Switchable Constraints for Robust Pose Graph SLAM

Abdelaziz Ben Othman

***Abstract*—SLAM algorithms that can deduce a trustable and accurate mapping without being affected by the presence of outliers has recently attracted increasing attention.**
The task of constructing the graph is delegated to the front-end that has access to some visual and inertial sensors. In the other hand, the back-end of the system relies heavily on these information and any data association errors in the front-end level will lead to catastrophic implications on the resultant map. Instead of focusing on another data fusion approach, the proposed solution focuses on building a robust back-end capable of detecting and rejecting outliers using switchable constraints during optimization.
Keywords: Self-Localization, loop closure, robust back-end

I. Introduction

HUMANS invented robots to perform for them tasks that are dangerous or boring with both consistency and precision. By the time, robots evolved and are now doing more complex tasks (searching and rescuing people, exploring deep oceans, dunes on Mars etc.)
Such tasks are most often done in unknown environments where robots have to properly react to unknown occurrences. To do so, these intelligent machines have to reason about their environment by using the SLAM approach, which is an acronym for simultaneous localization and mapping. The architecture of SLAM systems is composed by two main components: the front-end and the back-end [5]. The front-end collects information using sensors, performs data association, builds and maintains the graph representation of the SLAM problem while the back-end solves this problem by applying non-linear least squares equations. While Simultaneous Localization and Mapping is in principle a solved problem, building a robust SLAM system is a challenging engineering problem and is still far from being an established and reliable technology.
This is noticed during what has become known as loop closing where the front-end has the full responsibility of implementing sophisticated algorithms that actively prevent the occurrence of data association or place recognition errors. In the other hand, the back-end relies heavily on the performance of the front-end and like all least squares problems solver, the presence of outliers like false positive loop closures will lead to a corrupted solution of the SLAM problem. Although, the problem is acknowledged in literature, no concise solution has been proposed so far.[1] [2] Since robustness is what prevents the robots from being applied outside controlled environments or specialized domains, enhancing the robustness of the SLAM is the scope of this paper. In this approach, hidden variables known as the switchable variables are introduced to allow the back-end changing 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.

II. SLAM as a Nonlinear Least Squares Optimization Problem

After introducing the SLAM in general, we are going to concentrate now on a specific type of SLAM and show how it can be solved as a least squares optimization problem.

Advisor: Prof.Dr Marcus Baum and Shishan Yang, Lehrstuhl für Informatik mit Schwerpunkt Sensorik, University of Passau, WS 2017/2018.

A. Motion models

Let's define x_t as the robot's poses over discrete time steps t and M as the map the robot is discovering. The two terms are unknown a priori and have to be estimated using the odometry information that describe the movement between single poses.
The relation between two successive poses is given as:

$$x_{t+1} = f(x_t, u_t) + w_t \tag{1}$$

where f is a non-linear motion model, u_t is the odometry measurement between the poses and w_t is the noise of the odometry sensor system.
The odometry noise follows a Gaussian distribution with a covariance matrix $\sum_t$:

$$w_t \sim \mathcal{N}(0, \textstyle\sum_t) \tag{2}$$

as a result, x_{t+1} follows as well a Gaussian distribution:

$$x_{t+1} \sim \mathcal{N}(f(x_t, u_t), \textstyle\sum_t) \tag{3}$$

B. Sensor Models and Measurement Functions

In addition to the odometry, the other sensor information are collected into a single variable vector and can be written as:
$$Z_t = \{z_0, z_1, ..., z_T\}$$
T means we take care of all available sensor measurements in time. We define h as the sensor model function that allows us to predict the measurements given the current estimates of the map and the robot pose:

$$z_t = h(x_t, M) + \lambda_t \tag{4}$$

where λ_t captures the sensor noise which is modeled as a Gaussian distribution:

$$z_t \sim \mathcal{N}(h(x_t, M), \Lambda_t) \tag{5}$$

C. The pose graph SLAM problem

The map we're interested in can be expressed as a graph where the vertices represent robot poses x_i and edges represent the spatial constraints between those poses. Such a map is called a pose graph. Both odometry and loop closure constraints are necessary for the pose graph SLAM. The odometry constraint connects two successive states x_i and x_{i+1} via the f model such that:

$$x_{i+1} \sim \mathcal{N}(f(x_i, u_i), \textstyle\sum_i) \tag{6}$$

To perform loop closing, the robot has to recognize places it already visited before. This is done in the front-end and introduces to us the second type of constraint, the loop closure constraints. These constraints connect two not necessarily successive poses x_i and x_j such that:

$$x_j \sim \mathcal{N}(f(x_i, u_{ij}), \Lambda_{ij}) \tag{7}$$

As the robot moves through its environment it uses the odometry sensors to create an initial guess of its trajectory.

D. Deriving a Nonlinear Least Squares Formulation

The key solution of the full SLAM problem for the pose graph is estimating the posterior probability of the robot's trajectory X given all the measurements U.

$$P(X|U) \tag{8}$$

Given the set of the odometry and loop closures constraints u_i, u_{ij}[1] $\in U$, we seek the most likely configuration of robot poses and we denote it as X^* where the distribution of $P(X|U)$ has its maximum.

$$X^* = \underset{X}{\operatorname{argmax}} P(X|U) \tag{9}$$

In order to solve this problem, we can factor the joint probability distribution as:

$$P(X|U) \propto \underbrace{\prod_i P(x_{i+1}|x_i, u_i)}_{\text{Odometry Constraints}} \cdot \underbrace{\prod_{ij} P(x_j|x_i, u_{ij})}_{\text{Loop Closure Constraints}} \tag{10}$$

Under the assumption of the equations (6) and (7), the conditional probabilities above are all Gaussian. Thus we can write the odometry constraint as:

$$\begin{aligned} P(x_{i+1}|x_i, u_i) = \frac{1}{\sqrt{2\pi|\sum_i|}} exp(-\frac{1}{2}(f(x_i, u_i) - x_{i+1})^T \\ \sum_i^{-1}(f(x_i, u_i) - x_{i+1})) \end{aligned} \tag{11}$$

The term $\frac{1}{\sqrt{2\pi|\sum_i|}}$ is the normalizer constant and we replace it by η. We apply the Mahalanobis [2] distance definition to the term inside the exponent component in order to write the odometry constraint more conveniently:

$$P(x_{i+1}|x_i, u_i) = \eta exp(-\tfrac{1}{2}\|f(x_i, u_i) - x_{i+1}\|^2_{\sum_i}) \tag{12}$$

By following the same steps for the loop closure constraints we gain:

$$P(x_j|x_i, u_{ij}) = \eta exp(-\tfrac{1}{2}\|f(x_i, u_{ij}) - x_j\|^2_{\lambda_{ij}}) \tag{13}$$

This leads to the following factorization problem:

$$\begin{aligned} P(X|U) \propto \prod_i exp(-\frac{1}{2}\|f(x_i, u_i) - x_{i+1}\|^2_{\sum_i}). \\ \prod_{ij} exp(-\frac{1}{2}\|f(x_i, u_{ij}) - x_j\|^2_{\lambda_{ij}}) \end{aligned} \tag{14}$$

We transform the products into sum by taking the negative logarithm:

$$-\log P(X|U) \propto \sum_i \|f(x_i, u_i) - x_{i+1}\|^2_{\sum_i} + \sum_{ij} \|f(x_i, u_{ij}) - x_j\|^2_{\lambda_{ij}} \tag{15}$$

Notice that due to our interest on the proportionality in this equation, we dropped the $\frac{1}{2}$ and the η terms. We can now solve the maximum a posteriori solution X^*:

$$\begin{aligned} X^* &= \underset{X}{\operatorname{argmax}} P(X|U) = \underset{X}{\operatorname{argmin}} -\log P(X|U) \\ &= \underset{X}{\operatorname{argmin}} \underbrace{\sum_i \|f(x_i, u_i) - x_{i+1}\|^2_{\sum_i}}_{\text{odometry Constraints}} + \underbrace{\sum_{ij} \|f(x_i, u_{ij}) - x_j\|^2_{\lambda_{ij}}}_{\text{Loop Closure Constraints}} \end{aligned} \tag{16}$$

This however is a least squares optimization problem, since the sought X^* is a minimizer over a sum of squared terms.

[1]The displacement associated associated to a loop between two poses x_i and x_j

[2]The squared Mahalanobis distance is defined as $\|a - b\|^2_{\sum_i} = (a - b)^T \sum^{-1}(a - b)$

E. When Optimization fails

When the robot is exploring the map, it relies on fixed landmarks observations to update its position. The front-end part associates to each known landmark some sensor measurements and to the unknown one new sensor measurements. However, taking into account the sensor reading errors and difficulties to decide on many known landmarks occurring near each other, the robot as consequence can't decide and makes inevitable data association errors. Such errors are known as false-positive loop closure: Due to the high self-similarity of many indoor and outdoor environments, two distinct places can actually appear to be very similar to the sensor (figure.1).

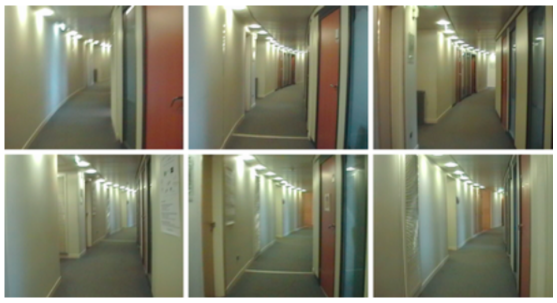


Fig. 1. Images from different places: note the strong similarities between these images.
Source: [3]

III. A ROBUST BACK-END FOR SLAM

In previous section, we saw that false loop closure constraints are a sever problem. They corrupt the pose graph formulation of the SLAM problem with erroneous edges, leading to a topologically incorrect graph. Our main idea is to increase the robustness of the back-end by rendering the topology of the graph subject to optimization instead of keeping it fixed.
Figure1 illustrates the main idea: If we identify and remove outliers during the optimization process, the graph topology would be corrected and the optimization could converge towards a correct solution.

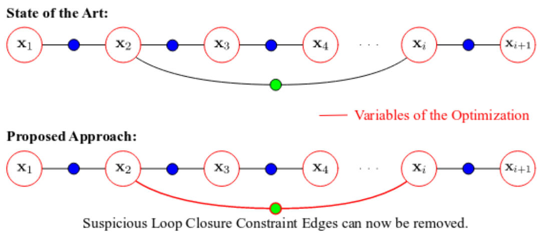


Fig. 2. In current state of the art approaches, only the robot poses x_i are variables in the optimization. We propose to augment the problem and also make the loop closure constraint edges subject to the optimization.
Source: [4]

A. First step towards a Mathematical formulation

Removing an edge from the graph corresponds to removing the constraint associated with that edge. A Loop closure constraint are expressed as we recall in (7). We Disable a loop closure constraint on the graph by completely removing it from the formulation problem. A binary weight w_{ij} would allow us to do so: if $w_{ij} = 0$, the associated constraint is removed else we keep it. Together with the odometry constraint we can write:

$$X^* = \underset{X}{\operatorname{argmin}} \underbrace{\sum_i \|f(x_i, u_i) - x_{i+1}\|^2_{\sum_i}}_{\text{odometry Constraints}} + \underbrace{\sum_{ij} \|w_{ij}.f(x_i, u_{ij}) - x_j\|^2_{\Lambda_{ij}}}_{\text{Loop Closure Constraints}} \quad (17)$$

In next steps, we will make the weights also subject to the optimization instead of keeping them constant in order to achieve the desired behaviour: The topology of the constraint graph is subject to the optimization process.

B. The Switch Variables and Switch Function:

For each weight w_{ij}, we introduce a continuous variable s_{ij} and we call it switch variable. We introduce the switch function $\Psi(s_{ij})$ such that:

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

This function maps the continuous set of the switchable variable s_{ij} to the desired weights $w_{ij} \in \{0, 1\}$.
The s_{ij} would be then the variables in the optimization problem. Different switch functions can be defined e.g a step function or a sigmoid. However the sigmoid function shows an undesired behaviour where the gradient is very small over a large range of values and where the iterative optimizer can become stuck. More elaborate experiments showed that a simple linear function of the form: $w_{ij} = \Psi^{lin}(s_{ij}) = s_{ij}$ results in a better convergence behaviour if we constrain $0 \leq s_{ij} \leq 1$

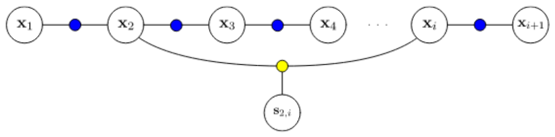


Fig. 3. Factor graph representation of the augmented pose graph SLAM problem after introducing the switch variables

We are now ready to augment the optimization problem and introduce the new switch variables $S = \{s_{ij}\}$

$$S^*, X^* = \underset{X,S}{\operatorname{argmin}} \sum_{ij} \|\mathbf{f}(\mathbf{x_i}, \mathbf{u_i}) - \mathbf{x_{i+1}}\|^2_{\sum_i} + \sum_{ij} \|\mathbf{\Psi}(\mathbf{s_{ij}})(\mathbf{f}(\mathbf{x_i}, \mathbf{u_{ij}}) - \mathbf{x_j})\|^2_{\Lambda_{ij}} \quad (18)$$

Notice now that the equations depends over two sets of hidden variables, the robot poses $X = \{x_{ij}\}$ and the switch variable $S = \{s_{ij}\}$.

C. The Switch Prior Constraint

The switch variables and like all other variables need to be initialized. Keeping in mind that the loop closures are proposed in the front-end. Some of them are false positive and need to be identified and disabled. Initially, we will accept all loop closures constraints, i.e. letting $w_{ij} = \Psi s_{ij} \approx 1$. For the following, we call these initial values γ_{ij} and collect them into the set $\Gamma = \{\gamma_{ij}\}$. The switch variable s_{ij} are modeled as normally distributed Gaussian variables[3].

$$s_{ij} \approx \mathcal{N}(\gamma_{ij}, \Xi_{ij})$$

Note that Ξ_{ij} is the covariance matrix of the initial variables γ_{ij}. The initial values γ_{ij} will be part of the optimization problem using prior constraints. These prior factors express the problem of maximizing $P(S|\Gamma)$.

$$S^* = \underset{S}{\operatorname{argmin}} \textstyle\sum_{ij} \|\gamma_{ij} - s_{ij}\|^2_{\Xi_i} \quad (19)$$

Obviously, we can say that the solution for this sub-problem is $s_{ij} = \gamma_{ij}$. However, the switch variables are involved in equation(17), so that on a global scale the optimal solution may be $s_{ij} \neq \gamma_{ij}$.

D. Putting it all together

We add now the switch prior constraint to the problem formulation of (17) to obtain the final robust optimization problem for pose graph SLAM:

$$S^*, X^* = \underset{X,S}{\operatorname{argmin}} \sum_{ij} \|\mathbf{f}(\mathbf{x_i}, \mathbf{u_i}) - \mathbf{x_{i+1}}\|^2_{\sum_i} + \sum_{ij} \|\mathbf{\Psi}(\mathbf{s_{ij}})(\mathbf{f}(\mathbf{x_i}, \mathbf{u_{ij}}) - \mathbf{x_j})\|^2_{\Lambda_{ij}} + \sum_{ij} \|\gamma_{ij} - s_{ij}\|^2_{\Xi_{ij}} \quad (20)$$

This is our proposed robust problem formulation for the pose graph SLAM. It is an optimization problem equation over two sets of hidden variables, the robot poses $X = \{x_i\}$ and the switch variable $S = \{s_{ij}\}$. We have three type of constraints: the odometry constraints representing the pose-to-pose motion information, the loop closure constraints which is a constraint between three variables (x_i, x_j, s_{ij}) and weighted by the variable w_{ij} and the switch prior constraints which penalizes the deactivation of loop closure constraints.

IV. Preparing the Evaluation

The evaluation part will answer the following questions:

- How robust is the robust back-end?
- What is the impact of the outlier loop closure constraints on the final error of the optimization?
- What influence on the runtime behaviour has robust back-end?

A. Errors metrics for SLAM

While we can judge the performance by visual inspection of the resulting map, we prefer to rely more on quantitative errors metrics to make good decisions. We used the root-mean square error (RMSE) and the relative pose error (RPE). Also, we used the precision-recall to judge how well the robust back-end distinguishes between the true and the false positive loop closures.

B. Datasets for the evaluation

TABLE I
THE DATASET USED DURING THE EVALUATION

dataset	Synthetic/real	2D/3D	Poses	Loop closures
Manhattan(original)	Synthetic	2D	3500	2099
Manhattan(g^2o version)	Synthetic	2D	3500	2099
City 10000	Synthetic	2D	10000	10688
Sphere2500	Synthetic	3D	2500	2450
Intel	real	2D	943	894
Parking Garage	real	3D	1661	4615

We used six different datasets during the evaluation as shown in Table 1. The synthetic datasets are created from simulation, while the two real-world datasets have been recorded in a 2D (Intel) and in a 3D (Parking Garage) environment respectively. For evaluation purposes, different initialization of the same dataset were used to see the impact of the initial estimates on the overall behavior of the back-end system.
For the two latter real-world datasets, we used the estimation results for the outlier-free dataset as pseudo ground truth.

[3]remember from equation (6) and (7)

C. General Methodology

Different methodologies were proposed to introduce how these constraints were added. In real life, these wrong imposed outliers are introduced by the front-end after failing to recognize a place in the map.

D. Policies for Adding Outlier Loop Closure Constraints

For the evaluation, false positive closure constraints were added between two robot poses x_i and x_j to the spoiled datasets. The indices i and j are determined using four different policies.

- Random constraints: This method adds random loop closure between two randomly chosen vertices x_i and x_j following a uniform distribution over all available indices.
  Most of the constraints that are created using this policy will span over large areas of the dataset since they connect two distant poses.
- Local constraints: Following this policy, the first pose is chosen randomly from all possible vertices. Then, the second vertex is chosen so that it's in the spatial vicinity of the first vertex. This follows the fact that nearby poses are most likely to appear similar than distant poses do. Thus false place recognitions are more likely to be established between these nearby poses.
- Randomly Grouped Constraints: The randomly grouped policy picks up first randomly two poses i and j and then adds 20 successive constraints between the poses with indices i...i+20 and j ...j+20.
- Locally Grouped Constraints: This policy is a mixture between local and randomly grouped policies. The first index i is chosen randomly, the second vertex j has to be near the vicinity of i. Then 20 successive constraints between the vertices with indices i... i+20 and j...j+20 are added.

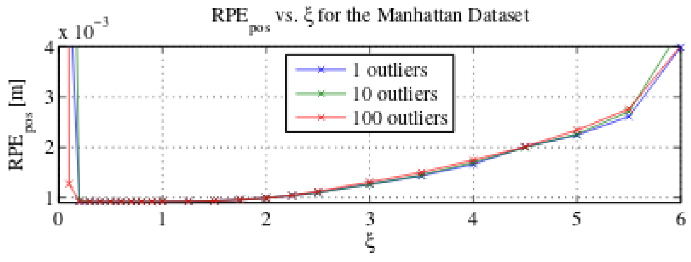
V. Evaluation

After discussing all the theoretical information, we will evaluate in this section the proposed robust-back end approach.

A. The influence of Ξ_{ij} on the Estimation Results

The proposed formula described in equation(19) include the switch prior constraints term $\|\gamma_{\mathbf{ij}} - \mathbf{s_{ij}}\|^2_{\Xi_{ij}}$. Although we can't mathematically deduce the value of the switch prior variances Ξ_{ij}, we can set it empirically. Therefore we will explore the influence of the switch prior variances on the estimated results.
Ξ_{ij} controls the penalty the system gains when the front-end deactivates a loop closure between two poses. By taking into consideration this value for each loop constraint, the front-end could express a degree of confidence about that particular loop closure. Thus, the front-end assign small Ξ_{ij} to loop closures when it is very certain about and large Ξ_{ij} to loop closure that appear more doubtful about.
In the extreme case, when the front-end found difficulties to decide about certain loop closure it will assign them all the same value of Ξ_{ij}. To show the influence of the Ξ_{ij}, we assume $\Xi_{ij} = \zeta$ to all constraints.

1) Methodology: To explain the influence of ζ, we study the variance of the relative pose errors RPE_{pos} and RPE_{ori} for different values of random outliers (1, 10, and 100) and ζ on the Manhattan dataset. Figure illustrates the results we have: Every data point represents the mean RPE_{pos} of 10 trials for a particular pairing of ζ and number of outliers.



2) Results and Interpretation: The plot in confirms that the value of $\Xi_{ij} = \zeta$ indeed influences the quality of the optimization result. We can see clearly that the quality of the estimation drops drastically if ζ is too large or small ($0.3 < \zeta > 2.0$). The most important result is that the RPE stays relatively constant for values of ζ between 0.3 and 1.5 where the error is minimal. This result is independent from the number of outliers.

- Result: If the front-end found difficulties to assign sound values to Ξ_{ij}, it is safe to set $\Xi_{ij} = 1$ since this value is close to the individual optimal choice independently of the number of outliers.

B. The Robustness in the presence of Outliers

After choosing a sound value for Ξ_{ij}, we now examine how well the back-end performs in the presence of outliers. The evaluation will compare the performance of the proposed robust back-end against those proposed in the state of the art in presence of outliers. We will furthermore see how much number of outliers can influence the estimation result.

1) Methodology: As we mentioned before, large number of datasets were considered during the evaluation. We tunned the number of added outliers (between 0 and 1000) using the four policies described above. For each number of additional wrong outliers, 10 trials per policy were calculated, resulting in a total of 500 trials per dataset. For each trial, the error metrics RPE and $RMSE$ were calculated. Notice from table that the number of correct loop closure constraints in the datasets varied between 10688 (City10000) and only 894 (Intel). Therefore, adding extra 1000 loop closures will increase the loop closure ratio (112% for the Intel dataset). While in real life applications we expect much less loop closure ratio, we will keep this high ratio in our experiments to see how robust the designed system is.

2) Results and Interpretation: Table II summarizes the results we got. Different measurements are mentioned here as : The minimum, maximum and median RPE_{pos}, as well as a success rate which measures the percentage of correct solutions. From table . we can see also that except for the parking garage dataset, the overall success rate are very high. In total, from all 2500 trials, only two failed, leading to a success rates $\simeq$ 100%. We mention here that the two failure cases (The Manhattan and the Sphere world datasets) would be successfully implemented if the Huber function was used in combination with the implemented robust back-end.
The Parking garage dataset is the exception from the successful implementation. We will discuss in details the reasons for this unexpected exception.

- Result: The proposed back-end was able to solve 2498 out of 2500 trials on different datasets with up to 1000 outlier constraints. In combination with the Huber Cost function, all the 2500 trials were successfully solved leading to a success rate of 100%.

TABLE II
OVERALL RPE_{pos} METRIC FOR THE DIFFERENT DATASETS, WITH...1000 OUTLIERS AND 500 TRIALS PER DATASET.

Dataset	max outl. ratio	min RPE_{pos}	max RPE_{pos}	median RPE_{pos}	incorrect solutions	success rate
Manhattan (g^2o)	47.6%	0.0009	0.0009	0.0009	0	100%
Manhattan (orig.)	47.6%	0.0009	5.9659	0.0009	1	99.8%
City 10000	9.4%	0.0005	0.0005	0.0005	0	100%
Sphere2500	40.8%	0.0953	0.0953	0.0964	1	99.8%
Intel	111.9%	0.2122	0.2122	0.2132	0	100%

We had two quantitative performance metrics to evaluate the robustness of the proposed method :

- RPE
- precision recall

While RPE compares the deviation of the estimated trajectory from the ground truth, the precision-recall allow us to determine how well the back-end performs i.e. how well it can identify, eliminate false loop

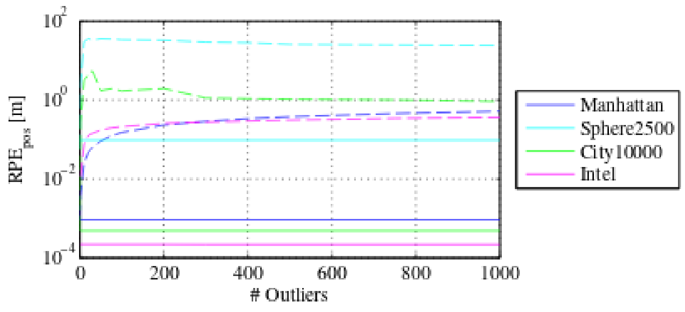


Fig. 4. Comparison of RPE measures between the proposed robust (solid line) and the state of the art non-robust back-ends (dashed). Notice how the robust solution is up to two orders of magnitude more accurate for large numbers of outliers and stays constant, independently of the amount of outliers.
Source: [5]

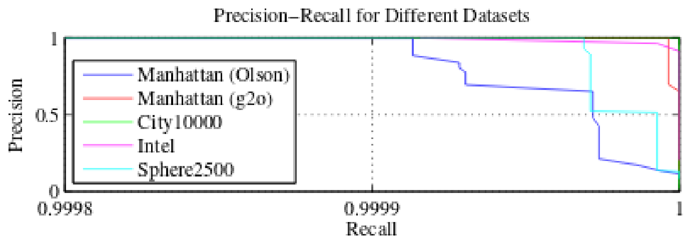


Fig. 5. Precision-recall statistics for the various datasets. Notice the scale of the X-axis (recall). The results indicate a close to optimal performance of the proposed system.
Source: [5]

closures and keep the true ones intact.
Optimality is reached when the system have precision-recall equal to 1, by deactivating all false positives and keeping true positives ones untouched. However, we mention that the back-end never made a binary decision on whether a constraint is supposed to be active or deactivated. It is only the precision-recall benchmark, that emulates such a behaviour.
From (figure4) we can deduce that the back-end reached optimal performance for the Intel, City10000, and Manhattan(g^2o) datasets, the recall is exactly 1 for a large span of precision. For sphere2500 and Olsons version of the Manhattan world datasets, the recall is slightly smaller, due to the two failure cases.

- Result: The proposed back-end reaches almost optimal results in terms of precision-recall. This confirms that all false positives are identified and eliminated while almost all true positive closure constraint are left intact.

C. Convergence and Runtime Behaviour

We expect the convergence time to increase with the number of the added outliers to the datasets.

1) Methodology: The datasets used in previous section were spoiled by 0 to 1000 outliers using all four policies. The time of convergence was used on a Core2-Duo desktop machine running at 2.4 Ghz. To inspect the convergence behaviour, 10 trials with 1000 outliers were selected from each dataset and each outlier policy. We define χ^2 to be the error measure the optimizer tries to minimize. Both χ^2 and time of convergence were recorded and normalized during the optimization so that their final value tend to a value of 1. During this investigation, the final values we got were averaged over the 10 trials belonging to the same dataset and same outlier policy.

2) Interpretation: The results we had are very different and rely too much to the policy used to add the outliers to the datasets. For the two non-local policies, the convergence time increases with the number of outliers while for the local policies, the convergence time increases slower. Obviously the non-local outlier constraints that often connect two very distant places in the dataset require more time to be resolved.

- Result: The structure of the dataset, number of outliers and the applied policy are parameters on which the convergence time depend heavily. Connecting local poses of the robot result in faster convergence than constraints connecting distant places on the dataset.

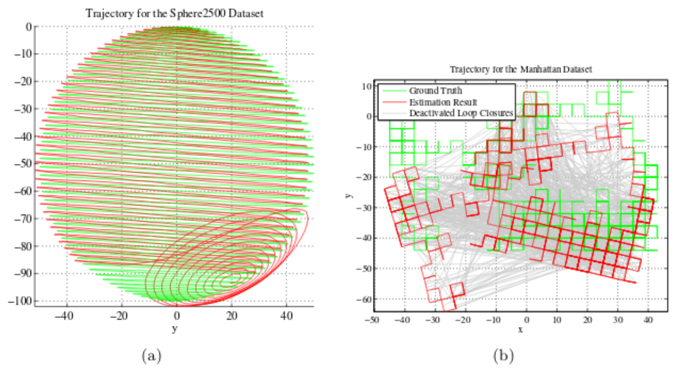


Fig. 6. Two failure cases for the Sphere World dataset in (a) and the Manhattan dataset (Olsons original). Despite the failure cases, the maps are still locally consistent. Adding a Huber cost function to the robust back-end can resolve both cases.
Source: [5]

D. Discussion of the Failure Cases

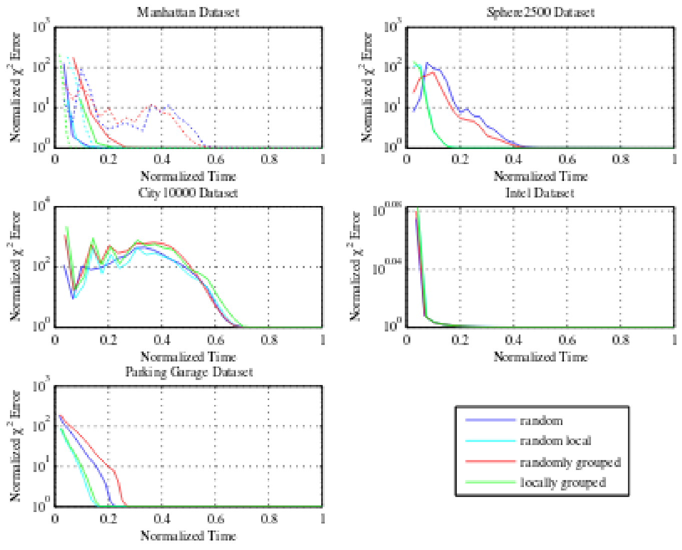


Fig. 7. Convergence time for different outlier policies for the different datasets. Notice that the two local policies require much less time for convergence than the non-local policies.
Source: [5]

1) Manhattan and Sphere Datasets: Two trials in the Sphere2500 and the Olsons version of the Manhattan datasets failed to converge to a correct solution (see Figure 5). Although the resulting maps are significantly distorted when compared to the ground truth on a global level, they are still locally intact. For the sphere world dataset, the resulting map consists of two individually consistent and intact sub-maps that are however misaligned to each other. For the Manhattan dataset, a false positive loop closure constraint was not detected in the center of the map which is the reason of the failure.

- Result: Even in case of failures, the proposed robust back-end degrades gracefully. Apparently not deactivating single false positives correctly leads to punctual errors that cause global distortion but retain local consistency.

2) The garage Dataset: Here, the robust back-end approach didn't perform better performance than the non-robust approach.
Figure.7 shows the ground truth trajectory and illustrates the results we had using the robust back-end technique. Here we scaled the z-axis so the data could be seen in the context of its spatial structure.
The dataset contains four parking decks connected by only two strands of odometry constraints that originate from the driveways. The problem arising from this sparse connection structure can be seen in Fig.7(b). The insufficient amount of information on the relative pose of the individual decks leads to a small number of constraints (compared to previous datasets) which as a consequence leads to a failure. In simple words, we don't have enough odometry constraints that could vote against a false loop closure request.
Since SLAM systems has no knowledge about the structure of the environment, the proposed back-end fails to eliminate false positive loop closures.

- Results: The robust back-end solution fails to eliminate false positive loops in environments composed by sparsely connected parts.

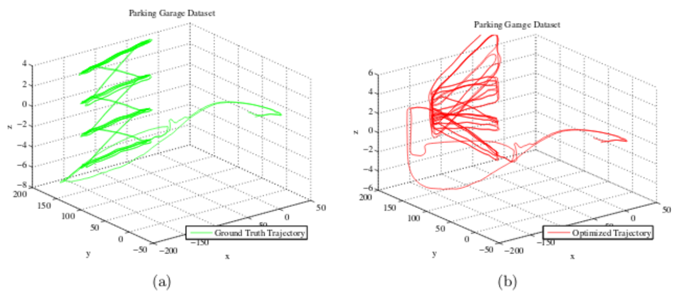


Fig. 8. (a) shows the ground truth trajectory from the side, (b) illustrates a failure exemplary produced by the robust back-end approach.
Source: [5]

VI. Conclusion

In this seminar paper, we proposed and evaluated a method to identify and reject outliers in the back-end of a SLAM system by using switchable constraints. The feasibility of the proposed approach has been demonstrated in a variety of 2D and 3D datasets that contain false positive loop closure constraints. The proposed extension to the back-end part correctly identifies and disables false positive loop closure constraints, while maintaining the correct (true positive) constraints. Even when it fails, the system degrades gracefully and the resulting map retains its local consistency.
We could increase the robustness of the proposed system by combining it with other measures such as the Huber cost function that decreases the influence of outliers.
Also choosing a suitable front-end to the overall SLAM system increases the robustness against errors in the place recognition.
The proposed approach is extensible and can be also applied to other domains where least squares problems have to be solved like multi-path mitigation in GNSS-based localization . [5]

References

[1] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3281–3288. IEEE, 2011.

[2] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[3] Computer Science and System Engineering Laboratory. Loop Closure Detection, 2004.

[4] Niko Sünderhauf. *Robust optimization for simultaneous localization and mapping*. PhD thesis, Technischen Universitat Chemnitz, 2012.

[5] Niko Sünderhauf and Peter Protzel. Switchable constraints for robust pose graph slam. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1879–1884. IEEE, 2012.