3
Robustness to Incorrect Data Association and Outliers

Heng Yang, Josh Mangelson, Yun Chang, Jingnan Shi,
Niko Sunderhauf, and Luca Carlone

In Chapter 1, we have seen that factor graphs are a powerful representation to model and visualize SLAM problems, and that maximum a posteriori (MAP) estimation provides a grounded and general framework to infer variables of interest (*e.g.,* robot poses and landmark positions) given a set of measurements (*e.g.,* odometry and landmark measurements). For instance, we observed that when the measurements $\boldsymbol{z}_i$ are affected by additive and zero-mean Gaussian noise with covariance $\boldsymbol{\Sigma}_i$, MAP estimation leads to a *nonlinear least-squares* optimization:

$$\boldsymbol{x}^{\mathrm{MAP}} = \underset{\boldsymbol{x}}{\arg\min} \sum_i \|\boldsymbol{z}_i - \boldsymbol{h}_i(\boldsymbol{x}_i)\|^2_{\boldsymbol{\Sigma}_i}, \tag{3.1}$$

where $\boldsymbol{x}_i$ denotes the subset of the states involved in measurement i.[1] In this chapter we notice that in practice many measurements $\boldsymbol{z}_i$ —possibly due to incorrect data association— may have large errors, which are far from following a zero-mean Gaussian (Section 3.1); these measurements typically induce large perturbations in the estimate $\boldsymbol{x}^{\mathrm{MAP}}$ from eq. (3.1). Therefore, we discuss how to reject gross outliers in the SLAM front-end (Section 3.2) and then focus on how to increase robustness to remaining outliers in the SLAM back-end (Section 3.3). We close the chapter with a short review of recent trends and extra pointers to related work (Section 3.4).

3.1 What Causes Outliers and Why Are They a Problem?

This section argues that outliers are inevitable in most SLAM applications and that not handling them appropriately leads to grossly incorrect estimates.

3.1.1 Data Association and Outliers

To understand the cause of outlier measurements, let us consider two examples.

First, consider a landmark-based SLAM problem, where we have to reconstruct

[1] While for simplicity eq. (3.1) assumes that measurements belong to a vector space, the algorithms in this chapter apply to arbitrary SLAM problems where variables belong to manifolds, see Chapter 2.

the trajectory of the robot and the position of external landmarks from odometry measurements and relative observations of landmarks from certain robot poses. Assuming (as we did in Chapter 1) that the landmark measurements have zero-mean Gaussian noise leads to terms in the optimization in the form $\|\boldsymbol{z}_{ij} - \boldsymbol{h}(\boldsymbol{p}_i, \boldsymbol{\ell}_j)\|^2_{\boldsymbol{\Sigma}}$. These terms model the fact that a given measurement $\boldsymbol{z}_{ij}$ is an observation of landmark $\boldsymbol{\ell}_j$ from pose $\boldsymbol{p}_i$ up to Gaussian noise, where $\boldsymbol{h}(\cdot)$ is the function describing the type of relative measurement (*e.g.,* range, bearing, etc.). In practice, the measurements $\boldsymbol{z}_{ij}$ are obtained by pre-processing raw sensor data in the SLAM front-end. For instance, if the robot has an onboard camera and $\boldsymbol{z}_{ij}$ is a visual observation of the bearing to a landmark $\boldsymbol{\ell}_j$, the measurement $\boldsymbol{z}_{ij}$ might be extracted by performing object (or more generally, feature) detection and matching in the image, and then computing the bearing corresponding to the detected pixels. Now, the issue is that the detections are imperfect and a landmark detected as $\boldsymbol{\ell}_j$ in the image, might be actually a different landmark in reality. This causes $\boldsymbol{z}_{ij}$ to largely deviate from the assumed model. The problem of associating a measurement to a certain landmark is typically referred to as the *data association problem* and is common to many other estimation problems (*e.g.,* target tracking). Therefore, incorrect data association creates outliers in the estimation problem.

As a second example, consider a pose-graph optimization problem, where we are primarily interested in estimating the trajectory of the robot (represented as a set of poses), and the measurements are either odometry measurements (which relate consecutive poses along the trajectory) or *loop closures* (which relate non-consecutive and possibly temporally distant poses). In practice, the loop closures are detected using (vision-based or lidar-based) place recognition methods, which are in charge of detecting if a pair of poses $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ have observed the same portion of the environment. Unfortunately current place recognition methods are prone to making mistakes and detecting loop closures between poses that are *not* observing the same scene. This is partially due to limitations of current methods, but it is often due to *perceptual aliasing*, that is the situation where two similarly looking locations actually correspond to different locations (think of two classrooms in a university building, or similarly looking cubicles in an office environment). This can be again understood as a failure of data association, where we mistakenly associate the loop closure measurement to two incorrectly chosen robot poses.

Note that outliers are not only caused by incorrect data associations, but can also be caused by violations of the assumptions made in the SLAM approach. For instance, the majority of SLAM approaches assume landmarks to be static, hence detections of a moving object —even when correctly associated to that object— may lead to outlier measurements with large residuals. Similarly, sensor failure and degradation, *e.g.,* a faulty wheel encoder or dust on the camera lens, might contribute to creating outliers in the measurements.

3.1.2 Least-Squares in the Presence of Outliers

In the presence of outliers, the estimate resulting from the least-squares formulation (3.1) can be grossly incorrect. From the theoretical standpoint, the Gaussian noise we assumed for the measurement is "light-tailed", in that it essentially rules out the possibility of measurements with very large error. From a more practical perspective, the outliers lead to terms in the objective function where the residual error $\boldsymbol{r}_i(\boldsymbol{x}) := \|\boldsymbol{z}_{ij} - \boldsymbol{h}(\boldsymbol{p}_i, \boldsymbol{\ell}_j)\|_{\boldsymbol{\Sigma}}$ is very large, when evaluated near the ground truth. Since the residuals are squared in the objective of the optimization, *i.e.,* the objective is $\sum_i \boldsymbol{r}_i(\boldsymbol{x})^2$, these residuals have a disproportionately large impact on the cost, and the optimization focuses on minimizing the large terms induced by the outliers rather than making good use of the remaining (inlier) measurements.

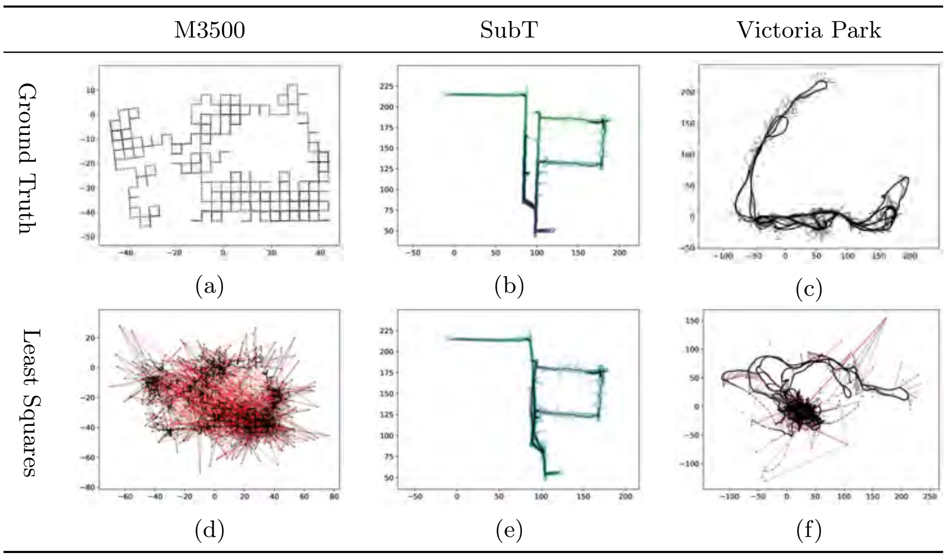


Figure 3.1 SLAM problems with outliers: (a)-(c) Ground truth trajectories for the M3500, SubT, and Victoria Park datasets. (d)-(f) Trajectory estimates obtained with the least-squares formulation in the presence of outliers. Inlier measurements are visualized as gray edges, while outliers are visualized as red edges. In the SubT dataset, we also visualize a dense map built from the SLAM pose estimate.

To illustrate this point, Figure 3.1 shows results for three SLAM problems with outliers. The first column is a simulated pose-graph optimization benchmark, known as M3500, with poses arranged in a grid-like configuration; the dataset includes 3500 2D poses and 8953 measurements. The second column is a real-world pose-graph dataset, denoted as SubT, collected in a tunnel during the DARPA Subterranean Challenge [307]; the dataset includes 682 3D poses and 3278 measurements. The third column is a real-world landmark-SLAM dataset, known as Victoria Park [815]; the dataset includes 7120 2D poses and landmarks, and 17728 measurements. Fig-

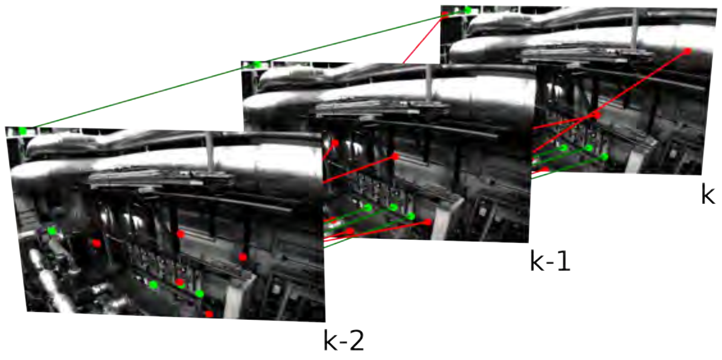


Figure 3.2 Feature tracking across three frames (collected at time $k-1$, k, and $k+1$) in a visual SLAM problem. Inliers are visualized in green, while outliers are visualized in red.

ure 3.1(a)-(c) show the ground truth trajectories for the three problems. Figure 3.1 (d)-(f) show the estimate produced by the least-squares formulation in the presence of outliers. In particular, for M3500 and Victoria Park we add 15% random outliers to the (loop closures or landmark) measurements, while the SubT dataset already includes outliers. In the figure, we visualize outlier measurements in red. We observe that the presence of outliers leads to completely incorrect trajectories and map estimates. Moreover, the outliers often expose perceptual aliasing in the environment: for instance, the two similarly looking vertical corridors in the middle of the SubT dataset induce many spurious loop closures, which mislead the back-end to create a map with a single vertical corridor.

3.2 Detecting and Rejecting Outliers in the SLAM Front-end

The main role of the SLAM front-end is to extract intermediate representations or (pseudo-)measurements —which will be converted into factors for the back-end— from the raw sensor data. Typical SLAM front-ends accomplish this by first computing an initial set of measurements (possibly corrupted by many outliers) and then post-processing the initial set to remove outliers. This section discusses two approaches to reject outliers in the SLAM front-end.

3.2.1 RANdom SAmple Consensus (RANSAC)

RANSAC is a well-established tool for outlier rejection [336] and is a key component of many landmark-based SLAM systems. In order to understand what RANSAC is and its role in SLAM, consider a landmark-based visual SLAM approach.

Example 3.1 (Outliers in landmark-based visual SLAM) A landmark-based (or feature-based) visual-SLAM approach extracts 2D feature points in each image and then associates them across consecutive frames using either optical-flow-based feature tracking or descriptor-based feature matching (Figure 3.2). In particular, at time k, the approach detects 2D feature points and matches them with corresponding points observed in the previous frame (say, at time $k-1$); the matching pixels are typically referred to as *2D-2D correspondences*. Due to inaccuracies of optical flow or descriptor-based matching, this initial set of correspondences might contain outliers. Therefore, it is important to filter out gross outliers before passing them to the back-end, which estimates the robot poses and landmark positions.

RANSAC is a tool to quickly detect and remove outliers in the correspondences before passing them to the back-end. Detecting outliers relies on two key insights. The first insight is that in SLAM problems, inlier correspondences must satisfy geometric constraints. For instance, in our example, inlier correspondences picture the observed pixel motion of static 3D points as the camera moves. The resulting pixel motion cannot be arbitrary, but must follow a precise geometric constraint, known as the *epipolar constraint*, which dictates how corresponding pixels in two frames are related depending on the camera motion. In particular, for calibrated cameras, the epipolar constraint imposes that corresponding pixels $\mathbf{z}_i(k-1), \mathbf{z}_i(k)$ —picturing landmark i at time $k-1$ and k, respectively— satisfy

$$\mathbf{z}_i(k-1)^{\mathsf{T}} \; \left([\boldsymbol{t}_k^{k-1}]_\times \, \boldsymbol{R}_k^{k-1}\right) \; \mathbf{z}_i(k) = 0, \tag{3.2}$$

where $\boldsymbol{t}_k^{k-1}$ and $\boldsymbol{R}_k^{k-1}$ are the relative position and rotation describing the (unknown) motion of the camera between time $k-1$ and k.[2] More generally, if we denote the i-th correspondence as $\mathbf{z}_i$ (in the example above, $\mathbf{z}_i = \{\mathbf{z}_i(k-1), \mathbf{z}_i(k)\}$), these geometric constraints are in the form

$$C(\mathbf{z}_i, \boldsymbol{x}) \leq \gamma, \tag{3.3}$$

which states that the correspondences have to satisfy some inequality, which is possibly a function of the unknown state $\boldsymbol{x}$; in (3.3) the parameter γ on the right-hand-side is typically tuned to account for the presence of noise. For instance, while ideally the epipolar constraint in (3.2) is exactly satisfied, in practice it might have small errors since the pixel detections are noisy, and hence we would relax the constraint to only require $\left|\mathbf{z}_i(k-1)^{\mathsf{T}} \; \left([\boldsymbol{t}_k^{k-1}]_\times \boldsymbol{R}_k^{k-1}\right) \mathbf{z}_i(k)\right| \leq \gamma$, for some small γ.

The second insight is that —assuming we do not have too many outliers— we can find the inliers as the largest set of correspondences that satisfy the geometric

[2] Clearly, different problems will have different geometric constraints, but luckily there is a well-established literature in robotics and computer vision, that studies geometric constraints induced by different types of sensor measurements. The example in this section considers 2D-2D correspondences, and the corresponding constraints have been studied in the context of 2-view geometry in computer vision, see [444].

constraint (3.3) for some $\boldsymbol{x}$:

$$\begin{aligned} \mathsf{S}^*_{\mathrm{CM}} &= \underset{\boldsymbol{x}, \mathsf{S} \subset \mathsf{M}}{\operatorname{argmax}} \, |\mathsf{S}| \\ \text{s.t.} \quad & C(\mathbf{z}_i, \boldsymbol{x}) \le \gamma, \quad \forall i \in \mathsf{S} \end{aligned} \tag{3.4}$$

where M is the set of initial putative correspondences, and $|\mathsf{S}|$ denotes the cardinality (number of elements) in the subset S [735]. In words, the optimization (3.4) looks for the largest subset S of the set of putative correspondences M, such that measurements in S satisfy the geometric constraints for the same value of $\boldsymbol{x}$. Intuitively, problem (3.4) captures the intuition that the inliers (estimated by the set S) must "agree" on the same $\boldsymbol{x}$ (*e.g.,* they must all be consistent with the actual motion of the robot). Problem (3.4) is known as *consensus maximization* in computer vision. Note that (3.4) does not require solving the entire SLAM problem (which might involve many poses and landmarks), since it only involves a small portion of the SLAM state; for instance, the epipolar constraint (3.2) only involves the relative pose between two frames rather than the entire SLAM trajectory. At the same time, (3.4) is still a hard combinatorial problem, which clashes with the fast runtime requirements of typical SLAM front-ends. Therefore, rather than looking for exact solutions to (3.4), it is common to resort to quick heuristics to approximately solve (3.4).

RANdom SAmple Consensus (RANSAC) is probably the most well-known approach to find an approximate solution to the consensus maximization problem in (3.4). RANSAC builds on the key assumption that $\boldsymbol{x}$ in (3.4) is relatively low-dimensional and can be estimated from a small set of measurements (the so-called *minimal set*), using fast estimators (the so called *minimal solvers*).[3] For instance, in our visual SLAM example, one can estimate the relative motion between two camera frames using only 5 pixel correspondences, using Nister's 5-point method [816]. Then the key idea behind RANSAC is that, instead of exhaustively checking every possible subset $\mathsf{S} \subset \mathsf{M}$, one can *sample* minimal sets of measurements looking for inliers. More in detail, RANSAC iterates the following three steps:

1 Sample a subset of n correspondences, where n is the size of the minimal set for the problem at hand;[4]
2 Compute an estimate $\hat{\boldsymbol{x}}$ from the n sampled correspondences using a minimal solver;[5]
3 Select the correspondences $\mathsf{S} \subset \mathsf{M}$ that satisfy the geometric constraint $C(\mathbf{z}_i, \hat{\boldsymbol{x}}) \le$

3 The development of minimal solvers can be considered a sub-area of computer vision research, hence for typical problems it is well-understood what is the size of the minimal set and there are well-developed (and typically off-the-shelf) minimal solvers one can use.
4 In our example with pixel correspondences between calibrated camera images, the minimal set has size $n = 5$, since 5 non-collinear measurements are sufficient to determine the pose between two cameras up to scale.
5 In our example, this involves computing the relative motion (up to scale) between time $k - 1$ and k using the 5-point method.

γ for the $\hat{\boldsymbol{x}}$ computed at the previous step. Store the set S if it is larger than the set computed at the previous iterations.

The set S computed in the last step is called the *consensus set* and RANSAC typically stops after computing a sufficiently large consensus set (as specified by a user parameter) or after a maximum number of iterations. RANSAC essentially attempts to sample n inliers from the set of measurements, since these are likely to "agree" with all the other inliers and hence have a large consensus set.

RANSAC is the go-to solution for many outlier-rejection problems. In particular, it quickly converges to good estimates (*i.e.,* good sets of correspondences) in problems with small number of outliers and small minimal sets. Assuming that the probability of sampling an inlier from the set of measurements is ω,[6] it is easy to conclude that the expected number of iterations RANSAC requires for finding a set of inliers is $\frac{1}{\omega^n}$. For instance, when $n = 5$ and $\omega = 0.7$ (*i.e.,* 70% of the measurements are inliers), the expected number of iterations is less then 10. This, combined with the fact that non-minimal solvers are extremely fast in practice (allowing even thousand of iterations in a handful of milliseconds), makes RANSAC extremely appealing. Moreover, RANSAC also provides an estimate $\hat{\boldsymbol{x}}$ (*e.g.,* the robot odometry), that can be useful as an initial guess for the back-end.

On the downside, RANSAC may not be the right approach for all problems. In particular, the expected number of iterations becomes impractically large when the number of inliers is small or when the minimal set is large; for instance, when $n = 10$ and $\omega = 0.1$, the expected number of iterations to find a set of inliers becomes 10^{10}, and terminating RANSAC after a smaller number of iterations is likely to return incorrect solutions (*i.e.,* incorrect $\hat{\boldsymbol{x}}$ and correspondences). As we discuss in the next section, in context of many SLAM problems, the assumptions of having many inliers and small minimal sets are not always valid.

3.2.2 Graph-theoretic Outlier Rejection and Pairwise Consistency Maximization

As we mentioned, RANSAC is very effective when the number of outliers is reasonable (say, below 70%) and the size of the minimal set is small (say, less than 8). However, environments with severe perceptual aliasing might have very high number of outliers. Moreover, not all the problems we are interested in have a fast minimal solver with a small minimal set. For instance, if we consider a pose-graph SLAM problem with N nodes, the minimal set must include at least $N - 1$ measurements (forming a spanning tree of the pose-graph), and N is typically in the thousands.

For these reasons, this section introduces an alternative approach, known as *pair-*

[6] Assuming that samples are drawn uniformly at random, ω can be thought of as the fraction of measurements that are inliers.

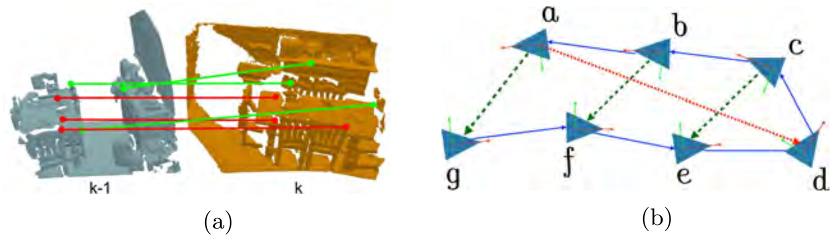


Figure 3.3 (a) 3D-3D correspondences from two RGB-D scans representing two partial views of a scene. The green lines indicate inlier correspondences and the red lines indicate outlier correspondences. (b) Pose graph with outliers in the loop closures. The dashed green lines indicate inlier loop closures while the dotted red line is an outlier loop closure.

wise consistency maximization (PCM), that, rather than sampling minimal sets, seeks to find the largest set of measurements that are internally "consistent" with one another, using graph theory. This approach can be used to sort through sets of measurements with upwards of 90% outliers and prune gross outliers before passing them to the back-end. The approach was initially proposed in [735] and extended beyond pairwise consistency in [1011, 340].

The key insight behind PCM is that for many problems one can define *consistency functions* that capture whether a pair of measurements are consistent with each other. Let's elucidate on this point with two examples.

Example 3.2 (Consistency Function in landmark-based visual SLAM with RGB-D cameras) A landmark-based visual-SLAM approach with RGB-D cameras extracts 3D feature points in each RGB-D frame and then associates them across consecutive frames (Figure 3.3(a)). In particular, at time k, the approach detects 3D feature points and matches them with corresponding points observed in the previous frame (say, at time $k-1$); the matching 3D points are typically referred to as *3D-3D correspondences*. We observe that the 3D points collected at time k and $k-1$ ideally correspond to the same set of 3D static points observed from two different viewpoints; therefore, the distance between a pair of corresponding points $\{\boldsymbol{z}_i(k-1), \boldsymbol{z}_j(k-1)\}$ and $\{\boldsymbol{z}_i(k), \boldsymbol{z}_j(k)\}$ has to be constant over time (up to noise):

$$\left| \|\boldsymbol{z}_i(k-1) - \boldsymbol{z}_j(k-1)\| - \|\boldsymbol{z}_i(k) - \boldsymbol{z}_j(k)\| \right| \leq \gamma \tag{3.5}$$

We observe that contrary to the geometric constraints used in RANSAC, the consistency function (3.5) (i) does not depend on the state, hence it can be evaluated directly without the need for a minimal solver, and (ii) involves a pair of correspondences regardless of the size of the minimal set. While the previous example could also be solved with RANSAC,[7] let us now consider a higher dimensional problem.

[7] Motion estimation from 3D-3D correspondences admits a fast 3-point minimal solver, *e.g.,* Horn's method [485].

Example 3.3 (Consistency Function in pose-graph SLAM) Consider a pose-graph SLAM problem where loop closures might contain outliers due to place recognition failure and perceptual aliasing; we assume the odometry is reliable and outlier free. In order to understand if two loop closures are consistent with each other, we observe that in the noiseless case, pose measurements along cycles in the graph must compose to the identity (Figure 3.3(b)).[8] Therefore, a pair of loop closures $\boldsymbol{T}_{ab}$ (between poses a and b) and $\boldsymbol{T}_{cd}$ (between poses c and d) must satisfy:

$$\operatorname{dist}(\boldsymbol{T}_{ab} \cdot \bar{\boldsymbol{T}}_{bc} \cdot \boldsymbol{T}_{cd} \cdot \bar{\boldsymbol{T}}_{da}, \mathbf{I}) \leq \gamma \tag{3.6}$$

where $\bar{\boldsymbol{T}}_{bc}$ and $\bar{\boldsymbol{T}}_{da}$ are the chain of odometry measurements from node b to node c, and from node d to node a, respectively, and dist is a suitable distance function that measures how far is $\boldsymbol{T}_{ab} \cdot \bar{\boldsymbol{T}}_{bc} \cdot \boldsymbol{T}_{cd} \cdot \bar{\boldsymbol{T}}_{da}$ from the identity pose. As usual, γ is a parameter chosen to account for the noise: measurements along a loop might not compose to the identity due to noise in the odometry and loop closures.[9]

More generally, a *consistency function* is a function relating two measurements and that have to satisfy a given constraint. For a pair of measurements $\mathbf{z}_i$ and $\mathbf{z}_j$, the resulting *pairwise consistency constraints* are in the form:

$$F(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma, \tag{3.7}$$

where F is the consistency function, and γ is a user-specified parameter that accounts for measurement noise. We remark that the pairwise consistency constraint are state independent, hence they can be efficiently checked without resorting to a minimal solver by just inspecting every pair of measurements.

Using (3.7), we can formulate an alternative approach for outlier rejection, which selects the largest set of measurements that are pairwise consistent:

$$\begin{aligned} \mathsf{S}^*_{\mathrm{PCM}} &= \operatorname*{argmax}_{\mathsf{S} \subset \mathsf{M}} |\mathsf{S}| \\ \text{s.t.} \quad & F(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma, \quad \forall i, j \in \mathsf{S} \end{aligned} \tag{3.8}$$

Problem (3.8) looks for the largest subset S of measurements such that every pair of measurements in S are pairwise consistent. We refer to this as the *pairwise consistency maximization* (PCM) problem. This problem is still combinatorial in nature, but appears slightly easier than (3.4): the problem does not involve $\boldsymbol{x}$, and the constraints $F(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma$ can be pre-computed for every pair (i, j) in M. Furthermore, the problem admits a graph-theoretic interpretation, which allows solving (3.4) using well-established tools from graph theory, namely, *maximum clique* algorithms.

In order to draw a connection between problem (3.8) and graph theory, let us visualize the outlier-rejection problem as a graph $\mathcal{G}$, where the nodes of the graph are the putative measurements $i \in \mathsf{M}$ and an edge exists between two nodes i and j if

[8] Intuitively, if we walk back along a loop in the environment we come back to our initial location.
[9] In practice, one would select γ to account for the size of the loop: intuitively, longer loops will accumulate more noise, see [735, 306].

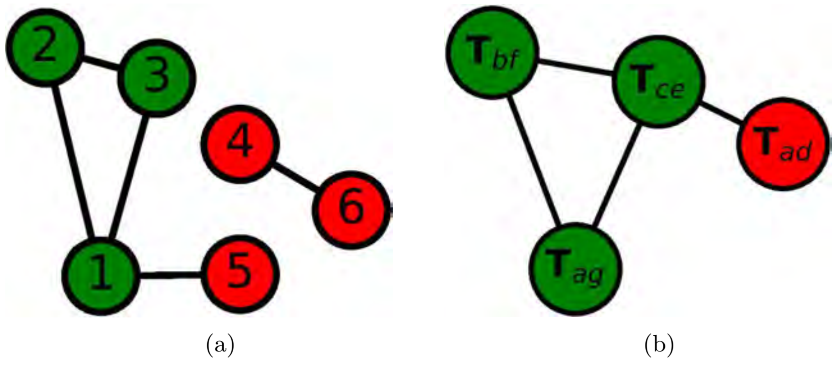


Figure 3.4 (a) Consistency graph of the 3D-3D correspondences example in Figure 3.3(a). (b) Consistency graph of the loop closures for the pose-graph example in Figure 3.3(b)

$F(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma$. This is typically called the *consistency graph*. Now problem (3.8) asks to select the largest subset of nodes S such that every pair of nodes in S is connected by an edge: this is exactly the definition of *maximum clique* of a graph. More in detail, a *clique* in graph theory is a subset of nodes in which every pair of nodes has an edge between them, and the *maximum clique* is the largest such subset of nodes in the graph. Therefore, the solution to problem (3.8) is the maximum clique of the consistency graph $\mathcal{G}$. This graph theoretic connection is really useful in practice, since the problem of finding the maximum clique for a given graph is a well-studied problem in graph theory and is called the maximum clique problem. The maximum clique problem is an NP-hard problem [1194] and hard to approximate [1309, 330], meaning that finding a solution arbitrarily close to the true solution is also NP-hard. However, dozens of potential solutions have been proposed, some of which can handle significantly sized problems depending on the density of the graph. The majority of proposed methods can be classified as either exact or heuristic-based methods. All of the exact algorithms are exponential in complexity and are usually based on branch and bound, while the heuristic algorithms often try to exploit some type of structure in the problem, making them faster, at the expense of not necessarily guaranteeing the optimal solution [1194]. Relatively recent works, *e.g.,* [859], propose maximum clique algorithms that are parallelizable and able to quickly find maximum cliques in sparse graphs.

In summary, solving the PCM problem using a maximum clique algorithm involves the following steps:

1 Select a Consistency Function F for the problem at hand.
2 Evaluate the Consistency Function for every pair of putative measurements $(i, j) \in \mathsf{M}$, and create a consistency graph with edges between i and j when $F(\mathbf{z}_i, \mathbf{z}_j) \leq \gamma$.
3 Solve for the Maximum Clique of the Consistency Graph using exact or approximate maximum clique algorithms.

4 Return measurements S in the (possibly approximate) maximum clique.

We remark that the choice of consistency function is problem-dependent. Moreover, choosing a good consistency function might largely influence the quality of the outlier rejection. For instance, one could select a dummy function $F(\mathbf{z}_i, \mathbf{z}_j) = 0$ which always returns zero regardless of the arguments; such a function would not allow rejecting any outliers, hence making PCM ineffective. On the other hand, if we make the function such that only the inliers can pass the test, then we would exactly reject all the outliers. A selection of potential consistency functions for broad variety of geometric problems is discussed in [1011, 340].

Before concluding this section a few remarks are in order. While we observed that PCM has the ability to handle a large number of outliers compared to RANSAC and is more suitable for certain problems (*e.g.,* pose-graph SLAM), the trade-offs between PCM and RANSAC are more nuanced. RANSAC evaluates the consistency of individual measurements using an estimate computed by a minimal solver; PCM, on the other hand, evaluates the consistency of a set of measurements to each other in a pairwise manner. In certain cases, RANSAC's individual consistency is insufficient to evaluate the set of measurements as a whole: this is often the case in pose-graph optimization where individual consistency of a pair of loop-closure measurements does not necessarily ensure pairwise consistency of the two loop-closures.[10] On the other hand, for certain problems such as 3D-3D pose estimation (Example 3.2), the pairwise consistency function (3.5) used in PCM might be more permissive then RANSAC and lead to classifying certain outliers as inliers.

In the context of PCM, it is also important to note that exact maximum clique solvers tend to be slow in dense consistency graphs (*i.e.,* when many pairs of measurements are consistent), hence heuristic-based maximum-clique solutions may be a better choice for certain problems. Finally, for certain problems, it might be hard to design a suitable consistency function; for instance, for 2D-2D correspondences, there is no easy way to rigorously design a general consistency function due to the lack of suitable invariances (see discussion in [1011]).

3.3 Increasing Robustness to Outliers in the SLAM Back-end

Front-end outlier rejection, including both RANSAC and PCM, might still miss outliers and pass an outlier-contaminated set of measurements to the back-end.[11] As we have seen in Section 3.1.2, a handful of outliers can lead to completely wrong

[10] This is especially pronounced in the context of the multi-robot pose-graph optimization — where the goal is to estimate the trajectory of two (or more) robots jointly within a single pose-graph. In these contexts in particular, PCM has been shown to dramatically outperform RANSAC [735].

[11] Intuitively, both the geometric constraints (3.3) —even when evaluated at the ground truth $\boldsymbol{x}$— and the pairwise consistency constraints (3.7) are necessary (but not sufficient) conditions for measurements to be inliers. Moreover, consensus maximization and PCM are often solved with approximation algorithms that do not guarantee an optimal selection of the inliers.

results when using standard least squares estimation. Therefore, it is important to enhance the back-end to be robust to remaining outliers.

In Section 3.1.2, we observed that the use of squared residuals "amplifies" the impact of outlying measurements on the cost function. In this section we slightly modify the objective function in the SLAM optimization to regain robustness to outliers, following the standard theory of M-Estimation in robust statistics [502].

M-Estimation ("Maximum-likelihood-type Estimation") is a framework for robust estimation and suggests replacing the squared loss in eq. (3.1) with a suitably chosen *robust loss* function ρ:

$$\boldsymbol{x}^{\mathrm{MAP}} = \arg\min_{\boldsymbol{x}} \sum_i \boldsymbol{r}_i(\boldsymbol{x})^2 \quad \Longrightarrow \quad \boldsymbol{x}^{\mathrm{MEST}} = \arg\min_{\boldsymbol{x}} \sum_i \rho\left(\boldsymbol{r}_i(\boldsymbol{x})\right). \tag{3.9}$$

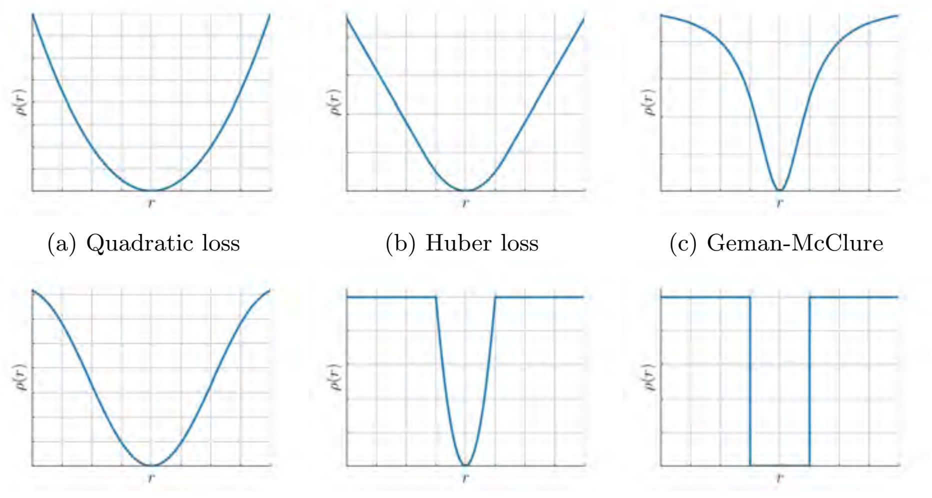
The key requirement for the robust loss ρ is to be a non-negative function and grow less than quadratically for large residuals; in other words, robust loss functions need to have derivative $\frac{\partial\rho(\boldsymbol{r}_i)}{\partial\boldsymbol{r}_i} \ll \frac{\partial\|\boldsymbol{r}_i\|^2}{\partial\boldsymbol{r}_i} = 2\boldsymbol{r}_i$ as $\boldsymbol{r}_i$ becomes large; in many cases, it is desirable for $\frac{\partial\rho(\boldsymbol{r}_i)}{\partial\boldsymbol{r}_i}$ to approach zero as $\boldsymbol{r}_i$ becomes large. To elucidate this requirement, consider the case where we solve $\min_{\boldsymbol{x}} \sum_i \rho\left(\boldsymbol{r}_i(\boldsymbol{x})\right)$ using gradient descent. Using the chain rule, the gradient of the objective $f(\boldsymbol{x}) \doteq \sum_i \rho\left(\boldsymbol{r}_i(\boldsymbol{x})\right)$ becomes:

$$\frac{\partial f}{\partial \boldsymbol{x}} = \sum_i \frac{\partial\rho\left(\boldsymbol{r}_i\right)}{\partial\boldsymbol{r}_i} \cdot \frac{\partial\boldsymbol{r}_i(\boldsymbol{x})}{\partial\boldsymbol{x}} \tag{3.10}$$

From (3.10), it is clear that if we start for a good initial guess (*i.e.,* relatively close to the ground truth), outlier measurements will have large residual and hence very small $\frac{\partial\rho(\boldsymbol{r}_i)}{\partial\boldsymbol{r}_i}$, thus having a minor influence on the overall descent direction. Hence they will have almost no influence in the estimation. Indeed, the function $\psi(\boldsymbol{r}_i) := \frac{\partial\rho(\boldsymbol{r}_i)}{\partial\boldsymbol{r}_i}$ is typically referred to as the *influence function* [92].

Rather than a single choice of robust loss function, the robust estimation literature provides a "menu" of potential choices. Figure 3.5 lists common choices of loss functions. This list includes common robust losses, such as Huber, Geman-McClure, Tukey's biweight and the truncated quadratic loss, and also includes a more radical choice, named *maximum consensus* loss. The latter is not typically listed among the loss functions in the robust estimation literature, but we mention it here, since it connects back to the consensus maximization problem we discussed in (3.4).[12] The choice of robust loss is fairly problem-dependent [562, 1079]. For instance, loss functions with hard cut-offs (*e.g.,* the truncated quadratic loss, where there is a sudden transition between the quadratic and the "flat" portion of the function) are preferable when a reasonable threshold for the cut-off (*i.e.,* the maximum error expected from the inliers) is known. One also has to take into account computational

[12] Intuitively, the maximum consensus loss "counts" the outliers in the estimation problem, since it is constant (typically equal to 1) for large residuals and zero for small residuals. Therefore, minimizing such loss leads to an estimate that produces the least number of outliers. This is the same as solving the consensus maximization problem in (3.4).



(a) Quadratic loss (b) Huber loss (c) Geman-McClure

(d) Tukey's biweight (e) Truncated Quadratic (f) Maximum Consensus loss

Figure 3.5 Quadratic loss and examples of robust loss functions. The shape of the robust loss functions is controlled by a parameter that controls the separation between inliers and outliers.

considerations. For instance, Huber is often used in bundle adjustment problems since it is a convex function and is better-behaved during the optimization, despite leaving non-zero influence for the outliers (the influence becomes zero only if the loss is constant for large residuals). On the other hand, the truncated quadratic and maximum consensus losses are known to be particularly insensitive to outliers, but they often require ad-hoc solvers.[13]

Figure 3.6(g)-(l) show the SLAM trajectories obtained by applying gradient descent to two of the robust losses mentioned above: the Huber loss and the truncated quadratic loss. Here we consider the same datasets used in Figure 3.1. We implemented the gradient descent solver using GTSAM's NonlinearConjugateGradientOptimizer [261] with the gradientDescent flag enabled, and using robust noise models to instantiate the robust loss functions. We set the maximum number of iterations and the stopping conditions thresholds (relative and absolute tolerance) to 10000 and 10^{-7}, respectively. All other parameters were left to the default GTSAM values. In the figure, an edge is colored in gray if it is correctly classified as inlier or outlier by the optimization (*i.e.,* an inlier that falls in the quadratic region of the Huber or truncated quadratic loss); it is colored in red if it is an outlier incorrectly classified as an inlier (a "false positive"); it is colored in blue if it is an inlier incorrectly classified as an outlier (a "false negative"). Compared to

[13] For instance, RANSAC (Section 3.2.1) optimizes the maximum consensus loss via sampling (an option that is only viable for the low-dimensional optimization problems arising in the front-end), while graduated non-convexity allows optimizing a broad variety of losses including the truncated quadratic loss (more on this in Section 3.3.4 below).

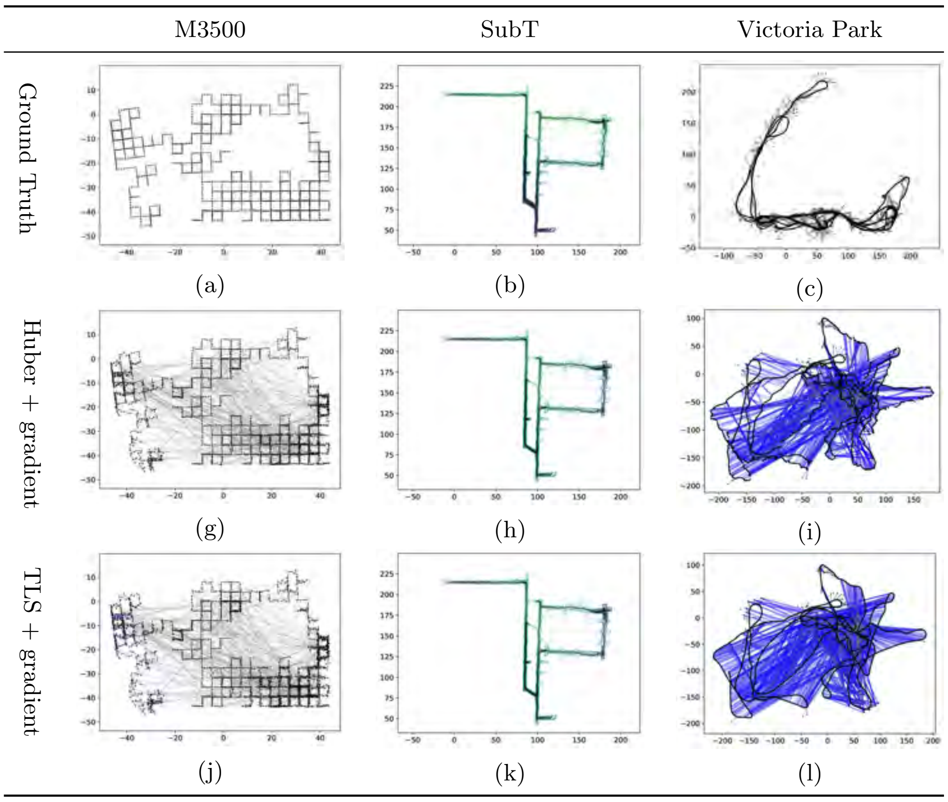


Figure 3.6 Solving SLAM problems with outliers using robust loss functions and gradient descent: (a)-(c) Ground truth trajectories for the M3500, SubT, and Victoria Park datasets. (d)-(f) Trajectory estimates obtained using gradient descent and Huber loss. (h)-(l) Trajectory estimates obtained using gradient descent and truncated quadratic loss. Measurements are visualized as colored edges. In particular, an edge is colored in gray if it is correctly classified as inlier or outlier by the optimization (*i.e.,* an inlier that falls in the quadratic region of the Huber or truncated quadratic loss); it is colored in red if it is an outlier incorrectly classified as an inlier (a "false positive"); it is colored in blue if it is an inlier incorrectly classified as an outlier (a "false negative").

(non-robust) least squares optimization (Figure 3.1(d)-(f)), we note that the use of the robust losses allows us to quickly regain robustness to outliers in the case of the M3500 and SubT datasets, but a simple gradient descent method may still fail to correctly optimize heavily non-convex functions such as the truncated quadratic cost, as in the case of the Victoria Park dataset. We will address this issue with a better solver, based on *graduated non-convexity*, below. Moreover, while gradient descent already improves performance in many of the instances as shown in Figure 3.6, it has slow convergence tails. For instance, in our experiments, it often takes thousands of iterations to converge. Therefore, in the rest of this section we discuss more advanced solvers, that improve both convergence quality and speed.

As a concluding remark before delving into more advanced solvers, we observe that while it might seem that we gave up on our probabilistic framework when switching to robust loss functions, it is actually possible to derive several robust losses by applying MAP estimation to heavy-tailed noise distributions. For instance, the truncated quadratic loss results from MAP estimation when assuming the noise follow a max-mixture distribution between a Gaussian density (describing the inliers) and a uniform distribution (describing the outliers) [41].

3.3.1 Iteratively Reweighted Least Squares

M-Estimation replaces the least-squares loss by a robust loss ρ in (3.9) — a function that grows sub-quadratically for large residuals. This comes with two prices. First, we lose the efficient solutions already developed for least-squares formulations; for instance, the Gauss-Newton and the Levenberg-Marquardt methods are designed for least squares problems. Second, due to the typical non-convex landspace of M-Estimation, iterative solvers (*e.g.,* based on gradient descent) are sensitive to the quality of initialization and often converge to undesired suboptimal estimates. In this section, we introduce a popular algorithm for solving M-Estimation called *iteratively reweighted least squares* (IRLS) which, as the name suggests, allows reusing the efficient least-squares solvers. In the next section, we introduce graduated non-convexity as a technique to improve the convergence of IRLS.

The basic idea behind IRLS is to optimize (3.9) by solving a *weighted* least squares problem at each iteration

$$\boldsymbol{x}^{(t+1)} = \arg\min_{\boldsymbol{x}} \sum_i w_i(\boldsymbol{x}^{(t)}) r_i^2(\boldsymbol{x}), \tag{3.11}$$

where the weights w_i's depend on the estimate $\boldsymbol{x}^{(t)}$ from the last iteration. We wish the iterative solutions $\boldsymbol{x}^{(t)}$ to converge to the optimal solution of M-Estimation (3.9). This implies that the gradient of the robust loss ρ, shown in (3.10), must match the gradient of the loss in (3.11). By writing down the gradient of (3.11) as

$$\sum_i 2 w_i(\boldsymbol{x}^{(t)}) r_i(\boldsymbol{x}) \frac{\partial r_i(\boldsymbol{x})}{\partial \boldsymbol{x}}$$

and comparing it to (3.10), we obtain the IRLS weight update rule

$$w_i(\boldsymbol{x}^{(t)}) = \frac{1}{2 r_i(\boldsymbol{x}^{(t)})} \frac{\partial \rho(r_i(\boldsymbol{x}^{(t)}))}{\partial r_i(\boldsymbol{x}^{(t)})} = \frac{\psi(r_i(\boldsymbol{x}^{(t)}))}{2 r_i(\boldsymbol{x}^{(t)})}, \tag{3.12}$$

where we recall that $\psi(r_i) := \frac{\partial \rho(r_i)}{\partial r_i}$ is the influence function. Therefore, IRLS alternates computing the weights $w_i(\boldsymbol{x}^{(t)})$ for each measurement i, with performing an optimization step (*i.e.,* a Gauss-Newton or Levengberg-Marquardt iteration) on the weighted least squares problem (3.11).

Figure 3.7 shows the performance of IRLS on the M3500, SubT, and Victoria Park datasets. IRLS converges in tens of iterations and is typically much faster than gradient descent; for instance, gradient descent requires around 5 seconds to optimize the Huber loss in our M3500 experiments, while IRLS took less than 1.5 seconds. On the other hand, this faster convergence often comes at the cost of a slightly decreased accuracy, as can be seen by comparing Figure 3.7 and Figure 3.6. The convergence properties of the update rule (3.12) has been studied in [17, 823].

3.3.2 Black-Rangarajan Duality

The weight update rule (3.12) is widely used in practice, but its derivation was somewhat heuristic. It also has the issues that (3.12) is not well-defined at the non-differentiable points of ρ (*e.g.,* the cut-off point of the truncated quadratic loss). We now introduce a more principled framework, namely the *Black-Rangarajan (B-R) duality* [92], to solve M-Estimation using IRLS.

Let us present the intuition of B-R duality using the truncated quadratic loss

$$\rho(r_i(\boldsymbol{x})) := \min\{r_i^2(\boldsymbol{x}), \beta_i^2\}, \tag{3.13}$$

where β_i^2 is a bound on the i-th residual such that the i-th measurement is an inlier if $r_i^2(\boldsymbol{x}) \leq \beta_i^2$ and an outlier otherwise. We observe that the cost in (3.13) can be equivalently written as a sum of two terms by introducing a new weight variable $w_i \in [0, 1]$

$$\rho(r_i(\boldsymbol{x})) := \min_{w_i \in [0,1]} w_i r_i^2(\boldsymbol{x}) + (1 - w_i)\beta_i^2, \tag{3.14}$$

where the first term is exactly the weighted least squares, and the second term is a function of w_i that does not depend on $\boldsymbol{x}$. With (3.14), the M-Estimation problem (3.9) with a truncated quadratic loss can be reformulated as

$$\min_{\substack{\boldsymbol{x} \\ w_i \in [0,1], i=1,\ldots,N}} \sum_i \left[w_i r_i^2(\boldsymbol{x}) + (1 - w_i)\beta_i^2 \right], \tag{3.15}$$

where we have introduced one w_i for each measurement residual $r_i(\boldsymbol{x})$. Problem (3.15) is easy to interpret: $w_i = 1$ implies $r_i^2(\boldsymbol{x}) \leq \beta_i^2$ and the i-th measurement is an inlier; $w_i = 0$ implies $r_i^2(\boldsymbol{x}) > \beta_i^2$ and the i-th measurement is an outlier. Moreover, all the residuals with $w_i = 0$ are effectively discarded from the optimization (3.15) and hence robustness is ensured.

B-R duality generalizes the derivation above to a family of robust losses.

Theorem 3.4 (Black-Rangarajan Duality [92]) *Given a robust loss function $\rho(\cdot)$, define $\phi(z) := \rho(\sqrt{z})$. If $\phi(z)$ satisfies $\lim_{z\to 0} \phi'(z) = 1$, $\lim_{z\to\infty} \phi'(z) = 0$, and*

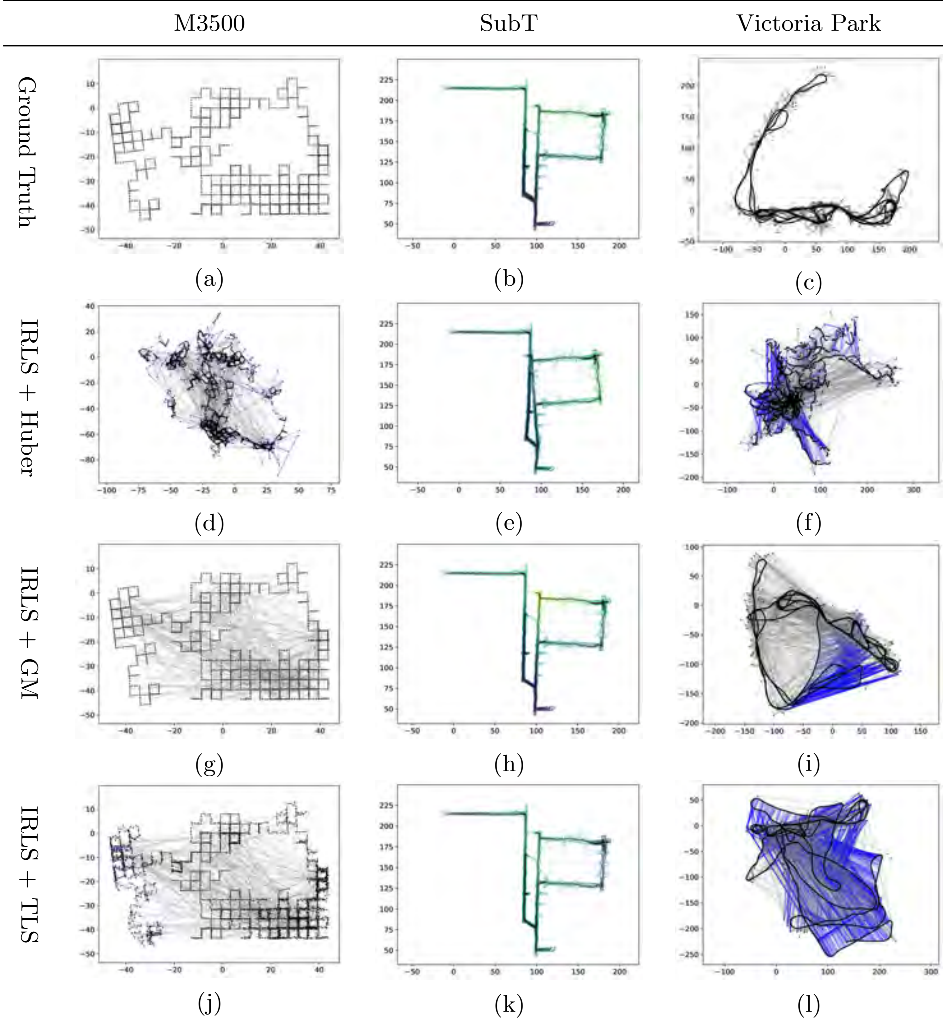


Figure 3.7 Solving SLAM problems with outliers using robust loss functions and Iteratively Re-weighted Least-Squares (IRLS): (a)-(c) Ground truth trajectories for the M3500, SubT, and Victoria Park datasets. (d)-(f) Trajectory estimates obtained using the Huber loss. (g)-(i) Trajectory estimates obtained using the Geman-McClure loss. (j)-(l) Trajectory estimates obtained using the truncated quadratic loss. Measurements are visualized as colored edges. In particular, an edge is colored in gray if it is correctly classified as inlier or outlier by the optimization (*i.e.,* an inlier that falls in the quadratic region of the Huber or truncated quadratic loss); it is colored in red if it is an outlier incorrectly classified as an inlier (a "false positive"); it is colored in blue if it is an inlier incorrectly classified as an outlier (a "false negative").

$\phi''(z) < 0$, then the M-Estimation problem (3.9) is equivalent to

$$\min_{\substack{\boldsymbol{x} \\ w_i \in [0,1], i=1,\ldots,N}} \sum_{i=1}^{N} \left[w_i r_i^2(\boldsymbol{x}) + \Phi_\rho(w_i) \right], \tag{3.16}$$

where $w_i \in [0,1], i = 1, \ldots, N$ are weight variables associated to each residual r_i, and the function $\Phi_\rho(w_i)$, referred to as an outlier process, defines a penalty on the weight w_i whose form is dependent on the choice of robust loss ρ.

In the case of ρ being the truncated quadratic loss, we easily derived from (3.14) that $\Phi_\rho(w_i) = (1-w_i)\beta_i^2$. When ρ takes other forms, [92] provides a recipe to derive $\Phi_\rho(w_i)$. We give an example for the Geman-McClure (G-M) robust loss.

Example 3.5 (B-R Duality for G-M Loss) Consider the G-M robust loss function

$$\rho(r_i(\boldsymbol{x})) = \frac{\beta_i^2 r_i^2(\boldsymbol{x})}{\beta_i^2 + r_i^2(\boldsymbol{x})}, \tag{3.17}$$

where β_i^2 is a noise bound for the i-th residual similar to (3.13). The outlier process associated to (3.17) is

$$\Phi_\rho(w_i) = \beta_i^2(\sqrt{w_i} - 1)^2. \tag{3.18}$$

To verify the correctness of (3.18), consider

$$\min_{w_i \in [0,1]} w_i r_i^2(\boldsymbol{x}) + \Phi_\rho(w_i), \tag{3.19}$$

whose optimal solution is (via setting the gradient of (3.19) to zero)

$$w_i^\star = \left(\frac{\beta_i^2}{r_i^2(\boldsymbol{x}) + \beta_i^2} \right)^2. \tag{3.20}$$

Plugging (3.20) back to the objective of (3.19) recovers the G-M robust loss (3.17).

3.3.3 Alternating Minimization

With the introduction of B-R duality, the IRLS algorithm naturally comes out using a common optimization strategy called *alternating minimization* [1116, 87]. The idea is that, although it is difficult to jointly optimize both $\boldsymbol{x}$ and $w_i \in [0,1], i = 1, \ldots, N$ in (3.16), optimization of either $\boldsymbol{x}$ or w_i's when fixing the other is easy. To see this, observe that when w_i's are fixed, problem (3.16) becomes a weighted least squares; analogously, when $\boldsymbol{x}$ is fixed, problem (3.16) becomes

$$\min_{w_i \in [0,1], i=1,\ldots,N} \sum_{i=1}^{N} \Phi_\rho(w_i) + w_i r_i^2(\boldsymbol{x}),$$

which splits into N subproblems, each optimizing a scalar w_i

$$\min_{w_i \in [0,1]} \Phi_\rho(w_i) + w_i r_i^2(\boldsymbol{x}). \tag{3.21}$$

Problem (3.21) is easy to solve and often admits a closed-form solution. In fact, for the G-M robust loss, the solution of (3.21) is just (3.20). For the truncated quadratic loss, problem (3.21) reads

$$\min_{w_i \in [0,1]} (1 - w_i)\beta_i^2 + w_i r_i^2(\boldsymbol{x})$$

and admits a closed-form solution

$$w_i^\star = \begin{cases} 1 & \text{if } r_i^2(\boldsymbol{x}) < \beta_i^2 \\ 0 & \text{if } r_i^2(\boldsymbol{x}) > \beta_i^2 \\ [0,1] & \text{otherwise} \end{cases} .$$

In summary, the t-th iteration of IRLS in the context of B-R duality alternates between two steps

1 **Variable update**: solve a weighted least squares problem using the current weights $w_i^{(t)}$

$$\boldsymbol{x}^{(t)} \in \arg\min_{\boldsymbol{x}} \sum_i w_i^{(t)} r_i^2(\boldsymbol{x}). \tag{3.22}$$

2 **Weight update**: update the weights using $\boldsymbol{x}^{(t)}$

$$w_i^{(t+1)} \in \arg\min_{w_i \in [0,1]} \Phi_\rho(w_i) + w_i r_i^2(\boldsymbol{x}^{(t)}), i = 1, \ldots, N. \tag{3.23}$$

The weight update rule obtained via B-R duality actually matches the popular weight update rule (3.12). Interestingly, instantiating the above IRLS algorithm in SLAM using the G-M robust loss leads to the *dynamic covariance scaling* algorithm [18], which has been proposed in the context of outlier-robust SLAM.

3.3.4 Graduated Non-Convexity

The previous section leveraged Black-Rangarajan duality and alternating minimization to derive the IRLS framework that alternates in solving (3.22) and (3.23). However, due to the non-convexity of common robust losses, the convergence of the IRLS framework can be highly sensitive to the quality of initialization, *i.e.,* how close is $\boldsymbol{x}^{(0)}$ to the optimal solution of (3.9) or how well does $w_i^{(0)}$ reflect the inlier-outlier membership of each measurement. For example, [1012] showed that IRLS with the truncated quadratic loss and the Geman-McClure loss might fail when there are as little as 10% outliers in the measurements (*cf.* also with our results in Figure 3.7).

In this section, we introduce the *graduated non-convexity* (GNC) scheme that can make IRLS significantly less sensitive to the quality of initialization. Given a robust cost function ρ, the basic idea of GNC is to create a smooth version of ρ, denoted as ρ_μ, using a scalar smoothing factor μ. Tuning μ controls the amount of non-convexity in ρ_μ: ρ_μ is convex at one end of the spectrum and recovers the original ρ at the other end of the spectrum. Let us illustrate this using two examples.

Example 3.6 (GNC Truncated Quadratic Loss) Consider the GNC truncated quadratic loss function

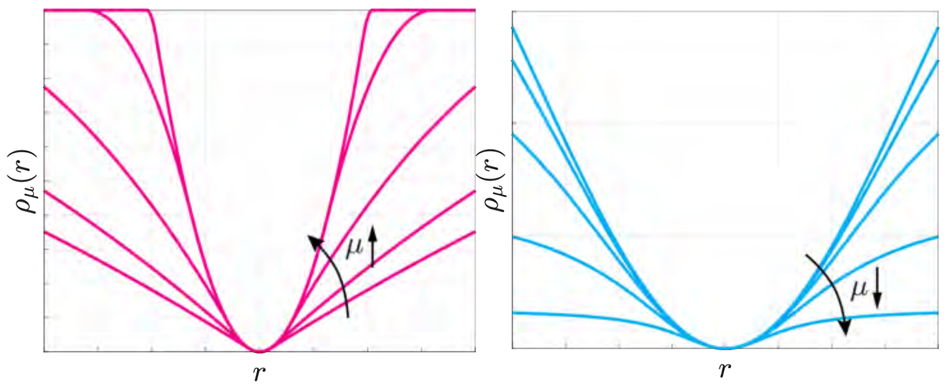
$$\rho_\mu(r_i(\boldsymbol{x})) = \begin{cases} r_i^2(\boldsymbol{x}) & \text{if } r_i^2(\boldsymbol{x}) \in \left[0, \frac{\mu}{\mu+1}\beta_i^2\right] \\ 2\beta_i|r_i(\boldsymbol{x})|\sqrt{\mu(\mu+1)} - \mu(\beta_i^2 + r_i^2(\boldsymbol{x})) & \text{if } r_i^2(\boldsymbol{x}) \in \left[\frac{\mu}{\mu+1}\beta_i^2, \frac{\mu+1}{\mu}\beta_i^2\right] \\ \beta_i^2 & r_i^2(\boldsymbol{x}) \in \left[\frac{\mu+1}{\mu}\beta_i^2, +\infty\right]. \end{cases} \tag{3.24}$$

ρ_μ is convex for μ approaching zero and retrieves the truncated quadratic loss in (3.13) for μ approaching infinity.

Example 3.7 (GNC Geman-McClure Loss) Consider the GNC Geman-McClure loss function

$$\rho_\mu(r_i(\boldsymbol{x})) = \frac{\mu\beta_i^2 r_i^2(\boldsymbol{x})}{\mu\beta_i^2 + r_i^2(\boldsymbol{x})}. \tag{3.25}$$

ρ_μ is convex for μ approaching ∞ and recovers the G-M loss (3.17) when $\mu = 1$.



(a) (b)

Figure 3.8 Graduated Non-Convexity with control parameter μ for (a) Truncated Quadratic loss and (b) Geman-McClure loss. [1218] (©2020 IEEE)

Figure 3.8(a) and (b) plot the GNC truncated quadratic loss and the GNC Geman-McClure loss, respectively. Observe how increasing or decreasing the control parameter μ adds more non-convexity to the function.

One nice property of the smoothed GNC functions in (3.24) and (3.25) is that the B-R duality still applies. For the GNC truncated quadratic loss (3.24), applying

B-R duality leads to the outlier process

$$\Phi_{\rho_\mu}(w_i) = \frac{\mu(1 - w_i)}{\mu + w_i}\beta_i^2.$$

For the GNC Geman-McClure loss (3.25), applying B-R duality leads to the outlier process

$$\Phi_{\rho_\mu}(w_i) = \mu\beta_i^2(\sqrt{w_i} - 1)^2.$$

We are now ready to state the GNC algorithm, which at each iteration performs three steps

1 **Variable update**: solve a weighted least squares problem using the current weights $w_i^{(t)}$

$$\boldsymbol{x}^{(t)} \in \arg\min_{\boldsymbol{x}} \sum_i w_i^{(t)} r_i^2(\boldsymbol{x}). \tag{3.26}$$

2 **Weight update**: update the weights using $\boldsymbol{x}^{(t)}$

$$w_i^{(t+1)} \in \arg\min_{w_i \in [0,1]} \Phi_{\rho_\mu}(w_i) + w_i r_i^2(\boldsymbol{x}^{(t)}), i = 1, \ldots, N. \tag{3.27}$$

3 **Control parameter update**: Increase or decrease μ to add more nonconvexity to ρ_μ.

The GNC algorithm is similar to the IRLS algorithm, except that it starts with a convex, smoothed function ρ_μ and then iteratively updates the control parameter μ to gradually add more non-convexity to ρ_μ to approach the original loss function ρ. Depending on the definition of the smoothed loss ρ_μ, one would recover the original ρ by either increasing or decreasing μ. For instance, the smoother GNC truncated quadratic loss recovers the original truncated quadratic loss when μ is large, hence μ is increased by a constant factor $\gamma > 1$ at each GNC iteration (*e.g.,* $\gamma = 1.4$ in [1218]). Conversely, the smoother Geman-McClure loss recovers the original GM loss when μ is close to 1, hence μ is *divided* by $\gamma > 1$ at each GNC iteration.

Figure 3.9 showcases the SLAM trajectories obtained by applying GNC on the same three datasets of Figure 3.6, with two different robust losses: the Geman-McClure loss and the truncated quadratic loss. We implemented GNC using GT-SAM's GNCOptimizer. By comparing the figure with Figure 3.6 and Figure 3.7 we observe that GNC ensures better convergence (*i.e.,* it is less prone to being stuck in local minima) and recovers fairly accurate trajectories in all the three datasets. While GNC has been shown to be extremely resilient to outliers (*e.g.,* it has shown to tolerate around 80-90% incorrect loop closures in real-world problems [1218, 171]), we remark that the approach does not provide any convergence guarantees. Moreover, its performance has been empirically seen to be problem-dependent, and while it leads to superb performance in pose-graph optimization problems, its performance largely degrades in other perception problems [1011].

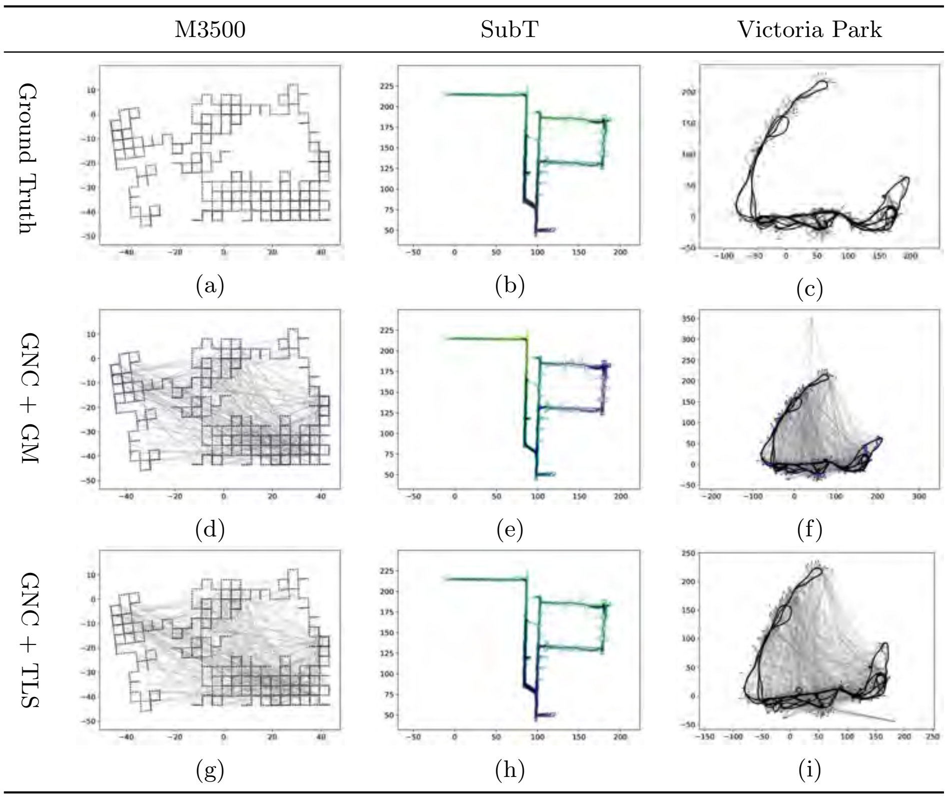


Figure 3.9 Solving SLAM problems with outliers using robust loss functions and graduated non-convexity (GNC): (a)-(c) Ground truth trajectories for the M3500, SubT, and Victoria Park datasets. (d)-(f) Trajectory estimates obtained using the Geman-McClure loss. (g)-(i) Trajectory estimates obtained using the truncated quadratic loss. Measurements are visualized as colored edges. In particular, an edge is colored in gray if it is correctly classified as inlier or outlier by the optimization (*i.e.,* an inlier that falls in the quadratic region of the Huber or truncated quadratic loss); it is colored in red if it is an outlier incorrectly classified as an inlier (a "false positive"); it is colored in blue if it is an inlier incorrectly classified as an outlier (a "false negative").

Below we showcase a problem when GNC fails and also show that combining front-end and back-end outlier rejection can be beneficial. Towards this goal, we are going to consider a slightly more challenging SLAM setup compared to the ones discussed above. Earlier in this chapter, we considered pose-graph optimization problems (*e.g.,* Figure 3.1) where the odometry is reliable but there might be outliers in the loop closures or in the landmark measurements. This setting essentially assumes the presence of an "odometry backbone" that largely simplifies the problem by providing a set of trusted measurements while also allowing building an initial guess for the robot poses. While this assumption is realistic in many problems,[14] certain SLAM applications might lack an odometry backbone. For instance, certain odometry sensors might produce incorrect measurements, *e.g.,* due to wheel slippage in wheel odometry, or incorrect lidar alignment in lidar odometry. Another example arises in multi-robot SLAM, where each robot has an odometry backbone, but the overall SLAM problem (including the trajectory of all robots) does not [735].

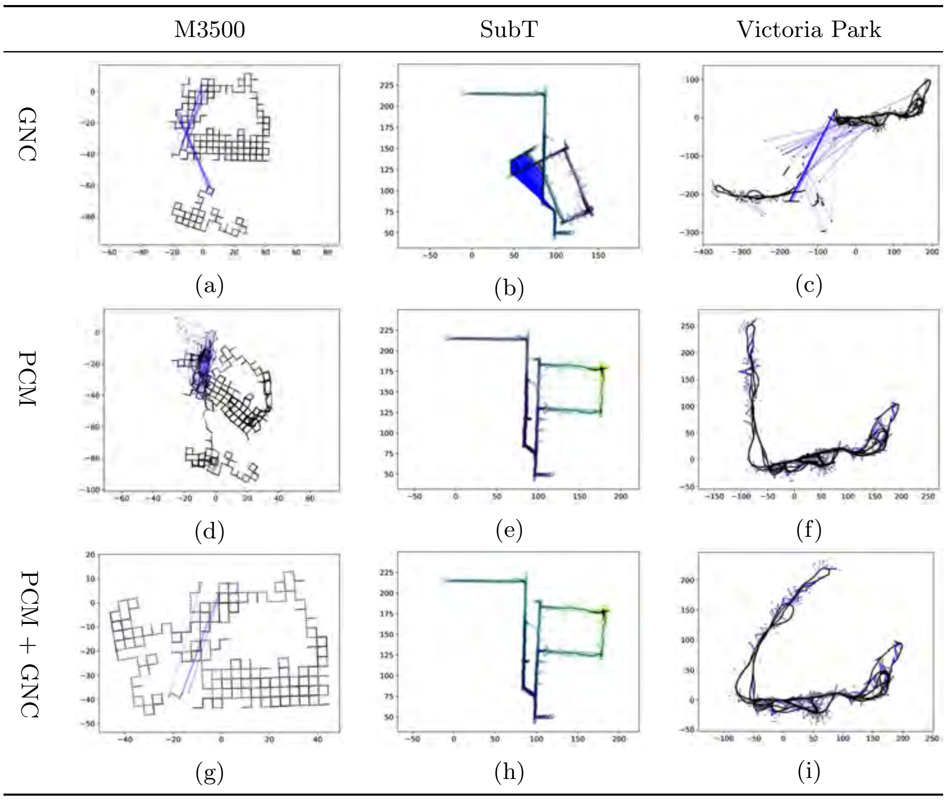


Figure 3.10 SLAM problems with outliers in both the loop closures and landmark measurements, as well as the odometry: (a)-(c) Trajectory estimates obtained using GNC with the truncated quadratic loss. (d)-(f) Trajectory estimates obtained using PCM for front-end outlier rejection followed by least squares optimization. (g)-(i) Trajectory estimates obtained using PCM for front-end outlier rejection followed by GNC with truncated quadratic loss.

Robustly solving SLAM problems where both odometry and loop closure measurements can be outliers is extremely hard. To showcase the shortcomings of the

[14] The fact that odometric measurements are computed between consecutive frames makes data association relatively simpler, in particular when the rate of the sensor (*e.g.,* camera framerate) is much faster than the motion of the robot. Intuitively, consecutive frames will provide snapshots of the scene from similar viewpoints, thus reducing sources of errors for feature matching, including illumination changes, occlusions, lack of viewpoint invariance, or perceptual aliasing.

approaches we discussed, in this more complex setting with potentially incorrect odometry measurements, we first modify the pose-graph problems in Figure 3.1 by corrupting two randomly selected odometry measurements. Then we attempt to solve the problem with GNC. In particular, in GNC we fix all odometry measurements except the two potentially corrupted measurements as inliers. Figure 3.10 shows the trajectories obtained by solving the problem with GNC. GNC fails due to the corrupted initialization and converges to a local minima by categorizing all measurements (including inliers) as outliers (blue edges). The same figure also shows the result of using PCM to filter out gross outliers before using a least-squares back-end. PCM is agnostic to the initial guess, hence is able to converge to better solutions in the case of the SubT and Victoria Park datasets (Figure 3.10 (e) and (f)). Interestingly, we get best results in the SubT and Victoria Park datasets by combining front-end outlier rejection (PCM) with GNC. At the same time, all three methods (GNC, PCM, and PCM+GNC) fail to converge to acceptable solutions in the M3500 dataset, confirming the hardness of this SLAM setup and the limitations of existing SLAM algorithms in terms of outlier rejection.

3.4 Further References and New Trends

Consensus Maximization. While in this chapter we discussed the most basic instantiation of a RANSAC algorithm (according to the initial proposal in [336]), it is worth mentioning that the literature offers many RANSAC variants, including variants that refine estimates through local optimization [224], use better scores (rather than the size of the consensus set) in the RANSAC iterations (*e.g.,* MLESAC [879]), or bias the sampling in the RANSAC iterations (*e.g.,* PROSAC [223]). The recent literature also includes differentiable variants of RANSAC [1172] and variants that attempt to find the inliers when the parameter γ in (3.4) is unknown (*e.g.,* [53]). A recent survey and evaluation of RANSAC variants can be found in [1113].

Beyond RANSAC, the literature also includes approaches for *exact* consensus maximization, typically based on *branch-and-bound* [71, 443, 1285, 658, 1032, 211, 515, 138, 1221, 1220]. Despite its global optimality guarantees, branch-and-bound has exponential runtime in the worst case and does not scale to high-dimensional problems.

Pairwise Consistency Maximization. The PCM approach described in Section 3.2.2 was originally proposed in the context of multi-robot SLAM in [735], where this approach showed particular promise. Graph-theoretic outlier rejection has also been investigated in computer vision. Segundo and Artieda [972] build an association graph and find the maximum clique for 2D image feature matching. Perera and Barnes [876] segment objects under rigid body motion with a clique formulation. Leordeanu and Hebert [646] establish image matches by finding strongly-connected clusters in the correspondence graph with an approximate spectral method. Enqvist *et al.* [317] develop an outlier rejection algorithm for

3D-3D and 2D-3D registration based on approximate vertex cover. Yang and Carlone [1213, 1219] and Parra *et al.* [139] investigate graph-theoretic outlier rejection based on maximum clique for 3D-3D registration. The idea of checking consistency across a subset of measurements also arises in Latif *et al.* [635], which perform loop-closure outlier rejection by clustering measurements together and checking for consistency using a Chi-squared-based test. The PCM paper [735], similarly to the discussion in this chapter, focuses on pairwise consistency. More recently, PCM has been extended to group-k consistency (*i.e.,* the case where the consistency constraint (3.7) involves k measurements instead of only 2 measurements) in [1011, 1012, 340]. These papers essentially generalize the notion of consistency graphs to consistency *hypergraphs*, where each hyper-edge involves k nodes. Related work also considers soft variations of the maximum clique problem, where the binary condition (3.7) is relaxed to produce continuous weights on the edges of the consistency graph [720, 719]. These methods have been used in practical applications, including subterranean exploration [306], lidar point-cloud localization [721], multi-robot metrics-semantic mapping [1099], and global localization in unstructured environments [40].

Alternating Minimization and Graduated Non-Convexity. M-estimation has been a popular approach for robust estimation in robotics [105] and vision [983, 179]. Tavish *et al.* [1079] investigate the performance of different loss functions. Several papers investigate formulations with auxiliary variables as the one in (3.16), without realizing the connection to M-estimation provided by the Black-Rangarajan duality (Theorem 3.4). For instance, Sünderhauf and Protzel [1064, 1063] and Agarwal *et al.* [18] augment the problem with latent binary variables responsible for deactivating outliers. Lee *et al.* [644] use expectation maximization. Olson and Agarwal [833] use a max-mixture distribution to approximate multi-modal measurement noise. Recently, Barron [64] proposes a single parametrized function that generalizes a family of robust loss functions in M-estimation. Chebrolu *et al.* [180] design an expectation-maximization algorithm to simultaneously estimate the unknown quantity $\boldsymbol{x}$ and choose a suitable robust loss function ρ. The graduated non-convexity algorithm was first introduced in [93, 92] for outlier rejection in early vision applications; more recently, the algorithm was used for point cloud registration [1292, 1219], SLAM [1218], and other applications [41]. Recently, Peng *et al.* [867] has proposed an algorithm similar to GNC and IRLS, that is based on the idea of *smooth majorization* in optimization and can be applied to a broad set of robust losses. Moreover, [867] derives global and local convergence guarantees for GNC.

Certifiable Algorithms. The algorithms described so far can be broadly divided into two categories: (i) *fast heuristics* (*e.g.,* RANSAC or local solvers for M-estimation), which are efficient but provide little performance guarantees, and (ii) *global solvers* (*e.g.,* branch-and-bound), which offer optimality guarantees but scale poorly with the problem size. Recent years have seen the advent of a new type of methods, called *certifiable algorithms*, that try to strike a balance between tractability and optimality. Certifiable algorithms relax non-convex robust estimation problems into convex *semidefinite programs* (SDP),[15] whose solutions can be obtained in polynomial time and provide readily checkable *a posteriori* global optimality certificates. Certifiable algorithms for robust estimation have been proposed in the context of rotation estimation [1214], 3D-3D registration [1219], and pose-graph optimization [626, 159]. A fairly general approach to derive certifiable algorithms for problems with outliers is described in [1215, 1217], while connections with parallel work in statistics is discussed in [153]. With few notable exceptions, these algorithms, albeit running in polynomial time, are still computationally expensive and typically much slower than heuristics methods. In some cases, the insights behind these algorithms can be used to certify optimality of a solution obtained with a fast heuristic [1215], hence getting the best of both worlds.

[15] We are going to review the notion of certifiable algorithms in the context of SLAM in Chapter 6.