Master's Thesis of Engineering

Robust pose graph optimization with loop closure outliers

루프 폐쇄 이상치를 사용한 강력한 포즈 그래프 최적화

August 2023

Graduate School of engineering
Seoul National University
Naval Architecture and Ocean Engineering Major

CAMILLE KESSELER

Robust pose graph optimization with loop closure outliers

Thesis supervisor: Tae-Wan Kim

Submitting a master's thesis of engineering

June 2023

Graduate School of Engineering
Seoul National University
Naval Architecture and Ocean Engineering Major

CAMILLE KESSELER

Confirming the master's thesis written by
July 2023

Chair	Myung-Il Roh	(Seal)
Vice Chair	Tae-Wan Kim	(Seal)
Examiner	Bo Woo Nam	(Seal)

Abstract

With the increasing need of autonomous robots for complicated environment situation such as underwater application, more robust algorithm is needed. In simultaneous localization and mapping algorithm, one of the core parts is the back end. The noisy measurement and robot trajectory are process to correct the drifting error using loop closure measurement (recognition of previously visited place). The process of optimizing the robot poses with respect to the sensor measurements is called pose graph optimization (PGO). Solving a PGO problem is equivalent to solve a maximum likelihood estimation problem where the objective function is the error between the measurement and the poses. The classical framework is to use a least−square formulation. However, this formulation has several drawbacks: The sensitivity to poses initialization first can lead to a local minima solution as it is a nonconvex problem. Then the presence of wrong measurement with large error, also called outliers, can lead to arbitrary wrong solution. In this research, we aim at studying a method for PGO which leverages the problem of initialization and is robust to outliers' presence. The initialization sensitivity problem comes from the nonconvexity of the minimization problem as it introduces multiple local minima. The proposed solution is to relax the problem into a convex one with a single global minimum. The solution of the relaxed problem can be reprojected on the initial nonconvex problem feasible set. Additionally, using this method we have a contract on the certifiability of our solution, i.e we can ensure that the solution is the global minima, or we detect the failure. For the sensitivity to outliers, it mainly comes from the fact that the

formulation is quadratic in the error terms so if one measurement contains a wrong large error it will dominate the objective function. The proposed approach here is the use of M−estimator. A M−estimator is adding a loss function around the error term to mitigate its impact if it is too large. This thesis aims at comparing different loss function that can be used on the chosen convex relaxation approach. Additionally, we suppose that only edge which are loop closure can be outliers. After deriving the formulation corresponding to our choice, we test on 3 synthetic datasets the different loss function and compare them. Our results show that the convex loss function, i.e L1, L2, identity tested here do well for highly connected pose graph but failed to stay robust for low connectivity pose graph.

Keyword: Pose graph optimization, Outliers, Convex relaxation, M-estimators, Loss function.

Student Number: 2020-29586

TABLE OF CONTENTS

List of Figure

List of Table

1 Introduction

1.1 Background

Nowadays, we have seen a rapid and steady improvement in robotics techniques, and noticeably for the autonomous robot field. Autonomous robots are the ones operating without human control. For instance, when a robot is given the task to go to a precise position, in the past, we had either to detail its trajectory beforehand or to use a remotely operated vehicle (ROV). For autonomous robot, we only need to fix the final position. The robot will decide by himself how to attain the goal. Researchers first developed autonomous technologies for small robots in indoor environments. Yet the recent algorithm and sensor improvements made it possible to apply autonomous robots for outdoor applications such as self-driving cars. And recently, more projects using marine robots are taking place and pinpointing the possibility of using similar technologies and algorithms for naval robotics as shown in Zereik et al., 2018.

There is a wide range of possible uses for autonomous robots in marine robotics such as for military operations, scientific and environmental research, transport of people or good or mining and oil industry. The marine robots are usually separated in two categories: unmanned surface vehicles and unmanned underwater vehicles. The former type is design to navigate at the surface of the sea. One application is for merchandise transport, and more specifically for self-docking. The latter type is aiming for underwater operation. It can be for deep sea observation or ship

hull inspection as illustrate in figure 1-1. More applications examples can be found in Yuh et al. (2011).

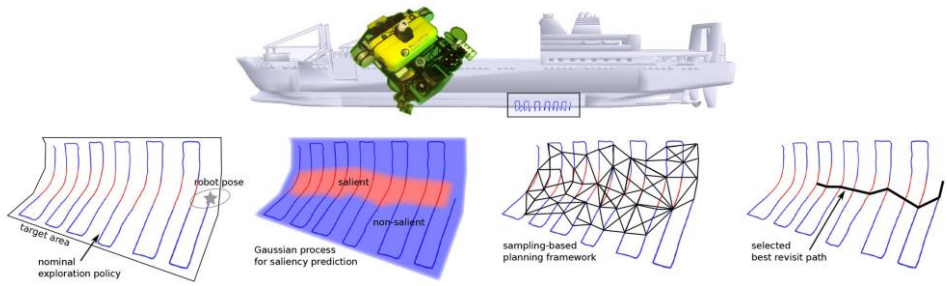

Figure 1-1 Active SLAM framework, demonstrated for ship hull inspection (Chaves et al., 2016)

In all type of applications for autonomous navigation, the mainstream strategy which give the best result so far is simultaneous localization and mapping (SLAM). In this algorithm, the robot is computing its position and creating the map of its environment at the same time. This bring an additional complexity as the localization and mapping problem are intimately link. SLAM was usually called a chicken or the egg problem, because we need the map to compute our position and to build a map, we need to know our position. Nowadays different type of SLAM algorithms exists in function of the sensor data available on the robot and the real-life application. Still, they all tend to follow the same architecture show in figure 1-2.

The input data for SLAM depends on the robot's sensor at disposition, it can be images from a camera, point cloud from a lidar or spatial position from a GPS for instance. The output is generally the robot position and the map of the environment, but how we

represent the robot position, and the map depends on our need. The map can be a full 3D map representation or just the localization of interesting object in the space. The robot localization information is usually stored in a pose graph, the nodes of the graph are the estimated positions, and each edge of the graph contains a sensor measurement between these 2 positions.

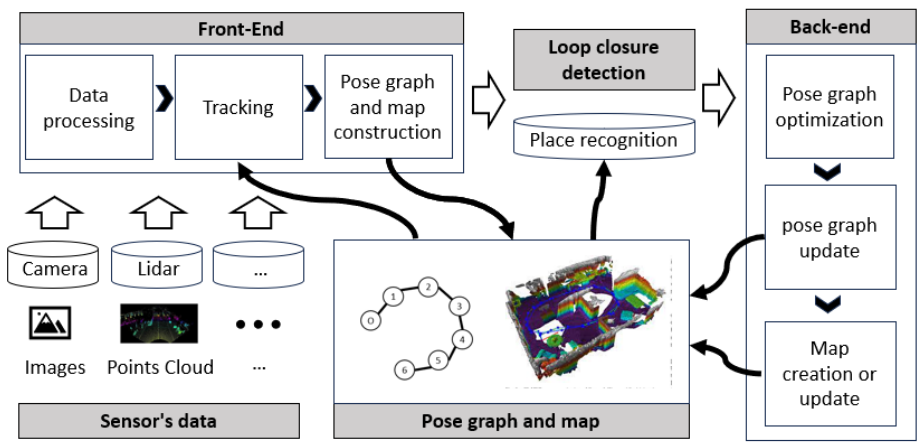

Figure 1-2 SLAM overall architecture

As for the main core of the algorithm, it can be separated into 2 interconnected parts: front end and back end. The front-end is a quick real-time process which process the raw sensor input data to compute a first approximate position of the robot, it also uses this processed data to initialize and update the pose graph and the map with noisy information. Sometimes the robot can recognize a place it already visits when comparing sensor data and the created map, when this happens, we create an extra edge in the pose graph linking the two corresponding poses. This type of edge is called loop closure edge as they form a loop in the pose graph. After the loop closure detection, we call the back-end part. Usually, it is a

slower process than the front-end and cannot be perform real-time. Its main goal is to reduce the error in the estimated poses and map by the front-end due to the noise in the sensor. It first performs a pose graph optimization (PGO) to reduce the overall error in robot poses and then update the pose graph when finished. Finally, it also corrects the map using this new pose and the data available. In this approach the front-end is solving the localization problem using sensor data and the past information corrected by the back end. The back end is doing the mapping jobs based on the approximated positions compute by the front-end. They run at the same time on different thread, and so they can solve a SLAM problem.

1.2 Objective and method

One crucial step in the whole SLAM process is the pose graph optimization during the back end and we will focus on this part. As mentioned before, the noisy full trajectory is corrected using previous poses, new poses, and loop closure information. Pose graph optimization suffers from several short-coming in classical approach: timescale, initialization sensibility and outlier sensibility.

In more details, the first important point to look at in PGO is the large size of the data to optimize and not all method for solving it scale well i.e., some methods are too slow to use in SLAM and can only be used for offline mapping. Another point is that the estimated poses are computed from all the measurements (edges) using a Maximum Likelihood (ML) formulation. That is solving a non-linear least squares minimization problem. Classical solvers such as g2o from Kümmerle et al. (2011) are based on gradient descend

approach which is sensitive to initialization as the problem contains multiple local minima (non−convex formulation). The scale and time problem will not be study in these first approach. For the initialization sensibility problem, we choose to focus on method using convex relaxation of the MLE formulation to solve PGO such as in Carlone et al. (2018). This method leverages the problem of multiple local minima.

The last point is the sensitivity to outliers, we want to improve the robustness of the pose graph estimation algorithm against outlier. In the pose graph representation of the robot's trajectory, each node is an estimated pose, and each edge is a measurement between two poses. Using this information, PGO try to minimize the error on edge by correcting the robot poses. But in this approach, if one edge measurement is corrupted by a larger noise or simply is an edge that does not describe the reality of our robot position, then the optimization process will converge to a wrong trajectory. We call this type of edge outliers, and one can severally impact the quality of the result trajectory. The figure 1−3 illustrates a failure of PGO because of wrong loop closure detection.

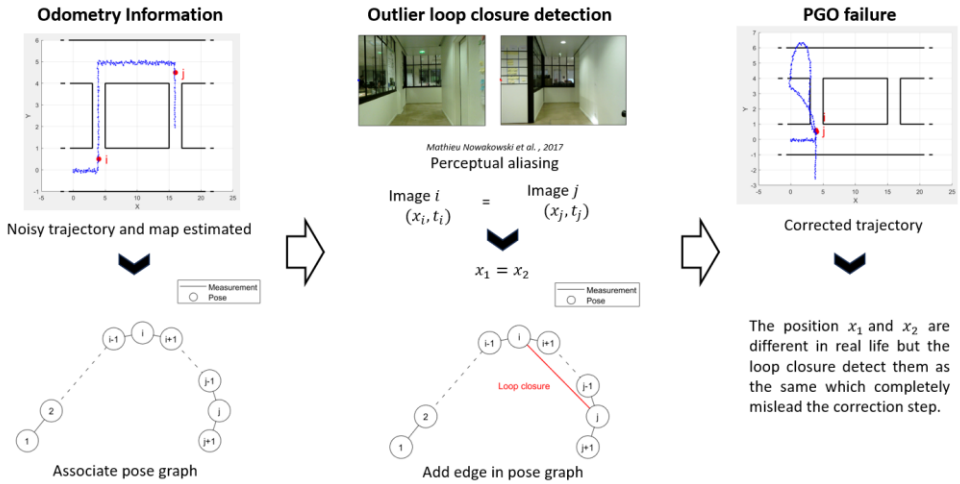

Figure 1-3 Example of failure due to loop closure outlier

In SLAM, the front-end will generate new pose and measurement for the pose graph. When using images as input data it is more likely that some outliers will appear over time, especially when trying to recognize previously visited place (loop closure edge generation). To reduce the impact of these outliers on the PGO, we take the mitigation approach which means we want to keep using the information of this edge, but we reduce the importance given to it as it seems suspicious to have large error in the pose graph. One way to achieve this that we pick is to add a loss function to the error term in the MLE formulation, which a method called M-estimators (Bosse et al., 2016).

We want to study the impact of different loss functions apply to the loop closure edges to reduce the outlier impact in pose graph optimization solved with a convex relaxation approach in 3D. For this we propose a 2-stage approach where we solve first only the rotation problem by relaxing the problem, then round the solution and finally solve the translation problem using the rounded rotation

matrix. The flowchart of our algorithm is show in image 1-4.

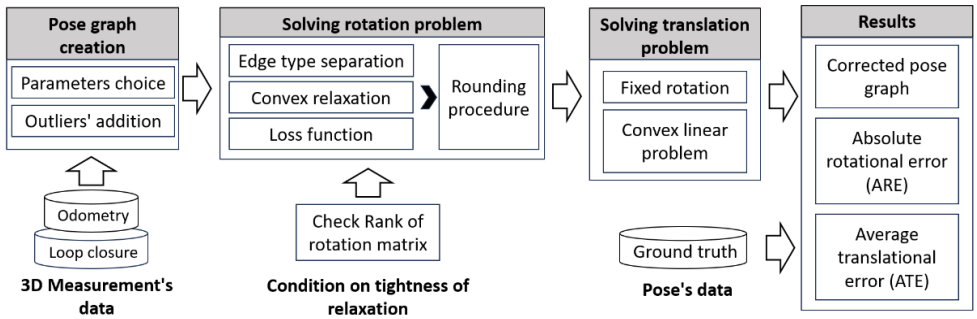

Figure 1-4 Input, output, and flowchart of the proposed PGO

1.3 Contribution

Our thesis brings new items and more details study to the research about robust pose graph optimization:

1) Derivation of the convex relaxation for the MLE general formulation with loss function in 3D.

2) Study and comparison of different loss functions use only on the loop closure edge.

1.4 Summary

In Chapter 1, we introduce the topic of autonomous robotics and the drawbacks of pose graph optimization when use in SLAM algorithm. We also explain how we intend to contribute to the robust pose graph optimization literature. In Chapter 2, we will define the key concepts necessary for the understanding of the thesis and precisely detail the theoretical background needed. In Chapter 3, different PGO algorithms link to our approach are presented and compared with our approach. The chapter 4 and 5 explains in more details the proposed PGO algorithm with the different choice made in its formulation, such as the 2-steps approach, the convex relaxation, and the loss function. In Chapter 6, the different loss functions are explained and studied. Starting from chapter 7, we will focus on the testing with first the description of the datasets used and the different parameters for testing. In chapter 8, the simulation results will be show and interpreted. In last, we will conclude and highlight the possible future works for this thesis.

2 Key Concepts and Theoretical background

To understand completely this thesis, we need to explain in more details several concepts and their theoretical parts. In the introduction, we write that the trajectory of the robot is represent by a pose graph which is optimized to correct the error present in the measurement. In the part (2.1), we will detail the different representation possible for the positions and measurements and how to mathematically represent the pose graph optimization problem. Moreover, since we want to study robust pose graph optimization, the part (2.2) is given a review of how to handle outliers. In the last part (2.3), we explain the theory of convex optimization and how to use it even when the problem is not convex.

2.1 Pose graph optimization

The trajectory of a robot is defined as the path that it takes through space as a function of time. An equivalent definition can be all the spatial positions of the robot at the different time instants. And so, following this idea, we intuitively represent the trajectory as a pose graph. Each node is a position x_i at a certain instant t_i of the robot. Between each computed position we have a sensor measurement x_{ij} which is represented by an edge containing the transformation between the two positions (x_i to x_j). The edges between consecutive poses model odometry measurements and are from sensor such as IMU or wheel encoders. The edges between nonconsecutive nodes are for loop closure measurements as they permit to create loop in the graph model and are necessary against the drift of odometry measurement. To detect them, images of the environment are mainly used with the map information to interpret

if we already visit the position and if we can create an edge. Grisetti et al. (2010) gives a deeper explanation of Graph-Based SLAM.

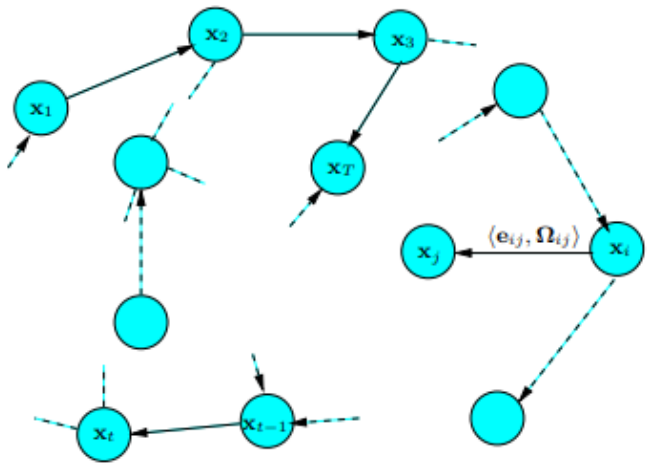

Figure 2-1 Pose graph representation of robot trajectory (Grisetti et al., 2010)

Using the graph representation as illustrated in figure 2.1, we can used general graph optimization techniques to find the "best" positions/nodes given all the measurement/edges. This approach to determine the trajectory of our robot highlights the spatial structure of the problem and separates the work in 2 parts: first builds the graph from the raw sensors data (front-end) and second optimizes the node position to fit the overall measurements (back-end). The optimization part is not related to the sensors measurement but to the pose graph model chosen, which means the result will depend on the robot pose model and the sensor measurement model define when creating the graph.

2.1.1 Robot's pose models

The pose of a robot (rigid body assumption) is composed of 2 elements: the position and the orientation. In a classical robotics setup as explained in Gao et al. (2021), we define a fixed world coordinate system as a basic for the map and a moving coordinate system which represents the robot. The position and orientation of the robot are described as the coordinate transform from the world coordinate to the robot coordinate system as illustrated in Figure 2-2. So, the robot position will be seen as a translation and the robot orientation as a rotation. As we consider a rigid body motion, we can use Euclidean transform to represents the motion. It is composed as we said of rotation and translation. The translation is simply a dimension 3 vector $\boldsymbol{t}_{WC}$ which is the vector from the world system's origin pointing to the robot system origin express in the world coordinate system.

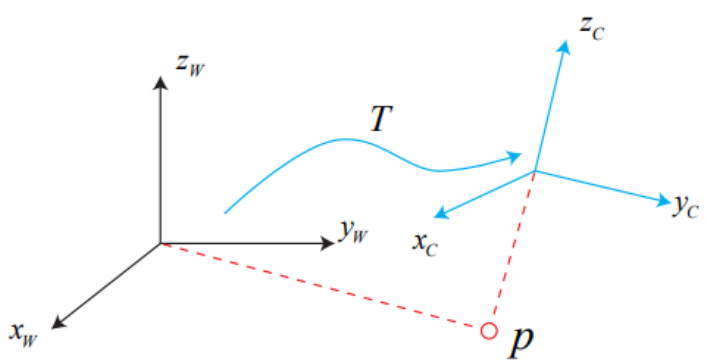

Figure 2-2 Coordinate transform (Xiang Gao et al., 2021)

For the rotation, it exists different way of expressing it mathematically. The most well-known is the rotation matrix representation. They are orthogonal matrices with determinant 1 and form the special orthogonal group (SO). For the 3D case, all the rotation of the three-dimensional space are the set SO(3) defined as:

$$SO(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times3} | \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}, det(\boldsymbol{R}) = \boldsymbol{1}\} \quad (2-1)$$

So, the rotation from the robot coordinate system to the world coordinate system can be represented by the 3x3 matrix $\boldsymbol{R}_{WC}$. Using this matrix representation, if we have the vector $\boldsymbol{a}_C$ measure in the robot coordinate system, then its coordinate $\boldsymbol{a}_W$ in the world coordinate system can be compute using the formula:

$$\boldsymbol{a}_W = \boldsymbol{R}_{WC}\boldsymbol{a}_C + \boldsymbol{t}_{WC} \quad (2-2)$$

This representation used 9 quantities to describes 3 degrees of freedom. So, an alternative method is to use a single 3-dimensional rotation vector which represent 3 Euler angles of rotation (the actual rotation is decomposed into 3 consecutives rotation around predefined Euler axis). This is a minimal representation for the rotation, but it suffers from the singularity problem and so cannot be used for expressing pose directly. The last possible representation is a 4-dimensional vector called quaternion, it a compact and not singular form but non intuitive. They are extended complex numbers with a real part and three imaginary parts. We will use rotation matrix for the sake of simplicity in our research.

In our pose graph, each pose to estimate is noted $\boldsymbol{T_i} \triangleq [\boldsymbol{R_i} \quad \boldsymbol{t_i}]$ which comprises a translation vector $\boldsymbol{t_i} \in \mathbb{R}^3$ and a rotation matrix $\boldsymbol{R_i} \in SO(3)$. Additionally, they can be represented together in a matrix form using the special Euclidean group, define for the 3D case as:

$$SE(3) = \left\{ \boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{O}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \middle| \, \boldsymbol{R} \in SO(3), \boldsymbol{t} \in \mathbb{R}^3 \right\} \qquad (2-3)$$

2.1.2 Sensor measurements models

Each edge in the pose graph contains information about a measurement from either processed sensors or computed loop closure. Since the pose is a rotation matrix and a translation vector, the measurement is also define using a rotation matrix and a translation vector, but they define a motion between 2 coordinates system not a coordinate system. In Euclidean geometry, if we have the pose $\boldsymbol{T_i}$ and the pose $\boldsymbol{T_j}$, the transform between the two of them is $\boldsymbol{T_{ij}} \triangleq [\boldsymbol{R_{ij}} \quad \boldsymbol{t_{ij}}]$ and follows the formulae $\boldsymbol{t_{ij}} = \boldsymbol{R_i^T}(\boldsymbol{t_j} - \boldsymbol{t_i})$ and $\boldsymbol{R_{ij}} = \boldsymbol{R_i^T R_j}$.

If the measurement were perfect, we could use this model, but the reality is that the measurements are noisy. So, we need to add an additional term to the measurement model, a noise term $[\boldsymbol{R_{ij}^{\epsilon}} \quad \boldsymbol{t_{ij}^{\epsilon}}]$. The model for the noisy measurement $\overline{\boldsymbol{T_{ij}}} \triangleq [\overline{\boldsymbol{R_{ij}}} \quad \overline{\boldsymbol{t_{ij}}}]$ is then:

$$\overline{\boldsymbol{t_{ij}}} = \boldsymbol{R_i^T}(\boldsymbol{t_j} - \boldsymbol{t_i}) + \boldsymbol{t_{ij}^{\epsilon}}, \qquad \overline{\boldsymbol{R_{ij}}} = \boldsymbol{R_i^T R_j R_{ij}^{\epsilon}} \qquad (2-4)$$

The noise is supposed to follow a probability distribution. The choice of the probability distribution type will strongly impact the

PGO results because it models how much of the measurement information can be wrong. There are several distributions that are commonly used for PGO formulation derivation (Carlone et al. (2018); Gómez, E. et al. (1998); Gao et al. (2021)), we will list them in the table 2-1 with their density function and parameters.

Table 2−1 Measurement noise distribution model

Distribution	Probability density function	Variable	Parameters
	Translation $t_{ij}^{\epsilon}\sim$		
Multivariate Gaussian or Normal	$\mathcal{N}(\boldsymbol{x};\,\mu,\Sigma)=\frac{1}{\sqrt{(2\pi)^d\lvert\Sigma\rvert}}*\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$	$\boldsymbol{x}\in\mathbb{R}^d$	Mean $\mu\in\mathbb{R}^d$ Covariance $0\preccurlyeq\Sigma\in\mathrm{Sym(d)}$
Multivariate exponential power	$ExpPow(\boldsymbol{x};\boldsymbol{\mu},\Sigma,\beta)=c*\exp\left(-\frac{1}{2}[(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})]^{\beta}\right)$	$\boldsymbol{x}\in\mathbb{R}^d$	Mean $\mu\in\mathbb{R}^d$ Covariance $0\prec\Sigma$ Kurtosis $\beta>0$ Normalization constant c

Rotation $\boldsymbol{R_{ij}^{\epsilon}}\sim$			
Wrapped gaussian for SO(3)	$\boldsymbol{R_{ij}^{\epsilon} = exp\left(\begin{bmatrix} 0 & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & 0 & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & 0 \end{bmatrix}\right)}$, where $\boldsymbol{\epsilon \sim \mathcal{N}(\mu, \Sigma)}$	$\boldsymbol{R_{ij}^{\epsilon} \in SO(d)}$ $\boldsymbol{\epsilon \in \mathbb{R}^3}$	Mean $\mu \in \mathbb{R}^3$ Covariance $0 \preccurlyeq \Sigma \in \mathrm{Sym(d)}$
Von Mises	$VonMises(\theta, \mu, \kappa) = c(\kappa)\exp(\kappa\cos(\theta - \mu))$	$\boldsymbol{R_{ij}^{\epsilon}(\theta) \in SO(2)}$ Angle θ	Mean μ Concentration κ
Isotropic Langevin	$Langevin(\boldsymbol{X}; M, \kappa) = \frac{1}{c_d(\kappa)}\exp\left(\kappa\, tr(M^T X)\right)$	$\boldsymbol{X \in SO(d)}$	Mode $M \in SO(d)$ Concentration $\kappa \geq 0$ Normalization constant $c_d(\kappa)$
Directional Laplace	$DLaplace(\theta; \mu, \kappa) = c * exp(-\kappa \left\lvert \sin\left(\frac{\theta - \mu}{2}\right)\right\rvert)$	$\boldsymbol{R_{ij}^{\epsilon}(\theta) \in SO(2)}$ $\theta \in (-\pi, +\pi]$	Mean $\mu \in (-\pi, +\pi]$ Scale $\kappa > 0$ Normalization constant c

2.1.3 Maximum likelihood estimation problem

The poses in the node are the variables of the optimization problem and the measurement edges are the information uses to adjust these poses variables. And more precisely, we know the sensor measurements and the chosen measurement model, so what we want is to minimize the errors between them by choosing the "best" value for the poses. This is equivalent to solve a maximum likelihood estimation problem where the θ is the unknow pose and the X_i are the measurement.

$$\max_{\theta} L(\theta|X_1, \dots, X_N) \tag{2-5}$$

The translation of this optimization problem is 'we search the poses θ that are more likely to be knowing the measurement X_i'. We rewrite the generic MLE formula first by replacing the generic variable by our problem one and the likelihood by a probability distribution.

$$\max_{\{\boldsymbol{t}_i\},\{\boldsymbol{R}_i\}} \mathbb{P}\left(\{\overline{\boldsymbol{R}_{ij}}\},\{\overline{\boldsymbol{t}_{ij}}\} \mid \{\boldsymbol{t}_i\},\{\boldsymbol{R}_i\}\right) \tag{2-6}$$

The probability distribution is for all nodes i, j, to simplify the formulation we suppose that all the measurement $\{\overline{\boldsymbol{R}_{ij}}\},\{\overline{\boldsymbol{t}_{ij}}\}$ are independent so the total probability becomes the product of all probably over the edges of the graph.

$$\max_{\{\boldsymbol{t}_i\},\{\boldsymbol{R}_i\}} \prod_{(i,j)\in\mathcal{E}} \mathbb{P}(\overline{\boldsymbol{R}_{ij}}, \overline{\boldsymbol{t}_{ij}} \mid \boldsymbol{t}_i, \boldsymbol{t}_j, \boldsymbol{R}_i, \boldsymbol{R}_j) \tag{2-7}$$

The next step is a well-known trick for MLE estimation problem reformulation, we take the negative logarithm of the objective function. Searching for the maximum of a function is the same as

searching for the minimum of the opposite function ($\max f = \min -f$). The logarithm function is monotone increasing, so it scales the function but doesn't change the position of the maxima/minima ($\min f = \min \log f$).

$$\min_{\{\boldsymbol{t}_i\},\{\boldsymbol{R}_i\}} -\log\left(\prod_{(i,j)\in\mathcal{E}} \mathbb{P}(\overline{\boldsymbol{R}_{ij}}, \overline{\boldsymbol{t}_{ij}} \mid \boldsymbol{t}_i, \boldsymbol{t}_j, \boldsymbol{R}_i, \boldsymbol{R}_j)\right) \qquad (2-8)$$

From this we just separate the logarithm of the product into the sum of logarithm. Additionally, we can suppose that the translation and rotation are separate term too.

$$\min_{\{\boldsymbol{t}_i\},\{\boldsymbol{R}_i\}} -\sum_{(i,j)\in\mathcal{E}} \ln\left(\mathbb{P}(\overline{\boldsymbol{R}_{ij}} \mid \boldsymbol{R}_i, \boldsymbol{R}_j)\right) - \sum_{(i,j)\in\mathcal{E}} \ln\left(\mathbb{P}(\overline{\boldsymbol{t}_{ij}} \mid \boldsymbol{t}_i, \boldsymbol{t}_j, \boldsymbol{R}_i)\right) \qquad (2-9)$$

This formulation is the MLE used in PGO as show Carlone et al. (2015). The only last step to obtain a computational useful formula is to pick the probability distribution model for the measurement noise and replace $\mathbb{P}(.)$ by its explicit form. In a more general case, when solving a MLE the most common assumption for the noise is the gaussian distribution which leads to a non-linear least-square estimator:

$$\min_{\{\boldsymbol{T}_i\}} \sum_{(i,j)\in\mathcal{E}} \left\| e(\boldsymbol{T}_{ij}, \boldsymbol{T}_i, T_j) \right\|_2^2 \qquad (2-10)$$

2.2 Handling Outliers

The standard form of the MLE in PGO assumes a nominal distribution of the noise (Gaussian distribution usually) which is a light-tailed noise distribution. This assumption leads to a quadratic form of the MLE objective function, so if some errors are far from the nominal value, they will strongly impact the results. If all

measurements are inliers, then the optimization process will work. But if there is one spurious measurement which move away from the nominal noise, then the algorithm fails, and the computed poses are incorrect estimates. The spurious measurements which are far away from the nominal noise are represented by a heavy-tailed noise distribution which violated the light-tailed noise distribution assumption.

Outliers are measurements between 2 poses with a large error not representative of the reality of our robot movement. For instance, the robot arrives at position T_j and see a blue chair. The front-end detects that a blue chair was seen before and so supposed the place was visited previously by the robot at the corresponding position T_i. It computed the difference of position T_{ij} between i and j supposing the chair is the same one in the two images. Next in the pose graph a loop closure edge is created with the T_{ij} measurements information. However, the chair was a different one in reality and the position T_i and T_j are far from each other's. So, the error $T_j - T_i T_{ij}$ contains in the graph edge will be high compares to the other edge errors. When the back end minimizes the overall error on all edges, if the cost is quadratic this "wrong" edge will heavily impact the results.

Sometimes outliers can come from a sensor failure. But most of the time in PGO, the outlier's creation comes from incorrect data association as the example developed before show. This means supposing the loop closure edges are more likely to be outlier. Since we need loop closure edge in the pose graph to correct the drift error and increasing the connectivity of the graph, we tend to

be laxist on the condition for the loop closure detection and then considers that they will be outlier in the pose graph. Especially since in human made environments it is more likely that places will generate similar visual or information footprint that algorithms will not be able to tell apart. This phenomenon is called perceptual aliasing and is known to create highly correlated loop closure in pose graph as supposed in Lajoie et al. (2019). Techniques for handling outliers can be classified either as rejection or mitigation.

2.2.1 Rejection techniques

Also called removal techniques, they aim at explicitly identify spurious measurements or expressed differently, they try to establish if a given edge is an outlier or an inlier. A first type of approach is maximum consensus where we first used an algorithm such as RANSAC (random sample consensus) or RRR (realizing, reversing and recovering) from Latif et al. (2013) to identify the largest set of edge which are jointly consistent. Then we apply non robust optimization techniques on this pre-computed set. However, this add complexity and is inefficient. The other type of rejection techniques approach is aiming at finding the rejected outlier at the same time as optimization the pose, and so is a robust estimation technique which is better for computation time.

The most well-known algorithm using rejection is switchable constraints (SC) from Niko Sünderhauf et al. (2012). The basic idea is to add a switch variable to all loop closure edge as shown in the figure 2-7. These extra variables decide if edges are use in the optimization process or not. They are optimized at the same time as the pose as they appear in the switch function

$\Psi(s_{ij})$: $\mathbb{R} \rightarrow [0,1]$ which is multiply to the loop closure error in the objective function. There is a, open-source implementation in C++ called VERTIGO which permits to solve pose graph in g2o and gtsam with the switchable constraints method.

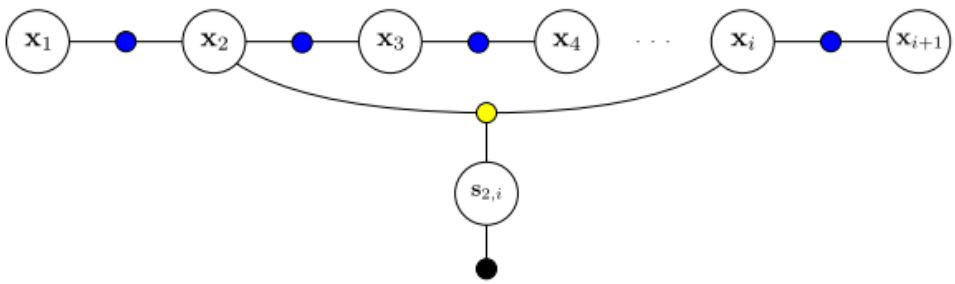

Figure 2-3: Augmented pose graph with switch variables (Niko Sünderhauf et al., 2012)

2.2.2 Mitigation techniques

In opposition with the removal techniques, the mitigation techniques want to keep all the measurements during the optimization. They want to reduce the impact of the outlier without neglecting some measurements. They exist several algorithms taking this approach, the dynamic covariance scaling (DCS) algorithm (Agarwal et al., 2013) is an extension of switchable constraints but instead of introducing new variables, it gives an analytic formula for the weight in front of the loop closure edge. Another approach which uses adaptive coefficient in front of the loop closure error is the expectation-maximization (EM) algorithm from Lee et al. (2013). It supposes the weight to be Cauchy function and first computes their value by minimizing the error (the

weights are the only variables), then using the computed weight it optimizes the pose graph. The Max-mixtures algorithm (Max-Mix) from Olson et al. (2013) supposes the noise distribution to be a mixture of gaussians instead of a single gaussians.

But the mainstream approach for robust mitigation of outlier is the use of M-estimators as explain in Bosse et al. (2016). It is basically an extended MLE, where instead of solving the classical non-linear least-squares problem over the residuals, they add a loss function over the squared error:

$$\min_{\theta} \sum \boldsymbol{\rho}\left(||e(\theta)||^2 \right) \qquad (2-11)$$

The goal is to use a loss function $\boldsymbol{\rho}$ which reduce the influence of large error in the sum, for instance Huber or Geman-McClure (GM) function. This can be solve using Iterative Re-weighted non-linear least-squares. The DCS and expectation-maximization algorithms show similar behavior than M-estimator for specific loss function. Graduated non convexity technics are based on a M-estimator where the loss function convexity can be control by a parameter.

2.3 Convex optimization

All the theory from this part is explained in greater details in Boyd et al. (2004). Mathematical optimization problems have the following standard form with an optimization variable x, an objective function f_0, inequality constraint functions $f_i(x)$ and equality constraint functions $h_i(x)$.

$$\begin{aligned} &\textit{minimize} && f_0(x) \\ &\textit{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\ & && h_i(x) = 0, \quad i = 1, \dots, p \end{aligned} \tag{2-12}$$

Optimization problem can be classified in function of the forms of the objective and constraint functions. For instances, when all the functions are linear, we call the optimization problem a linear program. But PGO is known to be a non-linear optimization problem, which means that the objective and/or the constraints functions are non-linear. Most family of optimization problem have been studied and solution methods (algorithm) with their software implementation exists. But nonlinear optimization stays a challenging problem as we need to compromise between accuracy and time. The local optimization approach seeks to find a local minimum usually using derivative of the objective function. The most common algorithms are Gauss-Newton (GN) or Levenberg-Marquardt (LM) and are implemented in solver such as g2o or ceres. It is a fast but need initialization and no guarantee of obtaining the global minima. Whereas global optimization method finds the global minimum but doesn't scale well.

A convex optimization problem has only one global minimum, so we can apply to it a fast local optimization algorithm without being worried about finding a local minimum (non-global minima). Given this nice property, convex optimization problem and their use for non-convex optimization problem have study and we will present the theory here.

2.3.1 Convex sets and function

A convex set is a set which contains the line segment between any two points in the set as illustrated in figure 2−3. The hexagon on the right is a convex set. But the kidney shaped set in the middle is not convex as part of a line segment is out the set. Similarly, the square shape on the right is not a convex set as some boundary points are not contains in the set so the line segment overlapping with the boundary will not be contains in this set.

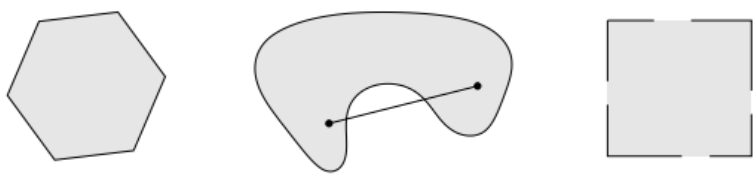

Figure 2−4 Examples of simple convex set (Left hexagon) and nonconvex sets (Middle kidney shape, Right square) (Boyd et al., 2004)

The mathematical definition of a convex set is just a formalization of this idea, for all point $x_1, x_2 \in C$, the line segment $\theta x_1 + (1-\theta)x_2$ is in the set:

$$x_1, x_2 \in C, \quad 0 \le \theta \le 1 \quad => \quad \theta x_1 + (1-\theta)x_2 \in C \qquad (2-13)$$

A convex function is a function define on a convex set which has all line segment between any two points of the function above the function as show in figure 2−4.

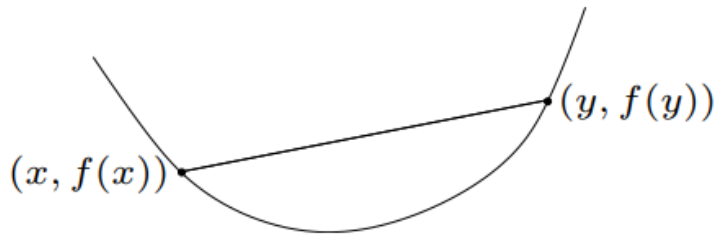

Figure 2−5 Graph of a convex function (Boyd et al., 2004)

The formal definition is: a function $f: \mathbb{R}^n \to \mathbb{R}$ is convex if $\boldsymbol{dom}\, f$ is a convex set and if for all $x, y \in \boldsymbol{dom}\, f$, and θ with $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) \qquad (2-14)$$

2.3.2 Convex, polynomial, and semidefinite optimization problem

A convex optimization problem is a problem of the form:

$$\begin{aligned} &\textit{minimize} && f_0(x) \\ &\textit{subject to} && f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & && a_i^T x = b_i, \quad i = 1, \ldots, p \end{aligned} \qquad (2-15)$$

where $f_0, \ldots, f_m$ are convex functions. So compared to the standard form of optimization problem, the additional constraints are the convexity of the objective and inequality constraint functions, and the equality constraint functions must be affine.

A polynomial optimization problem (POP) is a problem of the form:

$$\begin{aligned} &\textit{minimize} && f_0(x) \\ &\textit{subject to} && f_i(x) \le 0, \quad i = 1, \dots, m \\ & && h_i(x) = 0, \quad i = 1, \dots, p \end{aligned} \tag{2-16}$$

where $f_O, \dots, f_m, h_0, \dots, h_p$ are real polynomials functions. Many fundamental problems in geometric perception such as PGO can be reformulated as POP. But POP are non-deterministic polynomial-time hard problems (NP-hard), which means that we cannot prove that a polynomial time solution exists. Most robust estimators are hard to compute as show in Bernholt, T. (2006).

A semidefinite programming problem (SDP) is a problem of the form:

$$\begin{aligned} &\textit{minimize} && tr(CX) \\ &\textit{subject to} && tr(A_i X) = b_i, \quad i = 1, \dots, p \\ & && X \succcurlyeq 0 \end{aligned} \tag{2-17}$$

Where the variable is a real symmetric matrix ($X \in \boldsymbol{S}^n$), and the matrices $C, A_1, \dots, A_p$ are also real symmetric matrix. The condition $X \succcurlyeq 0$ means that we search a positive semidefinite solution matrix. SDP program are convex optimization problem which can be solved efficiently for small and medium size problems with a global minimum.

2.3.3 Duality

For any optimization problem in standard form (equation 2-12), we can define the Lagrangian dual function g using the Lagrangian multiplier λ_i, ν_i.

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} \left(f_o(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right) \qquad (2-18)$$

The Lagrangian dual function is concave and can have infinite value in some points. Using this function, we can write the dual problem of the primal initial optimization problem.

$$\begin{aligned} &maximize \quad g(\lambda, \nu) \\ &subject\ to \quad \lambda \succcurlyeq 0 \end{aligned} \qquad (2-19)$$

The optimal value for the primal problem is noted p^* and the one for the dual problem d^*. In function of the problem type, the duality between the 2 problems can be a strong ($d^* = p^*$), or weak ($d^* \leq p^*$). Weak duality always holds but strong duality does not hold in general. But for convex problems strong duality usually holds, and conditions that guarantee strong duality in convex problem are called constraint qualifications.

One example of constraint qualifications largely used in practice is Slater's constraint qualification defined as:
Strong duality holds for a convex problem if any of the following equivalent conditions are met:

1. The problem is strictly feasible.
2. There exists a solution x^* that satisfies all the constraints

and fulfills the nonlinear constraints with strict inequalities

3. $\exists x \in \boldsymbol{int}\,\mathcal{D}: \quad f_i(x) < 0, \quad i = 1, \dots, m, \quad a_i^T x = b_i, \quad i = 1, \dots, p$

We can also define the Karush−Kuhn−Tucker (KKT) conditions for a problem with differentiable constraints and objective functions (equation 2−12 with additional differentiability conditions). The KKT conditions are important as we know that if strong duality holds and x, λ, ν are optimal, then they must satisfy the KKT conditions. Moreover, if $\tilde{x}, \tilde{\lambda}, \tilde{\nu}$ satisfy the KKT for a convex problem, then they are optimal. So, we can solve the KKT conditions problem and obtain optimal solution for the initial problem. Not all convex problem admits a solution to the KKT conditions, but if a convex problem satisfy Slater's condition, then it is sure that a solution to the KKT conditions problem exists.

KKT conditions are:

1. Primal constraints : $f_i(x) \leq 0, \;\; i = 1, \dots, m, \quad h(i) = 0, \quad i = 1, \dots, p$
2. Dual constraints : $\lambda \geqslant 0$
3. Complementary slackness: $\lambda_i f_i(x) = 0, \;\; i = 1, \dots, m$
4. Gradient of Lagrangian with respect to x vanishes:

$$\nabla f_o(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + \sum_{i=1}^{p} \nu_i \nabla h_i(x) \; = 0$$

2.3.4 Algorithms for resolution

To solve convex optimization problem, there are different approach depending on the complexity of the problem, or more

precisely depending on the constraints of the problem. The idea is that there is a hierarchy in the convex optimization algorithms. The simplest problem are the unconstrained optimization problems which can be solve quickly. Then we have equality constrained problem which need extra steps. And finally convex optimization problem with inequality constrained are the hardest to solve.

If the convex problem is unconstrained then it can be solve using a descent method. This method computes iteratively a solution $x^{k+1} = x^k + t^k \Delta x^k$ which is always at a lower point that the previous solution $f(x^{k+1}) < f(x^k)$. The principle is simple and the figure 2−5 shows the algorithm for a general descent method. There are 2 main steps: determining the descent direction Δx and choosing the step size t.

Algorithm 9.1 *General descent method.*

given a starting point $x \in$ **dom** f.
repeat
1. Determine a descent direction Δx.
2. *Line search.* Choose a step size $t > 0$.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Figure 2−6 Descent method algorithm (Boyd et al., 2004)

Different algorithms exist for the line search step (2) such as exact line search where we choose t to minimize the objective function or backtracking line search where we approximately minimize the objective function. Several methods have been proposed for the descent direction choice step (1) and recapitulated in table 2−2.

Table 2-2 Descend direction method

Item	Gradient descend	Steepest descend	Newton's
Search direction Δx	$-\nabla f(x)$	$\left\|\left\|\nabla f(x)\right\|\right\|_* * \Delta x_{nsd}$ $\Delta x_{nsd} = argmin\{\nabla f(x)^T v \mid \left\|\left\|v\right\|\right\| = 1\}$	$-\nabla^2 f(x)^{-1}\nabla f(x)$
Stopping criterion	$\left\|\left\|\nabla f(x)\right\|\right\|_2 \leq \eta$	$\left\|\left\|\nabla f(x)\right\|\right\|_2 \leq \eta$	$\nabla f(x)^T \nabla^2 f(x)^{-1}\nabla f(x) \leq 2\epsilon$
Convergence	Approximately linear	Approximately linear	Rapid and quadratic near solution
Notes	Simplest method however the convergence rate is too slow for most applications.	The convergence rate depends on the norm chosen: good as we can find a norm to improve the speed, but bad as we need to find such a norm from a large panel.	Affine invariant, scale well with problem size and the parameter's choice does not impact the convergence performance but storing the Hessian is costly.

When the problem has only equality constraint, we can take different approaches: use tricks to reduce it in an unconstrained equivalent problem or solve the dual problem to recover the solution. But if we want to exploit the problem structure (sparsity or others), we need to directly solve the equality constrained problem. And to do so, we can use an extension of the Newton's method. The objective function is replaced by its second-order Taylor approximation near x which made the problem a convex quadratic minimization problem solvable using the KKT matrix (if nonsingular). The Newton step Δx_{nt} in this case is compute at the same time as the associated optimal dual variable w by resolving the system:

$$\begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{nt} \\ w \end{bmatrix} = \begin{bmatrix} -\nabla \mathrm{f(x)} \\ 0 \end{bmatrix} \qquad (2-20)$$

The newton decrement used as stopping criterion is the same as in the unconstrained case. The convergence characteristics are the same than the ones for Newton's unconstrained method, but we additionally have condition on the KKT matrix for convergence. It is also important to said that basic Newton's methods are feasible descend methods (the initial points need to be feasible). Algorithms exist to make the method works even with initial points, and iterates, that are not feasible.

For convex problems with inequalities constraints, the interior-point methods are commonly used. They apply Newton's method to a sequence of equality constrained problems which approximate the inequalities constrained problem. Algorithms exist such that the barrier method or primal-dual interior point methods. The primal-dual interior-point methods are the more efficient in

many cases so we will briefly detail the basic one which algorithm is show in Figure 2−6.

Algorithm 11.2 *Primal-dual interior-point method.*

given x that satisfies $f_1(x) < 0, \ldots, f_m(x) < 0$, $\lambda \succ 0$, $\mu > 1$, $\epsilon_{\text{feas}} > 0$, $\epsilon > 0$.
repeat
1. *Determine* t. Set $t := \mu m/\hat{\eta}$.
2. Compute primal-dual search direction Δy_{pd}.
3. *Line search and update.*
   Determine step length $s > 0$ and set $y := y + s\Delta y_{\text{pd}}$.

until $\|r_{\text{pri}}\|_2 \leq \epsilon_{\text{feas}}$, $\|r_{\text{dual}}\|_2 \leq \epsilon_{\text{feas}}$, and $\hat{\eta} \leq \epsilon$.

Figure 2−7 Basic Primal−dual interior−point algorithm (Boyd et al., 2004)

First, the initial complex problem is rewritten to make the inequality constraints implicit in the objective function. Then the objective function contains the indicator function which is not differentiable and so is approximate by a similar differentiable function. The quality of the estimate compares to the initial problem depends on the extra parameters t introduce by the approximation. In the algorithm t is chosen in function of the current surrogate duality gap $\hat{\eta}(x,\lambda) = -f(x)^T\lambda$. From this approximated problem, the modified KKT conditions can be found:

$$r_t(x,\lambda,v) = 0 = \begin{bmatrix} \nabla f_0(x) + Df(x)^T\lambda + A^Tv \\ -\boldsymbol{diag}(\lambda)f(x) - \frac{1}{t}\mathbf{1} \\ Ax - b \end{bmatrix} \qquad (2-21)$$

And in step 2, this nonlinear set of equations is solved for the fixed t parameters using Newton step. The primal−dual search direction $\Delta y_{pd} = (\Delta x_{pd}, \Delta\lambda_{pd}, \Delta v_{pd})$ is the solution of the following matrix

system:

$$\begin{bmatrix} \nabla^2 f_0(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 f_i(x) & Df(x)^T & A^T \\ -\boldsymbol{diag}(\lambda) Df(x) & -\boldsymbol{diag}(f(x)) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} \nabla f_0(x) + Df(x)^T \lambda + A^T \nu \\ -\boldsymbol{diag}(\lambda) f(x) - \frac{1}{t}\mathbf{1} \\ Ax - b \end{bmatrix} \qquad (2-22)$$

The stopping criteria are when $\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}$ are feasible for the problem (in the tolerance range define) and the surrogate gap is under the wanted tolerance.

2.3.5 Nonconvex problem and convex relaxation

Convex optimization is a powerful tool. The hard part of the job is not to resolve the optimization problem but to formulate the problem as a convex one. It is sometimes possible to find an equivalent convex problem for the initial problem but not all problem can be reduced to a convex problem. In this case, we can still use convex optimization in different ways to help solving the nonconvex problem. One idea is to initialize a local optimization process. The nonconvex problem is approximate to a convex one which is solve easily and without initialization. The solution, which is more likely to be close to the actual global minima, is used as a starting point for the local optimization method and then prevent the optimization to find a local minimum. Graduated non convexity (GNC) problem used this idea as in the article from McGann et al. (2022). By controlling the convexity of the problem with an extra variable, they solve step by step a less convex problem. Sometimes it is also possible to derive the dual problem and solve it as in Carlone et al. (2016) or using it to verify the quality of the solution as in Carlone et al. (2015).

Another used of convex optimization for nonconvex problem is relaxation and lower bound computation. First, if we have a nonconvex problem, we can detail where the nonconvexity lies, i.e in the objective function or in the constraint functions. Then we can either find convex relaxation functions of the initial nonconvex functions or just drop nonconvex constraint functions. If we have a nonconvex function $f : S \to \mathbb{R}$ where $S \subset \mathbb{R}^n$ is a nonempty convex set, then a convex function $h : S \to \mathbb{R}$ is a convex relaxation of f if $h(x) \leq f(x) \quad \forall x \in S$. The figure 2-6 illustrates the definition.

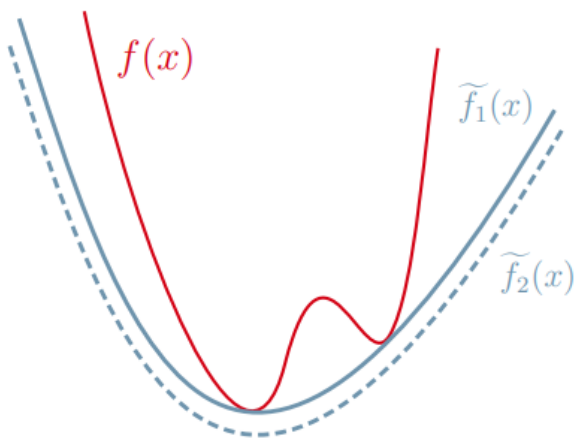

Figure 2-8 Convex relaxation concept of a function (Erik F. Alvarez; 2019)

Using this we can approximate a nonconvex problem into a convex problem, but there is no equivalence between them, i.e. the solution of one problem is not ensured to be the same as the solution of the other problem. However, since one problem is a relaxation of the other, there is a condition between their minima which provides a lower bound for the optimal value of the nonconvex problem.

$$h(x^*) \leq f(x^*) \qquad (2-23)$$

Some examples of relaxation are Lagrangian relaxation (solve the Lagrangian dual), L1 relaxation (Carlone et al., 2014), Lasserre hierarchy Relaxation (use the moment matrix as in Yang et al. (2023)) and SDP relaxation (the final problem is an SDP problem).

3 State of the art

The problem of robust pose graph optimization has already been studied and several approaches have been proposed with different advantages and drawbacks. 5 algorithms are described here as they represent the different trend on robust pose graph optimization. We also compared them to the thesis content to emphasis the contribution of our work to the state of the art.

3.1 SE-sync

An algorithm to solve synchronization over the special Euclidean group using convex relaxation was published by Rosen et al. in 2017. It efficiently recovers certifiably globally optimal solutions for problem when the noise is non-adversarial. First, the MLE problem is reformulate into a semidefinite convex relaxation which is proven to be exact when the noise corrupting the data is under a certain threshold. This reformulation also provides an a posteriori condition to check the optimality of our obtained pose solution. Additionally, they use the low-rank, geometric and graph-theoretic structure of the semidefinite relaxation to rewrite the problem on a Riemannian manifold. They utilize this to form a Riemannian truncated-Newton trust-region method which can solve large-scale problem efficiently. The last step is a simple rounding procedure to obtain the final solution of the synchronization problem.

Algorithm 3 The SE-Sync algorithm

Input: An initial point $Y \in \mathrm{St}(d, r_0)^n$, $r_0 \geq d+1$.

Output: A feasible estimate $\hat{x} \in \mathrm{SE}(d)^n$ for the maximum-likelihood estimation Problem 1 and the lower bound p^*_{SDP} for Problem 1's optimal value.

function SE-Sync(Y)
  Set $Y^* \leftarrow$ RiemannianStaircase(Y). ▷ Solve Problems 7 & 9
  Set $p^*_{\mathrm{SDP}} \leftarrow F(\tilde{Q}Y^{*\mathsf{T}}Y^*)$. ▷ $Z^* = Y^{*\mathsf{T}}Y^*$
  Set $\hat{R} \leftarrow$ RoundSolution(Y^*).
  Recover the optimal translational estimates $\hat{t}$ corresponding to $\hat{R}$ via (21).
  Set $\hat{x} \leftarrow (\hat{t}, \hat{R})$.
  return $\{\hat{x}, p^*_{\mathrm{SDP}}\}$
end function

Figure 3-1 The SE-Sync algorithm (David M. Rosen et al., 2017)

This algorithm has 2 main advantages: it obtains an exact global optimal solution with a condition to check the optimality and it can be use for large-scale instance problem. They tested it on different synthetic datasets and large-scale real-world examples. SE-Sync can recover optimal solution and an order of magnitude faster than Gauss-Newton based approach. The main drawback is that it is design to be operating when the noise is under a certain threshold, so it does not contain any mechanism against outliers.

3.2 Convex relaxation for 2D robust PGO

In 2018, Carlone et al. developed a robust estimator to solve a 2D PGO problem where measurement with heavy noise can appears. To do so, they derive a convex relation process of the maximum likelihood initial problem for different PGO formulation setups. This leads to 3 robust estimator formulations which are presented: the first one is a least unsquared deviation estimator (L2 loss function), the second is a least absolute deviation estimator (L1 loss function), and the last is a Huber estimator (Huber loss function). In addition, the difference between a 1-stage approach or 2-stage approach for the resolution is tested. A 1-stage approach

means solving the optimization over the rotation and translation poses variables at the same time. Whereas a 2−stage approach decouple the rotation and translation problem. Only the rotational non−convex subproblem is relaxed and solve, then the rotation matrix is used in the translation subproblem to solve the already convex problem.

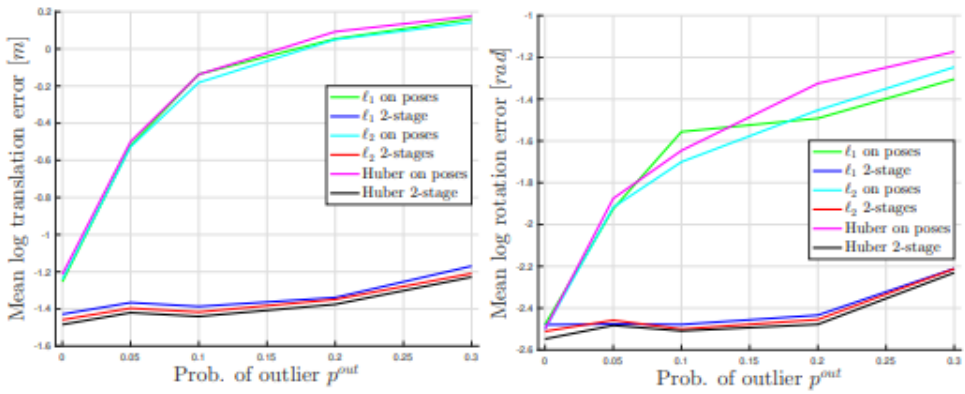

Figure 3−2 Estimation errors for the 6 approaches for Geometric random graphs (Carlone et al., 2018)

The advantage of using convex relaxation is first the fact that they don't need an initial guess for resolution. And second, they can obtain guarantee condition on the suboptimality of the rounded solution by checking the rank of the relaxed solution matrix X and if the first n 2×2 blocks of its rank−2 approximation matrix are in $SO(2)$. They tested and compared the different formulation on synthetic pose graph dataset corrupted with an adaptive percentage of outliers with main results: The 2−stage approach outperformed the 1−stage for all dataset and percentage of outliers and shows robustness for high level of outliers when the graph is highly connected. The drawbacks for their implementation of the method

are the computation time which make it harder to use for large pose graph instance.

3.3 DC-GM

Pierre-Yves Lajoie et al. developed a discrete-continuous graphical model to represent the robot position and measurement edge as show in figure 3-3 in 2019. The first contribution of their model is the distinction between inliers and outliers for the edges between poses by using additional discrete variables. Additionally, the edges between theses discrete variables represent the correlation between measurements. Adding this information to the graph is meaningful as usually the outliers are highly correlated. So, if we detected one outlier, the correlated measurements with this outlier have a high chance to be outlier too. The second contribution is the use of convex relaxation for the MLE to optimize without initialization of the poses and with some condition on the sub-optimality of the results.

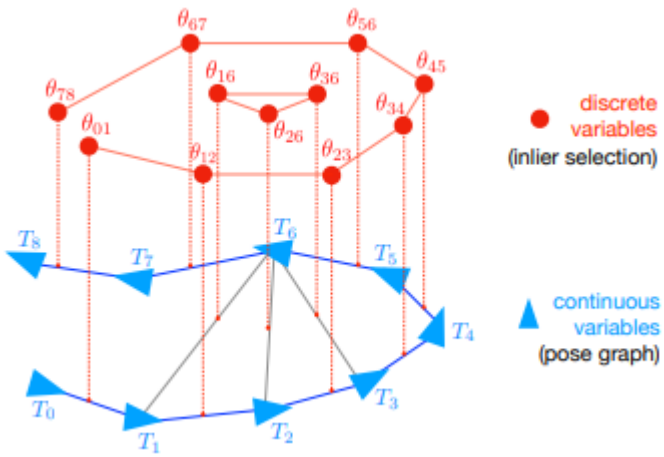

Figure 3-3 Discrete-continuous graphical model (Pierre-Yves

Lajoie et al., 2019)

The use of binary variables to robustify the pose graph is equivalent to use a truncated LS cost on the measurement edges. The difference is that the truncated LS is non-convex and non-differentiable, whereas minimizing over a binary variable do not add complexity. The DC-GM approach was tested on synthetic and real-world 2D datasets and showed good results against the state of art approach with no initialization. An advantage of this formulation is the intuitive way of tuning the parameter for differentiating the outliers and the inliers (maximum admissible residual). The main limitation of their paper is the computation time, which is the reason why they tested only on 2D data. The reason is the use of MATLAB solver. Another possible improvement is to make it in an online process.

3.4 AEROS

In 2022, Milad Ramezani et al. proposed an algorithm with an adaptive robust loss function against the outliers. First, they used on the MLE the Barron loss function which is an adaptive cost function with an additional parameter. The parameter chooses the shape of the loss function as show on the figure 3-4. The novelty of their approach is that they reformulate the objective function, so they optimize the parameter at the same time as the pose, so the curve is closely fitting the distribution. Moreover, the final form of they function use standard gaussian factors so they can of any classical incremental estimation approaches (for instance iSAM).

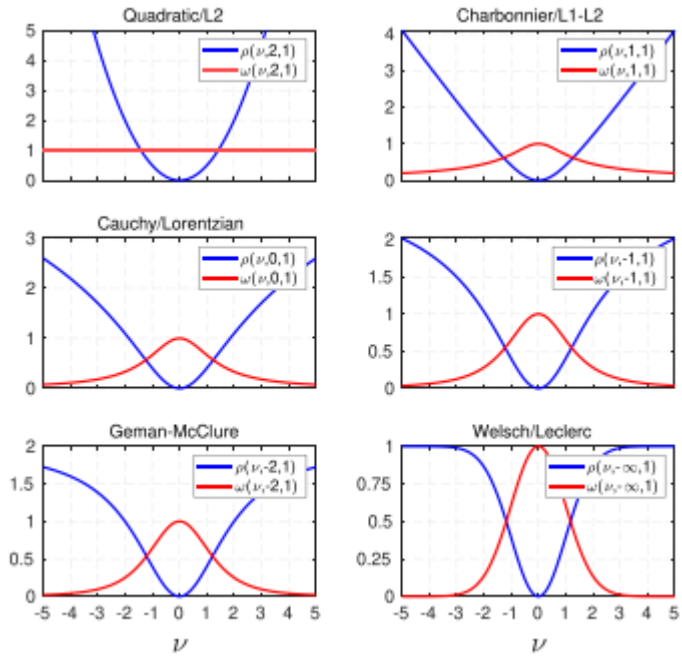

Figure 3-4 Adaptive loss kernel and its weight (Milad Ramezani et al., 2022)

The main advantage of these method is the adaptivity of the loss function with only adding one latent parameter. The testing was done on 2D synthetic and real-world datasets where they add outliers to test the robustness. And they also test on one large 3D real-world dataset from LiDAR data preprocessed with Iterative Closest Point. The result demonstrates that their method is competitive against other methods and the parameter is changed in function of the data used. The optimization of the extra parameter adds time to process, and the algorithm is a batch process. This algorithm also needs initialization of the poses.

3.5 riSAM

An incremental solver for robust PGO was developed by Daniel Mc Gann et al. in 2022. It is based on the idea of graduated

non convexity (GNC), which is basically that we first convexify the problem and then solve a series of progressively less convex problems as illustrated on the figure 3-5. To make GNC efficient and incremental, they only compute a single non-linear update step instead of the full optimization. Additionally, they propose a new kernel for online efficiency: the scale invariant graduated (SIG) which admits a known constant number of GNC iterations. The last contribution is the incrementalization of the Dog-Leg line search which is a trust region optimization algorithm.

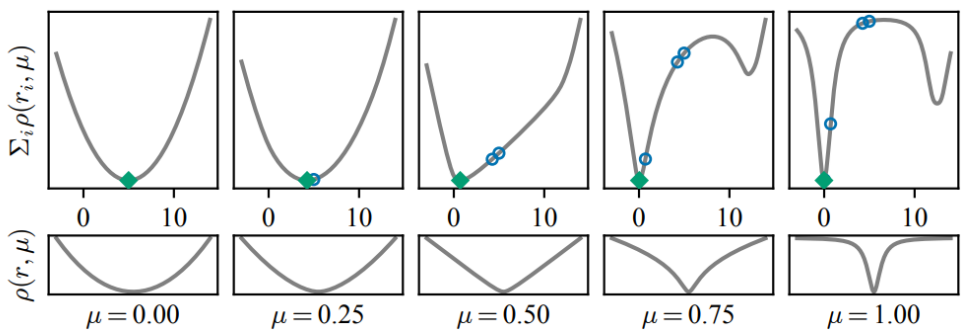

Figure 3-5 Example of GNC solving a linear problem with outliers (Daniel McGann et al., 2022)

riSAM is tested on Synthetic (2D and 3D) and real-world (2D) datasets. It shows same precision as its batch variants. And more importantly it achieves online efficiency on large scale problem with robustness to outlier and initialization. But it still needs initialization, and it is dependent on the user defined parameters.

3.6 Comparison of related works with this thesis

This master thesis aims at studying robust pose graph optimization in 3D only when using a convex relation of the initial problem and loss function to mitigate outlier impact. In Luca Carlone et al. work they study the 2D case of the convex relaxation with no separation of the edge type. The SE−Sync algorithm is not robust against outliers and the DC−GM algorithm focus on one loss function where outliers are correlated. For the AEROS algorithm, the loss function is adaptive as an extra optimized parameter choose the shape of the function, however it does not use convex relaxation so need initialization. The riSAM algorithm is achieving real−time computation with a custom−made loss function but they use GNC (convex only for initialization of the optimization) and so does not have certifiability on the correctness of the solution. We want to focus on using an algorithm with convex relaxation to not need initialization and to have performance guarantees on our algorithm (if the solution is wrong, the algorithm can detect the failure). Starting from this setup, we want to study different loss function for robustness.

Table 3-1 Comparison of related work (robust pose graph optimization approach)

Item	Related work					This thesis
	1. Se-sync David M. Rosen et al. 2017	2. 2D_relax Luca Carlone et al. 2018	3. DC-GM Pierre-Yves Lajoie et al. 2019	4. AEROS Milad Ramezani et al. 2022	5. RiSAM Daniel McGann et al. 2022	
Problem definition						
Input	$T_{ij} \in SE(d)$	$R_{ij} \in SO(2)$, $t_{ij} \in \mathbb{R}^2$ with outliers	$T_{ij} \in SE(d)$ with correlated outliers	$T_{ij} \in SE(d)$ with outliers	$T_{ij} \in SE(d)$ with outliers	$R_{ij} \in SO(3)$, $t_{ij} \in \mathbb{R}^3$ with outliers
Algorithm	Synchronization over SE(d)	PGO	PGO	PGO	SLAM	PGO
Output	$T_i \in SE(d)$	$R_i \in SO(2)$, $t_i \in \mathbb{R}^2$	$T_i \in SE(d)$	$T_i \in SE(d)$	$T_i \in SE(d)$	$R_i \in SO(3)$, $t_i \in \mathbb{R}^3$
Dimension	d	2	d	d	d	3

MLE formulation and resolution						
Processing	Batch	Batch	Batch	Batch	Online	Batch
Nb stages	2−stages	1−stage and 2−stages	1−stage	1−stage	1−stage	2−stages
Edges separation	Not considered	Not considered	Considered	Considered	Considered	Considered
Outlier handling	Not considered	Loss function: Huber, L1 and L2	Loss function TLS & outlier c orrelation	Loss function Barron	Loss function SIG	Loss function: Huber, L1 and L2
Optimization approach	SDP relaxation	SDP relaxation	SDP relaxation	Iterative approach	Graduated non convexity	SDP relaxation

Optimization algorithm	Riemannian staircase	Interior−point in cvx solver	Interior−point in cvx solver	iSAM2	Incremental Powell's Dog −Leg	Interior−point in cvx solver
Initialization	Yes, to speed c omputation	No	No	yes	yes	No
Certifiability contract	Yes	Yes	Yes	No	No	Yes

Table 3-2 Comparison of related work (Simulation setup)

Item	Related work					This thesis
	1. Se-sync David M. Rosen et al. 2017	2. 2D_relax Luca Carlone et al. 2018	3. DC-GM Pierre-Yves Lajoie et al. 2019	4. AEROS Milad Ramezani et al. 2022	5. RiSAM Daniel McGann et al. 2022	
Implementation						
Language	MATLAB	MATLAB	MATLAB	cpp	cpp	MATLAB
Solver	Truncated-Newton RTR	cvx	cvx	iSAM from GTSAM	GTSAM	Cvx
Monte Carlo run	50	30	5/10	10	10/50	10
Compared methods	GN	G2o, and DCS	Vertigo, RRR, and DCS	SC, DCS, GNC, and GM	GNC, GM, Max-Mix, Huber, and SC	?

Datasets						
Synthetic 2D	Not tested	Erdos–Rényi and geometric random graph, and Manhattan 3500 world	Grid	Manhattan3500, City10k	Random grid, Manhattan, and City10k	Maybe
Synthetic 3D	Cube, Sphere2500, torus, and grid	Not tested	Not tested	Sphere2500	Sphere	Random and grid graph, Manhattan world Cube
Real 2D	Not tested	Not tested	CSAIL, FRO79 and FRH	CSAIL, INTEL	CSAIL, and Intel	Not tested
Real 3D	Garage, cubicle and rim	Not tested	Not tested	Nezer College	Not tested	Partial garage

4 Proposed PGO algorithm

The proposed PGO algorithm idea was showed in figure 1-1. It recapitulated the main part briefly, and now we will detail more the different steps. The figure 4-1 shows the global flow chart of our work. In our algorithm, we want to emphasis on the convex relaxation and the loss function choice for robustness. These specific parts will be details latter in the chapter 5 and 6. In this chapter, we will present first the MLE formulation derivation and the 2-steps approach for the PGO resolution.

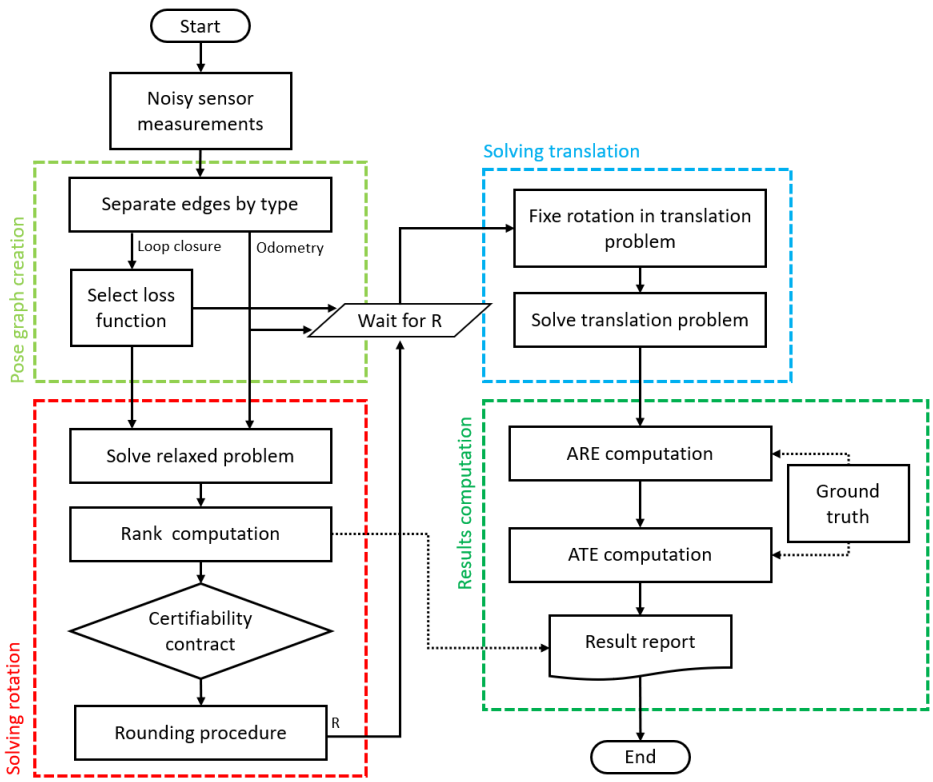

Figure 4-1 Flow chart of our proposed PGO

4.1 MLE formulation

As said before, PGO is the process of estimate a set of n poses (rotation and translation) in 3D from m pairwise relative pose measurements. To do so, we construct a pose graph where:

1) Each node represents a pose to estimate and is noted $\boldsymbol{T_i} \triangleq [\boldsymbol{R_i}\ \boldsymbol{t_i}]$ which comprises a translation vector $\boldsymbol{t_i} \in \mathbb{R}^3$ and a rotation matrix $\boldsymbol{R_i} \in \boldsymbol{SO}(3)$.

2) Each edge represents a relative pose measurement and is noted $[\overline{\boldsymbol{R_{ij}}}\ \overline{\boldsymbol{t_{ij}}}]$ which comprises a translation vector $\overline{\boldsymbol{t_{ij}}} \in \mathbb{R}^3$ and a rotation matrix $\overline{\boldsymbol{R_{ij}}} \in \boldsymbol{SO}(3)$. They describe a noisy measurement of the relative pose between $\boldsymbol{T_i}$ and $\boldsymbol{T_j}$.

Starting from this, we need to derive the MLE problem to define in the optimization solver. We already explained how the MLE is on the form of the sum of the log of the probability distribution of the noise as in equation 2−9.

4.1.1 Measurements noise model choice

The inliers measurement between 2 poses nodes is model as:

$$\overline{\boldsymbol{t_{ij}}} = \boldsymbol{R_i^T}(\boldsymbol{t_j} - \boldsymbol{t_i}) + \boldsymbol{t_{ij}^{\epsilon}}, \qquad \overline{\boldsymbol{R_{ij}}} = \boldsymbol{R_i^T R_j R_{ij}^{\epsilon}} \qquad (4-1)$$

Where the translation noise model is a multivariate gaussian distribution of mean $\boldsymbol{0_3}$ and covariance matrix $\frac{1}{w_t}\boldsymbol{I_3}$. The rotation noise model is an isotropic Langevin distribution on $\boldsymbol{SO}(3)$ with mode $\boldsymbol{I_3}$ and concentration parameter $\boldsymbol{w_R}$, as in Carlone et al. (2018).

The first term is about the error on the translations part of the pose and the distribution chosen is a multivariate gaussian. So when replacing the probability density function of $\boldsymbol{t_{ij}^{\epsilon} = \overline{t_{ij}} - R_i^T(t_j - t_i) \sim \mathcal{N}\left(O_3, \frac{1}{w_t} I_3\right)}$ in the formula, we obtain the following form:

$$-\ln\left(\mathbb{P}\left(\overline{t_{ij}} \mid t_i, t_j, R_i\right)\right)$$

$$\Leftrightarrow \quad -\ln\left(\frac{1}{\sqrt{\left(\frac{2\pi}{w_t}\right)^3}} * \exp\left(-\frac{1}{2}\left(t_{ij}^{\epsilon}\right)^T w_t\left(t_{ij}^{\epsilon}\right)\right)\right)$$

$$\Leftrightarrow \quad \ln(A) - ln\left(\exp\left(-\frac{1}{2}\left(t_{ij}^{\epsilon}\right)^T w_t\left(t_{ij}^{\epsilon}\right)\right)\right)$$

$$\Leftrightarrow \quad \ln(A) + \frac{1}{2}\left(t_{ij}^{\epsilon}\right)^T w_t\left(t_{ij}^{\epsilon}\right)$$

$$\Leftrightarrow \quad \ln(A) + \frac{w_t}{2}\left(\overline{t_{ij}} - R_i^T(t_j - t_i)\right)^T \left(\overline{t_{ij}} - R_i^T(t_j - t_i)\right)$$

$$\Leftrightarrow \quad \ln(A) + \frac{w_t}{2}\left|\left|\overline{t_{ij}} - R_i^T(t_j - t_i)\right|\right|_2^2 \qquad (4-2)$$

For the second term, it represents the error on the rotation part of the pose. Instead of choosing the classical gaussian distribution, we choose the isotropic Langevin distribution (or Von Mises-Fisher). This distribution is directly defined on the special orthogonal group, so it is easier to analyze and leads to simpler estimator form. In the probability density function the normalization term $c_3(\kappa)$ depends in the concentration parameter only and can be derived using modified Bessel function $c_3(\kappa) = exp(\kappa)\left(I_0(2\kappa) - I_1(2\kappa)\right)$. We can think about the concentration parameter κ in terms of information content, the higher it is the more concentrated around the mean is the rotation pool. We can then replace the probability

density function of $\boldsymbol{R_{ij}^{\epsilon} = R_j^{-1} R_i \overline{R_{ij}} \sim Langevin(I_3, w_R)}$ in the second term of the formula:

$$
\begin{aligned}
& -\ln\left(\mathbb{P}(\overline{\boldsymbol{R}_{ij}} \mid \boldsymbol{R}_i, \boldsymbol{R}_j)\right) \\
\Leftrightarrow \quad & -\ln\left(\frac{1}{c_d(w_R)} \exp\left(w_R\, tr\left(\boldsymbol{R_{ij}^{\epsilon}}\right)\right)\right) \\
\Leftrightarrow \quad & \ln(B) - ln\left(\exp\left(w_R tr\left(\boldsymbol{R_{ij}^{\epsilon}}\right)\right)\right) \\
\Leftrightarrow \quad & \ln(B) - w_R tr\left(\boldsymbol{R_{ij}^{\epsilon}}\right) \\
\Leftrightarrow \quad & \ln(B) - w_R tr\left(\boldsymbol{R_j^{-1} R_i \overline{R_{ij}}}\right) \\
\Leftrightarrow \quad & \ln(B) - w_R tr\left(\boldsymbol{R_j^{-1} R_i \overline{R_{ij}}}\right) \qquad (4-3)
\end{aligned}
$$

Replacing the equations 4−2 and 4−3 in the objective function, the constant terms $\ln(A)$ and $\ln(B)$ can be suppress as they do no impact the minimization results. So, the intermediary form of the MLE for inliers is:

$$
\min_{\substack{\boldsymbol{t_i} \in \mathbb{R}^3, \\ \boldsymbol{R_i} \in SO(3)}} \sum_{(i,j)\in\mathcal{E}} \frac{w_t}{2} ||\overline{\boldsymbol{t_{ij}}} - \boldsymbol{R_i^T}(\boldsymbol{t_j} - \boldsymbol{t_i})||_2^2 - w_R tr\left(\boldsymbol{R_j^{-1} R_i \overline{R_{ij}}}\right) \qquad (4-4)
$$

Additionally, we can reform the *trace* term using Frobenius norm as we know $\boldsymbol{tr\left(\left(R_i^{-1} R_j\right)^{-1} \overline{R_{ij}}\right) = 3 - \frac{1}{2} ||R_i^{-1} R_j - \overline{R_{ij}}||_F^2}$ (for the formula to be true, we need the matrix to be orthogonal and all rotation matrix are in $SO(3) \subset O(3)$). We also then use the fact that $\boldsymbol{R_i^{-1} = R_i^T}$ and that the L_2 norm and the Frobenius norm are invariant with respect to orthogonal multiplication $\boldsymbol{\left|\left|R_i x\right|\right|_2 = \left|\left|x\right|\right|_2}$. The L_2 norm is even in the translation part $\boldsymbol{\left|\left|x\right|\right|_2^2 = \left|\left|-x\right|\right|_2^2}$ We finally obtain the final equivalent form of the MLE for inliers as equation 4−5 which is a

non-robust one also used in SE-sync by Rosen et al. (2017).

$$
\begin{aligned}
&\min_{\substack{\boldsymbol{t}_i\in\mathbb{R}^3,\\ \boldsymbol{R}_i\in SO(3)}} \sum_{(i,j)\in\mathcal{E}} \frac{w_t}{2}||\overline{\boldsymbol{t}_{ij}}-\boldsymbol{R}_i^T(\boldsymbol{t}_j-\boldsymbol{t}_i)||_2^2 - w_R tr\left(\boldsymbol{R}_j^{-1}\boldsymbol{R}_i\overline{\boldsymbol{R}_{ij}}\right)\\
\Leftrightarrow\quad &\min_{\substack{\boldsymbol{t}_i\in\mathbb{R}^3,\\ \boldsymbol{R}_i\in SO(3)}} \sum_{(i,j)\in\mathcal{E}} \frac{w_t}{2}||\overline{\boldsymbol{t}_{ij}}-\boldsymbol{R}_i^T(\boldsymbol{t}_j-\boldsymbol{t}_i)||_2^2 - 3w_R\\
&\qquad\qquad + \frac{w_R}{2}||\boldsymbol{R}_i^{-1}\boldsymbol{R}_j - \overline{\boldsymbol{R}_{ij}}||_F^2\\
\Leftrightarrow\quad &\min_{\substack{\boldsymbol{t}_i\in\mathbb{R}^3,\\ \boldsymbol{R}_i\in SO(3)}} \sum_{(i,j)\in\mathcal{E}} w_t||\overline{\boldsymbol{t}_{ij}}-\boldsymbol{R}_i^T(\boldsymbol{t}_j-\boldsymbol{t}_i)||_2^2 + w_R||\boldsymbol{R}_j - \overline{\boldsymbol{R}_{ij}}||_F^2\\
\Leftrightarrow\quad &\min_{\substack{\boldsymbol{t}_i\in\mathbb{R}^3,\\ \boldsymbol{R}_i\in SO(3)}} \sum_{(i,j)\in\mathcal{E}} w_t||\boldsymbol{t}_j-\boldsymbol{t}_i-\boldsymbol{R}_i\overline{\boldsymbol{t}_{ij}}||_2^2 + w_R||\boldsymbol{R}_i^T\boldsymbol{R}_j - \overline{\boldsymbol{R}_{ij}}||_F^2 \qquad (4-5)
\end{aligned}
$$

4.1.2 M-estimators on loop closure edge

But the version of MLE of equation (4-5) is not robust to outliers as it is a quadratic form. We supposed that the odometry edge are not prone to outlier so we can apply the extra treatment only to the loop closure edge. We choose to use a M-estimators approach on the loop closure edge as describe in Bosse et al. (2016). This means we add an extra loss function around the square norm of the error. The new formulation is then the one in equation 4-6 where first the distinction between the odometry edge and the loop closure edge is made by having two separate sum and second the loss function ρ is add only to the loop closure sum.

$$
\begin{aligned}
&\min_{\substack{\boldsymbol{t}_i\in\mathbb{R}^3,\\ \boldsymbol{R}_i\in SO(3)}} \sum_{(i,j)\in\mathcal{E}_{odometry}} w_t||\boldsymbol{t}_j-\boldsymbol{t}_i-\boldsymbol{R}_i\overline{\boldsymbol{t}_{ij}}||_2^2 + w_R||\boldsymbol{R}_i^T\boldsymbol{R}_j - \overline{\boldsymbol{R}_{ij}}||_F^2\\
&+ \sum_{(i,j)\in\mathcal{E}_{loop\ closure}} \rho\left(w_t\left|\left|\boldsymbol{t}_j-\boldsymbol{t}_i-\boldsymbol{R}_i\overline{\boldsymbol{t}_{ij}}\right|\right|_2\right) + \rho(w_R||\boldsymbol{R}_i^T\boldsymbol{R}_j - \overline{\boldsymbol{R}_{ij}}||_F)
\end{aligned}
\qquad (4-6)
$$

The M-estimator method performance depends highly on the loss function choice, and it is why chapter 6 is totally dedicated to the study of the loss function. We choose this approach mainly because to obtain a good estimate using pose graph optimization, we need a graph with high connectivity. Adding a lot of loop closure is how we can make the connectivity of the graph higher, however this means adding more outlier to measurements. If we just truncated the measurement set without be sure to take out the outlier, the connectivity will decrease, and the outlier rate can be higher. Keeping all measurements and making sure that outliers do not impact too much the quality of the results is then the approach we choose.

4.2 2-steps algorithm

It is possible to solve the full problem of the poses T_i with one optimization process using the equation 4-6. This is called a 1-step approach as we obtain in one go the rotation and translation estimates. Another approach called 2-steps algorithm is based on the observation that the rotation variables appear in the translation error term, but the translation variables don't appear in the rotational error term. Additionally (Carlone & Censi, 2012) show with empirical evidence that when optimizing only on the rotation variables the error term on translation is negligible. So, using only the rotational error term as an objective function for finding the rotation variable is enough. This observation is even more important because when the rotation matrix poses are fixed, the optimization problem over the translation variables reduces to a convex problem with the translational error term as objective

function. The figure 4−2 illustrated a 2−steps algorithm using the equation 4−6 as a basic MLE formulation.

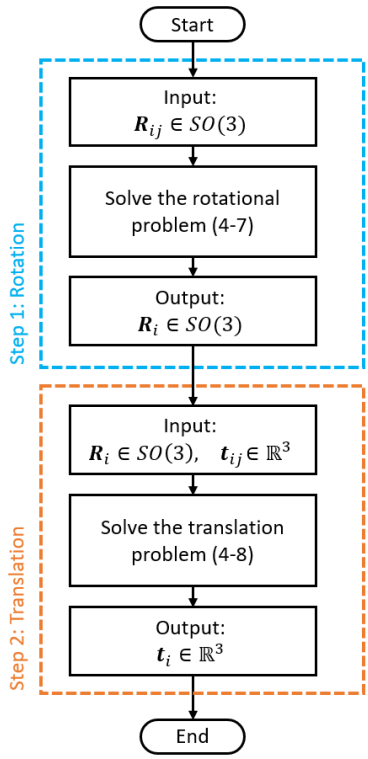

Figure 4−2 Flow chart of the 2−steps algorithm

$$\min_{R_i \in SO(3)} \sum_{(i,j)\in\mathcal{E}_{odo}} w_R ||R_i^T R_j - \overline{R_{ij}}||_F^2 + \sum_{(i,j)\in\mathcal{E}_{lc}} \rho(w_R ||R_i^T R_j - \overline{R_{ij}}||_F) \qquad (4-7)$$

$$\min_{t_i \in \mathbb{R}^3} \sum_{(i,j)\in\mathcal{E}_{odo}} w_t ||t_j - t_i - R_i \overline{t_{ij}}||_2^2 + \sum_{(i,j)\in\mathcal{E}_{lc}} \rho(w_t \left|\left| t_j - t_i - R_i \overline{t_{ij}} \right|\right|_2) \qquad (4-8)$$

The advantage of using a 2−steps approach is that it reduces the complexity of the optimization process as the matrix smaller. But since we don't use all the available information for optimizing the rotation, we should loose in accuracy. In our case, we want to

use a convex relaxation of the problem, which will be good if the relaxation is tight. (Carlone et al., 2018) compare the 1-step approach to the 2-steps in the 2D case for robust PGO and all their results show that the 1-step approach is not tight when they are outlier. But the 2-steps algorithm when using the convex relaxation only of the rotation problem is tight even with more outlier. Then, we need to use a 2-steps approach if we want to be robust against outlier and use convex relaxation.

5 Rotation sub-problem resolution

The 2-steps approach is breaking the problem into 2 problems: the rotation non-convex (because of the $SO(3)$ group which is the feasible set) and the translation convex when the rotation is fixed. Now, we want to go further and relax the rotation sub-problem of equation 4-7 into a convex problem. The reasons for the relaxation are double. The initial idea was to solve a problem with a global optimum so first we don't need to care about the initialization of the solver and always obtain the global solution. Then, secondly when deriving a convex relaxation, we can check after resolution if the relaxation was tight and so we have a way to certify that the rotation solution is the global optima of the initial problem.

Briefly, we first need to derive the convex relaxation of the problem in chapter 5.1. But then the feasible set of the convex problem is wider than the initial one as shows figure 5-1, so the solution found can be non-feasible for the initial problem 4-7. Then we need a rounding procedure to reproject the relaxed solution into $SO(3)$. How to achieve the rounding is presented in

chapter 5.2. Lastly, we will talk in more details about the certifiability contract in part 5.3. In rest of this chapter, we will suppose that we know the rotation/translation covariance and that the loss function is chosen to be convex.

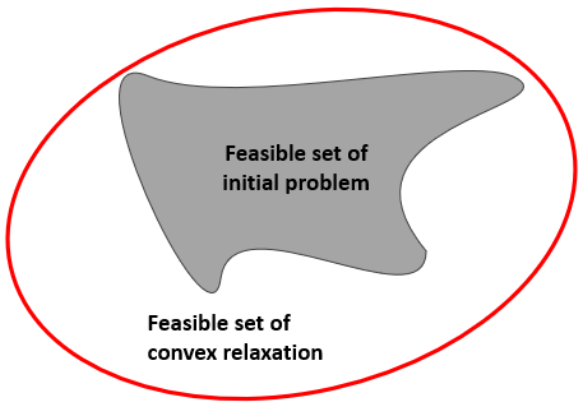

Figure 5−1 Illustration of non−convex feasible set relaxation

5.1 Convex relaxation

We start from the problem 1 which is nonconvex as the feasible set is the special orthogonal group which is nonconvex. To obtain a nice convex formulation for our relaxed problem, we follow simple step which can be replicated for others problem if necessary. The overall flow of the problem derivation is given in figure 5−2.

Problem 1 (SO(3) feasible set formulation of maximum−likelihood orientation estimation)
Given the observation $\overline{\boldsymbol{R_{ij}}} \in \boldsymbol{SO(3)}$, *for* $(i,j) \in \mathcal{E}$, *the rotation variance* w_R *and a convex loss function* ρ, *find the set of minimizers* $\widehat{\boldsymbol{R}}_i \in SO(3)$ *that satisfies*

$$\widehat{\boldsymbol{R}}_i = \min_{\boldsymbol{R}_i \in SO(3)} \sum_{(i,j)\in\mathcal{E}_{odo}} w_R ||\boldsymbol{R_i^T R_j} - \overline{\boldsymbol{R_{ij}}}||_F^2 + \sum_{(i,j)\in\mathcal{E}_{lc}} \rho(w_R ||\boldsymbol{R_i^T R_j} - \overline{\boldsymbol{R_{ij}}}||_F) \qquad (5-1)$$

Having fixed the first node orientation to $R_0 = \boldsymbol{I}_3$

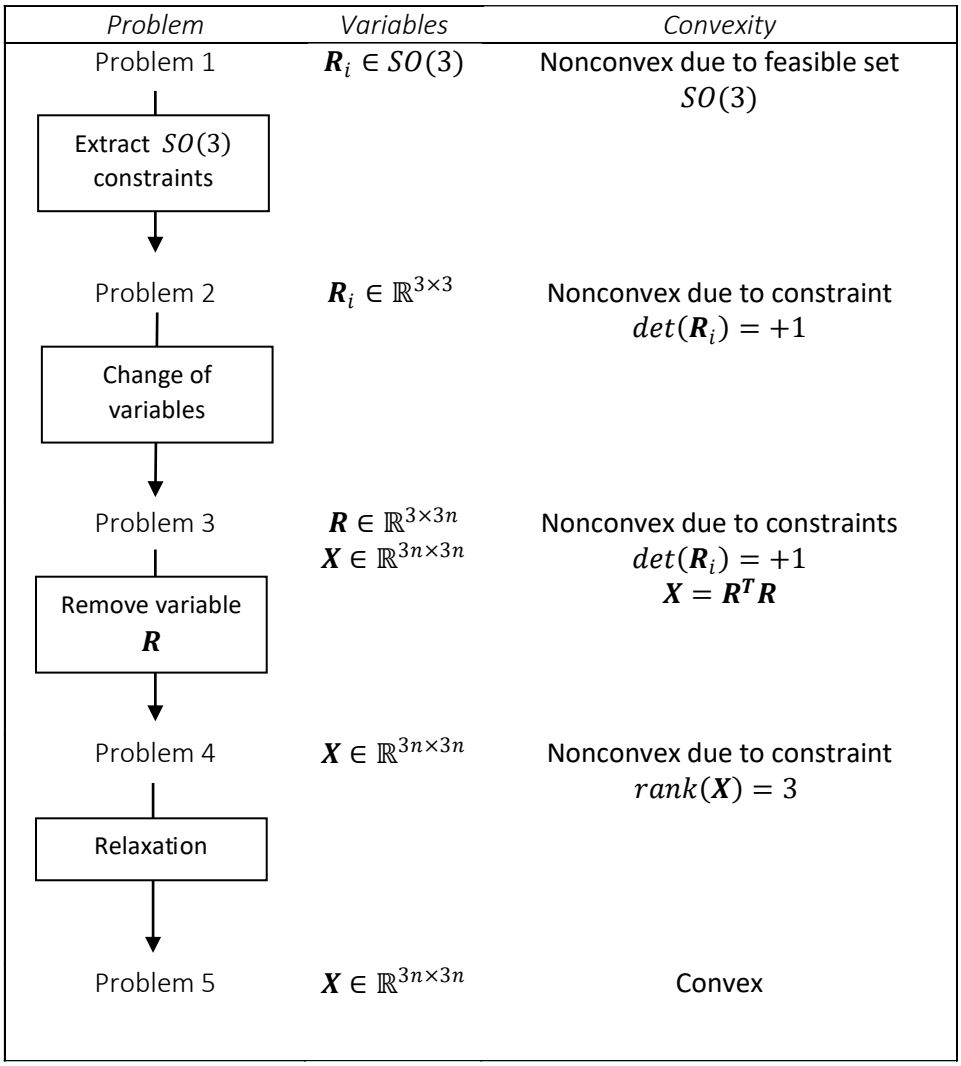

Figure 5−2 Problem derivation chart

Starting from the problem 1, our main goal is to obtain a convex problem, so the first step is to explicitly make appear the condition on the non−convex condition on the $SO(3)$ feasible set. To do so, we change the feasible set to be $\mathbb{R}^{3\times3}$ and force the variable to be part of the $SO(3)$ by adding constraints 5−2b and 5−2c which leads to the Problem 2.

Problem 2 ($\mathbb{R}^{3\times3}$ feasible set formulation of maximum−likelihood orientation estimation)

Given the observation $\overline{\boldsymbol{R_{ij}}} \in \boldsymbol{SO(3)}$, *for* $(i,j) \in \mathcal{E}$, *the rotation variance* w_R *and a convex loss function* ρ, *find the set of minimizers* $\widehat{\boldsymbol{R}}_i \in \mathbb{R}^{3\times3}$ *that satisfies*

$$\widehat{\boldsymbol{R}}_i = \min_{\boldsymbol{R}_i \in \mathbb{R}^{3\times3}} \sum_{(i,j)\in\mathcal{E}_{odo}} w_R ||\boldsymbol{R_i^T R_j} - \overline{\boldsymbol{R_{ij}}}||_F^2 + \sum_{(i,j)\in\mathcal{E}_{lc}} \rho(w_R ||\boldsymbol{R_i^T R_j} - \overline{\boldsymbol{R_{ij}}}||_F) \tag{5−2a}$$

Subject to $\boldsymbol{R_i^T R_i} = \boldsymbol{I_3}$ (5−2b)

$\boldsymbol{det(R_i)} = +1$ (5−2c)

Having fixed the first node orientation to $R_0 = \boldsymbol{I_3}$

The nonconvexity comes from the constraint 5−2c, but first before relaxing the problem we want to reparametrize the problem into a more compact and convenient matrix notation. We introduce 2 new variables: a vector stacking all the rotation matrix $\boldsymbol{R} = [\boldsymbol{R_1}, \dots, \boldsymbol{R_n}] \in \mathbb{R}^{\boldsymbol{3\times3n}}$ and a matrix construct by multiplying the vector

$$\boldsymbol{X} = \boldsymbol{R^T R} = \begin{bmatrix} \boldsymbol{R_1^T R_1} & \cdots & \boldsymbol{R_1^T R_n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{R_n^T R_1} & \cdots & \boldsymbol{R_n^T R_n} \end{bmatrix} \in \mathbb{R}^{3n\times3n} \tag{5−3}$$

The variables $\boldsymbol{R_i}$ can be replaced using $\boldsymbol{R}$ and the products of the variables $\boldsymbol{R_i^T R_j}$ and $\boldsymbol{R_i^T R_i}$ can be replaced using $\boldsymbol{X}$. As we define the matrix $\boldsymbol{X}$ using the rotation vector, we need to add the constraint 5−4d in the problem 3.

Problem 3 (Double matrix variables formulation of maximum−likelihood orientation estimation)

Given the observation $\overline{\boldsymbol{R_{ij}}} \in \boldsymbol{SO(3)}$, *for* $(i,j) \in \mathcal{E}$, *the rotation variance* w_R *and a convex loss function* ρ, *find the set of minimizers* $\boldsymbol{R} \in \mathbb{R}^{3\times3n}, \boldsymbol{X} \in \mathbb{R}^{3n\times3n}$ *that satisfies*

$$\boldsymbol{R},\boldsymbol{X} = \min_{\substack{\boldsymbol{R}\in\mathbb{R}^{3\times 3n}\\ \boldsymbol{X}\in\mathbb{R}^{3n\times 3n}}} \sum_{(i,j)\in\mathcal{E}_{odo}} w_R \|[\boldsymbol{X}]_{ij} - \overline{\boldsymbol{R}_{ij}}\|_F^2 + \sum_{(i,j)\in\mathcal{E}_{lc}} \rho\left(w_R \left\|[\boldsymbol{X}]_{ij} - \overline{\boldsymbol{R}_{ij}}\right\|_F\right) \tag{5-4a}$$

Subject to $[\boldsymbol{X}]_{ii} = \boldsymbol{I}_3$, (5-4b)

$det([\boldsymbol{R}]_i) = +1$, (5-4c)

$\boldsymbol{X} = \boldsymbol{R}^T\boldsymbol{R}$, (5-4d)

Having fixed the first node orientation to $R_0 = \boldsymbol{I}_3$

The problem 3 is an optimization over 2 variables. First, we know that the constraint 4-12d is equivalent to have $\boldsymbol{X} \succcurlyeq \boldsymbol{0}$ and $rank(\boldsymbol{X}) = 3$ so we can replace it in the problem. So, the only term where the rotation vector appears is on the constraint 4-12c. It constraint the rotation matrix to be in the Special orthogonal group, if we don't enforce it the rotation matrix will be in the orthogonal group 0(3), which means the determinant will be ± 1. But previous study of relaxation over SO(3) have shown that dropping this constraint doesn't impact the relaxation process and so we will not use it in problem 4.

Problem 4 (Single matrix variable formulation of maximum-likelihood orientation estimation)
Given the observation $\overline{\boldsymbol{R}_{ij}} \in \boldsymbol{SO(3)}$, *for* $(i,j) \in \mathcal{E}$, *the rotation variance* w_R *and a convex loss function* ρ, *find the set of minimizers* $\boldsymbol{X} \in \mathbb{R}^{3n\times 3n}$ *that satisfies*

$$\boldsymbol{X} = \min_{\boldsymbol{X}\in\mathbb{R}^{3n\times 3n}} \sum_{(i,j)\in\mathcal{E}_{odo}} w_R \|[\boldsymbol{X}]_{ij} - \overline{\boldsymbol{R}_{ij}}\|_F^2 + \sum_{(i,j)\in\mathcal{E}_{lc}} \rho\left(w_R \left\|[\boldsymbol{X}]_{ij} - \overline{\boldsymbol{R}_{ij}}\right\|_F\right) \tag{5-5a}$$

Subject to $[\boldsymbol{X}]_{ii} = \boldsymbol{I}_3$, (5-5b)

$\boldsymbol{X} \succcurlyeq \boldsymbol{0}$, (5-5c)

$rank(\boldsymbol{X}) = 3$ (5-5d)

Having fixed the first node orientation to $R_0 = I_3$

The problem 4 is close to be a convex problem but the constraint 5−5d on the rank is nonconvex. So, we need to relax it and let the solution matrix to have full rank if needed. In practice the solution matrix will have rank 3 if there are no outliers. And the rank will go up with outliers' addition as the relaxation will be less tight.

Problem 5 (Semi−definite formulation of maximum−likelihood orientation estimation)
Given the observation $\overline{R_{ij}} \in SO(3)$, *for* $(i,j) \in \mathcal{E}$, *the rotation variance* w_R *and a convex loss function* ρ, *find the set of minimizers* $\widehat{R}_i \in \mathbb{R}^{3\times3}$ *that satisfies*

$$X = \min_{X \in \mathbb{R}^{3n\times3n}} \sum_{(i,j)\in\mathcal{E}_{odo}} w_R \|[X]_{ij} - \overline{R_{ij}}\|_F^2 + \sum_{(i,j)\in\mathcal{E}_{lc}} \rho(w_R \left\|[X]_{ij} - \overline{R_{ij}}\right\|_F) \tag{5−6a}$$

Subject to $[X]_{ii} = I_3$, (5−6b)

$X \succcurlyeq 0$ (5−6c)

Having fixed the first node orientation to $R_0 = I_3$

The final form of our rotation problem is the problem 5 as it is convex. Even more it is semidefinite so we can use off−the−shelf solver to find the solution. One details that will be important is the hypotheses on the convexity of the loss function which make the problem convex.

5.2 Rounding procedure

After solving the problem 5, we obtain the matrix X solution which contains all the necessary information to retrieve the rotation matrix. It can also be seen as the step to reproject the solution on

the problem 1 feasible set $SO(3)$. The algorithm 1 illustrates how to get the wanted rotation matrix from the estimate matrix (Hartley et al., 2013; Wang et al., 2013) we do 2 approximations that break the equivalence of the different problem: drop the determinant and rank constraint on the matrix. Starting from the assumption that the matrix wanted is supposed to have rank 3, we compute in step 2 the rank−3 singular value decomposition of the matrix. By construction most of the solution components contain redundant information, we can use only the first column $\begin{bmatrix} \boldsymbol{R}_1^T \boldsymbol{R}_1 \\ \vdots \\ \boldsymbol{R}_n^T \boldsymbol{R}_1 \end{bmatrix}$ which already has the information about all the rotation matrix $\widehat{\mathbf{R}}_\mathbf{i}$. So, from the SVD decomposition we recompose the different $\boldsymbol{R}_i^T \boldsymbol{R}_1$ terms as the $\mathbf{T}_\mathbf{i}$ matrix in step 5 especially as we implicitly fixed the rotation $\boldsymbol{R}_1 = \boldsymbol{I}_3$. Finally, we fixed back the determinant to 1 in the step 9 if needed.

ALGORITHM 1: THE ROUNDING PROCEDURE

Input: *Estimated rotation matrix* $\boldsymbol{X} \in \mathbb{R}^{3n \times 3n}$

Output: *Rounded rotation matrix* $\widehat{\boldsymbol{R}}_i \in SO(3)$

1: ***function*** *ROUNDING_ROTATION(**X**)*
2: $[U \Sigma V^T]$ ← *rank-3 singular value decomposition of* $\boldsymbol{X}$
3: *Set* $\boldsymbol{T} = [\boldsymbol{v}_1\ \boldsymbol{v}_2\ \boldsymbol{v}_3]$
4: ***For*** *i = 1, …, n* ***do***
5: *Set* $\boldsymbol{T}_i$ *as the i-th matrix of size 3×3*
6: $[M \Xi N^T]$ ← *SVD of* $\boldsymbol{T}_i$
7: $\widehat{\boldsymbol{R}}_i = MN^T$
8: ***If*** $det(\widehat{\boldsymbol{R}}_i) = -1$ ***then***
9: *Set Det(* $\widehat{\boldsymbol{R}}_i$*)* $= 1$
10: ***End***

11: ***End***

12: ***return*** $\widehat{\boldsymbol{R}}_i$

13: ***end***

5.3 Certifiability contract

In a world where the failure of the algorithm can have important consequence, the need of contract on the quality of the estimate solution is of primal importance. So, what we call a certifiability contract is a posteriori performance guarantees. They will tell us if the estimated solution from the convex relaxation problem is also the global solution of the initial nonconvex problem. This is equivalent to checking if the convex relaxation is tight.

We note the objective function of problem 1 as f_1 and its optimal cost f_1^*. We note for the convex problem 5 the objective function as f_5 and its optimal cost f_5^*. When we relaxed a nonconvex problem, we relax the feasible set of the problem, so we know as equation (2−23) stated that $f_5^* \leq f_1^*$. This condition means that the optimal cost obtain from the relaxation problem is smaller or equal to the optimal cost of the initial problem. We additionally note $\widehat{\boldsymbol{X}}$ the solution associate to the optimal cost f_5^* and $\widehat{\boldsymbol{R}}_i$ the corresponding rounded rotation matrix. We can then derive the following equivalent inequalities:

$$f_5^* \leq f_1^*$$

$$\Leftrightarrow \quad f_5(\widehat{\boldsymbol{X}}) \leq f_1^*$$

$$\Leftrightarrow \quad -f_1^* \leq -f_5(\widehat{\boldsymbol{X}})$$

$$\Leftrightarrow \quad f_1(\widehat{\boldsymbol{R}}_i) - f_1^* \leq f_1(\widehat{\boldsymbol{R}}_i) - f_5(\widehat{\boldsymbol{X}}) \qquad (5-7)$$

In the left term of equation 5−7, the difference between the cost obtained with the convex relaxation and the actual unknow optimal cost represent the suboptimality gap of our convex relaxation. And since we don't know f_1^* value, we can't compute it directly. But using the inequality 5−7 we can compute a bound for it as the left term of the equation can be compute after the rounding. If the left term is 0 then we can tell that $f_1^* = f_1(\widehat{\boldsymbol{R}}_i)$ and that the rounded rotation matrices are the optimal value for the initial nonconvex problem.

We can directly compute the gap $f_1(\widehat{\boldsymbol{R}}_i) - f_5(\widehat{\boldsymbol{X}})$ to check the quality of our estimate. However, we can also see that this difference is a representation of the difference between the estimated matrix and the rounded one. So, if the matrix $\widehat{\boldsymbol{X}}$ is exactly rank 3 and if the first n 3×3 block of its rank 3 decomposition are already in SO(3) then the gap will be 0. So instead of computing the gap, we can have a more straightforward criterion which is the rank of $\widehat{\boldsymbol{X}}$.

6 Loss function

For the robustness of the formulation against outlier, we use M−estimator. The principle is to add a loss function around the norm of the error to reduce the impact of large error. There are a variety of different loss function already existing with desirable properties. We will first talk about the theory of loss function and of their properties in part 6.1. Then the most common loss function

used in PGO will be presented in part 6.2.

6.1 Theory

A loss function is a function that aims to reduce the influence of the outliers which have large errors. The loss function $\rho: t \mapsto \rho(t) \geq 0$ should be non−negative, non−decreasing and with a unique minimum at $t = 0$, so it doesn't change the minima of the objective function. There are some interesting indicators that can be compute on loss function to understand and compare their effects on the estimator. Usually, estimators are compared in terms of bias and efficiency. The bias is defined as the difference between the estimator's expected value and the true value. When the efficiency is how good is the estimator. If we look at the estimator as a random variable with a distribution η, then the bias is link to the mean and the efficiency to the variance. In real application, we cannot have the best estimator, there is a trade−off between the quality/efficiency and the robustness/bias of the estimator.

To visually assess the robustness of a M−estimator with a chosen loss function, we can plot the influence curve. The desired properties here is that we the error goes to infinity, the influence function should go to 0, which means the error measurement will be neglected. These are called redescending M−estimators. From this influence curve, we can go further and compute the gross error sensitivity as in equation 6−1 It represents the maximum effect that an outlier can have.

$$GES(\eta) = \sup_{z} ||IF(z, \eta)|| \qquad (6-1)$$

Another curve is the maximum bias one which represent the maximum bias in function of the proportion of outliers. Using this curve, we can define the breakdown point which is the maximum proportion of outlier that the estimator can handle. It can be read on the maximum-bias curve directly. To derive all this indicator, we can use the derivative of the loss function $\rho'(t)$ also called the weight function and so it is useful to look at it. We want to choose the loss function such that the efficiency is high around 0 so we correct the inliers position accurately. The gross error sensitivity should be lowest as possible, so an outlier doesn't affect too much the solution. The breakdown point should be large so we can handle many outliers. An additional property of the loss function that we want to look at in our study case is the convexity of the function. We can only use convex function to keep the convexity of the relaxed problem.

6.2 Common loss function comparison

The study of some loss function for M-estimator by Bosse et al. (2016) and by MacTavish et al. (2015) and give us the following list of useful loss function in table 6-1. We also add the Charbonnier loss function that is a function that appears in the adaptive Barron loss function used in AEROS by Milad Ramezani et al. (2022). The loss function formulas are given. The variable s in the formula is a tuning constant using to fix the scale of the loss function. The figure 6-1 shows the curve of the different loss function with the parameter $s = 1$. First, we want a convex function so we cannot use the Tukey, threshold, Welsch, Geman function.

From the remaining one, the one which have the slowest growing rate are the one the more robust to outlier

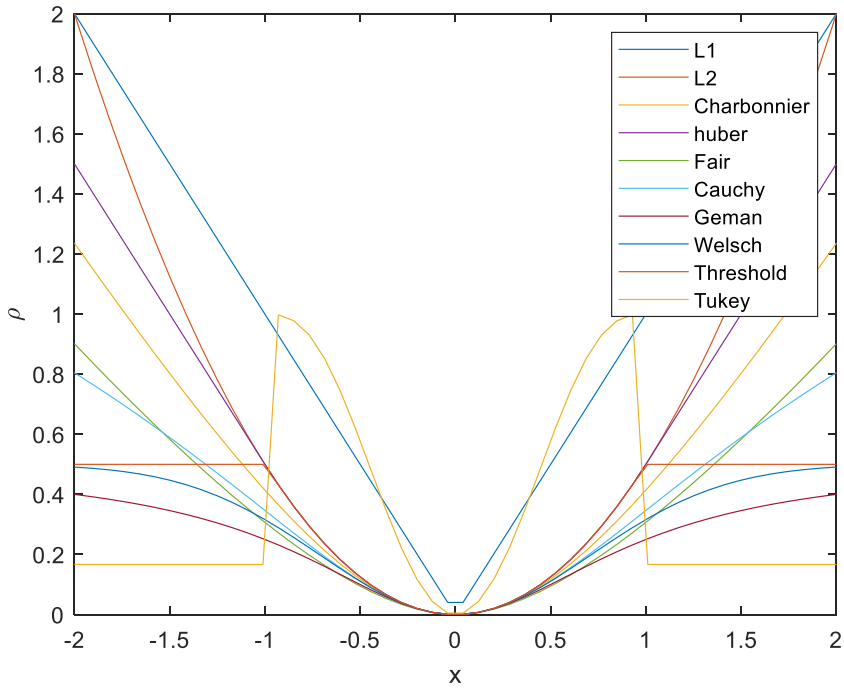

Figure 6-1 Common loss function shape

From the shape of the function and the convexity analysis, we can already tell that the best functions for the M-estimator are nonconvex and so cannot be used in our formulation. The only possible functions are L1, L2, Charbonnier which is a combination of L1 and L2, Huber and fair function.

Table 6−1 Loss function list

Loss function	$\boldsymbol{\rho(t)}$	$\boldsymbol{\rho'(t)}$	Convexity
Gaussian/L2	$\frac{t^2}{2}$	t	Convex
L1/Laplace	$\|t\|$	$sgn(t)$	Convex
Charbonnier	$\sqrt{\left(\frac{t}{s}\right)^2 + 1} - 1$	$\frac{t}{\sqrt{s^2t^2 + s^4}}$	Convex
Huber	$\begin{cases} \frac{t^2}{2} & \|t\| \le s \\ s(\|t\| - \frac{s}{2}) & otherwise \end{cases}$	$\begin{cases} t & t \le s \\ s * sgn(t) & otherwise \end{cases}$	Convex
"Fair"	$s^2[\frac{\|t\|}{s} - \log\left(1 + \frac{\|t\|}{s}\right)]$	$\frac{t}{1 + \frac{\|t\|}{s}}$	Convex
Cauchy	$\frac{s^2}{2}\log\left(1 + t^2/s^2\right)$	$\frac{t}{1 + \left(\frac{t}{s}\right)^2}$	Nonconvex

Geman−McClure	$\frac{\frac{t^2}{2}}{s+t^2}$	$\frac{t}{\left(1+\frac{t^2}{s^2}\right)^2}$	Nonconvex
Welsch	$\frac{s^2}{2}[1-\exp\left(-\left(\frac{t}{s}\right)^2\right)]$	$t*\exp\left(-\left(\frac{t}{s}\right)^2\right)$	Nonconvex
Threshold	$\begin{cases}\frac{t^2}{2} & \lvert t\rvert \le s \\ \frac{s^2}{2} & otherwise\end{cases}$	$\begin{cases} t & \lvert t\rvert \le s \\ 0 & otherwise\end{cases}$	Nonconvex
Tukey	$\begin{cases}\frac{s^2}{6}(1-\left[1-\left(\frac{t}{s}\right)^2\right]^3) & \lvert t\rvert \le s \\ \frac{s^2}{6} & otherwise\end{cases}$	$\begin{cases} t\left(1-\left(\frac{t}{s}\right)^2\right)^2 & \lvert t\rvert \le s \\ 0 & otherwise\end{cases}$	Nonconvex

7 Datasets and evaluation method description

We want to study the relaxation approach with the M−estimator on the loop closure. For comparing the different loss function, we will focus on synthetic dataset as we didn't try to optimize the process in terms of running time. Also, cvx in MATLAB is well−known for not scaling well with the amount of data. The datasets created are inspired by the one used in Carlone et al. (2018) extended for 3D testing and from Rosen et al. (2017) with additional outliers.

7.1 Synthetic Dataset

We create different type of synthetic pose graph dataset for testing. The first one is a totally random pose graph with high connectivity. The second is less random in the edges position but still has high connectivity. The last one is demonstrating a robot predefined trajectory and has low connectivity. The steps for the creation of the different pose graph are shown in algorithm 2. The subfunction in line 2 and 3 are different in function of the graph type and will be details later.

ALGORITHM 2: SYNTHETIC POSE GRAPH CREATION

Input: ***DataType***, ***n*** *number of node,* ***n_lc*** *number of loop closure,* ***w_r*** *rotation variance,* ***w_t*** *translation variance,* ***p_out*** *percentage of outliers in the loop closure.*

Output: *Ground truth poses* $\boldsymbol{T}_i \in SO(3)$

Noisy measurement edges $\overline{\boldsymbol{T}}_{ij} \in SO(3)$

1: ***function*** $[\boldsymbol{T}_i, \widehat{\boldsymbol{R}}_{ij}] =$ *DATA_CREATION*$(\boldsymbol{DataType}, \boldsymbol{n}, \boldsymbol{n_{lc}}, \boldsymbol{w_r}, \boldsymbol{w_t}, \boldsymbol{p_{out}})$

2: $\boldsymbol{T}_i \leftarrow$ *GroundTruthPoseCreation*$((\boldsymbol{DataType}, \boldsymbol{n})$

3: *Compute the Edge connected set* $\boldsymbol{E_c}$

4: *If* $\boldsymbol{E_C}$ *is creating a non-connected graph, restart at 3*

5: $\overline{\boldsymbol{T}}_{ij} \leftarrow$ *OdometryMeasurementCreation*$(\boldsymbol{DataType}, \widehat{\boldsymbol{R}}_i, \boldsymbol{E_c}, \boldsymbol{w_r}, \boldsymbol{w_t})$

6: ***For*** $\boldsymbol{i}$ = *1, …,* $\boldsymbol{n\_lc}$ ***do***

7: *Draw Bernouilli variable* $\boldsymbol{b}$ *of probability* $\boldsymbol{p\_out}$

8: *Pick an edge* $\boldsymbol{e_c}$ *which is not in* $\boldsymbol{E_c}$

9: ***If*** $\boldsymbol{b} = 1$ ***then***

10: $\overline{\boldsymbol{T}}_{ij} \leftarrow$ *LoopClosureOutlierCreation*$(\boldsymbol{DataType}, \widehat{\boldsymbol{R}}_i, \boldsymbol{e_c}, \boldsymbol{w_r}, \boldsymbol{w_t})$

11: ***Else***

12: $\overline{\boldsymbol{T}}_{ij} \leftarrow$ *LoopClosureInlierCreation*$(\boldsymbol{DataType}, \widehat{\boldsymbol{R}}_i, \boldsymbol{e_c}, \boldsymbol{w_r}, \boldsymbol{w_t})$

13: ***end***

14: ***end***

15: ***return*** $\boldsymbol{T}_i, \overline{\boldsymbol{T}}_{ij}$

16: ***end***

We can give more details on some important line in algorithm 2:

(4) If the create graph has some node or group of nodes not connected, then we discard it and recreated a new one.

(5) The odometry measurement links to this edge is compute using the model of equation 4−1 where the noise is Gaussian for the translation of mean $\mathbf{0_3}$ and variance $w_t^2 = 0.1$. The rotation matrix noise is construct from the Euler angles which are generated from a gaussian of mean 0 and variance $w_R = 0.01$.

(8) The loop closure edges are chosen from the set of edges which are not odometry one.

(10) If the loop closure is an inlier, then it is generated with the

same model as the odometry.

(12) If the loop closure is an outlier, then we create completely wrong measurement by setting the noise in equation 4−1 to follow a uniform distribution over $[-\frac{D}{4},\frac{D}{4}]$ for the translation and $[0,2\pi]$ for the rotation.

7.1.1 Erdos−Rényi pose graph

(2) The ground truth positions are randomly draw from a cube of dimension $\boldsymbol{D}\times\boldsymbol{D}\times\boldsymbol{D}$ from a uniform distribution. The orientations are also randomly pick from a uniform distribution between $[0,2\pi]$ for each Euler angle and then transform into the matrix representation.

(3) Using Erdos−Rényi graph model, we first create odometry edges. An edge between 2 poses exists with a probability of 0.5. This type of graph is highly connected has more than half of the edges between nodes exists. An example with $n=20$ and $n_{lc}=10$ is show in figure 7−1 where the grey edges are odometry and the red loop closure.

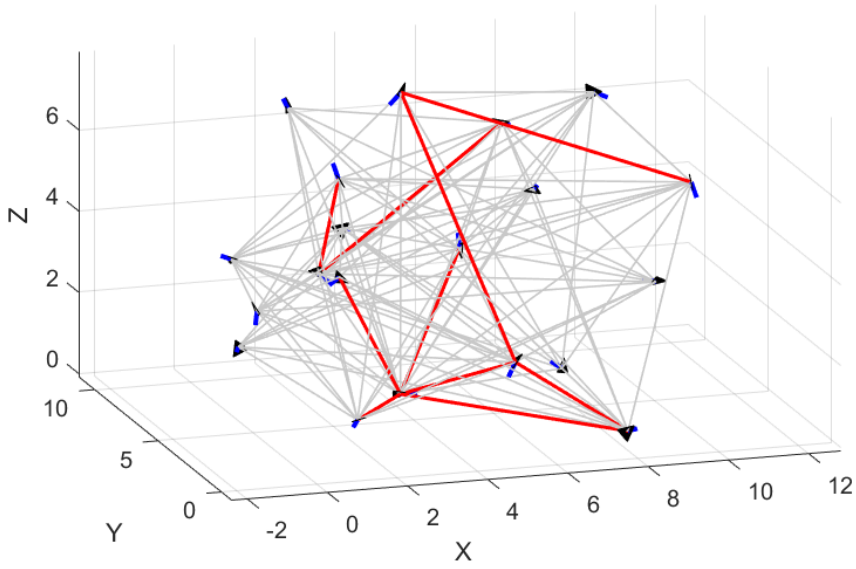

Figure 7−1 Erdos−Rényi pose graph example ($n = 20, n_{lc} = 10$)

7.1.2 Geometric random pose graph

(2) The ground truth positions are randomly draw from a cube of dimension $\boldsymbol{D \times D \times D}$ from a uniform distribution. The orientations are also randomly pick from a uniform distribution between $[0,2\pi]$ for each Euler angle and then transform into the matrix representation.

(3) Using Geometric random graph model, we first create odometry edges. All the pose which are at a distance radius smaller than $\frac{\boldsymbol{D}}{\boldsymbol{3}}$ are connected. This type of graph is less connected than Erdos−Rényi graph but still highly connected compared to real SLAM pose graph. An example with $n = 20$ and $n_{lc} = 10$ is show in figure 7−2 where the grey edges are odometry and the red loop closure.

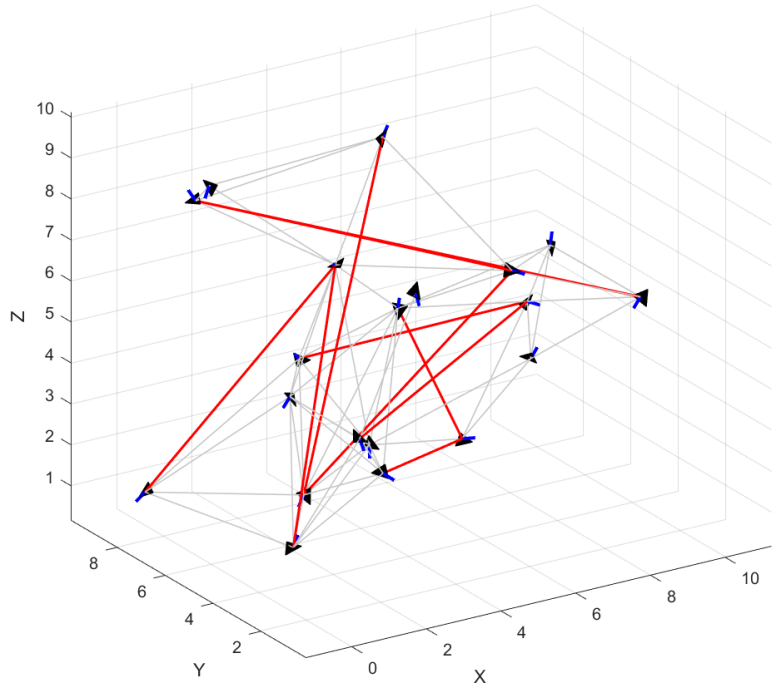

Figure 7−2 Geometric random pose graph example ($n = 20, n_{lc} = 10$)

7.1.3 Cube pose graph

(2) This type of pose graph is inspired from real pose graph for SLAM application. We assume the robot moves following a grid of size $\boldsymbol{D}$. We used the wanted number of nodes to compute the step size between two nodes as $\delta = \frac{D}{n^{1/3}-1}$

(3) As we reproduce a real trajectory, the connected edges set is just all the edge such that $j = i + 1$. As we create the poses following the specific path that the robot is following. An example with $n = 20$ and $n_{lc} = 10$ is show in figure 7−3 where the grey edges are odometry and the red loop closure.

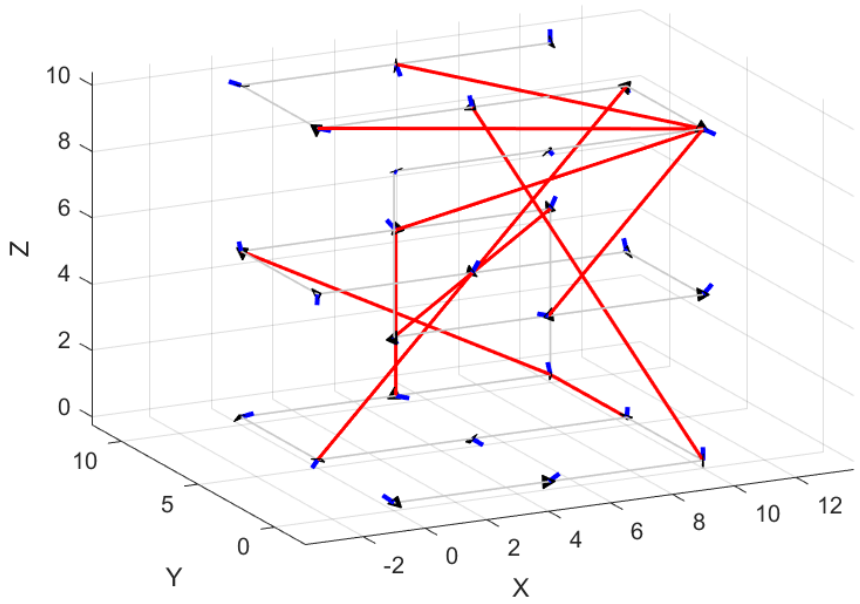

Figure 7−3 Cube pose graph example ($n = 27, n_{lc} = 10$)

7.2 Results estimation methods

To evaluate the proposed algorithm, we compute some indicators. All the indicators will be computed for 10 runs to see the overall trend. The solver selected in cvx is the SDPT3, which is one of the free solvers (unlicensed one).

7.2.1 Tightness of the relaxation

One of the main information about the quality of the estimate is the rank of the solution of the convex relaxation problem. We compute the stable rank which is a real instead of an integer and give more precise information about how close we are from the rank 3. The stable rank is defined as the squared ration between the Frobenius norm and the spectral norm.

$$r_s = \left(\frac{||X||_F}{||X||_2}\right)^2 \qquad (7-1)$$

The Frobenius norm is just the concatenation of all the coefficients which can be compute as $||X||_F = \left(\sum_{i,j=1}^{n} |x_{ij}|^2\right)^{1/2} = \sqrt{tr(XX^*)}$ and the spectral norm is defined using the eigenvalues as $||X||_2 = \sqrt{\rho(XX^*)}$ where $\rho(X) = \max_{1 \le i \le n} |\lambda_i|$. We know that the two norms will always follow the property $||X||_2 \le ||X||_F$ and so the stable norm will be minimum 3 and go up if the relaxation is not tight.

We also compute the relaxation gap to check if the rounding procedure is projecting correctly our matrix in SO(3) by using the formula 7-2

$$Gap = f_1(\widehat{\boldsymbol{R}}_i) - f_5(\widehat{\boldsymbol{X}}) \qquad (7-2)$$

7.2.2 Poses error

As we used synthetic dataset for testing, we have the ground truth data available to estimate the quality of our PGO directly. We will compute the mean error on the rotation and on the translation separately. During the optimization, the only available data are measurements between poses. These are relative information, so the final poses computed are respecting the relative measurements but are not the same as the ground truth poses. To compare with the ground truth, we first project the estimated and the ground truth pose in the same coordinate system by setting in the two trajectories the first pose to be at the origin with 0 angles. For the rotation we use the quaternion notation in the ARE formula of

equation 7−3 and for the translation we use the ATE formula of equation 7−4.

$$ARE_i = 2 * \text{acos}\left(|q_{esti} \cdot q_{groundtruth}|\right) \quad (7-3)$$

$$ATE_i = ||t_{esti} - t_{groundtruth}||_2^2 \quad (7-4)$$

8 Simulation results

The different function that will be compared are the following one: L2, Identity, L1, Huber. For L2, as it squares the Frobenius norm, we have still a quadratic cost. The identity is using only the Frobenius norm not square, so we excepted more robustness. For L1, as we want to take the absolute value, we slightly change the formulation and use the L1 norm instead of the Frobenius norm. The Huber implementation has the most potential as it hard implements a boundary for the large error. The other convex function cannot be implemented in their primary form in cvx as they don't follow the Disciplined Convex Programming rules. To use them in cvx, they need to be reformulated in a way such that cvx can ensure the convexity of the final form which is not always possible.

We plot all the indicators: rank, relaxation gap, average rotational error, and average translation error. We plot first the comparison of the mean indicator value for all the different loss function. Then for more details analysis we plot for each function the mean, the minimum and the maximum of each indicator for the monte Carlo run in function of the outlier rate.

8.1 Pose graph optimization results

8.1.1 Erdos-Rényi graph

We plot for the 4-loss function (Identity, L2, L1 and Huber) the average of the 4 indicators for 20 poses and 10 loop closure erdos-Rényi graph. The figure 8-1, 8-2, 8-3 and 8-4 shows that the Huber function seems to fail in most case when there are outliers. We were expecting it to be the best at handling outlier so we will look first in more details the results only for the Huber function Monte Carlo run given in Table 8-1.

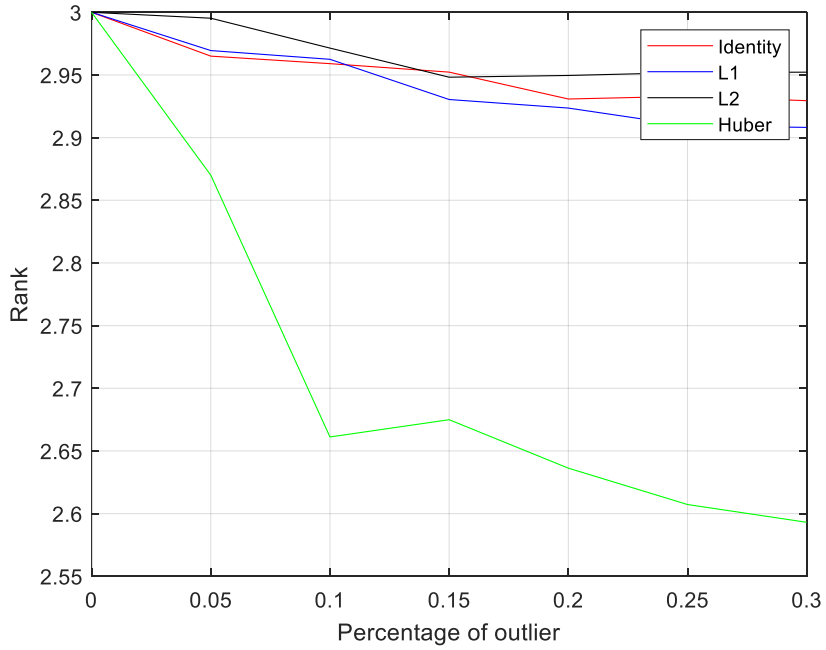

Figure 8-1 Stable rank comparison for Erdos-Rényi graph

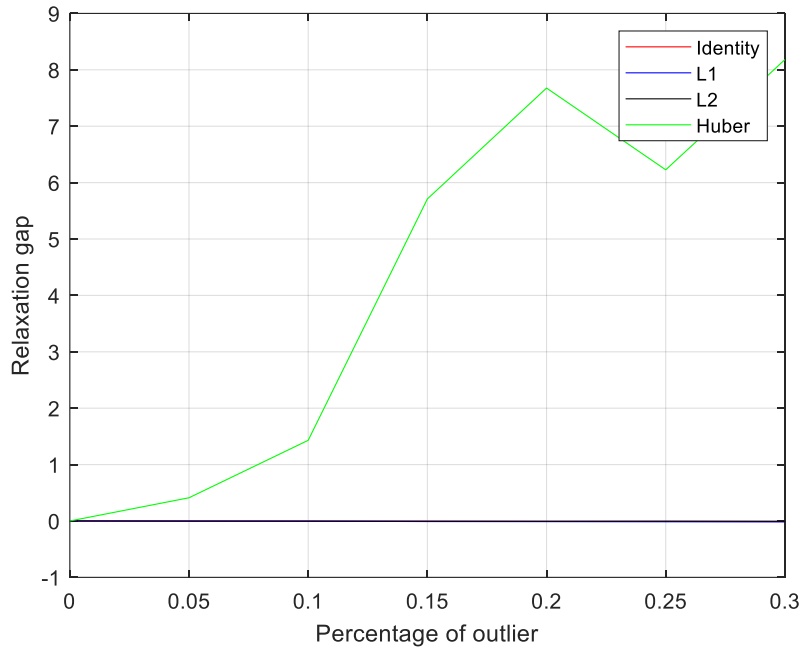

Figure 8-2 Relaxation gap comparison for Erdos-Rényi graph

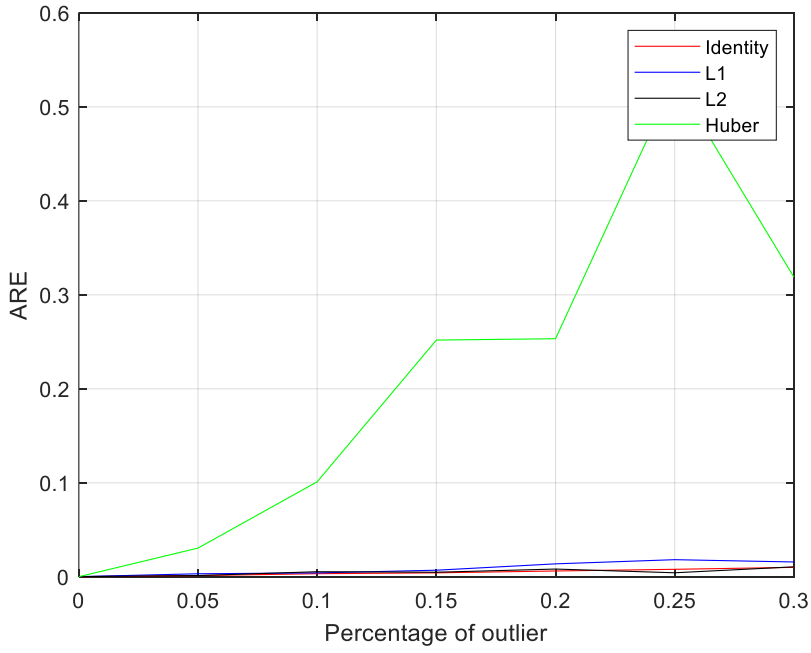

Figure 8−3 Rotational error comparison for Erdos−Rényi graph

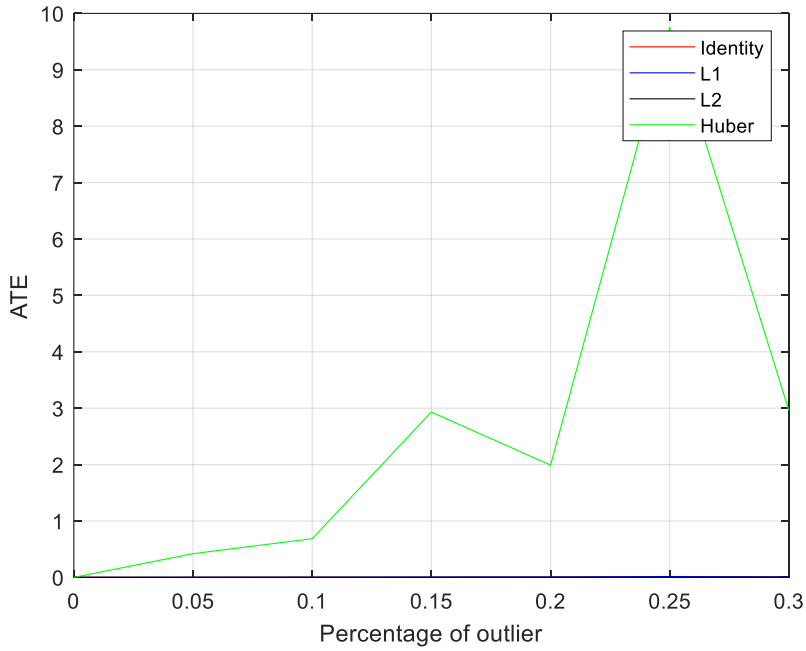

Figure 8−4 Translation error comparison for Erdos−Rényi graph

In the table, the rank data are rounded to 3 digits. So, when the rank is 3, it is 2,9999. We highlight in red all the run when the rank of the relaxed solution was 3. We can see from the table 8−1 that our certifiability contract holds. When the rank of the matrix is 3, the error on the rotation and translation is small, in all others case the algorithm fails to retrieve the solution. If we refer with other state of the art algorithm such as Carlone et al. (2018), the Huber loss function should be able to retrieve the solution even at an outlier rate of **0.3**. In my case, I'm guessing the failure is coming from my implementation of the Huber loss function with the Frobenius norm as I used a custom function to overpass a problem in the DCP ruleset.

Table 8-1 Details of Monte Carlo run for the Huber loss on Erdos-Rényi

p_out	0	0.05	0.1	0.15	0.2	0.25	0.3
				RANK			
1	3.000	2.437	3.000	2.662	2.395	2.481	2.727
2	3.000	3.000	2.459	3.000	2.563	2.531	2.375
3	3.000	3.000	2.567	2.722	2.615	2.469	2.727
4	3.000	3.000	3.000	2.665	2.662	2.766	2.799
5	3.000	3.000	2.408	2.374	3.000	2.625	2.421
6	3.000	3.000	2.602	2.584	2.780	2.633	2.734
7	3.000	3.000	2.372	2.558	2.530	2.634	2.609
8	3.000	3.000	3.000	3.000	2.752	2.594	2.603
9	3.000	3.000	2.586	2.543	2.401	2.514	2.532
10	3.000	2.264	2.617	2.641	2.665	2.826	2.403
				RELAXATION GAP			
1	9.39E-08	4.61E+00	9.10E-08	3.15E+00	-1.07E+00	8.30E+00	1.27E+01
2	-6.26E-08	2.11E-07	-8.52E-01	3.47E-08	-3.85E+00	-6.41E-01	-3.60E+00
3	7.19E-08	6.02E-08	2.39E+00	3.74E+00	1.43E+01	-7.63E+00	-7.77E+00
4	1.16E-09	1.91E-07	8.11E-08	3.77E+01	5.75E+00	-1.26E+01	2.24E+00
5	1.08E-07	-1.28E-08	-1.32E+01	3.03E+00	1.66E-07	5.05E+00	5.73E+00
6	-2.20E-08	-8.27E-09	1.62E+01	4.33E+00	1.27E+01	3.48E+00	2.03E+01
7	-3.44E-08	7.45E-09	-4.51E+00	-9.70E-01	3.73E+00	1.10E+00	1.39E+01
8	7.99E-08	2.09E-07	-1.45E-07	2.17E-08	2.35E+01	1.01E+01	4.49E+00

9	1.16E-07	1.83E-07	4.53E+00	2.02E+00	-1.47E+00	1.80E+01	1.17E+01
10	9.95E-08	-4.95E-01	9.71E+00	4.11E+00	2.31E+01	3.71E+01	2.22E+01
				ARE			
1	8.25E-05	1.12E-01	1.09E-04	1.13E+00	1.97E-01	7.55E-01	6.36E-01
2	1.12E-04	7.24E-05	1.37E-01	1.01E-04	2.11E-01	5.50E-01	1.97E-01
3	1.20E-04	8.95E-05	1.28E-01	3.50E-01	5.01E-01	6.94E-01	3.50E-01
4	1.11E-04	9.88E-05	1.08E-04	4.35E-01	6.96E-01	4.36E-02	1.78E-01
5	9.38E-05	9.44E-05	6.84E-02	1.78E-01	7.52E-05	1.92E+00	2.56E-01
6	1.20E-04	1.00E-04	3.45E-01	1.03E-01	2.43E-01	3.52E-01	3.58E-01
7	1.79E-04	1.29E-04	7.77E-02	5.71E-02	1.13E-01	9.28E-02	4.36E-01
8	9.09E-05	1.59E-04	7.66E-05	7.99E-05	2.21E-01	2.37E-01	2.87E-01
9	1.10E-04	9.87E-05	1.28E-01	8.19E-02	1.05E-01	2.51E-01	2.62E-01
10	7.82E-05	1.92E-01	1.26E-01	1.78E-01	2.45E-01	3.91E-01	2.29E-01
				ATE			
1	6.96E-05	3.98E-01	2.30E-04	1.90E+01	5.92E-01	1.14E+01	5.45E+00
2	5.69E-05	6.79E-05	5.58E-01	6.72E-05	8.29E-01	8.06E+00	1.93E+00
3	8.03E-05	5.59E-05	8.54E-01	2.45E+00	2.66E+00	1.08E+01	3.17E+00
4	1.34E-04	1.43E-04	6.85E-05	1.15E+00	8.24E+00	3.09E-01	3.08E+00
5	1.06E-04	5.06E-05	3.77E-01	3.50E+00	4.85E-05	5.64E+01	1.98E+00
6	8.47E-05	6.15E-05	1.65E+00	6.08E-01	5.51E-01	4.05E+00	1.46E+00
7	1.20E-04	5.93E-05	6.45E-01	7.87E-02	4.62E-01	3.30E-01	7.30E+00
8	8.75E-05	9.21E-05	4.81E-05	6.94E-05	3.54E+00	3.15E+00	1.69E+00
9	3.48E-05	9.56E-05	1.15E-01	3.82E-01	1.95E+00	1.80E+00	2.49E+00
10	1.36E-04	3.81E+00	2.66E+00	2.15E+00	1.09E+00	1.24E+00	1.01E+00

From now on, I will not plot the data for the Huber loss on the graph to be able to visually compare the 3 other functions. Another information that we can obtain is that only looking at the mean of the indicator is not meaningful as some iteration fails. So first we will look at the distribution of the indicator for the 3 other loss functions in figure 8-5. The first remark we can say is that in most the 3-loss function manage to retrieve a good estimate of the poses even with outliers. The L2 function is given better results, then the identity and finally the L1. So here, in this case the classical quadratic formulation gives better results and the L1 which should be the more robust gives the worse results. This comes the fact that this type of graph is highly connected. If we see in Figure 7-1, the number of loop closure is really small compared to the number of odometry edges so even with outliers the graph stay resilient and using loss robust function on the loop closure actually made us loose some precision.

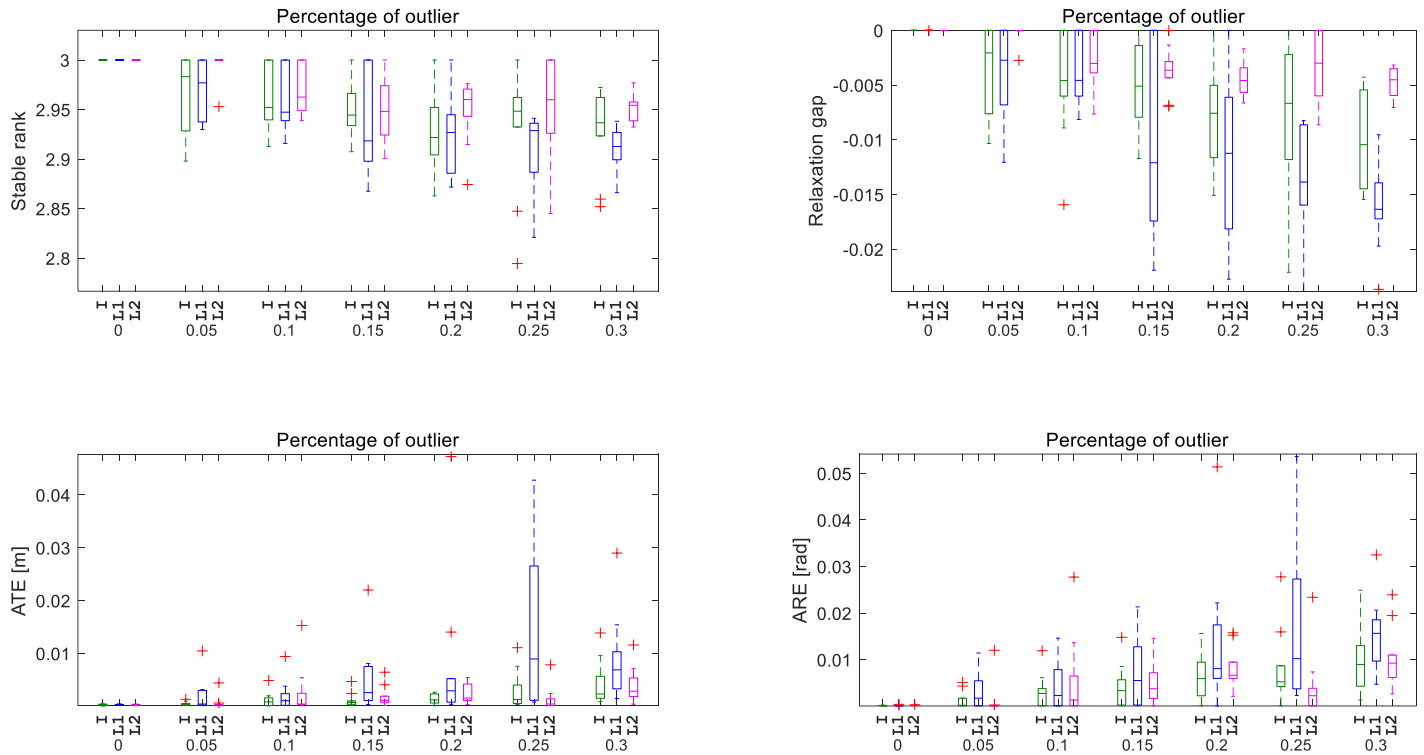

Figure 8-5 Details of all runs for the Erdos-Rényi pose graph

8.1.2 Geometric random graph

We run the analysis for the 4−loss function (Identity, L2, L1 and Huber) and Huber loss function is showing the same pattern as in the Erdos−Rényi graph. So, we don't show the results in the figure 8−6 which details the results obtained for 20 nodes and 10 loop closure for a Geometric random graph. The first think we can immediately tell is that the error in all runs is bigger than in the previous type of graph, this was predictable as the number of edges is reduced. In most case the 3 loss functions manage to retrieve a good estimate, but we start to see failure at 15 percent of outlier in the loop closure set. Here the Identity function and the L1 are slightly better in average than L2 but all function gives order of error at the end.

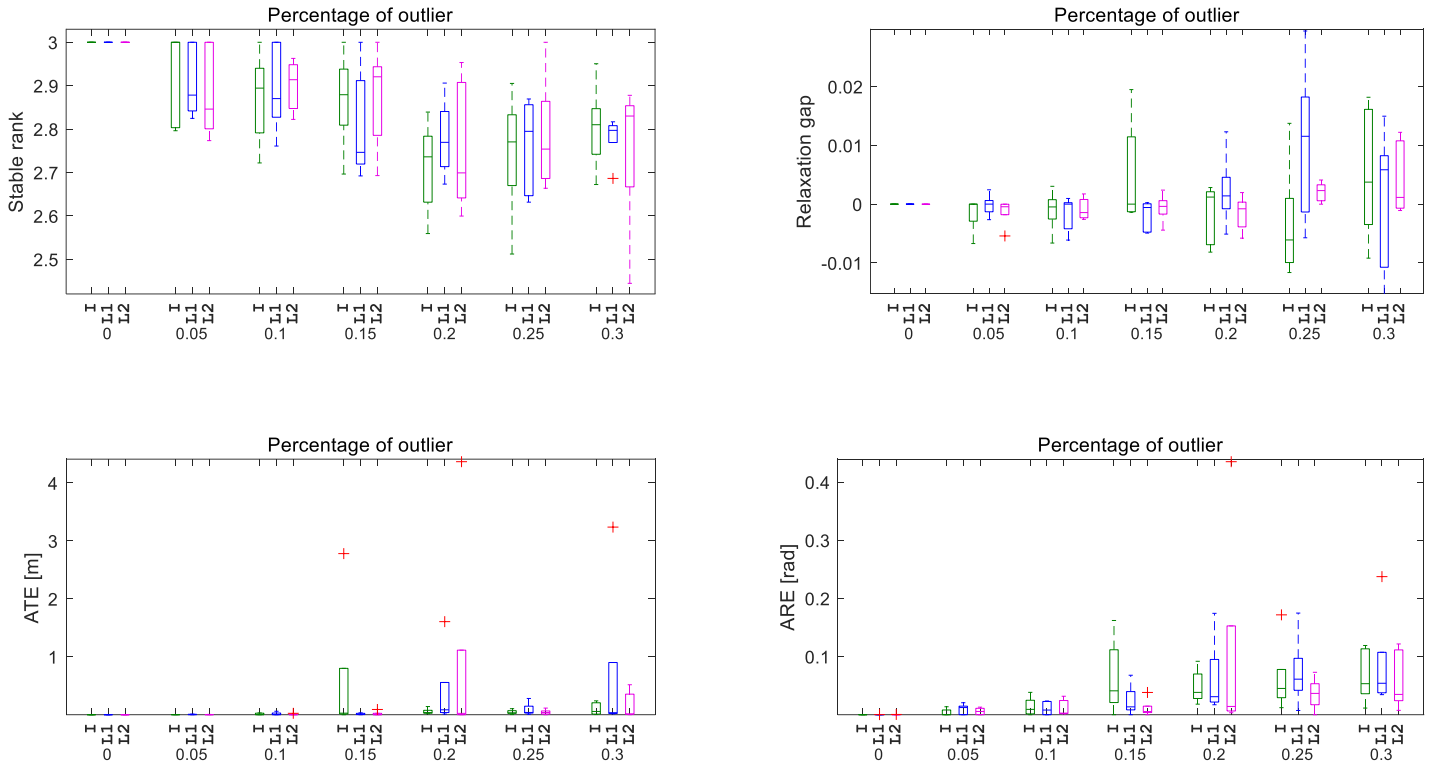

Figure 8−6 Details of all runs for the geometric random pose graph

8.1.3 Cube graph

We do the same process as before, with the Huber loss function still left out of the plot so we can compare the others. The cube graph is the one that reproduce a real trajectory the most and the less connected of all type of graph. This low connectivity is directly impacting the quality of the solution as we can see than in many cases with all the loss function, our proposed algorithm is failing to retrieve a good estimate. In this case of low connectivity graph even five percent of outliers can make our approach fails with the L1, L2 or Identity loss function.

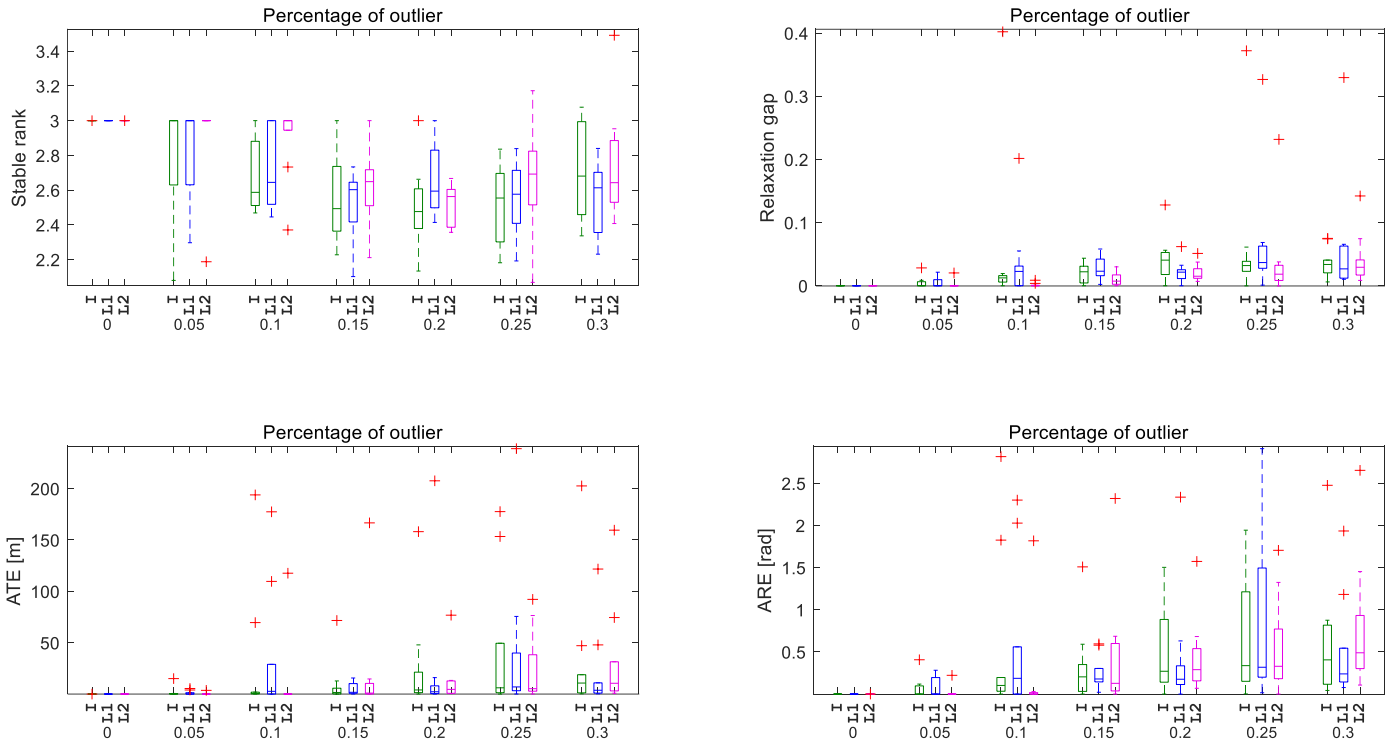

Figure 8-7 Details of all runs for the cube pose graph

8.2 Results analysis

First, from our results, we can't really compare the Huber loss function with the other as the results seems wrong due to the implementation in cvx. But for the other 3, they show the same tendencies to give better results when the graph is highly connected. When we have a low number of edges, using the L1, L2 or identity function is not enough to be robust against outliers. For highly connected graph, any of this function will give good estimates in a certain measure. Now we will compare additional parameters influence and give some interesting comment on the algorithm.

8.2.1 Number of failures

In every pose graph problem, there are time when the estimated poses are wrong, we saw that it was directly correlated with the rank of the estimated rotation solution. If look in details as the value of the final rotation error and translation error, we will define the boundary value for the rank to be not under 2,8. For the 3 types of data, we plot the number of times when the rank is over this value in Figure 8-8, figure 8-9 and figure 8-10. For the Erdos-Rényi pose graph, only the Huber fails, the 3 others always obtain a possible pose graph. For the geometric random pose graph, we confirm our impression that more solution are completely wrong one with the L2 which seems to stay closer at least for a small number of outliers. For the cube graph, we fail to retrieve the solution in most case when they are outliers.

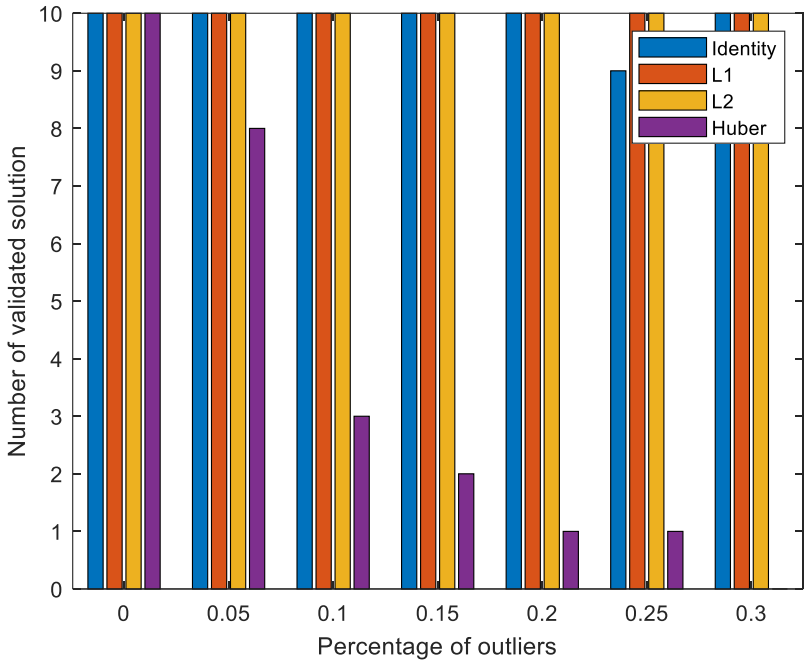

Figure 8-8 Number of certifiable contracts validate for Erdos-Rényi pose graph

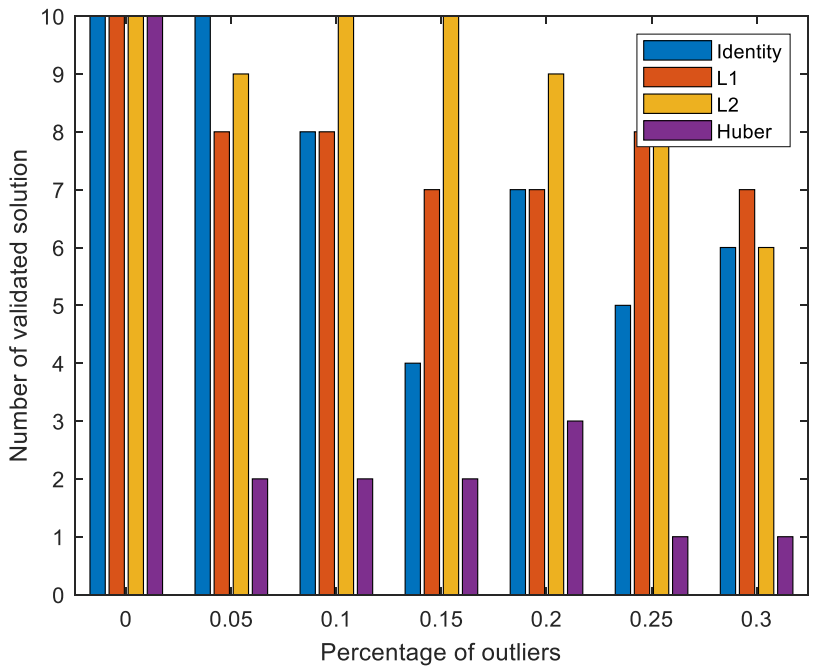

Figure 8-9 Number of certifiable contracts validate for geometric random pose graph

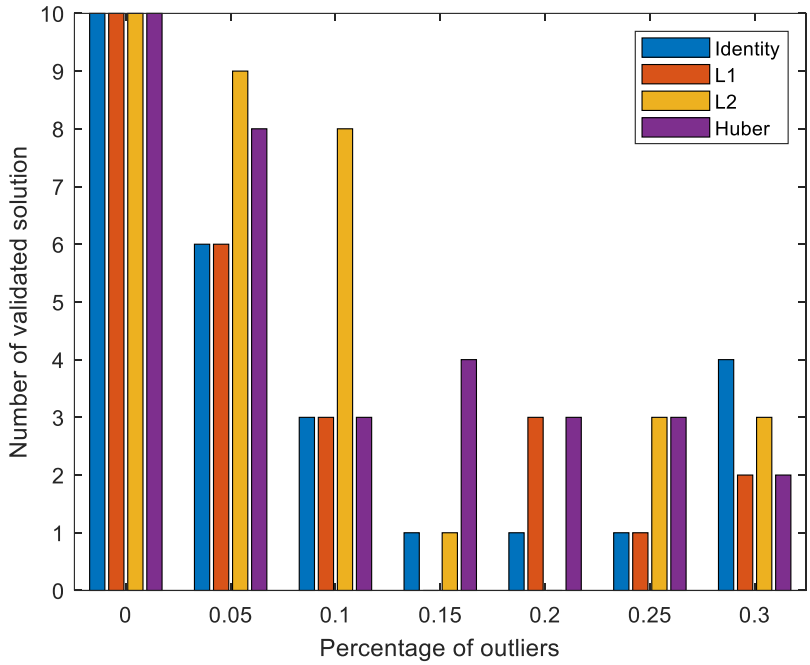

Figure 8-10 Number of certifiable contracts validate for Erdos-Rényi pose graph

8.2.2 Computation time

We can look at the computation time in function of the number of nodes in the graph. The graph from figure 8-11 is taken from the cube pose graph. In our formulation, the part that takes the most time is the rotation estimation by cvx especially because we need to reproject after. The time for retrieving the rotation is exponential in function of the number of nodes and the time for the translations is linear.

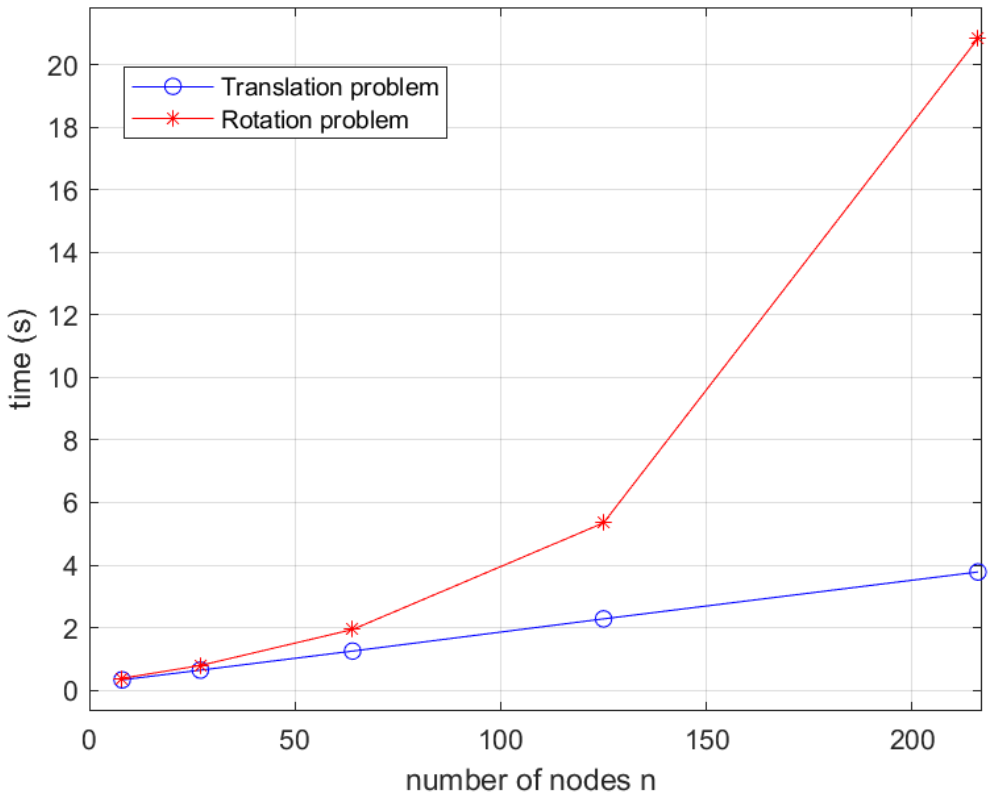

Figure 8-11 Computation time in function of the number of nodes

9 Conclusion and future works

From the results, several conclusions can be drawn. First the computation time is too long for real application which is a common problem for all convex relaxation algorithm existing nowadays as they always run batch. The study of Semidefinite solver and the design of a special solver for large instance of pose graph is something that we need for the future of this method.

Secondly, for low connectivity graph we need to use a more robust function than the basic L1, L2. And real life pose graphs are not always highly connected. Or the more connected they are, the more outliers there are. The paper from Yang et al. (2023) used Truncated Least square as robust loss function.

Thirdly, in our algorithm we naively tried to keep all the information during the optimization and check if the relaxation is tight for study only. This is a good idea when optimizing the first time the pose graph. But as SLAM is usually an incremental process, it will be important to use the rank for discarding wrong solution and to separate the outlier after the detection for the next back-end update. This can help keep the outlier percentage low and still manageable.

Lastly, the main restriction on comparing the different loss function comes from the convexity condition on it. So, if we want to keep the approach of convex relaxation, we need to rewrite the cost function another way. Recent work from Yang et al. (2023). have shown a way to rewrite many costs function into an additional variable

optimizable along the poses. The relaxation used is a different one, i.e the Lasserre' s hierarchy relaxation. We plan in the future to study this type of relaxation which is more adapted for outlier rejection as it can use better loss function such as the Truncated Least square.

References

Agarwal, P., Tipaldi, G.D., Spinello, L., Stachniss, C., Burgard, W. (2013). Robust map optimization using dynamic covariance scaling. 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, pp. 62−69.
DOI: 10.1109/ICRA.2013.6630557

Alvarez, E. (2019). Semidefinite Relaxation for the Optimal Operation and Expansion Planning of Power Transmission Systems.
DOI: 10.13140/RG.2.2.12881.28006.

Bernholt, T. (2006). Robust Estimators are Hard to Compute. Technical Report, No. 2005,52, Universität Dortmund, Sonderforschungsbereich 475 − Komplexitätsreduktion in Multivariaten Datenstrukturen, Dortmund.

Boyd, S., Vandenberghe, L. (2004). Convex optimization, Cambridge university press.

Bosse, M., Agamennoni, G., Gilitschenski, I. (2016). Robust Estimation and Applications in Robotics. Foundations and Trends® in Robotics, 4(4), pp. 225−269, 2016.
DOI: 10.1561/2300000047.

Carlone, L., & Censi, A. (2012). From Angular Manifolds to the Integer Lattice: Guaranteed Orientation Estimation With Application to Pose Graph Optimization. IEEE Transactions on Robotics, 30, 475−492.

Carlone, L., Calafiore, G.C., Tommolillo, C., Dellaert, F. (2016). Planar Pose Graph Optimization: Duality, Optimal Solutions, and Verification. IEEE Transactions on Robotics, 32(3), pp. 545−565.
DOI: 10.1109/TRO.2016.2544304

Carlone, L., Calafiore, G.C. (2018). Convex Relaxations for Pose Graph Optimization with Outliers. IEEE Robotics and Automation Letters, 3(2), pp. 1160-1167, April 2018.
DOI: 10.1109/LRA.2018.2793352.

Carlone, L., Censi, A., Dellaert, F. (2014). Selecting good measurements via ℓ1 relaxation: A convex approach for robust estimation over graphs. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, pp. 2667-2674.
DOI: 10.1109/IROS.2014.6942927

Carlone, L., Dellaert, F. (2015). Duality-based verification techniques for 2D SLAM. Proceedings - IEEE International Conference on Robotics and Automation, 2015, pp. 4589-4596.
DOI: 10.1109/ICRA.2015.7139835.

Carlone, L., Rosen, D., Calafiore, G., Leonard, J., Dellaert, F. (2015). Lagrangian Duality in 3D SLAM: Verification Techniques and Optimal Solutions. arXiv e-prints. URL:
https://arxiv.org/abs/1506.00746

Chaves, S., Kim, A., Galceran, E., Eustice, R. (2016). Opportunistic sampling-based active visual SLAM for underwater inspection. Autonomous Robots, 40.
DOI: 10.1007/s10514-016-9597-6.

Gao, X., Zhang, T. (2021). Introduction to Visual SLAM: From Theory to Practice. Springer Singapore.
DOI: https://doi.org/10.1007/978-981-16-4939-4.

Gómez, E., Gómez-Villegas, M., Marin, J. (1998). A multivariate generalization of the power exponential family of distributions. Communications in Statistics-theory and Methods - COMMUN STATIST-THEOR METHOD, 27, pp. 589-600.
DOI: 10.1080/03610929808832115.

Grisetti, G., Kümmerle, R., Stachniss, C., Burgard, W. (2010). A tutorial on graph-based SLAM. IEEE Transactions on Intelligent Transportation Systems Magazine, 2, pp. 31-43.
DOI: 10.1109/MITS.2010.939925.

Hartley, R., Ponce, J., Kuang, Y., Gorban, D., Gasparini, F. (2013). Rotation Averaging. International Journal of Computer Vision, 103, pp. 267-305.
DOI: 10.1007/s11263-012-0601-0

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W. (2011). G2o: A general framework for graph optimization. 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, pp. 3607-3613.
DOI: 10.1109/ICRA.2011.5979949.

Lajoie, P.-Y., Hu, S., Beltrame, G., Carlone, L. (2019). Modeling Perceptual Aliasing in SLAM via Discrete-Continuous Graphical Models. IEEE Robotics and Automation Letters, 4(2), pp. 1232-1239, April 2019.
DOI: 10.1109/lra.2019.2894852.

Latif, Y., Lerma, C.C., Neira, J. (2013). Robust Loop Closing Over Time. Robotics: Science and Systems VIII, MIT Press, pp. 233-240.

Lee, G.H., Fraundorfer, F., Pollefeys, M. (2013). Robust pose-graph loop-closures with expectation-maximization. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, pp. 556-563.
DOI: 10.1109/IROS.2013.6696406

MacTavish, K., Barfoot, T. (2015). At all Costs: A Comparison of Robust Cost Functions for Camera Correspondence Outliers. In: 2015 12th Conference on Computer and Robot Vision (CRV), Halifax, NS, Canada, pp. 62-69.
DOI: 10.1109/CRV.2015.52.

McGann, D., Rogers, J.G., Kaess, M. (2022). Robust Incremental Smoothing and Mapping (riSAM). arXiv e-prints. URL: https://arxiv.org/abs/2209.14359.
DOI: 10.48550/arXiv.2209.14359.

Olson, E., Agarwal, P. (2013). Inference on Networks of Mixtures for Robust Robot Mapping. The International Journal of Robotics Research, 32, pp. 826-840.
DOI: 10.1177/0278364913479413.

Protzel, P., Sünderhauf, N. (2012). Switchable constraints for robust pose graph SLAM. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, pp. 1879-1884.
DOI: 10.1109/IROS.2012.6385590

Ramezani, M., Mattamala, M., & Fallon, M.F. (2021). AEROS: AdaptivE RObust Least-Squares for Graph-Based SLAM. Frontiers in Robotics and AI, 9.

Rosen, D.M., Carlone, L., Bandeira, A.S., Leonard, J.J. (2017). SE-Sync: A Certifiably Correct Algorithm for Synchronization over the Special Euclidean Group. arXiv e-prints. URL: https://arxiv.org/abs/1612.07386

Wang, L., Singer, A. (2013). Exact and Stable Recovery of Rotations for Robust Synchronization. arXiv e-prints. URL: https://arxiv.org/abs/1211.2441

Yang, H., Carlone, L. (2023). Certifiably Optimal Outlier-Robust Geometric Perception: Semidefinite Relaxations and Scalable Global Optimization. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(3), pp. 2816-2834, 1 March 2023.
DOI: 10.1109/TPAMI.2022.3179463.

Yuh, J., Marani, G., Blidberg, D.R. (2011). Applications of marine robotic vehicles. Intel Serv Robotics, 4, pp. 221–231. DOI: 10.1007/s11370-011-0096-5.

Zereik, Enrica & Bibuli, Marco & Miskovic, Nikola & Ridao, Pere & Pascoal, Antonio. (2018). Challenges and future trends in marine robotics. Annual Reviews in Control. 46. 10.1016/j.arcontrol.2018.10.002.