2023

Robust state estimation methods for robotics applications

Shounak Das
West Virginia University, sd0111@mix.wvu.edu

Robust state estimation methods for robotics applications

Shounak Das

Dissertation submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in
Aerospace Engineering

Jason N. Gross, Ph.D., Chair
Yu Gu, Ph.D.
Guilherme A. S. Pereira, Ph.D.
Natalia A. Schmid, Ph.D.
Nicholas Szczecinski, Ph.D.
Ryan M. Watson, Ph.D

Department of Mechanical and Aerospace Engineering
West Virginia University
Morgantown, West Virginia, USA
2023

Keywords: Localization, Kalman filter, robust estimation, nonlinear least squares, factor graphs

Abstract

Robust state estimation methods for robotics applications

Shounak Das

State estimation is an integral component of any autonomous robotic system. Finding the correct position, velocity, and orientation of an agent in its environment enables it to do other tasks like mapping and interacting with the environment, and collaborating with other agents. State estimation is achieved by using data obtained from multiple sensors and fusing them in a probabilistic framework. These include inertial data from Inertial Measurement Unit (IMU), images from camera, range data from lidars, and positioning data from Global Navigation Satellite Systems (GNSS) receivers. The main challenge faced in sensor-based state estimation is the presence of noisy, erroneous, and even lack of informative data. Some common examples of such situations include wrong feature matching between images or point clouds, false loop-closures due to perceptual aliasing (different places that look similar can confuse the robot), presence of dynamic objects in the environment (odometry algorithms assume a static environment), multipath errors for GNSS (signals for satellites jumping off tall structures like buildings before reaching receivers) and more. This work studies existing and new ways of how standard estimation algorithms like the Kalman filter and factor graphs can be made robust to such adverse conditions without losing performance in ideal outlier-free conditions. The first part of this work demonstrates the importance of robust Kalman filters on wheel-inertial odometry for high-slip terrain. Next, inertial data is integrated into GNSS factor graphs to improve the accuracy and robustness of GNSS factor graphs. Lastly, a combined framework for improving the robustness of non-linear least squares and estimating the inlier noise threshold is proposed and tested with point cloud registration and lidar-inertial odometry algorithms followed by an algorithmic analysis of optimizing generalized robust cost functions with factor graphs for GNSS positioning problem.

Contents

Dedicated to Aurjomon

Acknowledgments

The research in this thesis has been supported in part by

- NASA EPSCoR Research Cooperative Agreement WV-80NSSC17M0053
- Alpha Foundation for the Improvement of Mine Safety and Health, Inc. (ALPHA FOUNDATION)
- USDA Grant 2022-67021-36124
- USDA NRI grant "Collaborative Research: NRI: StickBug - an Effective Co-Robot for Precision Pollination" 1027440

This work would not be possible without the contributions of many people. Firstly, I want to thank my advisor Dr. Jason Gross for his continued support and valuable insights into all the research ideas I worked on. I am also grateful to all committee members Dr. Yu Gu, Dr. Guilherme Pereira, Dr. Natalia Schmidt, Dr. Nicholas Szczecinski, and Dr. Ryan Watson for their valuable suggestions to improve this dissertation. I am indebted to Dr. Cagri Kilic, Dr. Ryan Watson, Eduardo Gutierrez, Uthman Olawoye, and David Akhihiero for their contributions to the collaborative research work done at the WVU Navigation Lab. I would especially like to thank Dr. Kilic and Dr. Watson for helping me in applying my ideas to their existing work on wheel-inertial odometry and GNSS positioning respectively. Finally, I am grateful to my family in India who supported me throughout my PhD journey.

Contributing Papers

A portion of the results and discussion presented within this document were originally contained within other articles. The articles that contribute to specific chapters are presented below.

- Chapter 3: *Das, Shounak, Cagri Kilic, Ryan Watson, and Jason Gross. "A comparison of robust kalman filters for improving wheel-inertial odometry in planetary rovers." In Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021), pp. 2621-2632. 2021.*

- Chapter 4: *Das, Shounak, Ryan Watson, and Jason Gross. "Review of factor graphs for robust GNSS applications." arXiv preprint arXiv:2112.07794 (2021).*

- Chapter 4: *Kilic, Cagri, Shounak Das, Eduardo Gutierrez, Ryan Watson, and Jason Gross. "Zupt aided gnss factor graph with inertial navigation integration for wheeled robots." In Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021), pp. 3285-3293. 2021.*

- Chapter 5: *S. Das and J. N. Gross, "Analysis of Scale-Variant Robust Kernel Optimization for Non-linear Least Squares Problems," in IEEE Transactions on Aerospace and Electronic Systems, doi: 10.1109/TAES.2023.3290142.*

Other contributions not included in the thesis are mentioned here.

- *Akhihiero, David, Uthman Olawoe, Shounak Das, Jason Gross."Cooperative Localization for GNSS-Denied Subterranean Navigation: A UAV-UGV Team Approach.", 2023 (submitted)*

- *Kilic, Cagri, Bernardo Martinez, Christopher A. Tatsch, Jared Beard, Jared Strader, Shounak Das, Derek Ross, Yu Gu, Guilherme AS Pereira, and Jason*

N. Gross. "NASA space robotics challenge 2 qualification round: an approach to autonomous lunar rover operations." IEEE Aerospace and Electronic Systems Magazine 36, no. 12 (2021): 24-41.

1

Introduction

Estimation theory deals with the process of calculating values of desired variables in a model using noisy observations. The observations come from a sensor, so the model can be thought of as a sensor measurement model. Examples of estimation problems include locating aircrafts and submarines with radar and sonar measurements respectively, estimating the underground depth of oil deposits from sound reflections,

denoising electronic signals, tracking aircraft using radars, estimating the orbit of an asteroid, aircraft performance parameter estimation from real-time data obtained from onboard sensors and many more [12, 26, 63, 69, 100]. Any engineering control system needs to estimate the current system variables of the model to calculate the required control input [100]. For robotics applications, a very important problem is estimating the position, attitude, and velocity of aerial or ground robots using onboard sensors like cameras, lidars, and body acceleration measurements [12, 43, 64]. If the environment is unknown then the robot might need to map the environment to find specific targets [20]. Another important application of estimation theory is sensor calibration which is critical for autonomous systems [1, 122]. A more ubiquitous application of estimation theory is positioning using signals from satellites [37] which we use on a day-to-day basis without giving it much thought. Thus importance of estimation theory in the modern world cannot be overstated. With the advent of powerful computers, estimation for robotics has seen rapid developments resulting in impressive achievements like self-driving cars [131], autonomous drones [59], and highly automated robotic arms for industrial and medical applications [75].

1.1 Motivation

Correct estimation of desired variables of a system is critical for its safe operation. However, sensors are inherently noisy and affected by the environment it senses. The most commonly used sensors in robotics have their own limitations. For example, pose estimation with two camera images is highly dependent on the number of fea-

ture matches and the correctness of matches. Similarly calculating pose with two point clouds requires accurate feature matching. Even though IMU is not affected by environmental conditions, it suffers from error growth over time. GNSS signals give the highest accuracy of positioning but can get degraded due to atmospheric and multipath effects. Visual loop closures can help reduce pose errors by identifying revisited places and correcting the current pose using the pose from the last visit. But loop closures can suffer from visual aliasing (two places looking similar even though they are not nearby). It is crucial that the estimation algorithms be able to detect such conditions and adapt accordingly to provide reliable estimation performance. The algorithms that have been designed to be adaptive and deal with the outliers mentioned above are called robust estimation algorithms.

1.2 Research Objective

The objective of this dissertation is to develop solutions to challenging robotics localization problems by exploring the use and development of novel robust estimation methods. Problems related to measurement outlier rejection and sensor degradation are considered. Further, multiple applications related to robotics localization are considered including Global Navigation Satellite Systems, wheel-inertial odometry, and point cloud registration. In order to offer insight, several different techniques and estimation frameworks are compared and analyzed and a new robust estimation algorithm is developed and analyzed using both real-world and synthetic data.

1.3 Contributions

This dissertation looks at robust estimation from three different directions. The first direction focuses on applying existing robust estimation algorithms to new problems. One such problem discussed in the next chapter is wheel-inertial odometry on high-slip terrain. The second direction is improving robustness by using a multi-sensor approach. For this direction, a GNSS positioning algorithm is fused with inertial data. The third direction aims to improve the adaptability of non-linear least squares and learn inlier noise threshold for sensors in a combined approach.

The main contributions of this dissertation are discussed below:

- Five methods, chosen from robust Kalman filtering literature, have been applied to the problem of wheel-inertial odometry on high-slip terrain to offer insight into their benefits and drawbacks to the research community. These methods include the Huber Kalman filter, the covariance scaling Kalman filter, and variational filters. An error-state Extended Kalman filter is implemented with its update step augmented with these robust methods. Variational Kalman filters perform the best in reducing the effect of erroneous wheel encoder measurements. Parameter analysis shows the residual distribution to be close to Gaussian. This work is presented in Chapter 3.

- Two different ways of fusing inertial data with GNSS factor graphs are presented in Chapter 4. In one version, motion constraints are added to the GNSS factor graph from a parallel running wheel-inertial Kalman filter. The other version does not use inertial odometry directly but uses the information that the rover is

static to constrain static nodes. Tests with real-world data demonstrate the superior positioning performance of inertial-aided GNSS factor graphs compared to GNSS factor graphs without odometry information.

- A novel way of increasing the adaptivity of non-linear least squares is presented in Chapter 5. The new algorithm also estimates the inlier noise threshold along with increasing adaptivity in a combined approach. The algorithm is tested with synthetic and real-world point clouds and shown to have a more robust registration performance than existing M-estimator-based methods in the literature. Lastly, an analysis is presented on adaptive non-linear least squares using square-root factors for GNSS positioning. It is demonstrated that learning the shape parameter in a GNSS factor graph using Black-Rangarajan duality can result in divergence of state estimates for some M-estimators.

1.4 Organization

The rest of the dissertation is divided into five chapters. Chapter 2 discusses the fundamental estimation tools used in this dissertation. Chapter 3 contains a brief overview of different robust Kalman filtering algorithms followed by their detailed implementation and performance analysis for robust wheel-inertial odometry in high-slip terrain. Chapter 4 presents two different frameworks for fusing inertial information from an EKF into a GNSS factor graph to increase its robustness to multipath errors. Chapter 5 presents a novel way of increasing the adaptivity of robust cost functions and learning inlier noise thresholds in a combined framework. Adaptive cost function

optimization is implemented for both point cloud registration and GNSS positioning problems. Chapter 6 concludes the thesis with more discussion on the results and future research directions. Some robustness metrics of adaptive M-estimators are discussed in Appendix A.

2

Background

Even though estimation algorithms have become more complex over the years, theoretical foundations of all such algorithms like [129], [78], [77] and [107] have remained the same. As discussed in much detail in [12] and [32], probability theory (especially Gaussian), Bayes' theory, linear algebra, and optimization (especially linear and non-linear least squares) form the foundation for estimation in robotic systems. This

chapter gives a brief overview of some of these key concepts which will be crucial for understanding the rest of this work.

Most estimation problems in robotics can be formulated as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{k} \|\mathbf{r}_k(\mathbf{x})\|^2 , \tag{2.1}$$

where

$$\mathbf{r}_k(\mathbf{x}) = \mathbf{z}_k - h_k(\mathbf{x}). \tag{2.2}$$

$\mathbf{r}_k$ is called the residual error, which indicates how well the kth predicted measurement fits the kth actual measurement. $\mathbf{x}$ is a vector of variables that can include robot position, velocity, orientation, map points, camera intrinsic, and other model parameters depending on the sensor. $h()$ is the function that maps the parameters $\mathbf{x}$ to the measurements $\mathbf{z}$ and is called the measurement function. The objective here is to find $\mathbf{x}$ that minimizes the residual cost above; in other words best explains the observed data. Unless mentioned separately, usually vectors will be referred to with bold lower case letters, scalars in lower case non-bold letters, and matrices with upper case non-bold letters.

2.1 Bayesian Probability

Bayesian estimation theory can be understood from two different perspectives. One is the probabilistic point of view and the other is optimization. These are two sides of the same coin and is important to understand from both perspectives. Probability theory

is useful in modeling the fact that all sensor measurements are inherently noisy due to their own manufacturing limitations and also the environment they sense. Thus instead of assuming the unknown states and measurements as having fixed values, they are assumed to be random variables that follow a specific distribution (usually Gaussian).

Let X and Z be the random variables representing the state and measurement respectively. The *Bayes' theorem* states

$$P(X|Z)P(Z) = P(Z|X)P(X) = P(X, Z), \tag{2.3}$$

where $P(X)$ represents the prior belief of the unknown state, $P(Z|X)$ is the likelihood of a measurement given a state, $P(Z)$ is the probability of the measurement and $P(X|Z)$ is called the posterior. $P(X, Z)$ is called the joint probability of X and Z. The main objective of Bayesian estimation is finding the posterior. There are two frameworks for estimation from a probabilistic point of view. One is called the *Maximum Likelihood Principle* (MLE), where given some measured Z, the state can be solved as

$$\hat{X} = \arg\max_{\mathbf{x}} P(Z|X). \tag{2.4}$$

Here the prior is not considered in the estimation. This is usually considered a non-Bayesian approach since the idea of prior and posterior is not used. The other approach is considered the proper Bayesian approach called the *Maximum A-Posteriori*

(MAP) problem, where the state is solved as

$$\hat{X} = \arg\max_{\mathbf{x}} P(Z|X)P(X). \tag{2.5}$$

This problem takes into account the prior belief of your unknown state. The formulation comes from

$$P(X|Z) \propto P(Z|X)P(X). \tag{2.6}$$

$P(Z)$ can be ignored here since it is not a function of X. Assuming the prior and likelihood are Gaussian distributions (note: the likelihood is not a true probability distribution since Z is known and X is unknown; statisticians refer to it as a function of X but the formulation remains the same as Gaussian), the maximization problem can be converted to a minimization problem by using a negative logarithm operation. MAP solves for the mode of the posterior, unlike Bayesian inference which tries to find the full posterior distribution $P(X|Z)$. For linear Gaussian cases, MAP and Bayesian inference compute the same solution since the mean and the mode of a Gaussian distribution are the same. For non-linear and non-Gaussian cases, these solutions do not match. Due to the intractability of full posterior distribution inference in many formulations, MAP or sampling-based methods are used.

2.2 Least Squares

The optimization problem derived from MLE and MAP for the Gaussian case is called the *least squares* problem. If the $h()$ function is linear i.e represented by a matrix

$$\mathbf{z} = J\mathbf{x} + \epsilon,$$

where ϵ is Gaussian white noise, the problem

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} (\mathbf{z} - J\mathbf{x})^T (\mathbf{z} - J\mathbf{x})$$

has a closed-form solution given by

$$\hat{\mathbf{x}} = (J^T J)^{-1} J^T \mathbf{z}. \tag{2.7}$$

Unfortunately, linear least square formulation is insufficient for most estimation problems in robotics due to non-linearity of $h()$ function. The most common $h()$ functions like the inverse-projection function for visual odometry, range functions for ranging measurements, and frame transformation functions for point clouds are all non-linear [42]. In such cases, non-linear least square solvers need to be used. The solution then changes to an iterative one which requires linearizing the non-linear measurement function $h()$ to form a matrix

$$H = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}$$

called the Jacobian. The iterative solution is similar to the least squares solution and can be obtained by solving for $\Delta\mathbf{x}$ from the equation

$$\left(H^T H + \lambda \mathbf{I}\right) \Delta\mathbf{x} = H^T \mathbf{r}. \tag{2.8}$$

$\mathbf{r} = \mathbf{z} - h(\mathbf{x}^0)$ where $\mathbf{x}^0$ is the initialization point or linearization point which is updated every iteration i using

$$\mathbf{x}^0_{i+1} = \mathbf{x}^0_i + \alpha \Delta\mathbf{x}_i. \tag{2.9}$$

$\Delta\mathbf{x}$ is the descent direction for minimizing the sum of the squared residuals. The updated solution can be obtained by moving along the descent direction. The magnitude of this movement is controlled by the parameter α. λ is a damping term set to a large value when the current iteration is far from the solution but lowered when it gets close giving it an almost quadratic convergence rate [72].

A batch version of non-linear least squares can also be solved by stacking residuals and Jacobians to create

$$\mathbf{r} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_n \end{bmatrix}, \quad H = \begin{bmatrix} H_1 \\ \vdots \\ H_n \end{bmatrix}, \quad W = \begin{bmatrix} w_1\mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & w_n\mathbf{I} \end{bmatrix}.$$

The weight matrix W sets the weight for the residual of each measurement epoch. The solved state vector is a long vector where all the states along the trajectory are stacked together $\mathbf{x} = [\mathbf{x}_1 \dots \mathbf{x}_n]$. Also common in batch non-linear least squares

is segmenting the state variable of an epoch into blocks. For example, in visual Simultaneous Localization and Mapping (SLAM) problems, the pose at time t ($\mathbf{T}_t$) will have its Jacobian block and a landmark tracked at the time ($\mathbf{L}_t$) will have its separate Jacobian. For every iteration, the Jacobians and residuals are calculated using the current estimate of $\mathbf{x}$ and stacked to form $\mathbf{r}$ and H. W helps de-weight outliers or noisy measurements.

The weighted non-linear least squares version of the solution is given by

$$\left(H^T W H + \lambda \mathbf{I}\right) \Delta \mathbf{x} = H^T W \mathbf{r}. \tag{2.10}$$

All optimization packages like *Ceres* [7], *g2o* [45], *GTSAM* [30] use some variation of this descent strategy to get the best solution $\hat{\mathbf{x}}$.

2.3 Robustness Theory

The idea of *robustness* of an estimator follows from [50], where it is defined as "*insensitivity to small deviations from the assumptions*". This insensitivity of the estimator is measured from a probabilistic or distributional point of view. That is, robust estimators can estimate parameters even when the measurement noise does not follow the assumed distribution, which is usually Gaussian. This non-Gaussian distribution is called the *contaminated* Gaussian in [74]. That is $(1 - \epsilon)$ proportion of the measurement is generated from a Gaussian and other ϵ is generated from an unknown distribution. For example, the median and mean absolute deviation are more robust estimators of the mean and spread of data respectively than the mean and stan-

dard deviation when data follows a long-tailed distribution instead of the assumed Gaussian distribution.

2.3.1 M-estimators

One popular choice of robust estimator is the *M-estimator* (Maximum likelihood type estimator) which is formulated as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_k \rho\left(r_k(\mathbf{x})\right). \tag{2.11}$$

This is equivalent to solving the equation

$$\sum_k \psi\left(r_k(\mathbf{x})\right) \frac{\partial r_k(\mathbf{x})}{\partial \mathbf{x}} = 0, \tag{2.12}$$

where $\psi(x) = \frac{\partial \rho(x)}{\partial x}$. ρ replaces the standard squared loss function with a function with a lower growth rate which reduces the effect of measurements with large residuals. One measure of robustness for M-estimators is the change in the estimate with infinitesimal change Δ in one of the measurements, called the Influence Function Curve (IF curve) which is proportional to ψ. It represents the effect of a measurement on the estimation process as a function of its residual. Diagrams of these ψ functions for different M-estimators are shown in [18, 29]. Due to the nature of the loss functions, the influence functions show constant, sub-linear, or re-descending (increasing first and then decreasing) trends.

The distribution of M-estimators has been proved to be asymptotically normal. That is, for $n \to \infty$, the estimates obtained from M-estimators are normally dis-

tributed around the true value [74]. For the scalar location problem, the estimate follows the distribution

$$\hat{x} \sim \mathcal{N}(x*, \frac{v}{n}), \tag{2.13}$$

where

$$v = \frac{\mathbf{E}_F(\psi(x - x*))^2}{(\mathbf{E}_F\psi'(x - x*))^2}. \tag{2.14}$$

v is the asymptotic variance of that specific M-estimator. $x*$ is the solution of $\mathbf{E}_F\psi(r(x)) = 0$ where F is the true error distribution. One way to compare estimators is called *efficiency* ($\mathcal{E}$) which gives a measure of estimator performance when compared to a reference estimator. It is defined as

$$\mathcal{E} = \frac{v_0}{v}, \tag{2.15}$$

where v_0 is the asymptotic variance of the reference estimator. For example, the M-estimators will have higher asymptotic variance than the least squares estimator when the true distribution of errors is Gaussian resulting in low efficiency but will have less asymptotic variance in case of contaminated distribution producing higher efficiency. As shown in table (2.2) of [74], for the case of location estimate with scalar x, the asymptotic variance of the Huber M-estimator is slightly more than that of least squares for the Gaussian case. Still, it has a much lower variance than least squares for the contaminated Gaussian distribution.

Another metric that characterizes the robustness of an estimator is called the Breakdown Point (BP). It refers to the largest fraction of outliers with which the estimator still remains bounded. Theoretically, estimators cannot have BP more than

0.5 due to it not being able to extract the true distribution from the contaminating distribution [94]. Monotonic and re-descending ψ functions have a BP of 0.5 for location estimation problems [74]. Huber and Laplace estimators have a higher BP than Cauchy and Welsch estimators [18]. The least squares estimator has a BP of 0.

2.3.2 Solving M-estimators

Equation (2.11) is a continuous optimization problem by comparing it to a weighted least square problem. For scalar residuals $\mathbf{r}_k$, the weighted least squares problem is

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_k w_k r_k(\mathbf{x})^2, \tag{2.16}$$

where w_k is the weight of the kth residual. Two types of methods are generally used: line search methods, such as Gauss-Newton, and trust region methods, such as Levenberg-Marquardt, both of which are iterative descent methods [44, 71]. Partially differentiating the least squares and M-estimation expressions in their scalar forms with respect to the unknown variable $\mathbf{x}$ we get,

$$\frac{1}{2}\frac{\partial\left(w_k r_k^2(\mathbf{x})\right)}{\partial \mathbf{x}} = w_k r_k(\mathbf{x})\frac{\partial r_k(\mathbf{x})}{\partial \mathbf{x}} \tag{2.17}$$

and

$$\frac{\partial\left(\rho\left(r_k(\mathbf{x})\right)\right)}{\partial \mathbf{x}} = \rho'\left(r_k(\mathbf{x})\right)\frac{\partial r_k(\mathbf{x})}{\partial \mathbf{x}}. \tag{2.18}$$

Comparing these two expressions, it is apparent that M-estimation can be solved exactly like a weighted least squares problem. The weights in this case are given by

$$w_k = \frac{\rho'(r_k(\mathbf{x}))}{r_k(\mathbf{x})}. \tag{2.19}$$

Any scalar multiplied to this expression can be ignored since it applies to all residuals. This method of solving M-estimation problems is called Iterative Re-weighted Least Squares (IRLS) [18]. The IRLS algorithm is described in chapter 4. The conditions for convergence of IRLS solutions for M-estimators have been discussed in [2]. They are:

- Functions ρ and r are continuous.
- $\rho(\sqrt{x})$ is concave and differentiable for $x \geq 0$.
- $\arg\min_{\mathbf{x}} \sum_k w_k r_k(\mathbf{x})^2$ is a continuous function of w.
- The sublevel set $\{\mathbf{x} | \sum_k \rho\left(r_k(\mathbf{x})\right) \leq \sum_k \rho\left(r_k(\mathbf{x^0})\right)\}$ is bounded.

These conditions also assume that the weighted least squares solution can be solved efficiently. The concavity condition makes sure that minimizing the weighted least squares cost results in the minimization of the robust cost expression. Given these conditions, if the expression $\sum_k \rho\left(r_k(\mathbf{x})\right)$ is convex then IRLS should converge to the global minimum.

M-estimators fall within the de-weighting group of methods, which do not directly remove measurements. The intuition behind this is that, instead of assuming a Gaussian distribution for the measurement noise, these M-estimators have heavier

tails, which solve for the parameters that best fit the overall data. An interesting connection between M-estimators and elliptical distributions was shown in [5], which was used for parameter estimation. The cost functions obtained from the negative log-likelihood of these distributions are modified versions of the squared loss function, which are optimized to get to the correct solution. Some of these functions are non-convex and suffer from the local minima problem (e.g., re-descending M-estimators). To tackle this, Yang et al. [125] use the concepts of graduated non-convexity, along with the Black-Rangarajan duality [17], to devise an iterative algorithm for robust perception. Another common area of application for robust estimation is loop closures. To mitigate the effect of false loop closures, several researchers have considered approaches for adding robustness to the back-end of SLAM applications. Sunderhauf et al. [103] added binary scalars, or switch constraints, to measurements to turn them on/off in the optimization. Dynamic Covariance Scaling [6] was then developed to give the same theoretical benefit in a more efficient manner. These methods do the same job of de-weighting outliers as the M-estimators. The importance of M-estimators in point cloud registration and visual navigation has been discussed in [11, 70]. These methods were used for robust GNSS positioning in a factor graph framework in [115, 117, 118]. Recently Ramezani et al. [90] modeled all loop closures using robust cost functions with a single adaptive parameter and improved back-end SLAM optimization. Generalized robust cost functions are analyzed further in chapter 5.

2.3.3 Maximum Consensus

The other group of robust estimation methods focuses on finding the maximum number of measurements that satisfy a specific inlier condition, which is also called the Maximum Consensus (MC) problem [25]. One of the most well-known methods to solve these kinds of problems is called Random Sample Consensus (RANSAC) [40]. RANSAC is an iterative algorithm that samples a random minimal set of data points every iteration and solves for the parameter values until it reaches a specific inlier count threshold. It can be shown easily that for a dataset with a specific outlier fraction, the probability of sampling a minimal set of inliers for solving the parameters increases exponentially with the number of iterations. However, this method can also get very expensive if the dataset has a large proportion of outliers.

Algorithm 1: Random Sample Consensus (RANSAC)

Input: numInliers, data

Output: inliers, bestEstimate

$n \leftarrow 0$;

while $n <$ numInliers **do**

  Sample a minimal set from data;

  Solve to get bestEstimate;

  Find the number of inliers n using some residual threshold;

end

Antonante et al. [10] provide an in-depth discussion of various robust estimation methods across different disciplines along with their computational limits. They

develop minimally tuned algorithms that can tolerate a large number of outliers. Shi et al. [99] use invariance relations between measurements to create consistency graphs. The nodes in the graph represent measurements and two nodes will be connected by an edge if they are consistent under some invariance relation. Then the MC problem can be solved by searching for the maximal clique (largest set of fully connected nodes) in the consistency graph and estimating the state using those. The advantage of these methods is that they are independent of the estimate of $\mathbf{x}$, unlike residual-based methods like M-estimators. However, invariance relations do not always exist for all measurement models. Loop closures for multi-robot systems have been similarly modeled as a maximal clique problem analyzed in [73].

3

Robust Kalman Filtering

The *Kalman filter* [56] is the most well-known Bayesian filtering algorithm. It is a recursive algorithm that estimates the current state $\mathbf{x}_k$ given the current measurement $\mathbf{z}_k$ and the state estimate at the previous time step $\hat{\mathbf{x}}_{k-1}$ following the Bayesian framework. The most common version is the Extended Kalman filter (EKF) which uses the linear Gaussian assumption to give exact posterior estimates of the current

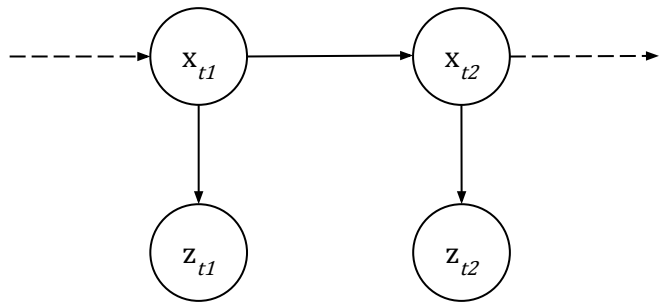


Figure 3.0.1: *The Hidden Markov Model (HMM) is a very common representation of many estimation problems. The direction signifies conditional dependence i.e* $\mathbf{x}_{t2}$ *is only dependent on* $\mathbf{x}_{t1}$ *(this is the Markov assumption) and the measurement* $\mathbf{z}_{t1}$ *is dependent on* $\mathbf{x}_{t1}$ *(hence the term "hidden" since the state* $\mathbf{x}$ *is not directly observed). The Kalman filter does inference on this model.*

state recursively using the current measurements and estimated state for the last state in time. The filter equations are presented here.

Algorithm 2: Extended Kalman filter (EKF)

Input: $\hat{\mathbf{x}}_{k-1}$, $\hat{P}_{k-1}$, F_k, H_k Q_k, R_k

Output: $\hat{\mathbf{x}}_k$, $\hat{P}_k$

% Prediction / Propagation

$\hat{\mathbf{x}}_{k/k-1} = F_k \hat{\mathbf{x}}_{k-1}$;

$\hat{P}_{k/k-1} = F_k \hat{P}_{k-1} F_k^T + Q_k$;

% Correction / Update

$\hat{\mathbf{v}}_k = \mathbf{z}_k - H_k \hat{\mathbf{x}}_{k/k-1}$;

$K_k = \hat{P}_{k/k-1} H_k^T \left[H_k \hat{P}_{k/k-1} H_k^T + R_k \right]^{-1}$;

$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k/k-1} + K_k \hat{\mathbf{v}}_k$;

$\hat{P}_k = [I - K_k H_k] \hat{P}_{k/k-1}$;

$\hat{\mathbf{x}}_k$ is the current estimate of the state using the measurement $\mathbf{z}_k$. F_k and H_k are the

state transition and measurement matrices respectively. Matrix $\hat{P}_k$ represents the state uncertainty, Q_k is the process noise covariance matrix and R_k is the measurement noise covariance matrix. The filter gain at time k is denoted by K_k which controls the weight imposed by the prior versus the measurement in the estimation. This recursive method is optimal in the Bayesian perspective since it maximizes (MAP) or infers (Bayesian inference) the posterior $P(\mathbf{x}_k|\mathbf{z}_0..\mathbf{z}_k)$ for the linear Gaussian case i.e assuming $\mathbf{x}_k|\mathbf{x}_{k-1} \sim \mathcal{N}(F\mathbf{x}_{k-1}, Q_k)$ and $\mathbf{z}_k|\mathbf{x}_k \sim \mathcal{N}(H\mathbf{x}_k, R_k)$. A detailed derivation of Kalman filtering for both MAP and Bayesian Inference contexts is provided in [12]. Other interesting directions of deriving Kalman filters come from gain optimization [12] and weighted, regularized right pseudo-inverse of $\mathbf{z} = H\mathbf{x}$. Variations of the standard EKF exist to handle non-linear motion and measurement functions, like the Unscented Kalman filter (UKF) [112] and particle filters [91]. Recent works on Kalman filters include adaptive and robust variations where the measurement model is assumed to be heavy-tailed and posterior inference is achieved by approximation methods like the variational inference [3, 4, 47, 96]. Adaptive and robust Kalman filters are discussed in detail in the next section. The Invariant Extended Kalman filter (IEKF) [13] extends the EKF framework to Lie groups. Lie theory [102] helps in dealing with state variables like pose and attitude which do not lie on a vector space, unlike position and velocity. This is due to constraints on the variables (like rotation matrices are orthogonal). They lie on manifolds called $SE3$ and $SO3$ resulting in error calculations not following standard addition and subtraction operations but special mapping functions. Thus optimization involving these variables is not straightforward. A detailed description of how lie theory is applied to robotics esti-

mation problems has been discussed in [102]. Contrary to EKF, the gain matrix K in IEKF is not a function of the H and F resulting in better convergence properties. It also solves the inconsistent covariance estimation problems faced in EKF-based SLAM formulations [13].

3.1 Robust and Adaptive Filters

There is a rich body of literature that has considered modifying the standard Kalman filter to accommodate outliers. A covariance scaling method is introduced in [22] using the ratio between the Mahalanobis distance between the observed data and the predicted data and the critical value of χ^2 distribution to reduce the effect of observation outliers or heavy-tailed measurement noise distribution. An adaptive factor is defined in [127] that helps the filter to balance the contribution of discrepancies due to dynamic modeling errors and new measurements as well as measurement outliers. Fang et al. [38] use an adaptive innovation-saturation mechanism to estimate the changing measurement error bound. An adaptive version of the Kalman filter is derived in [8] using a fading factor and the current innovation and residual values to recursively calculate process and measurement noise matrices. The Huber cost function has been incorporated into the update step of the Kalman filter using M-estimation principles in [36, 57]. A similar method has been used with tightly and loosely coupled GNSS/INS EKF with simulated faults in [27], where the robust version has comparable performance to that of fault detection and exclusion techniques. Wang et al. [114] derive a robust filter using the Double-Gaussian Mixture Corren-

tropy loss between the data and the predicted measurements. Variational inference has also been used to approximate state posteriors with heavy-tailed distribution for the measurement models in [3, 4, 47, 86, 96]. Faulty measurements are excluded using indicator variables with beta-Bernoulli prior and approximated with the state using mean-field variational inference in [113] in a simulated target tracking scenario. Bayesian weights are assigned to each measurement and estimated along with the filter parameters using the Expectation-Maximization (EM) framework in [108].

In this chapter, five robust Kalman filtering methods have been chosen from the literature and implemented on wheel-inertial odometry in high-slip terrain to offer insights into their advantages and disadvantages. The following subsections give a brief description of these robust Kalman filters.

3.1.1 Huber Kalman Filter (HKF)

Since EKF essentially is derived from a least square perspective, it is natural to extend it to a robust cost version instead of the squared error version. An iterative version of M-estimation has been discussed in [36, 57] which uses the Huber cost function [49] at every time step instead of the squared error function. This causes the error between the predicted measurement and the actual measurement to grow like the squared function when below a parameter Δ and linearly when above it.

The following derivation follows [58]. A single filter prediction and update cycle can be written as

$$\begin{bmatrix} \hat{\mathbf{x}}_{k/k-1} \\ \mathbf{z}_k \end{bmatrix} = \begin{bmatrix} I \\ H_k \end{bmatrix} \mathbf{x}_k + E_k, \tag{3.1}$$

where

$$E_k = \begin{bmatrix} \delta_k \\ \mathbf{v}_k \end{bmatrix}$$

and

$$\mathbf{E}[E_k E_k^T] = \begin{bmatrix} \hat{P}_{k/k-1} & 0 \\ 0 & R_k \end{bmatrix} = S_k S_k^T.$$

Now the new linear equation can be written as

$$\mathbf{Y}_k = B_k \mathbf{x}_k + \xi_k, \tag{3.2}$$

where $B_k = S_k^{-1} \begin{bmatrix} I \\ H_k \end{bmatrix}$, $\xi_k = S_k^{-1} E_k$, $\mathbf{Y}_k = S_k^{-1} \begin{bmatrix} \hat{\mathbf{x}}_{k/k-1} \\ \mathbf{z}_k \end{bmatrix}$, $\mathbf{E}[\xi_k \xi_k^T] = I$. The least squares solution of the modified linear equation is

$$\hat{\mathbf{x}}_k = (B_k^T B_k)^{-1} B_k^T \mathbf{Y}_k \tag{3.3}$$

$$\hat{P}_k = (B_k^T B_k)^{-1}. \tag{3.4}$$

S_k can be calculated by Cholesky decomposition of the covariance matrix of E_k. However, least squares estimation is highly affected by outliers due to the unbounded nature of the squared error function. So, a different cost function can be used. The new problem can be stated as

$$\hat{\mathbf{x}}_k = \arg\min_{\mathbf{x}} J(\mathbf{x}) = \arg\min_{\mathbf{x}} \sum_i \rho\left(y_{ik} - b_{ik}\mathbf{x}\right), \tag{3.5}$$

where the sum is over the dimension of $\mathbf{Y}_k$, y_{ik} is the ith element of $\mathbf{Y}_k$ and b_{ik} is the

i^{th} row of B_k. If ρ is quadratic, then it reduces to the standard Kalman filter solution. But this function can be set such that the effect of large residuals is reduced. The function ρ can be squared for smaller residuals but should increase slowly for larger values. Another important requirement is that the influence function $\psi = \rho'$ be continuous and bounded. It should be bounded so that no single measurement can affect the total cost largely and continuity helps in reducing rounding errors. One such function is the Huber cost function [49] given by

$$\rho(z) = \begin{cases} z^2/2 & |z| \leq \Delta \\ \Delta|z| - \Delta^2/2 & |z| > \Delta \end{cases}. \tag{3.6}$$

The derivative ρ' or ψ is given by

$$\psi(z) = \rho'(z) = \begin{cases} z & |z| \leq \Delta \\ \Delta\,\mathrm{sgn}(z) & |z| > \Delta \end{cases}. \tag{3.7}$$

Other robust functions are discussed in [18]. To minimize the cost function in (3.5), we have to differentiate $J(\mathbf{x})$ with respect to $\mathbf{x}$ and set it to zero. This leads to the equation

$$\sum_i b_{ik}^T \psi\left(y_{ik} - b_{ik}\mathbf{x}_k\right) = 0. \tag{3.8}$$

This equation can be re-written in the form

$$B_k^T \Psi(B_k\mathbf{x}_k - \mathbf{Y}_k) = 0, \tag{3.9}$$

where

$$\Psi = \mathsf{diag}\left(\frac{\psi\left(y_{ik} - b_{ik}\mathbf{x}_k\right)}{y_{ik} - b_{ik}\mathbf{x}_k}\right).$$

Ψ is a diagonal matrix, where each diagonal element is the weight of each residual component. Note, that the weight formula is the same as in equation (2.19). This equation cannot be solved explicitly since Ψ depends on the unknown state $\mathbf{x}_k$. However $\mathbf{x}_k$ can be solved in an iterative manner using IRLS. The solution for iteration $j+1$ uses the solution from iteration j and is given by

$$\hat{\mathbf{x}}_k^{j+1} = (B_k^T \Psi^{(j)} B_k)^{-1} B_k^T \Psi^{(j)} \mathbf{Y}_k. \tag{3.10}$$

This solution converges if ψ is non-decreasing. The IRLS algorithm can be started with the least-square solution $\hat{\mathbf{x}}_k^0 = (B_k^T B_k)^{-1} B_k^T \mathbf{Y}_k$. After the state has converged, the covariance can be estimated by the expression

$$\hat{P}_k = (B_k^T \Psi B_k)^{-1}. \tag{3.11}$$

Choosing the tuning parameter Δ is important, and depends on the perturbing parameter ϵ, which represents the proportion of contamination in the assumed residual distribution [58]. If the perturbing parameter is known, then the optimal choice Δ^* is given in [50]. One way to calculate this parameter is by maximizing the efficiency with respect to a reference estimator like least squares. When the perturbing parameter is unknown, Δ is usually chosen between 1.0 and 2.0.

3.1.2 Covariance Scaling Kalman Filter (CSKF)

The Mahalanobis distance ratio, defined as

$$\gamma_k = \frac{M_k^2}{\chi_{m\alpha}}, \tag{3.12}$$

can be used for assessing if the measurement is inlier or outlier. M_k is the Mahalanobis distance of the actual measurement from the predicted measurement which has the expression

$$M_k^2 = \hat{\mathbf{v}}_k^T [H_k \hat{P}_{k/k-1} H_k^T + R_k]^{-1} \hat{\mathbf{v}}_k. \tag{3.13}$$

$\chi_{m\alpha}$ is the critical value of the χ^2 distribution corresponding to significance level α with the degree of freedom m equal to the measurement dimension. Following [22], the measurement noise covariance matrix (R_k) and the measurement prediction covariance ($H_k \hat{P}_{k/k-1} H_k^T$) are scaled for every outlier detected using the χ^2 criterion. This is also used as the *Independent Innovation Test* in [109]. If γ_k is less than 1 then measurement $\mathbf{z}_k$ is an inlier and R_k remains the same. But if the scaling factor is greater than 1, denoting an outlier, the inflated R_k is given by

$$\hat{R}_k = (\gamma_k - 1) H_k \hat{P}_{k/k-1} H_k^T + \gamma_k R_k. \tag{3.14}$$

The scaling of R_k results in the de-weighting of suspected outlier measurements. The rest of the steps remain the same as the standard filter equations, i.e. replace $\hat{R}_k$ in place of R_k in algorithm (2).

3.1.3 Variational Filters

Variational Bayes filters [97] use a prior for auxiliary variable in the measurement model such that the marginalized measurement distribution has heavier tails. Due to the non-Gaussian nature of this model, state posteriors have to be approximated using variational inference [79]. The state and the auxiliary variables are estimated iteratively until convergence. The most common method in variational inference uses the mean-field assumption to factor the joint distribution of the state and the auxiliary variable as a product of each of the variable distributions. Then, the posteriors of both state and auxiliary variable are approximated by minimizing the Kullback–Leibler divergence [69] between the true posterior and the approximate posterior.

The variational inference filters presented in [4, 47, 96] are implemented here. All three methods share the same inference method, they only differ in the auxiliary random variable that is considered in the measurement model. An inverse-Wishart prior is assumed for R_k in [4]. The prior can be written as

$$R_k \sim \mathcal{W}^{-1}(sR, s), \tag{3.15}$$

where R is the nominal measurement noise, $s > d - 1$ is the degree of freedom with d as the dimension of the measurements. The conditional distribution of measurement $\mathbf{z}_k$ given the state $\mathbf{x}_k$ and the covariance R_k is

$$\mathbf{z}_k | \mathbf{x}_k, R_k \sim \mathcal{N}(H\mathbf{x}_k, R_k). \tag{3.16}$$

Variational inference approximates the posterior of the state $q(\mathbf{x}_k)$ with the standard filter correction step i.e Gaussian and $q(R_k)$ as

$$R_k|\mathbf{z}_k \sim \mathcal{W}^{-1}((s+1)\Lambda_k, s+1), \tag{3.17}$$

where

$$\Lambda_k = \frac{sR + S_k}{s+1},$$

and

$$S_k = (\mathbf{z}_k - H_k\hat{\mathbf{x}}_{k/k-1})(\mathbf{z}_k - H_k\hat{\mathbf{x}}_{k/k-1})^T + H_k\hat{P}_{k/k-1}H_k^T.$$

This posterior from [4] has an elegant interpretation. The posterior noise matrix is a convex combination of the nominal noise matrix and the expected sufficient statistics, s denoting the relative importance between the two.

Sarkka et al. [96] assume that R_k is fixed but is scaled by a scalar random variable λ_k which follows a Gamma distribution [111]

$$\lambda_k \sim \Gamma(\frac{\nu}{2}, \frac{\nu}{2}), \tag{3.18}$$

and the measurement distribution is

$$\mathbf{z}_k|\mathbf{x}_k, \lambda_k \sim \mathcal{N}(H\mathbf{x}_k, \frac{1}{\lambda_k}R_k). \tag{3.19}$$

Similar to the previous method, variational inference approximates $q(\mathbf{x}_k)$ with the

standard filter correction step and $q(\lambda_k)$ as

$$\lambda_k|\mathbf{z}_k \sim \Gamma(\frac{\nu+d}{2}, \frac{\nu+\bar{\gamma}_k}{2}), \tag{3.20}$$

where

$$\begin{aligned}\tilde{\gamma}_k &= \mathbf{E}_{\mathbf{x}}[(\mathbf{z}_k - h(\mathbf{x}_k))^T R_k^{-1}(\mathbf{z}_k - h(\mathbf{x}_k))] \\ &= \mathsf{tr}(\mathbf{E}_{\mathbf{x}}[(\mathbf{z}_k - h(\mathbf{x}_k))^T(\mathbf{z}_k - h(\mathbf{x}_k))]R_k^{-1}).\end{aligned} \tag{3.21}$$

The expectation inside the *trace* operator (tr) can be approximated using sigma-point operations [110]. The intuition behind the posterior of λ_k can be understood when examining the mean which is

$$\mathbf{E}(\lambda_k|\mathbf{z}_k) = \frac{\nu+d}{\nu+\bar{\gamma}_k}. \tag{3.22}$$

For the case of inliers, the measurements will be closer to their corresponding predictions resulting in smaller $\tilde{\gamma}_k$, thus larger λ_k and thus smaller uncertainty (larger weights). The opposite would happen for outliers resulting in larger $\tilde{\gamma}_k$, thus smaller λ_k and thus larger uncertainty (smaller weights).

The common aspect of both these methods is that after marginalizing out the auxiliary variable, the resulting measurement distribution has heavier tails (in this case, it is the Student's t distribution [79]). Both s in [4] and ν in [96] represent the extent of heavy-tailedness of the marginalized distribution, with $s, \nu = 1$ being the Cauchy distribution [111] and $s, \nu \to \infty$ denoting the Gaussian distribution. Huang et al. [47] assume inverse-Wishart priors for the state uncertainty and measurement noise covariance matrices and utilize the same inference method as above. Readers

are referred to [47] for further details. In the results sections, [4] is represented as $ORKF1$, [96] as $ORKF2$ and [47] as $ORKF3$ ($ORKF$ stands for Outlier Robust Kalman filter). Filtering (non-smoothed) versions of these algorithms are tested since this work analyses only real-time solutions.

3.2 Robust Wheel-Inertial Odometry for High-slip Terrain

Performances of the above-mentioned robust and adaptive Kalman filters are tested for wheel-inertial odometry in high-slip terrain. Wheel-inertial odometry can be useful in maintaining positioning performance in scenarios where cameras and lidars fail (places lacking visual or geometric features). IMU sensors suffer from error accumulation problems which can be reduced by non-holonomic constraints, and zero-velocity constraints [60]. However, the wheel velocity data obtained from wheel encoders is susceptible to wheel slippage resulting in erroneous wheel velocity measurements. In this chapter, the usefulness of robust Kalman filters for improving odometry performance is tested in such degraded scenarios. The base algorithm used here is $CoreNav$ [60, 62] which is an error-state EKF with the state vector represented in the local navigation frame by

$$\mathbf{x}^n_{err} = \begin{pmatrix} \delta\mathbf{\Phi}^n_{nb} & \delta\mathbf{v}^n_{eb} & \delta\mathbf{p}_b & \mathbf{b}_a & \mathbf{b}_g \end{pmatrix}^T, \tag{3.23}$$

where $\delta\mathbf{\Phi}^n_{nb}$ is the error in the attitude of the rover from the body frame (b) to the local navigation frame (n) expressed in the local navigation frame, $\delta\mathbf{v}^n_{eb}$ is the velocity error in the navigation frame, and $\delta\mathbf{p}_b$ is the position error. $\mathbf{b}_a$ and $\mathbf{b}_g$ are the IMU acceleration and gyro biases respectively. The position error vector is

$$\delta\mathbf{p}_b = \begin{pmatrix} \delta\mathbf{lat}_b & \delta\mathbf{lon}_b & \delta\mathbf{h}_b \end{pmatrix}^T, \tag{3.24}$$

which is expressed in latitude, longitude, and height frame. The states are propagated forward with measurements from IMU and corrected with wheel odometry measurements. The measurement $\mathbf{z}_k$ is also an error vector represented as

$$\delta\mathbf{z}_O = \begin{pmatrix} \tilde{\mathbf{v}}_{lon,O} - \tilde{\mathbf{v}}_{lon,i} \\ -\tilde{\mathbf{v}}_{lat,i} \\ -\tilde{\mathbf{v}}_{ver,i} \\ \tilde{\dot{\psi}}_{nb,o} - \tilde{\dot{\psi}}_{nb,i} cos\hat{\theta}_{nb} \end{pmatrix}, \tag{3.25}$$

where $\tilde{\dot{\psi}}_{nb}$ and $cos\hat{\theta}_{nb}$ are heading rate and pitch angle of the rover body frame with respect to the navigation frame respectively. $\tilde{\mathbf{v}}_{lon}$, $\tilde{\mathbf{v}}_{lat}$, and $\tilde{\mathbf{v}}_{ver}$ are predicted longitudinal, lateral and vertical rear wheel speed, respectively. The subscript i is used for values derived from the Inertial Navigation System (INS) solution, and O is used for values derived from the wheel odometry measurements. Motion constraints, such as zero velocity of the vehicle along the rotation axis of the wheels, and zero velocity perpendicular to the traversal surface are used as pseudo measurements to constrain the state vector [34]. It also utilizes zero-velocity-constraints [101], i.e. detects it

has stopped using wheel velocity data and uses that information to maintain INS alignment and reduce the rate of error growth.

3.2.1 Test Setup

The field tests are done with a testbed rover *Pathfinder* (figure 3.2.1). The IMU, wheel encoder, and Global Positioning System (GPS) receiver used in the rover, as similar to the work in [60, 62], are provided in table (3.2.1). To determine the reference truth solution, differential GPS (DGPS) is used. This solution is obtained by using two dual-frequency GPS receivers and Pinwheel (L1/L2) [81] mounted both on a GPS Base Station and on the testbed rover. A loosely coupled GPS/IMU fusion algorithm in [60] is used for the state initialization process. The GPS measurements (carrier phase and pseudorange) are recorded on both rover and base station receivers. The reference truth solution is generated by post-processing on RTKLIB 2.4.2 software [106]. Note that, DGPS accuracy is expected as cm/dm level [76].

Table 3.2.1: *Pathfinder's Sensor Specifications*

Sensor	Model	Data Rate
IMU	ADIS-16495 [9]	50 Hz
Wheel Encoder	Custom	10 Hz
GPS Receiver	Novatel [80]	10 Hz

Figure 3.2.1: *Pathfinder testbed platform*

3.2.2 Implementation

To compare the robust Kalman filtering methods, the implementation here builds upon the work in [60] which uses the Robot Operating System (ROS) [89] for collecting data from sensors. The error-state EKF uses IMU measurements (50 Hz) for state propagation and wheel odometry measurements (coming in at 10 Hz) as corrections. The state propagation step remains the same as $CoreNav$, whereas the correction step is modified to implement the robust methods. The HKF method depends on the parameter Δ. This parameter is determined by the degree of contamination in the assumed Gaussian error distribution, and empirically tuned, in that it is difficult to estimate for the application of four-wheel odometry. In this section Δ is set to 1.5, which provides a good balance of robustness and efficiency. The CSKF method does not require any tuning parameters, and the critical value of χ^2 distribution of degree of freedom 4 (size of measurement vector) with a significance level of 0.05, which is 9.488 is used for outlier detection. In $ORKF1$, s is 250. ν is set to 300

in $ORKF2$. Following [110], the sigma-point calculations in equation (3.21) sets α to 1, β to 2, and κ to 0. Following notations in [47], u is initialised to 2000, τ also 2000 with ρ set to 0.9999. All variational methods were run for 5 iterations. This low number is partly due to the fact that the increasing number of iterations resulted in skipping IMU measurements which is received at a very high rate within the real-time ROS implementation. Skipped IMU measurements resulted in unreliable results when repeated with the same data. For all implemented methods, parameters have been tuned for one of the data sets and re-used in the other ones, due to the similarity in the terrain and their trajectory shapes. The prior measurement and process noise remain the same for all methods. They are expected to vary from the ones mentioned above with different kinds of terrain requiring tuning from scratch. These parameters specify certain properties about the underlying process and measurement noise distribution which is discussed in the next section.

3.2.3 Results and Discussion

The algorithms $CoreNav$, HKF, $CSKF$, $ORKF1$, $ORKF2$, and $ORKF3$ are tested on the Pathfinder platform in three separate runs around a field of coal-ash residuals in Point Marion, PA. The three data sets from [46] are referred in this work as $test1$ (ashpile_mars_analog1.zip), $test2$ (ashpile_mars_analog2.zip), and $test3$ (ashpile_mars_analog3.zip). This terrain was chosen to be reasonably representative of planetary terrains. The solutions of the algorithms and truth (DGPS) are shown in ENU $(East - North - Up)$ frame, which is the local navigation frame with origin at the start location for each of the data sets. Figure (3.2.2) shows the trajectories

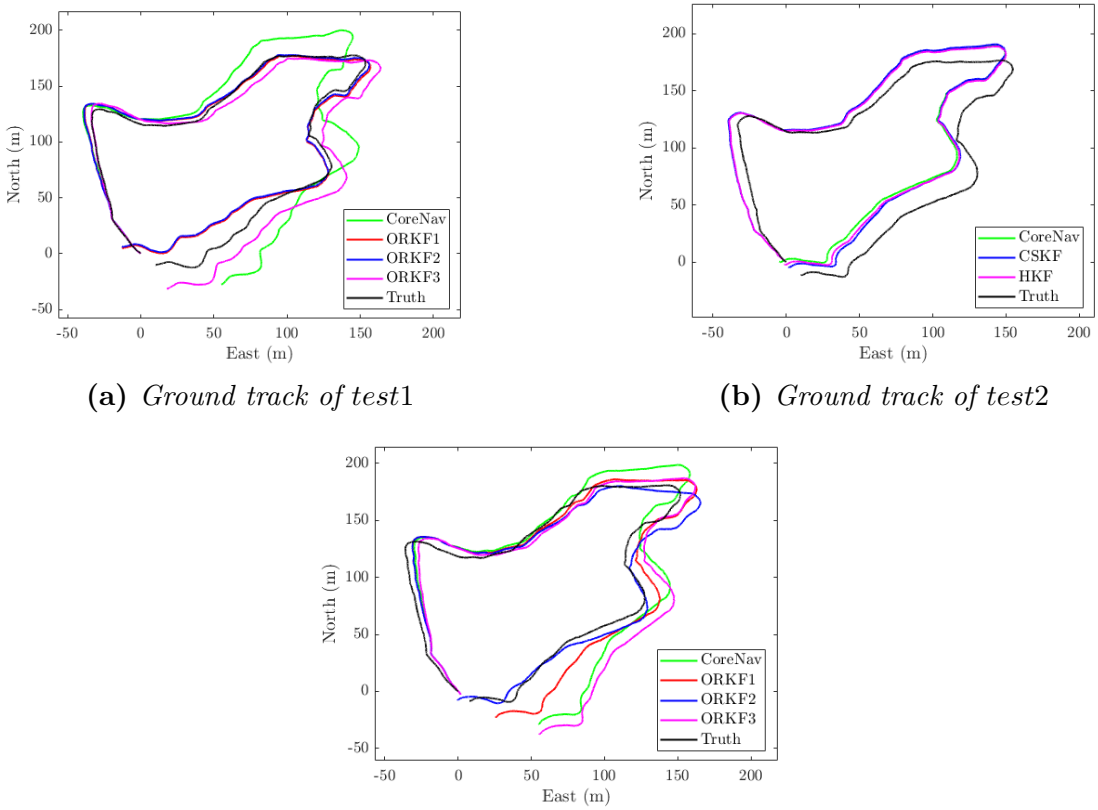
in the East-North plane. For ease of differentiating the trajectories, the figure shows only the methods that perform better compared to $CoreNav$ for each of the data sets. The 3D norm error and Up error statistics are shown as box plots in figures (3.2.3) and (3.2.4) respectively. Tables (3.2.2) and (3.2.3) show the 3D RMS errors and 3D maximum norm errors respectively. In data sets $test1$ and $test3$, $CSKF$ and HKF have comparable performance to $CoreNav$ while $ORKF1$ and $ORKF2$ outperform all other methods. In $test2$, however, the variational methods perform much worse with $ORKF1$ and $ORKF2$ being the worst affected. $CSKF$ and HKF produce good results, better than $CoreNav$ in this case. A general improvement in the Up (vertical) axis solution can be seen with the variational filters $ORKF1$ and $ORKF2$.

Table 3.2.2: *RMS Errors (m) of all solutions for data sets test1, test2, test3*

Method	$test1$	$test2$	$test3$
$CoreNav$	26.65	16.09	25.82
HKF	21.78	<u>14.67</u>	26.30
$CSKF$	34.76	**13.82**	27.18
$ORKF1$	<u>13.78</u>	33.32	<u>14.08</u>
$ORKF2$	14.13	43.86	**11.67**
$ORKF3$	**13.34**	18.18	26.13

* Best performance in bold and second-best underlined.

It is important to understand what the parameters in each of these methods mean and their assumptions. In HKF, as discussed before, the Δ parameter depends



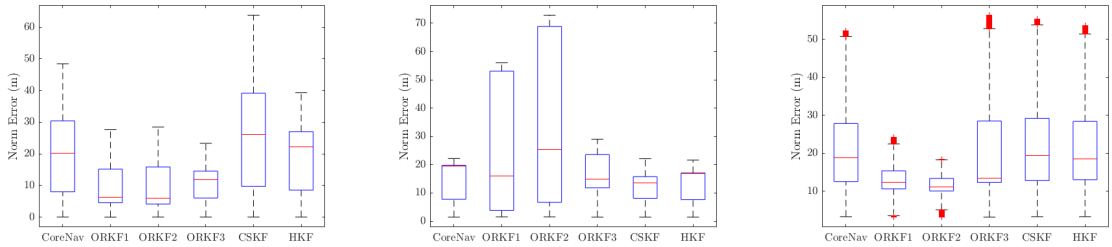
(a) *Ground track of test*1

(b) *Ground track of test*2

(c) *Ground track of test*3

Figure 3.2.2: *Ground Track (East-North planar trajectory) solutions of algorithms for data sets test*1*, test*2*, test*3 *from [46]. For ease of display, only trajectories that perform better than CoreNav are shown for each data set.*

on a perturbing parameter ϵ which represents the amount of contamination in the Gaussian error distribution. For $\Delta = 0$, the error distribution is assumed to follow the Laplace distribution [111] and $\Delta \to \infty$ follows the Gaussian distribution. Thus the distribution of the innovations $(y_{ik} - b_{ik}\mathbf{x}_k)$ will determine the best value of Δ at every time step. But with the smaller number of these innovations in each time step (15 states + 4 measurements = 19), it is hard to do so in this problem. However,

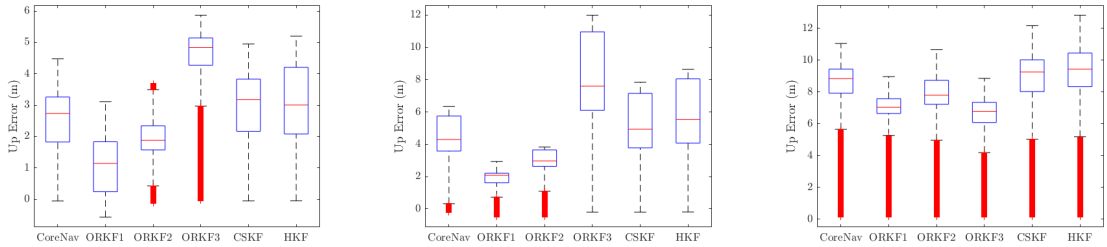


(a) *Norm Error (m) statistics of test*1

(b) *Norm error (m) statistics of test*2

(c) *Norm error (m) statistics for test*3

Figure 3.2.3: L_2 *norm error statistics of all solutions for data sets test*1*, test*2*, test*3



(a) *Up error (m) statistics of test*1

(b) *Up error (m) statistics of test*2

(c) *Up error (m) statistics of test*3

Figure 3.2.4: *Up error statistics of all solutions for data sets test*1*, test*2*, test*3

it is shown experimentally that increasing Δ results in convergence with *CoreNav* solution which assumes Gaussian error distribution (figure 3.2.5). This also points to the fact that the residuals do not follow a very heavy-tailed distribution for these data sets.

CSKF assumes the large residual is due to outliers in the measurements but could also be caused by wrong estimate of the predicted state. In that case, scaling the covariance matrix does not solve the problem. As discussed in [22], another disadvantage is that an outlier in one dimension of a measurement vector results in the

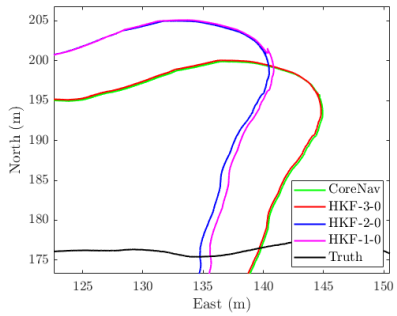
Table 3.2.3: *Max. 3D Norm Errors (m) of all solutions for data sets test1, test2, test3*

Method	$test1$	$test2$	$test3$
$CoreNav$	48.40	22.17	52.20
HKF	39.34	**21.62**	53.57
$CSKF$	63.79	<u>22.13</u>	55.40
$ORKF1$	<u>27.67</u>	56.05	<u>24.15</u>
$ORKF2$	28.43	72.85	**18.33**
$ORKF3$	**23.31**	29.04	56.35

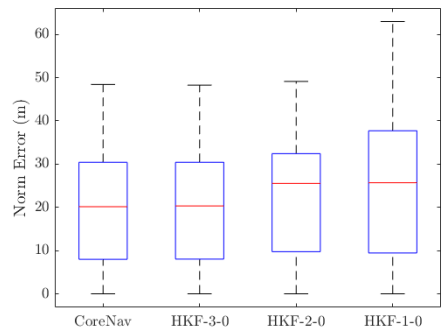
* Best performance in bold and second-best underlined.

de-weighting of the whole vector.

Variational filters also de-weight the measurements assuming a heavy-tailed model which appear to be more effective in this case. A large value of s (250) in $ORKF1$ points to more importance given to the nominal measurement noise covariance. This is confirmed by the parameter value of ν (300) in $ORKF2$ which is close to the value of s in $ORKF1$ since they essentially represent the same measurement model. Also, the value of the λ when running the algorithm remains within the range $(0.9, 1)$, which confirms our high belief in the nominal R value. Similarly in $ORKF3$, ρ value close to 1.0 results in lesser effect of new residuals on the measurement noise estimate. The advantage of $ORKF3$ is the ability to set the degree of belief in the prior state uncertainty for every epoch. The large value of τ represents a higher influence of the state uncertainty of the previous epoch on that of the current epoch. This can be favorable if the state uncertainty of the previous epoch was estimated correctly, but can get worse if it is not the case. Overall, $ORKF3$ performs more consistently



(a) *A magnified section of CoreNav, HKF, and truth (GPS) trajectory for test1 dataset*



(b) *Norm Error (m) statistics of CoreNav and HKF solutions for test1 dataset*

Figure 3.2.5: *Convergence of HKF solutions to CoreNav solutions with increasing* Δ *values from* 1.0 *to* 3.0 *for data set test*1

than the other two variational filters over the three datasets. The anomaly in the results is *test*2, which in spite of being very similar to *test*1 and *test*3, causes $ORKF1$ and $ORKF2$ to perform poorly. One of the probable reasons could be the lack of convergence in the variational update step. One of the main problems with applying the methods discussed above is the lesser number of measurements in this scenario unlike GPS or visual odometry which results in less information about the underlying measurement noise distribution.

As mentioned before, the parameter s in $ORKF1$ denotes the relative importance between the prior noise and sufficient statistics. There is another way of interpreting the effect of s. The harmonic mean of the posterior distribution of the covariance matrix is

$$\Lambda_k = \frac{sR + S_k}{s+1},$$

which can be rewritten as

$$\Lambda_k = \frac{sR}{s+1} + \frac{S_k}{s+1}.$$

The first part of the left-hand side expression is the prior noise which is not affected by the measurements and also relatively unaffected by s due to its ratio form. S_k in the second part is large for outliers and smaller for inliers. Now considering only the case of outliers, larger values of s will produce a smaller Λ_k than smaller values of s, thus de-weighting outliers less. A similar conclusion can be drawn for inliers as well. Thus lesser value of s leads to more de-weighting of both inliers and outliers and vice versa. Also since s is the degrees of freedom of the Student-t-distributed measurement model in [4], decreasing value of s makes the distribution more heavy-tailed and thus greater weighting. Table (3.2.4) shows better localization performance for larger values of s, which suggests the true residual distribution is close to a Gaussian.

Table 3.2.4: *RMS Errors (m) and Max. Norm Errors (m) of $ORKF1$ for data set $test1$ with varying values of s*

Method - s	$Max.Norm(m)$	$RMS(m)$
$ORKF1-10$	66.11	46.35
$ORKF1-50$	**28.15**	16.94
$ORKF1-250$	29.20	**14.45**

* Best performance in bold.

Table (3.2.5) shows better localization performance for larger values of ν which agrees with results of $ORKF1$. This can explain why the Huber cost function is

Table 3.2.5: *RMS Errors (m) and Max. Norm Errors (m) of $ORKF2$ for data set $test1$ with varying values of ν*

Method - ν	$Max.Norm(m)$	$RMS(m)$
$ORKF2-10$	83.54	50.28
$ORKF2-100$	57.51	33.28
$ORKF2-300$	**20.50**	**14.78**

* Best performance in bold.

not able to improve performance in this scenario. Since the residual distribution has been found to be close to Gaussian, the contamination parameter ϵ will be close to zero. Variation of the optimal parameter in Huber cost function with respect to ϵ has been shown in [58], confirming Δ values greater than 2 for ϵ close to zero. Thus greater values of Δ produced better results for HKF since it agrees more with the true measurement noise distribution.

The main disadvantage with this Kalman filter setup in the case of wheel inertial odometry is the small number of measurement residuals (4 from the 4 wheels) at hand at a time to actually learn the real-time residual distribution. One obvious solution could be to implement these filters in a sliding window setup, so that it has enough residuals from current and past epochs to correctly learn the residual distrbution. Recently, neural networks have been successfully used to learn noise characteristics of Kalman filters. Brossard et al. [19] implement an IEKF that learns pseudo-measurement noise for the update step from Convolutional Neural Network (CNN) using raw IMU data by minimizing the translational error with respect to GNSS solutions. Chen et al. [24] learn pedestrian translational and rotational dis-

placements for a window of IMU data using recurrent neural networks (RNN). Residual neural nets have been used in [68] to learn the displacements and uncertainties from a set of IMU measurements. However, all deep learning based methods used some form of high accuracy *truth* estimate for training their models which might not be always available. Variational methods on the other hand do not require training data. Thus, variational filter can still help in learning the measurement noise characteristics.

4

Factor Graphs

Factor graphs combine probabilistic graphical modeling [65] with least squares to solve estimation problems. The primary motivation of a factor graph is to represent a global function of many variables as a product of local functions with smaller subsets of variables. The factor graph is not a method but a framework for modeling any system using its locality structure, that is each variable is only dependent on a few

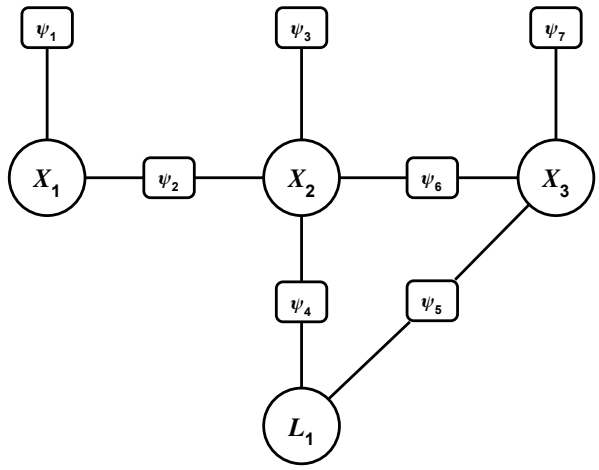


Figure 4.0.1: *Similar to HMM, a factor graph also models estimation problems. However, unlike HMM, they are undirected. Factor nodes (ψ) represent probabilistic constraints between state nodes ($\mathbf{X}$, L).* ***Note****: The factor ψ is not to be confused with the derivative of robust cost functions ($\psi = \rho'$).*

other local variables and is independent of the others. For example, an odometry measurement only depends on two states in the trajectory, pixel measurements of a landmark are a function of only the state that observed it and the landmark itself. As explained in [31], it is this locality property that makes it useful in modeling a variety of problems including mapping, visual-inertial odometry, motion planning, trajectory estimation, and deep learning. The factor graph is defined as a Bipartite graph that has two types of vertices, one is the variable vertex (e.g. unknown state) which is to be estimated and another one is the factor vertex which encodes the constraints (e.g., a set of GNSS observations) applied to the variable vertices. An edge can only exist between a factor vertex and a variable vertex. The factor vertices represent the local functions that depend on the variable vertices with which it shares edges. A

common estimation problem in robotics uses the factor graph framework to estimate the unknown robot pose along with other map points in a combined format. This is achieved by solving the MAP problem which maximizes the product of factors ψ that are probabilistic constraints between states and measurements. The factors ψ for robotic estimation are usually unary (prior factors) or binary (IMU pre-integration, odometry, loop closure factors, projection factors). For example, in figure (4.0.1), ψ_1, ψ_3, and ψ_7 are examples of prior factors. ψ_2 and ψ_6 are examples of odometry constraints. ψ_4 and ψ_5 are measurement constraints created by two poses observing the same landmark. A rigorous mathematical description of how this maximization problem is solved in the field of robotic perception is presented in [32]. The factor graph optimization problem can be written as

$$\mathbf{x}_{MAP} = \arg\max_{\mathbf{x}_0..\mathbf{x}_k} P(\mathbf{x}_0..\mathbf{x}_k|\mathbf{z}_0..\mathbf{z}_k) = \arg\max_{\mathbf{x}} \prod \psi_k(\mathbf{x}), \tag{4.1}$$

where

$$\psi_k(\mathbf{x}) \propto \exp\left\{-\frac{1}{2}\left\|h_k(\mathbf{x}_k) - \mathbf{z}_k\right\|^2_{\Sigma_i}\right\}. \tag{4.2}$$

Taking the negative logarithm of the MAP expression results in a batch non-linear least squares problem. This framework has attracted widespread use due to the flexibility of the approach and the ease of implementation granted through the availability of open-source graph optimization libraries like GTSAM [30], g2o [45] and Ceres [7]. Factor graphs can be compared to other estimation tools like the Kalman filter and its variants. To begin this comparison, it is to be noted that the factor graph ultimately encodes an objective function, which is solved through repeated relinearization via a

non-linear optimization routine (e.g., Gauss-Newton, Levenberg–Marquardt). Previous work has demonstrated that iterating on the measurement update of the Extended Kalman Filter and re-linearizing the system models between iterations is equivalent to a Gauss-Newton optimization [16]. Under a certain set of constraints, the factor graph operating in batch mode is equivalent to a backward-smoothing EKF [87] that re-linearizes system and observation models and iterates.

4.1 Advantages of Factor Graphs over EKF

Factor graph optimization has some advantages over the standard non-iterated Kalman filter which may be valuable for certain applications. Firstly, factor graphs are essentially batch optimization problems where iterative solvers can be used which helps in non-linear non-Gaussian cases. It linearizes the non-linear measurement model every iteration step for every state unlike the single linearization performed by the standard Kalman filter. They have also been shown to better exploit the time correlation between past and current epochs, which has been attributed to the batch nature of the estimation method. In particular, when operating in a batch mode, a factor graph would be equivalent to a forward filter and backward smoother after each measurement update. Factor graphs have been successfully applied to the field of visual-inertial SLAM [78, 88, 93], lidar SLAM [98]. It also helps easy implementation of loop closures which not only corrects the current states but also improves past states, resulting in accurate map generation. For GNSS/INS applications, these benefits have been supported by experimental results in [120] where factor graphs have

been shown to perform better than an EKF in urban environments. It might appear that with accumulation of new measurements over a longer time, the batch estimation might lose real-time performance. A sliding window approach can also be used similar to [123] to relieve computational cost. The window size is crucial for good optimization results and can depend upon environmental conditions [120]. Factor graphs achieve efficient computation by utilizing the sparse nature of the Jacobian and information matrices. This helps in fast matrix factorization and back-substitutions. Directly removing earlier poses from the graph can lead to information loss. This can be avoided using marginalization in the square-root information form which removes variables from the Bayes net derived from the factor graph using the elimination algorithm. Due to the sparseness in the graph, incremental QR factorization can be also achieved efficiently [30]. Beyond fixed-lag smoothing, the *isam2* formulation [55] achieves real-time performance by converting the factor graph to the Bayes tree [54] when a new constraint is added. This is a more accurate incremental and smoothing method for highly non-linear measurement models. The vertices of the Bayes tree represent cliques in the Bayes net obtained from the factor graph. Only states contained within the same cliques as the states in the new constraint and their predecessors in the Bayes tree need to be updated. Watson and Gross [116] used *isam2* in a GNSS factor graph to show improved positioning performance than a traditional EKF-Precise Point Positioning (PPP) method. Wen et al. [119] applied factor graph optimization to the problem of both GNSS and GNSS-Real-time Kinematic (RTK) positioning and showed better performance than an EKF. The optimization in factor graphs can be extended to include Lie groups without the need for any core changes

in the formulations. Robust estimation methods have been added to factor graphs to make the solutions more resilient to outliers like wrong feature matches, and visual aliasing for loop closures.

4.2 Robust Factor Graphs

Robustness has been added to factor graphs in different ways originating from the need to have robust loop closures in SLAM. Sunderhauf et al. [103] defined Switch Constraints (SC), which is a lifted optimization [128] methodology, that defines an observation weighting function $\varphi()$ that is a function of switch variables s, which is estimated in conjunction with the state parameters of interest. The SC method was initially developed for robust loop closure detection in SLAM and then extended to GNSS for multipath mitigation [105]. When utilizing switch constraints, the error associated with a factor is expressed as a scaled version of the Mahalanobis cost between the predicted and actual measurement. This is written as

$$\left\|\mathbf{e}_k^{\text{switch}}\right\|_{\mathbf{\Sigma}_k}^2 = \left\|\varphi\left(s_k\right) \cdot \left(y_k - h_k(x_k)\right)\right\|_{\mathbf{\Sigma}_k}^2, \tag{4.3}$$

where the function φ is a linear function of the switch variable. Prior factors are added for each switch variable to stop the optimization from setting all s_k to zero. A transition factor can also be added to model the change between s_{k-1} and s_k depending on the estimation problem. These switch functions help in automatically de-weighting erroneous measurements (e.g., multipath-affected GNSS measurements) and are seen to perform better than computationally expensive ray tracing methods [105]. An ex-

tension of SC was derived in [6] called Dynamic Covariance Scaling (DCS) where the switch variables are taken out of the optimization method and calculated separately using the residual, current measurement uncertainty, and a prior switch uncertainty. After calculating s_k, the information matrix associated with the factor is scaled by $\varphi(s_k)^2$. Max-mixtures (MM) [82] was also developed to tackle false loop closures using a Gaussian Mixture Model (GMM) but instead of the *sum* operator which is unsuitable for maximum likelihood when multi-modal uncertainty model is utilized, the objective function is converted to use the *max* operator, as shown in equation (4.4).

$$p\left(y|x\right) = \max_{k} w_k \mathcal{N}\left(\mu_k, \Lambda_k^{-1}\right) \tag{4.4}$$

The benefits of SC, DSC, and MM have been evaluated in [85, 115] for GNSS factor graph applications with real-world data. Both of these studies showed the substantial positioning improvement that can be granted via the utilization of robust estimation techniques when conducting optimization with degraded GNSS observations. To extend upon the max-mixtures work, Watson et al. [117] proposed to learn the GMM during run-time based upon clustering of the observation residuals. Initially, this work was implemented in a batch framework; however, it was later extended to work incrementally [118], through an efficient methodology for incrementally merging GMMs. M-estimators have also been recently tested within the GNSS framework in batch form [28] and found to perform better than non-robust estimators. The Huber cost

function [49], as provided in equation (4.5) is one such function.

$$\rho(x) = \begin{cases} \frac{x^2}{2} & |x| \leq \Delta \\ \Delta|x| - \frac{\Delta^2}{2} & |x| > \Delta \end{cases} \tag{4.5}$$

Increasing the Δ parameter makes this function closer to the squared loss function. As discussed in Chapter 2, M-estimators can be solved iteratively with weighted least squares method [18, 28]. Selecting a suitable Δ parameter is not straightforward, since it depends on the measurement noise statistics. Agamennoni et al. [5] uses the fact that some M-estimators like Huber, Cauchy, and Laplace have a corresponding elliptical distribution to estimate the Δ and the states in an Expectation Maximization (EM) framework. Barron [14] jointly optimizes for the states and the parameters for computer vision applications. A factor graph gives greater flexibility in the M-estimator application since it can help in de-weighting not only the current measurements but also changing the weights of the past measurements. It also can help in totally removing some past measurements if it is found to be an outlier later whereas in the Kalman filter, the contribution of past measurements cannot be changed in a real-time manner. Most graph optimization libraries also have built-in functionality to use robust cost functions which is also helpful.

4.2.1 Square-root Factors in Factor Graphs

As discussed before, factor graphs solve a batch non-linear least squares problem,

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \prod_k \psi_k(\mathbf{x}) = \arg\min_{\mathbf{x}} \sum_k \|\mathbf{r}_k(\mathbf{x})\|^2 .$$

This squared form comes from the Gaussian distribution formula. However, this squared form is lost when the Gaussian assumption is replaced by a different distribution like the Cauchy or Exponential distributions. Can residuals coming from non-Gaussian distribution be still integrated into a factor graph framework and solved by solvers like GTSAM?

Rosen et al. [92] show that a residual function can be added into a factor graph solver by representing them in square root form. Let ψ_k be the non-Gaussian distribution function and c_k be a real number such that $c_k \geq ||\psi_k||_\infty$. Then residual can be defined as

$$r_k(\mathbf{x}) = \sqrt{c_k - ln(\psi_k(\mathbf{x})}. \tag{4.6}$$

The expression inside the square root is non-negative by design. Thus MLE problems for non-Gaussian distributions can still be converted to non-linear least squares problems. This idea essentially presents a way to replace the squared cost in factor graphs with robust cost functions from M-estimators. The distribution associated with M-estimators is a generalized version of Elliptical distribution [5]. Converting the MLE problem to least squares problem for M-estimators results in the error

function $\sqrt{\rho(r_k(\mathbf{x}))}$. The Jacobian can be expressed as

$$H = \frac{\partial\sqrt{\rho(r_k(\mathbf{x}))}}{\partial\mathbf{x}} = \frac{1}{2}\frac{\rho^{'}(r_k(\mathbf{x}))}{\sqrt{\rho(r_k(\mathbf{x}))}}\frac{\partial r_k(\mathbf{x})}{\partial\mathbf{x}}. \tag{4.7}$$

Note here $\frac{1}{2}\frac{\rho^{'}(r_k(\mathbf{x}))}{\sqrt{\rho(r_k(\mathbf{x}))}}$ acts as the weight to the standard jacobian $\frac{\partial r_k(\mathbf{x})}{\partial\mathbf{x}}$.

4.2.2 Black-Rangarajan Duality

Previously in Chapter 2, the connection between M-estimators and weighted least squares has been discussed. Black et al. [17] derived an equivalence between the M-estimator formulation and weighted least squares from the perspective of regularized least squares. The equivalence can be described as,

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{k=1}^{n} \rho(r_k(\mathbf{x}))$$

$$\Updownarrow$$

$$\hat{\mathbf{x}}, \hat{\mathbf{w}} = \arg\min_{\mathbf{x},\mathbf{w}} \sum_{i=k}^{n} w_k r_k(\mathbf{x})^2 + \Psi(w_k).$$

Ψ is used as a regularizer or penalty function which decreases with increasing weights. As discussed before, the standard way of solving M-estimators is IRLS. This is an iterative process, where the algorithm starts with an initial estimate of $\mathbf{x_0}$ and iterates over solving $\mathbf{w}$ and $\mathbf{x}$ until convergence.

Algorithm 3: Iterative Re-weighted Least Squares (IRLS)

Input: $\hat{\mathbf{x}}_0$

Output: $\hat{\mathbf{x}}$, $\hat{\mathbf{w}}$

while !converged **do**

% Step 1: solve for $\mathbf{w}$

$\hat{\mathbf{w}}_j = \arg\min_{\mathbf{w}} \sum_{k=1}^{n} w_k r_k(\hat{\mathbf{x}}_{j-1})^2 + \Psi(w_k)$;

% Step 2: solve for $\mathbf{x}$

$\hat{\mathbf{x}}_j = \arg\min_{\mathbf{x}} \sum_{k=1}^{n} \hat{w}_{kj} r_k(\mathbf{x})^2$;

end

Step 1 in algorithm (3) can be calculated in two ways. As shown in chapter 2, one way to get $\mathbf{w}$ is to equate the derivatives with respect to $\mathbf{x}$, which gives the expression $w_k = \frac{\rho'(r_k(\mathbf{x}))}{r_k(\mathbf{x})}$. This expression uses the ρ expression. The other way is to use the Ψ function and solve the equation,

$$\frac{\partial}{\partial w} w r(\mathbf{x})^2 + \Psi(w) = 0 \tag{4.8}$$

This method can still suffer from local minima issues in the case of re-descending M-estimators. To solve this problem, authors of [125, 133] use a technique called Graduated Non-Convexity(GNC). The idea is to replace the function ρ (or Ψ) with a surrogate function ρ_μ (or Ψ_μ) whose convexity is controlled by the parameter μ. The optimization starts with a convex form of ρ_μ after which μ is changed iteratively such that the non-convexity increases. The intuition behind GNC is that by gradually moving from the convex to concave functions the initialization point for each step

remains close to the global minima which reduces the chance of getting stuck at local minima. Wen et al. [121] show the importance of applying this robust estimation technique in GNSS factor graphs for mitigating multipath effects in urban canyons. The switchable constraints method for robust loop closures [104] discussed before presents the same idea, where the switch function φ is the weight and the switch prior is the Ψ penalty function.

4.3 GNSS Factor Graphs

As discussed in Chapter 3, one disadvantage of the wheel-inertial odometry problem was that there weren't enough measurements coming in at a specific time to confidently learn the error distribution. This motivates the application of robust estimators on the GNSS positioning and the point cloud registration problems.

The use of GNSS positioning is ubiquitous in the modern world. From the simple task of navigating a city in Google Maps to complex tasks like precise commercial and military aircraft landings, satellite positioning systems are integral to the safe operation of many modern infrastructures. Before going into the details of robust GNSS algorithms it is important the know the variables usually estimated in GNSS positioning problems. This problem can be described as solving for the position of the receiver given the distance measurements from the satellites and given the satellite locations in the Earth-Centered-Earth-Fixed (ECEF) frame. The unknown states include the P_{xr}, P_{yr}, P_{zr} values of the receiver 3D position, the tropospheric propagation delay T_d and the receiver clock bias δt_r. The measurement model can be

written as

$$d_o = d + c\delta t_r + T_d\mathcal{M}(el) + \epsilon. \tag{4.9}$$

d_o is referred to as pseudorange observable. This is the measured *distance* between the satellite and the receiver which includes the geometric distance between the satellite and receiver (d), the added distance due to clock synchronization errors between the satellite and the receiver ($c\delta t_r$), and errors caused by atmospheric effects from troposphere ($T_d\mathcal{M}(el)$). $\mathcal{M}$ is a mapping function which takes the elevation (el) of the satellite as an input. If the noise ϵ is assumed to be Gaussian, then the positioning problem becomes a non-linear least squares problem. The non-linearity comes from the term d which can be expressed as

$$d = \sqrt{(P_{xr} - P_{xs})^2 + (P_{yr} - P_{ys})^2 + (P_{zr} - P_{zs})^2}, \tag{4.10}$$

where $\{P_{xs},\, P_{ys},\, P_{zs}\}$ is the position of the satellite. If a receiver receives N satellite signals at a specific time, the position of the receiver can be estimated by solving the equation

$$\hat{X}_r = \arg\min_{X_r} \sum_{i=1}^{N} (d_i - d - c\delta t_r - T_d\mathcal{M}(el_i))^2 \tag{4.11}$$

A visual representation of the GNSS factor graph is provided in figure (4.3.1), where ψ represents any probabilistic constraint that might exist between the states and the measurements. In this specific implementation, ψ^p encodes the prior belief on each state, which depends on the specific data set and environmental characteristics. ψ^b is a motion constraint between two consecutive states along the trajectory which could, for example, incorporate motion data from an IMU or wheel odometry. Using GT-

SAM nomenclature, ψ^b is referred to here as a between factor. A common example of ψ^b used in GNSS/IMU navigation is the factor that uses IMU-preintegration [41] to calculate transformation between two locations with multiple IMU measurements integrated between them. Finally, ψ^m are measurement constraints between a state and the measurements that were perceived from that state, for example, GNSS pseudorange or carrier-phase measurements. The MAP estimate for the GNSS factor graph is the set of states that maximize the product of factors. However, in practice, this optimization problem can be greatly simplified by employing the Gaussian noise assumption, which enables the conversion of the problem from maximizing the product of the factors to a non-linear least squares problem where each component of the sum is a Mahalanobis cost, which represents sum of squares of the normalized residuals, as provided in equation (4.12),

$$\begin{aligned}\hat{X} &= \underset{x}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \|\psi_i^p\|_\Sigma^2 + \sum_{j=1}^{J} \left\|\psi_j^b\right\|_\Lambda^2 + \sum_{k=1}^{K} \|\psi_k^m\|_\Xi^2 \right] \\ &= \underset{x}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \|x_o - x_i\|_\Sigma^2 + \sum_{j=1}^{J} \|x_j - f_j(x_{j-1})\|_\Lambda^2 + \sum_{k=1}^{K} \|y_k - h_k(x_k)\|_\Xi^2 \right]\end{aligned} \tag{4.12}$$

where $f(*)$ is a mapping between states at different epochs and $h(*)$ is a mapping from the state space to the observation space. A detailed description of creating factors with GNSS observations is presented in [116].

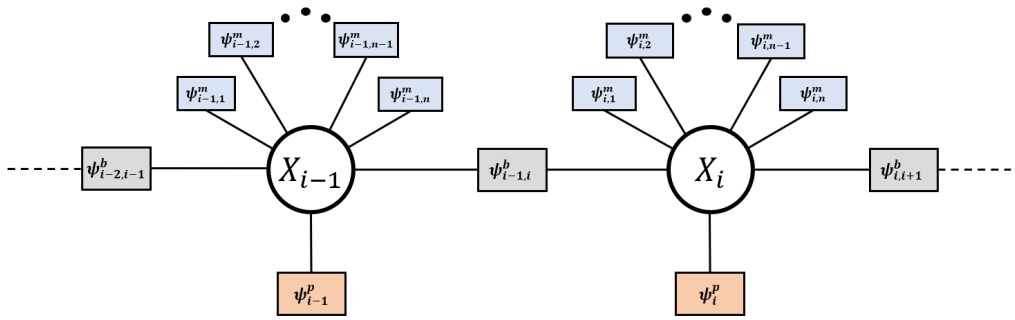


Figure 4.3.1: *A GNSS factor graph example. Each blue node represents GNSS measurement from a satellite. Grey nodes represent odometry constraints, and red nodes represent prior constraints.*

4.4 ZUPT aided GNSS Factor Graph

The work described in this section does not fall within the typical definition of robust estimation algorithm, since other sensor sources are used to reduce the effect of the noisy sensor. This is a fusion algorithm where a GNSS factor graph uses information from the error state EKF discussed in the previous chapter in the form of added motion constraints.

4.4.1 Zero-velocity Update (ZUPT)

Zero velocity information is often used in wheeled motion where the lateral and vertical motion in the body frame is assumed to be zero. Another velocity information that is useful for constraining the state of a body is when it is stationary. For instance, ZUPT is commonly used as an aiding process for pedestrian navigation [66, 130]. ZUPT can put a bound on the velocity error and help to calibrate IMU sensor

noises [101]. This process helps to reduce the INS positioning error growth from cubic to linear since the error state model justifies the correlation between the position and velocity errors of the error covariance matrix. Using ZUPT in state estimation does not need any dedicated sensor to provide zero velocity information, and this information can be obtained by the sensors already on board (e.g., IMU, wheel encoders). ZUPT requires stationary conditions, and it can be used as an opportunistic navigational update if a wheeled robot stops for other reasons (e.g., obstacle avoidance, re-planning, waiting for pedestrians, stopping at traffic lights). Also, ZUPT can be used to improve WO/INS proprioceptive localization with periodic stops in GNSS-denied (or degraded), poor lighting/feature areas [60] and with autonomous stops by deciding when to stop [61]. The small-wheeled robots have more freedom of stopping than autonomous cars, which makes utilizing ZUPT a well-suited application for them.

4.4.2 GNSS/WO/INS Integration with ZUPT

This section presents two factor graph strategies: 1) utilizing only the zero velocity information in a factor graph method, and 2) leveraging the CoreNav position estimates (which are improved with ZUPT and non-holonomic constraints) in a factor graph method. The GTSAM library is used for graph optimization with the standard squared loss function (L2).

The first method does not use INS state estimates in the factor graph directly. It only uses zero velocity signals when INS detects the rover has stopped. To do this, a high certainty zero displacement between-factor $\psi_{b,j}$, referred to in the figure

(4.4.1) as ψ^z, is added between the two adjacent state vertices, which are known to be stationary. A low certainty zero-displacement between factor is also added between non-ZUPT vertices, where the rover moves, to represent process noise and let the ZUPT information propagate throughout the graph and improve the overall solution. The ψ^z factors are also referred to as the ZUPT factors in the results section. An illustration of this first method is given in figure (4.4.1) and called L2-ZUPT.

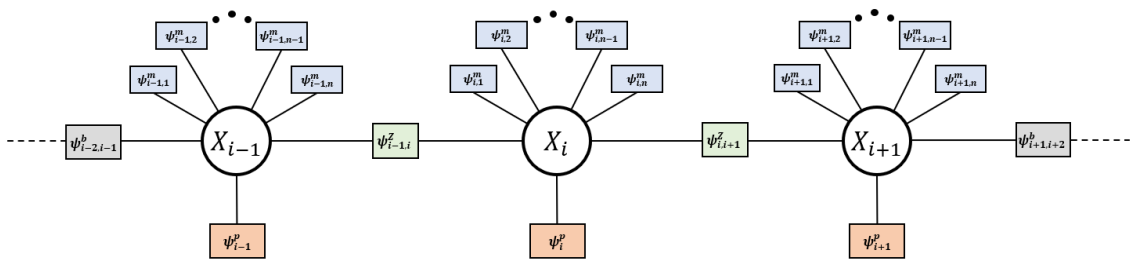


Figure 4.4.1: *L2-ZUPT factor graph has high uncertainty zero displacement factors (grey) for dynamic epochs and low uncertainty zero displacement factors (green) for static epochs.*

The second method has a more direct coupling between the GNSS and the INS/WO part. Here instead of utilizing the zero velocity information directly in the factor graph, the positioning solution from the CoreNav error state EKF sensor fusion method following [60] is used. Note that, in the CoreNav method, the zero velocity information is used as a ZUPT and also includes the non-holonomic constraints to improve the localization solution further. To couple the EKF estimates with the GNSS factor graph, the obtained positioning estimates from the CoreNav method are added as between-factors ψ^{CN} among all state vertices. The ψ^{CN} factors are referred to as the CoreNav factors. A depiction of the second method called L2-CN is given

in figure (4.4.2). It needs to be clarified that the implementation of L2 factor graph in this work has no explicit between factor between nodes but instead, state nodes are connected through shared phase bias of the same satellite.

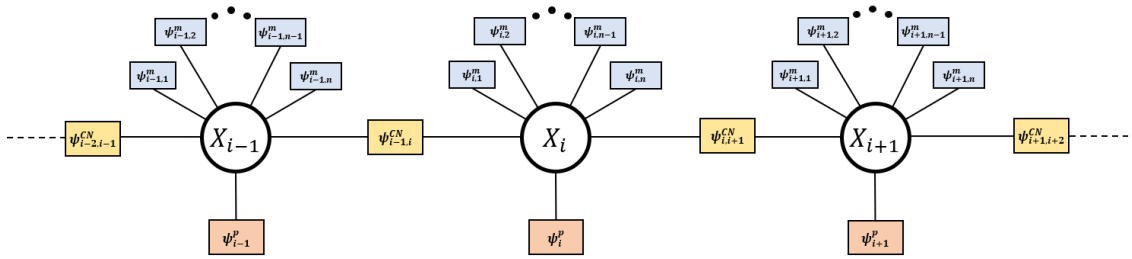


Figure 4.4.2: *L2-CN factor graph uses odometry constraints (yellow) from a parallel running wheel-inertial EKF.*

4.5 Experimental Results

To test this fusion algorithm, the rover Pathfinder (same as in Chapter 3) is used. The localization sensor suite setup includes a Novatel pinwheel L1/L2 GNSS antenna [81], and receiver [80], an Analog Devices ADIS-16495 IMU [9], and quadrature wheel encoders. The robot is also equipped with an Intel RealSense T265 camera [52]; however, this sensor is not used in the localization solution. The computer is an Intel NUC Board NUC7i7DN [51] which hosts an i7-8650U processor. To evaluate the methods, three datasets from [46] are used and referred to in this chapter as Test 1 (ashpile_mars_analog1.zip), Test 2 (ashpile_mars_analog2.zip) and Test 3 (ashpile_mars_analog3.zip). Noisy versions of these datasets are generated by adding simulated multipath noise based on the elevation of the satellites following the early-

minus-late discriminator formulation in [67]. The noise is randomly added to 2 % of the data in each dataset to create the noisy versions. Histogram plots of the simulated range and phase noises are shown in figure (4.5.1). The original data is referred to as clean data in the results. The rover stopped 9 times for test 1, 19 times for test 2, and 20 times for Test 3 to obtain the zero velocity information. The distances covered in tests 1, 2, and 3 are 671m, 652m, and 663m, respectively. The reference position solutions are obtained by integer-ambiguity-fixed carrier-phase differential GPS (DGPS) processed with RTKLIB [106]. The GNSS factor graph is run in an offline manner where it reads the solutions of the EKF from a file and adds the motion constraints by matching time stamps. The three methods compared

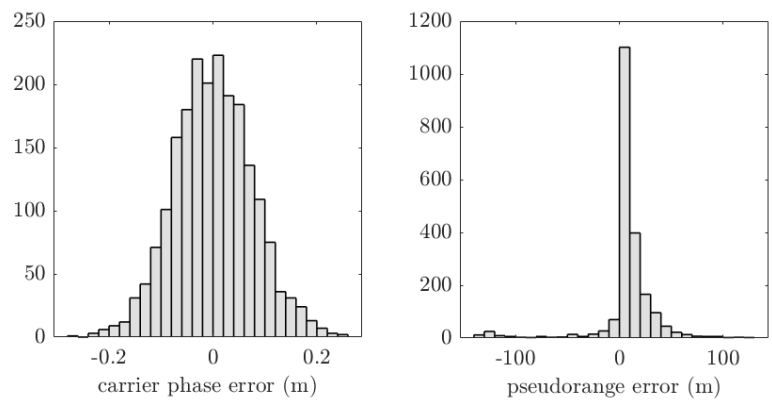


Figure 4.5.1: *Histogram of simulated multipath phase and range errors [67]*

here are the standard GNSS factor graph (L2), the GNSS factor graph with ZUPT factors (L2-ZUPT), and the GNSS factor graph with CoreNav factors (L2-CN). A comparison of the methods for the clean version of the datasets is given in table (4.5.1). The comparison for the noisy datasets is shown in table (4.5.2). Figures

(4.5.2) and (4.5.3) show the time variation of the errors in the East-North-Up frame. The larger peaks in figure (4.5.3) in the standard factor graph results are due to the large simulated multipath noise.

Table 4.5.1: *Comparison of the methods for the clean datasets.*

Clean Dataset		RMSE (m)				Max Norm Error (m)
		E	N	U	3D	3D
Test 1	L2	0.62	0.87	3.82	3.97	5.65
	L2-ZUPT	0.62	0.34	2.09	2.20	3.06
	L2-CN	0.62	0.78	3.34	3.49	4.97
Test 2	L2	0.49	0.31	1.30	1.43	4.31
	L2-ZUPT	0.46	0.93	1.24	1.62	2.62
	L2-CN	0.51	0.24	0.90	1.06	3.08
Test 3	L2	0.16	0.92	3.86	3.97	6.14
	L2-ZUPT	0.16	0.96	3.55	3.69	4.47
	L2-CN	0.16	0.92	3.58	3.70	5.65

* The best results marked up with green boxes

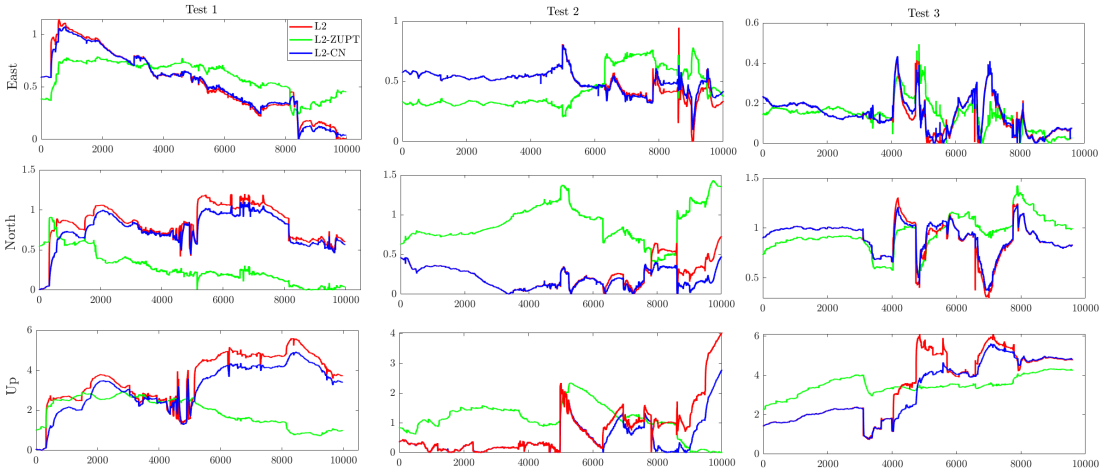


Figure 4.5.2: *Time variation of the errors (m) in the East-North-Up frame for clean datasets.*

Table 4.5.2: *Comparison of the methods for noisy datasets.*

Noisy Dataset		RMSE (m)				Max Norm Error (m)
		E	N	U	3D	3D
Test 1	L2	0.88	0.87	2.85	3.10	67.81
	L2-ZUPT	0.88	0.78	2.27	2.55	22.39
	L2-CN	0.67	0.77	2.72	2.90	7.29
Test 2	L2	0.53	0.40	1.77	1.90	16.12
	L2-ZUPT	0.59	0.33	1.36	1.52	4.85
	L2-CN	0.53	0.24	1.01	1.16	3.41
Test 3	L2	0.23	0.90	3.59	3.71	7.52
	L2-ZUPT	0.17	0.93	3.65	3.77	5.08
	L2-CN	0.23	0.88	3.55	3.66	6.33

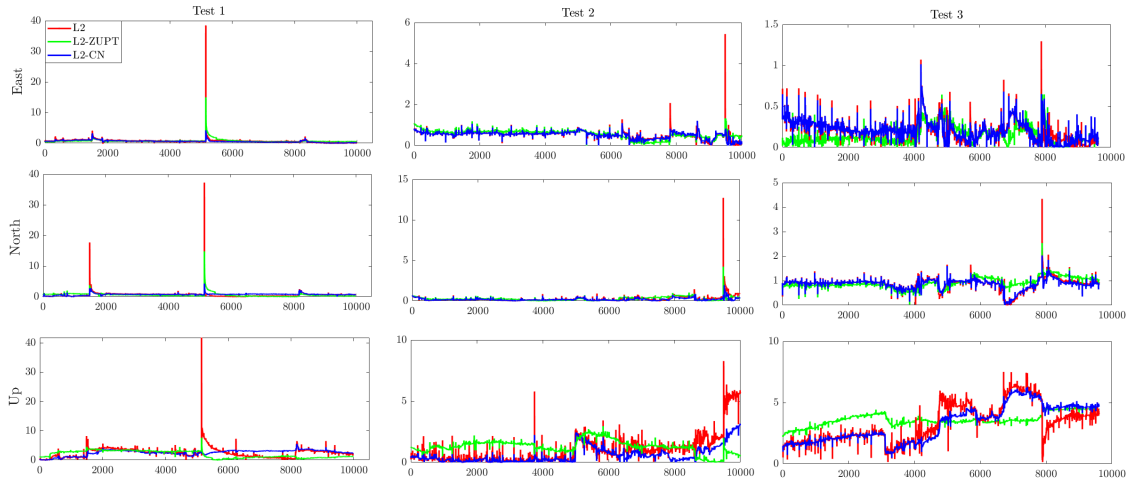


Figure 4.5.3: *Time variation of the errors (m) in the East-North-Up frame for noisy datasets.*

L2-ZUPT and L2-CN have better performance than the GNSS-only factor graph, L2, for clean and noisy datasets 1 and 2. Comparable performances can be seen from all three methods for dataset 3. L2-ZUPT is found to perform best for clean data, and L2-CN performs best for noisy data. The effect of leveraging zero velocity information can be seen in the noisy data results, where the larger errors in the standard factor graph are dampened by constraining the states during the ZUPTs to

be the same since the rover is stationary at that time. The better performance of L2-CN for clean and noisy data can be explained by the fact that the factor graph utilizes more information since it uses the IMU-WO solution from CoreNav. CoreNav also uses additional non-holonomic constraints which are not used in the L2 and L2-ZUPT methods. The performance gap between L2-ZUPT and L2-CN is affected by the number of stops in the datasets. For example, a more significant performance gap can be seen between L2-ZUPT and L2-CN in Test 1, which has fewer stops. This performance gap is smaller in Test 2, and 3 where the number of stops is more than in Test 1, which indicates that using only ZUPT factors in the GNSS factor graph can provide similar localization performance as using a GNSS/WO/INS coupled solution.

Future works will involve testing with more extended datasets with real multipath noise and investigating the connection with robust filtering algorithms. The incremental covariance estimation method discussed in [118] was also tested. The ZUPT factors are expected to help learn the measurement noise covariance model better. However, the parameter tuning of incremental covariance estimation and ZUPT became a challenge, and due to this, consistent results could not be found with these datasets, which could be attributed to the shorter length of the trajectories.

5

Generalised Robust Cost Functions

M-estimators come in various functional forms, some convex and some concave. So the natural question to ask is: are there some functions more favorable in terms of robustness and efficiency than others? Maronna et al. [74] show that the Bi-square estimator has a better efficiency trade-off than the Huber estimator for Gaussian and Cauchy distributions when estimating the location of a scalar parameter. Huber and

Pseudo-Huber estimators have been recommended in [2] for rotation averaging, triangulation, and point cloud registration problems. However, Geman McClure estimator has been suggested in [70] for the initialization of IRLS and the truncated least squares for the ending. From the literature, there does not seem to be a clear consensus on which estimator to use. One option would be to use adaptive estimators based on the learned noise characteristics. It is advantageous to have an adaptive M-estimator that can adapt its functional shape depending on the current set of residuals.

5.1 One Function for All

The approach proposed here builds on Barron's work on unifying different robust cost functions [14].

$$\rho(x, \alpha, c) = \begin{cases} \frac{1}{2}(x/c)^2 & \text{if } \alpha = 2 \\ \log\left(\frac{1}{2}(x/c)^2 + 1\right) & \text{if } \alpha = 0 \\ 1 - \exp\left(-\frac{1}{2}(x/c)^2\right) & \text{if } \alpha = -\infty \\ \frac{|\alpha-2|}{\alpha}\left(\left(\frac{(x/c)^2}{|\alpha-2|} + 1\right)^{\alpha/2} - 1\right) & \text{otherwise} \end{cases} . \tag{5.1}$$

This form of cost function is convenient because different variations of M-estimators can be expressed by changing the parameter α. x is the residual value depending on the estimation problem at hand. c is sometimes referred to as the scale parameter. For readability reasons, $r(\mathbf{x})$ is replaced by x. This section aims to understand the effects of changing α and c values in different robust estimation scenarios. As described

earlier, M-estimators de-weight suspected outlier residuals instead of removing them completely. This is helpful in cases where removing data can affect the solution accuracy such as GNSS estimation with a low number of available observations, or visual odometry in environments with limited features. The weight depends on the derivative ρ' and the residual x. The partial derivative of this cost function with respect to x is given by

$$\frac{\partial \rho}{\partial x}(x, \alpha, c) = \begin{cases} \frac{x}{c^2} & \text{if } \alpha = 2 \\ \frac{2x}{x^2+2c^2} & \text{if } \alpha = 0 \\ \frac{x}{c^2} \exp\left(-\frac{1}{2}(x/c)^2\right) & \text{if } \alpha = -\infty \\ \frac{x}{c^2}\left(\frac{(x/c)^2}{|\alpha-2|} + 1\right)^{(\alpha/2-1)} & \text{otherwise.} \end{cases} \tag{5.2}$$

The unknown state can be solved by optimizing the cost function given by

$$\hat{X} = \underset{\theta,\alpha,c}{\arg\min} \sum_k \rho\left(x_k(\boldsymbol{\theta}), \alpha, c\right). \tag{5.3}$$

To clarify the mathematical notation again here, $\boldsymbol{\theta}$ and x are equivalent to the state $\mathbf{x}$ and $r(\mathbf{x})$ from previous chapters respectively. Robustness metrics along with bias and variance of this generalized M-estimator for a simplified model are presented in the Appendix A section. A better understanding of the optimization problem can be obtained by looking at the partial derivatives of ρ with respect to α. Taking the partial derivative it can be seen that

$$\frac{\partial \rho}{\partial \alpha}(x, \alpha, c) \geq 0. \tag{5.4}$$

Since ρ in equation (5.1) is even with respect to c, only positive values for c are used. Equation (5.4) shows the cost decreases with decreasing α when c is constant. The problem of optimizing equation (5.3) for (x, α) is that the solution will trivially move towards lower values of α, thus not representing the true distribution of the residuals and, in turn, affecting the estimates of the unknown parameters. Barron [14] removes this issue by assuming a distribution given by

$$P_\star(x, \alpha, c) = \frac{1}{cZ(\alpha)} e^{-\rho(x,\alpha,c)}, \tag{5.5}$$

where

$$Z(\alpha) = \int_{-\infty}^{\infty} e^{-\rho(x,\alpha,1)} dx. \tag{5.6}$$

This creates a regularized version of the cost function. Using negative log-likelihood on equation (5.5),

$$\rho_\star(x, \alpha, c) = \rho(x, \alpha, c) + \log(cZ(\alpha)). \tag{5.7}$$

With this, whenever $\rho_\star$ is optimized for (x, α), the solution cannot trivially go to the least value of α due to the newly added penalty term. The optimization process attempts to balance the lower cost of larger residuals and the higher cost of the inliers. However, another problem arises with this shifted expression, which is that $Z(\alpha)$ is unbounded for negative values of α. Thus the optimization cannot be done in the negative domain of α, which is not ideal because negative α values can be useful for large residuals. To circumvent this issue Chebrolu et al. [23] used a truncated version

of $Z(\alpha)$,

$$\hat{Z}(\alpha) = \int_{-\tau}^{\tau} e^{-\rho(x,\alpha,1)} dx. \tag{5.8}$$

$\hat{Z}(\alpha)$ can be calculated for both positive and negative values of α. The only assumption with this formulation is that any residual with a magnitude greater than τ has zero probability. Thus $Z(\alpha)$ in equation (5.5) can be replaced with $\hat{Z}(\alpha)$.

Instead of jointly optimizing over (x, α), Chebrolu et al. [23] first find the α that has the lowest negative log-likelihood with the current residuals and then solve equation (5.3) with iterative re-weighted least squares with this last optimal value of α. These two steps are repeated until convergence is achieved. c is kept constant in this method and depends on the inlier measurement noise. The algorithm is described in algorithm (4). This algorithm is referred to in this dissertation as RKO.

Algorithm 4: Robust Kernel Optimization (RKO) [23]

Input: $\boldsymbol{\theta}^0$, α^0, c

Output: $\hat{\boldsymbol{\theta}}$, $\hat{\alpha}$

while !converged **do**

% Minimize for α

$\alpha^t = \arg\min_{\alpha} - \sum_k \log P_\star \left(x_k \left(\boldsymbol{\theta}^{t-1}\right), \alpha, c\right)$;

% Minimize for $\boldsymbol{\theta}$ using IRLS

$\boldsymbol{\theta}^t = \arg\min_{\boldsymbol{\theta}} \sum_k \rho \left(x_k(\boldsymbol{\theta}), \alpha^t, c\right)$;

end

5.2 Scale-variant Robust Kernel Optimization

Algorithm (4) has been shown to work well for Lidar SLAM in the presence of dynamic objects as well as bundle adjustment [23]. However, manually setting the scale parameter c is difficult and is often done by trial and error. This section focuses on finding a way to learn c along with x and α to yield further improvements in such situations and reduce parameter tuning efforts. To that end, a slightly different variation of the probability distribution $\hat{P}$ is proposed.

$$\hat{P}(x, \alpha, c) = \frac{1}{\hat{Z}(\alpha, c)} e^{-\rho(x,\alpha,c)}, \tag{5.9}$$

where

$$\hat{Z}(\alpha, c) = \int_{-\tau}^{\tau} e^{-\rho(x,\alpha,c)} dx. \tag{5.10}$$

Following the assumption that residuals fall within the range $[-\tau, \tau]$, $\hat{P}(x, \alpha, c)$ is valid distribution since $\int_{-\tau}^{\tau} \hat{P}(x, \alpha, c) dx = 1$. Similar to the regularized cost discussed in the previous section, the cost corresponding to this distribution is obtained by taking the negative log-likelihood

$$\hat{\rho}(x, \alpha, c) = \rho(x, \alpha, c) + \log(\hat{Z}(\alpha, c)). \tag{5.11}$$

The behavior of this probability distribution can be understood by examining figure (5.2.1). In the figure (5.2.1 left), $\hat{P}$ shows behavior similar to the $P_\star$. In the presence

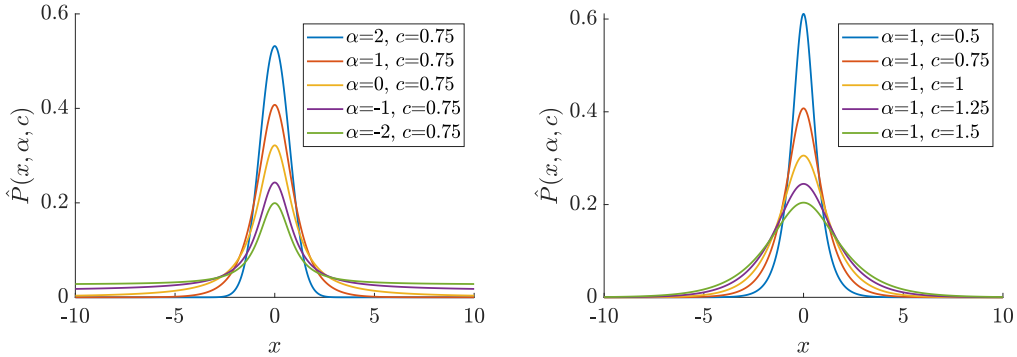


Figure 5.2.1: *Left:Probability density function for constant c and changing α, Right:Probability density function for constant α and changing c*

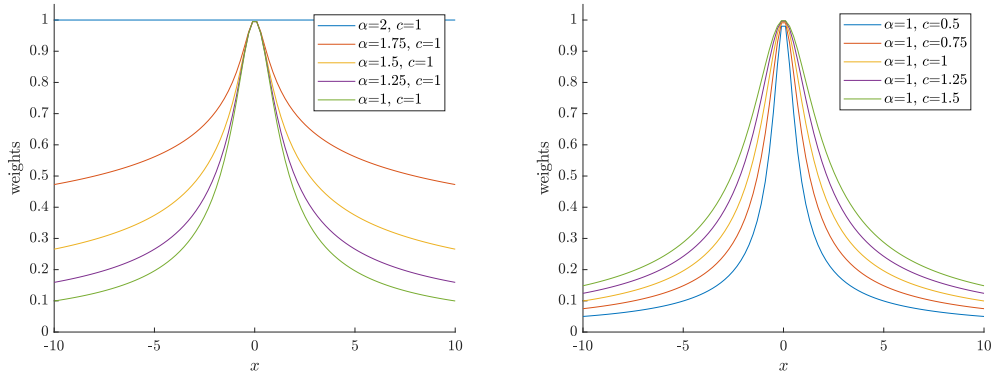


Figure 5.2.2: *Left: weights $\frac{\rho'(x,\alpha,c)}{x}$ for constant c changing α, Right: weights $\frac{\rho'(x,\alpha,c)}{x}$ for constant α changing c*

of large noise or outliers, α decreases and thus creates a heavier-tailed distribution, with probability mass moving from the smaller residuals towards the larger residuals. Changing α moves the probability mass mostly between large and low residuals with less change in the mid-range residuals. This is where the increased adaptivity of $\hat{P}$ over $P_\star$ can be understood. Figure (5.2.1 right) shows varying c with a constant α moves probability mass between smaller residuals and mid-range residuals

and minimal change in probability for larger magnitudes. Essentially, optimizing $\hat{P}$ gives an additional "degree of freedom" of better fitting the existing residuals by adapting c along with α. From a weight perspective, increasing and decreasing c results in a smoother and sharper drop in the weights respectively as residuals increase (Fig. 5.2.2). Thus, with the addition of changing c, the optimization explores more values in the weight space which helps the IRLS step. Note for $\alpha = 2$, changing c does not change the weights, they all remain 1. This is because the parameter c only affects the variance for the Gaussian distribution. Lower values of c help fit tighter residuals which is something RKO cannot do when c is fixed.

Now, given algorithm (4), another step can easily be added in this method, where, after minimizing the negative log-likelihood for α, the negative log-likelihood for c is minimized. This version can be called the Scale-variant Robust Kernel Optimization (SRKO) method. The steps of this algorithm are shown in the proposed Algorithm 5. It is very similar to the original RKO algorithm. It starts with initial guesses $\boldsymbol{\theta}^0$, α^0 and c^0. Then, for any time step t, the following steps are conducted: first, with the current value of c^{t-1}, and the residuals $x(\boldsymbol{\theta}^{t-1})$, find α^t that minimizes the negative log-likelihood of the residuals. This can be done easily with a grid search. Next, c^t is obtained similarly by minimizing the negative log-likelihood that is calculated with $x(\boldsymbol{\theta}^{t-1})$ and α^t. Note, ideally this search needs to be done over a 2D grid but is approximated here with learning best α and c separately for reducing the computational cost. Lastly, with α^t and c^t, the loss function in Eq. (5.3) can be optimized iteratively using Gauss-Newton method. In steps 1 and 2 of this algorithm, to search for optimum values of α and c they are discretized over their possible ranges.

A pre-computed table of values of $\hat{Z}(\alpha, c)$ is used for each of the grid searches.

Algorithm 5: Scale-variant Robust Kernel Optimization (SRKO)

Input: $\boldsymbol{\theta}^0$, α^0, c^0

Output: $\hat{\boldsymbol{\theta}}$, $\hat{\alpha}$, $\hat{c}$

while !converged **do**

% Minimize for α

$\alpha^t = \arg\min_{\alpha} - \sum_k \log \hat{P}\left(x_k\left(\boldsymbol{\theta}^{t-1}\right), \alpha, c^{t-1}\right)$;

% Minimize for c

$c^t = \arg\min_{c} - \sum_k \log \hat{P}\left(x_k\left(\boldsymbol{\theta}^{t-1}\right), \alpha^t, c\right)$;

% Minimize for $\boldsymbol{\theta}$ using IRLS

$\boldsymbol{\theta}^t = \arg\min_{\boldsymbol{\theta}} \sum_k \rho\left(x_k(\boldsymbol{\theta}), \alpha^t, c^t\right)$;

end

5.3 Decoupling Scale from Shape

In the presented SRKO algorithm, the approach is to estimate the scale(c) and shape(α) in a coupled manner. That is, SRKO is designed such that c gives a better estimate of the shape of the residual distribution but as the residuals are un-scaled, c also estimates the scale of the residual along with the shape. This is expected since c is the scale (i.e., inlier noise threshold) in equation (5.1). However, while it can offer increased performance, sometimes the coupling of scale and shape can result in incorrect estimates of either of these parameters as discussed in [23]. To combat this, it is possible to de-couple scale and shape by pre-computing scale. One method of

pre-computing scale is offered in [126], which uses the critical value of χ^2 distribution to set this value. Another method for pre-computing an estimate of the scale is using the formula offered in [74] as shown:

$$\hat{c} = \frac{1}{0.675}\text{Median}(x_k | x_k \neq 0). \tag{5.12}$$

In this formula, the residuals x are calculated with the L1 estimate. Even though L1 regression estimation is not straightforward to compute, one way to find an approximate solution is by optimizing equation (5.3) with $\alpha = 1$. Since L1 estimate does not need scale, $c = 1$. Putting $\alpha = 1$ and $c = 1$ in equation (5.1) results in

$$\rho(x) = ((x^2 + 1)^{1/2} - 1). \tag{5.13}$$

Therefore, one can first solve equation (5.3) for $\alpha = 1$ and $c = 1$ with IRLS and then estimate $\hat{c}$ with equation (5.12). Now, it is possible to obtain the scaled residuals $\hat{x} = \frac{x}{\hat{c}}$. SRKO can then be used on the scaled residuals to learn α and c, which help best fit the shape of the residual distribution. To differentiate between this version with the previous version of SRKO, SRKO with the pre-computed $\hat{c}$ is referred to as SRKO* in the analysis below.

5.4 Experimental Evaluation

The proposed algorithms are tested on point cloud registration with synthetic data and lidar odometry with real-world data sets and compared with other robust methods

from literature.

5.4.1 Point Cloud Registration

Given two sets of points P and Q, the objective is to solve for the transformation $\mathbf{T}$ such that the distance between matching correspondences of the two data sets (collected in set $\mathcal{K}$) is minimized. The optimization problem can be written as

$$E(\mathbf{T}) = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}} \rho(\|\mathbf{p} - \mathbf{T}\mathbf{q}\|). \tag{5.14}$$

The de-weighting direction is chosen here because of the effectiveness of robust cost functions for point cloud registration problems shown in several studies [11, 14, 23, 125, 133]. They are also easy to implement inside a non-linear least squares framework whereas the MC approaches are usually implemented as a pre-processing step before the non-linear least squares. Lastly, robust cost functions have been implemented in many state-of-the-art lidar SLAM packages like [15, 33, 83].

5.4.2 Synthetic Data

The proposed algorithms, SRKO and SRKO*, are tested for the problem of pairwise point cloud registration problem with the open source implementation and synthetic range data sets provided in [133]. The registration algorithm of [133], referred to in this chapter as FastReg, rewrites the scaled Geman-McClure estimator as an outlier process using Black-Rangarajan duality [17] and solves it iteratively. Since this cost function is non-convex, to avoid local minima, the method starts with a convex version

of this function and changes the scale parameter after every few iterations to increase the non-convexity. The readers are referred to [133] for details about these data sets which come from AIM@SHAPE repository, the Berkeley Angel dataset, and the Stanford 3D Scanning repository. For these data sets, performances of fixed Huber, RKO, SRKO, SRKO*, and FastReg are compared. Results that are obtained with a pre-computed estimate of $\hat{c}$ are labeled as SRKO*-const and the results with L1 scale estimate are labeled as SRKO*-L1. Similar to [133], two versions of point clouds are used. One is a clean version with no noise and the other is a noisy version with added Gaussian noise of $\sigma = 0.005$. Target point clouds are generated with truth transformation for proper evaluation. Both source and target point clouds have been normalized to the diameter of their surface. Correspondences between the source and target point clouds are obtained by matching Fast Point Feature Histogram (FPFH) features [95]. The registration is done by minimizing the point-to-point distance between correspondences using the Gauss-Newton method. Note the only difference between the methods that are being compared is the way the residuals are weighted in IRLS. For completeness, the Huber and Geman McClure weight formulas are shown here :

$$w_{huber}(x, c) = \begin{cases} 1 & |x| \leq c \\ \frac{c}{|x|} & |x| > c \end{cases} \tag{5.15}$$

$$w_{geman}(x, \mu) = \frac{\mu^2}{(\mu + x^2)^2} \tag{5.16}$$

5.4.3 Real World Data

Huber, RKO, SRKO, SRKO*, and FastReg are also tested with the lidar-inertial SLAM package LIO-SAM [98]. Due to the lack of GPS data, for evaluation, performances of the above four methods without loop-closures are compared to loop-closure-assisted standard LIO-SAM as a reference. The open-source implementation provided by the authors of [98] uses a weight function of the form $1 - 0.9|x|$ where x is the distance between an edge feature and its corresponding edge in the map or the distance between a planar feature to its corresponding plane along the plane normal in the map. In [98], the authors also remove residuals that are larger than a certain residual threshold. In our implementations of the four methods, all residuals are allowed and it is left to the robust methods to de-weight suspected outliers. These methods are evaluated on the *park*, *garden*, *rotation* and *campus* data sets provided in [98].

Note, unlike the synthetic data tests, SRKO* is tested with pre-set $\hat{c} = 0.1$ and not estimated with equation (5.12). There are multiple reasons behind this decision. First, due to the real-time performance needs of LIO-SAM, it is desirable to avoid any extra online computation. Secondly, the estimates from equation (5.12) were found to be sensitive to gross outliers leading to larger estimates of $\hat{c}$, thus affecting performance. Lastly, $\hat{c}$ being the inlier noise threshold should be expected to be a constant value irrespective of the presence of noise or gross outliers. Thus estimating $\hat{c}$ comes with the possibility of violating this constraint.

5.4.4 Results and Discussion

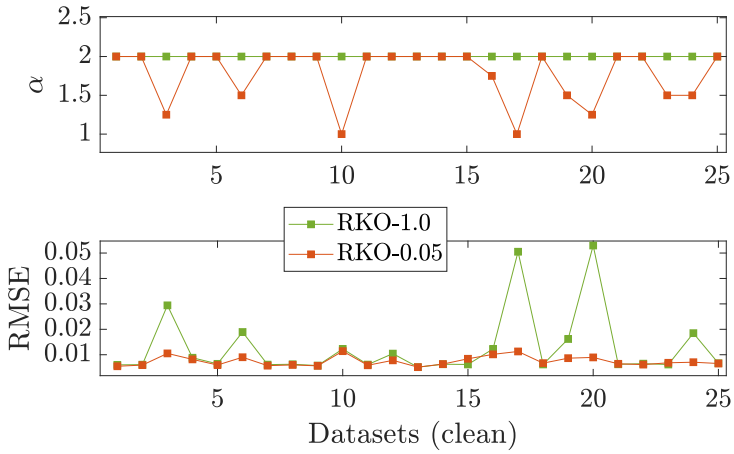


Figure 5.4.1: *Effect of residual scaling in RKO*

In this part of the analysis, results for the synthetic data sets are discussed. For algorithm implementations of RKO, SRKO, and SRKO*, $\hat{Z}(\alpha, c)$ is calculated with τ values set to 10. This parameter signifies the range of residual values that are used to learn the parameters. For Huber kernel, the scale parameter value is set to 1.3 which is a common choice [50]. For RKO, c is fixed to 1 and α has a discretized range of $[-4 : 0.25 : 2]$. The residual used here is the point-to-point distance between correspondences. The initialization points for RKO,SRKO, and SRKO* are $(\alpha, c) = (2, 1)$, which is the standard Gaussian distribution.

When tested with 25 clean and noisy data sets, the performances of Huber and RKO are worse than FastReg. Next, the residuals are scaled by s before learning. That is, instead of x, the distribution of $\frac{x}{s}$ is learned. The scale values tested are 0.1 and 0.05. Lowering the scales resulted in $\approx$ 2x improvement in performance for both

clean and noisy data (Tables 5.4.1, 5.4.2).

The motivation for residual scaling can be understood from figure (5.4.1). The top plot shows the learned α and the bottom one shows the registration performances for scales 1 and 0.05. Scale 0.05 case improves performance in data sets for which learned α lower than 2. This points to the fact that scale 1 is under-fitting the residuals with $\alpha = 2$ resulting in equal weights for all residuals. For scale 0.05, RKO learns a more robust α which de-weights the larger residuals. This happens because the un-scaled residual values for these data sets lie close to 0 (figure (5.4.2)). This results in RKO weighing them equally. However, just being close to zero does not guarantee that the residual is an inlier since the true scale is unknown. Scaling with 0.05 increases the residual by a factor of 20 which helps RKO find a better fit with a more robust α and de-weights the larger residuals. The same thing happens for the Huber kernel where all un-scaled residuals being less than 1.3 results in equal weighting. Scaling increases the residual magnitude causing the de-weighting of larger residuals.

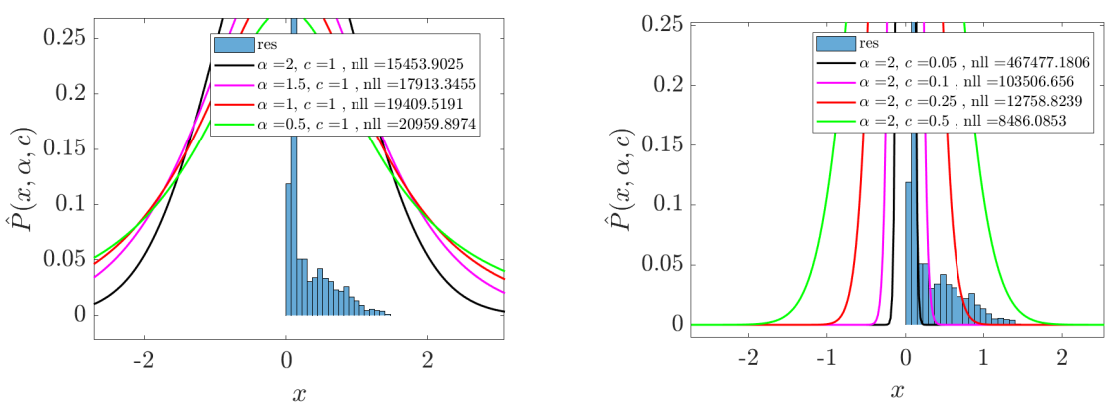


Figure 5.4.2: *Learning best fitting α and c for SRKO with un-scaled residuals*

Instead of manually tuning the residual scale, one way to learn this scale directly is to let SRKO look for the c values. Notice that changing the scale s is the same as changing the scale c in equation (5.1). In the implementation, SRKO can search through c values within the range $[0.05 : 0.05 : 2]$. The initialization point is kept the same at $(\alpha, c) = (2, 1)$. Figure (5.4.2) shows SRKO choosing the best α and c with starting $c = 1$ and a real set of residuals extracted during registration. As illustrated, in this scenario, the negative log-likelihood decreases as a good fit for the residuals is found. This step is followed by optimization with the best α and c. This learning-optimization cycle carries on until convergence. Figure (5.4.2) also shows how SRKO ends up learning the scale of un-scaled residuals even though it is designed to better fit the shape.

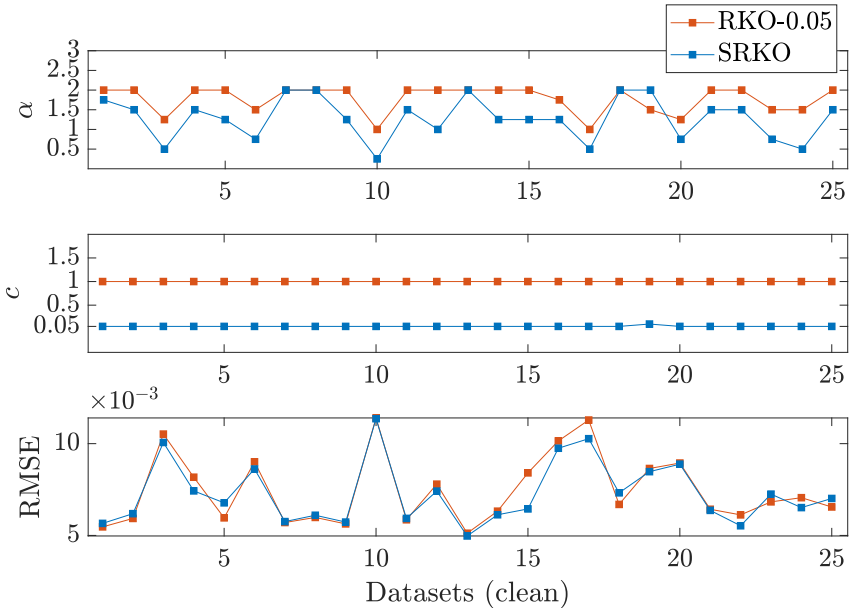


Figure 5.4.3: *Clean data results. Top: Learned α values, Middle: Learned c values, Bottom: RMS errors normalized*

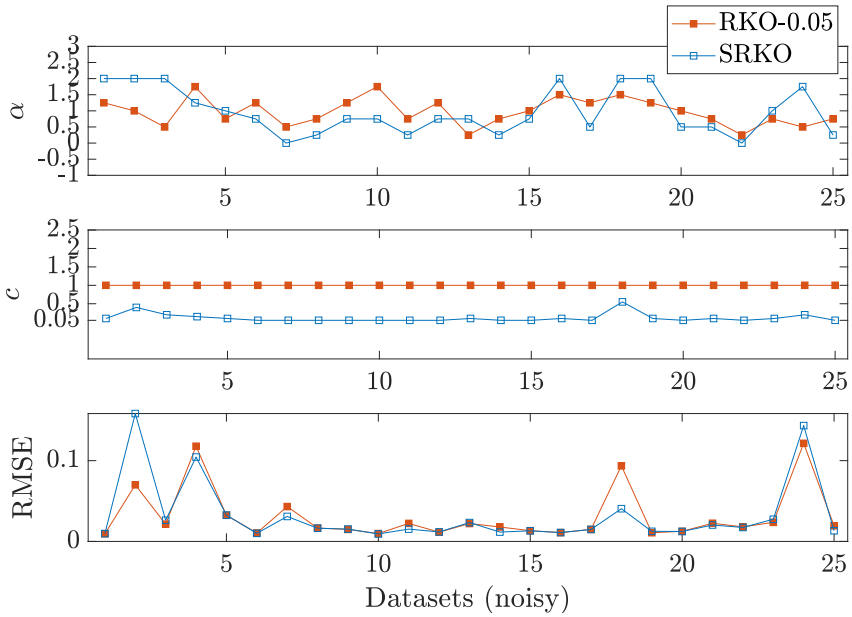


Figure 5.4.4: *Noisy data results. Top: Learned α values, Middle: Learned c values, Bottom: RMS errors normalized*

Figures (5.4.3) and (5.4.4) further show the parameters learned and the registration performance of RKO vs SRKO for clean and noisy data. These figures show that SRKO can find low scale values close to 0.05 and give similar performance to scaled RKO thus removing the need for manual tuning. Figure (5.4.4) shows the effects of noise on SRKO where it is not able to learn correct c and robust α values leading to larger errors for some data sets. This effect motivates the need for SRKO*.

Next, the performances of SRKO and SRKO* are compared. Figure (5.4.5) shows the learned parameters and performances of SRKO*-L1 and SRKO*-const for clean data. The bottom graph shows the $\hat{c}$ estimated by SRKO*-L1 which on average is close to the $\hat{c} = 0.05$ assumed by SRKO*-const. It can be seen that SRKO*-L1 and SRKO*-const learns a more robust α than SRKO. Another difference between SRKO

and SRKO* is the learned c parameter. Since SRKO* used scaled residuals, it can use c as a shape parameter. Whereas, SRKO is mainly learning the scale and not able to utilize it for learning shape. Figure (5.4.6) shows the performance of SRKO* for noisy data. SRKO*-const and SRKO*-L1 again are able to learn more robust α values than SRKO resulting in better performance. One disadvantage of using the scale formula in equation (5.12) can be seen in this figure. The estimates of $\hat{c}$ increases for the noisy data which results in large errors for SRKO*-L1 for some data sets. For this reason, SRKO*-const is preferable to SRKO*-L1.

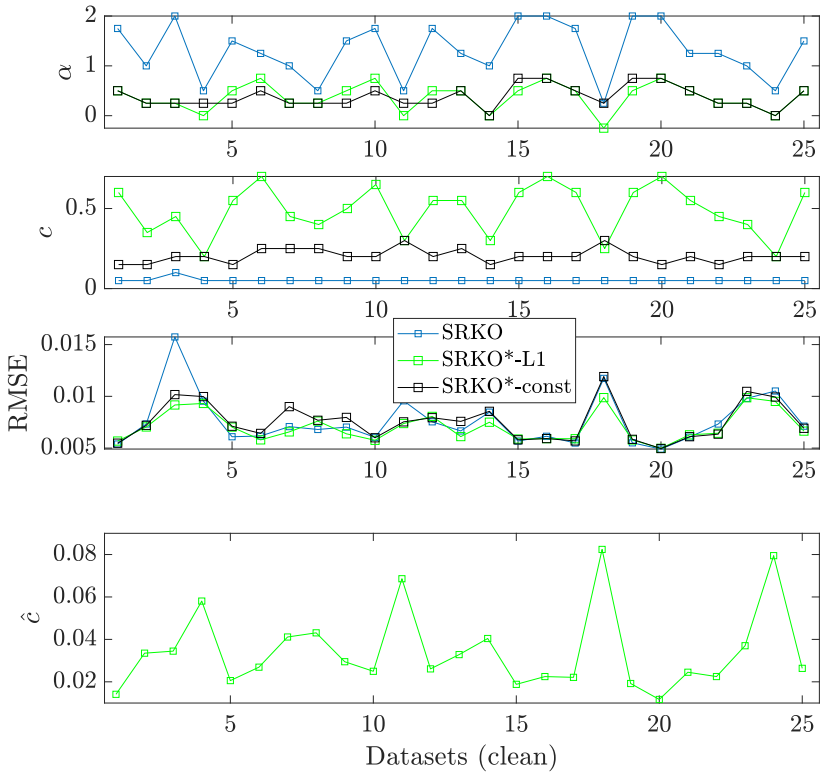


Figure 5.4.5: *Clean data results. From Top 1: Learned α values, From Top 2: Learned c values, From Top 3: RMS errors normalized, Bottom: $\hat{c}$ learned by SRKO*-L1*

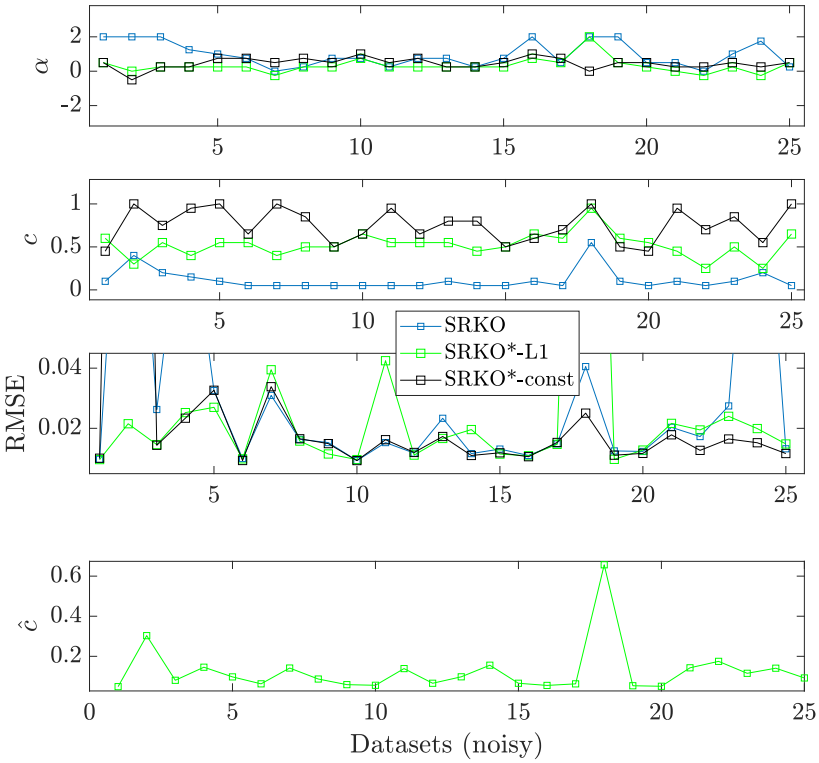
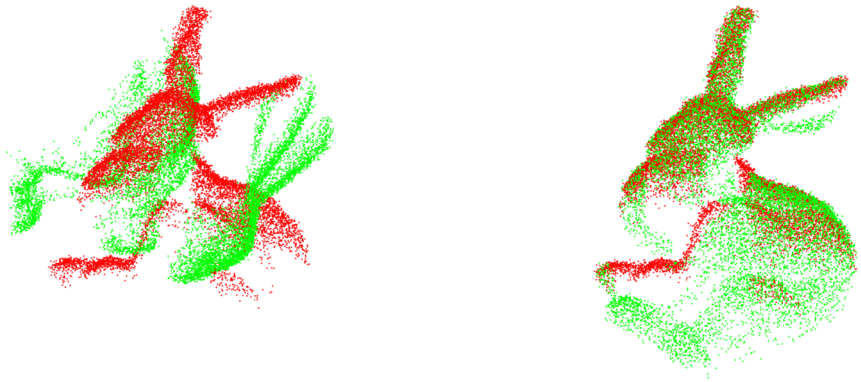


Figure 5.4.6: *Noisy data results. From Top 1: Learned α values, From Top 2: Learned c values, From Top 3: RMS errors normalized, Bottom: $\hat{c}$ learned by SRKO*-L1*

Robustness of SRKO* is tested with increasing number of correspondence mismatches (10% to 50% of the total correspondences). In Figure (5.4.9), RKO and SRKO* can be seen to be generally more robust than FastReg, but affected by convergence issues with an increase in outlier proportion. This problem is solved by increasing τ, that is the parameters are learned for a larger range of residuals (Figure (5.4.10)) and demonstrate superior robustness properties compared to FastReg. Tables (5.4.1) and (5.4.2) show the average RMSE performances for Huber, RKO, FastReg, SRKO, SRKO*-const and SRKO*-L1. SRKO*-L1 performs best for clean



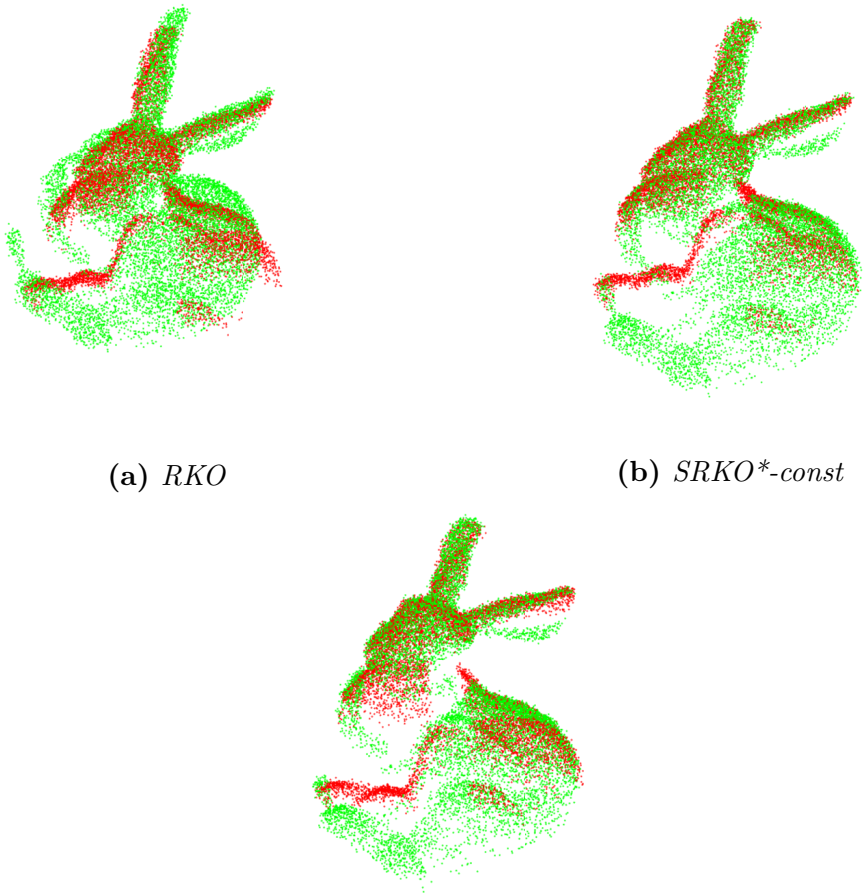
(a) *Input scans* **(b)** *Truth registration*

Figure 5.4.7: *(a): Noisy versions of target (red) and source (green) point clouds of Stanford Bunny; (b): Truth transformation of source point cloud to target point cloud*

data and FastReg performs best for noisy data. Barron [14] solves equation (5.3) in a similar way to FastReg but anneals the shape instead of scale (shape-annealed gFGR and gFGR*). SRKO*-L1 performs better than these methods for clean data but is worse for noisy data. Even though the RMS error of SRKO*-const is more than SRKO*-L1 in table (5.4.2), the larger error is mainly due to one bad estimate out of 25. Figure (5.4.6) shows that SRKO*-const has a better performance than SRKO*-L1 for the majority of the datasets for the noisy case.

Table 5.4.1: *Average RMS errors over 25 clean data*

methods	Huber	RKO	FastReg	SRKO	SRKO*-const	SRKO*-L1
$scale - 1$	0.0123	0.0129				
$scale - 0.1$	0.0080	0.0094	0.0074	0.0073	0.0075	0.0071
$scale - 0.05$	0.0074	0.0074				



(a) *RKO*

(b) *SRKO*-const*

(c) *SRKO*-L1*

Figure 5.4.8: *Registered noisy point clouds with (a) RKO (RMS error: 0.12); (b) SRKO*-const (RMS error: 0.015); (c) SRKO*-L1 (RMS error: 0.019)*

For LIO-SAM implementations of RKO, the searchable α range is $[-4 : 0.5 : 2]$. c is fixed to 1.0. For SRKO and SRKO*, α range is same and c range is $[0.05, 0.1, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0]$. $\hat{c}$ is set to 0.1. Initialization points for both are set to $(\alpha, c) = (2, 1)$. Here smaller ranges of α and c are chosen due to the real-time running of LIO-SAM. FastReg is initialized with $\mu = 20$ and the Huber pa-

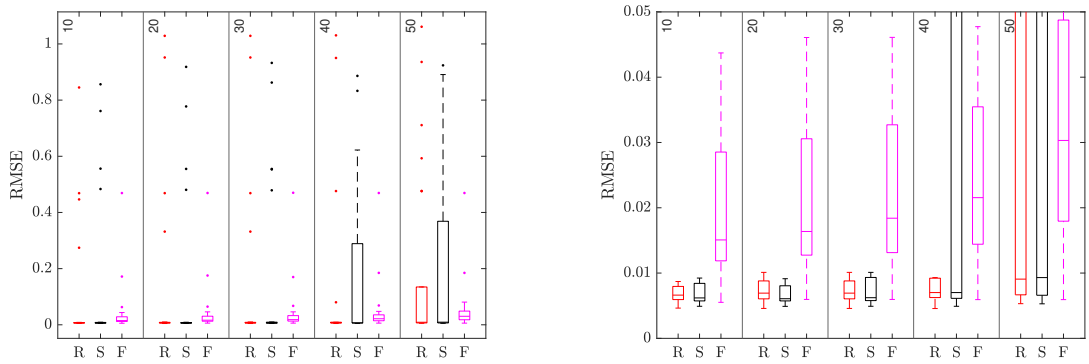


Figure 5.4.9: *Top: Registration performance of RKO (R), SRKO*-const (S) and FastReg (F) over 25 data sets with increasing outlier ratios (10% to 50%). Bottom: Magnified version of Top*

Table 5.4.2: *Average RMS errors over 25 noisy data*

methods	Huber	RKO	FastReg	SRKO	SRKO*-const	SRKO*-L1
$scale - 1$	0.0532	0.0520				
$scale - 0.1$	0.0287	0.0241	0.0203	0.0320	0.0352	0.0291
$scale - 0.05$	0.0240	0.0312				

rameter is the same as the synthetic version. Table (5.4.3) shows the horizontal RMS errors of all methods with respect to standard LIO-SAM with loop closures. SRKO* performs best for *park*, *garden*, and *campus*(*large*) data sets and also performs well for the other tests. Figure (5.4.12) shows the learned parameters of RKO, SRKO, and SRKO* for *park* data. The trend is similar to figure (5.4.5). SRKO* learns more robust α values than SRKO and is able to use c as a shape parameter due to residuals being scaled. Whereas, SRKO mainly learns the scale due to the residuals being un-scaled.

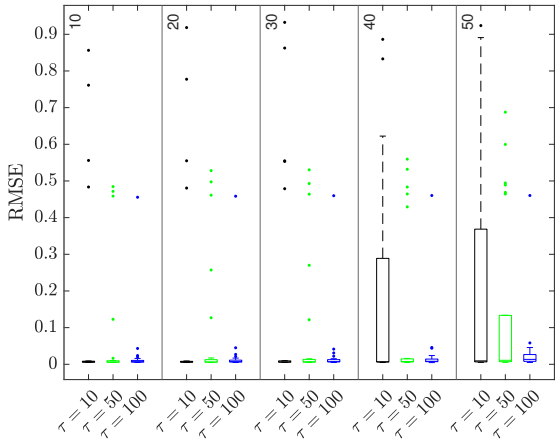
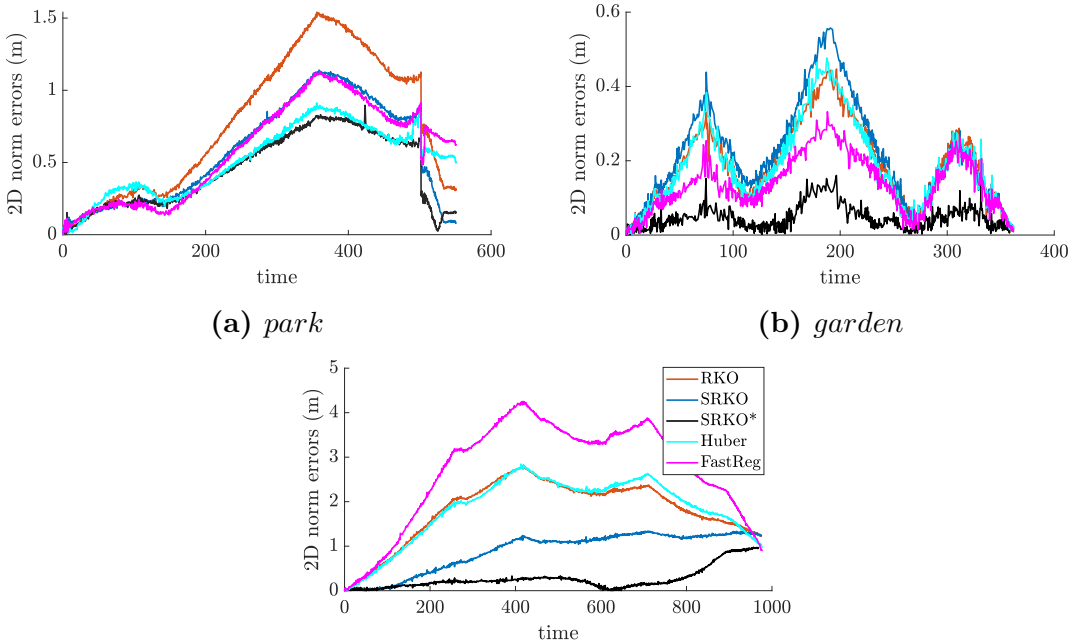


Figure 5.4.10: *SRKO* robustness with respect to τ*

Table 5.4.3: *Horizontal RMS errors (m) w.r.t LIO-SAM+LC*

methods	Huber	FastReg	RKO	SRKO	SRKO*
park	0.58	0.70	0.93	0.68	0.52
garden	0.19	0.14	0.18	0.22	0.04
rotation	0.05	0.04	0.06	0.06	0.06
campus(small)	0.17	0.12	0.24	0.23	0.16
campus(large)	2.09	2.95	2.02	1.29	0.83

The scale-variant algorithm has been shown to work well in both synthetic and real-world scenarios and has good robustness properties. The analysis here also shows that learning-based methods can perform as well and in some cases better than graduated non-convexity-based methods. The main disadvantage of this method is the computational cost of learning c and α for a large set of residuals which can affect real-time performance. In such scenarios, a GNC-based method is more helpful. GNC-based methods also help avoid local minima, which is an advantage. The generalized cost function analyzed here also satisfies the convergence requirements discussed in section 1.6.2. The weights are well defined everywhere and $\rho(\sqrt{(x)})$ is concave for



(a) *park* **(b)** *garden*

(c) *campus(large)*

Figure 5.4.11: *2D norm errors(m) for 3 data sets provided by [98]*

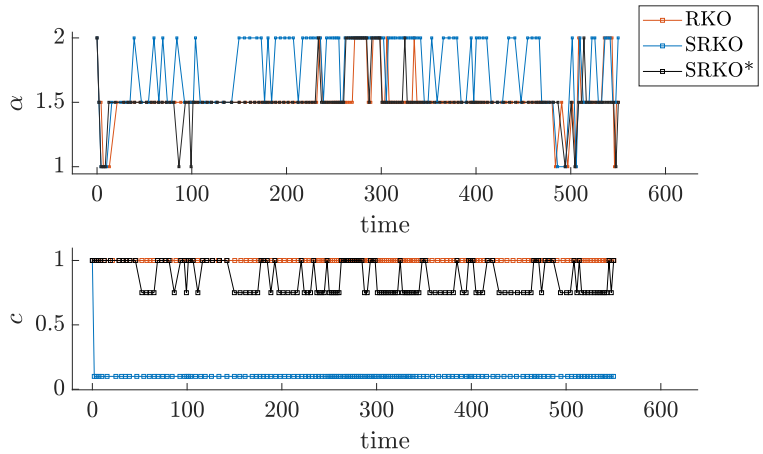


Figure 5.4.12: *Learned α and c values for park data*

$x > 0$ and $\alpha < 2$. Singularities that exist at $\alpha = 0$ and $\alpha = 2$ can be avoided by using numerical stable versions of the functions ([14] supplementary). Another way

of avoiding computational cost is to add c and α in the estimated state of the non-linear least squares by computing the augmented Jacobian matrix. Another natural extension of this work is robust loop closure detection. Ramezani et al. [90] uses α to define non-Gaussian loop closure factors in a factor graph and estimates lidar poses and α. This method can be extended to include c as a variable in the factor graph.

5.5 Estimating Shape with Square-root Factors

Ramezani et al. [90] describe using square root factors for robust loop closures using the Black-Rangarajan duality and learn the shape parameter at the same time. Recall from Chapter 4, that this duality shows equivalence between an M-estimator and regularized weighted least squares,

$$\hat{\boldsymbol{\theta}}, \hat{\mathbf{w}} = \arg\min_{\boldsymbol{\theta}, \mathbf{w}} \sum_k w_k x(\boldsymbol{\theta})^2 + \Psi(w_k).$$

The robust cost function used here has the general form described in equation (5.1). Following the previous discussion, the weights of the general cost function can be found easily by using the expression $\frac{\rho'(x)}{x}$. Following the derivation method in [17], the Ψ penalty function can also be derived as

$$\Psi(w, \alpha) = \begin{cases} -\log(w) + w - 1 & \text{if } \alpha = 0 \\ w\log(w) - w + 1 & \text{if } \alpha = -\infty \\ \frac{|\alpha-2|}{\alpha}\left(\left(1 - \frac{\alpha}{2}\right) w^{\frac{\alpha}{(\alpha-2)}} + \frac{\alpha w}{2} - 1\right) & \text{if } \alpha < 2. \end{cases} \tag{5.17}$$

Since the cost ρ is a function of α, both the weights and the Ψ are functions of α as well. This fact was utilized in [90] to create robust loop closure factors in SLAM. In their work, the authors assumed a single α parameter node for all loop closure factors and learned the parameter which controls the robustness in the graph. Ψ can be added as a factor in the graph as a square root factor as described in Chapter 4.

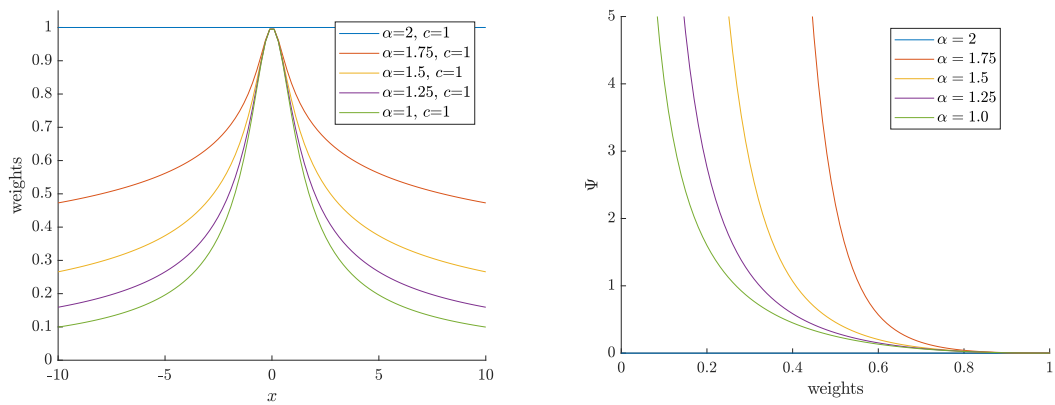


Figure 5.5.1: *weights as a function of residuals (left); Ψ as a function of weights(right)*

The figure above shows the weights as a function of residuals, which has been discussed before. The figure on the right shows the Ψ as a function of weights. The Ψ function acts as a penalty function which prevents the weights from going to zero. From a Bayesian perspective, it can also be thought of as a prior to the weights. Since the weights are functions of the residuals (hence the state), the optimization is actually solving for $\boldsymbol{\theta}$ and α which can be written as

$$\hat{\boldsymbol{\theta}}, \hat{\alpha} = \arg\min_{\boldsymbol{\theta},\alpha} \sum_k w_k(x_k(\boldsymbol{\theta}), \alpha) x_k(\boldsymbol{\theta})^2 + \Psi(x_k(\boldsymbol{\theta}), \alpha). \tag{5.18}$$

Figure (5.5.2) (left) shows the variation of Ψ with respect to α for fixed values of x. It

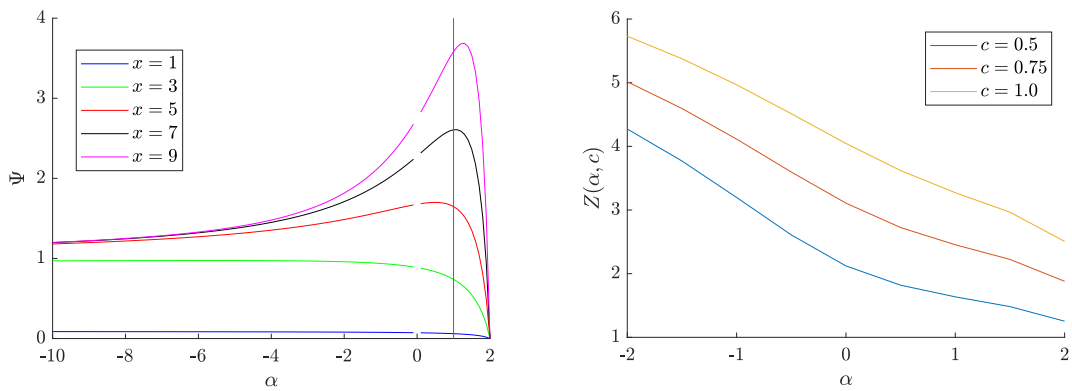


Figure 5.5.2: Ψ *as a function of* α*(left),* $Z(\alpha)$ *as a function of* α *(right)*

is interesting to see that behavior is different for large and small residuals. For small residuals, Ψ increases initially for α close to 2, but levels off for lower values of α. For larger residuals. Ψ also increases for α close to 2 but starts decreasing around 1 and finally levels off for lower shape values. On the right, the variation of $Z(\alpha)$ (equation 5.11) with α is shown. $Z(\alpha)$ acts as a regularizer or penalty function which prevents all the weights from going to zero. Comparing these figures, it can be seen that for larger residuals Ψ will only stop α from going to $-\infty$ between the values of 2 and 1 (approximate), since it is a decreasing function in that range. As α decreases from 1, the penalty function starts decreasing, which means there is nothing stopping the optimization from taking the α value to $-\infty$, which is the Welsch loss function. For smaller residuals, Ψ levels off for lower values of α again not satisfying the property of a penalty function like $Z(\alpha)$. Thus it can be concluded that this method is not really solving for the shape for highly robust α values since it will always tend towards $-\infty$.

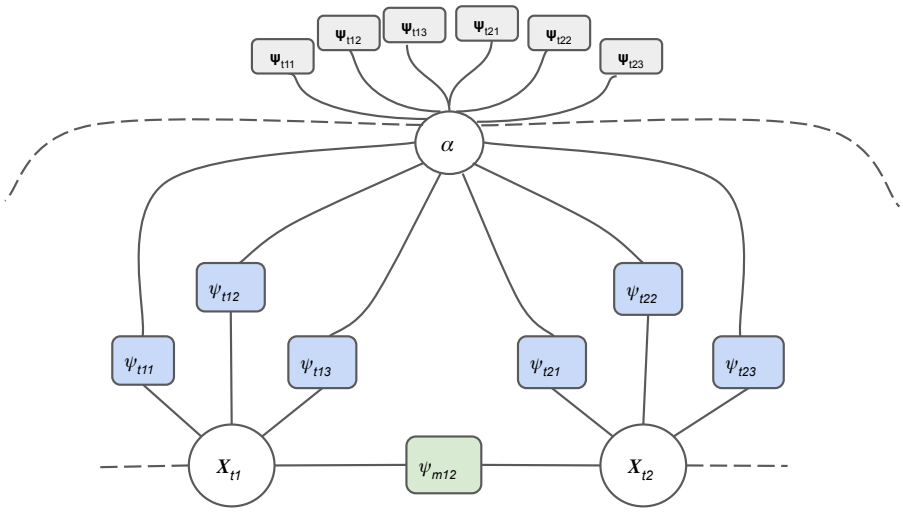


Figure 5.5.3: *GNSS factor graph with a shape parameter (α) node. α controls the robustness of the factor graph. Each satellite measurement factor (blue) has a weight that is controlled by a penalty factor (grey).*

To demonstrate this in a practical scenario, this algorithm is implemented for factor graph based GNSS positioning problem. A single shape parameter α is chosen for all satellite measurements over the whole data trajectory. 3 open-source data sets obtained from [117] are used which have been collected from both open sky and urban areas of Morgantown, WV. These data sets have 2 quality levels (low and high) obtained offline by changing tracking parameters using open-source software [39]. A detailed analysis of the data collection is described in [117].

Figures (5.5.4) and (5.5.5) show the horizontal RMS errors of the square root factor graphs for different prior variances of α compared to standard least squares results for data set 1. The prior and the initial values of the shape parameter are

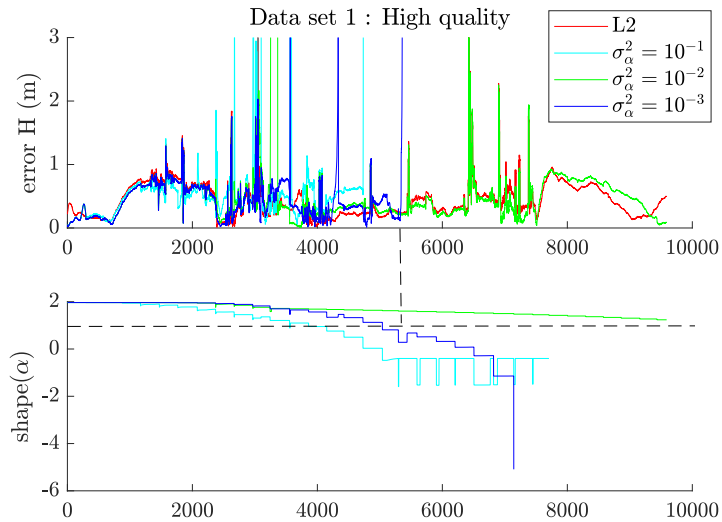


Figure 5.5.4: *Horizontal RMS errors (top) and estimated α (bottom) for different prior variances of α.*

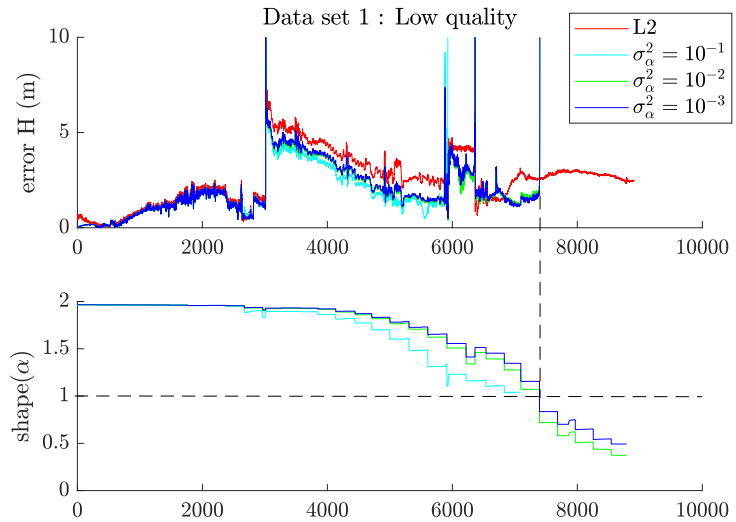


Figure 5.5.5: *Horizontal RMS errors (top) and estimated α (bottom) for different prior variances of α.*

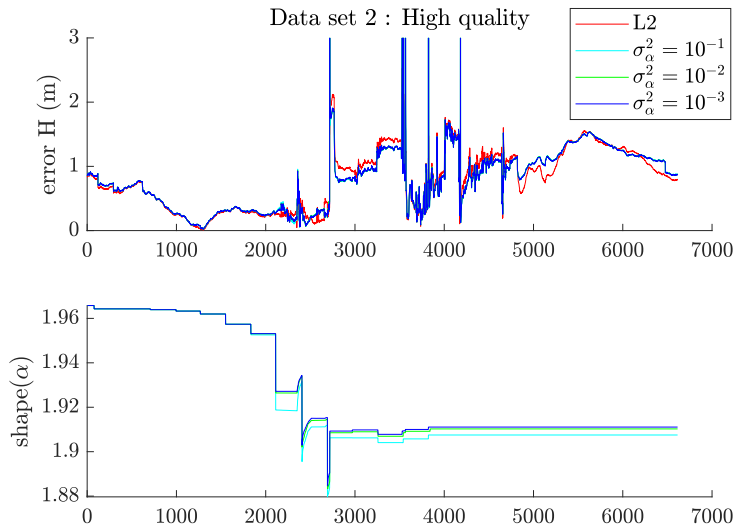


Figure 5.5.6: *Horizontal RMS errors (top) and estimated α (bottom) for different prior variances of α.*

set to 2. For the high-quality case, the performances of all methods initially are similar but the solution for the square-root factor graph starts getting unstable as α starts getting close to 1. The dotted lines show that the solutions for prior variances of 10^{-1} and 10^{-3} diverge when the shape parameter drops below 1 which confirms our prediction from the gradients of Ψ. The same behavior can be seen for the low-quality version of data set 1 even though the square-root factor graphs have better performance than the standard factor graph before divergence.

For Figures (5.5.6), (5.5.7), (5.5.8), and (5.5.9), square-root factor graphs outperform the L2 version for low-quality data and have similar performance for high-quality data. Note, that there is also no divergence in the solutions due to α not going under the value of 1. For low-quality data sets, the shape parameter solution also moves

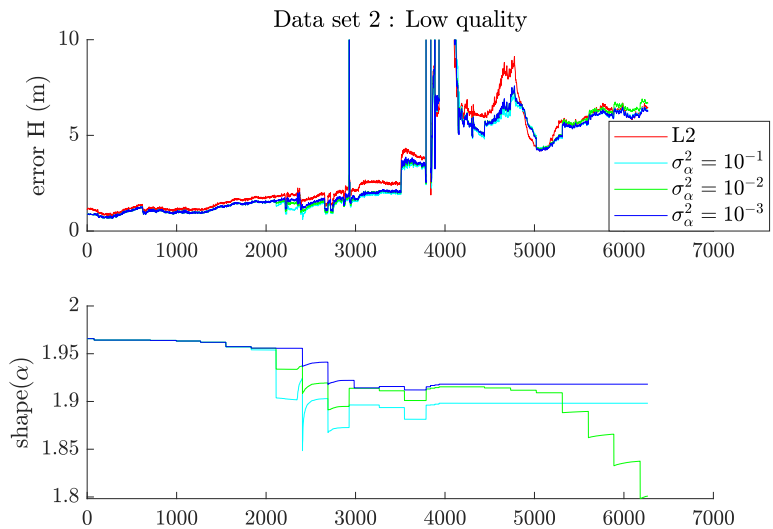


Figure 5.5.7: *Horizontal RMS errors (top) and estimated α (bottom) for different prior variances of α.*

away from its initial value of 2 when the prior uncertainty is the largest, which is expected.

The trend of shape parameters in all results seems to be going down as new measurements are added. This is surprisingly similar to GNC-based approaches where the parameter that controls the convexity of the loss function is gradually lowered every few iterations. Here α is the parameter that controls the convexity of the function ρ. Another way of thinking about the behavior of α is that since there is a single shape parameter for all factors of a graph, as more measurements are added, the number of large residuals keeps increasing requiring the need for lower values of α. From our analysis, this method is not suitable for highly robust cost functions due to the diverging properties discussed above. But this method still can be helpful

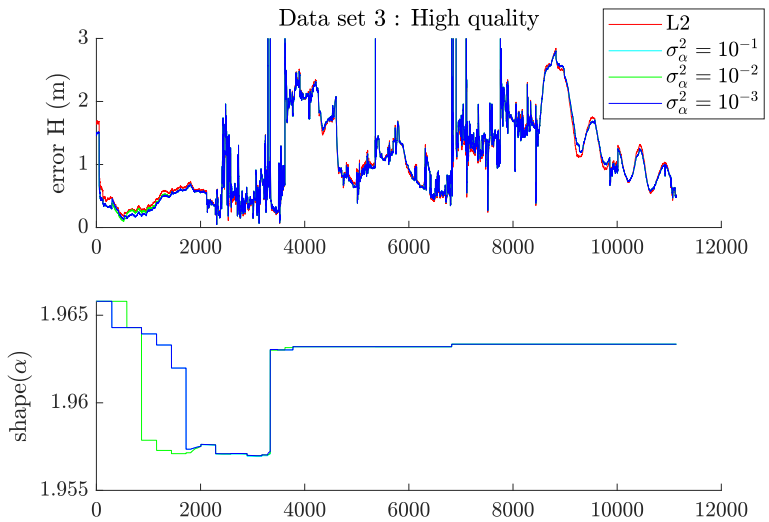


Figure 5.5.8: *Horizontal RMS errors (top) and estimated α (bottom) for different prior variances of α.*

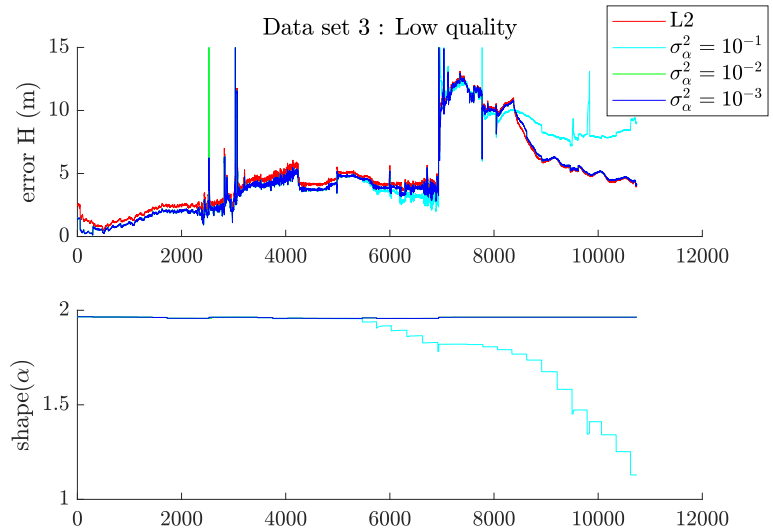


Figure 5.5.9: *Horizontal RMS errors (top) and estimated α (bottom) for different prior variances of α.*

and needs more analysis. Different penalty functions can be investigated in this setup along with multiple shape nodes instead of one.

6

Concluding Remarks

6.1 Conclusions

The first part of the work gives a birds-eye view of different estimation tools with different sensors, robustness in estimation, and new ways of developing robust algorithms. Chapter 2 presents a concise introduction to Bayesian estimation, linear and

non-linear least squares, M-estimators, and MC problems. The rest of the dissertation contains work with two different estimation tools, the Kalman filter and factor graphs. The robust techniques presented here are all residual-based, which helps in their applications to a wide variety of estimation problems.

Chapter 3 demonstrates the importance of robust Kalman filtering for wheel-inertial odometry in high slip conditions. Five methods have been chosen from robust Kalman filtering literature and applied as robust update steps in an error state EKF to offer insight into their benefits and drawbacks. These five methods include the Huber Kalman filter, covariance scaling filter, and variational filters. The variational Kalman filters perform the best in reducing the effect of erroneous wheel encoder measurements. Tables (3.2.2) and (3.2.3) demonstrate the localization improvement using these methods. Tables (3.2.5) and (3.2.4) show parameter analysis that helps in understanding the nature of measurement noise distribution for wheel odometry data. In the datasets used in this work, the distribution was found to be closer to Gaussian.

Factor graphs have been discussed as an estimation tool in Chapter 4 along with its advantages over the Kalman filter. Two different GNSS and inertial odometry fusion strategies are tested for clean and noisy data. In one version, motion constraints are added to the GNSS factor graph from a parallel running wheel-inertial Kalman filter. The other version does not use inertial odometry directly but uses the information that the rover is static to constrain static nodes. Testing is done with original and noise-added GNSS data. The positioning errors in tables (4.5.1) and (4.5.2) show that zero-velocity enabled EKF inertial odometry can make GNSS factor graphs more

robust to multipath errors and have superior positioning performance compared to GNSS factor graphs without inertial information.

Chapter 5 discusses a novel way of improving the adaptivity and robustness of robust cost functions and learning the inlier measurement noise threshold in a combined manner. The proposed algorithm is tested on point cloud registration and lidar-inertial odometry. The learned shape and scale parameters (figures (5.4.3), (5.4.4), (5.4.5), (5.4.6), and (5.4.12)) and the estimation performances (tables (5.4.1), (5.4.2), and (5.4.3)) show that the new algorithm can be used in learning the inlier noise threshold and increasing the robustness of generalized M-estimators by more accurately learning the residual distribution. The proposed algorithm also shows improved robustness in the presence of feature mismatches attained by this adaptivity when compared to a graduated non-convexity-based method from literature (figure (5.4.9)). Increasing the residual range used to learn the residual distributions in the algorithm produces more robust registration results for large outlier proportions (figure (5.4.10)). Lastly, an analysis is presented on the adaptive robust cost function optimization for GNSS factor graphs in figures (5.5.4), (5.5.5), (5.5.6), (5.5.7), (5.5.8), and (5.5.9), which show that joint optimization of the shape of the generalized M-estimators and unknown states in factor graphs can provide increased robustness than a standard GNSS factor graph but can also result in divergence when the shape value goes below 1.

6.2 Future work

Several future directions of robust estimation research look promising. One of them is data-driven learning algorithms. Deep learning methods [19, 24, 68] have been shown to work well in learning measurement noise distributions as well as full state transformations using a series of measurements. However, these learning methods use GNSS or motion-capture solutions as truth for optimizing their networks which might not always be available. Variational methods discussed in [3, 4, 96] can be helpful in such cases. The lack of physical modeling in deep learning methods can be concerning but from a different perspective, they are more expressive and are not constrained by specific model assumptions.

The importance of non-Gaussian estimation has been discussed in this dissertation. Even though analysis of heavy-tailed distributions using variational Kalman filters, square-root factors, and IRLS have been presented here, these algorithms are not suitable for multimodal distributions. Recently, multimodal frameworks have been adopted to improve robustness of Kalman filters [48] and factor graphs [35, 84, 118]. M-estimators only assume one mode to be the correct and de-weighting all others by learning an unimodal distribution. Multimodal M-estimators can be helpful where the estimator needs to learn a multimodal distribution. There is another aspect of estimation that is overlooked a lot of times, and that is non-convexity. Estimation problems are inherently optimization problems, which give rise to the problem of non-convexity. Recent developments in the field of non-convex optimization [53] can immensely help the estimation field of research. Ideas from this field can provide rigorous convergence

analysis for SRKO since it is an example of an alternating minimization algorithm. This family of algorithms is helpful for non-convex problems [53]. Robust statistics is also another field of study that can provide invaluable insights into applied robust estimation [21]. Recent research has focused on the certifiable optimality of robust estimation algorithms. Such algorithms like [124, 126] have been recently developed and shown to be resilient to high outlier rates and high dimensional problems. Theoretical foundations for consistent estimation using truncated least squares and MC for different outlier rates have been discussed from the perspective of robust statistics and robotics in [21].

However, these algorithms running on a single sensor should not be relied on solely for reliable localization in all scenarios for long-term trajectories. This is partly because of the physical limitations of the sensor used and partly due to distributional assumptions that might not be followed by the actual measurements and environmental variations. These algorithms running on each sensor should act as building blocks for cooperative multimodal sensor fusion architecture involving different sensor types like visual, lidar, inertial, and thermal. GNSS is the most accurate sensor of them all but cannot be used indoors, in caves, or in densely forested areas. Tightly coupled systems usually have high accuracy but it can be hard to switch between sensors since they are connected to a single estimation machinery. In loosely coupled systems, each sensor has its own estimation branch which makes it easier to switch one for the other depending on environmental conditions. Zhao et al. [132] discuss the importance of an IMU-centric lidar-visual-inertial system where IMU is the core estimation source since it is not affected by environmental conditions, unlike lidars and cameras. IMU

can provide reliable estimation as long as the biases are correctly tracked. One important problem is to decide how much each sensor's estimation should contribute to the overall estimation. Failure modes are usually easy to detect for sensors like cameras and lidars. However in working conditions, accurately estimating the measurement noise is not always straightforward. A tunable adaptive multi-sensor fusion framework can get rid of the individual shortcomings of each sensor.

References

[1] A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[2] Khurrum Aftab and Richard Hartley. Convergence of iteratively re-weighted least squares to robust m-estimators. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 480–487. IEEE, 2015.

[3] Gabriel Agamennoni, Juan I Nieto, and Eduardo M Nebot. An outlier-robust kalman filter. In *2011 IEEE International Conference on Robotics and Automation*, pages 1551–1558. IEEE, 2011.

[4] Gabriel Agamennoni, Juan I Nieto, and Eduardo M Nebot. Approximate inference in state-space models with heavy-tailed noise. *IEEE Transactions on Signal Processing*, 60(10):5024–5037, 2012.

[5] Gabriel Agamennoni, Paul Furgale, and Roland Siegwart. Self-tuning m-estimators. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635. IEEE, 2015.

[6] Pratik Agarwal, Gian Diego Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard. Robust map optimization using dynamic covariance scaling. In *2013 IEEE International Conference on Robotics and Automation*, pages 62–69. Ieee, 2013.

[7] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[8] Shahrokh Akhlaghi, Ning Zhou, and Zhenyu Huang. Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation. In *2017 IEEE power & energy society general meeting*, pages 1–5. IEEE, 2017.

[9] *ADIS16495 Data Sheet*. Analog Devices, 2017. Rev. A.

[10] Pasquale Antonante, Vasileios Tzoumas, Heng Yang, and Luca Carlone. Outlier-robust estimation: Hardness, minimally tuned algorithms, and applications. *IEEE Transactions on Robotics*, 2021.

[11] Philippe Babin, Philippe Giguere, and François Pomerleau. Analysis of robust functions for registration algorithms. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1451–1457. IEEE, 2019.

[12] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, 2017.

[13] Axel Barrau and Silvere Bonnabel. Invariant kalman filtering. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:237–257, 2018.

[14] Jonathan T Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4331–4339, 2019.

[15] Jens Behley and Cyrill Stachniss. Efficient surfel-based slam using 3d laser range data in urban environments. In *Robotics: Science and Systems*, volume 2018, page 59, 2018.

[16] Bradley M Bell and Frederick W Cathey. The iterated kalman filter update as a gauss-newton method. *IEEE Transactions on Automatic Control*, 38(2): 294–297, 1993.

[17] Michael J Black and Anand Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International journal of computer vision*, 19(1):57–91, 1996.

[18] Michael Bosse, Gabriel Agamennoni, Igor Gilitschenski, et al. Robust estimation and applications in robotics. *Foundations and Trends® in Robotics*, 4(4):225–269, 2016.

[19] Martin Brossard, Axel Barrau, and Silvère Bonnabel. Ai-imu dead-reckoning. *IEEE Transactions on Intelligent Vehicles*, 2020.

[20] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[21] Luca Carlone. Estimation contracts for outlier-robust geometric perception. *arXiv preprint arXiv:2208.10521*, 2022.

[22] Guobin Chang. Robust kalman filtering based on mahalanobis distance as outlier judging criterion. *Journal of Geodesy*, 88(4):391–401, 2014.

[23] Nived Chebrolu, Thomas Läbe, Olga Vysotska, Jens Behley, and Cyrill Stachniss. Adaptive robust kernels for non-linear least squares problems. *IEEE Robotics and Automation Letters*, 6(2):2240–2247, 2021.

[24] Changhao Chen, Xiaoxuan Lu, Andrew Markham, and Niki Trigoni. Ionet: Learning to cure the curse of drift in inertial odometry. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[25] Tat-Jun Chin and David Suter. The maximum consensus problem: recent algorithmic advances. *Synthesis Lectures on Computer Vision*, 7(2):1–194, 2017.

[26] Girish Chowdhary and Ravindra Jategaonkar. Aerodynamic parameter estimation from flight data applying extended and unscented kalman filter. *Aerospace science and technology*, 14(2):106–117, 2010.

[27] Omar Garcia Crespillo, Daniel Medina, Jan Skaloud, and Michael Meurer. Tightly coupled gnss/ins integration based on robust m-estimators. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1554–1561. IEEE, 2018.

[28] Omar Garcıa Crespillo, Alice Andreetti, and Anja Grosch. Design and evaluation of robust m-estimators for gnss positioning in urban environments. In *Proceedings of the 2020 International Technical Meeting of The Institute of Navigation, San Diego, CA, USA*, pages 21–24, 2020.

[29] DQF De Menezes, Diego Martinez Prata, Argimiro R Secchi, and José Carlos Pinto. A review on robust m-estimators for regression analysis. *Computers & Chemical Engineering*, 147:107254, 2021.

[30] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

[31] Frank Dellaert. Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:141–166, 2021.

[32] Frank Dellaert, Michael Kaess, et al. Factor graphs for robot perception. 2017.

[33] Jean-Emmanuel Deschaud, Pierre Dellenbach, Bastien Jacquet, and François Goulette. Ct-icp: Real-time elastic lidar odometry with loop closure. 2021.

[34] Gamini Dissanayake, Salah Sukkarieh, Eduardo Nebot, and Hugh Durrant-Whyte. The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. *IEEE transactions on robotics and automation*, 17(5):731–747, 2001.

[35] Kevin Doherty, Dehann Fourie, and John Leonard. Multimodal semantic slam with probabilistic data association. In *2019 international conference on robotics and automation (ICRA)*, pages 2419–2425. IEEE, 2019.

[36] Zeljko M Durovic and Branko D Kovacevic. Robust estimation with unknown noise statistics. *IEEE Transactions on Automatic Control*, 44(6):1292–1296, 1999.

[37] Per K Enge. The global positioning system: Signals, measurements, and performance. *International Journal of Wireless Information Networks*, 1(2): 83–105, 1994.

[38] Huazhen Fang, Mulugeta A Haile, and Yebin Wang. Robustifying the kalman filter against measurement outliers: An innovation saturation mechanism. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6390–6395. IEEE, 2018.

[39] Carles Fernandez-Prades, Javier Arribas, Pau Closas, Carlos Aviles, and Luis Esteve. Gnss-sdr: An open source tool for researchers and developers. In *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, pages 780–794, 2011.

[40] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[41] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Georgia Institute of Technology, 2015.

[42] Xiang Gao and Tao Zhang. *Introduction to visual SLAM: from theory to practice*. Springer Nature, 2021.

[43] Xiang Gao, Tao Zhang, Yi Liu, and Qinrui Yan. 14 lectures on visual slam: from theory to practice. *Publishing House of Electronics Industry*, 2017.

[44] Henri P Gavin. The levenberg-marquardt algorithm for nonlinear least squares curve-fitting problems. *Department of Civil and Environmental Engineering, Duke University*, pages 1–19, 2019.

[45] Giorgio Grisetti, Rainer Kümmerle, Hauke Strasdat, and Kurt Konolige. g2o: A general framework for (hyper) graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*, pages 9–13, 2011.

[46] Cagri Kilic; Jason Gross. Pathfinder gps, imu, and wheel odometry data on various terrains, 2020. URL https://dx.doi.org/10.21227/vz7z-jc84.

[47] Yulong Huang, Yonggang Zhang, Zhemin Wu, Ning Li, and Jonathon Chambers. A novel adaptive kalman filter with inaccurate process and measurement noise covariance matrices. *IEEE Transactions on Automatic Control*, 63(2):594–601, 2017.

[48] Yulong Huang, Yonggang Zhang, Yuxin Zhao, and Jonathon A Chambers. A novel robust gaussian–student's t mixture distribution based kalman filter. *IEEE Transactions on signal Processing*, 67(13):3606–3620, 2019.

[49] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.

[50] Peter J Huber. *Robust statistics*, volume 523. John Wiley & Sons, 2004.

[51] *Intel NUC Board/Kit NUC7i7DN Technical Product Specification*. Intel, 2019. Revision 109.

[52] *Intel RealSense Tracking Module 2 (TM2) Datasheet*. Intel, 2019. Revision 4.

[53] Prateek Jain, Purushottam Kar, et al. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–363, 2017.

[54] Michael Kaess, Viorela Ila, Richard Roberts, and Frank Dellaert. The bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Algorithmic Foundations of Robotics IX*, pages 157–173. Springer, 2010.

[55] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2): 216–235, 2012.

[56] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[57] Christopher D Karlgaard. Nonlinear regression huber–kalman filtering and fixed-interval smoothing. *Journal of guidance, control, and dynamics*, 38(2): 322–330, 2015.

[58] Christopher David Karlgaard. *Robust adaptive estimation for autonomous rendezvous in elliptical orbit*. PhD thesis, Virginia Tech, 2010.

[59] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[60] Cagri Kilic, Jason N Gross, Nicholas Ohi, Ryan Watson, Jared Strader, Thomas Swiger, Scott Harper, and Yu Gu. Improved planetary rover inertial navigation and wheel odometry performance through periodic use of zero-type constraints. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 552–559, Nov 2019. doi: 10.1109/IROS40897.2019.8967634.

[61] Cagri Kilic, Nicholas Ohi, Yu Gu, and Jason N. Gross. Slip-based autonomous zupt through gaussian process to improve planetary rover localization. *IEEE Robotics and Automation Letters*, 6(3):4782–4789, 2021. doi: 10.1109/LRA.2021.3068893.

[62] Cagri Kilic, Nicholas Ohi, Yu Gu, and Jason N Gross. Slip-based autonomous zupt through gaussian process to improve planetary rover localization. *IEEE Robotics and Automation Letters*, 6(3):4782–4789, 2021.

[63] Youngjoo Kim and Hyochoong Bang. Introduction to kalman filter and its applications. *Introduction and Implementations of the Kalman Filter*, 1:1–16, 2018.

[64] Manon Kok, Jeroen D Hol, and Thomas B Schön. Using inertial sensors for position and orientation estimation. *arXiv preprint arXiv:1704.06053*, 2017.

[65] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[66] SP Kwakkel, G Lachapelle, and ME Cannon. Gnss aided in situ human lower limb kinematics during running. In *Proceedings of the 21st International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2008)*, pages 1388–1397, 2008.

[67] Liyu Liu and Moeness G Amin. Tracking performance and average error analysis of gps discriminators in multipath. *Signal Processing*, 89(6): 1224–1239, 2009.

[68] Wenxin Liu, David Caruso, Eddy Ilg, Jing Dong, Anastasios I Mourikis, Kostas Daniilidis, Vijay Kumar, and Jakob Engel. Tlio: Tight learned inertial odometry. *IEEE Robotics and Automation Letters*, 5(4):5653–5660, 2020.

[69] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[70] Kirk MacTavish and Timothy D Barfoot. At all costs: A comparison of robust cost functions for camera correspondence outliers. In *2015 12th conference on computer and robot vision*, pages 62–69. IEEE, 2015.

[71] Kaj Madsen, Hans Nielsen, and O Tingleff. Methods for non-linear least squares problems (2nd ed.). page 60, 01 2004.

[72] Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff. Methods for non-linear least squares problems. 2004.

[73] Joshua G Mangelson, Derrick Dominic, Ryan M Eustice, and Ram Vasudevan. Pairwise consistent measurement set maximization for robust multi-robot map merging. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2916–2923. IEEE, 2018.

[74] Ricardo A Maronna, R Douglas Martin, Victor J Yohai, and Matías Salibián-Barrera. *Robust statistics: theory and methods (with R)*. John Wiley & Sons, 2019.

[75] Matthew T Mason. Toward robotic manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:1–28, 2018.

[76] Pratap Misra and Per Enge. Global positioning system: signals, measurements and performance second edition. *Massachusetts: Ganga-Jamuna Press*, 2006.

[77] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE international conference on robotics and automation*, pages 3565–3572. IEEE, 2007.

[78] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33 (5):1255–1262, 2017.

[79] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[80] *OEM615 Receivers Data Sheet*. Novatel, 2015. Ver.8.

[81] *Pinwheel L1/L2 Antennas Data Sheet*. Novatel, 2015. Ver.5.

[82] Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. *The International Journal of Robotics Research*, 32(7): 826–840, 2013.

[83] Yue Pan, Pengchuan Xiao, Yujie He, Zhenlei Shao, and Zesong Li. Mulls: Versatile lidar slam via multi-metric linear least square. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11633–11640. IEEE, 2021.

[84] Tim Pfeifer and Peter Protzel. Expectation-maximization for adaptive mixture models in graph optimization. In *2019 international conference on robotics and automation (ICRA)*, pages 3151–3157. IEEE, 2019.

[85] Tim Pfeifer, Peter Weissig, Sven Lange, and Peter Protzel. Robust factor graph optimization-a comparison for sensor fusion applications. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4. IEEE, 2016.

[86] Robert Piché, Simo Särkkä, and Jouni Hartikainen. Recursive outlier-robust filtering and smoothing for nonlinear systems using the multivariate student-t distribution. In *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2012.

[87] Mark L Psiaki. Backward-smoothing extended kalman filter. *Journal of guidance, control, and dynamics*, 28(5):885–894, 2005.

[88] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34 (4):1004–1020, 2018.

[89] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[90] Milad Ramezani, Matias Mattamala, and Maurice Fallon. Aeros: Adaptive robust least-squares for graph-based slam. *Frontiers in Robotics and AI*, page 23, 2022.

[91] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2003.

[92] David M Rosen, Michael Kaess, and John J Leonard. Robust incremental online inference over sparse factor graphs: Beyond the gaussian case. In *2013 IEEE International Conference on Robotics and Automation*, pages 1025–1032. IEEE, 2013.

[93] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696. IEEE, 2020.

[94] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. John wiley & sons, 2005.

[95] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[96] Simo Särkkä and Jouni Hartikainen. Non-linear noise adaptive kalman filtering via variational bayes. In *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2013.

[97] Simo Sarkka and Aapo Nummenmaa. Recursive noise adaptive kalman filtering by variational bayesian approximations. *IEEE Transactions on Automatic control*, 54(3):596–600, 2009.

[98] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE, 2020.

[99] Jingnan Shi, Heng Yang, and Luca Carlone. Robin: a graph-theoretic approach to reject outliers in robust estimation using invariants. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13820–13827. IEEE, 2021.

[100] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[101] Isaac Skog, Peter Handel, John-Olof Nilsson, and Jouni Rantakokko. Zero-velocity detection—an algorithm evaluation. *IEEE transactions on biomedical engineering*, 57(11):2657–2666, 2010.

[102] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

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

[104] Niko Sünderhauf and Peter Protzel. Switchable constraints vs. max-mixture models vs. rrr-a comparison of three approaches to robust pose graph slam. In *2013 IEEE International Conference on Robotics and Automation*, pages 5198–5203. IEEE, 2013.

[105] Niko Sünderhauf, Marcus Obst, Gerd Wanielik, and Peter Protzel. Multipath mitigation in gnss-based localization using robust optimization. In *2012 IEEE Intelligent Vehicles Symposium*, pages 784–789. IEEE, 2012.

[106] Tomoji Takasu. Rtklib: Open source program package for rtk-gps. *Proceedings of the FOSS4G*, 2009.

[107] Tomoji Takasu and Akio Yasuda. Development of the low-cost rtk-gps receiver with an open source program package rtklib. In *International symposium on GPS/GNSS*, volume 1, pages 1–6. International Convention Center Jeju Korea Seogwipo-si, Korea, 2009.

[108] Jo-Anne Ting, Evangelos Theodorou, and Stefan Schaal. A kalman filter for robust outlier detection. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1514–1519. IEEE, 2007.

[109] Chi Hay Tong and Timothy D Barfoot. Batch heterogeneous outlier rejection for feature-poor slam. In *2011 IEEE International Conference on Robotics and Automation*, pages 2630–2637. IEEE, 2011.

[110] Rudolph Van Der Merwe, Eric Wan, and Simon Julier. Sigma-point kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 5120, 2004.

[111] Christian Walck et al. Hand-book on statistical distributions for experimentalists. *University of Stockholm*, 10, 2007.

[112] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter. *Kalman filtering and neural networks*, pages 221–280, 2001.

[113] Hongwei Wang, Hongbin Li, Jun Fang, and Heping Wang. Robust gaussian kalman filter with outlier detection. *IEEE Signal Processing Letters*, 25(8): 1236–1240, 2018.

[114] Hongwei Wang, Wei Zhang, Junyi Zuo, and Heping Wang. An outlier-robust kalman filter with mixture correntropy. *arXiv preprint arXiv:1907.00307*, 2019.

[115] Ryan M Watson and Jason N Gross. Robust navigation in gnss degraded environment using graph optimization. In *Proceedings of the 30th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2017)*, pages 2906–2918, 2017.

[116] Ryan M Watson and Jason N Gross. Evaluation of kinematic precise point positioning convergence with an incremental graph optimizer. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 589–596, 2018.

[117] Ryan M Watson, Jason N Gross, Clark N Taylor, and Robert C Leishman. Enabling robust state estimation through measurement error covariance adaptation. *IEEE Transactions on Aerospace and Electronic Systems*, 56(3): 2026–2040, 2019.

[118] Ryan M Watson, Jason N Gross, Clark N Taylor, and Robert C Leishman. Robust incremental state estimation through covariance adaptation. *IEEE Robotics and Automation Letters*, 5(2):3737–3744, 2020.

[119] Weisong Wen and Li-Ta Hsu. Towards robust gnss positioning and real-time kinematic using factor graph optimization. *arXiv preprint arXiv:2106.01594*, 2021.

[120] Weisong Wen, Tim Pfeifer, Xiwei Bai, and Li-Ta Hsu. Factor graph optimization for gnss/ins integration: A comparison with the extended kalman filter. *NAVIGATION, Journal of the Institute of Navigation*, 68(2): 315–331, 2021.

[121] Weisong Wen, Guohao Zhang, and Li-Ta Hsu. Gnss outlier mitigation via graduated non-convexity factor graph optimization. *arXiv preprint arXiv:2109.00667*, 2021.

[122] Kamin Whitehouse and David Culler. Calibration as parameter estimation in sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 59–67, 2002.

[123] Daniel Wilbers, Christian Merfels, and Cyrill Stachniss. Localization with sliding window factor graphs on third-party maps for automated driving. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5951–5957. IEEE, 2019.

[124] Heng Yang and Luca Carlone. Certifiably optimal outlier-robust geometric perception: Semidefinite relaxations and scalable global optimization. *IEEE transactions on pattern analysis and machine intelligence*, 45(3):2816–2834, 2022.

[125] Heng Yang, Pasquale Antonante, Vasileios Tzoumas, and Luca Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robotics and Automation Letters*, 5 (2):1127–1134, 2020.

[126] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2020.

[127] Yuanxi Yang, Haibo He, and Guo-chang Xu. Adaptively robust filtering for kinematic geodetic positioning. *Journal of geodesy*, 75(2-3):109–116, 2001.

[128] Christopher Zach and Guillaume Bourmaud. Iterated lifting for robust cost optimization. In *BMVC*, 2017.

[129] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014.

[130] Rui Zhang, Hai Yang, Fabian Höflinger, and Leonhard M Reindl. Adaptive zero velocity update based on velocity classification for pedestrian tracking. *IEEE Sensors journal*, 17(7):2137–2145, 2017.

[131] Yuxiao Zhang, Alexander Carballo, Hanting Yang, and Kazuya Takeda. Perception and sensing for autonomous vehicles under adverse weather conditions: A survey. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196:146–177, 2023.

[132] Shibo Zhao, Hengrui Zhang, Peng Wang, Lucas Nogueira, and Sebastian Scherer. Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8729–8736. IEEE, 2021.

[133] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016.

A

Robustness Metrics for Adaptive M-estimators

This chapter presents some quantitative and qualitative robustness metrics for the generalized M-estimator (equation (5.1)) presented in Chapter 5. The model chosen is a simple scalar location model

$$z = x + u, \tag{A.1}$$

where $z, x, u \in \mathbb{R}$. z is the measured value, x is the true unknown value and u is the noise or error generated from a contaminated distribution

$$u \sim (1 - \epsilon)F + \epsilon\, G. \tag{A.2}$$

G is the contaminating distribution with a contamination proportion of ϵ. The optimal solution $\hat{x}$ can be obtained by solving

$$\hat{x} = \arg\min_{x} \sum_{k} \rho(z_k - x). \tag{A.3}$$

The robustness metrics [18, 74] for M-estimators are divided into 2 categories. The first category measures the sensitivity of the estimator to errors in a single measurement and the second category measures sensitivity to errors in multiple measurements.

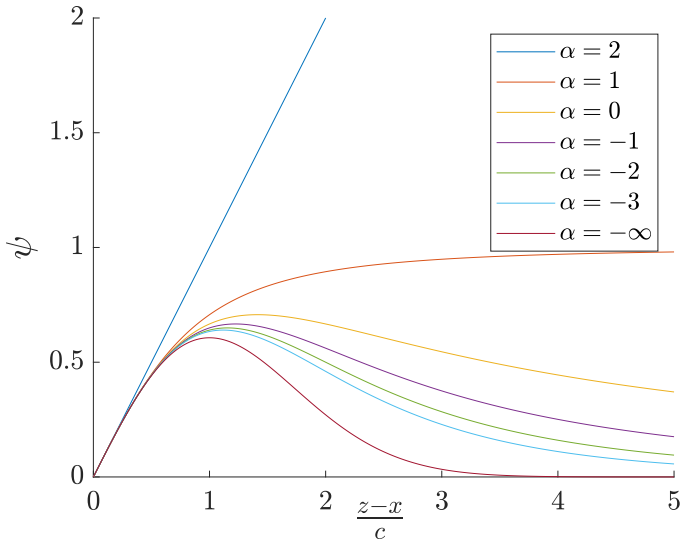


Figure A.1.1: *IF curve of generalized M-estimators [14]*

A.1 Robustness to Error in Single Measurement

This section discusses robustness metrics that do not require knowledge of the true distribution of the error u. These metrics are the Influence (IF) curve and the Gross Error Sensitivity (GES). The IF curve represents the sensitivity of the estimate $\hat{x}$ to infinitesimal perturbations to a single measurement. Essentially it represents the asymptotic bias caused by a single outlier as a function of normalised residuals. The IF curve is proportional to the ρ' or ψ function.

GES is the maximum value of the IF curve. From figure (A.1.1) it can be seen the GES of least squares ($\alpha = 2$) is ∞. Also, GES goes down with the decrease in the shape parameter α.

A.2 Robustness to Errors in Multiple Measurements

The Maximum Bias (MB) curve measures the maximum possible bias or error of an M-estimator given a certain proportion (ϵ) of outliers. The Breaking Point (BP) is the value of ϵ for which MP value goes to ∞. The equation for calculating MB value

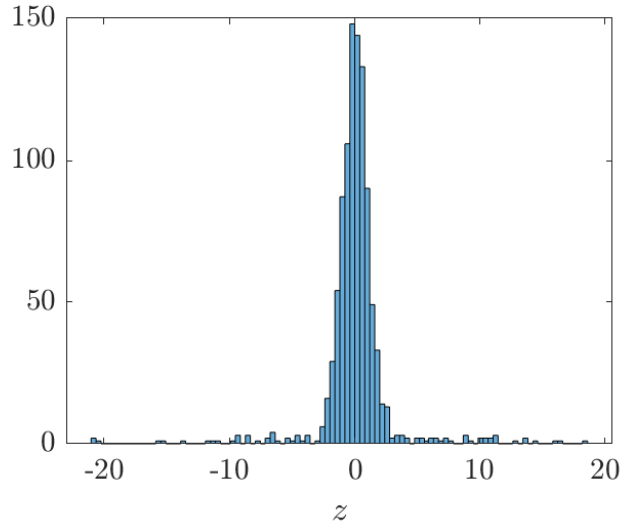


Figure A.2.1: *Sampled measurements z from* $0.9\mathcal{N}(0,1) + 0.1\mathcal{N}(0,100)$

for a known ϵ ([18]) is given by

$$\text{MB} = |x|, \tag{A.4}$$

such that

$$(1-\epsilon)\mathbf{E}_z[\psi(z-x)] + \epsilon = 0. \tag{A.5}$$

The solution to equation (A.5) can be found for the simple model (A.1). Let us assume the errors in the measurements come from

$$u \sim (1-\epsilon)\mathcal{N}(0,1) + \epsilon\mathcal{N}(0,100), \tag{A.6}$$

and the true value of x is 0. Now 1000 measurements of x are sampled from this distribution with $\epsilon = 0.1$, as shown in figure (A.2.1). The MB value can be found by plotting the value of $\mathbf{E}_z[\psi(z-x)]$ for different values of x and checking where it has a value of $-\frac{\epsilon}{1-\epsilon}$. The expectation can be approximated with the mean of the function ψ for all the sampled measurements.

$$\mathbf{E}_z[\psi(z-x)] \approx \frac{1}{N}\sum_k \psi(z_k - x), \tag{A.7}$$

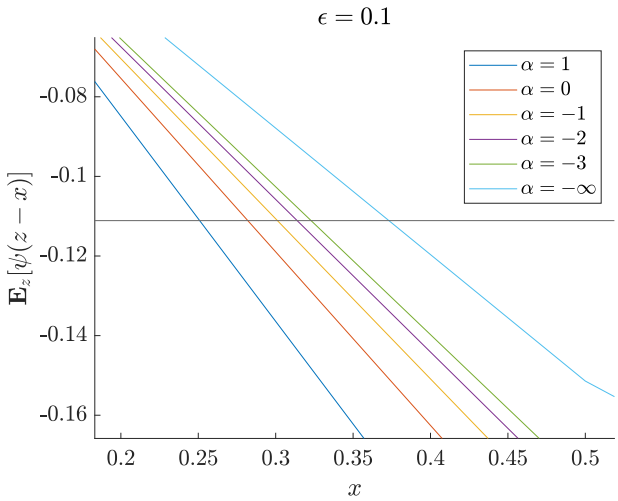


Figure A.2.2: *MB values for each α is the value of x where expectation has the value $-\frac{\epsilon}{1-\epsilon}$ (black line). Here $\epsilon = 0.1$*

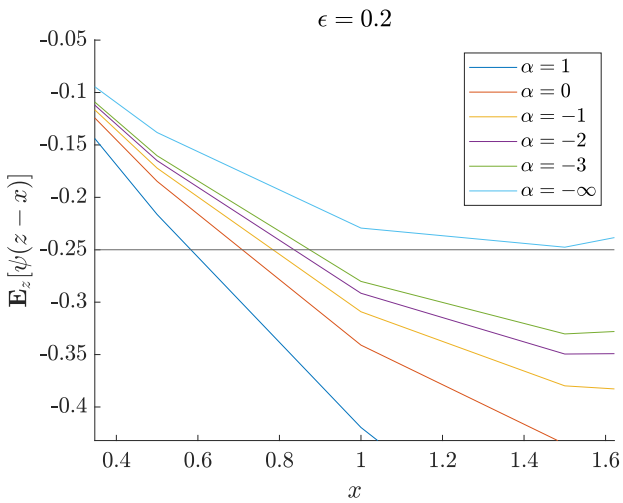


Figure A.2.3: *MB values for different α for $\epsilon = 0.2$*

where N is 1000 here.

Figures (A.2.2) and (A.2.3) show MB values for contamination proportions of 0.1 and 0.2 respectively. Interestingly, an opposite trend to that of IF curve can be seen here. A decrease in α values causes an increase in the MB values. In these figures, $c = 1$ which is the correct scale value. The MB values increase for all shape values

with the increase in the contamination proportion which is expected but does not increase by a large amount which demonstrates their robustness properties.

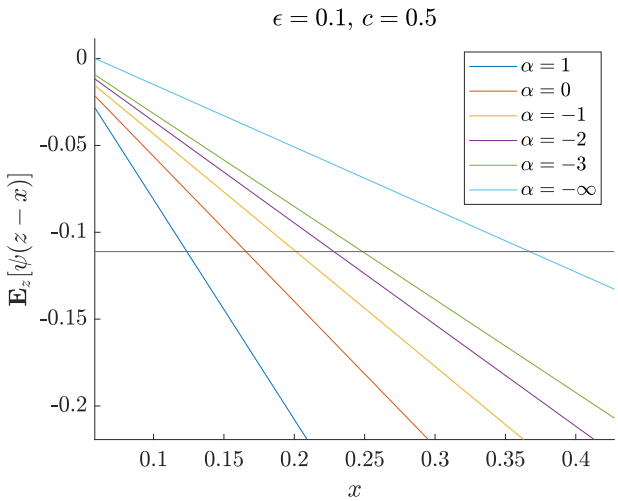


Figure A.2.4: *MB values with* $\epsilon = 0.1$ *and* $c = 0.5$.

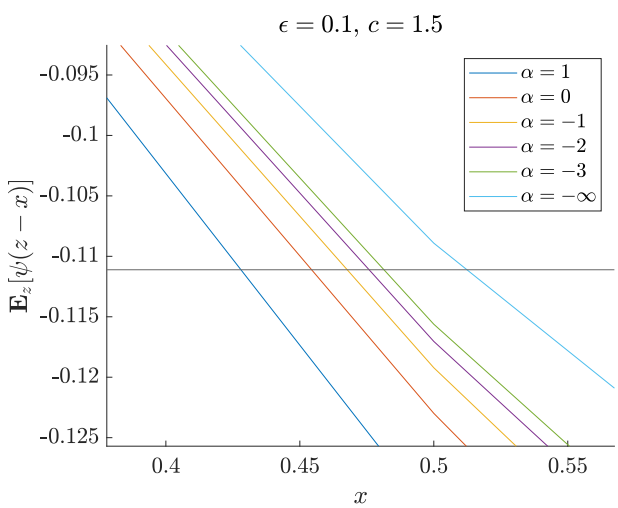


Figure A.2.5: *MB values with* $\epsilon = 0.1$ *and* $c = 1.5$.

If the scale parameter c is lowered to 0.5, the MB values decrease compared to $c = 1$ (figure (A.2.4)). This is because c sets a lower inlier threshold and helps reduce

the effect of outliers further. Increasing c to 1.5 increases the MB values which is caused due to increase in the inlier threshold as can be seen in figure (A.2.5).

A.3 Asymptotic Variance

Recall from Chapter 2, that M-estimators are asymptotically normal. That is, the estimate $\hat{x}$ in equation (A.3) will follow a Gaussian distribution given by

$$\hat{x} \sim \mathcal{N}(x*, \frac{v}{n}), \tag{A.8}$$

where

$$v = \frac{\mathbf{E}_H(\psi(z - x*))^2}{(\mathbf{E}_H\psi'(z - x*))^2}. \tag{A.9}$$

n is the number of measurements. $x*$ is the solution of $\mathbf{E}_H\psi(z - x) = 0$ where H is the error distribution. v is called the asymptotic variance of the M-estimator. Asymptotic variances for model (A.1), with H given by (A.6), are shown in table (A.3.1). It can be seen that the least squares estimator has the highest variance for

Table A.3.1: *Asymptotic variance v for different values of α for two different contamination proportions.*

α	2	1	0	-1	-2	-3	$-\infty$
$\epsilon = 0.1$	8.0576	1.4092	1.2667	1.2990	1.3331	1.3605	1.5457
$\epsilon = 0.4$	37.6115	3.9036	2.3766	2.2964	2.3162	2.3450	2.6113

contamination ratios. Its variance is also much larger than all other M-estimators. For contamination of 0.1, as α decreases, the variance goes down and reaches its lowest value at $\alpha = 0$ after which it starts increasing again. The same behavior can be seen for contamination of 0.4, where $\alpha = -1$ has the lowest variance.

A.4 Bias of Estimate

Biases of the M-estimators for the simple model (A.1) can be compared by calculating the estimate $\hat{x}$ using IRLS. Due to the linear scalar model here, the IRLS algorithm is easy to implement. The scale parameter c is set to 1. The algorithm is initialized with the mean of the measurements. Then for every iteration, the weight of each

measurement is calculated with the standard M-estimator weight formula followed by solving for x as the weighted mean of the measurements. It is important to note here that Maronna et al. [74] suggest initialization with robust estimates of x and c. Even though this is easy for a scalar location model, where the initialization can be done with the median value of $\{z_k\}$, it is not straightforward for a more complex regression model.

Algorithm 6: Simplified IRLS

Input: $\{z_k\}$, α, c
Output: $\hat{x}$
% Initialize with mean
$\hat{x}_0 = \frac{\sum_k z_k}{N}$;
while !converged **do**
  % Step 1: calculate weight of every measurement
  $w_k = \frac{\rho'(z_k - \hat{x}_{j-1}, \alpha, c)}{z_k - \hat{x}_{j-1}}$;
  % Step 2: solve for x
  $\hat{x}_j = \frac{\sum_k w_k z_k}{\sum_k w_k}$;
end

Table A.4.1: *Bias of estimate for* $(1-\epsilon)\mathcal{N}(0,1) + \epsilon\mathcal{N}(0,100)$

α	2	1	0	-1	-2	-3	$-\infty$
$\epsilon = 0.1$	0.0669	0.0347	0.0264	0.0240	0.0232	0.0228	0.0222
$\epsilon = 0.4$	0.0014	0.0496	0.0415	0.0335	0.0290	0.0265	0.0190

Table (A.4.1) shows the bias values for different α values. Note, that the least squares estimator ($\alpha = 2$) only calculates the mean of the data. For $\epsilon = 0.1$, $\alpha = -\infty$ gives the best estimate of x (remember the true value of x is 0). This makes sense from the IF curve point of view since the influence of outliers in the estimation decreases as α goes to $-\infty$. Interestingly least square estimator has the least bias when $\epsilon = 0.4$. This is because of the true value centering at 0. This causes larger outliers on both sides of the mean to create a canceling effect in the mean formula.

To circumvent this issue, a different error distribution is selected, where $u \sim (1-\epsilon)\mathcal{N}(10,1) + \epsilon\mathcal{N}(15,1)$. Table (A.4.2) shows the bias values for measurements sampled from this distribution. The least squares estimator again performs the worst

and $\alpha = -\infty$ performs the best for both contamination ratios. Generally, the bias goes down with a decrease in α.

Table A.4.2: *Bias of estimate for* $(1-\epsilon)\mathcal{N}(10,1) + \epsilon\mathcal{N}(15,1)$

α	2	1	0	-1	-2	-3	$-\infty$
$\epsilon = 0.1$	10.4771	10.1892	10.0941	10.0571	10.0401	10.0314	10.0149
$\epsilon = 0.4$	11.9871	11.3131	10.6249	10.3396	10.2138	10.1503	10.0300

Table A.4.3: *Bias of estimate for* $0.6\mathcal{N}(10,1) + 0.4\mathcal{N}(15,1)$

α	2	1	0	-1	-2	-3	$-\infty$
$c = 3$	11.9871	11.7658	11.6648	11.6057	11.5670	11.5395	11.3794
$c = 5$	11.9871	11.8875	11.8686	11.8604	11.8557	11.8527	11.8390

Table (A.4.3) shows the importance of correctly estimating the scale parameter c. The incorrect scale parameter estimate causes the M-estimators to lose their robustness properties. This is because a larger inlier threshold would mark the outliers as inliers resulting in them getting a larger influence in the estimation. The bias-variance trade-off is also clearly visible in the tables (A.3.1, A.4.1, A.4.2). It can be seen that the estimators with more robust estimates have higher asymptotic variance.