying data association and place recognition.

Our paper addresses this problem and proposes an extended formulation of the pose graph SLAM problem: Instead of

The authors are with the Department of Electrical Engineering and Information Technology, Chemnitz University of Technology, 09111 Chemnitz, Germany. {niko.suenderhauf, peter.protzel}@etit.tu-chemnitz.de

strictly relying on the front-end, the optimizer will be able to naturally change the topological structure of the problem during the optimization itself. This significantly increases the robustness against outliers of the whole SLAM system and closes the gap between the front-end and the back-end.

In the following, we shortly repeat the general formulation for pose graph SLAM problems before we present our extensions. We describe how the increased robustness was verified both on synthetic benchmarks and on a large-scale real-world dataset consisting of video footage of a 66 km long course through urban streets.

II. POSE GRAPH SLAM

A. Problem Formulation

In the state of the art formulation of pose graph SLAM, we are given a set of odometry constraints $\mathbf{u}_i$ between two successive poses $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, so that

$$\mathbf{x}_{i+1} = f(\mathbf{x}_i, \mathbf{u}_i) + \mathbf{w}_i \tag{1}$$

Furthermore, the front-end part of the system can detect loop closures between two poses $\mathbf{x}_i$ and $\mathbf{x}_j$, for example by a visual place recognition system like [4] or [29]. These loop closures are expressed as a constraint of the form

$$\mathbf{x}_j = f(\mathbf{x}_i, \mathbf{u}_{ij}) + \boldsymbol{\lambda}_{ij} \tag{2}$$

Here f is a usually nonlinear function that implements the motion model of the robot and the $\mathbf{x}_i$ and $\mathbf{x}_j$ are the unknown robot poses. $\mathbf{w}_i$ and $\boldsymbol{\lambda}_{ij}$ are zero-mean Gaussian error terms with covariances Σ_i and Λ_{ij} respectively.

The pose graph SLAM problem can be conveniently modelled as a factor graph [17]. Fig. 2 illustrates the general layout of such a graph. The large nodes represent the unknown robot poses and small nodes represent the probabilistic constraints, the so called factors. The conditional probability over all variables (robot poses) $X = \{\mathbf{x}_i\}$ and constraints $U = \{\mathbf{u}_i \cup \mathbf{u}_{ij}\}$ can be expressed as

$$P(X|U) \propto \underbrace{\prod_i P(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{u}_i)}_{\text{Odometry Constraints}} \cdot \underbrace{\prod_{ij} P(\mathbf{x}_j|\mathbf{x}_i, \mathbf{u}_{ij})}_{\text{Loop Closures}} \tag{3}$$

Given the set of constraints U and variables X, we seek the *optimal*, i.e. maximum a posteriori configuration of robot poses, X^*. This most likely variable configuration is equal to the mode of the joint probability distribution $P(X, U)$. In simpler words, X^* is the point where that distribution has its maximum. So under the assumption that above conditional probabilities are Gaussian, the optimal variable configuration X^* can be determined by maximizing the joint probability from above:

$$\begin{aligned} X^* &= \operatorname*{argmax}_X P(X|U) = \operatorname*{argmin}_X -\log P(X|U) \\ &= \operatorname*{argmin}_X \sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\Sigma_i} \\ &\quad + \sum_{ij} \|f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j\|^2_{\Lambda_{ij}} \end{aligned} \tag{4}$$

which is a nonlinear least squares problem. Here and throughout the paper $\|a - b\|^2_\Sigma$ denotes the squared Mahalanobis distance with covariance Σ.

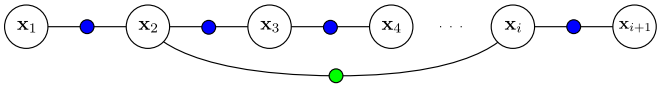


Fig. 2: Factor graph representation of the pose graph SLAM problem. The large vertices represent the unknown robot poses, while probabilistic constraints between them are expressed by the small vertices. Odometry constraints between successive robot states are shown in blue. The green vertex represents a loop closure factor between two non-successive poses.

B. Discussion

As we have seen in (4), the pose graph SLAM problem can be formulated as a nonlinear least squares problem. Due to its inherent sparse structure the problem can be solved efficiently with state of the art frameworks like g2o [18] or GTSAM [1] that use iterative solvers such as Gauss-Newton or Levenberg-Marquardt. However, *outliers* like data association errors and especially false positive loop closures can cause the least squares optimization to fail and converge towards a wrong solution. Fig. 3(a) and 3(b) illustrate this problem.

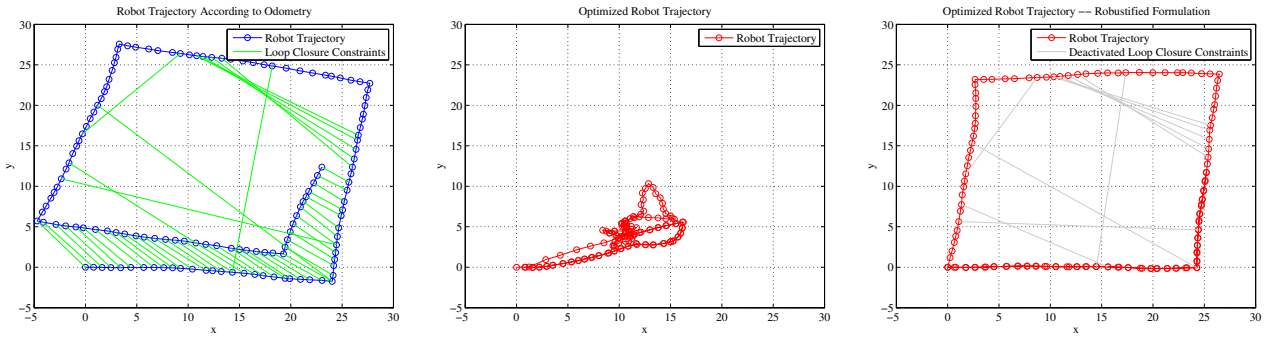
The best outlier mitigation strategy is to avoid outliers in the first place. Many proposed place recognition approaches explicitly try to detect and filter potential false positive loop closures. Although many of these strategies [2], [4], [15], [23], [25], [27] are appealing and reach high precision and recall measures in several benchmarks, the bottom line is that *none* of the current approaches can *guarantee* to always work perfectly and never let a false positive pass.

Therefore, outliers arising from data association errors such as false positive loop closures are still considered a serious problem by the current SLAM literature [2], [5], [8], [9], [20]–[22] As even a single wrong loop closure constraint can cause the whole SLAM system to fail, the back-end should not have to rely solely on the front-end data association. It should rather be able to mitigate the existing outliers or even change the data association decisions made by the front-end, if they appear to be false at a later time during the optimization.

So called *robust cost functions*, like the Huber function [12], can reduce the influence of potential outliers. The idea of Huber is that the error function for data points whose error is above a certain threshold (also called the *kernel width*) should raise linearly instead of quadratically as is normally the case in least squares. This behaviour can be added easily to existing least squares solvers and is for instance optionally available in g2o [18]. However, robust cost functions are not sufficient to deal with outlier constraints like false-positive loop closures since the influence of outliers is merely reduced, but not removed. This however can still lead to defective solutions,as we see in Fig. 3(b) where a Huber function with kernel width 0.1 was used.

III. A ROBUST BACK-END FOR SLAM

In Fig. 3(b) we saw that false positive loop closure constraints are a severe problem. They corrupt the pose



(a) (b) (c)

Fig. 3: A small synthetic dataset: A simulated robot was driven on a square-formed trajectory, collecting odometry information and performing visual place recognition on the way. (a) Initial trajectory (blue) as estimated by the noisy odometry sensor alone. Loop closures requested by the place recognition in the front-end are shown in green. Notice that ten of the loop closures are obviously false-positives, as they connect non-corresponding places. (b) Maximum a posteriori estimate of the robot trajectory calculated by g2o based on the state of the art pose graph formulation. Although the Huber cost function, that is supposed to reduce the influence of outliers, was used (kernel width 0.1), the optimization converged to a clearly defective solution. (c) The robust formulation proposed in this paper allows the optimizer to identify and disable the false positive loop closures (still visible in grey) and converge towards a correct solution. Notice that this works not only for sporadic single errors, but also for the systematic and mutually consistent wrong loop closures in the upper right corner of (a).

graph formulation of the SLAM problem with erroneous edges, leading to a topologically incorrect graph. Current solvers that rely on the front-end to produce a correct graph are doomed to converge towards a defective solution in the presence of outliers.

Our main idea to overcome this problem is that the topology of the graph should be subject to the optimization instead of keeping it fixed. If constraint edges representing outliers and data association errors could be identified and removed during the optimization process, the graph topology would be corrected and the optimization could converge towards a correct solution.

This leads to an augmented optimization problem: We do not only seek the optimal configuration X^* of robot poses, but at the same time we also seek the optimal topology of the constraint graph. With *optimal* topology we mean that it is free of outlier edges. Optimizing the topology of the graph means to alter the optimization problem during the optimization process, which appears to be a "chicken or the egg" type of problem. It is clear that we can not allow the optimizer to change or reformulate the optimization problem totally randomly. What we want to achieve is that suspicious edges representing data association errors or outliers can be removed from the graph. So we can limit the operations the optimizer can conduct on the graph representation of its problem: We only allow to remove existing edges. No other operations are permitted, especially not to add new edges or to add or remove any of the vertices.

Removing an edge from the factor graph corresponds to disabling the constraint associated with that edge, meaning that the disabled constraint should not have any influence on the optimization process, it should be completely removed from the problem formulation. A binary weight factor ω_{ij} would allow us to do just that: It could disable or enable its associated constraint if $\omega_{ij} \in \{0, 1\}$, i.e. ω_{ij} is either 0 or 1:

$$X^* = \underset{X}{\operatorname{argmin}} \sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\Sigma_i} + \sum_{ij} \omega_{ij} \cdot \|f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j\|^2_{\Lambda_{ij}} \quad (5)$$

Equally we can put the weight ω_{ij} into the squared Mahalanobis distance.

If the weight ω_{ij} is 1, the associated constraint is fully respected in the optimization process. In contrast, if the weight is 0, the constraint is completely ignored in the optimization and has no influence on the optimization result, as if it would not exist at all.

The standard pose graph SLAM problem formulation corresponds to the case where all weights are constant and fixed to $\omega_{ij} = 1$. However, if these weights were not fixed, but were themselves subject to the optimization and could be changed by the optimizer during the optimization process, we would in principle have achieved the desired behaviour: The topology of the constraint graph would be subject to the optimization process.

As the weights ω_{ij} shall not be fixed but subject to the optimization, they have to be *variables* of the optimization problem, just like the unknown robot poses $\mathbf{x}_i$ are variables. Since the weights shall either enable or completely disable their associated loop closure constraint, the domain of the weights should be the set $\{0, 1\}$. However, such discrete variables are not suited for least squares optimization methods like Levenberg-Marquardt, which require continuous domains.

Instead of using the discrete weights themselves as variables in the optimization problem, we introduce a continuous variable $s_{ij} \in \mathbb{R}$ for each weight ω_{ij}. We call the set $S = \{s_{ij}\}$ the *switch variables*. We furthermore need a

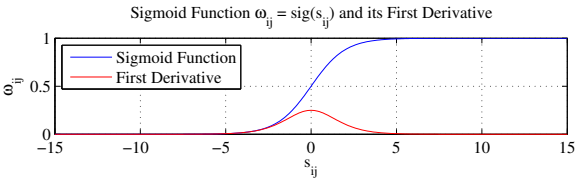


Fig. 4: The sigmoid function $\text{sig}(s_{ij}) = 1/(1 + e^{-s_{ij}})$ and its derivative $\text{sig}'(s_{ij}) = \text{sig}(s_{ij}) \cdot (1 - \text{sig}(s_{ij}))$

switch function

$$\omega_{ij} = \Psi(s_{ij}) : \mathbb{R} \to \{0, 1\} \tag{6}$$

that maps the continuous inputs s_{ij} to the desired weights $\omega_{ij} \in \{0, 1\}$. The s_{ij} would then be the variables in the optimization problem. A suitable function is the sigmoid function

$$\omega_{ij} = \text{sig}(s_{ij}) : \mathbb{R} \to (0, 1) = \frac{1}{1 + e^{-s_{ij}}} \tag{7}$$

which is continuously differentiable with derivative

$$\text{sig}'(s_{ij}) = \text{sig}(s_{ij}) \cdot (1 - \text{sig}(s_{ij})) \tag{8}$$

Both functions are shown in Fig. 4. Notice that the sigmoid function asymptotically converges towards 0 and 1 but never exactly reaches those values. Although we originally wanted the weights to be elements of the discrete set $\{0, 1\}$, now they are elements of the open interval $(0, 1)$ on $\mathbb{R}$. This does not pose a problem, as we are going to see. For input values $s_{ij} \geq 5$ or $s_{ij} \leq -5$, the resulting weight is approximately 1 or 0 respectively.

Like all other variables, the switch variables must be initialized before the optimization starts. Since the loop closure constraints were proposed or requested by the front-end and we want to identify and disable the false-positives, it is reasonable to initially accept all loop closure constraints, i.e. letting $\omega_{ij} = \text{sig}(s_{ij}) \approx 1$. A proper and convenient initial value for all s_{ij} would therefore be 10. For the following, we call these initial values γ_{ij}.

After introducing the new switch variables, the augmented problem formulation is still incomplete: Intuitively nothing prevents the optimization procedure from driving the switch variables s_{ij} towards small values so that $\text{sig}(s_{ij}) \approx 0$. This would basically mean that all loop closure constraints are ignored and cannot contribute to the overall cost function, hence cannot influence the solution of the sought robot poses X^*.

We could easily avoid this behaviour if we would, in simple words, penalize the optimizer whenever it tries to disable a loop closure constraint. This can be achieved by introducing a *prior constraint* for each of the switch variables. The prior constraints should loosely anchor the switch variables s_{ij} at their initial values γ_{ij}. The penalty should therefore be a function of the difference between s_{ij} and its initial value γ_{ij}. We can conveniently use the squared Mahalanobis distance $\|\gamma_{ij} - s_{ij}\|^2_{\Xi_{ij}}$ with covariance Ξ_{ij} to express this additional constraint. Notice that we will have to find a suitable Ξ_{ij}, which we will explain later.

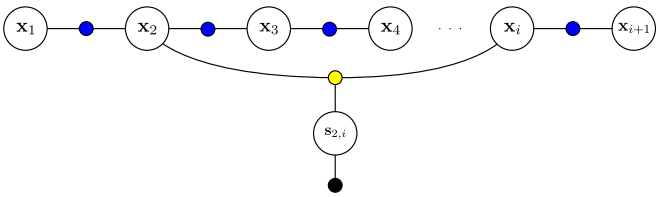


Fig. 5: Factor graph representation of the robustified pose graph SLAM problem proposed in this paper. Notice the additional switch variable $s_{2,i}$ that governs the loop closure factor (now shown in yellow). Depending on the value assigned to the switch variable s_{ij}, the loop closure factor is switched on or off, i.e. it is activated or deactivated as part of the optimization process. The switch variable is governed by a prior factor (black) that penalizes the deactivation of loop closures.

We can now add the switch variables and their prior constraints to the problem formulation of (4) to arrive at the final robust optimization problem for pose graph SLAM:

$$\begin{aligned} X^*, S^* = \underset{X,S}{\text{argmin}} & \underbrace{\sum_i \|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\Sigma_i}}_{\text{Odometry Constraints}} \\ & + \underbrace{\sum_{ij} \|\,\text{sig}(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j)\,\|^2_{\Lambda_{ij}}}_{\text{Switched Loop Closure Constraints}} \\ & + \underbrace{\sum_{i,j} \|\gamma_{ij} - s_{ij}\|^2_{\Xi_{ij}}}_{\text{Switch Prior Constraints}} \end{aligned} \tag{9}$$

Fig. 5 illustrates the robustified factor graph, showing the new switch variable along with its prior factor.

Above formula (9) constitutes our proposed robust problem formulation for pose graph SLAM. It is an optimization problem over two sets of variables, the robot poses $X = \{\mathbf{x}_i\}$ and the switch variables $S = \{s_{ij}\}$, and consists of three types of constraints. The odometry constraints equal those in the standard problem formulation, and represent the pose-to-pose motion information. The loop closure constraints are now constraints between three variables ($\mathbf{x}_i, \mathbf{x}_j$, and s_{ij}) and are weighted by the variable weight factor $\text{sig}(s_{ij})$. This way, by driving the switch variable to small values, the optimization procedure can deactivate the associated loop closure constraints. The third type of constraint, the switch prior constraints, penalizes the deactivation of loop closure constraints.

As we have seen, it is indeed possible to find a formulation that allows the optimizer to change its own problem representation naturally, during the optimization process.

Notice that until now we only gave the robust problem formulation in terms of a cost function that is to be minimized in order to find the optimal variable configuration of X^* and S^*. We now have to analyze the constraints or factors from a probabilistic perspective, as we have to make sure that the optimal configuration of X^* and S^* corresponds to the

maximum a posteriori solution.

Remember from above that each switched loop closure constraint contributes to the overall cost function with the term

$$\|\mathbf{e}_{ij}^{\text{slc}}\|^2_{\Lambda_{ij}} = \| \operatorname{sig}(s_{ij}) \cdot (f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j) \|^2_{\Lambda_{ij}} \tag{10}$$

We arrived at this formulation after considerations that the optimizer should be able to disable any loop closure constraint: By driving s_{ij} towards negative values, the optimizer can "switch off" the loop closure constraint, because in this case $\operatorname{sig}(s_{ij}) \approx 0$ and the spatial distance between $f(\mathbf{x}_i, \mathbf{u}_{ij})$ and $\mathbf{x}_j$ does not add to the global error terms.

Although this interpretation is easy to understand intuitively, the effect of the switch variable can also be understood as acting upon the entries of the information matrix Λ_{ij}^{-1} that is associated with the loop closure constraint via the squared Mahalanobis distance $\| \cdot \|^2_{\Lambda_{ij}}$.

Starting from (10) and using the definition of the Mahalanobis distance we can write

$$\|\mathbf{e}_{ij}^{\text{slc}}\|^2_{\Lambda_{ij}} = [\operatorname{sig}(s_{ij}) \cdot \boldsymbol{\delta}_{ij}]^{\mathsf{T}} \, \Lambda_{ij}^{-1} \, [\operatorname{sig}(s_{ij}) \cdot \boldsymbol{\delta}_{ij}] \tag{11}$$

if we set $f(\mathbf{x}_i, \mathbf{u}_{ij}) - \mathbf{x}_j = \boldsymbol{\delta}_{ij}$. Using the fact that $\operatorname{sig}(s_{ij})$ is scalar, we can transform that equation and arrive at

$$\|\mathbf{e}_{ij}^{\text{slc}}\|^2_{\Lambda_{ij}} = \boldsymbol{\delta}_{ij}^{\mathsf{T}} \left[\operatorname{sig}(s_{ij})^2 \Lambda_{ij}^{-1} \right] \boldsymbol{\delta}_{ij} \tag{12}$$

This last formulation is interesting, because we see that the switch variables s_{ij} directly influence the resulting information matrices

$$\Phi_{ij}^{-1} = \operatorname{sig}(s_{ij})^2 \cdot \Lambda_{ij}^{-1} \tag{13}$$

In this interpretation, if the variable s_{ij} is driven towards negative values, then $\operatorname{sig}(s_{ij}) \approx 0$ and thus the resulting information matrix Φ_{ij}^{-1} will be close to zero. This however, informally expresses that the associated constraint is to be ignored in the optimization process, because literally nothing is known about it. In other words, the associated uncertainty expressed in the covariance matrix Φ approaches infinity.

Both interpretations, driving the information measure or the resulting error towards zero, topologically correspond to removing the associated edge from the graph that represents the optimization problem. However, the interpretation where the switch variables influence the information matrix is to be preferred, as it allows us to still consider the switched loop closure constraints as Gaussian conditional probabilities $P(\mathbf{x}_j | \mathbf{x}_i, \mathbf{u}_{ij}, s_{ij})$ with

$$\mathbf{x}_j \sim \mathcal{N} \left(f(\mathbf{x}_i, \mathbf{u}_{ij}), \frac{1}{\operatorname{sig}(s_{ij})^2} \boldsymbol{\Lambda}_{ij} \right) \tag{14}$$

This allows us to extend (3) and to formulate the conditional probability over all variables (robot poses $X = \{\mathbf{x}_i\}$ and switch values $S = \{s_{ij}\}$) and constraints $U = \{\mathbf{u}_i \cup \mathbf{u}_{ij}\}$:

$$P(X, U) \propto \prod_i P(\mathbf{x}_{i+1} | \mathbf{x}_i, \mathbf{u}_i) \cdot \prod_{ij} P(\mathbf{x}_j | \mathbf{x}_i, \mathbf{u}_{ij}, s_{ij}) \ldots \cdot \prod_{ij} P(s_{ij} | \gamma_{ij}) \tag{15}$$

Under the assumption that all probabilities above are Gaussian, the solution (X^*, S^*) of the optimization problem in (9) is indeed the maximum a posteriori solution, since (X^*, S^*) maximizes $P(X, U)$.

A. First Results

Now that the general idea of our approach and the mathematical formulation has been laid out, it is time to come back to the introductory example in Fig. 3. In addition to the odometry-based trajectory in Fig. 3(a) and the bad solution of g2o in Fig. 3(b), we show the results of our proposed robustified pose graph SLAM formulation in Fig. 3(c).

As one can clearly see, despite the wrong loop closures, the robustified optimization converges towards a correct solution. The grey lines in the plot represent the loop closure requests that have been disabled during the optimization, i.e. their associated switch variables s_{ij} have been assigned values between approximately -5 and -10 so that the resulting weight factor $\omega_{ij} = \operatorname{sig}(s_{ij}) \approx 0$.

It is important to notice that none of the correct loop closures were disabled by the optimizer. Their associated switch values do not differ from the initial values and remain stable at 10. In terms of precision-recall statistics this example therefore yields an optimal result with both 100% precision and recall.

B. Discussion

Before we present results of the proposed robustified pose graph SLAM formulation on several larger datasets, we want to discuss the implications that arise from it with respect to the size and hardness of the optimization problem.

1) Influence on the Problem Size: Obviously, the proposed robustified problem formulation significantly enlarges the original optimization problem. For each loop closure constraint present in the original problem, another variable and an associated prior factor are added to the problem. However, given today's efficient solvers that exploit the sparseness of the optimization problem, the *size* of the problem (i.e. the number of variables and constraints) is not the most crucial factor that determines the runtime behaviour. By far more important are the sparse structure of the system's Jacobian and a beneficial convergence behaviour (e.g. convexity or close-convexity of the problem).

2) Influence on the Sparseness of the Problem: The initial sparse structure of the optimization problem is not altered by the additional switch variables and their prior factors. This is clear because each of the switch variables governs only one loop closure edge. The Jacobian $\mathbf{J}_{ij}^{\text{slc}}$ for a switched loop closure constraint is formed by the partial derivatives with respect to the variables $\mathbf{x}$ and $\mathbf{s}$. The only non-zero entries in that Jacobian are the partial derivatives with respect to $\mathbf{x}_i$, $\mathbf{x}_j$, and s_{ij}. All other entries are $\mathbf{0}$. The Jacobians $\mathbf{J}_{ij}^{\text{sp}}$ for the new switch prior constraints are equally sparse, because each of the switch variables is influenced by only one prior factor. Thus the only non-zero entry of $\mathbf{J}_{ij}^{\text{sp}}$ is the partial derivative with respect to s_{ij}. The sparse structure of the overall Jacobian is illustrated in Fig. 6.

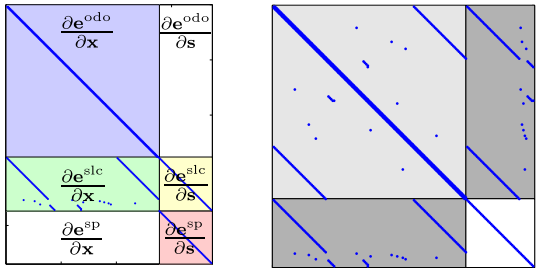


Fig. 6: Overall sparse Jacobian $\mathbf{J}$ (left) and Hessian $\mathbf{H} = \mathbf{J}^\top\mathbf{J}$ (right) for the example SLAM problem from Fig. 3, modelled using the proposed robust back-end. The non-zero values are marked by a blue point. The light grey block in $\mathbf{H}$ contains the connections between two pose nodes, connections between a switch variable and a pose node are represented in the darker areas.

3) Influence on the Problem's Convergence Properties: The influence of the additional variables on the convergence behaviour is harder to determine. However, currently not even the structure of the standard pose graph SLAM problem in terms of its convergence properties has been exhaustively explored. In [11] Huang et al. raised the question of how far SLAM is from a linear (i.e. convex) least squares problem and suggested a *close-convexity* under certain circumstances, e.g. small initial angular deviations and spherical covariance matrices without off-diagonal entries. They also showed that a relative pose formulation helps to strengthen this close-to-convex property. Recently Carlone et al. [3] presented a working closed-form, linear approximation to SLAM. In this formulation the SLAM problem, like all linear least squares problems, can be solved immediately without requiring an iterative solver or an initial guess. The prerequisite for this approach again are diagonal shaped covariances, with independent position and orientation measurements.

As we have seen, the convergence properties of the SLAM problem are still under research. It would be desirable to show that our robustified SLAM formulation does not change the convergence properties of the underlying standard SLAM problem. Although the results presented in this paper indicate a stable convergence behaviour, the mathematical examination is still left for future work.

IV. Implementation and Parameters

We implemented our robust back-end in C++ for both the GTSAM [1] and g2o [18] frameworks. Both libraries already provide efficient solvers for graph-based SLAM problems as well as classes for different factors common in 2D- and 3D-SLAM. The experiments described later in this paper have all been conducted with the GTSAM implementation.

There is only one free parameter that needs to be set: The covariance matrix Ξ is used in the switch prior constraint $\|s_{ij} - l_{ij}\|_{\Xi_{ij}}$: It is a one-dimensional variance measure and was empirically set to $\Xi_{ij} = 20^2$ for all experiments described later in this paper. The other covariance matrices Λ_{ij} and Σ_i are used to calculate the Mahalanobis distances in the odometry and loop closure factors and have to be provided by the front-end.

TABLE I: The synthetic datasets used during the evaluation.

Dataset	2D/3D	Poses	Loop Closures
Manhattan	2D	3500	2099
City	2D	10000	10688
Sphere 2500	3D	2500	2450

All experiments were conducted in an incremental fashion, i.e. data was fed into the optimizer 200 frames at a time, in contrast to performing batch optimization.

V. Results on Synthetic Data Sets

To benchmark the robustness of the proposed approach and to show its correctness and feasibility, we used three synthetic datasets that have been used in a number of publications before. We thank the original authors for providing the data: The Manhattan world was first published by Olson [26]. The datasets City and Sphere shipped with the open-source implementation of iSAM [14]. All of these datasets consist of pose graphs in 2D or 3D and contain several thousand poses and loop closure constraints (see Table I). We corrupted the data by adding different numbers (between 5 and 1000) of wrong loop closures between randomly chosen poses. These additional, wrong loop closures were given the same information matrix entries as the correct ones provided by the datasets.

Fig. 1 shows some of the results for the Manhattan World dataset. We conducted 10 experiments, adding 5, 10, 25, 50, 75, 100, 250, 500, 750 and 1000 random wrong loop closures. As expected, current state of the art frameworks are not able to cope with the outlier constraints and converge towards defective solutions. This is illustrated in the left part of Fig. 1, which shows the results of iSAM [14] version 1.5. for 10 outlier constraints. Even g2o [18], that can optionally use the Huber cost function is not able to converge correctly (not shown). Our robust back-end however, is able to converge to a correct solution even with 1000 false loop closure constraints (right part of figure). The outlier constraints are correctly deactivated by driving their associated switch variable s_{ij} to small values, so that the resulting weight factor for that constraint, ω_{ij} is approximately zero. Notice that 1000 outliers are roughly 50% of the original true loop closings in the dataset. Apart from merely visual inspection, the RMS errors and average χ^2 errors specified in Table II prove the good quality of the optimization result. From Fig. 7 we can see that the resulting average χ^2 errors appear to be more or less constant, fluctuating around a value of 0.07 for the different experiments with the Manhattan dataset. The required time until convergence rises super-linearly with the number of outliers.

Precision-recall statistics allow us to quantitatively judge how well the system identifies the outlier loop closure constraints, while maintaining the correct ones. Fig. 8(a) shows the precision-recall plot. It was calculated over a total of 30 experiments, 10 for each of the three datasets, with an increasing number of added (wrong) loop closure constraints.

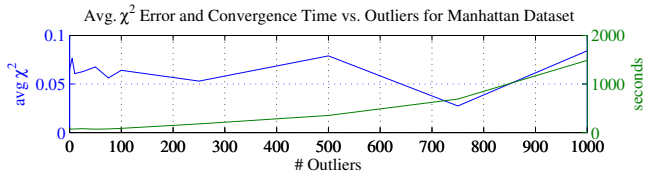


Fig. 7: With our robustified problem formulation, the average χ^2 error after the optimization is relatively constant, even for very large numbers of additional false-positive loop closure constraints. The time until convergence increases super-linearly with the number of outliers.

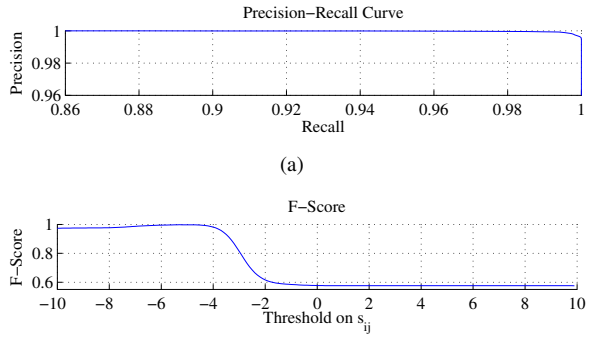


Fig. 8: (a) Precision-recall curve of accepted loop closure constraints calculated over a total of 30 experiments for three different datasets with up to 1000 wrong loop closure constraints. Notice the scale of the axes. At a precision of exactly 100%, the recall is $\approx 90\%$. (b) Correspondingt F-score plot. The maximum F-score of 0.9979 is reached at $l_{ij}^* = -5.3$.

The high quality of the system is apparent. At a precision of 100% (i.e. all wrong loop closures are correctly identified and discarded) a recall of 90% is reached. That means that only 10% of all correct loop closures are erroneously discarded by the system. This generally is not a problem as real loop closures usually contain many poses and chances are very high that at least one of these single loop closing constraints will be accepted by the optimizer.

The corresponding F-Score plot in Fig. 8(b) illustrates the very desirable behaviour: The maximum F-score is almost 1 (0.9979) and shows a clear peak in the plot. The best threshold on s_{ij} that decides whether a loop closure constraint should be considered active or discarded is $s_{ij}^* = -5.3$. This complies with the observation that the sigmoid function used to implement the switching behaviour (Fig. 4) significantly starts to differ from 0 for input values greater -5.

Selected results from the City and Sphere datasets are illustrated in Fig. 9(b) and 9(c), respectively. We also conducted experiments where wrong loop closures were not added totally randomly but in groups of 20 mutually consistent single loop closures. The achieved results are qualitatively equal to those of the random tests.

VI. RESULTS ON A REAL-WORLD DATA SET

Although the synthetic datasets already contain several thousand poses and constraints, we wanted to test our robust back-end in an even larger real-world scenario. We therefore chose the St. Lucia dataset that was first presented in [24]. It consists of video footage taken on a 66 km long course along the roads in a suburb of Brisbane, Australia. The camera was mounted on top of a car that drove through the street network for a little more than 1:40 hours, resulting in 57,858 image frames which correspond to distinct poses. No additional information is available, notably no GPS, or odometry information.

TABLE II: Error Measures for Manhattan Data Set.

Method	Robust Approach			Non-Robust
# Outliers	10	100	1000	10
RMSE_{xy} [m]	0.996	1.073	1.169	21.1
RMSE_{θ} [deg]	2.8	2.9	3.4	39.05
avg. χ^2-Error	0.061	0.064	0.084	22.9

The front-end part of our system therefore has to extract inter-frame motion information and detect loop closures solely from the camera images: Coarse odometry information was extracted from the images using image profile matching. Details can be found in [28]. Although this technique is rather simple, the extracted inter-frame motion estimates provide sufficient metric information for the SLAM back-end.

Potential loop closures were detected by a light-weight place recognition system we call BRIEF-Gist [29]. This rather simple place recognition system has a low false-negative rate, but a pretty high false-positive rate. However, due to the robust problem formulation, the optimizer is able to deactivate these wrong loop closures and converge to a correct solution, which is depicted in Fig. 9(a).

This real-world dataset shows why a naive approach that deactivates loop closure constraints based upon a simple error threshold cannot work: Some of the loops (especially the first and the 5th) are very large and the accumulated odometry errors already exceed several hundred meters. In order to accept these loop closures, any threshold would have to be set to such high values that most of the wrong loop closure candidates (that require shorter loops) would be erroneously accepted.

VII. CONCLUSIONS

We presented a novel approach to improve the robustness of optimization-based pose graph SLAM. Our modified problem formulation can be understood as transferring parts of the responsibility for correct data association from the front-end into the back-end. The back-end optimizer can now change the topological structure of the pose graph representation during the optimization process. Therefore, it can account for possible data association errors and ignore erroneous loop closure constraints. We feel that our work closes a gap between the recently developed back-end optimizers and the sensor-driven front-end.

The main advantages of the proposed formulation are:

- The overall SLAM system becomes tolerant and robust against errors in the data association.

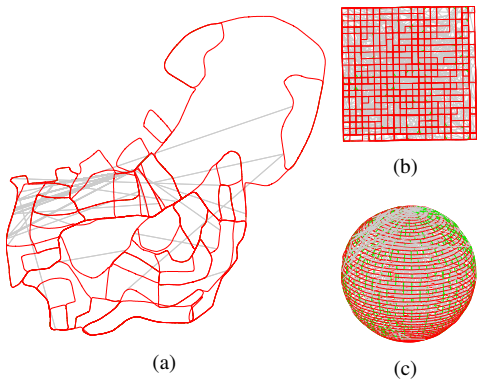


Fig. 9: (a) Results for the St. Lucia dataset [24], consisting of video footage of a 66 km long drive through an urban environment. Loop closure recognition was performed using a very simple technique based on the BRIEF descriptor. We described this approach to place recognition in previous work [29]. Despite the high number of wrong loop closure requests, the robust problem formulation allows the optimizer to converge. (b) Results on the City world dataset and on the Sphere world dataset (c). Both datasets were spoiled by respectively 500 and 300 additional false loop closure constraints. Despite that, the robust system converges. The additional false loop closures (grey) were identified and deactivated during the optimization.

- The data association algorithm does not need to work perfectly (precision $< 100\%$) and can be kept simple and fast, as a reasonable false positive rate is acceptable.
- No hard data association decisions are necessary in the front-end, the optimizer can take back decisions at any time.

The ideas we presented in this work do not only apply to pose graph SLAM, but can be extended to general nonlinear least squares optimization problems in which outliers may be present. Still left for future work is a mathematical discussion of the convergence properties of the robustified SLAM problem. Supplementary material is available at www.tu-chemnitz.de/etit/proaut/forschung/robustSLAM.html

Acknowledgements

We thank Michael Milford from Queensland University of Technology for providing the St. Lucia video footage that was presented in his paper [24]. Further material on RatSLAM is available at `http://ratslam.itee.uq.edu.au`.

References

[1] Gtsam. https://collab.cc.gatech.edu/borg/gtsam/.

[2] Cesar Cadena, John McDonald, John J. Leonard, and José Neira. Place Recognition using Near and Far Visual Information. In *Proc. of the 18th IFAC World Congress*, 2011.

[3] Luca Carlone, Rosario Aragues, Jose Castellanos, and Basilio Bona. A linear approximation for graph-based simultaneous localization and mapping. In *Proc. of Robotics: Science and Systems, RSS*, 2011.

[4] Mark Cummins and Paul Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[5] Mark Cummins and Paul Newman. Highly Scalable Appearance-Only SLAM – FAB-MAP 2.0. In *Robotics Science and Systems*, 2009.

[6] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *Intl. J. of Robotics Research, IJRR*, 25(12), Dec 2006.

[7] U. Frese and L. Schröder. Closing a Million-Landmarks Loop. In *Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[8] Dorian Galvez-Lopez and Juan D. Tardos. Real-time loop detection with bags of binary words. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 51 –58, sept. 2011.

[9] Arren Glover, William Maddern, Michael Milford, and Gordon Wyeth. FAB-MAP + RatSLAM : Appearance-Based SLAM for Multiple Times of Day. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA2010)*, Anchorage, Alaska, May 2010.

[10] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3), 2009.

[11] Shoudong Huang, Yingwu Lai, U. Frese, and G. Dissanayake. How far is SLAM from a linear least squares problem? In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

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

[13] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, 2011.

[14] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental Smoothing and Mapping. *IEEE Transactions on Robotics*, 24(6), 2008.

[15] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. *International Journal of Robotics Research (IJRR)*, 29(10), 2010.

[16] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent. Efficient sparse pose adjustment for 2d mapping. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[17] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001.

[18] 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 Robotics and Automation (ICRA)*, 2011.

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

[20] Will Maddern, Arren Glover, Michael Milford, and Gordon Wyeth. Augmenting RatSLAM using FAB-MAP-based Visual Data Association. In *Proceedings of Australasian Conference on Robotics and Automation (ACRA09)*, 2009.

[21] Will Maddern, Michael Milford, and Gordon Wyeth. Continuous Appearance-based Trajectory SLAM. In *International Conference on Robotics and Automation (ICRA)*, 2011.

[22] Lina Mar, Pedro Pini, and Dorian G. CI-Graph Simultaneous Localization and Mapping for Three-Dimensional Reconstruction of Large and Complex Environments Using a Multicamera System. *Journal of Field Robotics*, 27(5):561–586, 2010.

[23] John McDonald, Michael Kaess, Cesar Cadena, Jos Neira, and John J Leonard. 6-DOF Multi-session Visual SLAM using Anchor Nodes. In *Proc. of European Conference on Mobile Robots, ECMR*, pages 69–76, 2011.

[24] Micheal J. Milford and Gordon F. Wyeth. Mapping a Suburb with a Single Camera using a Biologically Inspired SLAM System. *IEEE Transactions on Robotics*, 24(5), October 2008.

[25] J. Neira and J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[26] Edwin Olson, John Leonard, and Seth Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Inl. Conf. on Robotics and Automation, ICRA*, 2006.

[27] Edwin Olson, Matthew Walter, Seth Teller, and John Leonard. Single-cluster spectral graph partitioning for robotics applications. In *Robotics: Science and Systems (RSS)*, 2005.

[28] Niko Sünderhauf and Peter Protzel. Beyond RatSLAM: Improvements to a Biologically Inspired SLAM System. In *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*, 2010.

[29] Niko Sünderhauf and Peter Protzel. BRIEF-Gist – Closing the Loop by Simple Means. In *Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[30] Thrun, Burgard, and Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, Massachusetts, London, England, 2005.