Cluster-based Penalty Scaling

Fang Wu

December 12, 2019

1 Introduction

This document provides details on the research work conducted during the Engage project. The objective of the project is to develop a pose graph optimization algorithm that is resilient to spurious loop closure detections. This document starts with providing prerequisites on the specific problem (Section 2). Section 3 discusses the related works on handling outliers and Section 4 details the methodology of our method. Experimental results on various datasets are presented in Section 5 and conclusions as well as future works are provided in Section 6.

2 Prerequisites

2.1 Factor graph

Factor graph as shown in Fig.1 is commonly used to illustrate the interdependent relations in state estimation problem. Within the context of pose graph SLAM, each node represents the robot pose at a distinctive moment in time (please note that duration of time or travel distance between sequential poses is not a fixed value). Odometry measurements between sequential poses can be expressed as the binary factors between consecutive nodes. Loop-closing measurements between non-sequential poses are represented by the binary factors between nonconsecutive nodes. Measurements from GPS modules and observations of landmarks can be represented as the unary factor that depend on only one node. From this insightful visualization, we can see clearly the sparsity of the problem, which enables the use of fast solvers in modern SLAM application.

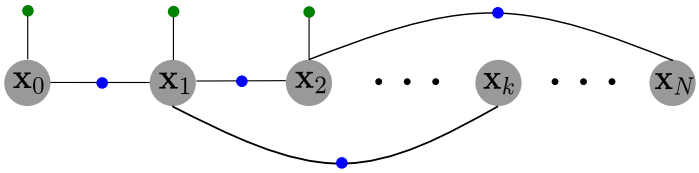


Figure 1: An illustration of a typical factor graph: two types of factors are present in this graph, blue dots representing the binary factors while green dots representing the unary factors

2.2 General formulation(from MAP to NLS)

This section will introduce some fundamental concepts in estimation theory, which is widely used in the field of robotics. State estimation problems can be solved in batch or incrementally, which correspond to offline and online applications respectively. Batch solving compute the solution at once after gathering all the measurements and variables while incremental solving involves adding variables as well as measurements in succession. In this session, we deal with batch solving only and a general problem setup is described as below.

Problem setup

We define the following motion and observation models:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k), \quad k = 1, \cdots, K \tag{1}$$

$$\mathbf{y}_i = \mathbf{g}(\mathbf{x}, \mathbf{n}_i), \quad i = 1, \cdots, I \tag{2}$$

where k is the discrete-time index for the states and i is the index of measurements. We note that the quantity of measurements I are usually different with the quantity of states K and one measurement could depend on multiple states. Other variables have the following meanings:

$$\begin{aligned}
\text{system state} &: \mathbf{x}_k \in \mathbb{R}^N \\
\text{initial state} &: \mathbf{x}_0 \in \mathbb{R}^N \sim \mathcal{N}(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0) \\
\text{input} &: \mathbf{v}_k \in \mathbb{R}^N \\
\text{process noise} &: \mathbf{w}_k \in \mathbb{R}^N \sim \mathcal{N}(0, \mathbf{Q}_k) \\
\text{measurement} &: \mathbf{y}_i \in \mathbb{R}^M \\
\text{measurement noise} &: \mathbf{n}_i \in \mathbb{R}^M \sim \mathcal{N}(0, \mathbf{R}_i)
\end{aligned}$$

We use $\check{(\cdot)}$ to indicate prior estimation (before considering measurements) and $\hat{(\cdot)}$ to indicate posterior estimation (after considering measurements).

Given the problem setup, our objective in batch state estimation is to solve a Maximum A Posteriori (MAP) problem

$$\hat{\mathbf{x}} = arg\max_x p(\mathbf{x}|\mathbf{v}, \mathbf{y}) \tag{3}$$

which says to find the best estimate for the state of the system at all timesteps, $\hat{\mathbf{x}}$, given the prior information, $\mathbf{v}$, and the measurements $\mathbf{y}$. Note that the prior information $\mathbf{v}$ here includes the initial state $\mathbf{x}_0$ and the inputs $\mathbf{v}_k$.

After substituting the Gaussian density functions into equation (3) and conduct some magical derivations using Bayes' rule (details can be found in 3.1.2 of [1]), we arrive at a nonlinear least square(NLS) optimization problem

$$\hat{\mathbf{x}} = arg\min_x \sum_{n=1}^{N} (\mathbf{E}_{v,k}(\mathbf{x}) + \mathbf{E}_{y,i}(\mathbf{x})) \tag{4}$$

where $\mathbf{E}_{v,k}(\mathbf{x})$ and $\mathbf{E}_{y,i}(\mathbf{x})$ refers to the individual cost terms from motion model and observation model. These cost terms take the form of a squared Mahalanobis distance. In classical formulation, N would be the number of actual measurements due to motion and observation models, $K + I$, according to equation (1) and (2). But it is common to add other terms that could affect the solution for the best estimate (e.g., penalty terms). We will talk about this more in section 3.1. Within this session, we only deal with cost terms from measurements.

From the problem setup, we can write the individual residual errors to be

$$\mathbf{e}_{v,k}(\mathbf{x}) = \left\{ \begin{array}{ll} \check{\mathbf{x}}_0 - \mathbf{x}_0, & k = 0 \\ \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, 0) - \mathbf{x}_k, & k = 1, \cdots, K \end{array} \right\} \tag{5}$$

$$\mathbf{e}_{y,i}(\mathbf{x}) = \mathbf{y}_i - \mathbf{g}(\mathbf{x}_i, 0), \qquad i = 1, \cdots, I \tag{6}$$

so that the individual cost terms can be expressed as

$$\mathbf{E}_{v,k}(\mathbf{x}) = \mathbf{e}_{v,k}^T(\mathbf{x}) \mathbf{W}_{v,k}^{-1} \mathbf{e}_{v,k}(\mathbf{x}) \tag{7}$$

$$\mathbf{E}_{y,i}(\mathbf{x}) = \mathbf{e}_{y,i}^T(\mathbf{x}) \mathbf{W}_{y,i}^{-1} \mathbf{e}_{y,i}(\mathbf{x}) \tag{8}$$

where $\mathbf{W}_{v,k}^{-1}$ and $\mathbf{W}_{y,i}^{-1}$ can be seen as weights that scale the residual errors according to the process noise and measurement noise. To further clarify the relation between objective function and the measurements as well as their uncertainties, the objective function in equation (4) can be written as matrix form below *

*The inverse of a block diagonal matrix is another block diagonal matrix, composed of the inverse of each block. This is why the inverse sign could be taken out of the square bracket.

$$\sum_{n=1}^{N}\left(\mathbf{E}_{v,k}(\mathbf{x})+\mathbf{E}_{y,i}(\mathbf{x})\right)=\begin{bmatrix}\mathbf{e}_v(\mathbf{x})\\ \mathbf{e}_y(\mathbf{x})\end{bmatrix}^T\begin{bmatrix}\mathbf{W}_v & \\ & \mathbf{W}_y\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{e}_v(\mathbf{x})\\ \mathbf{e}_y(\mathbf{x})\end{bmatrix}$$
$$=\begin{bmatrix}\mathbf{e}_v(\mathbf{x})\\ \mathbf{e}_y(\mathbf{x})\end{bmatrix}^T\begin{bmatrix}\mathbf{P}_0 & & & & & & & \\ & \mathbf{Q}_1 & & & & & & \\ & & \ddots & & & & & \\ & & & \mathbf{Q}_K & & & & \\ & & & & \mathbf{R}_1 & & & \\ & & & & & \ddots & & \\ & & & & & & \mathbf{R}_I\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{e}_v(\mathbf{x})\\ \mathbf{e}_y(\mathbf{x})\end{bmatrix} \tag{9}$$

Our goal is to estimate the system state $\mathbf{x}$, which is not exposed in equation (9). Only when the motion model and observation model are both linear, which means

$$\mathbf{x}_k=\mathbf{A}_{k-1}\mathbf{x}_{k-1}+\mathbf{v}_k+\mathbf{w}_k,\quad k=1,\cdots,K \tag{10}$$

$$\mathbf{y}_i=\mathbf{C}_i\mathbf{x}_i+\mathbf{n}_i,\quad i=0,\cdots,I \tag{11}$$

the objective function can be further reformulated as below to isolate the system state

$$\left(\mathbf{z}-\mathbf{H}\begin{bmatrix}\mathbf{x}_0\\ \vdots\\ \mathbf{x}_K\end{bmatrix}\right)^T\quad\mathbf{W}^{-1}\quad\left(\mathbf{z}-\mathbf{H}\begin{bmatrix}\mathbf{x}_0\\ \vdots\\ \mathbf{x}_K\end{bmatrix}\right) \tag{12}$$

where

$$\mathbf{z}=\begin{bmatrix}\check{\mathbf{x}}_0\\ \mathbf{v}_1\\ \vdots\\ \mathbf{v}_K\\ \mathbf{y}_0\\ \vdots\\ \mathbf{y}_I\end{bmatrix},\mathbf{H}=\begin{bmatrix}1 & & & \\ -\mathbf{A}_0 & 1 & & \\ & \ddots & \ddots & \\ & & -\mathbf{A}_{K-1} & 1\\ \mathbf{C}_0 & & & \\ & \mathbf{C}_1 & & \\ & & \ddots & \\ & & & \mathbf{C}_I\end{bmatrix},\mathbf{W}=\begin{bmatrix}\mathbf{P}_0 & & & & & & \\ & \mathbf{Q}_1 & & & & & \\ & & \ddots & & & & \\ & & & \mathbf{Q}_K & & & \\ & & & & \mathbf{R}_1 & & \\ & & & & & \ddots & \\ & & & & & & \mathbf{R}_I\end{bmatrix} \tag{13}$$

We can observe from equation (12) that $\mathbf{x}$ is the only unknown variable and the objective function is exactly a paraboloid, thus its minimum can be found in closed form. By setting the partial derivative with respect to $\mathbf{x}$ to zero, the solution can be computed as below:

$$\frac{\partial\mathbf{E}(\mathbf{x})}{\partial\mathbf{x}^T}=-\mathbf{H}^T\mathbf{W}^{-1}(\mathbf{z}-\mathbf{H}\mathbf{x})=0 \tag{14}$$

$$\Rightarrow(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H})\hat{\mathbf{x}}=\mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{15}$$

In reality, we would never actually invert $\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}$. Due to its special sparse structure, sparse solvers can be used to solve this equation efficiently. The most common method is to conduct cholesky decomposition on the symmetric positive-definite matrix $\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}$, and obtain a lower triangular matrix $\mathbf{R}$ so that

$$\mathbf{R}\mathbf{R}^T\hat{\mathbf{x}}=\mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{16}$$

Back substitution and forward substitution can then solve this equation efficiently. (More details can be found in 3.2.1-2 of [1])

When the motion model and observation model are nonlinear, which is the case for SLAM problem, we cannot obtain a closed-form solution to the MAP problem any more. However, since $\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}$ is symmetric positive-definite and cholesky decomposition of $\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}$ can be conducted, it is obvious that the objective function has a quadratic form. Gauss-Newton optimization techniques can be effectively used to find an approximate solution.

2.3 Linearization

We do not go into details regarding Gauss-Newton method here. Related information can be found in 4.3 of [1]. The key idea is that starting from an operation point, we can linearize the original problem, then compute the updates from the linearized equation that has the form of

$$(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H})\delta\mathbf{x} = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{e}(\mathbf{x}_{op}) \tag{17}$$

where $\mathbf{H}$ consists of the Jacobians of the nonlinear motion and observations model, $\mathbf{W}$ consists of the covariance matrix of initial state, process noise and measurement noise, as in equation (13), $\mathbf{e}(\mathbf{x}_{op})$ refers to the residual error at the operation point. We note that this equation has very similar structure with equation (15) and similar solving techniques for sparse systems can be used here. After obtaining the updates, actual updates are performed on the operation point and then relinearize about the mean of the best estimation. This procedure is iterated until the update step becomes sufficiently small.

Every iteration of the nonlinear solver involves building a new linear system using the current linearization point, calculating its factorization and solving the systems. Calculating the factorization is typically the most expensive step, which is particularly challenging in online applications, where the state changes every step. For large problem, updating and solving the nonlinear system at every step can become very expensive. This is usually handled by changing the linearization point less frequently so that factorization is not needed at every step. New variables can still be added to the factorization and although the Jacobian matrix does not correspond to the new state, Gauss-Newton method would still reduce the error unless close to an abrupt change in the derivatives.

It would seem that the solution is to incrementally update the linear system in the already factorized form and perform back subtitution whenever an solution is needed. However, simply updating the factorization with new variables would introduce another problem - fill-in, which refers to entries of a matrix that change from an initial zero to a non-zero value during the execution of an algorithm. SLAM problem can only be solved efficiently due to the sparsity of the system matrix, therefore fill-reducing ordering (by switching rows and columns in the matrix) is needed. Fill-in happens especially when loop-closures are added to the system which introduce far off-diagonal entries in the matrix form.

2.4 Covariance Recovery

As explained above, due to the specific sparse structure of $\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}$, which is a block tridiagonal matrix*, sparse solver can be used to efficiently obtain the most likely estimate of $\mathbf{x}$. Another important question to ask, is how certain we are in this solution. Interestingly, we can re-interpret the least-squares solution as a Gaussian estimate for $\mathbf{x}$ by employing Bayes' rule. The detailed derivation can be found in 3.1.5 of [1]. The conclusion is that $\mathcal{N}(\hat{\mathbf{x}}, \hat{\mathbf{P}})$ is a Gaussian estimator for $\mathbf{x}$ whose mean is the optimization solution and covariance is $\hat{\mathbf{P}} = (\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H})^{-1}$.

Inverting a sparse matrix is a computationally expensive operation, therefore recursive formulas [2] have been proposed to recover specific elements of the marginal covariance matrix from the factored matrix $\mathbf{R}$. Out of the commonly used optimizers for SLAM applications, isam, g2o and SLAM++ [3] implemented the recursive formula to recover covariance matrix. This enables efficient recovery of covariance of multiple variables at once. Latest vertion of gtsam (with isam2) is only efficient in computing one covariance of a single variable since it uses Bayes tree as the underlying data structure. SLAM++[4] is the state-of-art optimizer for efficient marginal covariance recovery.

Marginal covariance has been widely used for data association, active decisions and next best view. To the best of our knowledge, this work is the first attempt at utilizing marginal covariance matrix for loop closure clustering.

3 Related works

3.1 Robust pose graph optimization

Before diving into the robust estimation problem, we provide a brief overview of the common pose graph SLAM pipeline in Fig.2 to better explain the source of outliers. The architecture of a typical SLAM system can be divided into two components: the *front-end* and the *back-end*. The *front-end* extracts geometrical information from raw measurements of various sensor modules (camera

*This tridiagonal structure is due to the special sparsity of the upper half of $\mathbf{H}$, which comes from the fact that the system model obeys the Markov property. More details can be found in 3.1.3 of [1]

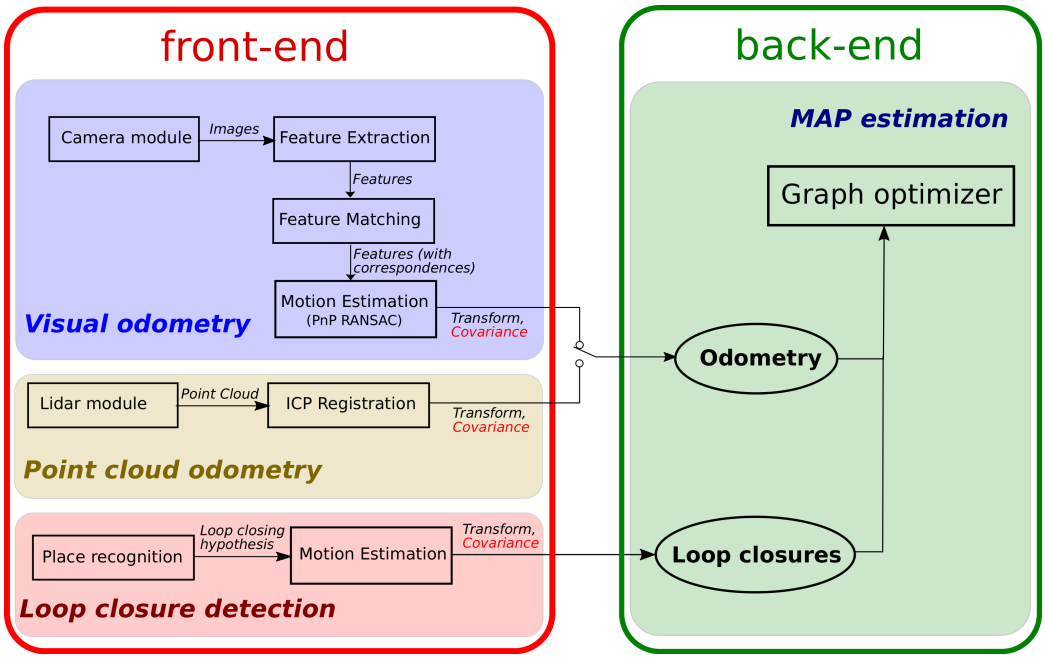


Figure 2: A typical pipeline for mapping

module and lidar module are shown in Fig.2 while many more exist) and the *back-end* performs inference based on the provided information. Within the *back-end*, which is usually formulated as an MAP estimation problem detailed in previous section, information from the *front-end* are used as cost terms within the optimization problem. They typically consist of odometry edges, which link consequential nodes on the factor graph, and loop closure edges, which link nonconsequential nodes on the factor graph. Naturally, long-term data association associated with loop closure edges has to deal with higher noise, therefore more prone to mistakes. To make things worse, a wrong loop closure edge would do much more damage to the map compared to a wrong odometry edge.

We now pay more detailed attention to the loop closure detection module. Loop closing hypotheses are first provided by the place recognition module. Thresholds would be set so that if a loop closing hypothesis exceeds the threshold, it would be passed on for motion estimation. RANSAC is commonly used to compute the transform of the loop closures in order to mitigate the effect of spurious measurements. If a transform can be successfully computed, a loop closure edge is generated with transform and covariance matrix, which reflect the uncertainty of motion estimation. After receiving the loop closure edges from the loop closure detection module and odometry edges from the odometry module, MAP estimation is conducted by a graph optimizer. Current libraries (e.g., g2o, gtsam, toro, ceres, SLAM++) are able to solve the problems efficiently. The classical NLS formulation for this specific MAP estimation problem is shown below

$$\hat{\mathbf{x}} = arg\min_{x} \sum \mathbf{E}(\mathbf{x}) = arg\min_{x} \left(\sum_{k=1}^{K} \mathbf{e}_k^T(\mathbf{x}) \mathbf{W}_k^{-1} \mathbf{e}_k(\mathbf{x}) \; + \; \sum_{i=1}^{I} \mathbf{e}_i^T(\mathbf{x}) \mathbf{W}_i^{-1} \mathbf{e}_i(\mathbf{x}) \right) \tag{18}$$

where $\mathbf{e}_k$ refers to the residual error of odometry edges and $\mathbf{e}_i$ refers to the residual error of loop closure edges. $\mathbf{W}_k$ refers to the covariance matrix of the kth odometry measurement while $\mathbf{W}_i$ the covariance matrix of the ith loop closure measurement. It should become clear now that even with one wrong loop closure, named as outliers, the solution of the MAP estimation problem will change dramatically. It could easily contaminate the whole pose graph, and the resultant map would be severely distorted.

There are different scenarios that outliers could appear. For example, a low loop closing threshold would cause false positives from the place recognition module. Then two images (or scans) that are largely different from each other would be passed on to the motion estimation module. This usually generates outliers with high uncertainty, since the motion estimation module would have a hard time finding consistent transforms for two largely different images (scans). This type of outliers are usually not that bad since the covariance matrix serves as weights to scale the residual error. The cost term would already be scaled down so that optimizer would not exactly close the loop. A much more difficult type of outliers are called perceptual aliasing, which happens when two different

places actually appear the same to the sensor modules. Since the loop closure detection module has no information about the global graph structure when computing the covariance, it depends only on the similarity of the two images (or scans). This type of outliers would have low covariance matrix so that optimizer would be encouraged to actually close the loop. The problem is even more exasperated by the fact that perceptual aliasing tend to create a large number of mutually-consistent outliers, which further increase the incentives for optimizer to close the loops.

Numerous studies have been done to achieve robust pose graph optimization, which either preprocess the pose graph by detecting and then removing the outliers [5], or mitigate the effect of outliers by revising the objective function of NLS problem [6][7]. In [6], additional variables are introduced to allow the loop closures that introduce large residual error to be switched off during the optimization.

$$\hat{\mathbf{x}} = arg\min_{x} \left(\sum_{k=1}^{K} \mathbf{e}_k^T(\mathbf{x}) \mathbf{W}_k^{-1} \mathbf{e}_k(\mathbf{x}) \; + \; \sum_{i=1}^{I} s_i^2 \mathbf{e}_i^T(\mathbf{x}) \mathbf{W}_i^{-1} \mathbf{e}_i(\mathbf{x}) \; + \; \sum_{i=1}^{I} (\lambda_i - s_i)^2 \sigma_i^{-1} \right) \tag{19}$$

Additional variable s_i scales the cost term for the ith loop closure measurement. And the third term $(\lambda_i - s_i)^2 \sigma_i^{-1}$ works as a penalty when the additional variable s_i varies from its initial value λ_i. Comparing equation (18) and (19), it should be clear that by introducing this additional variable s, the optimizer is provided with another gradient direction to minimize the objective function instead of closing the loop. And which direction the optimizer would move in the solution space depends on the balance between $\mathbf{e}_i^T(\mathbf{x}) \mathbf{W}_i^{-1} \mathbf{e}_i(\mathbf{x})$ and σ_i^{-1}. Agarwal [7] has leveraged this insight and proposed a revised form of equation (19) to explicitly scale the cost term according to the residual error. Additional variables s_i was eliminated by simply scaling the covariance matrix, but covariance σ_i in the penalty term is still needed to compute the scaling parameter. Intuitively, σ_i should reflect the threshold on admissible residual error. However, as we are using an iterative optimizer due to the nonlinearity of pose graph estimation problem, the scaling variables s_i (or scaled covariance matrix) are being reevaluated at each iteration based on the state from the previous iteration (this falls into the category of *M-estimation* or *iteratively reweighted least squares* and more details can be found in 5.3.1 of [1]). This apparently works in a more indirect way compared to a simplified clean-cut scenario that any terms exceeding the admissible residual error would be immediately switched off. There is no mathematically rigorous way to set the value of σ_i and we will see more on its ambiguity later in the experimental results.

The second line of work [5] exploits the probabilistic interpretation of the objective function, which is a χ^2 distribution under Gaussian noise assumption, to detect outliers. Following the assumption that an outlier should cause larger residual error, Latif [5] conducted χ^2 test on spatially clustered loop closures and aimed at finding a set of consistent measurements. From our observations, when the spatial clustering algorithm produces a heterogeneous cluster of loop closures, *rrr* [5] tends to be too conservative since it would reject the whole cluster upon failing the χ^2 test. Also, since the algorithm solves the pose graph optimization problem after including the measurements (which could be an inlier or outlier), the resultant χ^2 error could pass the χ^2 test due to high quality measurements compensating poor quality measurements.

Another line of work originated from a slightly different perspective. As mentioned above, iterative nonlinear optimization is commonly used to solve the SLAM problem and can only guarantee local convergence. Its performance heavily depends on the initial estimation. Many researchers have attempted to avoid convergence to local minimal by performing convex relaxation on the original formulation. Regarding outlier rejection, Lajoie [8] proposed a novel formulation to model perceptual aliasing and developed a semidefinite relaxation to remove the nonconvexity. The formulation differs from Sunderhauf's [6] by introducing a correlation term between subsets of the binary variables which penalize a mismatch between variables from one consistent cluster. While showing promising results with the convex optimizer, the authors acknowledged that a heterogeneous cluster of loop closures (contains both inlier and outlier) would cause the algorithm to perform poorly. Strategies on clustering are not explored. It also pointed out the importance of setting the threshold on maximum admissible residual error.

3.2 Incremental outlier rejection

Works discussed in the previous section focused on batch outlier rejection problem, which means a complete graph with all the nodes and edges is known when trying to decide the nature of a loop closure edge. This is beneficial because it makes use of all the information for estimating every state. However, this cannot be used online because of the obvious reason that future data would not be

available at the time of the estimation. Incremental outlier rejection techniques such as [5] and [9] perform different evaluation techniques after adding new measurements into the system. Both studies utilized χ^2 test on a subset of the pose graph after solving the problem. Bai [10] proposed a metric to predict the objective function change in an incremental pose graph optimization scheme without actually solving the problem. Instead of checking for inconsistency after adding the measurement and solving the problem, this technique predicts the quality of specific measurment before adding it into the system.

In [10], estimation problem before adding a loop closure edge is considered as primal NLS problem and estimation after adding a loop closure edge an augmented NLS problem. It showed that the linearized primal NLS problem has the same solution $\mathbf{x}^\star$ as well as objective function value $f(\mathbf{x}^\star)$ with the original primal NLS problem. Same thing can be said about the augmented NLS problem. Without solving for the solution $\mathbf{x}^{\star\star}$ of augmentated NLS problem, the objective function change of adding the loop closure i can be computed as below,

$$\Delta f = \mathbf{e}_i^T(\mathbf{x}^\star) \cdot [\mathbf{J}_i(\mathbf{x}^\star)\mathbf{M}_i(\mathbf{x}^\star)\mathbf{J}_i^T(\mathbf{x}^\star) + \mathbf{W}_i]^{-1} \cdot \mathbf{e}_i(\mathbf{x}^\star) \tag{20}$$

where $\mathbf{e}_i^T(\mathbf{x}^\star)$ refers to the residual error for loop closure i at the state $\mathbf{x}^\star$; $\mathbf{J}_i(\mathbf{x}^\star)$ refers to the Jacobians of loop closure measurement with respect to individual poses; $\mathbf{M}_i(\mathbf{x}^\star)$ refers to the marginal covariance matrix of the two poses constructing loop closure i; $\mathbf{W}_i$ refers to the covariance matrix of loop closure edges i.

The authors claimed equation (20) is a good approximation of the objective function change under one condition: difference between the objective function value of linearized augment problem and linearized primal problem at the solution of original primal problem is much larger than the changes in objective function value of linearized augmented problem at the solution of original primal and the solution of the augmented problem. Although this condition is not yet mathematically proved, the authors showed empirically that this condition usually holds for an online incremental SLAM problem. Intuitively, equation (20) takes the form of a mahalanobis distance between the solution $\mathbf{x}^\star$ of the primal NLS problem (this is considered to be deterministic here since the primal NLS problem is solved) and a distribution of the possible solutions of augmented NLS problem. And $[\mathbf{J}_i(\mathbf{x}^\star)\mathbf{M}_i(\mathbf{x}^\star)\mathbf{J}_i^T(\mathbf{x}^\star) + \mathbf{W}_i]^{-1}$ contains the fisher information matrix after including loop closure i, accounting fo the covariance matrix of this distribution. Thus, equation (20) can be interpreted as scaling the error vector at the primal solution with a combined matrix of solution uncertainty and measurement uncertainty.

The authors conducted preliminary experiments to use the predicted objective function change for outlier detection. χ^2 difference test, a variation of χ^2 test employed on the difference of χ^2 error and the change of the degree of freedom, is used instead of χ^2 test. Despite showing some promising results, the authors acknowledged objective function change has its limitation on reflecting the quality of a measurement. When the optimization problem is strongly nonlinear, the solution could change dramatically while the objective function barely changes (more details can be seen in [11]). And with noisy odometry and densely connected pose graph, a naive use of this metric is not robust enough, which we will show later in experimental results.

Another limitation of this method is that the computation time of marginal covariance matrix grows with the number of poses. Despite using SLAM++ [4] which implements state-of-the-art strategies to compute the marginal covariance in real time, the solution time is not bounded.

4 Methodology

Here, we propose a cluster-based approach that leverage existing works [6][10][5][8]. By first generating intra-consistent clusters based on their local consistency and then model this local consistence in a cluster based robust formulation, we achieve a good balance between local consistence and global consistence.

4.1 Clustering

We start with the spatial clustering techniques proposed by Latif [5] to extract topologically related loop closures. The set of spatial clusters is defined as C in equation (21). Each cluster C_m is subset of LC, the set of loop closure edges, and contains the loop closures that link the same portions (index wise) of the trajectory. Then we conduct incremental consistency check for each cluster by comparing predicted objective function changes Δf and the threshold $\chi^2_\alpha(DOF)$, which depends on α, the confidence probability that usually set to 0.95, and the degree of freedom(DOF) that

depends on the dimension of measurements. Algorithm 1 describes the detailed strategies. We note the difference between our method and [10]: individual edges are evaluated by themselves in [10] and the quality of a specific edge is determined from that edge's consistency with the odometry edges and all the previously accepted loop closures; on the contrary, our method examines a spatially coherent cluster, and aims to exploit the local consistency within the cluster. Symbols used in Algorithm 1 and 2 are explained as below.

Definitions of Symbols

$$
\begin{aligned}
\text{Edge between two poses} &: E = \{from,\ to\} \\
\text{Ordered set of edges} &: L = \{E_k \mid E_k = \{from_k,\ to_k\},\ k = 1 \cdots K\} \\
\text{Set of loop closure edges} &: LC = \{E_i \mid E_i = \{from_i,\ to_i\},\ i = 1 \cdots I\} \\
\text{Set of odometry edges} &: OD = \{E_j \mid E_j = \{from_j,\ to_j\},\ j = 1 \cdots J\} \\
\text{Set of spatial clusters} &: C = \{C_m \mid C_m \subseteq LC,\ m = 1 \cdots M\}
\end{aligned} \tag{21}
$$

I represents the quantity of loop closures while J the quantity of odometry edges. K represents the quantity of all edges, therefore $K = I + J$. L is an ordered set and sorted by to index. M represents the quantity of spatial clusters.

Algorithm 1: outlier detection against the odometry edges

Input: A spatial cluster C_m
Output: A intra-consistent cluster with its quality indicator

```
for k ← 1 to K do
    if E_k ∈ OD then
        Add E_k into Optimizer
    else
        if E_k ∈ C_m then
            Optimize and Compute Δf
            if Δf ≤ χ²_0.95(DOF) then
                inlierSet ← {inlierSet, E_k}
                Add E_k into Optimizer
            else
                outlierSet ← {outlierSet, E_k}
if inlierSet.size() > outlierSet.size() then
    quality = 1
    return inlierSet, quality
else
    return Algorithm 2(outlierSet)
```

As shown in algorithm 1, we first incrementally examining all the loop closures within one spatial cluster against the odometry backbone. If the loop closure passes the χ^2 difference test, indicating it is consistent with the odometry edges and the loop closure edges that have come before this time step, it is considered as an inlier and added into the optimizer; otherwise, it is rejected and stored as an outlier. After processing all the edges within this cluster, we examine how many inliers and outliers are within this cluster. If there are more inliers than outliers, the inliers are returned with a binary quality indicator, value set to 1. If there are more outliers than inliers, algorithm 2 is executed with the outliers. The purpose of this second test is to prevent from rejecting all the loop closures that are inconsistent with odometry edges. Under a scenario that this specific cluster attempts to close long loops without information on the smaller loops that support the same inference, the second test would save it from being rejected immediately. Algorithm 2 has similar logic with algorithm 1 except that the first loop closure is always added into the optimizer without consistency check. This would change the system state assuming the first loop closure were an inlier, and the following consistency check is conducted upon this assumption. After processing all edges, we again check the inlier and outlier quantity within this specific set. If there are more inliers than outliers, the inliers are returned with a quality indicator of value 0, which means although intra-consistent, these edges

are not consistent with the odometry edges. Otherwise, the new outliers are processed again. By iteratively executing algorithm 2, we manage to isolate an intra-consistent cluster out of the outliers.

Algorithm 2: consistency check within *outlierSet*

Input: A set of loop closure edges C_m
Output: A intra-consistent cluster with its quality indicator

```
for k ← 1 to K do
    if E_k ∈ OD then
        Add E_k into Optimizer
    else
        if E_k ∈ C_m then
            if no loop closure in Optimizer then
                inlierSet ← {inlierSet, E_k}
                Add E_k into Optimizer
            else
                Optimize and Compute Δf
                if Δf ≤ χ²_0.95(DOF) then
                    inlierSet ← {inlierSet, E_k}
                    Add E_k into Optimizer
                else
                    outlierSet ← {outlierSet, E_k}
if inlierSet.size() ≥ outlierSet.size() then
    quality = 0
    return inlierSet, quality
else
    return Algorithm 2(outlierSet)
```

The algorithms described in this section are implemented within SLAM++ framework, due to its state-of-art implementation in marginal covariance recovery and the program will produce an output file as below. Label "CLUSTER" refers to intra-consistent clusters and "CLUSTER_R" indicates all the edges that have been rejected after the clustering process.

Output text file template

CLUSTER $< from >$ $< to >$ $< quality >$
CLUSTER $< from >$ $< to >$ $< quality >$
$\vdots$ $\vdots$ $\vdots$ $\vdots$
CLUSTER $< from >$ $< to >$ $< quality >$

an empty line indicates the start of a new cluster

CLUSTER $< from >$ $< to >$ $< quality >$
CLUSTER $< from >$ $< to >$ $< quality >$
$\vdots$ $\vdots$ $\vdots$ $\vdots$
CLUSTER $< from >$ $< to >$ $< quality >$

an empty line indicates the start of a new cluster

CLUSTER_R $< from >$ $< to >$
$\vdots$ $\vdots$ $\vdots$ $\vdots$

Intuitively, our clustering techniques comb through the spatial clusters and untangle the inter-dependence between loop closures by extracting only the majority set after every consistency check. Compared to the intra-cluster consistency check techniques in [5], our method preserve more intra-consistent clusters due to the use of χ^2 difference test. In the next session, we will discuss how to model the clusters as well as their quality in a unified formulation.

4.2 Cluster-based Switchable Constraints

Inspired by [6] and [8], we propose a robust estimation formulation in the form below, where the objective function for minimization includes four types of cost terms: odometry edges, scaled loop closure edges, prior constraints for the scale s, and clustering constraints that bind the scales s of the edges belonging to similar cluster.

$$\begin{aligned}\hat{\mathbf{x}} = arg\min_{x}\Big(&\underbrace{\sum_{k=1}^{K}\mathbf{e}_k^T(\mathbf{x})\mathbf{W}_k^{-1}\mathbf{e}_k(\mathbf{x})}_{\text{Odometry}} \\ &+\underbrace{\sum_{i=1}^{I}s_i^2\mathbf{e}_i^T(\mathbf{x})\mathbf{W}_i^{-1}\mathbf{e}_i(\mathbf{x})}_{\text{Scaled Loop Closure}} + \underbrace{\sum_{i=1}^{I}(\lambda_i - s_i)^2\sigma_i^{-1}}_{\text{Prior}} \\ &+\underbrace{\sum_{m=1}^{M}\sum_{\substack{E_i,E_{i'}\in C_m \\ E_i\neq E_{i'}}}(s_i - s_{i'})^2\sigma_m^{-1}}_{\text{Clustering}}\Big)\end{aligned} \tag{22}$$

We can see that the first cost term in the optimization, which correspond to odometry edges, remains unchanged from the classical formulation. And the following scaled loop closure term and prior term have the same meaning with [6], except that [6] sets a fixed value as the covariance of the prior constraints for all loop closure edges, which can be interpreted intuitively as the admissible error threshold being the same for all loop closure edges. On the contrary, we propose using different thresholds for different loop closures depending on their quality, provided by the clustering process. As explained in previous session, two types of intra-consistent clusters are generated, one that is consistent with the odometry edges, the other not. The optimizer should be encouraged to accept the former type of edges. We use the linear switchable constraint proposed by [6] and the recommended value for σ is 1. For a good cluster that is consistent with the odometry edges, we decrease the value of σ to increase the penalty if it is scaled down.

$$\sigma_i = \begin{cases} 1 & if\ \ E(i) \in C_m\ \ \&\ \ quality_{C_m} = 0 \\ 0.1 & if\ \ E(i) \in C_m\ \ \&\ \ quality_{C_m} = 1 \end{cases} \tag{23}$$

The last term in equation (22) accounts for the clustering, which encourage the edges within one cluster to move towards the same direction. After replacing the detailed Mahalanobis distance form with symbol χ_i^2, we can express the cost terms corresponding to all loop closure edges in cluster C_m as below.

$$Cost(C_m) = \underbrace{\sum_{E_i\in C_m} s_i^2\chi_i^2}_{\text{Scaled Loop Closure}} + \underbrace{\sum_{E_i\in C_m}(\lambda_i - s_i)^2\sigma_i^{-1}}_{\text{Prior}} + \underbrace{\sum_{\substack{E_i,E_{i'}\in C_m \\ E_i\neq E_{i'}}}(s_i - s_{i'})^2\sigma_m^{-1}}_{\text{Clustering}} \tag{24}$$

To determine the value of σ_m, we examine the cost terms corresponding to edge E_i from cluster C_m as below.

$$Cost(E_i) = \underbrace{s_i^2\chi_i^2}_{\text{Scaled Loop Closure}} + \underbrace{\underbrace{(\lambda_i - s_i)^2\sigma_i^{-1}}_{\text{Prior}} + \underbrace{\sum_{\substack{E_{i'}\in C_m \\ E_{i'}\neq E_i}}(s_i - s_{i'})^2\sigma_{ii'}^{-1}}_{\text{Clustering}}}_{\text{Penalty}} \tag{25}$$

By observing the penalty term, it is apparent that when $s_i = s_{i'}$, the clustering term is zero, therefore has no effect on the optimization. When $s_i \neq s_{i'}$, in order to let the cost terms representative of the prior constraints and clustering constraints have the same weights in the formulation, we assert:

$$\sigma_{ii'} = (D_m - 1)\sigma_i \tag{26}$$

where D_m represents the number of edges in cluster C_m. When accumulating all the cost terms for cluster C_m, cost term accounting for every pair of loop closure edges will show up twice since each

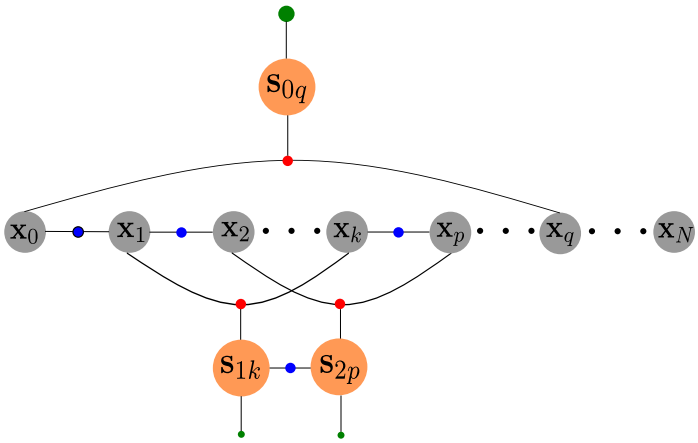


Figure 3: Factor graph representation of the proposed formulation

of them is related to two edges, leading to

$$\sum_{E_i \in C_m} Cost(E_i) = \sum_{E_i \in C_m} s_i^2 \chi_i^2 + \sum_{E_i \in C_m} (\lambda_i - s_i)^2 \sigma_i^{-1} + 2 \sum_{\substack{E_i, E_{i'} \in C_m \\ E_{i'} \neq E_i}} (s_i - s_{i'})^2 \sigma_{ii'}^{-1}. \tag{27}$$

Comparing equation (24) and (27), we obtain:

$$2\sigma_{ii'}^{-1} = \sigma_m^{-1}. \tag{28}$$

By substituting equation (26) into (28), the covariance value can be determined as

$$\sigma_m = \frac{(D_m - 1)\sigma_i}{2}. \tag{29}$$

In this way, we integrate the cluster structure as well as quality into the robust formulation and it can be interpreted on a factor graph, as shown in Fig.3.

The robust formulation is implemented within the OptimizerGTSAM file in rtabmap. Please note the authors of [6] made mistakes when computing the Jacobians in their gtsam implementation. They are corrected in our source code.

5 Experiments

5.1 Datasets

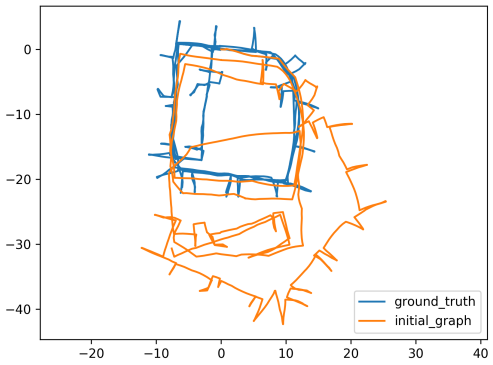
We conducted experiments with 2D datasets that are largely different in graph sparsity, trajectory length and odometry noise, details in Table 1. The mean value of relative pose error (RPE) between the ground truth and the initial graph is computed to reflect the odometry noise. RPE is considered to be well-suited for measuring the drift of odometry (more details can be found in [12]). It is often assumed that covariance provided by the *front-end* could accurately reflect the real odometry noise, however, that's not always the case according to our observations. The initial graphs from odometry are compared with ground truth in Fig.4. For Bicocca, the original dataset already contains outliers, therefore we process this dataset as it is. For other datasets, since we do not have access to the raw measurements from the *front-end*, we used a script [6] to generate outliers. The covariance of generated outliers is obtained by averaging the covariance of all inliers. We also conducted experiments with KITTI-00 and KITTI-02 sequences by tuning the parameters in the *front-end* to generate real-life outliers. Now we present and analyze the results for each dataset.

5.2 The Bicocca Dataset

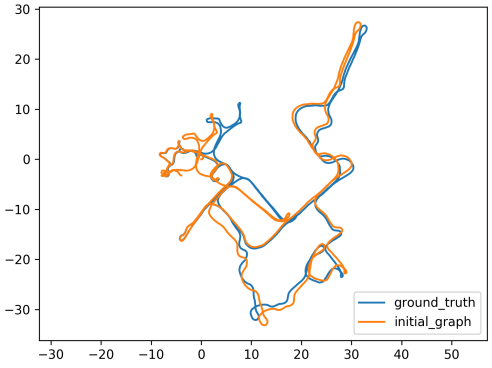
Latif [5] provided the Bicocca dataset with varying confidence parameter for the *front-end* place recognition system, and the number of loop closures ranged from 446 to 23. Here we are using the dataset with lowest confidence parameter, Bicocca-000, which contained 446 loop closures and is plotted in Fig. 5a. Four outlier rejection methods are compared: Switchable Constraints (SC)

Table 1: Information on the datasets used

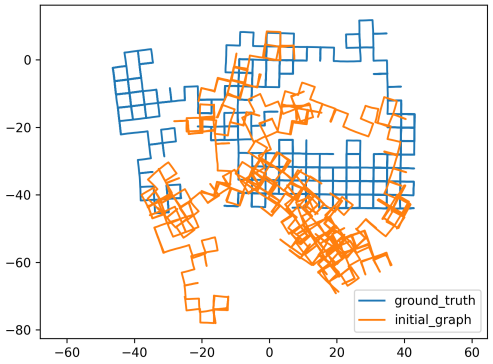
Dataset	Synthetic/ Real	2D/3D	Poses	Loop-Closures	RPE(m)	Outliers Source
Bicocca-000	Real	2D	8357	446	0.0005	Real
CSAIL	Real	2D	1045	128	0.004	Synthetic
Manhattan	Synthetic	2D	3500	2099	0.014	Synthetic
Intel	Real	2D	1228	256	0.019	Synthetic



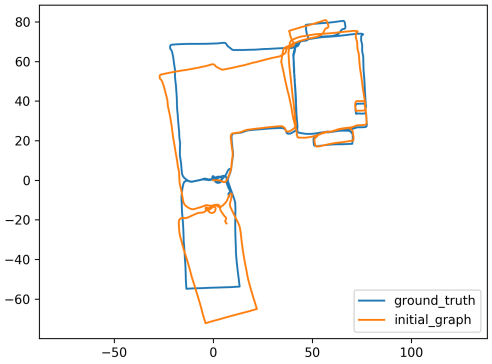
(a) Intel



(b) CSAIL

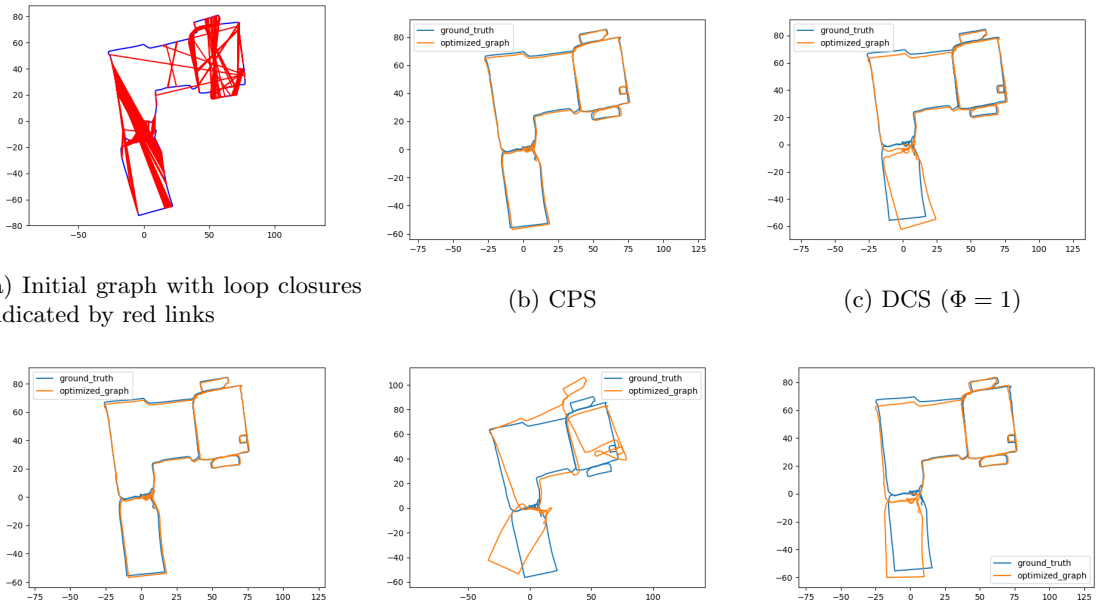


(c) Manhattan



(d) Bicocca-000

Figure 4: Comparison plot between ground truth and initial graph from odometry



(a) Initial graph with loop closures indicated by red links

(b) CPS

(c) DCS ($\Phi = 1$)

(d) DCS ($\Phi = 5$)

(e) OFCC

(f) SC

Figure 5: Comparison plot between ground truth and optimized graph

[6], Dynamic Covariance Scaling (DCS) [7], Objective Function Change Checking (OFCC) [10], and Cluster-based Penalty Scaling (CPS). In Fig.5, we compare the resultant graphs with ground truth, which is obtained by solving a clean dataset generated with high confidence parameter for the front end [5], using non-robust solver. And the mean of absolute translation error (ATE) is presented in Table 2. It is apparent that OFCC failed to reject all the outliers and the optimized graph is severely distorted. We can see that CPS and DCS with tuning parameter set to 5 performed similarly while DCS with tuning parameter 1 as well as SC had larger errors. We used $\Phi = 5$ for DCS because authors themselves proposed using this value for the Bicocca Dataset. Despite producing good results, the specific value for this tuning parameter cannot be explained. We argue that having a parameter that needs to be fine tuned for different datasets would make the method difficult to use.

5.3 The CSAIL Dataset

The CSAIL Dataset has low odometry noise and no outliers. We added varying number of outliers so that 10%, 20%, 30%, 40%, 50% of outliers are present in the resultant graph. For the 50% outliers scenario, we also experimented with grouped outliers as proposed in [6] and the results are shown in Table 2. Each trial is ran ten times. The mean of ATE and the percentage of accepted inliers are plotted in Fig.7 . All outliers have been rejected for all 200 trials, therefore not plotted. Overall, all methods performed reasonably well since the highest ATE value is as low as 0.05m. We can see that CPS manages to achieve the lowest ATE due to its capability to accept more than 99% of the inliers even when 50% of the loop closures are outliers. DCS and SC achieved slightly higher ATE because they rejected more inliers. We note that despite succeeding in all the trials to accept all inliers, OFCC has the highest ATE of all methods. Another interesting insight is that DCS with tuning parameter $\Phi = 1$ worked well for CSAIL dataset while tuning parameter $\Phi = 5$ worked well for Bicocca dataset, despite both datasets have low odometry noise. This further confirmed the difficulty in parameter turning for DCS.

5.4 The Manhattan Dataset

The manhattan dataset is a synthetic dataset consisting of grid patterns. Researchers developing outlier rejection techniques have frequently used this dataset to validate their performance and conveniently ignore the unrealistic scales of the covariance matrix. For all the edges, including odometry and loop closure, the information matrix , which is the inverse of covariance matrix, is a diagonal matrix with the value 44.7214 on the diagonal. Although it is difficult to argue whether the value itself makes much sense, the translation terms and rotations terms should not have the same

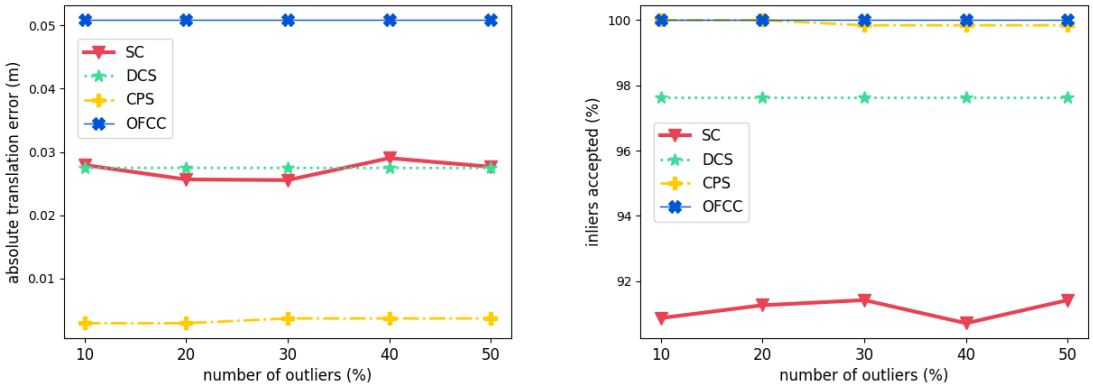


Figure 6: Comparison of different outlier rejections methods on CSAIL dataset

value due to their different units. We observe from real world datasets that the rotation terms of the information matrix are usually orders of magnitude larger than that of translation terms. Therefore, we conducted the experiments with varying number of outliers as well as grouped outliers on the original dataset, and then revised dataset so that the rotation terms of information matrix is fixed to 4472.14, two orders of magnitude larger than the translation terms. Experiments with different percentage of outliers are repeated. The ATE, inliers acceptance rate and outliers acceptance rate are plotted in Fig. 8.

When handling the original dataset, we can see that OFCC failed by rejecting large quantity of inliers even when only 10% of the loop closures are outliers. DCS performed reasonably when the number of outliers are low, but soon deteriorated as the number of outliers increased. SC achived the lowest outlier acceptance rate and highest inlier acceptance rate, however, its ATE when 50% of outliers are present is slightly larger than CPS. This is because CPS, even when accepting outliers, tend to only accept outliers that cause local distortion due to its explicit usage of local consistence within clusters. This is also confirmed by the comparison between DCS and CPS: despite having a higher outlier acceptance rate than DCS, CPS achieved much lower ATE than DCS. Typical scenarios when DCS, SC, CPS accepted outliers are shown in Fig. 8.

After modifying the rotation term of information matrix for all the edges, we observe largely different behaviors, plotted in the right columns of Fig. 8. CPS is the only method that maintained its performance, achieving low ATE despite slight drop in the inlier acceptance rate. On the contrary, both SC and DCS performed much worse by rejecting large quantity of inliers. This sensitivity to information matrix again exposed the drawback of fixing a threshold on the admissible error of a single edge. Since the information matrix served as weights, the cost terms of all edges increased after we increased the value of rotation terms in the information matrix. Therefore, a large group of inliers ended up being rejected for exceeding the threshold.

5.5 The Intel Dataset

The Intel Dataset has the highest odometry noise out of all the 2D datasets we used for experiments. We inject varying number of outliers and conduct experiments with four different methods. The ATE, inliers acceptance rate and outliers acceptance rate are plotted as Fig. 9. We can see that SC failed in this dataset by being too conservative, rejecting all the outliers however also rejecting too many inliers. The default threshold on the admissible error is apparently too low. DCS performed well when the number of outliers is relatively low, and its performance deteriorated as the number of outliers increase. OFCC and CPS performed well due to their capability to accept most of the inliers. ATE of CPS is slightly lower than OFCC when 50% of outliers are present due to its capability of rejecting more outliers. The detailed results are shown in Table 2.

5.6 The KITTI Datasets

To obtain real outliers, we processed the KITTI-00 and KITTI-02 sequences using rtabmap [13] and tuned the *front-end* parameters to produce more loop closures. As a result, we produced 101 loop closures for KITTI-00, which consists of 453 poses. For KITTI-02, we produced 36 loop closures with respect to its 464 poses. The initial graphs are plotted in Fig. 10 and using proposed method to mitigate the effect of outliers, we achieved optimized graph as in Fig 11. It is apparent that all

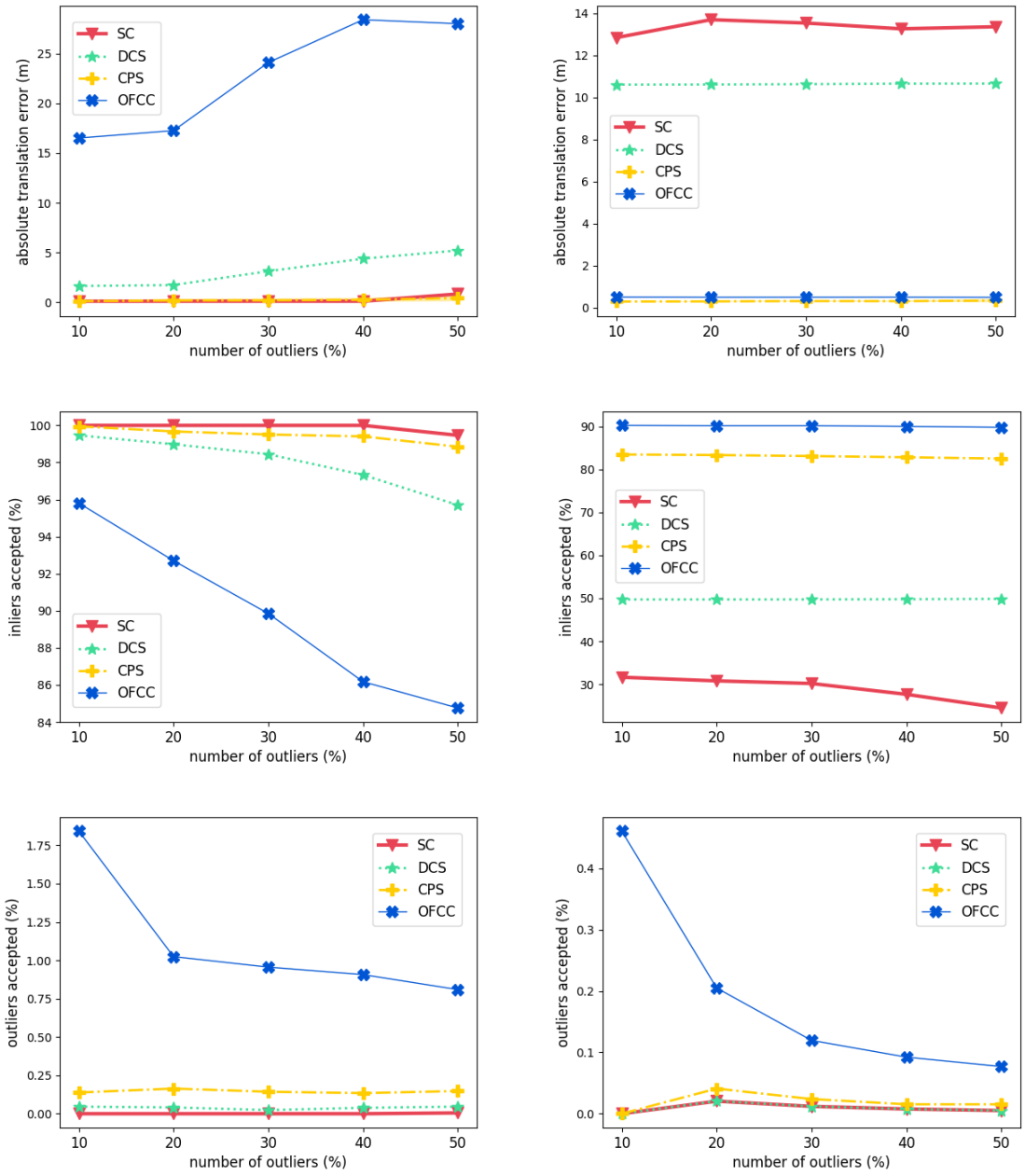
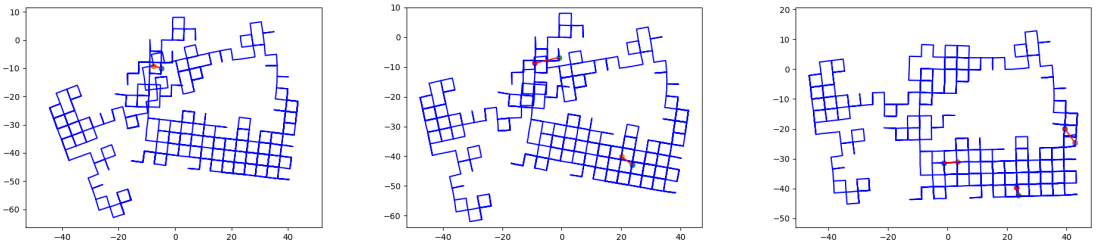


Figure 7: Comparison of different outlier rejections methods on Manhattan dataset: the left column shows experimental results from the original dataset and the right column shows experimental results from the dataset with modified rotation terms in information matrix



(a) SC failure case (b) DCS failure case (c) CPS failure case

Figure 8: Comparison of different outlier rejections methods when they accepted outliers on Manhattan dataset: blue lines indicates the optimized graph, and red lines indicates the accepted outliers

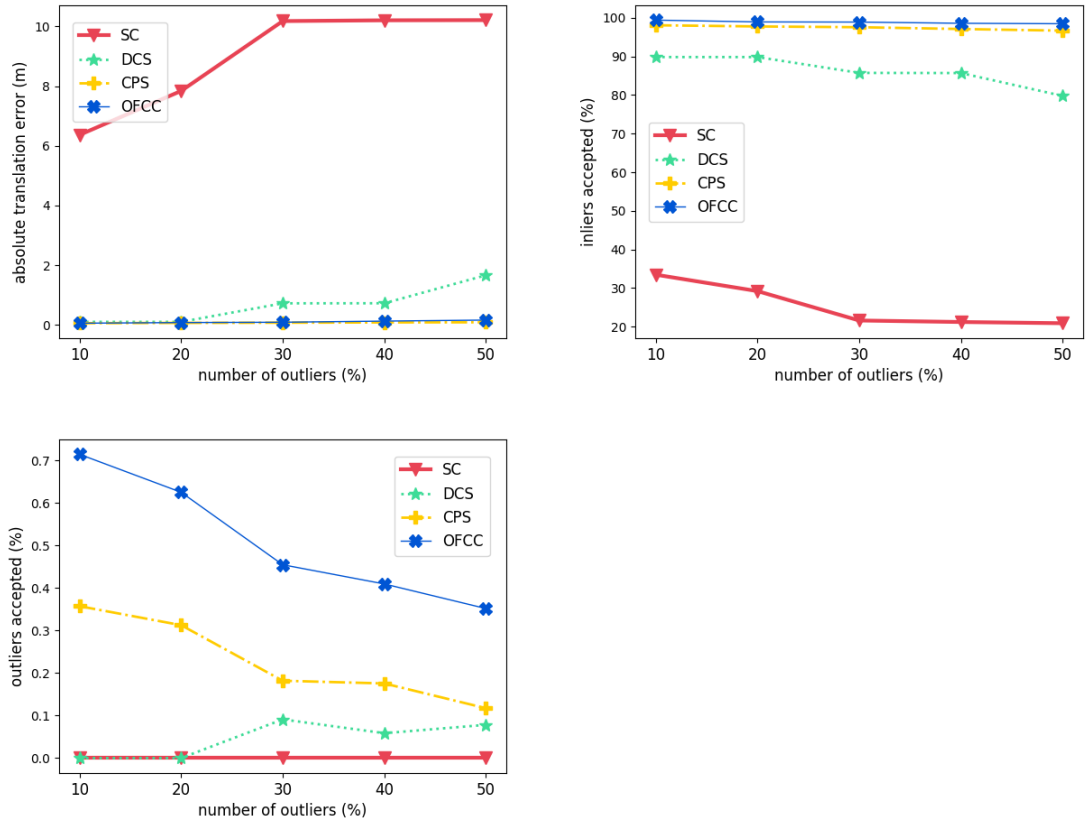
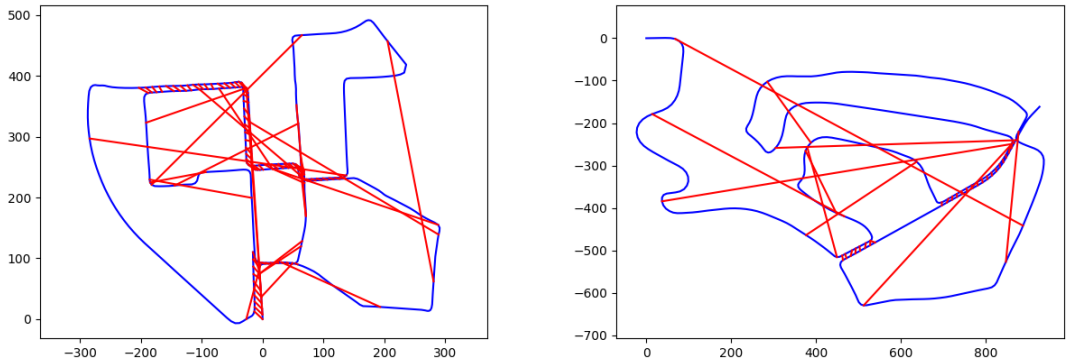


Figure 9: Comparison of different outlier rejections methods on Intel dataset

Table 2: Evaluation of absolute trajectory error (m)

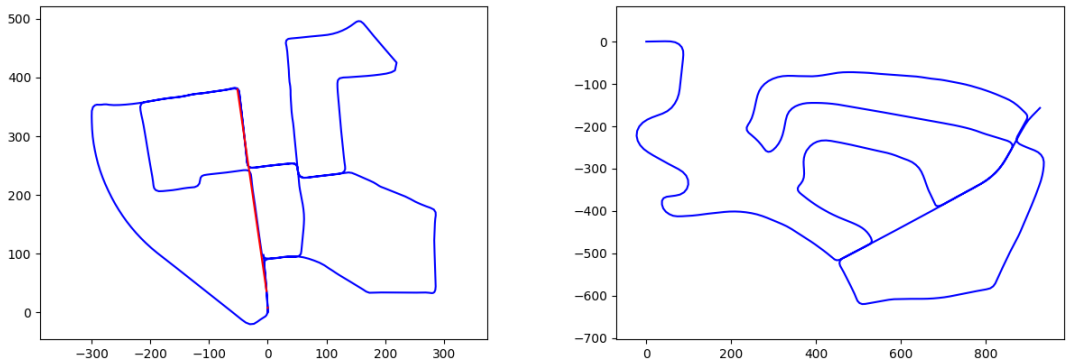
	Switchable Constraints				**Dynamic Covariance Scaling**			
	Random		**Group**		**Random**		**Group**	
	mean	**max**	**mean**	**max**	**mean**	**max**	**mean**	**max**
Bicocca	2.871	-	-	-	2.818	-	-	-
Csail	0.027	0.030	0.029	0.030	0.027	0.027	0.027	0.027
Manhattan	0.245	7.375	3.624	14.914	3.212	15.480	4.394	15.987
Manhattan (cov)	13.346	14.463	-	-	10.640	10.712	-	-
Intel	8.961	10.224	8.630	10.219	0.659	9.504	0.099	0.112
	Cluster-based Penalty Scaling				**OFC Checking**			
	Random		**Group**		**Random**		**Group**	
	mean	**max**	**mean**	**max**	**mean**	**max**	**mean**	**max**
Bicocca	1.102	-	-	-	10.531	-	-	-
Csail	0.003	0.007	0.003	0.007	0.051	0.051	0.138	0.921
Manhattan	0.219	1.196	0.482	0.991	22.856	38.010	24.163	34.021
Manhattan (cov)	0.320	0.577	-	-	0.505	0.561	-	-
Intel	0.070	0.133	0.257	1.437	0.099	0.371	0.273	0.956

outliers have been rejected.



(a) KITTI-00 (b) KITTI-02

Figure 10: Initial graphs of KITTI dataset



(a) KITTI-00 (b) KITTI-02

Figure 11: Optimized graphs of KITTI dataset

6 Conclusions and Future work

Without fine tuning the parameters, our methods performed better than other state-of-the-art outlier rejection methods on data sets with varying odometry noise, trajectory length, and graph sparsity. We circumvent the difficulty in finding the correct threshold for outlier rejection by extracting information on local consistence.

For future works, we will perform in-depth analysis of the processing time as well convergence speed of proposed method. We also plan to adapt the proposed method for online outlier rejection. In addition, we observed from our experiments that covariance estimation has a huge effect on the optimization performance. There are very limited studies on this issue, probably due to the fact that covariance of the edges are computed within *front-end* modules. We will explore possible ways to achieve a tighter integration between the *front-end* and the *back-end*, so that the effect of poorly estimated covariance can be studied.

References

[1] T. D. Barfoot, "State estimation for robotics," *State Estimation for Robotics*, pp. 1–368, 2017.

[2] G. H. Golub and R. J. Plemmons, "Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition," *Linear Algebra and its Applications*, vol. 34, pp. 3–28, 1980.

[3] Lukas Polok, "Accelerated Sparse Matrix Operations in Nonlinear Least Squares Solvers," *Ph.D. Thesis at Faculty of Information Technology, Brno University of Technology*, 2016.

[4] V. Ila, L. Polok, M. Solony, P. Smrz, and P. Zemcik, "Fast covariance recovery in incremental nonlinear least square solvers," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 4636–4643, 2015.

[5] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time for pose graph SLAM," *International Journal of Robotics Research*, vol. 32, no. 14, pp. 1611–1626, 2013.

[6] N. Sunderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1879–1884, 2012.

[7] P. Agarwal, G. D. Tipaldi, and L. Spinello, "Robot Map Optimization using Dynamic Covariance Scaling," *2013 IEEE International Conference onRobotics and Automation (ICRA)*, 2013.

[8] P.-Y. Lajoie, S. Hu, G. Beltrame, and L. Carlone, "Modeling Perceptual Aliasing in SLAM via Discrete-Continuous Graphical Models," pp. 1–13, 2018.

[9] M. C. Graham, J. P. How, and D. E. Gustafson, "Robust incremental SLAM with consistency-checking," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 117–124, 2015.

[10] F. Bai, T. Vidal-Calleja, S. Huang, and R. Xiong, "Predicting Objective Function Change in Pose-Graph Optimization," in *IEEE/RSJ 2018 International Conference on Intelligent Robots and Systems*, 2018.

[11] E. B. Olson and M. Kaess, "Evaluating the performance of map optimization algorithms," *RSS Workshop on Good Experimental Methodology in Robotics*, 2009.

[12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," *IEEE International Conference on Intelligent Robots and Systems*, pp. 573–580, 2012.

[13] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.