Reliable Robotic Perception: From Outlier-Robust Estimation to Task-Aware Runtime Monitoring

by

Pasquale Antonante

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2024

Author . . . .
Department of Aeronautics and Astronautics
January 12, 2024

Certified by . . . .
Luca Carlone, Thesis Supervisor
Associate Professor of Aeronautics and Astronautics

Certified by . . . .
Marco Pavone
Associate Professor of Aeronautics and Astronautics, Stanford University

Certified by . . . .
Chuchu Fan
Assistant Professor in the Department of Aeronautics and Astronautics

Accepted by . . . .
Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

“Tutta la nostra conoscenza ha origine nelle nostre percezioni.”

—Leonardo Da Vinci

Reliable Robotic Perception: From Outlier-Robust Estimation to Task-Aware Runtime Monitoring

by

Pasquale Antonante

Submitted to the Department of Aeronautics and Astronautics
on January 12, 2024, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Reliable perception is a key prerequisite for safe operation of robots and autonomous vehicles. The future of the field relies on public trust and provable correctness of behavior in real-world scenarios. Though commonly used, testing and simulation alone are insufficient to ensure correctness and do not provide sufficient evidence for safety certification. The current literature lacks a system-wide framework to formally verify the safety requirements of the perception system of an autonomous vehicle. Moreover, current perception algorithms tend to fail in the presence of many outliers and require extensive parameter tuning. This thesis presents a comprehensive exploration of outlier-robust estimation algorithms, perception monitoring, and risk assessment to enhance the robustness and safety of robots and autonomous vehicles.

The first part of the thesis focuses on *geometric perception*, which is the task of estimating geometric models (*e.g.,* poses) from sensor measurements (*e.g.,* LiDAR scans). Geometric perception is plagued by the presence of *outliers* —spurious measurements— that compromise the accuracy of the estimated geometric model. Computing robust estimates in the face of outliers has been a central topic in computer vision and robotics. In this thesis I introduce two unifying formulations for outlier-robust estimation, and investigate fundamental limits, practical algorithms, and applications. In particular I present two outlier-robust estimation algorithms (together with two variations that are parameter-free), that are able to robustly estimate geometric models in the presence of a high percentage of outliers.

The second part of the thesis focuses on *task-aware runtime monitoring* of perception systems in high-stakes robotics applications such as autonomous vehicles. Safety and performance are key enablers for autonomous driving: on the one hand we want our autonomous vehicles to be safe, while at the same time their performance (*e.g.,* comfort or progression) is key to adoption. In this thesis I formalize the problem of *task-aware runtime monitoring* and present a framework that uses the diagnostic information present in the perception system to detect and identify faults at runtime, while assessing the risk they pose to the autonomous vehicle.

Thesis Supervisor: Luca Carlone
Title: Associate Professor of Aeronautics and Astronautics

Thesis Committee Member: Marco Pavone
Title: Associate Professor of Aeronautics and Astronautics, Stanford University

Thesis Committee Member: Chuchu Fan
Title: Assistant Professor in the Department of Aeronautics and Astronautics

Acknowledgments

Research is a journey of daily learning and innovation, and it's the people I've worked with who have truly brought this journey to life. Their support and guidance have been invaluable, and I would like to thank them for this.

First of all, I would like to thank my advisor, Luca Carlone. I'm grateful not only for the opportunity he gave me to pursue a Ph.D. at MIT under his guidance, but also for his unwavering support both in academic and personal aspects of my life. I've learned a lot from Luca, he taught me everything I know about robotics, and has been a cornerstone in my growth. The conversations we had over the years have always been a source of inspiration and confidence, highlighting his qualities as an exceptional leader and mentor. I am more than lucky to be one of his students.

This work was greatly influenced by the people I collaborated with, especially my thesis committee members, Marco Pavone and Chuchu Fan. Their guidance was instrumental in navigating the world of robotics and autonomous vehicles. They always found time to meet with me and guide me through the challenges I faced. Chuchu taught me a lot about formal methods, a field I barely knew before meeting her. Her patience and expertise opened new horizons for me, for which I am immensely grateful. I want to thank Marco not only for having a central role in shaping this research, but also for the incredible time I spent at NVIDIA. His support was crucial for me to get the opportunity to work at NVIDIA, and I am very grateful for that.

I would like to thank my collaborators: Vasileios Tzoumas, Heng Yang, Sushant Veer, Xinshuo Weng, Yulong Cao, and Karen Leung. Their dedication in tackling challenging problems, whether in code or theory, was a constant source of motivation for me. I would also like to thank the entire MIT SPARK Lab. The research lab has grown a lot over the past few years, and it has been a humbling experience for me to see it grow and thrive. I will miss the time spent with the other students, the inspiring conversations, and the fun we had together.

A special acknowledgment is due to my parents. Their support in my decision to leave Italy and pursue my dreams, despite the difficulty of separation, is a testament

to their unconditional love. I thank them for their sacrifices and for always believing in me.

Similarly, I thank my wife's family –Elena, Elisa, and Dunia– for their unwavering support and for welcoming me into their family with warmth and love.

And I want to thank Yi, who has been incredibly supportive and understanding. I'm really lucky to have him as a co-founder of the company we'll be building over the years to come.

Finally, my deepest appreciation goes to my wife, Sara. She is an incredible person and has been a pillar in my life. She has shown an incredible amount of patience and strength, and I am grateful for every single moment we have spent together. I've been very fortunate to have her by my side, and I'm excited to share what is coming next.

My Ph.D. research has been partially funded by the NSF CAREER award "Certifiable Perception for Autonomous Cyber-Physical Systems", MathWorks, and NVIDIA.

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

Chapter 1

Introduction

When we imagine the future, we imagine a world where robots seamlessly navigate alongside humans. This vision hinges on a crucial aspect of evolution: perception. Human perception is a remarkable product of evolution that has enabled us to survive and thrive in a complex, unpredictable world. Our early ancestors depended on accurate and reliable perception of their surroundings in order to find food, avoid predators, and navigate terrain. While reliable perception was central for human survival, reliable robotic perception is central for the widespread adoption of robots.

Robots rely on sensors like cameras and LiDARs to capture data about their surroundings. The perception system then processes the raw sensory data to create a model of the environment. Despite decades of research advancing the state of the art, robots still struggle to replicate the ease and reliability of human perception. The key difference between human and robotic perception lies in the sophistication with which humans reason about perceptual data.

This thesis tries to make progress towards bridging the gap between human and robotic perception. Toward this goal we develop general-purpose algorithms to limit the impact of wrong sensor measurements (outlier-robust estimation) in spatial-perception problems, we then propose a system-level framework to detect failures (fault detection and identification) and develop tools to enable autonomous vehicles (AVs) to assess the risk perception failures pose to the robot and its environment (task-aware runtime monitoring). Such advances promise to enable safer, more trustworthy autonomy.

1.1 Outlier-Robust Estimation

Nonlinear estimation is a fundamental problem in robotics and computer vision, and is the backbone of modern perception systems for localization and mapping [1], object pose estimation [2], [3], motion estimation and 3D reconstruction [4], [5], shape analysis [6], [7], virtual and augmented reality [8], and medical imaging [9], among others.

Nonlinear estimation can be formulated as an optimization problem, where one seeks to find the estimate that best explains the observed measurements. A typical perception pipeline includes a *perception front-end* that extracts and matches relevant features from raw sensor data (*e.g.,* camera images, lidar point clouds). These putative feature matches are then passed to a *perception back-end* that uses nonlinear estimation to compute quantities of interest (*e.g.,* the location of the robot, the pose of external objects). In the idealized case in which the feature matches are all correct, the back-end can perform estimation using a least squares formulation:[1]

$$\min_{\boldsymbol{x} \in \mathcal{X}} \sum_{i \in \mathcal{M}} r^2(\boldsymbol{y}_i, \boldsymbol{x}), \tag{1.1}$$

where $\boldsymbol{x}$ is the variable we want to estimate (*e.g.,* a 3D pose), $\mathcal{X}$ is its domain (*e.g.,* the set of poses), $\mathcal{M}$ is the set of given measurements (*e.g.,* pixel observations of points belonging to the object), $\boldsymbol{y}_i$ is the i-th measurement ($i \in \mathcal{M}$), and $r(\boldsymbol{y}_i, \boldsymbol{x}) \geq 0$ is the *residual* error of the i-th measurement, which quantifies how well a given $\boldsymbol{x}$ fits a measurement $\boldsymbol{y}_i$, (*e.g.,* $r(\boldsymbol{y}_i, \boldsymbol{x}) = |y_i - \mathbf{a}_i^{\mathsf{T}} \boldsymbol{x}|$ for the linear, scalar measurement case). The problem in Eq. (1.1) produces a maximum-likelihood estimate when the measurement noise is Gaussian, see *e.g.,* [1]. However, despite its apparent simplicity, it is already hard to solve globally, since the cost function in Eq. (1.1) and the domain $\mathcal{X}$ are typically non-convex in robotics applications [3], [10].

The development of perception systems that can work in challenging real-world conditions requires the design of outlier-robust estimation methods. Perception front-ends are typically based on image or signal processing methods [11] or learning methods [12] for feature detection and matching. These methods are prone to produce incorrect matches, which result in completely wrong measurements $\boldsymbol{y}_i$ in Eq. (1.1) and compromise the accuracy of the solution returned by Eq. (1.1). Computing robust estimates in the face of these *outliers* has been a central topic in computer vision and robotics.

For low-dimensional estimation problems, *e.g.,* object pose estimation from images or point clouds, researchers have often resorted to combinatorial formulations for outlier rejection. In particular, a popular formulation is based on *consensus maximization* [13], [14], which looks for an estimate maximizing the number of measurements explained within a given *inlier threshold* ϵ (or equivalently, minimizes the number of outliers):

$$\min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M}}} \quad |\mathcal{O}| \quad \text{s.t.} \quad r(\boldsymbol{y}_i, \boldsymbol{x}) \leq \epsilon, \quad \forall i \in \mathcal{M} \setminus \mathcal{O}. \tag{1.2}$$

The *Maximum Consensus* (MC) problem in Eq. (1.2) is then solved using RANSAC [15] or branch-and-bound (BnB) [14]. However, RANSAC is non-deterministic, requires a minimal solver[2], and is limited to problems where the estimate can be computed from a small number of measurements [16]. Similarly, BnB runs in the worst-case in exponential time and does not scale to large problems.

For high-dimensional estimation problems, *e.g.,* bundle adjustment and SLAM, researchers have more heavily relied on *M-estimation* to gain robustness against outliers [17]. M-estimation replaces the least-squares cost in Eq. (1.1) with a function ρ

[1] We use lowercase characters, *e.g.,* x, to denote real scalars or functions, bold lowercase characters, *e.g.,* $\boldsymbol{x}$, for real vectors, bold uppercase characters, *e.g.,* $\boldsymbol{A}$ for real matrices, and calligraphic uppercase characters, *e.g.,* $\mathcal{M}$, for sets (either discrete or continuous); as exceptions, we use the standard notation $\mathbb{R}$ to denote the set of real numbers, and $\mathbb{N}$ to denote the set of non-negative integers. $|x|$ is the absolute value of x, and $|\mathcal{M}|$ is the cardinality of $\mathcal{M}$. If $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$, then $\|\boldsymbol{x}\|_1 \triangleq \sum_{i=1}^{n} |x_i|$ is $\boldsymbol{x}$'s Manhattan norm; $\|\boldsymbol{x}\|_2 \triangleq \sqrt{\sum_{i=1}^{n} x_i^2}$ is $\boldsymbol{x}$'s Euclidean norm; and $\|\boldsymbol{x}\|_\infty \triangleq \max\{|x_1|, |x_2|, \ldots, |x_n|\}$ is $\boldsymbol{x}$'s infinity norm.

[2] A minimal solver in robotic perception is a specialized algorithm designed to solve a specific problem with the least amount of input data.

that is less sensitive to measurements with large residuals:

$$\min_{\boldsymbol{x} \in \mathcal{X}} \quad \sum_{i \in \mathcal{M}} \rho(r(\boldsymbol{y}_i, \boldsymbol{x}), \epsilon), \tag{1.3}$$

where, for instance, ρ can be a Huber, Cauchy, or Geman-McClure cost [18]. The M-estimation problem in Eq. (1.3) has the advantage of leading to a continuous (rather than combinatorial as in MC) optimization problem, which however is still hard to solve globally due to the typical non-convexity of the cost function and constraint set $\mathcal{X}$. Typical robotics applications use iterative local solvers to minimize Eq. (1.3), see [19]–[21]. However, local solvers require a good initial guess (often unavailable in practical applications) and are easily trapped in local minima corresponding to poor estimates.

All in all, the literature is currently lacking an approach that simultaneously satisfies the following design constraints:

(i) is fast and scales to large problems,

(ii) is deterministic,

(iii) can operate without requiring an initial guess.

This gap in the literature is the root cause for the brittleness of modern perception systems and is limiting the use of perception in safety-critical applications, from self-driving cars [22], to autonomous aircrafts [23], and spacecrafts [24].

An additional limitation of state-of-the-art robust estimation algorithms is that they require knowledge of the expected (inlier) measurement noise. This knowledge is encoded in the parameter ϵ in both Eq. (1.2) and Eq. (1.3). However, in many problems, characterizing this parameter is time-consuming (*e.g.,* it requires collecting data in a controlled environment to compute statistics) or is based on trial-and-error (*i.e.,* requires manual parameter tuning by a human expert). Also, this approach is not suitable for long-term operation: imagine a ground robot performing life-long SLAM; after months of operations, the noise statistics may vary (*e.g.,* a flat tire leads to increased odometry noise), making the factory calibration unusable.

1.2 Fault Detection and Identification

While outlier-robust algorithms can make individual perceptual processes more reliable, achieving truly dependable autonomy requires a system-level perspective. Modern perception systems integrate outputs from a complex array of potentially heterogeneous perception modules, each susceptible to unique failure modes. Detecting perception failures before they can cause harm is especially important for safety-critical systems like autonomous vehicles.

The automotive industry is undergoing a change that could revolutionize mobility. Self-driving cars promise a deep transformation of personal mobility and have the potential to improve safety, efficiency (*e.g.,* commute time, fuel), and induce a paradigm shift in how entire cities are designed [25]. One key factor that drives the adoption of such technology is the capability of ensuring and monitoring safe operation. The American Automobile Association (AAA) reports [26] that vehicles with autonomous driving features consistently failed to avoid crashes with other cars or bicycles. An analysis by Business Insider [27] found that as autonomous vehicle companies increased testing, the number of accidents involving AVs surged. This is a clear sign that the industry needs a sound methodology, embedded in the design process, to guarantee safety and build public trust.

Safe operation requires AVs to correctly understand their surroundings, in order to avoid unsafe behaviors. In particular, AVs rely on onboard *perception systems* to provide situation awareness and inform the onboard decision-making and control systems. The perception system uses sensor data and prior knowledge (*e.g.,* high-definition maps) to create an internal representation of the surrounding environment, including estimates for the positions and velocities of other vehicles and pedestrians, or the presence of traffic signs and traffic lights. Modern perception systems use both data-driven and classical methods. While classical methods are well-rooted in signal processing and estimation theory and have been extensively studied in robotics and computer vision, they may still have unexpected failure modes in practice, *e.g.,* local convergence in the Iterative Closest Point for 3D object pose estimation [2] or premature termination of robust estimation techniques as RANSAC [15], among many other examples. The use of data-driven methods further exacerbates the problem of ensuring correctness of the perception outputs, since current neural network architectures are still prone to creating unexpected and often unpredictable failure modes [28].

Ensuring and monitoring the correct operation of the perception system of an AV is a major challenge. Industry heavily relies on simulation and testing to provide evidence of safety. Although there is an increasing interest in the area of safety certification and runtime monitoring, the literature lacks a system-level framework to organize and reason over the diagnostic information available at runtime for the purpose of detecting and identifying potential perception-system failures. Reliable runtime monitoring would enable the vehicle to have a better understanding of the conditions it operates in, and would give it enough notice to take adequate actions to preserve safety (*i.e.,* switch to fail-safe mode or hand over the control to a human operator) in case of severe failures. In this thesis, we use the term "failure" (or "fault") in a general sense, to also denote failures of the *intended functionality* [29], [30]. For instance, a neural network can execute correctly (*e.g.,* without errors in the implementation or in the hardware running the network) but can still fail to produce a correct prediction for out-of-distribution inputs. Then, *fault detection* is the problem of detecting the *presence* of a fault in the system, while *fault identification* is the problem of inferring which components of the system are faulty. The latter is particularly important since:

(i) not every fault has the same severity, hence understanding which component is failing may lead to different responses,

(ii) a designer can use fault statistics to decide to focus research and development efforts on certain components,

(iii) a regulator can use information about specific faults to trace the steps or even determine responsibilities after an accident.

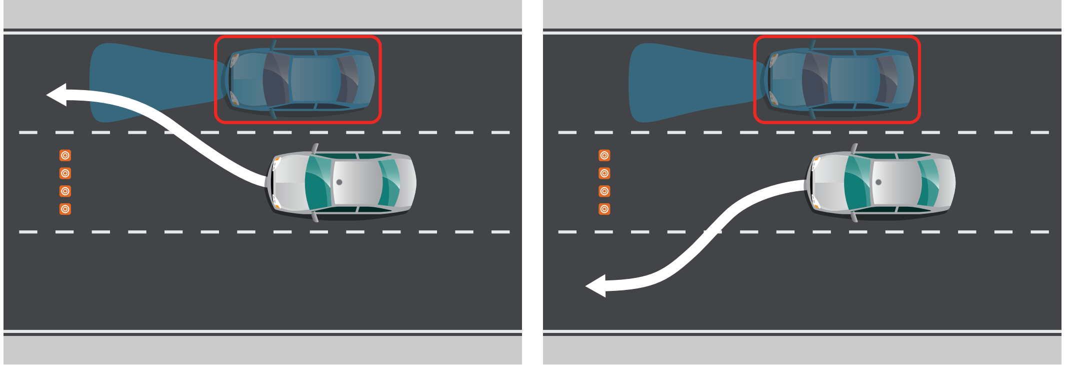
Most of the existing literature on runtime monitoring (which we review more extensively in Section 3.6) has focused on detecting failures of specific modules or specific algorithms, like localization [31], [32], semantic segmentation [33], or obstacle detection [34]. These methodologies often use a white-box approach (the monitor knows how the monitored algorithm works to some extent), and are sometimes computationally expensive to run [33]. However, the literature still lacks a framework for system-level monitoring of perception systems, which is able to detect and identify failures in complex systems involving both classical and data-driven (possibly asynchronous, multi-modal)[3] perception algorithms.

1.3 Task-Aware Perception Monitor

Despite the fast-paced progress in robotics and autonomous systems, perception modules in autonomous vehicles still encounter a spate of failure modes, which can compromise the safety of passengers, other drivers, and pedestrians. However, these failures occur frequently enough that reverting to a fallback safety maneuver for each such detection is prohibitively detrimental to the performance of the AV. In this thesis, we work towards developing a *task-aware perception monitor* that only triggers when the perception failure poses a significant risk to the AV's motion plan, thereby, promoting safe yet performant driving. The key insight is that the severity of a perception failure depends not only on the failure itself, but also on the ego-vehicle motion plan. For example, if the perception system incorrectly detects the position of a distant vehicle, the error does not pose a significant risk to the AV's motion plan and does not require a safety maneuver. Another, more nuanced example that highlights the importance of developing a task-aware perception monitor is illustrated in Fig. 1-1. The ego-vehicle misses an obstacle in the adjacent lane, but if the ego-vehicle's motion plan does not lead to a collision with the missing obstacle, the perception failure is not task-relevant and thus less risky.

We envision a task-aware perception monitor that embodies three main components, as shown in Fig. 1-2. First, the *perception failure detection and identification*

[3] Modern perception systems rely on data from multiple sensors and are implemented in multi-threaded architectures, where each algorithm may be executed at a different rate.



(a) Task-relevant failure

(b) Non-task-relevant failure

Figure 1-1: **Illustration of task-aware perception failure detection.** The white car is the ego vehicle and the blue car is an external (non-ego) vehicle. In this example, the non-ego vehicle has not been detected by the perception system of the ego vehicle. Then, Fig. 1-1a depicts a task-relevant missing obstacle, as the ego vehicle's motion plan will likely collide with the non-ego vehicle due to the misdetection. Fig. 1-1b depicts a non-task-relevant missing vehicle, as the ego vehicle's motion plan will not lead to a collision with the non-ego vehicle, regardless of the perception failure.

module identifies perception faults (developed in Chapter 3) and isolates the responsible modules and failure modes. Second, the *plausible scene generator* leverages the knowledge of the perception failure modes, provided by the failure identification, to construct a probabilistic (possibly multi-modal) description of plausible alternative models for the AV's surroundings that supports the actual world scene.[4] Finally, a *task-aware risk estimator* (developed in Chapter 4) assesses the increased risk to the AV's motion plan due to the perception failure. While there are several recent works on perception failure detection [34]–[37], and also some work on plausible scene generation [38]–[40], comparably much less work on task-aware risk estimation. In this thesis we will discuss the mechanics of plausible scene generation given a perception failure mode, and then turn our attention to the task-aware risk estimator.

[4] The probabilistic description of plausible AV surroundings might be highly stochastic and multi-modal. Planning in the plausible scene would be impractical and possibly not conducive to a good plan; however, we can still leverage it to estimate the risk of the perception failures to the AV using the approach we develop in this thesis.

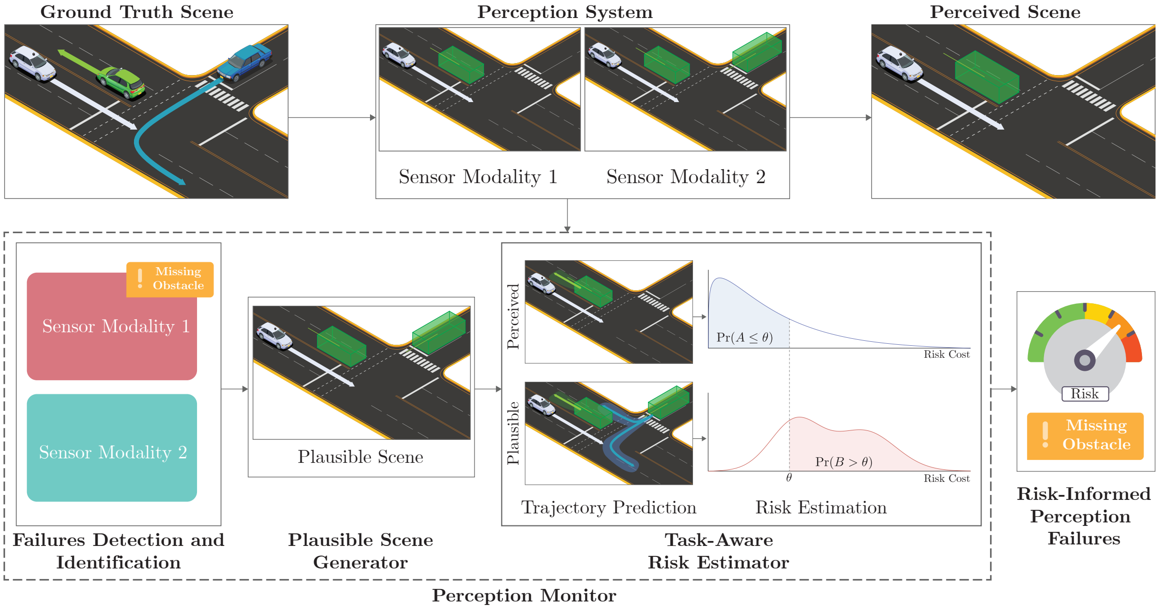


Figure 1-2: **Task-aware perception monitor overview.** The scene contains the ego vehicle (white car) and two non-ego agents (green and blue car). The top row shows a scenario in which a perception system fails to detect an obstacle (the blue car): one of the two sensor modalities used by the perception system is not able to detect the obstacle (top-center subfigure), inducing a missing-obstacle failure in the perception output (top-right subfigure). The bottom row depicts the proposed task-aware perception monitor. The failure detection and identification module detects that sensor 1 is failing (for example using spatio-temporal information). The plausible scene generator, uses the information about the active failures, generates a plausible scene from the perceived scene. Finally, the task-aware risk estimator computes the risk associated with the failure. The shaded (green and blue) regions in the bottom-row scenes represent the uncertainty in the trajectories, as computed by the non-ego trajectory prediction module. The possible trajectories induce a distribution of risk costs for each scene, which are used to estimate the risk associated with a perception failure. If the risk in the plausible scene is significantly higher than the risk in the perceived scene, we detect the failure as task relevant. Our detector uses a statistical tool called copula to estimate the tail dependency between the two cost distributions.

1.4 Contributions

This thesis advances the field of safe and reliable robotic perception. We achieve this goal through three key contributions.

Outlier-Robust Estimation

In Chapter 2 we advance our understanding of the fundamental computational limits of robust estimation, and design outlier-robust estimation algorithms that:

(i) are *general-purpose* and usable across many estimation problems,

(ii) scale to large problems with thousands of variables,

(iii) are deterministic,

(iv) do not rely on an initial estimate,

(v) can potentially work without manual fine-tuning and be resilient to changes in the measurement noise statistics.

General Formulations and Inapproximability. Section 2.1 introduces two unifying formulations for outlier-robust estimation, *Generalized Maximum Consensus* (G-MC) and *Generalized Truncated Least Squares* (G-TLS). G-MC is a combinatorial formulation and generalizes the popular MC in Eq. (1.2) and the proposed *Minimally Trimmed Squares* (MTS) [41]; G-TLS is a continuous-optimization formulation and generalizes the truncated least squares used in M-estimation. We provide probabilistic interpretations for both formulations: G-MC solves a likelihood-constrained estimation problem, while G-TLS is a maximum likelihood estimator.

We also provide necessary and sufficient conditions for G-MC and G-TLS to return the same solution. We demonstrate that, in general, the conditions may not be satisfied, and G-TLS can reject more measurements than G-MC. Notwithstanding, we provide counterexamples showing that, while G-TLS can reject more measurements, it may lead to more accurate estimates.

Section 2.2 proves that G-MC and G-TLS are inapproximable even by *quasi-polynomial* algorithms, which are slower than polynomial.[5] The result holds true subject to a believed conjecture in complexity theory, $\mathsf{NP} \notin \mathsf{BPTIME}(m^{\mathsf{poly}\log m})$, which is similar to the widely known $\mathsf{NP} \neq \mathsf{P}$.[6] The result captures the hardness of G-MC and G-TLS for the first time: even in simplified cases where one knows the number of outliers to reject or that the residual error for the inliers is zero, it is still impossible to compute an approximate solution for G-MC and G-TLS within a prescribed approximation bound. This result strengthens recent inapproximability results for MC that only rule out polynomial time algorithms [13].

General-purpose and Minimally Tuned Algorithms. Our second contribution is to discuss two general-purpose algorithms. Section 2.3 presents a combinatorial approach, named *Adaptive Trimming* (ADAPT), which is suitable for G-MC. The algorithm works by iteratively removing measurements with large errors, but contrarily to a naive greedy algorithm, it revisits past decisions and checks for convergence over a sequence of iterations, leading to more accurate estimates. Section 2.4 introduces the *Graduated Non-Convexity* (GNC) approach of [43], which is a continuous-optimization approach to solve G-TLS. Both algorithms have linear runtime, are deterministic, and do not require an initial guess.

Section 2.5 presents outlier-robust estimation algorithms that are able to automatically adjust their parameters to perform robust estimation without prior knowledge of the inlier noise statistics. We present two algorithms, ADAPT-MinT and GNC-MinT, where MinT stands for "Minimally Tuned", that automatically adjust the noise bounds in ADAPT and GNC without the need for manual fine-tuning. This is in stark contrast with the techniques in the literature, whose correct operation relies on the knowledge of the maximum inlier noise (*cf.* parameter ϵ in Eq. (1.2) and Eq. (1.3)). We call these algorithms "Minimally Tuned" (rather than Parameter-Free) since they still

[5] An algorithm is called *quasi-polynomial*, if its runtime is $k_1 2^{(\log m)^{k_2}}$, where k_1 and k_2 are constants, and m is the algorithm's input size. Evidently, quasi-polynomial time algorithms are slower than polynomial, since $k_1\ 2^{(\log m)^{k_2}} \ > \ k_1\ 2^{k_2 \log m} \ = \ k_1 m^{k_1}$.

[6] $\mathsf{NP} \notin \mathsf{BPTIME}(m^{\mathsf{poly}\log m})$ means there exists no randomized algorithm that outputs solutions to problems in NP with probability 2/3, after running for $O(m^{(\log m)^k})$ time, for an input size m and a constant k [42].

involve parameters, which however only depend on the type of application, rather than the problem instance (*e.g.,* the same parameter values are used to solve any SLAM problem).

System-level Monitoring of Perception Systems

We then propose a methodology for runtime monitoring (in particular, fault detection and identification) of complex perception systems.

Diagnostic Graph. Our first contribution in this context is to formalize the problem (Section 3.1) and to present a framework (Section 3.2) to organize heterogeneous diagnostic tests of a perception system into a graphical model, the *diagnostic graph*. In particular, we present different mathematical models (including both deterministic and probabilistic models) to describe common diagnostic tests. Then, we introduce the concept of diagnostic graph, and extend it to capture asynchronous information over time (leading to *temporal* diagnostic graphs). Our framework adopts a black-box approach, in that it remains agnostic to the inner workings of the perception algorithms, and only focuses on collecting results from diagnostic tests that check the validity of their outputs.

Algorithms for Fault Detection and Identification. Our second contribution (Section 3.3) is a set of algorithms that use diagnostic graphs to perform fault detection and identification. For the deterministic case, we provide optimization-based methods that find the smallest set of faults that explain the test results. For the probabilistic case, we transform a diagnostic graph into a factor graph and perform inference to find the set of faulty modules. Finally, we propose a learning-based approach based on graph neural networks that learns to predict failures in a diagnostic graph.

Fundamental Limits. Our third contribution (Section 3.4) is to investigate fundamental limits and provide deterministic and probabilistic guarantees on the fault detection and identification results. In the deterministic case, we draw connections between perception system monitoring and the literature on diagnosability in multiprocessor systems, and in particular the PMC model [44]. This allows us to establish formal guarantees on the maximum number of faults that can be uniquely identified in a given perception system, leading to the notion of *diagnosability*.[7] In the probabilistic case, we develop Probably Approximate Correctly (PAC) bounds on the expected number of mistakes our runtime monitors will make.

Task-Aware Risk Estimation

We finally propose a novel approach to risk assessment in AV perception failures. The task-aware risk estimator we develop in this thesis compares the risk to the AV's motion plan in the perceived scenario with the one in the generated plausible scenarios. The risk posed to the AV's motion plan in both the scenes (perceived and plausible) is expressed as a probability distribution on a risk metric, *e.g.,* time-to-collision. We introduce the notion of *relative scenario risk* (RSR), which measures the probability that the plausible scene has a high risk when the perceived scene does not. To empirically estimate RSR, we employ the statistical tool called *copula* [45], which models tail dependencies between distributions, and we provide probably approximately correct (PAC) bounds on the RSR estimate. Finally, we provide a detection algorithm based on the RSR PAC bounds that, with high probability, triggers an alarm when faced with high-risk task-relevant failures. In particular

(i) We formalize the notion of *relative scenario risk* (RSR), which underlies our task-aware risk estimation;

(ii) We develop an algorithm to estimate RSR at runtime by leveraging the copula and also provide probabilistic guarantees on the correctness of our estimation;

(iii) Finally, we demonstrate the efficacy of our framework by comparing our method with prior approaches on a dataset of 100 realistic perception failure scenarios created in NuPlan [46].

[7] As discussed in Section 3.4, *diagnosability* is related to the level of redundancy within the system and provides a quantitative measure of robustness.

1.5 Structure of the Thesis

In Chapter 2 we present generalized formulations of the outlier-rejection problem, discuss fundamental limits, and propose a set of practical algorithms for outlier-robust estimation.

In Chapter 3 we formalize the problem of fault detection and identification and present a framework to organize heterogeneous diagnostic information of a perception system into a graphical model, the *diagnostic graph*. Then we present a set of algorithms that use diagnostic graphs to perform fault detection and identification.

In Chapter 4, we leverage the results from Chapter 3 and develop a *task-aware risk estimator* that assesses the increased risk to the AV's motion plan due to the perception failure. We formalize the notion of *relative scenario risk* (RSR), which underlies our task-aware risk estimation and develop an algorithm to estimate RSR at runtime by leveraging a tool from statistic known as copula, and also provide probabilistic guarantees on the correctness of our estimation.

We conclude the thesis in Chapter 5 with a summary of the contributions and a discussion of future work.

Chapter 2

Outlier-Robust Estimation

In the previous chapter we saw that spatial perception problems can be formulated as a nonlinear optimization problems. In this chapter we delve deeper. We first introduce two generalized formulations for outlier-robust estimation (Section 2.1). We then study their fundamental limits (Section 2.2) and present practical algorithms to solve them (Sections 2.3 and 2.4). We then present two additional variations of the algorithms that do not require any parameter tuning (Section 2.5). We conclude the chapter concrete examples in robotics and computer vision (Section 2.6) and a discussion of related work (Section 2.7).

2.1 Generalized MC and TLS Formulations

Sections Section 2.1.1 and Section 2.1.2 present Generalized Maximum Consensus (G-MC) and Generalized Truncated Least Squares (G-TLS). Section Section 2.1.3 gives G-MC's and G-TLS's probabilistic justification (Propositions Propositions 2 and 4). Section Section 2.1.4 provides conditions for G-MC and G-TLS to be equivalent (Theorem Theorem 7).

We use the following notation:

- $\boldsymbol{x}^\circ$ is the true value of $\boldsymbol{x}$ we want to estimate;
- $\mathcal{O}^\circ$ is the true set of outliers;

- $\boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}}, \boldsymbol{x}) \triangleq [r(\boldsymbol{y}_i, \boldsymbol{x})]_{i \in \mathcal{I}}$, for any $\mathcal{I} \subseteq \mathcal{M}$; *i.e.,* $\boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}}, \boldsymbol{x})$ is the vector of residuals for the measurements $i \in \mathcal{I}$.

2.1.1 Generalized Maximum Consensus (G-MC)

We present a generalized maximum consensus formulation.

Problem 1 (Generalized Maximum Consensus (G-MC))**.** *Find an estimate* $\boldsymbol{x}$ *by solving the combinatorial problem*

$$\min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M}}} \quad |\mathcal{O}| \quad \text{s.t.} \quad \| \, \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}}, \boldsymbol{x}) \, \|_{\ell} \leq \tau, \tag{G-MC}$$

where $\tau \geq 0$ *is a* given *inlier threshold, and* $\| \cdot \|_{\ell}$ *denotes a generic vector norm* (*in this thesis,* $\ell \in \{2, \infty\}$).

Since the true number of outliers is unknown, G-MC rejects the least amount of measurements such that the remaining ones appear as inliers. In Eq. (G-MC), $\mathcal{O}$ is the set of measurements classified as outliers; correspondingly, $\mathcal{M} \setminus \mathcal{O}$ is the set of inliers. A choice of inliers $\mathcal{M} \setminus \mathcal{O}$ is feasible only if there exists an $\boldsymbol{x}$ such that the cumulative residual error $\| \, \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}}, \boldsymbol{x}) \, \|_{\ell}$ satisfies the inlier threshold τ (enforced by the constraint).

G-MC generalizes existing combinatorial outlier-rejection formulations. In particular, depending on the choice of ℓ and τ in Eq. (G-MC), G-MC is equivalent to Maximum Consensus (MC) or Minimally Trimmed Squares (MTS):

- **MC as G-MC.** If $\ell = +\infty$ and $\tau = \epsilon$ in Eq. (G-MC), then G-MC is equivalent to MC (Eq. (1.2)), since $\| \, \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}}, \boldsymbol{x}) \, \|_{\infty} \leq \epsilon^2$ implies $r(\boldsymbol{y}_i, \boldsymbol{x}) \leq \epsilon$, $\forall \, i \in \mathcal{M} \setminus \mathcal{O}$.

- **MTS as G-MC.** If $\ell = 2$ in Eq. (G-MC), then G-MC is equivalent to the MTS [41], [47] formulation

$$\min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M}}} \quad |\mathcal{O}| \quad \text{s.t.} \quad \textstyle\sum_{i \, \in \, \mathcal{M} \setminus \mathcal{O}} \, r^2(\boldsymbol{y}_i, \boldsymbol{x}) \leq \tau^2, \tag{2.1}$$

since $\| \, \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}}, \boldsymbol{x}) \, \|_2^2 = \sum_{i \, \in \, \mathcal{M} \setminus \mathcal{O}} \, r^2(\boldsymbol{y}_i, \boldsymbol{x})$.

2.1.2 Generalized Truncated Least Squares (G-TLS)

We present a second formulation that generalizes truncated least squares in M-estimation [48], [49].

Problem 2 (Generalized Truncated Least Squares (G-TLS)). *Find an estimate* $\boldsymbol{x}$ *by solving the program*

$$\min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M}}} \quad \nu_\ell(\mathcal{O}) \| \ \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}}, \boldsymbol{x}) \ \|_\ell^2 + \epsilon^2 |\mathcal{O}|, \tag{G-TLS}$$

where $\| \cdot \|_\ell$ *denotes a generic vector norm* (*in this thesis,* $\ell \in \{2, \infty\}$)*, and* $\nu_\ell(\mathcal{O}), \epsilon > 0$ *are* given *penalty coefficients; in particular,* $\nu_2(\mathcal{O}) = 1$ *and* $\nu_\infty(\mathcal{O}) = |\mathcal{M} \setminus \mathcal{O}|$.

G-TLS looks for an outlier-robust estimate $\boldsymbol{x}$ by separating the measurements into inliers and outliers such that the former are penalized with their weighted cumulative residual error $\nu_\ell(\mathcal{O}) \| \ \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}}, \boldsymbol{x}) \ \|_\ell^2$, and the latter with their weighted cardinality $\epsilon^2 |\mathcal{O}|$. For appropriate choices of ϵ, G-TLS reduces to Truncated Least Squares (TLS) or standard least squares (LS):

- **TLS as G-TLS.** If $\ell = 2$, then G-TLS becomes

  $$\min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M}}} \quad \sum_{i \in \mathcal{M} \setminus \mathcal{O}} r^2(\boldsymbol{y}_i, \boldsymbol{x}) \ + \ \sum_{i \in \mathcal{O}} \epsilon^2, \tag{2.2}$$

  which is equivalent to the TLS formulation [2], [48], [49], commonly written using auxiliary binary variables w_i as

  $$\min_{\boldsymbol{x} \in \mathcal{X}} \sum_{i \in \mathcal{M}} \min_{w_i \in \{0,1\}} \left[w_i \ r^2(\boldsymbol{y}_i, \boldsymbol{x}) + (1 - w_i) \ \epsilon^2 \right]. \tag{TLS}$$

- **LS as G-TLS.** If $\ell = 2$ and $\epsilon = +\infty$, then, G-TLS becomes the least squares formulation in Eq. (1.1).

2.1.3 Probabilistic Justification of G-MC and G-TLS

We provide a probabilistic justification for the G-MC and G-TLS formulations, under the standard assumption of independent noise across the measurements.

Assumption 1 (Independent Noise)**.** *If $i \neq j$, for any $i, j \in \mathcal{M}$, then $r(\boldsymbol{y}_i, \boldsymbol{x})$ and $r(\boldsymbol{y}_j, \boldsymbol{x})$ are independent random variables.*

The results below provide a probabilistic grounding for two G-MC's instances, Maximum Consensus (MC) and Minimally Trimmed Squares (MTS), via likelihood estimation.

Proposition 2 (Uniform Inlier Distribution Leads to MC)**.** *If $r(\boldsymbol{y}_i, \boldsymbol{x}^\circ)$ is uniformly distributed with support $[0, \epsilon)$ for any $i \in \mathcal{M} \setminus \mathcal{O}^\circ$, then* MC *in Eq.* (1.2) *is equivalent to*

$$\min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M}}} \ |\mathcal{O}| \quad \text{s.t.} \prod_{i \in \mathcal{M} \setminus \mathcal{O}} u(r(\boldsymbol{y}_i, \boldsymbol{x}), \epsilon) \ > 0, \tag{2.3}$$

where the inequality is strict, and $u(r, \epsilon)$ is the probability density function of the uniform distribution with support $[0, \epsilon)$.

The optimization in Eq. (2.3) is a *likelihood-constrained* estimation: Eq. (2.3) finds an $\boldsymbol{x}$ such that the joint likelihood of the inliers is greater than zero.

Proposition 3 (Normal Inlier Distribution Leads to MTS)**.** *If $r(\boldsymbol{y}_i, \boldsymbol{x}^\circ)$ follows a Normal distribution for any $i \in \mathcal{M} \setminus \mathcal{O}^\circ$, then* MTS *in Eq.* (2.1) *is equivalent to*

$$\min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M}}} \ |\mathcal{O}| \quad \text{s.t.} \prod_{i \in \mathcal{M} \setminus \mathcal{O}} g(r(\boldsymbol{y}_i, \boldsymbol{x})) \geq \frac{e^{-\frac{\tau^2}{2}}}{(\pi/2)^{\frac{|\mathcal{M} \setminus \mathcal{O}|}{2}}}, \tag{2.4}$$

where $g(r) \triangleq \sqrt{2/\pi}\ \exp(-r^2/2)$ is the density of a Normal distribution constrained to the non-negative axis ($r \geq 0$).

Proposition Proposition 3 implies that MTS is equivalent to a likelihood-constrained estimation, where the inliers follow a Normal distribution (in contrast to Proposition Proposition 2 where the inliers are uniformly distributed).

Similarly, we show that an instance of G-TLS, Truncated Least Squares (TLS), can be interpreted as a maximum likelihood estimator. Particularly, if the number of outliers is known, we show TLS selects a set of inliers and a maximum likelihood estimate assuming the inliers are Normally distributed (Proposition Proposition 4); and if the number of outliers is unknown, we provide a broader characterization by connecting TLS to a max-mixture of Normal and uniform distributions (Proposition Proposition 5).

Proposition 4 (Normal Distribution and Known Number of Outliers Lead to TLS)**.** *Assume $r(\boldsymbol{y}_i, \boldsymbol{x}^\circ) < \epsilon$ for any $i \in \mathcal{M} \setminus \mathcal{O}^\circ$ and $|\mathcal{O}^\circ|$ is known. If $r(\boldsymbol{y}_i, \boldsymbol{x}^\circ)$ is Normally distributed for each $i \in \mathcal{M} \setminus \mathcal{O}$, then* TLS *is equivalent to the cardinality-constrained maximum likelihood estimator*

$$\max_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M},\, |\mathcal{O}| = |\mathcal{O}^\circ|}} \prod_{i \in \mathcal{M} \setminus \mathcal{O}} g(r(\boldsymbol{y}_i, \boldsymbol{x})). \tag{2.5}$$

Proposition 5 (Normal with Uniform Tails Leads to TLS)**.** *For any $i \in \mathcal{M}$, assume (i) $r(\boldsymbol{y}_i, \boldsymbol{x}^\circ) \leq \alpha$ for some number α, and (ii) $r(\boldsymbol{y}_i, \boldsymbol{x}^\circ)$ follows a* modified *Normal distribution $\hat{g}(r)$ where the tail of the Normal distribution for $r \geq \epsilon$ is replaced with a uniform distribution with support $[\epsilon, \alpha]$; particularly,*

$$\hat{g}(r) = \frac{1}{\beta} \begin{cases} g(r), & r < \epsilon; \\ g(\epsilon), & r \in [\epsilon, \alpha]; \\ 0, & \textit{otherwise}, \end{cases} \tag{2.6}$$

where β is a normalization factor (that depends on α) such that $\hat{g}(\cdot)$ is a valid probability density ($\int_0^\alpha \hat{g}(r)\, dr = 1$). Then, TLS *is equivalent to the maximum likelihood estimator*

$$\max_{\boldsymbol{x} \in \mathcal{X}} \prod_{i \in \mathcal{M}} \hat{g}(r(\boldsymbol{y}_i, \boldsymbol{x})). \tag{2.7}$$

The interested reader can find an alternative probabilistic interpretation of TLS in Appendix B, where TLS is shown to minimize the probability that an estimate becomes inaccurate when measurements are misclassified as inliers instead of outliers, and vice versa. We describe this probability with a product of Weibull distributions.

2.1.4 Relationship Between G-MC and G-TLS

Theorem 6 (G-MC = G-TLS when $\ell = +\infty$)**.** *Choose* $\| \cdot \|_\ell$ *to be the infinity norm in* G-MC *and* G-TLS*, and* $\tau = \epsilon$ *in* G-MC*. Also, assume* G-MC *has an optimal solution* $(\boldsymbol{x}_{\mathsf{G-MC}}, \mathcal{O}_{\mathsf{G-MC}})$ *such that* $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{G-MC}}}, \boldsymbol{x}_{\mathsf{G-MC}}) \|_\infty < \epsilon$ (i.e., G-MC*'s inequality constraint is strict at an optimal solution*)*. Then,* G-MC *and* G-TLS *compute the same set of outliers.*

The inequality $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{G-MC}}}, \boldsymbol{x}_{\mathsf{G-MC}}) \|_\infty \leq \epsilon$ is strict with probability 1 when the measurements are random. Hence, G-MC = G-TLS with probability 1 when $\ell = +\infty$, and, thus, we henceforth focus only on the TLS instance of G-TLS ($\ell = 2$).

Theorem 7 (G-MC $\neq$ G-TLS when $\ell = 2$)**.** *Denote by:*

- $(\boldsymbol{x}_{\mathsf{MTS}}, \mathcal{O}_{\mathsf{MTS}})$ *an optimal solution to* MTS *(*G-MC*'s instantiation for* $\ell = 2$*);*

- $(\boldsymbol{x}_{\mathsf{TLS}}, \mathcal{O}_{\mathsf{TLS}})$ *an optimal solution to* TLS *(*G-TLS*'s instantiation for* $\ell = 2$ *and* $\nu_\ell(\mathcal{O}) = 1$*);*

- $r^2_{\mathsf{TLS}}(\epsilon)$ *the error of the measurements classified as inliers at* $(\boldsymbol{x}_{\mathsf{TLS}}, \mathcal{O}_{\mathsf{TLS}})$*:* $r^2_{\mathsf{TLS}}(\epsilon) \triangleq \| r(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{TLS}}}, \boldsymbol{x}_{\mathsf{TLS}}) \|^2_\ell$*;*

- $f_{\mathsf{TLS}}(\epsilon)$ *the value of* TLS*:* $f_{\mathsf{TLS}}(\epsilon) \triangleq r^2_{\mathsf{TLS}}(\epsilon) + \epsilon^2 |\mathcal{O}_{\mathsf{TLS}}|$.

Then, for any $\epsilon > 0$*,*

- *if* $\tau = r_{\mathsf{TLS}}(\epsilon)$*, then* $|\mathcal{O}_{\mathsf{TLS}}| = |\mathcal{O}_{\mathsf{MTS}}|$*, and, in particular,* $(\boldsymbol{x}_{\mathsf{TLS}}, \mathcal{O}_{\mathsf{TLS}})$ *is also a solution to* MTS*;*

- *if* $\tau > r_{\mathsf{TLS}}(\epsilon)$*, then* $|\mathcal{O}_{\mathsf{TLS}}| \geq |\mathcal{O}_{\mathsf{MTS}}|$*;*

- *if* $\tau < r_{\mathsf{TLS}}(\epsilon)$*, then* $|\mathcal{O}_{\mathsf{TLS}}| < |\mathcal{O}_{\mathsf{MTS}}|$*, and* MTS *and* TLS *compute different sets of outliers.*

Example Example 8 below elucidates the result in Theorem Theorem 7 by considering instantiations of MTS and TLS in a toy example. The example shows that although TLS may reject more measurements than MTS, TLS can lead to more accurate estimates of $\boldsymbol{x}^\circ$ since it tends to reject "biased" measurements.

Example 8 (Sometimes, Less is More)**.** *Consider an estimation problem where (i) the variable to be estimated is a scalar x with true value $x^\circ = 0$, (ii) three measurements are available, the inliers $y_1 = y_2 = 0$, and the outlier $y_3 = 4$, and (iii) the measurement model is $y_i = x + n_i$, for all $i = 1, 2, 3$, where n_i is zero-mean and unit-variance additive Gaussian noise. Also, fix $\epsilon = 2.58$ in* TLS *such that $|n_i| \leq \epsilon$ with probability $\simeq 0.99$, and, correspondingly, fix $\tau = 11.35$ in* MTS *such that $n_1^2 + n_2^2 + n_3^2 \leq \tau$ with probability $\simeq 0.99$.*[1] *Evidently, at $x^\circ = 0$, $r(y_1, x^\circ) = r(y_2, x^\circ) = 0$ and $r(y_3, x^\circ) = 4$.*

In this toy problem, MTS *returns an incorrect estimate:* MTS *classifies all measurements as inliers for $x = 4/3$, since then $r^2(y_1, x) + r^2(y_1, x) + r^2(y_3, x)$ is minimized and is equal to $32/3 \simeq 10.67$, which is less than τ.*[2]

On the other hand, TLS *rejects more measurements than* MTS *but finds the correct estimate:* TLS *returns $x = 0$, classifying the third measurement as an outlier.*

A comparison of TLS with MC is given in [2, Appendix C].

2.2 Inapproximability of G-MC and G-TLS

This section shows that G-MC and G-TLS are computationally hard to solve, and in particular it is hard to even approximate their solution in quasi-polynomial time, in the worst case.

We start by recalling the $O(\cdot)$ and $\Omega(\cdot)$ notations from computational complexity theory [42].

[1] If n_1, n_2, n_3 are Gaussian random variables, each with mean 0 and variance 1, then (i) $\mathbb{P}(|n_i| \leq 2.58) \simeq 0.99506$ for all $i = 1, 2, 3$ [50], where $\mathbb{P}(\cdot)$ denotes probability; also, (ii) $n_1^2 + n_2^2 + n_3^2$ follows a χ^2 distribution with 3 degrees of freedom and $\mathbb{P}(n_1^2 + n_2^2 + n_3^2 \leq 11.35) \simeq 0.99$ [51].

[2] MC (Eq. (1.2)) also leads to a wrong estimate, selecting all measurements as inliers (*e.g.,* $x = 2$ makes all measurements to have residual smaller than ϵ).

Definition 9 (O Notation)**.** *Consider two functions* $h : \mathbb{N} \to \mathbb{R}$ *and* $g : \mathbb{N} \to \mathbb{R}$*. Then,* $h(m) = O(g(m))$ *means there exists a constant* $k > 0$ *such that* $h(m) \leq kg(m)$ *for large enough* m*.*

Definition 10 (Ω Notation)**.** *Consider* $h : \mathbb{N} \to \mathbb{R}$ *and* $g : \mathbb{N} \to \mathbb{R}$*. Then,* $h(m) = \Omega(g(m))$ *means there exists a constant* $k > 0$ *such that* $h(m) \geq kg(m)$ *for large enough* m*.*

Definition 11 ((λ, p)-Approximability)**.** *Consider* $\lambda \geq 1$*, and* $p \geq 0$*.* G-MC *is* (λ, p)-approximable *if there exists an algorithm finding a sub-optimal solution* $(\boldsymbol{x}, \mathcal{O})$ *for* G-MC *such that* $|\mathcal{O}| \leq \lambda |\mathcal{O}_{\mathsf{G-MC}}|$ *and* $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}}, \boldsymbol{x}) \|_\ell^2 \leq \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}_{\mathsf{G-MC}}}, \boldsymbol{x}_{\mathsf{G-MC}}) \|_\ell^2 + p$*, where* $(\boldsymbol{x}_{\mathsf{G-MC}}, \mathcal{O}_{\mathsf{G-MC}})$ *is an optimal solution for* G-MC*.*

Similarly, G-TLS *is* (λ, p)-approximable *if there exists an algorithm finding a sub-optimal solution* $(\boldsymbol{x}, \mathcal{O})$ *for* G-TLS *such that* $|\mathcal{O}| \leq \lambda |\mathcal{O}_{\mathsf{G-TLS}}|$ *and* $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}}, \boldsymbol{x}) \|_\ell^2 \leq \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}_{\mathsf{G-TLS}}}, \boldsymbol{x}_{\mathsf{G-TLS}}) \|_\ell^2 + p$*, where* $(\boldsymbol{x}_{\mathsf{G-TLS}}, \mathcal{O}_{\mathsf{G-TLS}})$ *denotes an optimal solution to* G-TLS*.*

Definition 11 bounds the sub-optimality of an approximate solution to G-MC or G-TLS: if $(\boldsymbol{x}, \mathcal{O})$ is an (λ, p)-approximate solution, then $\mathcal{O}$ rejects up to a multiplicative factor λ more outliers than the optimal set of outliers; and $(\boldsymbol{x}, \mathcal{O})$ attains an inlier residual error up to an additive factor p more than the residual error attained at the optimal solution.

Theorem 12 (Inapproximability of G-MC and G-TLS)**.** *For any* $\delta \in (0, 1)$*, unless we have* $\mathsf{NP} \in \mathsf{BPTIME}(|\mathcal{M}|^{\mathsf{poly}\log|\mathcal{M}|})$*, there exist a* $\lambda(|\mathcal{M}|) = 2^{\Omega(\log^{1-\delta}|\mathcal{M}|)}$*, a polynomial* $p(|\mathcal{M}|)$*, and instances of* G-MC *such that no quasi-polynomial algorithm makes the instances* $(\lambda(|\mathcal{M}|), p(|\mathcal{M}|))$*-approximable. The result holds true even if the algorithm knows*

(i) $|\mathcal{O}_{\mathsf{G-MC}}|$*,*

(ii) that the optimal solution is such that $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}_{\mathsf{G-MC}}}, \boldsymbol{x}_{\mathsf{G-MC}}) \|_\ell^2 = 0$*.*

Similarly, the result holds true for G-TLS*, even if the algorithm knows (i)* $|\mathcal{O}_{\mathsf{G-TLS}}|$*, and (ii) that the optimal solution is such that* $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \setminus \mathcal{O}_{\mathsf{G-TLS}}}, \boldsymbol{x}_{\mathsf{G-TLS}}) \|_\ell^2 = 0$*.*

The theorem captures the extreme hardness of G-MC and G-TLS: in the worst case, any quasi-polynomial algorithm for G-MC and G-TLS cannot approximate the solution to G-MC and G-MC within an (λ, p)-approximation. This holds true even if the algorithm is informed with the optimal number of outliers to reject, or knows a priori that the optimal residual error is zero. The quality of the approximation depends on the parameter λ and p in Theorem 12, which are both polynomials. In particular, it can be seen that λ (*cf.* Definition 11) grows with the number of measurements, since $\lambda = \lambda(2^{\Omega(\log^{1-\delta}|\mathcal{M}|)})$ is proportional to $|\mathcal{M}|$ when δ is close to 0.

We remark that, since both λ and p in Theorem 12 depend on the number of measurements, $|\mathcal{M}|$, the theorem implies there is no quasi-polynomial time algorithm achieving constant sub-optimality bound for G-MC and G-TLS. As such, the theorem strengthens recent inapproximability results for MC that focus, instead, on polynomial-time algorithms only [13].

2.3 Adaptive Trimming (ADAPT) Algorithm

We present ADAPT, a general-purpose, deterministic, and linear time algorithm for G-MC, that requires no initial guess. We first describe a simple greedy algorithm in Section 2.3.1, to build intuition, and then introduce ADAPT in Section 2.3.2.

2.3.1 Gentle Start: Greedy Outlier Rejection

We start by describing a simple greedy algorithm for G-MC, to build intuition about ADAPT. The algorithm starts by solving a least squares problem akin to Eq. (1.1) over the entire set of measurements, and, at each iteration, it rejects the measurement with the largest residual. The algorithm stops once the condition $\|\boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}}, \boldsymbol{x})\|_\ell \leq \tau$ in Eq. (G-MC) is satisfied.

Although the described greedy algorithm is appealing for its simplicity and linear runtime,[3,4]

(i) it cannot correct past mistakes (once a measurement is rejected, it is never reconsidered),

(ii) the algorithm terminates once the threshold τ is met, without, however, assessing if all outliers have been rejected, *e.g.,* by checking whether $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}}, \boldsymbol{x}) \|_2$ has converged.

Indeed, the greedy algorithm, being an approximation procedure, may satisfy the threshold τ by simply over-rejecting measurements, instead of rejecting all outliers. Therefore, it may be the case $\mathcal{M}\setminus\mathcal{O}$ still contains outliers whose removal would largely reduce $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}}, \boldsymbol{x}) \|_2$. Instead, if $\mathcal{M} \setminus \mathcal{O}$ contains no outliers, $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}}, \boldsymbol{x}) \|_2$ would remain largely unchanged even if more measurements would to be removed from $\mathcal{M} \setminus \mathcal{O}$. Because of these, the algorithm can exhibit deteriorated performance; *cf.* SLAM experiments in Section 2.6.3. ADAPT improves upon the greedy algorithm by addressing the greedy's weaknesses described above.

2.3.2 Beyond Greedy: ADAPT Algorithm

We present the *Adaptive Trimming* (ADAPT) algorithm to solve the G-MC formulation in Problem 1. The algorithm processes all measurements at each iteration, and *trims* (rejects) measurements violating an inlier threshold (the threshold is set at each iteration and decreases iteration after iteration). The algorithm is *adaptive* in that it dynamically decides the threshold at each iteration. ADAPT is not greedy in that it can correct previous mistakes: a measurement that has been deemed to be an outlier at an iteration can be re-included in the set of inliers at subsequent iterations, and, similarly, a measurement that has been deemed to be an inlier at an iteration can

[3] In the literature, there exists an alternative greedy algorithm [52] that, at each iteration, tests the impact of rejecting each measurement (by solving multiple estimation problems), and then rejects only the measurement that induces the largest decrease in the objective function. We do not consider such a variant since it has quadratic complexity in the number of measurements, and does not scale to the problems we consider in Section 2.6.

[4] At each iteration, the described greedy algorithm rejects one measurement, and, as a result, has linear runtime in the number of measurements.

Algorithm 1: Adaptive Trimming (ADAPT).

```
   Input: Measurements $\boldsymbol{y}_i$, $\forall i \in \mathcal{M}$; thresholds $\tau, \theta$;
          $\mathit{MaxIterations}, \mathit{SamplesToConverg} > 0$; $\mathit{ThrDiscount} \in (0, 1)$.
   Output: Estimate of $\boldsymbol{x}^{\circ}$ and corresponding inliers.
 1 $\mathcal{I}^{(0)} = \mathcal{M}$;   $\boldsymbol{x}^{(0)} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} l\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(0)}}, \boldsymbol{x}) \|_2^2$;
 2 $\varepsilon^{(0)} = \mathit{ThrDiscount} \cdot \max_{i \,\in\, \mathcal{I}^{(0)}} \; r(\boldsymbol{y}_i, \boldsymbol{x}^{(0)})$;   $j = 0$;
 3 for $t = 1, \ldots, \mathit{MaxIterations}$ do
 4     $\mathcal{I}^{(t)} = \{i \in \mathcal{M} \text{ s.t. } r(\boldsymbol{y}_i, \boldsymbol{x}^{(t-1)}) \leq \varepsilon^{(t-1)}\}$;
 5     $\boldsymbol{x}^{(t)} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t)}}, \boldsymbol{x}) \|_2^2$;
 6     if $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t)}}, \boldsymbol{x}) \|_{\ell} < \tau$
 7     and $|\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t)}}, \boldsymbol{x}) \|_2^2 - \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t-1)}}, \boldsymbol{x}) \|_2^2| < \theta(|\mathcal{I}^{(t-1)}|, |\mathcal{I}^{(t)}|)$ then
 8         $j++$;
 9     else
10         $j = 0$;
11     end
12     if $j = \mathit{SamplesToConverg}$ then
13         break;
14     end
15     $\varepsilon^{(t)} = \mathit{ThrDiscount} \cdot \max_{i \,\in\, \mathcal{I}^{(t)}} \; r(\boldsymbol{y}_i, \boldsymbol{x}^{(t)})$;
16 end
17 return $(\boldsymbol{x}^{(t)}, \mathcal{I}^{(t)})$.
```

be (re-)included in the set of outliers at subsequent iterations. ADAPT is not greedy also in that it assesses whether all outliers have been rejected by checking whether $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}}, \boldsymbol{x}) \|_2$ has converged. Finally, ADAPT can reject multiple measurements at each iteration, whereas greedy rejects one. Its pseudo-code is given in Algorithm 1.

Initialization. ADAPT's Line 1 initializes the putative set of inliers to $\mathcal{I}^{(0)} = \mathcal{M}$ (all measurements); at the subsequent iterations $t = 1, 2, \ldots$, the set $\mathcal{I}^{(t)}$ will include only the measurements classified as inliers at t. Given $\mathcal{I}^{(0)}$, ADAPT sets $\boldsymbol{x}^{(0)} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(0)}}, \boldsymbol{x}) \|_2^2$, *i.e.,* $\boldsymbol{x}^{(0)}$ is the estimate assuming all measurements are inliers; the nonlinear least squares problem can be minimized using non-minimal solvers, see [43]. Using $\boldsymbol{x}^{(0)}$, Line 2 sets the initial inlier threshold $\varepsilon^{(0)}$ equal to $\mathit{ThrDiscount} \cdot \max_{i \,\in\, \mathcal{I}^{(0)}} \; r(\boldsymbol{y}_i, \boldsymbol{x}^{(0)})$, *i.e.,* a multiplicative factor *ThrDiscount* less than the maximum residual at $\boldsymbol{x}^{(0)}$. That way, at least one measurement will be classified as an outlier at the next iteration. In this thesis, we always set $\mathit{ThrDiscount} = 0.99$.

Main Loop. After the initialization, ADAPT starts the main outlier rejection loop (Line 3). We describe each step below.

a) Inlier Set Update. At iteration t, given $\varepsilon^{(t-1)}$, Line 4 updates the set of inliers $\mathcal{I}^{(t)}$ to contain measurements with residual smaller than $\varepsilon^{(t-1)}$. Since ADAPT checks *all* measurements in $\mathcal{M}$, $\mathcal{I}^{(t)}$ may *contain* measurements that *were not* in $\mathcal{I}^{(t-1)}$, and may *not contain* measurements that *were* in $\mathcal{I}^{(t-1)}$. This allows ADAPT to re-include measurements that were incorrectly rejected as outliers at previous iterations, and to reject measurements that were incorrectly classified as inliers. Notably, $\mathcal{I}^{(t)}$ depends on the history $\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(t-1)}$, since $\varepsilon^{(t-1)}$ depends on $\mathcal{I}^{(t-1)}$ (cf. line 15 of Algorithm 1), which in turn depends on $\varepsilon^{(t-2)}$, and so forth. Therefore, as ADAPT iterates, a sequence $(\mathcal{I}^{(1)}, \epsilon^{(1)}), \ldots, (\mathcal{I}^{(t)}, \epsilon^{(t)}), \ldots$ is generated, and, *ideally*, even if measurements are misclassified at early iterations, eventually all are classified correctly.

b) Variable Update. Given $\mathcal{I}^{(t)}$, a new estimate $\boldsymbol{x}^{(t)}$ is computed in Line 5. Line 5's minimization is a nonlinear least squares problem that is solved using non-minimal solvers [43].

c) Inlier Threshold Update. If the current estimate is infeasible for G-MC and/or convergence of $\| \ \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}}, \boldsymbol{x}) \ \|_2$'s value has not been observed for at least *SamplesToConverg* consecutive iterations (*i.e.,* the "if" conditions in lines 6-7 and Line 12 are not satisfied), ADAPT updates $\varepsilon^{(t)}$ (Line 15) and moves to the next iteration. Similarly to Line 2, Line 15 updates the threshold by applying a multiplicative factor *ThrDiscount* < 1 to the maximum residual at the current iteration; this ensures that at least 1 measurement is rejected at the next iteration.

Termination. ADAPT terminates when:

- a maximum number of iterations is reached (*cf.* "for" loop in Line 3; In this thesis, we set *MaxIterations* $= 1000$);

- a feasible estimate for G-MC is found and for *SamplesToConverg* iterations $\| \ \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}}, \boldsymbol{x}) \ \|_2$ changes by at most θ (*cf.* "if" conditions in lines 6-7 and Line 12). In this thesis, *SamplesToConverg* $= 3$. A probabilistically-grounded method to chose θ is described in Section 2.6.

Upon termination, ADAPT returns the current estimate $\boldsymbol{x}^{(t)}$ and inlier set $\mathcal{I}^{(t)}$ (Line 17). The following remark ensures that ADAPT terminates after at most $|\mathcal{M}|$ iterations.

Remark 13 (Linear Runtime)**.** ADAPT*'s policy to update $\epsilon^{(t)}$ (Line 15) implies that $|\mathcal{I}^{(t)}| \leq |\mathcal{I}^{(t-1)}|-1$, hence* ADAPT *terminates in at most $|\mathcal{M}|$ (number of measurements) iterations.*

Remark 14 (vs. RANSAC)**.** RANSAC *is a* randomized algorithm *for* G-MC*, whereas* ADAPT *is* deterministic*.* RANSAC *maintains only a* "local view' *of the measurement set $\mathcal{M}$, building an inlier set by sampling a* minimal *set of measurements; instead,* ADAPT *looks at all measurements in $\mathcal{M}$ to pick an inlier set.* RANSAC *assumes the availability of* minimal solvers*, while* ADAPT *assumes the availability of* non-minimal solvers*.* RANSAC *is unsuitable for high-dimensional problems, since the number of iterations required to sample an outlier-free set increases exponentially in the dimension of the problem [16]; in contrast,* ADAPT *runs in* linear time*, terminating in at most as many iterations as the number of measurements.*

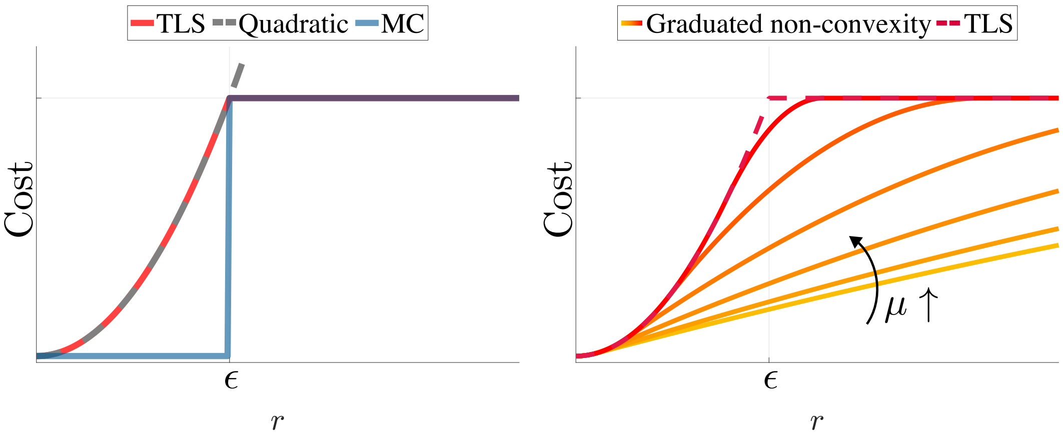
2.4 Graduated Non-convexity (GNC) Algorithm

We present the GNC algorithm to solve the TLS formulation. We show that —when considering TLS costs— the algorithm can be simply explained without invoking Black-Rangarajan duality. Moreover, we provide a local convergence result (Theorem 15), which enables simpler stopping conditions for the algorithm.

2.4.1 Preliminaries on Graduated Non-convexity

Before introducing the GNC algorithm we review the notion of graduated non-convexity [18], [43], [53], [54].

For convenience, we recall that our goal in this section is to solve the TLS problem



(a) TLS vs MC vs quadratic cost functions (b) Graduated Non-convexity

Figure 2-1: (a) TLS, quadratic, and MC cost functions, (b) graduated non-convexity with control parameter μ for TLS cost function.

(already introduced in Eq. (TLS)):

$$\min_{\boldsymbol{x} \in \mathcal{X}} \sum_{i \in \mathcal{M}} \min_{w_i \in \{0,1\}} \left[w_i \, r^2(\boldsymbol{y}_i, \boldsymbol{x}) + (1 - w_i) \, \epsilon^2 \right]. \tag{2.8}$$

Solving the minimization Eq. (2.8) is hard because the TLS objective function is highly non-convex in the residual errors r. Indeed, the i-th summand in Eq. (2.8), namely $\min_{w_i \in \{0,1\}} [w_i r^2(\boldsymbol{y}_i, \boldsymbol{x}) + (1-w_i)\epsilon^2]$, describes a truncated quadratic function, that is nonconvex as shown in Fig. 2-1(a).

Graduated non-convexity circumvents this non-convexity by using a *homotopy* (or *continuation*) method [54]. In particular, graduated non-convexity proposes to "soften" the non-convexity in TLS by replacing the cost with a surrogate function controlled by a parameter μ:

$$\min_{\boldsymbol{x} \in \mathcal{X}} \sum_{i \in \mathcal{M}} \min_{w_i \in [0,1]} \left[w_i \, r^2(\boldsymbol{y}_i, \boldsymbol{x}) + \frac{\mu(1 - w_i)}{\mu + w_i} \, \epsilon^2 \right], \tag{2.9}$$

where the "regularization" term $(1-w_i)\epsilon^2$ in Eq. (2.8) is replaced with ${}^{\mu(1-w_i)\epsilon^2}/_{(\mu+w_i)}$. The surrogate function in Eq. (2.9) is such that (i) for $\mu \to 0$, Eq. (2.9) becomes a convex optimization problem [43], and (ii) for $\mu \to +\infty$, the term ${}^{\mu(1-w_i)\epsilon^2}/_{(\mu+w_i)} \to (1-w_i)\epsilon^2$, *i.e.*, Eq. (2.9) retrieves the original TLS in Eq. (2.8). The family of surrogate

Algorithm 2: Graduated Non-Convexity for Truncated Least Squares (GNC-TLS) [43].

```
   Input: Measurements $\boldsymbol{y}_i$, $\forall i \in \mathcal{M}$; threshold $\epsilon \geq 0$; $MaxIterations > 0$;
          $MuUpdateFactor > 1$.
   Output: Estimate of $\boldsymbol{x}^{\circ}$ and corresponding inliers.
 1 $\mu^{(0)} = \frac{\epsilon^2}{2 \max_{i \,\in\, \mathcal{M}} \; r^2(\boldsymbol{y}_i, \boldsymbol{x}^{(0)}) - \epsilon^2}$;
 2 $\boldsymbol{w}^{(0)} = \mathbf{1}_{\mathcal{M}}$;  $\boldsymbol{x}^{(0)} = \text{VariableUpdate}(\boldsymbol{w}^{0})$; //eq. Eq. (2.10)
 3 for $t = 1, \ldots, MaxIterations$ do
 4     $\boldsymbol{w}^{(t)} = \text{WeightUpdate}(\boldsymbol{x}^{(t-1)}, \mu^{(t-1)}, \epsilon)$; //eq. Eq. (2.11)
 5     $\boldsymbol{x}^{(t)} = \text{VariableUpdate}(\boldsymbol{w}^{(t)})$; //Eq. (2.13)
 6     $\mu^{(t)} = MuUpdateFactor \cdot \mu^{(t-1)}$;
 7     if IsBinary($\boldsymbol{w}^{(t)}$) = true then break;
 8 end
 9 return ($\boldsymbol{x}^{(t)}$, supp($\boldsymbol{w}^{(t-1)}$)).
```

functions (parametrized by μ) is shown in Fig. 2-1(b).

Given the surrogate optimization problems in Eq. (2.9), graduated non-convexity starts by solving a convex approximation of the TLS problem (*i.e.,* for small μ) and then gradually increases the non-convexity (by increasing μ) till the original TLS cost is retrieved (*i.e.,* for large μ). The estimate at each iteration is used as initial guess for the subsequent iteration, to reduce the risk of convergence to local minima.

2.4.2 GNC-TLS Algorithm

The pseudo-code of GNC-TLS is given in Algorithm 2. Besides leveraging graduated nonconvexity, at each iteration, GNC-TLS minimizes the surrogate function in Eq. (2.9) by alternating a minimization with respect to $\boldsymbol{x}$ (with fixed w_i) to a minimization of the weights w_i (with fixed $\boldsymbol{x}$). Both minimizations can be solved efficiently, as described below.

Initialization. GNC-TLS's Line 1 initializes the parameter μ to a small number as suggested in [43]. Line 2 also initializes all weights to 1 (*i.e.,* $\boldsymbol{w}^{(0)} = \mathbf{1}_{\mathcal{M}}$, where $\mathbf{1}_{\mathcal{M}}$ is the vector of all ones with length equal to $|\mathcal{M}|$) and sets the initial $\boldsymbol{x}$ to be the solution of the least squares problem:

$$\boldsymbol{x}^{(0)} = \underset{\boldsymbol{x}\in\mathcal{X}}{\arg\min} \sum_{i\,\in\,\mathcal{M}} r^2(\boldsymbol{y}_i, \boldsymbol{x}). \tag{2.10}$$

which we denote in the algorithm as VariableUpdate($\boldsymbol{w}^0$).

Main Loop. After the initialization, `GNC-TLS` starts the main outlier rejection loop (Line 3). At iteration t, `GNC-TLS` minimizes the surrogate function in Eq. (2.9) by alternating a minimization over the weights (Line 4) and a minimization over the variable $\boldsymbol{x}$ (Line 5); then, `GNC-TLS` increases the amount of nonconvexity by increasing the parameter μ (Line 6). The details of these steps are given below.

a) Weight Update. At iteration t, `GNC-TLS` updates the weights $\boldsymbol{w}^{(t)}$ to minimize the surrogate function in Eq. (2.9) while keeping fixed $\boldsymbol{x}^{(t-1)}$ and $\mu^{(t-1)}$ (Line 4):

$$\boldsymbol{w}^{(t)} \in \underset{w_i\,\in\,[0,1]}{\arg\min} \sum_{i\,\in\,\mathcal{M}} \left[w_i\; r_i^{(t)} + \frac{\mu^{(t-1)}(1-w_i)}{\mu^{(t-1)}+w_i}\;\epsilon^2 \right], \tag{2.11}$$

where $r_i^{(t)} \triangleq r(\boldsymbol{y}_i, \boldsymbol{x}^{(t)})$; Eq. (2.11) splits into $|\mathcal{M}|$ scalar problems [43] and admits the following closed-form solution:

$$w_i^{(t)} = \begin{cases} 1, & r_i^{(t)} < \epsilon\,\sqrt{\frac{\mu^{(t-1)}}{\mu^{(t-1)}+1}} \\ 0, & r_i^{(t)} > \epsilon\,\sqrt{\frac{\mu^{(t-1)}+1}{\mu^{(t-1)}}} \\ \frac{\epsilon\sqrt{\mu^{(t-1)}(\mu^{(t-1)}+1)}}{r_i^{(t)}} - \mu^{(t-1)}, & \text{otherwise.} \end{cases} \tag{2.12}$$

b) Variable Update. Line 5 updates $\boldsymbol{x}^{(t)}$ by minimizing the surrogate function in Eq. (2.9) while keeping fixed $\boldsymbol{w}^{(t)}$:

$$\boldsymbol{x}^{(t)} \in \underset{\boldsymbol{x}\,\in\,\mathcal{X}}{\arg\min} \sum_{i\,\in\,\mathcal{M}} w_i^{(t)}\; r^2(\boldsymbol{y}_i, \boldsymbol{x}), \tag{2.13}$$

where we dropped the additional summand in Eq. (2.9), since it is independent of $\boldsymbol{x}$. The optimization problem in Eq. (2.13) is a weighted least squares problem (cf. Eq. (1.1)), and can be solved globally using certifiably optimal non-minimal solvers [43].

c) Increasing Non-convexity: $\boldsymbol{\mu}$ Update. At the end of each iteration, GNC-TLS increases μ by a multiplicative factor $MuUpdateFactor > 1$ (Line 6), getting one step closer to the original non-convex TLS cost function (*cf.* Fig. 2-1(b)). As in [43], we choose $MuUpdateFactor = 1.4$ in GNC.

Termination. GNC-TLS terminates when (i) the maximum number of iterations is reached (Line 3) —In this thesis, $MaxIterations = 1000$—, or (ii) the weight vector $\boldsymbol{w}^{(t)}$ becomes a binary vector (Line 7). The latter stopping condition is supported by the following theorem.

Theorem 15 ($\boldsymbol{w}^{(t)}$ Tends to a Binary Vector with Probability 1)**.** *If $t \to +\infty$, then $w_i^{(t)} \to w_i^{(\infty)}$, where, for all $i \in \mathcal{M}$,*

$$w_i^{(\infty)} = \begin{cases} 1, & r_i^{(\infty)} < \epsilon; \\ 0, & r_i^{(\infty)} > \epsilon; \\ 1/2, & r_i^{(\infty)} = \epsilon. \end{cases} \tag{2.14}$$

Moreover, since the measurements are affected by random noise, the case $r_i^{(\infty)} = \epsilon$ happens with zero probability.

Eq. (2.14) agrees with the TLS formulation in Eq. (2.8), since $w_i^{(\infty)} = 1$ only when $r_i^{(\infty)} < \epsilon$, *i.e.*, when measurement i is considered an inlier, while $w_i^{(\infty)} = 0$ otherwise.

2.5 Minimally Tuned ADAPT and GNC

We now present the minimally tuned versions of ADAPT and GNC, namely, ADAPT-MinT and GNC-MinT. In contrast to ADAPT and GNC, they do not require knowledge of a threshold to separate inliers from outliers (τ in ADAPT, ϵ in GNC).

2.5.1 ADAPT-MinT Algorithm

ADAPT-MinT is similar to ADAPT, but introduces a novel, inlier-threshold-free termination condition. In contrast to ADAPT, which terminates based on a given τ (which

Algorithm 3: Minimally tuned ADAPT (ADAPT-MinT).

```
Input: Measurements $\boldsymbol{y}_i$, $\forall i \in \mathcal{M}$; $MaxIterations > 0$; $ThrDiscount \in (0,1)$;
       $MinSamples$, $WindowSize$, $ConvergThr \geq 0$.
Output: Estimate of $\boldsymbol{x}^\circ$ and corresponding inliers.
$\mathcal{I}^{(0)} = \mathcal{M}$;   $\boldsymbol{x}^{(0)} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} l\| \, \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(0)}}, \boldsymbol{x}) \, \|_2^2$;
$\varepsilon^{(0)} = ThrDiscount \cdot \max_{i \, \in \, \mathcal{I}^{(0)}} \; r(\boldsymbol{y}_i, \boldsymbol{x}^{(0)})$;
$\delta^{(0)} = \text{ClustersSeparation}(\boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}}, \boldsymbol{x}^{(0)}))$;
for $t = 1, \ldots, MaxIterations$ do
    $\mathcal{I}^{(t)} = \{ i \;\in\; \mathcal{M} \;\text{ s.t. }\; r(\boldsymbol{y}_i, \boldsymbol{x}^{(t-1)}) \leq \varepsilon^{(t-1)} \}$;
    $\boldsymbol{x}^{(t)} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \; \| \, \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t)}}, \boldsymbol{x}) \, \|_2^2$;
    $\varepsilon^{(t)} = ThrDiscount \cdot \max_{i \, \in \, \mathcal{I}^{(t)}} \; r(\boldsymbol{y}_i, \boldsymbol{x}^{(t)})$;
    $\delta^{(t)} = 1/\delta^{(0)} \cdot \text{ClustersSeparation}(\boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}}, \boldsymbol{x}^{(t)}))$;
    $\sigma^{(t)} = \text{movstd}(\delta^{(t)}, \; WindowSize)$;
    if $t > MinSamples$ and $\sigma^{(t-MinSamples \, : \, t-1)} < ConvergThr$ then
        break
    end
end
return $(\boldsymbol{x}^{(t)}, \mathcal{I}^{(t-MinSamples)})$.
```

separates inliers from outliers) ADAPT-MinT (i) looks at the residuals of all measurements, given the current estimate $\boldsymbol{x}^{(t)}$, (ii) clusters them into two groups, a group of low-magnitude residuals —the "inliers" (left group in Fig. 2-2)— and a group of high-magnitude residuals —the "outliers" (right group in Fig. 2-2)— and (iii) terminates once the two groups "stabilize," in particular, when the distance δ between the centroids of two groups converges to a steady state. To cluster all residuals in $\mathcal{M}$ into two groups, and to compute their centroids and their in-between distance, ADAPT-MinT calls the subroutine ClustersSeparation presented in D.1 (Algorithm 6).

The pseudo-code of ADAPT-MinT is given in Algorithm 3.

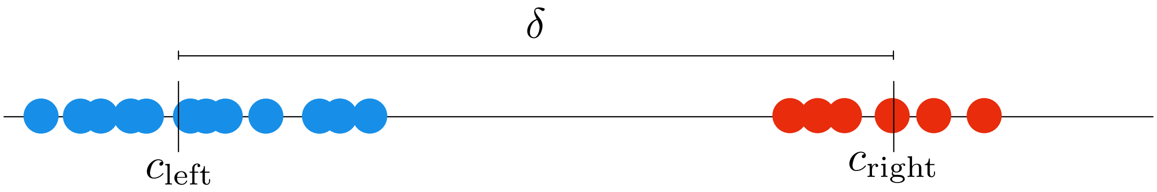


Figure 2-2: Two clusters of non-negative residuals: the low-magnitude ones (blue) are centered at c_{left}, the high-magnitude ones (red) at $c_{\text{right}} = c_{\text{left}} + \delta$.

Initialization. ADAPT-MinT's lines 1-2 are the same as ADAPT's, and initialize $\mathcal{I}^{(0)}$ and $\boldsymbol{x}^{(0)}$. Line 3 is new: given $\boldsymbol{x}^{(0)}$, it initializes $\delta^{(0)}$, *i.e.,* the distance between the inlier and outlier clusters at $\boldsymbol{x}^{(0)}$. At the subsequent iterations $t = 1, 2, \ldots$, the value $\delta^{(0)}$ is used as a normalization factor in the update of $\delta^{(t)}$ (Line 8, discussed below).

Inlier Set, Variable, and Inlier Threshold Update. Li- nes 5, 6, and 7 in ADAPT-MinT describe the same inlier set, variable, and inlier threshold updates used in ADAPT.

Inlier vs. Outlier Cluster Separation Update. ADAPT-MinT updates $\delta^{(t)}$ with the distance between the inlier and outlier clusters at the current $\boldsymbol{x}^{(t)}$, after normalizing it by $\delta^{(0)}$ (Line 8). The role of the normalization is discussed in Remark 16 below.

Termination. ADAPT-MinT terminates when (i) the maximum number of iterations is reached (*cf.* "for" loop in Line 4), or (ii) $\delta^{(t)}$ converges to a steady state value, indicating the inlier and outlier clusters have also converged to a steady state. Specifically, ADAPT-MinT declares convergence when for *MinSamples* consecutive iterations $\delta^{(t)}$'s moving standard deviation $\sigma^{(t)}$ is less than *ConvergThr* (Line 10). In more detail, $\sigma^{(t)}$ is the standard deviation of $\delta^{(t)}$ across the last *WindowSize* iterations and is computed in Line 9, where movstd is the corresponding MATLAB function. In this thesis, $\textit{WindowSize} = 3$, $\textit{MinSamples} = 5$ and $\textit{ConvergThr} = 10^{-4}$ always.

Remark 16 (Role of Normalization in ADAPT-MinT)**.** *The normalization by $\delta^{(0)}$ in Line 8 is necessary, since across different applications the residuals can differ by several orders of magnitude, and, as a result, the distance between the inlier and outlier clusters can differ by several orders of magnitude. The normalization reduces the impact of the magnitude of the residuals on the stopping conditions of* ADAPT-MinT.

Remark 17 (Tuning ADAPT's τ vs. Tuning ADAPT-MinT's *ConvergThr*)**.** *Tuning τ requires knowledge of the inlier threshold (or equivalently, the inlier noise), which varies not only across applications (*e.g., *mesh registration vs.* SLAM*) but also across problem instances within the same application (*e.g., *different* SLAM *datasets). In contrast, ConvergThr is fixed across instances of an application (for all applications in this the-*

sis, in particular, ConvergThr is the same), and its value can be set given a single dataset where the ground truth is known. In this sense, ADAPT-MinT *is* minimally tuned.

2.5.2 GNC-MinT Algorithm

GNC-MinT, in contrast to GNC-TLS, does not require knowledge of a suitable inlier threshold ϵ. Instead, GNC-MinT requires only an upper and lower bound for ϵ, denoted by *NoiseUpBnd* and *NoiseLowBnd* in the algorithm. GNC-MinT uses *NoiseUpBnd* as an initial guess $\epsilon^{(0)}$ to the unknown inlier threshold ϵ. Using $\epsilon^{(0)}$, GNC-MinT performs the same *weight, variable, and μ update* steps as GNC-TLS until convergence, when $\boldsymbol{w}^{(t)}$ becomes binary. At this point, GNC-MinT (i) scores how well the empirical distribution of the squares of the residuals fits a χ^2 distribution, using the Cramér–von Mises test, restricting the test to the measurements classified as inliers at iteration t,[5] (ii) stores the score and the current estimate, and (iii) decreases the value of $\epsilon^{(t)}$ to prepare for the next iteration. The algorithm terminates (i) when the χ^2 fitness score either remains unchanged or worsens across consecutive iterations, or (ii) when $\epsilon^{(t)}$ either remains unchanged across consecutive iterations or becomes less than *NoiseLowBnd*. GNC-MinT is given in Algorithm 4, and is described in detail below.

Initialization. GNC-MinT first initializes $\epsilon^{(0)}$ with *NoiseUpBnd*. Then $\mu^{(0)}$, $\boldsymbol{w}^{(0)}$, and $\boldsymbol{x}^{(0)}$ are initialized similarly to GNC but using $\epsilon^{(0)}$ instead of ϵ (lines 2-3). GNC-MinT also introduces the counter j (initialized to 1 in Line 1), which counts how many times $\epsilon^{(\cdot)}$ has been updated.

Weight, Variable, and $\boldsymbol{\mu}$ Update. Lines 5, 6, and 7 in GNC-MinT are the same as the corresponding updated in GNC, with the exception that the current guess $\epsilon^{(j-1)}$ is used in Line 5 instead of the unknown ϵ. Since these updates are the same as GNC, Theorem 15 guarantees that the weights $\boldsymbol{w}^{(t)}$ eventually become binary (for some t), *i.e.,* GNC-MinT's iterations of weight, variable, and μ update converge. Line 8 checks whether this is indeed the case.

[5] Proposition 4 implies that for TLS the inliers' generative probability distribution is a Normal distribution. As a result, the square of the inliers' residuals will follow a χ^2 distribution.

Algorithm 4: Minimally tuned GNC for TLS (GNC-MinT).

```
Input: Measurements $\boldsymbol{y}_i$, $\forall i \in \mathcal{M}$; $\mathit{MaxIterations} > 0$; $\mathit{MuUpdateFactor} > 1$;
       $\mathit{NoiseUpBnd}, \mathit{NoiseLowBnd} \geq 0$;
       $\chi^2$ distribution's degrees of freedom $d > 0$.
Output: Estimate of $\boldsymbol{x}^\circ$ and corresponding inliers.
$\epsilon^{(0)} = \mathit{NoiseUpBnd}$;   $j = 1$;
$\mu^{(0)} = \frac{(\epsilon^{(0)})^2}{2\max_{i \in \mathcal{M}} r^2(\boldsymbol{y}_i, \boldsymbol{x}^{(0)}) - (\epsilon^{(0)})^2}$;
$\boldsymbol{w}^{(0)} = \mathbf{1}_{\mathcal{M}}$;  $\boldsymbol{x}^{(0)} = \text{VariableUpdate}(\boldsymbol{w}^{(0)})$;
for $t = 1, \ldots, \mathit{MaxIterations}$ do
    $\boldsymbol{w}^{(t)} = \text{WeightUpdate}(\boldsymbol{x}^{(t-1)}, \mu^{(t-1)}, \epsilon^{(j-1)})$;
    $\boldsymbol{x}^{(t)} = \text{VariableUpdate}(\boldsymbol{w}^{(t)})$;
    $\mu^{(t)} = \mathit{MuUpdateFactor} \cdot \mu^{(t-1)}$;
    if IsBinary($\boldsymbol{w}^{(t)}$) then
        $\mathcal{I}^{(j)} = \text{supp}(\boldsymbol{w}^{(t)})$;
        $s^{(j)} = \text{Chi2Fit}(\boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(j)}}, \boldsymbol{x}^{(t)}), d)$;
        $\widetilde{\boldsymbol{w}}^{(j)} = \boldsymbol{w}^{(t)}$;   $\widetilde{\boldsymbol{x}}^{(j)} = \boldsymbol{x}^{(t)}$;
        $s_{\min} = \min_{z \in \{1,2,\ldots,j\}} s^{(z)}$;
        if $s^{(j)} = s^{(j-1)}$ then
            break;
        else if $s^{(j)} > s_{\min}$ then
            $k++$;                                          // Fitness worsens
            if $k = \mathit{SamplesToConverg}$ then
                break;
            end
        else
            $k = 0$;
        end
        $\tilde{\epsilon} = \max_{i \in \mathcal{I}^{(j)}} \{r(\boldsymbol{y}_i, \boldsymbol{x}^{(t)}) \text{ s.t. } r(\boldsymbol{y}_i, \boldsymbol{x}^{(t)}) < \epsilon^{(j-1)}\}$;
        $\epsilon^{(j)} = (\epsilon^{(j-1)} + \tilde{\epsilon})/2$;
        if $\epsilon^{(j)} = \epsilon^{(j-1)}$ or $\epsilon^{(j)} < \mathit{NoiseLowBnd}$ then
            break;
        end
        $\mu^{(t)} = \mu^{(0)}$;  $\boldsymbol{w}^{(t)} = \boldsymbol{w}^{(0)}$;  $\boldsymbol{x}^t = \boldsymbol{x}^{(0)}$;  $j++$;
    end
end
$j_{\min} = \arg\min_{z \in \{1,2,\ldots,j\}} s^{(z)}$;
return $(\widetilde{\boldsymbol{x}}^{(j_{\min})}, \text{supp}(\widetilde{\boldsymbol{w}}^{(j_{\min})}))$.
```

Application	Greedy(MC)	Greedy(MTS)	ADAPT(MC)	ADAPT(MTS)	ADAPT-MinT	GNC	GNC-MinT
Mesh Registration	80% [13.71 s]	80% [12.98 s]	80% [14.42 s]	80% [14.39 s]	80% [12.36 s]	80% [5.12 s]	80% [10.68 s]
Shape Alignment	80% [0.15 s]	80% [0.15 s]	80% [0.22 s]	80% [0.23 s]	80% [0.25 s]	80% [0.03 s]	80% [0.06 s]
PGO (2D)	60% [5.04 s]	10% [0.76 s]	80% [5.04 s]	80% [4.92 s]	60% [5.61 s]	90% [1.41 s]	80% [2.17 s]
PGO (3D)	60%[9.23 h]	40%[9.55 h]	60%[60.4 min]	40%[42.04 min]	90%[61.3 min]	90%[85.8 s]	90%[101.62 s]

Table 2.1: **Robustness of proposed algorithms.** Robustness to outliers and average of median running time of the proposed algorithms.

$\boldsymbol{\chi}^2$ Fitness Test. Once $\boldsymbol{w}^{(t)}$ has converged, GNC-MinT checks how well the residuals classified as inliers fit a χ^2 distribution. Line 9 collects the inliers, and Line 10 computes the fitness score $s^{(j)}$ by calling Chi2Fit (Algorithm 7 in D.2). The score $s^{(j)}$ is such that $s^{(j)} > 0$; smaller value indicates better fit. Line 11 stores the current estimate and weights.

Inlier Threshold Update. Once the fitness score at $\boldsymbol{x}^{(t)}$ has been computed, GNC-MinT updates the inlier threshold guess to the mean between the current inlier threshold guess and the largest residual among the measurements currently classified as inliers (Line 24). Evidently, $\epsilon^{(j)} \leq \epsilon^{(j-1)}$.

Re-initialization of Weights, Variable, and $\boldsymbol{\mu}$. Once $\epsilon^{(j)}$ has been updated, GNC-MinT re-initializes $\mu^{(t)}$, $\boldsymbol{w}^{(t)}$, and $\boldsymbol{x}^{(t)}$ (Line 28), in preparation for another round of GNC with the new threshold $\epsilon^{(j)}$. The counter j is also increased by 1 (Line 28).

Termination. GNC-MinT terminates when either

- the maximum number of iterations is reached (Line 4), or
- the fitness score remains unchanged across 2 consecutive iterations (Line 13) or the fitness score worsens for *SamplesToConverg* consecutive iterations (lines 15-21; In this thesis, *SamplesToConverg* $= 2$),[6] or
- it is no longer possible to decrease $\epsilon^{(j)}$ (Line 25) (when $\epsilon^{(j)} = \epsilon^{(j-1)}$, then GNC-MinT would converge again to the same solution if it were to continue running).

Upon termination, GNC-MinT returns the inlier set with the best χ^2 fitness score (lines 31-32).

Remark 18 (Tuning GNC-TLS's ϵ vs. Tuning GNC-MinT's *NoiseUpBnd* and *NoiseLowBnd*)**.** *Knowing ϵ, or estimating it accurately, can be hard and time consuming: ϵ typically varies across both applications and problem instances within the same application. In contrast, guessing upper and lower bounds for ϵ is easier, making* GNC-MinT minimally tuned.

[6] The intuition is that if outliers exist among the measurements, then decreasing $\epsilon^{(j-1)}$ to $\epsilon^{(j)}$ leads to rejecting more outliers, leading to a better χ^2 fit. But if all outliers have been rejected, then decreasing $\epsilon^{(j-1)}$ results into rejecting inliers, worsening the χ^2 fit or keeping it the same.

Remark 19 (Termination in GNC-MinT)**.** *In Proposition 4, we observed* TLS *implicitly searches for inliers with Normally distributed residuals. At the same time, the sum of the squares of Normally distributed variables follows a χ^2 distribution [55]. For this reason, the stopping condition for* GNC-MinT *is based on a χ^2 fitness test, performed by the* Chi2Fit *routine used in Line 10.* Chi2Fit *estimates the variance of the χ^2 distribution, hence it implicitly guesses the magnitude of the inlier noise.*

2.6 Experiments and Applications

We showcase the proposed algorithms in three robot perception problems: mesh registration (Section 2.6.1), shape alignment (Section 2.6.2), and Pose Graph Optimization (PGO) (Section 2.6.3). We performed all the experiments in MATLAB running on a Linux machine with the Intel i-97920X (4.3 GHz). No GPU support was used.

The results show that ADAPT and GNC outperform the state of the art and are robust up to $80-90\%$ outliers. Their minimally tuned versions achieve similar performance, without relying on the knowledge of the inlier noise. We summarize the observed performance of the algorithms (robustness to outliers and average median running time) in Table 2.1, where we also include Greedy's performance. In Table 2.1, we observe:

- Greedy is on average slower than the proposed algorithms (2 times slower than GNC in mesh registration and shape alignment, and up to 100 times slower than GNC in PGO); in addition to being slower, Greedy is also less robust than both ADAPT and GNC in PGO, and even against ADAPT's and GNC's minimally tuned versions.

- GNC and GNC-MinT achieve the lowest running time, retaining, at the same time, the robustness to outliers achieved by all proposed algorithms. Specifically, in mesh registration and shape alignment, ADAPT and GNC, as well as their minimally tuned versions, are practically on par with each other in terms of their robustness to outliers, yet GNC and GNC-MinT are 2 to 10 times faster; and in the PGO experiments, ADAPT and ADAPT-MinT can exhibit similar, or even higher accuracy than GNC and GNC-MinT (cf. Fig. H-1), yet GNC and GNC-MinT are on average 10 times faster than ADAPT and ADAPT-MinT.

Choice of Parameters. We refer to ADAPT as ADAPT(MC) if it solves the MC problem ($\ell = +\infty$ in Line 6 of Algorithm 1), and as ADAPT(MTS) if it solves the MTS problem ($\ell = 2$). We also compare against the greedy algorithm of Section 2.3.1, which we stop when the constraint in Eq. (G-MC) is satisfied. We denote the corresponding technique with the label Greedy(MC) and Greedy(MTS), when we use $\ell = +\infty$ and $\ell = 2$ in Eq. (G-MC), respectively. In all applications, we set in

- ADAPT: $\tau = \sqrt{\text{chi2inv}(0.99, nd)}$, where d is the number of degrees of freedom of the measurement noise and depends on the application, and n is the cardinality of the chosen inlier set at the current iteration (*i.e.,* at ADAPT's iteration t, $n = |\mathcal{I}^{(t)}|$; *cf.* ADAPT's Line 4); $\theta = \sqrt{\text{udchi2inv}(0.05, n_1 d, n_2 d, \sigma^2))}$, where σ is the standard deviation of the noise, $n_1 = |\mathcal{I}^{(t)}|$ and $n_2 = |\mathcal{I}^{(t-1)}|$, while udchi2inv is the inverse of the cumulative probability distribution of a random variable $z = |z_1 - z_2|$, where z_1 and z_2 are χ^2 random variables (*cf.* Line 7 of ADAPT);[7] *MaxIterations* = 1000; *SamplesToConverg* = 3; and *ThrDiscount* = 0.99.

- ADAPT-MinT: *MaxIterations* = 1000; *ThrDiscount* = 0.99; *MinSamples* = 2; *WindowSize* = 3; *ConvergThr* = 10^{-4}.

- GNC: $\epsilon = \sigma\sqrt{\text{chi2inv}(0.99, d)}$; *MaxIterations* = 1000; and *MuUpdateFactor* = 1.4.

- GNC-MinT: *MaxIterations* = 1000; *MuUpdateFactor* = 1.4^2;[8] *NoiseUpBnd* and *NoiseLowBnd* depend on the application, and are described in the subsections below.

[7] We set $\theta = \sqrt{\text{udchi2inv}(0.05, n_1 d, n_2 d, \sigma^2))}$ assuming the measurement noise is normally distributed, since, then, $z_1 = \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t)}}, \boldsymbol{x}) \|_2^2$ and $z_2 = \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}^{(t-1)}}, \boldsymbol{x}) \|_2^2$ are indeed χ^2 random variables.

[8] We set *MuUpdateFactor* = 1.4^2 in GNC-MinT such that the algorithm has similar runtime as GNC. On average, by choosing *MuUpdateFactor* = 1.4^2, instead of 1.4, we speed-up the convergence of the weights $\boldsymbol{w}^{(t)}$ to a binary vector (GNC-MinT's Line 8) by a multiplicative factor of 2.

2.6.1 Mesh Registration

In mesh registration, given a set of 3D points $\boldsymbol{a}_i \in \mathbb{R}^3$, $i \in \mathcal{M}$, and a set of primitives $\boldsymbol{P}_i$, $i \in \mathcal{M}$ (being points, lines and/or planes) with putative correspondences $\boldsymbol{a}_i \leftrightarrow \boldsymbol{P}_i$, we aim to find the best rotation $\boldsymbol{R} \in \mathrm{SO}(3)$ and translation $\boldsymbol{t} \in \mathbb{R}^3$ that align the point cloud to the 3D primitives. In practice, the primitives $\boldsymbol{P}_i$ often correspond to vertices, edges, or faces of the CAD model of an object, while the points $\boldsymbol{a}_i$ are measured points (*e.g.,* from a lidar observing a scene containing that object), and mesh registration allows retrieving the pose of the (known) object in the point cloud.

The residual error in mesh registration is $r(\boldsymbol{R}, \boldsymbol{t}) = \mathrm{dist}(\boldsymbol{P}_i, \boldsymbol{R}\boldsymbol{a}_i + \boldsymbol{t})$, where $\mathrm{dist}(\cdot)$ denotes the distance between a primitive $\boldsymbol{P}_i$ and a point $\boldsymbol{a}_i$ after the transformation $(\boldsymbol{t}, \boldsymbol{R})$ is applied. The formulation can also accommodate weighted distances to account for heterogeneous and anisotropic measurement noise. In the outlier-free case, Briales *et al.* [57] developed a certifiably optimal non-minimal solver when the 3D primitives include points, lines, and planes and the noise is anisotropic. We use GNC, ADAPT, and their minimally tuned versions to efficiently robustify Briales' non-minimal solver.

Experimental Setup. We use the "aeroplane-2" mesh model from the PASCAL+ dataset [56]. We compute statistics over 20 Monte Carlo runs, with increasing amounts of outliers. At each Monte Carlo run, we generate a new point cloud from the mesh by randomly sampling a subset of points lying on the vertices, edges, and faces of the mesh, and then apply a random transformation. We also add Gaussian noise with $\sigma = 0.05 d_{\text{mesh}}$, where d_{mesh} is the diameter of the mesh. We establish 40 point-to-point, 80 point-to-line, and 80 point-to-plane correspondences, and create outliers by adding incorrect point to point/line/plane correspondences. Since the number of degrees of freedom of the measurement noise is $d = 3$, $\epsilon = \sigma\sqrt{\mathrm{chi2inv}(0.99, d)} = 0.0128$. Moreover, we choose $NoiseUpBnd = 3\epsilon = 0.0384$, and $NoiseLowBnd = \epsilon/3 = 0.0043$.

We benchmark our algorithms against a RANSAC implementation with 400 maximum iterations, using the 12-point minimal solver presented in [58].

Mesh Registration Results. Fig. 2-4 shows the rotation error, translation error, and running time for each technique (all plots are in log-scale). The Greedy(MC), Greedy(MTS), GNC, ADAPT(MC), and ADAPT(MTS), as well as the minimally tunedADAPT-MinT have comparable performance, and are robust against up to 80% outliers. GNC-MinT has similar performance, exhibiting slightly higher errors. All proposed methods outperform RANSAC, which starts breaking at 30% of outliers.

In terms of runtime, RANSAC's runtime grows with the number of outliers. Instead, Greedy's, ADAPT's, and ADAPT-MinT's runtimes grow linearly with the number of outliers, while GNC's and GNC-MinT's remain roughly constant.

Qualitative results for mesh registration are given in Fig. 2-3.

2.6.2 Shape Alignment

In shape alignment, given 2D features $\boldsymbol{z}_i \in \mathbb{R}^2, i \in \mathcal{M}$ in a single image and 3D points $\boldsymbol{B}_i \in \mathbb{R}^3, i \in \mathcal{M}$ of an object with putative correspondences $\boldsymbol{z}_i \leftrightarrow \boldsymbol{B}_i$ (potentially including outliers), the goal is to find the best scale $s > 0$, rotation $\boldsymbol{R}$, and translation $\boldsymbol{t}$ of the object that projects the 3D shape to the 2D image under weak perspective projection. In practice, the 3D points $\boldsymbol{B}_i$ often correspond to distinguishable points on the CAD model of an object, while the 2D features $\boldsymbol{z}_i$ are measured pixels (*e.g.,* from a camera observing a scene containing that object), and shape alignment allows retrieving the pose of the (known) object in the image.

The residual error in shape alignment is $r(s, \boldsymbol{R}, \boldsymbol{t}) = \|\boldsymbol{z}_i - s\Pi\boldsymbol{R}\boldsymbol{B}_i - \boldsymbol{t}\|$, where $\Pi \in \mathbb{R}^{2\times 3}$ is the weak perspective projection matrix (equal to the first two rows of a 3×3 identity matrix). Note that $\boldsymbol{t}$ is a 2D translation, but under weak perspective projection one can extrapolate a 3D translation (*i.e.,* recover the distance of the camera to the object) using the scale s. We use the closed-form solution introduced in [60] as non-minimal solver. While potentially suboptimal, the solver in [60] works well in practice, and is faster than the certifiably optimal solver proposed in [43].

Experimental Setup. We test the performance of GNC, GNC-MinT, ADAPT, and ADAPT-MinT on the FG3DCar dataset [59] against (i) Zhou's method [61], and (ii) RANSAC with 400 maximum iterations using a 4-point minimal solver. We use the ground-truth 3D shape model as $\boldsymbol{B}$ and the ground-truth 2D landmarks as $\boldsymbol{z}$. To generate outliers for each image, we set random incorrect correspondences between 3D points and 2D features. We assume $\sigma = \sqrt{1 \times 10^{-5}} = 0.0032$, and, since $d = 2$, $\epsilon = \sigma\sqrt{\text{chi2inv}(0.99, d))} = 0.0096$. Also, similarly to mesh registration, $NoiseUpBnd = 3\epsilon = 0.0288$, and $NoiseLowBnd = \epsilon/3 = 0.0032$.

Shape Alignment Results. Fig. 2-5 shows in log-scale the rotation and translation error, and running time for all techniques. Statistics are computed over all the images in the FG3DCar dataset. Zhou's method degrades quickly with increasing number of outliers. Instead, all other algorithms are robust against 80% of outliers.

RANSAC's runtime grows exponentially with the number of outliers. GNC, GNC-MinT, and Zhou's method runtime is constant, being smaller than RANSAC's for outlier rates more than 40%. ADAPT's and ADAPT-MinT's runtimes grow linearly.

Qualitative results for shape alignment are given in Fig. 2-3.

2.6.3 Pose Graph Optimization (PGO)

Pose Graph Optimization (PGO) is a common backend for Simultaneous Localization and Mapping (SLAM) [1]. PGO estimates a set of poses $(\boldsymbol{t}_i, \boldsymbol{R}_i)$, $i \in \mathcal{M}$ from pairwise relative pose measurements $(\bar{\boldsymbol{t}}_{ij}, \bar{\boldsymbol{R}}_{ij})$ (potentially corrupted with outliers). The residual error is the distance between the expected relative pose and the relative measurements:

$$\sqrt{\|\text{Log}(\bar{\boldsymbol{R}}_{ij}^{\mathsf{T}}\boldsymbol{R}_i^{\mathsf{T}}\boldsymbol{R}_j)\|^2_{\boldsymbol{\Omega}_{ij}^R} + \|\bar{\boldsymbol{R}}_{ij}^{\mathsf{T}}(\bar{\boldsymbol{t}}_{ij} - \boldsymbol{R}_i^{\mathsf{T}}(\boldsymbol{t}_i - \boldsymbol{t}_j))\|^2_{\boldsymbol{\Omega}_{ij}^t}}$$

where $\boldsymbol{\Omega}_{ij}^R$ and $\boldsymbol{\Omega}_{ij}^t$ are respectively the known rotation and translation measurement information matrix. For a vector $\boldsymbol{a}$, the symbol $\|\boldsymbol{a}\|^2_{\boldsymbol{\Omega}}$ denotes the standard Mahalanobis norm: $\|\boldsymbol{a}\|^2_{\boldsymbol{\Omega}} = \boldsymbol{a}^{\mathsf{T}}\boldsymbol{\Omega}\boldsymbol{a}$. The $\text{Log}(\cdot)$ denotes the logarithm map for the rotation group, which, roughly speaking, converts a rotation matrix to a vector (in 3D) or to a scalar (in 2D).[9]

In the outlier free case, SE-Sync [10] provides a global solver for PGO, and we have used it in our 2D SLAM experiments in [41]. However, SE-Sync becomes too slow

[9] For simplicity, here we use the geodesic distance $\|\text{Log}(\bar{\boldsymbol{R}}_{ij}^{\mathsf{T}}\boldsymbol{R}_i^{\mathsf{T}}\boldsymbol{R}_j)\|$, while alternative rotation

in the 3D SLAM tests considered In this thesis: rather than a limitation of SE-Sync, this follows from the fact that in early iterations, both ADAPT and GNC (as well as their minimally tuned variants) solve problems with many outliers; in these cases, SE-Sync's relaxation is not tight,[10] and SE-Sync tends to perform multiple steps in the Riemannian staircase [10], becoming impractical.

To circumvent these issues, instead of SE-Sync, we use g2o [19], which is a local solver for PGO, and use the odometry as initial guess. We remark this option is only viable when the odometric guess is available and considered reliable. Appendix H in the appendix compares the use of SE-Sync and g2o within our algorithms and shows the two achieve comparable performance when the odometric guess is reliable.

Experimental Setup. We test the performance of our algorithms on synthetic and real datasets for 2D and 3D PGO. We use a synthetic grid [41], and CSAIL [65] in 2D, and a synthetic sphere, and Garage [62] in 3D. We compute statistics over 10 Monte Carlo runs, with increasing amounts of outliers. At each Monte Carlo run, we spoil existing loop closures with random outliers. We consider odometric measurements as inliers and use the odometry as initial guess for g2o. Since $d = 3$ in 2D SLAM, $\epsilon = \sqrt{\text{chi2inv}(0.99, 3)} = 3.3682$, and since $d = 6$ in 3D SLAM, $\epsilon = \sqrt{\text{chi2inv}(0.99, 6)} = 4.1$.[11] Regarding GNC-MinT and ADAPT-MinT, we normalize the measurements' covariance matrices provided by each dataset to simulate the case in which the covariances are unknown, and we set $NoiseUpBnd = 1\text{m}$ and $NoiseLowBnd = 0.01\text{m}$.[12]

We benchmark our algorithms against (i) g2o [19], (ii) *dynamic covariance scaling* (DCS) [21], and (iii) *pairwise consistent measurement set maximization* (PCM) [66]. The performance of DCS is fairly sensitive to the choice of the kernel size Φ, which

distances are often used in PGO, see [10], [62].

[10] Indeed, it has been observed that the presence of large noise can easily induce failures in relaxations of 3D SLAM [63], while their 2D counterparts are observed to remain tight in the presence of relatively large noise [64].

[11] In SLAM we do not need multiply by the covariance because the objective function performs a whitening transformation via the information matrix.

[12] In detail: we normalize each measurement's information matrix Ω_{ij} by a factor α_{ij} that represents the mean information (inverse variance) of the translation measurements (we ignore the effect of the rotation, since it has 1-2 orders of magnitude smaller errors).

is a parameter in the algorithm: we tested different kernel sizes $\Phi = \{1, 10, 100\}$ for DCS, and we used the same ϵ for GNC, ADAPT, and PCM. For clarity of visualization, we only report the best two parameters (leading to smallest errors) for DCS in the figures.

2D PGO Results. Fig. 2-6 shows the Average Trajectory Error (ATE) and the running time for the synthetic grid. g2o is a non-robust solver, and performs poorly even when few outliers are present. ADAPT(MC) and ADAPT(MTS) outperform the Greedy algorithm. GNC outperforms the state of the art, and is robust to 90% of outliers. GNC-MinT is also robust up to 90% of outliers, outperforming ADAPT-MinT, which breaks at 70% of outliers. DCS(10) has similar performance to GNC, being robust until 90% of outliers; DCS(100) degrades with increasing number of outliers, stressing the importance of parameter tuning in DCS. PCM starts degrading at relatively low outlier rates. Fig. 2-6 shows that the runtimes of GNC, GNC-MinT, g2o, DCS, and PCM are roughly constant, while ADAPT's and ADAPT-MinT's runtime grows linearly in the number of outliers.

Fig. H-1 shows the ATE and the running time for the CSAIL dataset. All ADAPT, ADAPT-MinT, GNC, and GNC-MinT outperform the state of the art, and are robust against 90% of outliers. DCS starts breaking at 50% of outliers. PCM and g2o perform poorly across the whole spectrum. Both ADAPT-MinT and GNC-MinT perform similarly to ADAPT and GNC, although being minimally tuned algorithms. Similarly to grid, the runtimes of GNC, GNC-MinT, g2o, DCS, and PCM are roughly constant; ADAPT's and ADAPT-MinT's grow linearly.

3D PGO Results. Fig. 2-8 and Fig. 2-9 show the ATE and the running time in the case of Sphere and Garage, respectively (both in log-scale). We omit Greedy, ADAPT, ADAPT-MinT, and PCM because their running times become impractical for these datasets (more than 10 minutes per run). In both Sphere and Garage, we observe that GNC and GNC-MinT outperform DCS and g2o, regardless of DCS's parameter choice. Importantly, GNC-MinT outperforms GNC in Garage. The covariances are unreliable in the Garage dataset, hence causing GNC to set an incorrect ϵ. On the other hand, GNC-MinT is able to *infer* the correct ϵ and ensure accurate estimation. GNC's and

GNC-MinT's running times slightly increase with increasing number of outliers, while DCS's and g2o's are constant.

Qualitative results are given in Fig. 2-3.

2.7 Extended Literature Review

We extend the literature review in Section 1.1, to discuss outlier-robust estimation in robotics and computer vision (Section 2.7.1), and in statistics and control (Section 2.7.2).

2.7.1 Outlier-robust Estimation in Robotics and Computer Vision

Outlier-robust estimation has been an active research area in robotics and computer vision [67]–[69]. Two of the predominant paradigms to gain robustness against outliers are *consensus maximization* [14] and *M-estimation* [69]. In both paradigms, the literature is mainly divided into (i) *fast heuristics*, algorithms that are efficient but provide little performance guarantees, and (ii) *global solvers*, algorithms that offer optimality guarantees but scale poorly with the problem size.

Fast Heuristics. For consensus maximization, RANSAC [15], [70] has been a widely adopted heuristic due to its efficiency and effectiveness in the low-outlier regime. Tzoumas *et al.* [41] proposed ADAPT for *minimally trimmed squares* (MTS) estimation, a formulation that bears similarity with consensus maximization (*cf.* Section 2.1). For M-estimation, local nonlinear optimization is typically employed, which relies on the availability of a good initial guess [71], [72]. Instead, the proposed GNC algorithm by Yang *et al.* [43] provides a method for solving M-estimation without requiring an initial guess (also see [73]). Barron [74] proposes a single parametrized function that generalizes a family of robust cost functions in M-estimation. Chebrolu *et al.* [75] design an expectation-maximization algorithm to simultaneously estimate the unknown quantity $\boldsymbol{x}$ and choose the best robust cost ρ in Eq. (1.3). These algorithms, however, still rely on an estimate of the inlier noise threshold ϵ.

Global Solvers. Global solvers essentially perform exhaustive search to ensure global optimality. For instance, *branch-and-bound* (BnB) has been exploited to globally solve consensus maximization in several low-dimensional perception tasks [16], [76]–[84]. Despite its global optimality guarantees, BnB has exponential running time in the worst case. It is also possible to globally solve consensus maximization and M-estimation by enumerating all possible minimizers [85], [86]. However, these algorithms are close to exhaustive search and do not scale to high-dimensional problems.

Certifiably robust algorithms are a class of global solvers that have been shown to strike a good balance between computational complexity and global optimality [2], [87]. Certifiable algorithms relax non-convex robust estimation problems into convex *semidefinite programs* (SDP), whose solutions can be obtained in polynomial time and provide readily checkable *a posteriori* global optimality certificates [48], [49], [88], [89]. Although solving large-scale SDPs is computationally expensive, recent work has shown that optimality certification (*i.e.,* verifying the global optimality of candidate solutions returned by fast heuristics) can scale to large problems by leveraging efficient first-order methods [87].

One approach to boost performance is adding a preprocessing layer to prune outliers using consensus maximization, M-estimation, and certifiable algorithms [2], [16], [90].

Below, we discuss representative outlier-robust methods for registration, shape alignment, and SLAM.

Robust Registration. Point cloud registration is a fundamental problem in robotics and computer vision, with applications to 3D reconstruction, localization, and mapping. The goal is to find the rigid transformation (translation and rotation) that best aligns two point clouds or a point cloud and a 3D mesh. We review correspondence-based registration methods, while we refer the reader to [2] for a broader review on 3D registration, including *Simultaneous Pose and Correspondence* methods (*e.g.,* ICP [91]). Correspondence-based registration methods assume availability of putative correspondence between the two point clods. Therefore, they first extract and match features in the two point clouds, using hand-crafted [92] or deep-learned [12], [93] features. Then, they solve an estimation problem to compute the rigid transformation that best aligns the set of corresponding features. In the presence of outliers (*i.e.,* incorrect correspondences), it's typical to resorts to RANSAC [15], [94], along with a 3-point minimal solver [95], [96]. However, in the high-outlier regime (*e.g.,* above 80%), RANSAC tends to be slow and brittle [16], [80]. To gain robustness against a high number of outliers, recent approaches adopt either M-estimation or consensus maximization. Zhou *et al.* [73] propose *fast global registration*, which minimizes the Geman-McClure robust cost function using GNC. Tzoumas *et al.* use ADAPT [41], and Yang *et al.* use GNC [43] to solve point cloud registration with robustness against up to 80% outliers. Bazin *et al.* [97] employ BnB to perform globally optimal rotation search (*i.e.,* 3D registration without translation). Parra *et al.* [16] remove gross outliers adding a preprocessing step before RANSAC or BnB. Yang and Carlone propose invariant measurements to decouple the rotation and translation estimation [89], and develop certifiably robust rotation search using semidefinite relaxation [49]. The joint use of fast heuristics (*e.g.,* GNC) and optimality certification for both point cloud registration and mesh registration has been demonstrated in [2], [87]. The registration approach [2] has been shown to be robust to 99% outliers.

Robust Shape Alignment. Shape alignment consists in estimating the absolute camera pose given putative correspondences between 2D image landmarks and 3D model keypoints (the problem is called 3D *shape reconstruction* when the 3D model is unknown [3], [61], [98]). When a full camera perspective model is assumed, the problem is usually referred to as the *perspective-n-point* (PnP) problem [99]. RANSAC is again the go-to approach to gain robustness against outliers, typically in conjunction with a 3-point minimal solver [100]. Ferraz *et al.* propose an efficient robust PnP algorithm based on iteratively rejecting outliers via detecting large algebraic errors in a linear system [101]. When the 3D model is far from the camera center, a weak perspective camera model can be adopted [61], which leads to efficient robust estimation using GNC [43]. Yang and Carlone [87] develop optimality certification algorithms for shape alignment with outliers, and demonstrate successful application to satellite pose estimation.

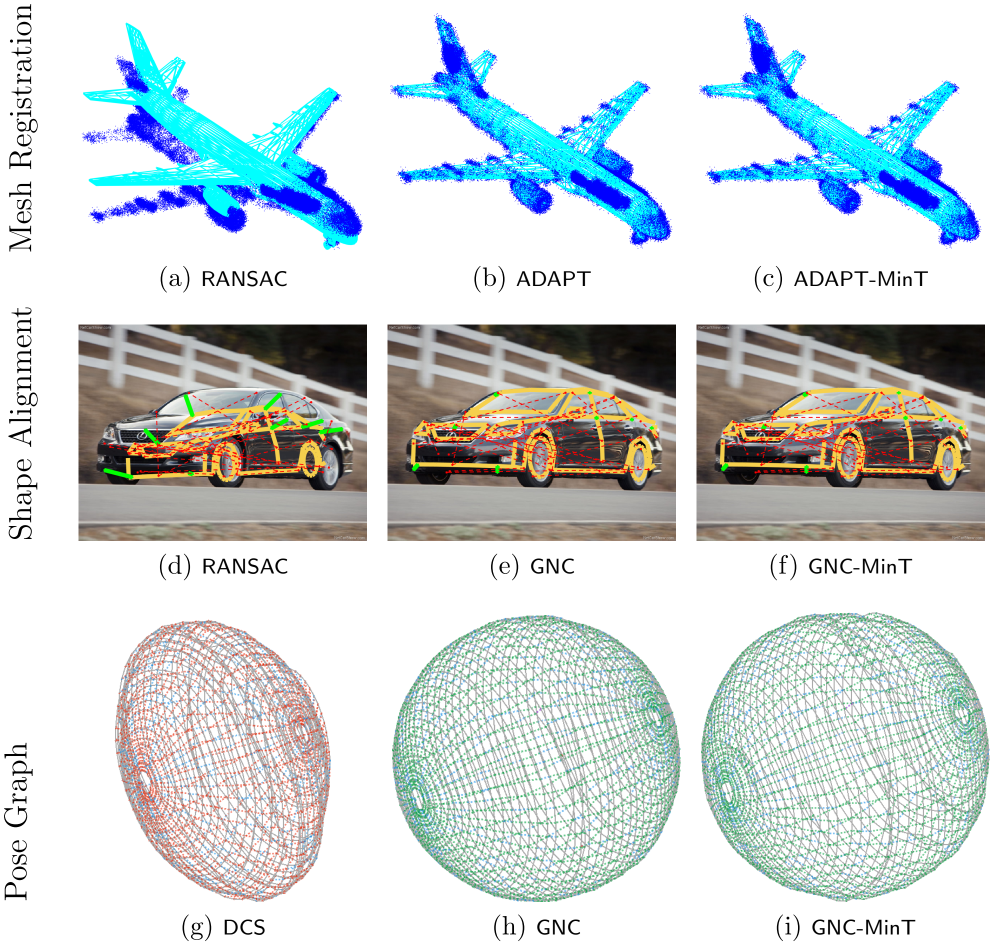
Robust SLAM. SLAM is a fundamental problem in robotics, with applications to autonomous navigation, augmented reality, and 3D reconstruction. The goal is to estimate the trajectory of a robot and a map of the environment, given a sequence of measurements. Outlier-robust SLAM often relied on M-estimators, see, *e.g.,* [69]. Olson and Agarwal [102] use a max-mixture distribution to approximate multimodal measurement noise. Sünderhauf and Protzel [20], [103] augment the problem with latent binary variables responsible for deactivating outliers. Tong and Barfoot [104], [105] propose algorithms to classify outliers via Chi-square statistical tests that account for the effect of noise in the estimate. Latif *et al.* [106] propose *realizing, reversing, and recovering*, which performs loop-closure outlier rejection, by clustering measurements together and checking for consistency using a Chi-squared-based test. Mangelson *et al.* [66] propose a *pair-wise consistency maximization* (PCM) approach for multi-robot SLAM. Agarwal *et al.* [21] propose *dynamic covariance scaling* (DCS), which adjusts the measurement covariances to reduce the influence of outliers. Lee *et al.* [107] use expectation maximization. These approaches rely either on the availability of an initial guess for optimization, or on parameter tuning. Recent work also includes convex relaxations for outlier-robust SLAM [48], [88], [108], [109]. Lajoie *et al.* [48] providesub-optimality guarantees, which however degrade with the quality of the relaxation.

2.7.2 Outlier-robust Estimation in Statistics and Control

Outlier-robust estimation has been also a subject of investigation in statistics and control [110], [111], where it finds applications to distribution learning [112], linear decoding [113], and secure state estimation [114], among others.

Statistics. In its simplest form, outlier-robust estimation aims at learning the mean and covariance of an unknown distribution, given (i) a portion of independent and identically distributed samples, and (ii) a portion of arbitrarily corrupted samples (outliers), where the percentage of corrupted samples is known a-priori. Researchers provide polynomial-time near-optimal algorithms [112], [115]. In cases where one instead aims to estimate an unknown parameter given corrupted measurements, Rousseeuw [116] propose *linear trimmed squares* (LTS), which aims to minimize the cumulative inlier residual error given a known number of outliers. Similar greedy-like algorithms, that also assume a known number of outliers, are the forward greedy by Nemhauser *et al.* [52], and forward-backward greedy by Zhang [117]. Both algorithms have quadratic running time, which is prohibitive in high-dimensional robotics and computer vision applications, such as SLAM. In contrast to [52], [116], [117], the greedy algorithm proposed in [118] considers the number of outliers to be unknown. However, it still requires parameter tuning, this time for an inlier threshold parameter.

Control. Outlier-robust estimation in control takes the form of secure state estimation in the presence of outliers, including adversarial measurement corruptions. Related works [114], [119], [120] propose exponential-time algorithms, achieving exact state estimation when the inliers are noiseless.

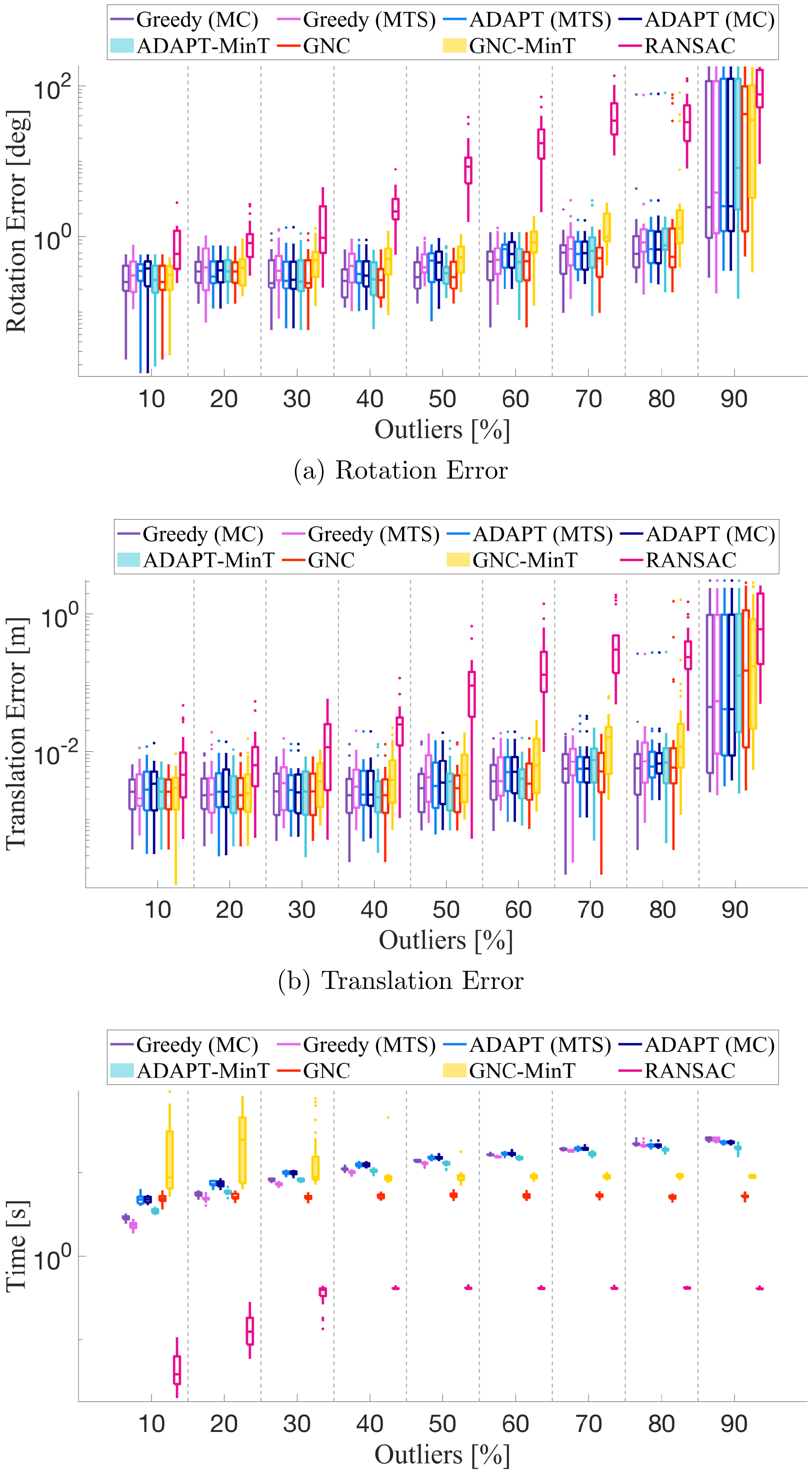


(a) RANSAC (b) ADAPT (c) ADAPT-MinT

(d) RANSAC (e) GNC (f) GNC-MinT

(g) DCS (h) GNC (i) GNC-MinT

Figure 2-3: **Qualitative comparison of the proposed robust estimation algorithms.** We investigate fundamental limits and practical algorithms for outlier-robust estimation. We discuss two algorithms, ADAPT and GNC, that outperform the state of the art (DCS [21] and RANSAC [15] in the figure) in mesh registration, shape alignment, and pose graph optimization. Moreover, we propose two variants, ADAPT-MinT and GNC-MinT, that perform favorably across robotics applications, and do not require parameter tuning (*e.g.,* kernel size in DCS, or maximum inlier noise in RANSAC).

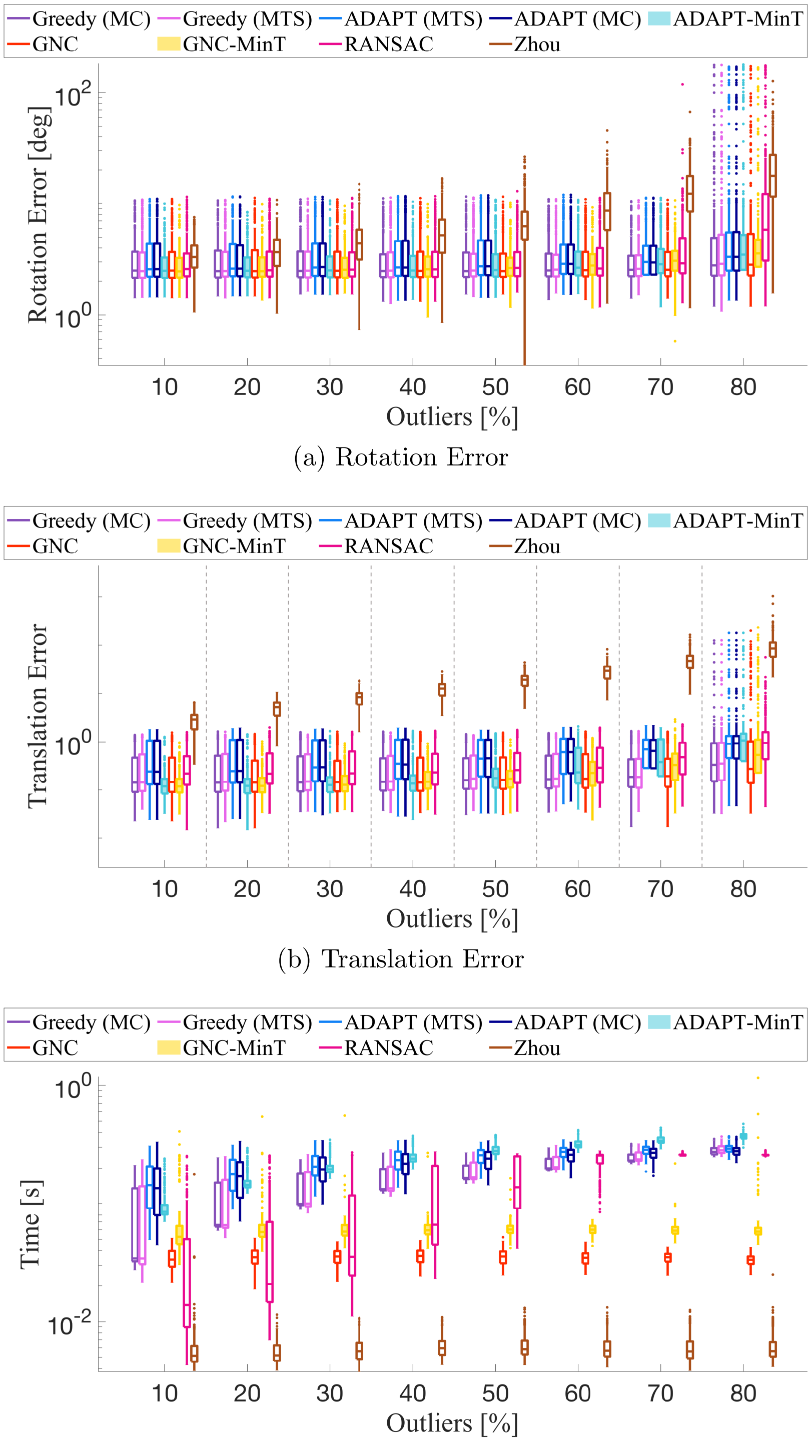


(a) Rotation Error

(b) Translation Error

(c) Running Time

Figure 2-4: **Mesh Registration.** Rotation error, translation error, and running time of the proposed algorithms, compared to RANSAC, on the PASCAL+ "aeroplane-2" dataset [56]. Statistics are computed over 25 Monte Carlo runs and for increasing percentage of outliers.

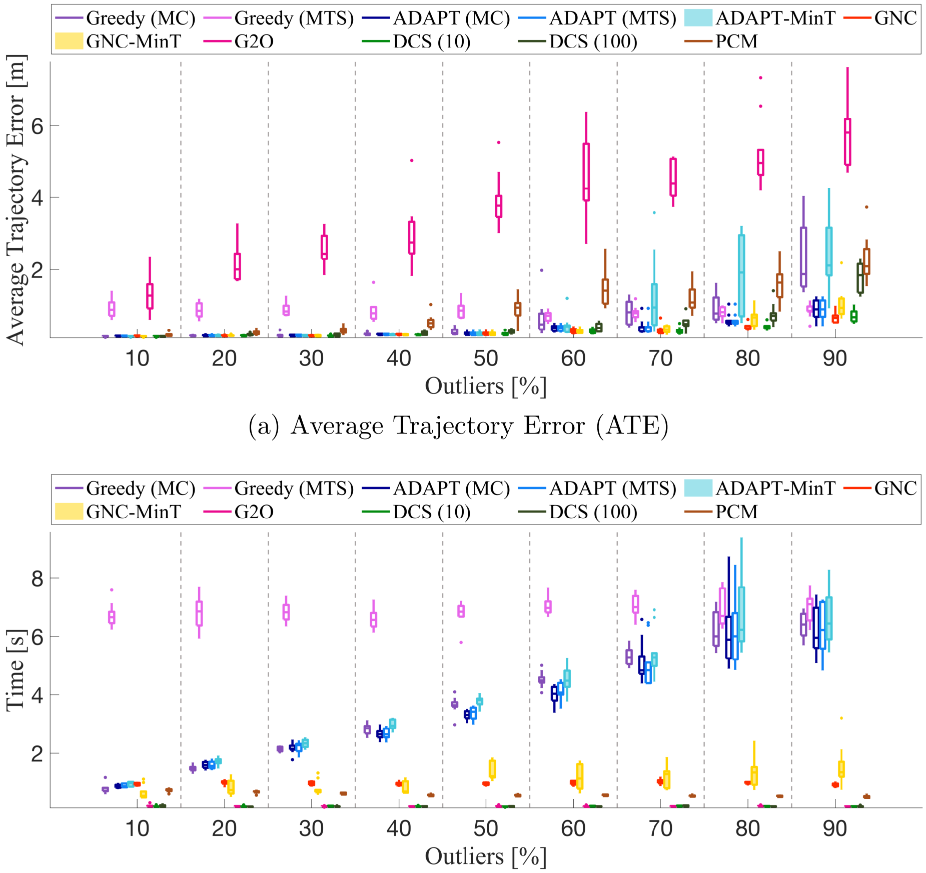


(a) Rotation Error

(b) Translation Error

(c) Running Time

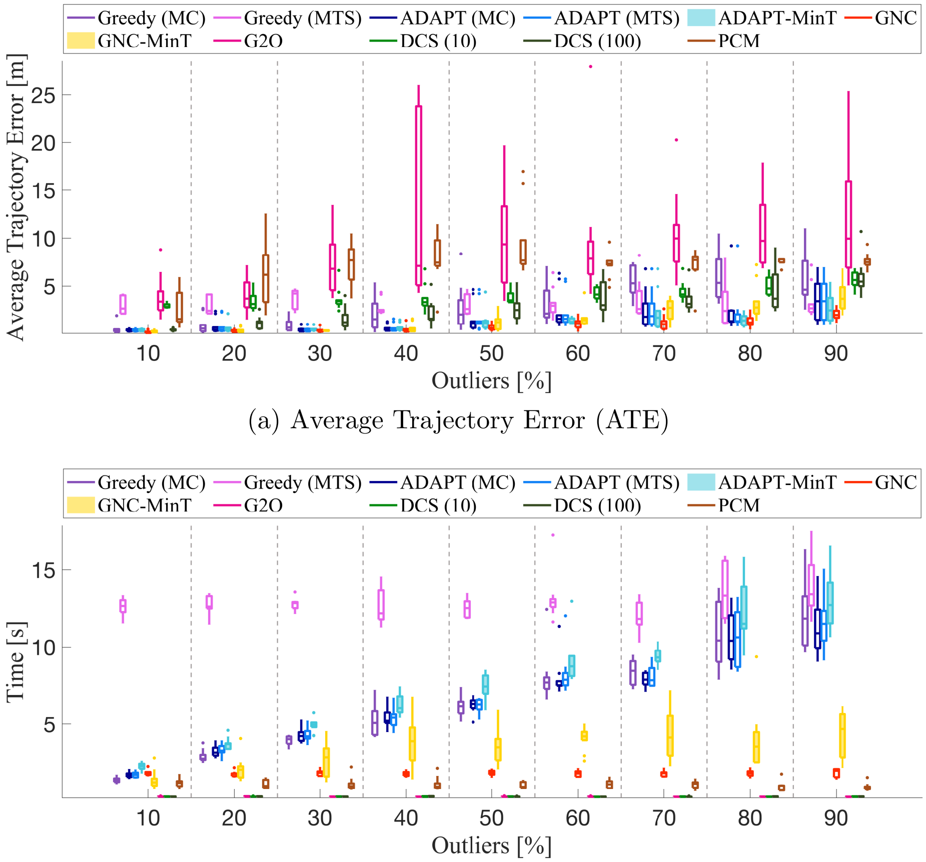
Figure 2-5: **Shape Alignment.** Rotation error (left), translation error (center), and running time (right) of the proposed algorithms, compared to state-of-the-art techniques, on the FG3DCar dataset [59]. Statistics are computed over 25 Monte Carlo runs and for increasing percentage of outliers.



(a) Average Trajectory Error (ATE)

(b) Running Time

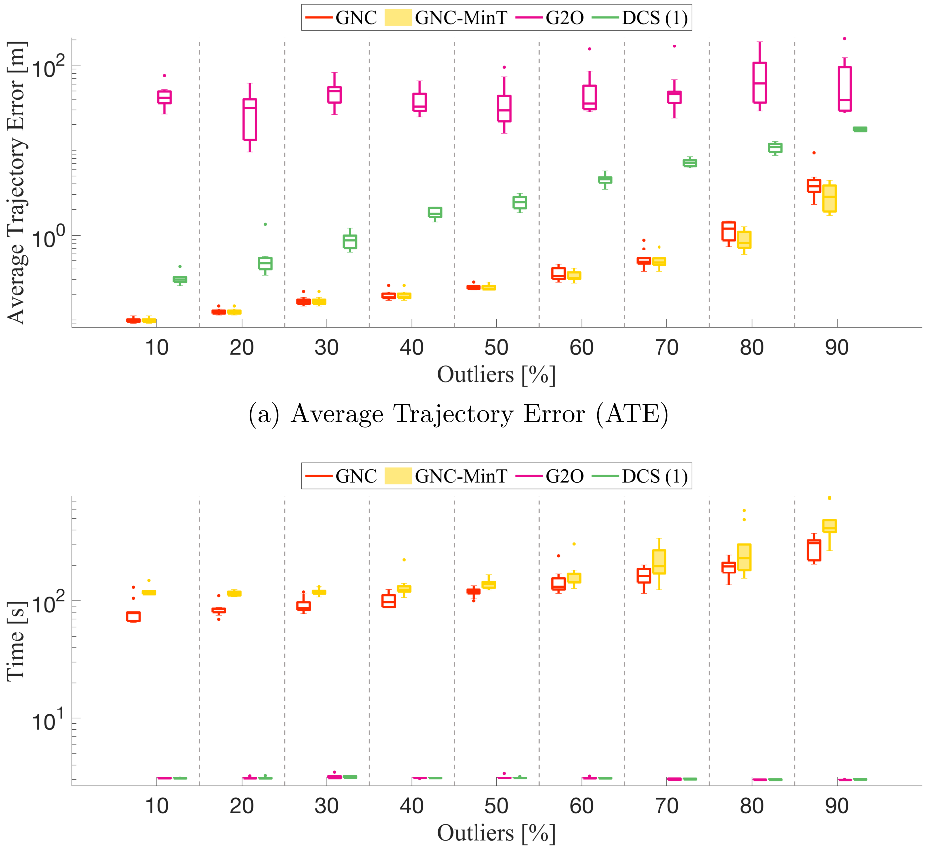
Figure 2-6: **2D SLAM (Grid).** Average Trajectory Error (ATE) and running time of the proposed algorithms compared to state-of-the-art techniques on a synthetic grid dataset for increasing outliers.



(a) Average Trajectory Error (ATE)

(b) Running Time

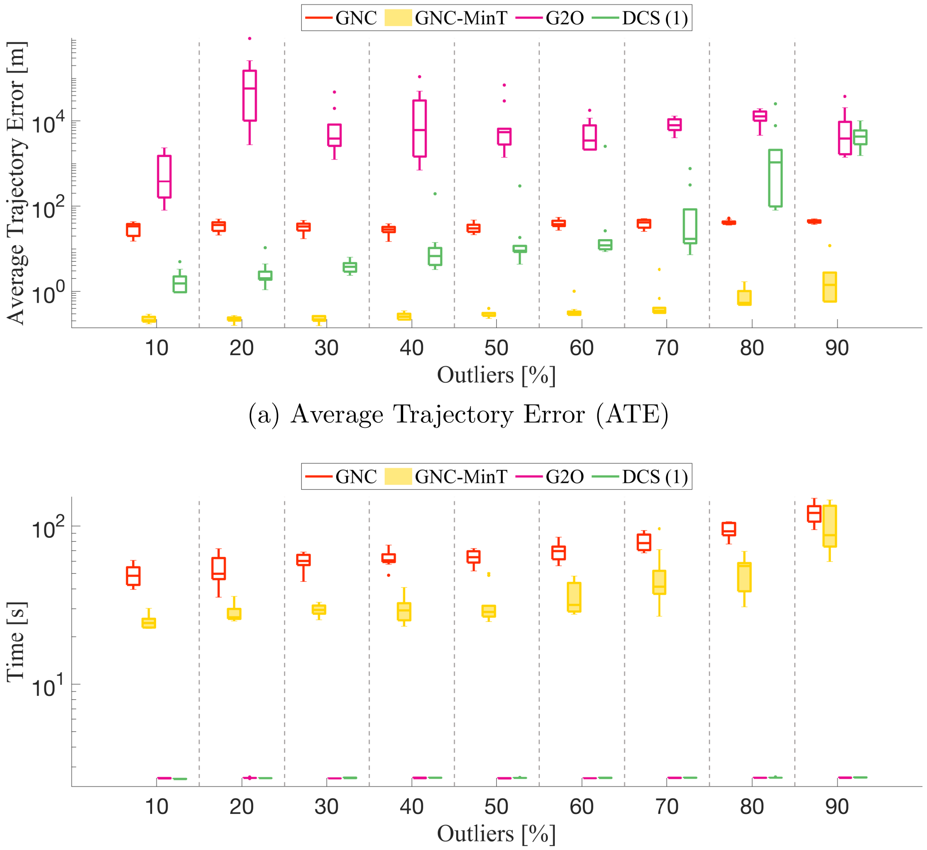
Figure 2-7: **2D SLAM (CSAIL).** Average Trajectory Error (ATE) and running time of the proposed algorithms compared to state-of-the-art techniques on the CSAIL dataset for increasing outliers.



(a) Average Trajectory Error (ATE)

(b) Running Time

Figure 2-8: **3D SLAM (Sphere).** Average Trajectory Error (ATE) and running time of the proposed algorithms compared to state-of-the-art techniques on a synthetic Sphere dataset for increasing outliers.



(a) Average Trajectory Error (ATE)

(b) Running Time

Figure 2-9: **3D SLAM (Garage).** Average Trajectory Error (ATE) and running time of the proposed algorithms compared to state-of-the-art techniques on the Garage dataset for increasing outliers.

Chapter 3

Perception Fault Detection and Identification

In this chapter we move our attention from individual perception problems to the system level. We start by formulating the problem of fault detection and identification in perception systems (Section 3.1). We then introduce the *diagnostic graph*, a graphical model that captures the diagnostic information available in the perception system and dependencies between the different perception modules (Section 3.2). We leverage the diagnostic graph to develop algorithms for fault detection and identification (Section 3.3) and study the fundamental limits of the approach (Section 3.4). We demonstrate, through simulations, that our algorithms can detect and identify faults in state-of-the-art perception systems (Section 3.5) and conclude the chapter with a discussion of related work (Section 3.6).

3.1 Fault Detection and Identification

3.1.1 Perception System: Modules and Outputs

A perception system $\mathcal{S}$ comprises a finite set of unique interconnected *modules* $\mathcal{M} = \{m_1, m_2, \ldots, m_{|\mathcal{M}|}\}$; for instance, the perception system of a self-driving car may include modules for lane detection, camera-based object detection, LiDAR-based motion estimation, ego-vehicle localization, etc. Each module $m \in \mathcal{M}$ produces a finite set of *outputs*, and each output is produced by a single module. For instance, the lane detection module may produce an estimate of the 3D location of the lane boundaries, while the pedestrian detection module may produce an estimate of the pose and velocity of pedestrians in the surroundings. Some of these outputs provide inputs for other perception modules, while other are the outputs of the perception system and feed into other systems (*e.g.,* to planning and control). The set of modules' outputs are disjoint (*i.e.,* each output is produced by a single module), and the set of all outputs is denoted by $\mathcal{O}$. We model the perception system as a graph of modules and outputs.

Definition 20 (Perception System)**.** *A perception system $\mathcal{S}$ is a directed graph $\mathcal{S} = (\mathcal{M} \cup \mathcal{O}, \mathcal{E})$, where the set of nodes $\mathcal{M} \cup \mathcal{O}$ describes modules and outputs in the system, while the set of edges $\mathcal{E}$ describes which module produces or consumes a certain output. In particular, an edge $(m_i, o_j) \in \mathcal{E}$ with $m_i \in \mathcal{M}$ and $o_j \in \mathcal{O}$ models the fact that module m_i produces output o_j. Similarly, and edge $(o_j, m_i) \in \mathcal{E}$ with $o_j \in \mathcal{O}$ and $m_i \in \mathcal{M}$ models the fact that module m_i uses output o_j.*

We treat each module as a *black-box* and remain agnostic to the algorithms they implement. This allows our framework to generalize to complex perception systems, possibly including a combination of classical and data-driven methods.

While we will consider more complex examples of perception systems in the experimental section, Fig. 3-1 shows a simple example of perception system to ground the discussion. The system comprises three modules: a *LiDAR-based* obstacle detector, a *camera-based* obstacle detector, and a *sensor fusion* module. Both the LiDAR-based and the camera-based obstacle detectors generate a set of obstacles detected in the environment, namely, the *LiDAR obstacles* and *camera obstacles.* The sensor fusion algorithm combines the two sets of obstacles to produce a new set of objects, called *fused obstacles.*

Remark 21 (Modules vs. Outputs)**.** *Our system model treats modules and outputs as separate nodes. This is convenient for two reasons. First, fault identification at*

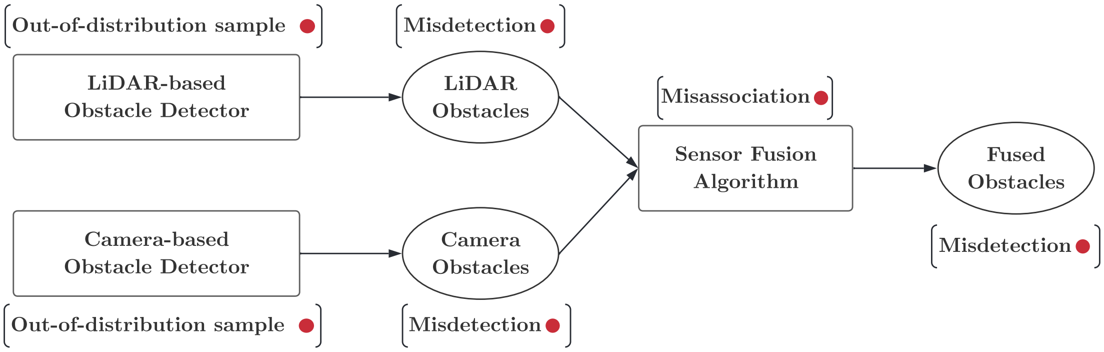


Figure 3-1: A simple example of a perception system including 3 modules (rectangles) and 3 outputs (circles). Modules are connected by edges describing which module produces or consumes a given output. The failure modes of each module (resp. output) are represented by red dots. The LiDAR-based and the Camera-based obstacle detection modules are subject to the *out-of-distribution sample* failure mode (*i.e.*, they saw a sample far from the training dataset), which might result in *misdetections* (*e.g.*, missing obstacles) in their respective outputs. The sensor fusion module is subject to the *misassociation* failure mode, which might result in *misdetections* in its output.

the modules and outputs may serve different purposes: output fault identification is more useful at runtime to identify unreliable information from the perception system and prevent accidents; module fault identification is typically more informative for designers and regulators. Second, in practical applications we can rarely measure *if a module is failing (indeed developing algorithms that can "self-diagnose" their failures is an active area of research, see work on* certifiable algorithms *[121]). On the other hand, we can directly measure the outputs of the modules and develop diagnostic tests to check if an output is plausible and consistent with other outputs in the system.*

3.1.2 Fault Detection and Fault Identification

Each module in $\mathcal{S}$ might fail at some point, jeopardizing the system performance or even its safety. In particular, each module $m \in \mathcal{M}$ is assumed to have a set of failure modes. A failure of a module is the deviation from its intended behavior.While the list of failures can include any software and hardware failures, In this thesis we particularly focus on failures of the intended functionality. For example, a neural-network-based camera-based object detection module might experience the failure mode "out-of-distribution sample" when it processes an input image, which indicates that while the module's code executed successfully, the resulting detection is expected to be incorrect.

Similarly, each output $o \in \mathcal{O}$ has an associated set of failure modes. A failure of an output is an error of its value. For instance, the output of the camera-based object detector might experience a "mis-detection" failure mode if it fails to detect an object, or a "mis-classification" failure mode if the object is detected but misclassified. A module's failure mode typically causes a failure in one of its outputs. Examples of failure modes are given in Fig. 3-1. For each module and output, the figure lists a potential failure mode: for instance, the LiDAR-based obstacle detection output may fail if it misdetects an obstacle, while the sensor fusion module may fail it it incorrectly associates the input obstacles.

Definition 22 (Failure Modes). *At each time instant, the i-th failure mode* $f_i \in \{\text{INACTIVE}, \text{ACTIVE}\} \cong \{0, 1\}$ *is either ACTIVE (also* 1*) if such failure is occurring, or INACTIVE (also* 0*). A module or an output is* failing *if at least one of its failure modes is ACTIVE. Similarly, a system is* failing *if at least one of its modules or outputs is failing. If we stack the status (ACTIVE/INACTIVE) of all failure modes into a single binary vector, the* fault state vector $\boldsymbol{f} \in \{0, 1\}^{N_f}$ *(where* N_f *is the number of failure modes), then* $\boldsymbol{f}$ *is all zeros if there are no faults, or has entries equal to ones for the active failure modes.*

The goal of this section is then to address the following problems:

Fault Detection decide whether the system is working in nominal conditions or whether a fault has occurred (*i.e.,* infer if there is at least an active failure mode in $\boldsymbol{f}$);

Fault Identification identify the specific failure mode the system is experiencing (*i.e.,* infer which failure mode is active in $\boldsymbol{f}$).

Fault detection is the easiest between the two problems, as it only requires specifying the presence of at least a fault, without specifying which modules or outputs are incorrect. Mathematically, this reduces to identifying whether the unknown vector $\boldsymbol{f}$ has at least an entry equal to 1. Fault identification goes one step further by explicitly indicating the set of active failure modes. Mathematically, this reduces to identifying exactly which entries of the unknown vector $\boldsymbol{f}$ are equal to 1. Identifying which module is faulty is particularly important to inform regulators (*e.g.,* to trace the steps that that led to an accident caused by an autonomous vehicle) and system designers (*e.g.,* to highlight modules that are likely to fail and require further development). Moreover, not all faults are equally problematic: for instance, a failure in localizing a car in the opposite lane of a divided highway is less consequential that failing to detect a pedestrian in front of the car. Note that solving fault identification implies a solution for fault detection (*i.e.,* whenever we declare one or more modules to be faulty, we essentially also detected there is a failure), hence in the rest of this paper we focus on the design of a monitoring system for fault identification.

Remark 23 (Assumptions and Terms of Use)**.** *We assume that the potential failure modes of the system are known to the system designer. In practice, these can be discovered using some form of hazard analysis, such as Failure Modes and Effects Analysis (FMEA) [122] or Fault tree analysis (FTA) [123]. Moreover, we can always add a generic "unknown failure mode" to capture any failure modes of a module or output that we cannot characterize, so this assumption is not restrictive. We also remark that our monitoring system's objective is to diagnose potential failures, while it does not prescribe what are the actions that need to be taken in response to each failure (*e.g., *whether to stop the car, provide a warning to the passenger, etc.), which is failure and system-dependent. An investigation on how to respond to or mitigate failures is left to future work.*

3.2 Modeling Fault Identification with Diagnostic Graphs

This section develops a framework to model fault identification problems in perception systems. In the previous section we have discussed how the goal is to identify the set of active failure modes associated to modules and outputs in a system. Here we introduce the concept of *diagnostic graphs* to study fault identification: diagnostic graph will allow developing fault identification algorithms (Section 3.3) and understanding fundamental limits (Section 3.4).

The intuition is that in a perception system we can perform a number of *diagnostic tests* that check the validity of the output of certain modules. For instance, we can compare the outputs of different modules to ensure they are consistent (*e.g.,* compare the obstacles detected by the LiDAR-based obstacle detection against the camera-based obstacle detection), or inspect that the output of a certain module respects certain requirements (*e.g.,* the vision-based ego-motion module is tracking a sufficient number of features). Then, we can model these checks as edges in a bipartite graph, the *diagnostic graph*, which can be used for fault identification. In the following, we formalize the notions of diagnostic tests and diagnostic graphs.

3.2.1 Diagnostic Tests

In our fault identification framework, the system is equipped with a set of *diagnostic tests* that can (possibly unreliably) provide diagnostic information about the state of a subset of failure modes. Each diagnostic test is a function $t : \mathbb{S} \rightarrow \{\text{PASS}, \text{FAIL}\}$, where $\mathbb{S} \subseteq \{1, \ldots, N_f\}$ is a subset of the failure modes that the test is checking, called the *scope* of the test, and the test returns a value $z \in \{\text{PASS}, \text{FAIL}\} \cong \{0, 1\}$, called the *outcome* of the test. A diagnostic test returns PASS (also denoted with 0) if there is no active failure mode in its scope, FAIL (also denoted with 1) otherwise. In general, tests can be *unreliable*, meaning that they can both fail to detect active failures or incorrectly detect failures as active (*i.e.,* false alarms). Each diagnostic test can be tuned to be more or less conservative, which affects the number of false alarms and missed failures (*i.e.,* precision and recall) of fault detection and identification, providing additional flexibility to practitioners.

While in the experimental section we will describe more complex tests (and provide an open-source framework to easily code new tests), it is instructive to consider a simple test between the outputs of the LiDAR-based obstacle detection and the camera-based obstacle detection in Fig. 3-2. The test in Fig. 3-2 compares the two sets of objects detected by the two detectors; whenever an inconsistency arises, the test returns FAIL. However, if both detectors are subject to the same failure, *e.g.,* they both misdetect an obstacle, the test might still pass, thus exhibiting unreliable behavior. We remark that a single test does not suffice for fault identification: for instance, if the test in Fig. 3-2 fails, we can only conclude that one of the two detectors had a failure (or that the test was a false alarm); therefore, we typically need to collect a number of tests and perform some inference process to draw conclusions about which modules failed. The collection of the outcomes of multiple diagnostic tests is called a *syndrome.*

Definition 24 (Syndrome)**.** *Assuming we have N_t diagnostic tests, the vector collecting the test outcomes $\boldsymbol{z} \in \{\text{PASS}, \text{FAIL}\}^{N_t}$ is called a* syndrome*.*

In the following, we describe how to mathematically model the relation between the failure modes and the test outcomes; this will be instrumental in solving the inverse problem of identifying the failure mode from a given syndrome. We provide a deterministic and a probabilistic model for the tests below.

Deterministic Tests. Deterministic diagnostic tests encode the set of possible test outcomes, by establishing a deterministic relation between failure modes in the test's scope and the test outcome. We discuss potential models for deterministic diagnostic test below.

Ideally we would like the test to return FAIL if and only if at least one of the failure modes in its scope is active. This leads to the definition of a "Deterministic OR" test.

Definition 25 (Deterministic OR)**.** *A diagnostic test $t(\boldsymbol{f}_{\text{scope}(t)})$ is a deterministic OR if its test outcome z is*

$$z = \begin{cases} \text{PASS} & \textit{if } \|\boldsymbol{f}_{\text{scope}(t)}\|_1 = 0 \\ \text{FAIL} & \textit{otherwise} \end{cases} \tag{3.1}$$

This kind of tests can be hard to implement in practice. For example, imagine a diagnostic test that compares the output of two object classifiers: if one of them produces a wrong label, it is easy to detect there is a failure; however, if both classifiers are trained on similar data and both report the incorrect label there is no way to detect the failure. In this case, the test outcome is unreliable. The following definition introduces a type of unreliable test.

Definition 26 (Deterministic Weak-OR [44], [124])**.** *A test* $t(\boldsymbol{f}_{\text{scope}(t)})$ *is a deterministic Weak-OR if its test outcome* z *is*

$$z = \begin{cases} \text{FAIL} & \textit{if } 0 < \|\boldsymbol{f}_{\text{scope}(t)}\|_1 < |\text{scope}(t)| \\ \text{PASS } \textit{or} \text{ FAIL} & \textit{if } \|\boldsymbol{f}_{\text{scope}(t)}\|_1 = |\text{scope}(t)| \\ \text{PASS} & \textit{otherwise} \end{cases} \tag{3.2}$$

This kind of tests is consistent with the tests used in [124]. Intuitively, a "Deterministic Weak-OR" may return PASS even if all failure modes are active, since the test might fail to detect an inconsistency if all faults are consistent with each others (again, think about two object classifiers failing in the same way). Even though the Weak-OR test may pass or fail when all failure modes are active, its outcome remains deterministic.

Finally, an even weaker type of deterministic test is what we call the Deterministic Weaker-OR (this is the easiest test to implement in practice).

Definition 27 (Deterministic Weaker-OR)**.** *A diagnostic test* $t(\boldsymbol{f}_{\text{scope}(t)})$ *is a Deterministic Weaker-OR if its test outcome* z *is*

$$z = \begin{cases} \text{PASS } \textit{or} \text{ FAIL} & \textit{if } \|\boldsymbol{f}_{\text{scope}(t)}\|_1 > 0 \\ \text{PASS} & \textit{if } \|\boldsymbol{f}_{\text{scope}(t)}\|_1 = 0 \end{cases} \tag{3.3}$$

In other words, the test is designed to pass in nominal conditions (*i.e.,* when no failure mode is active), but it can have arbitrary outcomes otherwise.

The types of deterministic tests presented above are not the only possible deterministic tests. Other examples include, for instance, diagnostic tests that fail to detect specific sets of failure modes. Deterministic tests can be designed using formal methods tools or certifiable perception algorithms [3], [87], [89],[1] see also Remark 29 below.

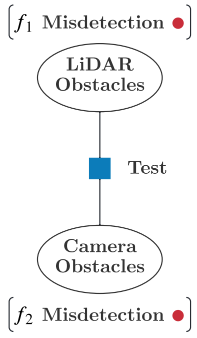


Figure 3-2: A test comparing two outputs, LiDAR Obstacles and Camera Obstacles

Scope		Test outcome z	
f_1	f_2	OR	Noisy-OR
0	0	0	0 with prob. $(1-p_{a,1})(1-p_{a,2})$ 1 with prob. $p_{a,1}+p_{a,2}-p_{a,1}p_{a,2}$
0	1	1	0 with prob. $(1-p_{a,1})(1-p_{d,2})$ 1 with prob. $p_{a,1}+p_{d,2}-p_{a,1}p_{d,2}$
1	0	1	0 with prob. $(1-p_{d,1})(1-p_{a,2})$ 1 with prob. $p_{d,1}+p_{a,2}-p_{d,1}p_{a,2}$
1	1	1	0 with prob. $(1-p_{d,1})(1-p_{d,2})$ 1 with prob. $p_{d,1}+p_{d,2}-p_{d,1}p_{d,2}$

Table 3.1: Table of possible outcomes for the Deterministic OR and the probabilistic Noisy-OR version of a test with scope f_1 and f_2.

Probabilistic Tests. Deterministic tests might not capture the complexity of real world diagnostic tests. Most practical tests are likely to incorrectly detect faults (*i.e.,* produce false positive) or fail to detect faults (*i.e.,* produce false negatives) with some probability. For this reason, in this thesis, we also allow for an arbitrary probabilistic relationship between test outcomes and failure modes in the test scope.

A simple-yet-expressive way to formalize a probabilistic test is to use what we call a "Noisy-OR" model. In particular, the Noisy-OR model represents the probability of a diagnostic test outcome as a conditional probability distribution over the failure modes in its scope $\Pr(z \mid \boldsymbol{f}_{\text{scope}(t)})$ [2] as defined below.

Definition 28 (Noisy-OR [125])**.** *A diagnostic test $t(\boldsymbol{f}_{\text{scope}(t)})$ is a probabilistic Noisy-OR if its test outcome z follows*

$$\Pr(z = \text{PASS} \mid \boldsymbol{f}_{\text{scope}(t)}) = \prod_{i \in \text{scope}(t)} \Pr(z = \text{PASS} \mid f_i) \tag{3.4}$$

[1] Certifiable perception algorithms are a class of model-based perception algorithms that provide a soundness certificate at runtime, allowing one to directly measure the presence (or absence) of certain failure modes, see [89], [121].

[2] We denote with $\Pr(A)$ the probability of event A, and with $\Pr(A \mid B)$ the conditional probability of A given B.

where $\Pr(z \mid f_i)$ *denotes the conditional probability of the test outcome (PASS/FAIL) conditioned on the status (ACTIVE/INACTIVE) of the failure mode* f_i. *Clearly,* $\Pr(z = \text{FAIL} \mid \boldsymbol{f}_{\text{scope}(t)}) = 1 - \Pr(z = \text{PASS} \mid \boldsymbol{f}_{\text{scope}(t)})$.

Now suppose each test has a probability $p_{d,i}$ of correctly identifying failure f_i (detection probability), and a probability $p_{a,i}$ of false alarm for f_i. Exploiting the fact that $f_i \in \{0, 1\}$, we can write Eq. (3.4) as:

$$\Pr(z = \text{PASS} \mid \boldsymbol{f}_{\text{scope}(t)}) = \prod_{i \in \text{scope}(t)} (1 - p_{d,i})^{f_i} (1 - p_{a,i})^{1 - f_i} \tag{3.5}$$

An example of probabilistic test outcome is given in Table 3.1.

Similarly to the deterministic case, the Noisy-OR model is not the only possible model. However, Section 3.5 shows that this model is particularly effective in modeling fault identification problems in practice. In Section 3.3, we discuss how to learn the probabilities involved in probabilistic tests (*i.e.,* $p_{d,i}$ and $p_{a,i}$ in Eq. (3.5)) given a training dataset, and how to use the test outcomes to infer the most likely failure modes. Towards that goal, we need to group diagnostic tests into a suitable graph structure, called a *diagnostic graph*, which we present in the following section.

We conclude this section with a remark.

Remark 29 (From diagnostic tests to fault identification)**.** *The diagnostic tests we introduced in this section are not dissimilar from the typical diagnostic tests or watchdogs considered in prior work or used by practitioners. Our goal here is to formalize these tests and use the test outcomes to infer the most likely set of system-wide failures. In this sense, our fault identification framework is designed to capitalize on (rather than replace) existing diagnostic tools used in practice. For example the detection mechanism proposed by Liu and Park [126], which is based on the idea of projecting the 3D LiDAR points onto camera images, and then checking whether objects detected from LiDAR and images match each other, can be formulated as a diagnostic tests with the camera and LiDAR misdetection in its scope, such that the test outcome is the output of the algorithm in [126]. Also, out-of-distribution detection based on epistemic uncertainty,* e.g., *[127], can be formulated as a diagnostic tests* *with the module's "out-of-distribution sample" failure mode in its scope, such that the test outcome is FAIL if the estimated uncertainty is above a threshold. Finally, while not explored In this thesis, diagnostic tests can also return a severity measure, which can be either discrete (*e.g., *low, medium, high) or continuous (*e.g., *real number in* $[0, 1]$*). Once the active failure modes are identified, the severity of each failure mode can be determined using some operation on the collected severity (*e.g., *max, weighted sum, etc.).*

3.2.2 Diagnostic Graph

A diagnostic graph is a structure defined over a perception system and has the goal of describing the diagnostic tests (as well as more general relations among failure modes) and their scope. We provide a formal definition below.

Definition 30 (Diagnostic Graph)**.** *A* diagnostic graph *is a bipartite graph* $\mathcal{D} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ *where the nodes are split into* variable nodes $\mathcal{V}$*, corresponding to the failure modes in the system, and* relation nodes $\mathcal{R}$*, where each relation* $\phi_k(\boldsymbol{f}) \in \mathcal{R}$ *is a function over a subset of failure modes* $\boldsymbol{f}$*. Then an edge in* $\mathcal{E}$ *exists between a failure mode* $f_i \in \mathcal{V}$ *and a relation* $\phi_k \in \mathcal{R}$*, if* f_i *is in the scope of the relation* ϕ_k *(*i.e., *if the variable* f_i *appears in the function* ϕ_k*).*

Relations capture constraints among the variables induced by the test outcomes or from prior knowledge we might have about the failure modes. We describe the two main types of relations below and for each we describe their implementation in the deterministic and probabilistic case.

Definition 31 (Test-driven Relations)**.** *A* test-driven relation ϕ_k *describes whether —for a test* t_k*— a given set of failure mode assignments might have produced a certain test outcome* z_k*. More formally, for a deterministic test* t_k*, a test-driven relation is a boolean function:*

$$\phi_k(\boldsymbol{f}) = \phi(\boldsymbol{f}_{\text{scope}(t_k)}; z_k) = \mathbb{1}\left[z_k = t\left(\boldsymbol{f}_{\text{scope}(t_k)}\right)\right] \tag{3.6}$$

where $\mathbb{1}$ *is the indicator function that returns* 1 *if the condition is satisfied or* 0 *otherwise. The function Eq.* (3.6) *checks if an assignment of failure modes* $\boldsymbol{f}$ *may have produced the test outcome* z_k *and where the notation* $\phi_k(\boldsymbol{f}) = \phi(\boldsymbol{f}_{\text{scope}(t_k)}; z_k)$ *clarifies that the function* ϕ_k *only involves a subset of failure modes* $\boldsymbol{f}_{\text{scope}(t_k)}$ *(the ones in the scope of test* t_k*) and depends on the (given) test outcome* z_k*. Similarly, for a probabilistic test* t_k*, a test-driven relation is a real-valued function:*

$$\phi_k(\boldsymbol{f}) = \phi(\boldsymbol{f}_{\text{scope}(t_k)}; z_k) = \Pr(z_k | \boldsymbol{f}_{\text{scope}(t_k)}) \tag{3.7}$$

which returns the likelihood of the test outcome z_k *given an assignment* $\boldsymbol{f}$*.*

Definition 32 (A Priori Relations)**.** *An* a priori relation *describes whether a given set of failure modes is plausible, considering a priori knowledge about the system. More formally, in the deterministic case, an a priori relation is a boolean function* $\phi_k(\boldsymbol{f})$ *that returns* 1 *if the assignment of* $\boldsymbol{f}$ *is plausible or* 0 *otherwise. Similarly, in the probabilistic case, an a priori relation is a real-valued function* $\phi_k(\boldsymbol{f})$ *that returns the likelihood of a given assignment* $\boldsymbol{f}$*.*

In the following we will denote the set of Test-driven Relations as $\mathcal{R}_{\text{test}}$ while the set of A Priori Relations as $\mathcal{R}_{\text{prior}}$. Therefore, $\mathcal{R} = \mathcal{R}_{\text{test}} \cup \mathcal{R}_{\text{prior}}$.

The aim of a priori relationship is to model the interactions between different modules, which includes interaction between modules of the same subsystem (*e.g.,* object detection) or interactions between different subsystems (*e.g.,* object detection and localization modules). While we have provided several examples of diagnostic tests in the previous section, we now provide examples of a priori relations. For instance, in the deterministic case, some failure modes of a module can be mutually exclusive (*e.g.,* "too many outliers", "not enough features" in the Lidar-based ego-motion estimation) or one can imply another (*e.g.,* if a module is experiencing an "out-of-distribution sample" failure mode, then its outputs will have at least an active failure mode). Not all relations are deterministic, for example in Fig. 3-3, the failure modes of the sensor fusion algorithm may have a complex probabilistic relationship with the failure modes of the lidar and camera obstacles failure modes. Note that the main difference between test-driven relations and a priori relations is that the former provides a measurable test outcome, while the latter relies on a priori knowledge about the system (*i.e.,* no outcome is measured).

We elucidate on the notion of diagnostic graph with two examples below.

Example 1: Multi-sensor Obstacle Detection. Consider the perception system in Fig. 3-1. We can associate a diagnostic graph to the system where the variable nodes of the diagnostic graph are the failure modes of modules and outputs in the system. The diagnostic graph, shown in Fig. 3-3, also includes two diagnostic tests and a priori relations encoding input/output relationship between modules and outputs. Each diagnostic test compares a pair of outputted obstacles, namely LiDAR obstacles and camera obstacles (with failures f_4 and f_5), and camera obstacles and fused obstacles (with failures f_4 and f_6).

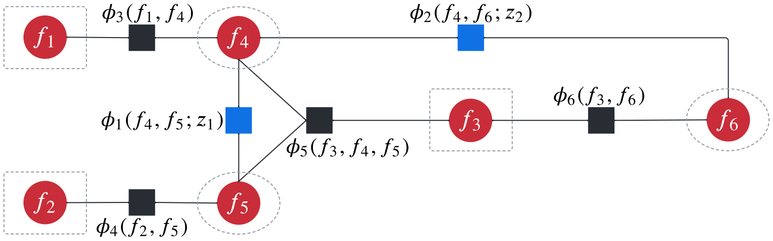


Figure 3-3: A diagnostic graph for the perception system example in Fig. 3-1. Red circles represent variable nodes (failure modes) while squares represent relations. Test-driven Relations are shown in blue, while a priori relations are shown in black.

Example 2: LiDAR-based Ego-motion Estimation. We provide a second example that also includes singleton diagnostic tests (having a single failure mode in their scope) and includes explicit tests over modules. The example consists of a LiDAR-based odometry system that computes the relative motion between consecutive LiDAR scans using feature-based registration, see *e.g.,* [1], [2]. The system $\mathcal{S}$ comprises two modules, a *feature extraction* module and a *point-cloud registration* module, as depicted in Fig. 3-4(left). The feature extraction module extracts 3D point *features* from input LiDAR data, while the point-cloud registration module uses the features to estimate the relative pose between two consecutive LiDAR scans. Suppose that the feature extraction module is based on a deep neural network and that it can experience an “out-of-distribution sample” failure, which causes the corresponding output to potentially experience “too-many outliers” or “few features” failures. Similarly, the module *point-cloud registration* can experience the failure “suboptimal solution”, which leads its outputs, the relative pose, to experience a “wrong relative pose” failure. Fig. 3-4(right) shows a diagnostic graph for the system. The system is equipped with four diagnostic tests. A diagnostic test (t_1) detects if the failure mode “few features” is active by checking the cardinality of the feature set. If the point-cloud registration module is a certifiable algorithm [121], we can attach a diagnostic test (t_2) to the point-cloud registration module that uses the module’s certificate to detect if the module is experiencing a “suboptimal solution” failure. Another diagnostic test (t_3) detects if the relative pose is wrong by checking that the relative pose does not exceed some meaningful threshold given the vehicle dynamics. Finally, another test (t_4) checks if under the computed relative pose, the feature extractor has “too many outliers”. This can be achieved by counting the number of features that are correctly aligned after applying the estimated relative pose. The diagnostic graph also contains a priori relations encoding constraints on the input/output relationships.

Temporal Diagnostic Graph

So far, we have considered a diagnostic graph as a representation of the diagnostic information available at a specific instant of time (*e.g.,* the examples above include tests and relations involving the behavior of modules and outputs at a certain time instant). However, perception systems evolve over time, and considering the temporal dimension offers further opportunities for fault identification, *e.g.,* by monitoring the temporal evolution of the outputs.

Suppose we have a collection of diagnostic graphs $\mathcal{T} = \{\mathcal{D}^{(t)}, \ldots, \mathcal{D}^{(t+K)}\}$, collected over and interval of time. We could think of *stacking* these diagnostic graphs, into a new *temporal* diagnostic graph $\mathcal{D}^{[K]}$. The temporal graph preserves the failure mode, relations and edges of each sub-graph $\mathcal{D}^{(k)} \in \mathcal{T}$. However, since $\mathcal{D}^{[K]}$ includes outputs produced at multiple time instants, we can also augment the graph to include

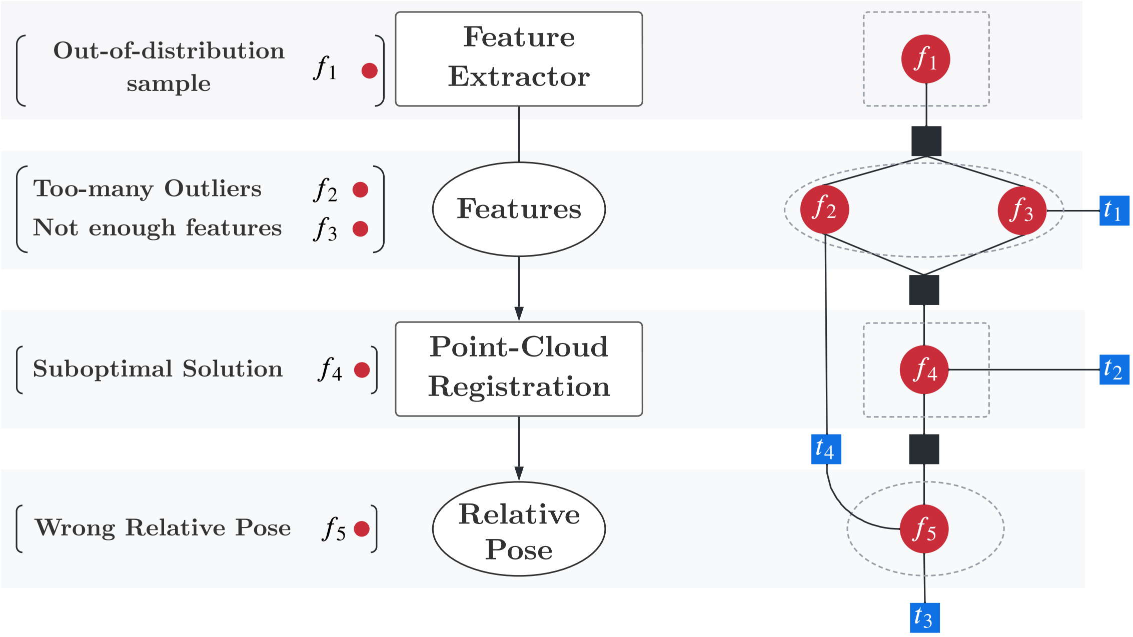


Figure 3-4: *(Left)* Example of the LiDAR-based ego-motion estimation system $\mathcal{S}$. The system is composed by two modules (rectangles), each producing one output (circles). *(Right)* The corresponding diagnostic graph, where red circles represent variable nodes (failure modes) while squares represent relations (test-driven Relations in blue, a priori relations in black).

temporal diagnostic tests and temporal relationships. For example, we might check that an obstacle does not disappear from the scene (unless it goes out of the sensor field of view), or that the pose of the ego-vehicle does not change too much over time. As we will see, the use of temporal diagnostic graph leads to slightly improved fault identification performance. An example of temporal diagnostic graph is given in Fig. 3-5.

The algorithms and results presented in the rest of this paper apply to both regular and temporal diagnostic graph, unless specified otherwise.

Remark 33 (Temporal Diagnostic Tests)**.** *Temporal diagnostic tests are used to monitor the evolution of the system over time. For example the Timed Quality Temporal Logic in [128] can be implemented with a temporal diagnostic test that spans multiple $\mathcal{D}^{(t)}$'s. More specifically, the test example considered in [128] requires that "At every time step, for all the objects in the frame, if the object class is* cyclist *with probability more than* 0.7*, then in the next* 5 *frames the same object should still be classified as a*

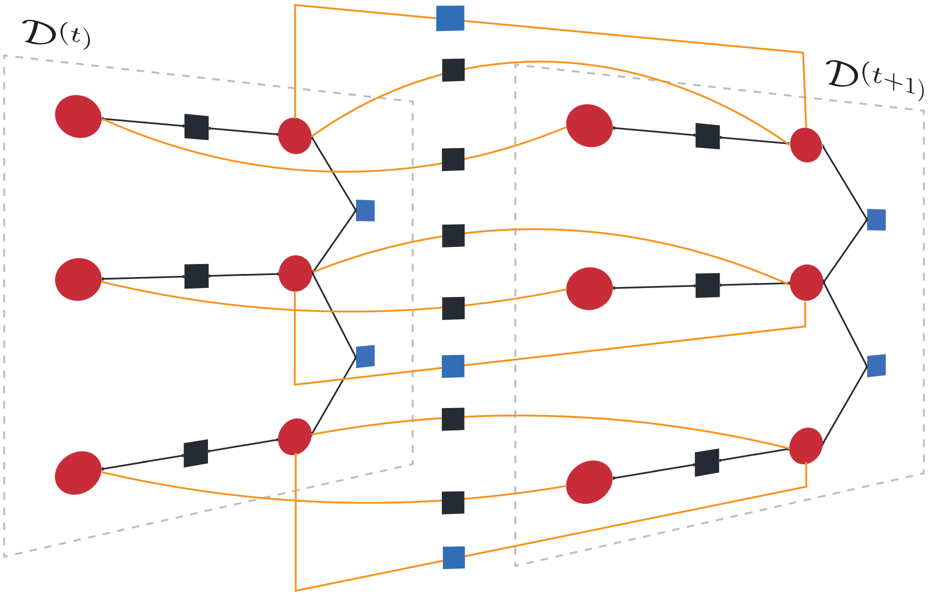


Figure 3-5: Example of Temporal Diagnostic Graph composed by two identical sub-graphs. We added temporal relations (both test-driven and a priori) between the two sub-graphs.

cyclist with probability more than 0.6*". This can be modeled as a diagnostic test that spans* 5 *diagnostic graphs and that returns FAIL if the predicate is false.*

3.3 Algorithms for Fault Identification

This section shows how to perform fault identification over a diagnostic graph. In particular, we present algorithms to infer which failure modes are active, given a syndrome. We study fault identification with deterministic tests in Section 3.3.1 and then extend it to the probabilistic case in Section 3.3.2. Finally, we present a graph-neural-network approach for fault identification in Section 3.3.3.

3.3.1 Inference in the Deterministic Model

In the deterministic case, our inference algorithm looks for the smallest set of active failure modes that explains a given syndrome. In Section 3.4, we will show that such approach is guaranteed to correctly identify the faults as long as the tests provide a sufficient level of redundancy, an insight we will formalize through the notion of "diagnosability".

Looking for the smallest set of active failures that explains the test outcomes (and more generally, the relations) in a diagnostic graph can be formulated as the following optimization problem (given a syndrome $\boldsymbol{z}$):

$$\begin{aligned} \underset{\boldsymbol{f} \in \{0,1\}^{N_f}}{\text{minimize}} \quad & \|\boldsymbol{f}\|_1 \\ \text{subject to} \quad & \phi_k(\boldsymbol{f}_{\text{scope}(t_k)}; z_k) = 1, \quad i = 1, \ldots, N_t, \\ & \phi_j(\boldsymbol{f}) = 1, \qquad\qquad\quad\;\; j = 1, \ldots, N_r, \end{aligned} \tag{D-FI}$$

where $\phi_k(\boldsymbol{f}_{\text{scope}(t_k)}; z_k)$ are the N_t test-driven relations in the diagnostic graph, while $\phi_j(\boldsymbol{f})$ are the N_r a priori relations in the graph. In words, Eq. (D-FI) looks for binary decisions (ACTIVE/INACTIVE) for the failure modes $\boldsymbol{f}$, and looks for the smallest set of faults (the objective $\|\boldsymbol{f}\|_1$ counts the number of ACTIVE failure modes) such that the faults satisfy the relations in the diagnostic graph. Eq. (D-FI) is our *Deterministic Fault Identification* algorithm.

The optimization in Eq. (D-FI) can be solved using standard computational tools from Integer Programming [129] or Constraint Satisfaction Programming [130]. While integer programming is better suited to find the solution to the minimization problem, constraint programming also allows finding all the solutions in the feasible set. The choice between the two depends on the application and the expression for the relations. In our experiments, we solve it using Integer Programming. We remark that while Integer Programming is NP complete, our problems typically only involve tens to hundreds of failure modes, and can be solved efficiently in practice.

The model presented above is generic and valid for any deterministic test and a priori relations. In the following, we provide an example to ground the discussion and show how to instantiate the optimization problem in practice.

Example 3: Deterministic Inference with Weaker-OR and Module-Output Relations. We consider a diagnostic graph with Deterministic Weaker-OR tests. Moreover, for a priori relations, we assume that whenever the output of a module has a failure, then also the module itself must have at least an active failure mode.

This is also the setup we use in our experiments in Section 3.5.

In Weaker-OR diagnostic tests, the PASS outcome is unreliable, meaning that if a test returns PASS it might have 0 or more failure modes active in its scope. However, when it the test returns FAIL, we know there must be at least one failure mode active. This can be easily enforced in the optimization by imposing the constraint:

$$\|\boldsymbol{f}_{\text{scope}(t_i)}\|_1 \geq 1 \qquad \forall t_i \in \{1, \dots, N_t\} \text{ such that } z_i = \text{FAIL},$$

We then have to enforce the relation that if an output has an active failure mode, then the module that produced it must have at least one active failure mode as well. Towards this goal, let $\mathcal{F}(o_i) \subseteq \{1, \dots, N_f\}$ be the set of failure modes associated to outputs of module m_i and $\mathcal{F}(m_i)$ be the set of failure modes associated to m_i; then the a priori relation can be enforced via the constraint:

$$\|\boldsymbol{f}_{\mathcal{F}(m_i)}\|_1 \geq \frac{1}{|\mathcal{F}(o_i)|} \|\boldsymbol{f}_{\mathcal{F}(o_i)}\|_1$$

Intuitively, when there is no active failure in the outputs (*i.e.,* $\|\boldsymbol{f}_{\mathcal{F}(o_i)}\|_1 = 0$) the constraint is trivially satisfied, while when there is at least an output failure (*i.e.,* $\|\boldsymbol{f}_{\mathcal{F}(o_i)}\|_1 > 0$) then $\|\boldsymbol{f}_{\mathcal{F}(m_j)}\|_1$ is forced to be at least 1. The resulting optimization problem finally becomes:

$$\begin{aligned} \underset{\boldsymbol{f} \in \{0,1\}^{N_f}}{\text{minimize}} \quad & \|\boldsymbol{f}\|_1 \\ \text{subject to} \quad & \|\boldsymbol{f}_{\text{scope}(t_i)}\|_1 \geq 1 \qquad \forall t_i \in \{1, \dots, N_t\} \text{ such that } z_i = \text{FAIL}, \\ & \|\boldsymbol{f}_{\mathcal{F}(m_i)}\|_1 \geq \frac{1}{|\mathcal{F}(o_i)|} \|\boldsymbol{f}_{\mathcal{F}(o_i)}\|_1 \qquad \forall m_i \in \mathcal{M}. \end{aligned} \tag{3.8}$$

3.3.2 Inference in the Probabilistic Model

This section shows how to use the formalism of factor graphs to find the most likely active failure modes that explain a given syndrome in a diagnostic graph with probabilistic tests.

Factor graphs are a powerful class of probabilistic graphical models. Probabilistic graphical models allow describing relationships between multiple variables using a concise language. In particular, they describe joint or conditional distributions over a set of unknown variables and a set of known observations, and can be used to infer the values of the unknown variables. In this work we limit ourselves to factor graphs over discrete (binary) variables. We start from the definition of a factor graph.

Definition 34 (Factor Graph [131])**.** *A factor graph is a bipartite graph $F = (\mathcal{V}, \Phi, \mathcal{E})$ consisting of a set $\mathcal{V}$ of variable nodes, a set Φ of factor nodes, and a set $\mathcal{E} \subseteq \mathcal{V} \times \Phi$ of edges having one endpoint at a variable node and the other at a factor node. Let $\mathcal{N}(\phi)$ the set of variables to which a factor node ϕ is connected, then, the factor graph defines a family of distributions that factorize according to*

$$\mu(\boldsymbol{f} \mid \boldsymbol{z}) = \frac{1}{Z} \prod_{\phi \in \Phi} \phi(\boldsymbol{f}_{\mathcal{N}(\phi)}; \boldsymbol{z}) \tag{3.9}$$

where the normalization factor Z, also known as the partition function*, ensures that $\mu(\boldsymbol{f})$ is a valid distribution:*[3]

$$Z(\boldsymbol{z}) = \sum_{\boldsymbol{f}} \prod_{\phi \in \Phi} \phi(\boldsymbol{f}_{\mathcal{N}(\phi)}; \boldsymbol{z}) \tag{3.10}$$

The notation $\phi(\boldsymbol{f}_{\mathcal{N}(\phi)}; \boldsymbol{z})$ emphasizes the fact that each factor is a function of a subset $\boldsymbol{f}_{\mathcal{N}(\phi)}$ of the failure modes $\boldsymbol{f}$, for given observed $\boldsymbol{z}$.

The factor graph F and the diagnostic graph $\mathcal{D}$ have a similar structure. In fact we can choose the set of variables $\mathcal{V}$ in the factor graph to be the same as the set of variables in the diagnostic graph, namely the set of failure modes. Then, we can choose the set of factors Φ to be the relations $\mathcal{R}$ of $\mathcal{D}$, and the set of edges to be the same. Therefore, for a given diagnostic graph $\mathcal{D}$, it is easy to devise the corresponding factor graph as:

$$\mu(\boldsymbol{f} \mid \boldsymbol{z}) = \frac{1}{Z} \prod_{\phi_k \in \mathcal{R}_{\text{test}}} \phi_k(\boldsymbol{f}_{\text{scope}(t_k)}; z_k) \prod_{\phi_j \in \mathcal{R}_{\text{prior}}} \phi_j(\boldsymbol{f}_{\mathcal{N}(\phi_j)}) \tag{3.11}$$

[3] The notation $\sum_{\boldsymbol{f}}$ means "sum over all possible values of $\boldsymbol{f}$."

where we have simply observed that the probability distributions induced by the relations in the diagnostic graph naturally factorize into factors, each one corresponding to a (test-driven or a priori) relation in the diagnostic graph.

Maximum a Posteriori Inference. Given a factor graph, a natural question to ask is what is the most likely assignment of variables that maximizes the probability distribution induced by the factor graph (*e.g.,* in our case, this is the most likely set of faults in the system). This leads to the concept of *maximum a posteriori* (MAP) inference, which —given a factor graph and a syndrome $\boldsymbol{z}$— looks for the most likely variables $\boldsymbol{f}^{\star}$, that maximize the posterior distribution:

$$\boldsymbol{f}^{\star} = \underset{\boldsymbol{f} \in \{0,1\}^{N_f}}{\arg\max} \; \mu(\boldsymbol{f} \mid \boldsymbol{z}) \tag{FG-FI}$$

Computing a MAP estimate is known to be NP-hard for general factor graphs [132], therefore it is common to use approximate methods. In our experiments we used belief propagation(Sec. 3 in [133]) to solve the MAP inference, which finds the optimal solution for tree-structured factor graphs, and is known to empirically return good approximations for the MAP estimate in general factor graphs.

Learning the Factor Graph Parameters. While in the deterministic case we know the expression of the relations ϕ_k, in the probabilistic case the probabilistic tests might depend on unknown parameters, *cf.* the expression in Eq. (3.5) that requires specifying the parameters $p_{d,i}$ (probability that a fault is not detected) and $p_{a,i}$ (probability of a false alarm). There are several paradigms to learn the factor graph parameters. In our experiments we use a method called *structured support vector machine (SSVM)* or *maximum margin learning* (Sec. 19.7 in [134]).

3.3.3 Graph Neural Networks for Fault Identification

The factor graph framework introduced in the previous section learns the factor graph parameters from training data, and then performs maximum a posteriori inference at runtime for fault identification. In this section, we propose a learning-based framework that is also trained on a dataset, but then learns directly how to predict which failure mode is active at runtime. In particular, we use *Graph Neural Networks* (GNN) to learn to identify active faults in a diagnostic graph.

GNNs provide a general framework for learning using graph-structured data, and have empirically achieved state-of-the-art performance in many tasks such as node classification, link prediction, and graph classification [135]. The fault identification problem considered In this thesis can be phrased as a *node classification* problem. In node classification, given a undirected graph $G = (\mathcal{V}, \mathcal{E})$ where each node $i \in \mathcal{V}$ has an (unknown) label y_i, the objective is to learn a representation vector $\boldsymbol{e}_i$ of node i such that label y_i can be predicted as a function of the node embeddings $\boldsymbol{e}_i$.

In the following, we recall common GNN architectures (Section 3.3.3) and then we discuss how to transform our diagnostic graph into a structure that can be fed to a GNN to predict active faults (Section 3.3.3).

Graph Neural Network Preliminaries

A GNN is an extension of recurrent neural networks that operates on graph-structured data. GNNs are based on the concept of *neural message passing* in which real-valued vector messages are exchanged between nodes of a graph —not dissimilarly to the belief propagation we used in Section 3.3.2— but were the messages (and node updates) are built using differentiable functions encoded as neural networks. To understand the basic idea of neural message passing consider an undirected graph $G = (\mathcal{V}, \mathcal{E})$. At the beginning, each node is assigned a feature vector $\boldsymbol{e}_i^{(0)}$ for each $i \in \mathcal{V}$. Then, during each message-passing iteration $k = 1, 2, \ldots$, the embedding $\boldsymbol{e}_i^{(k)}$ is updated by aggregating the embeddings of node i's neighborhood $\mathcal{N}(i)$

$$\begin{aligned} \boldsymbol{e}_i^{(k+1)} &= \text{update}\left(\boldsymbol{e}_i^{(k)}, \text{aggregate}\left(\left\{\boldsymbol{e}_j^{(k)} \ \mid \ j \in \mathcal{N}(i)\right\}\right)\right) && (3.12) \\ &= \text{update}\left(\boldsymbol{e}_i^{(k)}, \boldsymbol{a}_i^{(k)}\right) && (3.13) \end{aligned}$$

Where aggregate$(\cdot)$ and update$(\cdot)$ are two learned differentiable functions (*i.e.,* neural networks). At each iteration k, the aggregate$(\cdot)$ function takes the embeddings of node i's neighbors and generates a *message* $\boldsymbol{a}_i^{(k)}$. Then, the update$(\cdot)$ function combines the message with the previous embedding of node i, generating the new embedding of node i. The final embedding is obtained by running the neural message passing for K iterations. Finally, the node label is predicted by a learned differentiable function of the node embeddings:

$$y_i = \text{READOUT}(\boldsymbol{e}_i^{(K)}) \tag{GNN-FI}$$

The literature on GNN offers a number of potential choices for the update$(\cdot)$ and aggregate$(\cdot)$ functions. We review four popular choices below.

Graph Convolutional Networks (GCNs). One of the most popular graph neural network architectures is the graph convolutional network (GCN) [136]. The GCN model implements the update and aggregate function as:

$$\boldsymbol{e}_i^{(k+1)} = \sigma\left(\boldsymbol{W}^{(k+1)} \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{\boldsymbol{e}_j^{(k)}}{\sqrt{|\mathcal{N}(i)||\mathcal{N}(j)|}}\right) \tag{3.14}$$

where $\boldsymbol{W}^{(k+1)}$ is a trainable weight matrix and $\sigma(\cdot)$ is a nonlinear activation function. Note that Eq. (3.14) can also be written in a matrix form

$$\boldsymbol{E}^{(k+1)} = \sigma\left(\hat{\boldsymbol{P}}\boldsymbol{E}^{(k)}\boldsymbol{W}^{(k+1)}\right)$$

where $\hat{\boldsymbol{P}} = \hat{\boldsymbol{D}}^{-\frac{1}{2}}(\boldsymbol{A} + \mathbf{I})\hat{\boldsymbol{D}}^{-\frac{1}{2}}$, the matrix $\boldsymbol{A}$ is the adjacency matrix of the original graph, and $\hat{\boldsymbol{D}}$ is its diagonal degree matrix.

Graph Convolutional Network via Initial residual and Identity mapping (GCNII). The GCN is affected by the *over-smoothing* problem [137], where after several iterations of GNN message passing, the nodes' embeddings become very similar to each another; over-smoothing prevents the use of deeper GNN models, which in turn prevents the GNN from leveraging longer-term dependencies in the graph. To solve this problem, Chen *et al.* [138] propose the GCNII, where the update of the embedding vectors becomes:

$$\boldsymbol{E}^{(k+1)} = \sigma\left(\left((1-\alpha_k)\hat{\boldsymbol{P}}\boldsymbol{E}^{(k)} + \alpha_k\boldsymbol{E}^{(0)}\right)\left((1-\beta_k)\mathbf{I} + \beta_k\boldsymbol{W}^{(k)}\right)\right) \tag{3.15}$$

and where α_k and β_k are two hyper-parameters. GCNII improves on the basic GCN by adding a smoothed representation $\hat{\boldsymbol{P}}\boldsymbol{E}^{(k)}$ with an initial residual connection to the first layer $\boldsymbol{E}^{(0)}$, and adds an identity mapping to the k-th weight matrix $\boldsymbol{W}^{(k)}$.

Graph Sample and Aggregate (GraphSAGE). GraphSAGE is another approach for node classification [139]. The aggregate function takes the form

$$\boldsymbol{a}_i^{(k+1)} = \sigma\left(\boldsymbol{W} \cdot g\left(\{\boldsymbol{e}_j^{(k)} : j \in \mathcal{N}(i) \cup \{i\}\}\right)\right) \tag{3.16}$$

where $g(\cdot)$ is an aggregator function like the element-wise mean or max pooling. Then, the update function is a function over the concatenation of the old embedding and the message $\boldsymbol{a}_i^{(k)}$:

$$\boldsymbol{e}_i^{(k+1)} = \sigma\left(\boldsymbol{W}[\boldsymbol{e}_i^{(k)}, \boldsymbol{a}_i^{(k+1)}]\right) \tag{3.17}$$

Graph Isomorphism Network (GIN). The Graph Isomorphism Network (GIN) [140] is defined by the following aggregation function

$$\boldsymbol{a}_i^{(k+1)} = (1 + \epsilon^{(k+1)})\boldsymbol{e}_i^{(k)} + \sum_{j \in \mathcal{N}(i)} \boldsymbol{e}_j^{(k)} \tag{3.18}$$

where $\epsilon^{(k)}$ is a trainable (or fixed) parameter. The update function in GIN is

$$\boldsymbol{e}_i^{(k+1)} = \zeta^{(k+1)}(\boldsymbol{a}_i^{(k+1)}) \tag{3.19}$$

where $\zeta(\cdot)$ is also a neural network.

From Diagnostic Graphs to Graph Neural Networks

In order to apply GNNs to our diagnostic graph $\mathcal{D}$, we need to convert $\mathcal{D} = (\mathcal{V}_\mathcal{D}, \mathcal{R}_\mathcal{D}, \mathcal{E}_\mathcal{D})$ into an undirected graph $G = (\mathcal{V}_G, \mathcal{E}_G)$. Towards this goal, we take the set of nodes $\mathcal{V}_G$ to be *both* the set of failure modes and diagnostic test outcomes. Note that we add the diagnostic test outcomes as nodes in the graph since this allows attaching the test outcomes as features to these nodes. For each test t_k we form a clique[4] involving

[4] A clique is a subset of vertices of an undirected graph such that every two distinct vertices in the clique share an edge.

the set of nodes in the test's scope and the variable corresponding to the test z_k, namely the set $\text{scope}(t_k) \cup \{z_k\}$. We then form another clique for each a priori relation $\phi_j \in \mathcal{R}_{\text{prior}}$ using the set of failure modes $\mathcal{N}(\phi_k)$ connected to ϕ_j. For example if we have a factor $\phi(f_1, f_2; z_2)$ we add the following (undirected) edges to $\mathcal{E}_G$: (f_1, f_2), (f_1, z_2), (f_2, z_2). We attach a feature vector to each node in the graph. For the test nodes, we use a one-hot encoding describing the test outcome as node feature. For the module nodes, we use the failure probability (computed from the training data) as node features. We provide more details on the node features in Section 3.5.

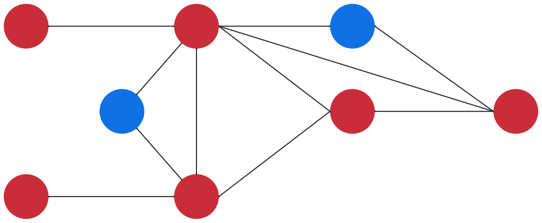


Figure 3-6: Example of conversion of the diagnostic graph in Fig. 3-3 into an undirected graph.

Learning to Identify Active Faults. In order to train the GNN to identify active faults, we use a supervised learning approach. In particular, we use a softmax classification function and negative log-likelihood training loss, which is available in standard libraries, such as PyTorch [141].

Remark 35 (Curate a balanced dataset)**.** *Datasets collected using real-world operation of modern perception systems often contain comparably less failure than nominal data. In practice the dataset can be curated with one (or more) of the following:*

- *Collecting real data from scenarios that have triggered a failure in the past (*e.g.*, resulted in the autopilot being disengaged by the safety driver/tester).*
- *Use of a simulator with a falsification engine that searches for scenarios where the perception experienced a failure (*e.g.*, [142]).*
- *Use of an offline perception system that uses both past and future information to generate the world model (*e.g.*, [143]); such perception systems are more accurate, giving the possibility of identifying failure-prone scenarios.*

All strategies (scenario-based, falsification-based and offline perception) are extensively used in industry and effective in generating a balanced dataset. If this approach is not possible and only an unbalanced dataset is available, one common approach to deal with unbalanced dataset is to use undersampling*, which consists of down-sizing the majority class by removing observations at random until the dataset is balanced. However, undersampling can induce a bias in the posterior probabilities. This is a well known problem in literature, Dal Pozzolo* et al. *[144] study the problem and propose a methodology to reduce such biases. We envision this framework to be used with fairly balanced datasets.*

3.4 Fundamental Limits

Given a diagnostic graph it is natural to ask if there is a maximum number of failure modes that can be correctly identified as active. In other words, for a given system, can we guarantee that our algorithms are able to correctly identify all faults? Under which conditions? We answer these questions in this section, where we introduce the concept of *diagnosability.* We discuss the deterministic case (*i.e.,* where the tests are assumed to be unreliable deterministic tests) in Section 3.4.1. Then we obtain more general guarantees for the probabilistic case (which also apply to our learning-based algorithms) in Section 3.4.2.

3.4.1 Deterministic Diagnosability

In this section, we assume diagnostic graphs with deterministic relations and present theoretical results on the maximum number of faults that can be correctly identified. Towards this goal, we borrow and extend results from fault identification in multiprocessor systems [44], which were partially presented in our previous work [124]. In particular, Lemma 37 and Theorem 38 below are a direct application of results in [44], while the others are our extensions.

We start with the definition of deterministic diagnosability.

Definition 36 (κ-diagnosability [44], [124])**.** *A diagnostic graph $\mathcal{D}$ is* κ-diagnosable *if, given any syndrome, all active failure modes can be correctly identified, provided that the number of active failure modes in the system does not exceed κ.*

The idea behind κ-diagnosability is that the number of failures that can be correctly identified is an intrinsic property of a system and its diagnostic graph, and somehow it measures if the system has enough redundancy to unambiguously identify the cause of certain failures.

Example 4: Multi-sensor Obstacle Detection (Fig. 3-1 and Fig. 3-3). Consider the example in Fig. 3-1 and assume that an output fails if and only if the module producing it fails. Also assume that the sensor fusion algorithm does not necessarily fail if its inputs are wrong (thus removing $\phi_5(f_3, f_4, f_5)$, or setting it to be always TRUE). If both diagnostic tests behave like Deterministic ORs, and they both return FAIL, we would not know if the state of the failure mode $(f_1, f_2, f_3, f_4, f_5, f_6)$ was $(0, 1, 0, 0, 1, 0)$, $(0, 1, 1, 0, 1, 1)$, $(1, 0, 1, 1, 0, 1)$, $(1, 1, 0, 1, 1, 0)$ or $(1, 1, 1, 1, 1, 1)$. In fact, all these failures would generate the same syndrome (FAIL,FAIL). However, if we impose that the maximum number of active failure mode is 2 (i.e., $\kappa = 2$), the number of feasible candidates drops to only one, namely $(0, 1, 0, 0, 1, 0)$. In other words, if we have at most two failures in the system, the two tests would allow us to uniquely identify which failure mode is active without any doubt.

After defining the notion of diagnosability in Definition 36, we are left with the question: can we develop an algorithm to compute the diagnosability of a certain diagnostic graph? It has been noted in [145] that a system is κ-diagnosable if the set of possible syndromes uniquely encodes the set of active failure modes. Such observation is formalized by the following lemma.

Lemma 37 (Diagnosability and Syndromes)**.** *Let* syndrome($\mathcal{A}$) *be the set of all possible syndromes produced by a set of active failure modes $\mathcal{A} \subseteq \{1, \ldots, N_f\}$. A diagnostic graph $\mathcal{D}$ is κ-diagnosable if and only if, given any $\mathcal{A}_1, \mathcal{A}_2 \subseteq \{1, \ldots, N_f\}$, such that $|\mathcal{A}_1|, |\mathcal{A}_2| \leq \kappa$ (with $\mathcal{A}_1 \neq \mathcal{A}_2$), we have* $\text{syndrome}(\mathcal{A}_1) \cap \text{syndrome}(\mathcal{A}_2) = \emptyset$*.*

The lemma intuitively establishes that for a κ-diagnosable system, two different sets of κ faults must produce different syndromes, such that for any given syndrome, there is no ambiguity on which set of active failure modes generated it, and we can perform fault identification without any mistake.

Lemma 37 suggests an algorithmic way to check if a diagnostic graph is κ-diagnosable, which however requires checking every subset of failure modes of cardinality up to κ (and their syndromes). In the following, we refine the result, showing that, under technical assumptions, one can directly compute the diagnosability by only looking at the topology of the diagnostic graph.

Theorem 38 (Characterization of κ-diagnosability [146])**.** *Let*

$$H(f) \doteq \{t \mid t \in \{1, \ldots, N_t\}, f \in \text{scope}(t)\}$$

be the set of tests involving a failure mode f, and let

$$\Gamma(f) \doteq \bigcup_{t \in H(f)} \text{scope}(t) \setminus \{f\}$$

be the set of failure modes that share a test with f. Also define $\Gamma(X) \doteq \bigcup_{f \in X} \Gamma(f) \setminus X$ the extension of Γ to a set of failure modes. Now assume that all tests follow the Deterministic Weak-OR model and have scope of cardinality 2. Then $\mathcal{D}$ is κ-diagnosable if all the following conditions are satisfied:

i. $\kappa \leq (N_f - 1)/2$

ii. $\kappa \leq \min_{i \in \{1,\ldots,N_f\}} |H(f_i)|$

iii. for each $q \in \mathbb{N}$ with $0 \leq q < \kappa$, and each $X \subset \{1, \ldots, N_f\}$ with $|X| = N_f - 2\kappa + q$ we have $|\Gamma(X)| > q$

Theorem 38 also shows that the diagnosability of a system depends on the amount of redundancy in the systems and how well the tests are able to capture it. The connection is particularly visible in condition *(iii)*: for each set of possible set X of active failure modes (of appropriate size), there must be a sufficient number the tests, that —using information coming from different modules/outputs— give an opinion on the state of the failure modes in X.

Let us now move our attention to temporal diagnostic graphs. Denote with $\kappa(\mathcal{D})$ the maximum value of κ for the diagnostic graph $\mathcal{D}$. Then the following result characterizes the diagnosability of temporal diagnostic graphs.

Theorem 39 (Diagnosability in Temporal Diagnostic Graphs)**.** *Let $\mathcal{D}^{[K]}$ a temporal diagnostic graph built by stacking a set of K regular diagnostic graphs $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(K)}$. Then $\kappa(\mathcal{D}^{[K]}) \geq \min_{i \in \{1,\ldots,K\}} \kappa(\mathcal{D}^{(i)})$.*

As an immediate result we have the following corollary, which characterizes the diagnosability of "homogeneous" temporal diagnostic graph, obtained by stacking multiple identical diagnostic graphs over time.

Corollary 40 (Diagnosability in Homogeneous Temporal Diagnostic Graphs)**.** *The diagnosability of the composition of identical diagnostic graph is monotonically increasing.*

This means that by stacking diagnostic graphs over time, we have the opportunity to increase the diagnosability, without any risk of harming it.

3.4.2 Probabilistic Diagnosability

The deterministic notion of κ-diagnosability introduced in the previous section imposes a strong condition on $\mathcal{D}$, as it requires that any syndrome unequivocally encodes all possible configurations of failure modes. When the tests are probabilistic, such a condition becomes too stringent: intuitively, since with some probability each test can produce different outcomes it is unlikely that Lemma 37 will be satisfied for any $\kappa > 0$. In other words, κ-diagnosability deals with the worst case over all possible test outcomes, which becomes too conservative when every outcome is possible (with some probability). For this reason, in this section, we extend the definition of diagnosability to deal with the case where the diagnostic graph includes probabilistic tests.

Towards defining a probabilistic notion of diagnosability, we introduce the *Hamming distance* $h(\boldsymbol{f}, \boldsymbol{f}')$ between two binary vectors $\boldsymbol{f}$ and $\boldsymbol{f}'$ as follows:

$$h(\boldsymbol{f}, \boldsymbol{f}') = \sum_{i=1}^{N_f} \mathbb{1}[f_i \neq f_i'] \tag{3.20}$$

where $\mathbb{1}$ is the indicator function. Assuming that $\boldsymbol{f}$ is the binary vector describing the active failures in the system, and that $\boldsymbol{f}'$ is an estimated vector of the fault states, the Hamming distance simply counts the number of *mis-identified faults*. We are now ready to introduce the following probabilistic definition of diagnosability.

Definition 41 ((Probably Approximately Correct) PAC-Diagnosability)**.** *Consider a fault identification algorithm $\Psi_{\mathcal{D}}$ applied to a diagnostic graph $\mathcal{D}$. The diagnostic graph $\mathcal{D}$ is (γ, p)-PAC-diagnosable under $\Psi_{\mathcal{D}}$, if, for some $1 \leq \gamma \leq N_f$*

$$\Pr_{(\boldsymbol{z},\boldsymbol{f})\sim distF} \left[h\left(\Psi_{\mathcal{D}}(\boldsymbol{z}), \boldsymbol{f}\right) \leq \gamma\right] \geq p \tag{3.21}$$

where distF is the joint distribution of potential failures and test outcomes.

This definition simply says that a given fault identification algorithm applied to the diagnostic graph $\mathcal{D}$ is (γ, p)-PAC-diagnosable if it expected to make less than γ mistakes with probability at least p. We observe that Definition 41 depends on the diagnostic graph, but also on the fault identification algorithm.

Clearly, since the outcome of the tests is a random variable, so is the Hamming distance $h(\Psi_{\mathcal{D}}(\boldsymbol{z}), \boldsymbol{f})$. Therefore, we can define its expected value as:

$$h_{\text{dist}F}(\Psi_{\mathcal{D}}) = \mathbb{E}_{(\boldsymbol{z},\boldsymbol{f})\sim\text{dist}F}\left[h(\Psi_{\mathcal{D}}(\boldsymbol{z}), \boldsymbol{f})\right]$$

This quantity is the number of mistakes that the fault identification algorithm $\Psi_{\mathcal{D}}$ is expected to make. Let us suppose we have a dataset $\mathcal{W}$ of i.i.d. samples of the underlying faults distribution dist$F$. Let

$$\hat{h}_{\mathcal{W}}(\Psi_{\mathcal{D}}) = \frac{1}{|\mathcal{W}|} \sum_{(\boldsymbol{z},\boldsymbol{f})\in\mathcal{W}} h(\Psi_{\mathcal{D}}(\boldsymbol{z}), \boldsymbol{f}) \tag{3.22}$$

be the empirical number of mistakes the fault identification algorithm $\Psi_{\mathcal{D}}$ makes on $\mathcal{W}$. For instance, if we are given a (labeled) dataset $\mathcal{W}$ describing the system execution, with the corresponding ground truth failure modes states $\boldsymbol{f}$, we can test our algorithm $\Psi_{\mathcal{D}}$ and calculate the empirical number of mistakes $\hat{h}_{\mathcal{W}}(\Psi_{\mathcal{D}})$ it makes. Then, we can use the following result to bound the expected number of mistakes our algorithm will make in expectation over all future scenarios.

Theorem 42 (Fault Identification Error Bound). *Consider a dataset $\mathcal{W}$ of i.i.d. samples of the underlying faults distribution $distF$, and a fault identification algorithm $\Psi_{\mathcal{D}}$ over $\mathcal{D}$. Then, for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$:*

$$h_{distF}(\Psi_{\mathcal{D}}) \leq \hat{h}_{\mathcal{W}}(\Psi_{\mathcal{D}}) + N_f \sqrt{\frac{\log(2/\delta)}{2|\mathcal{W}|}} \tag{3.23}$$

The previous result essentially says that the expected number of mistakes the algorithm $\Psi_{\mathcal{D}}$ makes stays close to the empirical mean $\hat{h}_{\mathcal{W}}(\Psi_{\mathcal{D}})$, and the distance from the empirical mean gets smaller when the training dataset gets larger (*i.e.,* for larger $|\mathcal{W}|$), but gets larger for larger number of failure modes (*i.e.,* for larger N_f). The following corollary easily follows.

Corollary 43 (Characterization of PAC-diagnosability). *For a given dataset $\mathcal{W}$ of i.i.d. samples of the underlying faults distribution $distF$, and a fault identification algorithm $\Psi_{\mathcal{D}}$ over $\mathcal{D}$, the diagnostic graph $\mathcal{D}$ is (γ, p)-PAC-diagnosable with p satisfying the following inequality:*

$$p \geq 1 - 2e^{-2\left({(\gamma - \hat{h}_{\mathcal{W}})}/{N_f}\right)^2 |\mathcal{W}|} \tag{3.24}$$

Remark 44 (Diagnosability over Subgraphs). *Given a diagnostic graph $\mathcal{D}$, we might be interested in running fault identification algorithms over a subgraph $\bar{\mathcal{D}} \subseteq \mathcal{D}$. Analyzing the diagnosability of certain subgraphs of $\mathcal{D}$ might suggest weaknesses of the perception pipeline. For example the system might have sufficient redundancy to be able to correctly identify the faults in the obstacle detection subgraph with low errors and high confidence, but might lack of redundancy to detect and identify faults in the* *traffic light recognition.*

Similarly, to avoid diagnostic tests with very low reliability (which might increase the false alarm rate), or to reduce the computational workload of executing tests, we may want to use a subset of the available diagnostic tests. Diagnosability is a handy tool to help the designer identify the most effective diagnostic tests. To minimize the number of diagnostic tests, a good rule of thumb is to choose a subset of diagnostic tests that covers the most failure modes, to avoid making the diagnostic graph overly dependent on a priori relationships. Then, more diagnostic tests can be added if they increase the diagnosability of the system. New diagnostic tests can be selected using some form of exhaustive (e.g., *branch-and-bound), greedy algorithms or heuristic search.*

The construction of the diagnostic graph relies on expert knowledge, in case of limited knowledge, it might occur that the diagnostic graph contains wrong or missing edges. In the case of wrong (extra) edges, the probabilistic diagnostic graph is generally able to learn to ignore wrong edges (*i.e.,* the values of the relation converge to zero). In the case of missing edges, however, the system designer must rely on diagnosability to recognize that the performance is unacceptable. In such cases, however, it is possible to add extra-edges (over-approximate the diagnostic graph) and leverage the training process to filter out the incorrect edges. It is worth noting that it is generally straightforward to add edges between diagnostic tests and failure modes because the diagnostic tests are either designed to detect a specific failure mode (*e.g.,* uncertainty estimation [127]) or uses a subset of the data produced by the system (*e.g.,* consistency-based tests) to detect the failure, so it is connected to any failure mode that affects the outputs used.

3.5 Experimental Evaluation

This section shows that diagnostic graphs are an effective model to detect and identify failures in complex perception systems. In particular, we show that the proposed monitors (i) outperform baselines in terms of fault identification accuracy, (ii) allow detecting failures and provide enough notice to prevent accidents in realistic test scenarios, and (iii) run in milliseconds, adding minimal overhead.

We test our runtime monitors in several scenarios, specifically designed to stress-test the perception system. The scenarios are simulated using the LGSVL Simulator [147], an open-source autonomous driving simulator. The simulator also generates ground-truth data, *e.g.,* ground-truth obstacles and active failure modes, and seamlessly connects to the perception system through the Cyber RT Bridge interface [147]. We apply our monitors to a state-of-the-art perception system. In particular, we use Baidu's Apollo Auto [148] version 7 [149]. Baidu's Apollo is an open-source, sate-of-the-art, autonomous driving stack that includes all the relevant functionalities for level 4 autonomous driving.

Section 3.5.1 provides more details about Apollo Auto and its perception system. Section 3.5.2 describes the diagnostic tests we design for Apollo Auto's perception system. Section 3.5.3 discusses implementation details for the proposed monitors. Section 3.5.4 describes our test scenarios. Section 3.5.5 provides quantitative fault detection and identification results, including an ablation study of the different GNN architectures. Appendix P provides qualitative results and discussion for a key test scenario.

3.5.1 Apollo Auto

Baidu's Apollo Auto [148] uses a flexible and modularized architecture for the autonomy stack based on the sense-plan-act framework. The stack includes seven subsystems: (i) the *localization subsystem* provides the pose of the ego vehicle; (ii) the *high-definition map* provides a high-resolution map of the environment, including lanes, stop signs, and traffic signs; (iii) the *perception subsystem* processes sensory information (together with the localization data) and creates a world model; (iv) the *prediction subsystem* predicts future evolution of the world state; (v) the *motion planning subsystems* and (vi) the *routing subsystem* generate a feasible trajectory for the ego vehicle, and finally, (vii) the *control subsystem* generates low-level control signals to move the vehicle. In our experiments, we focus on the *perception subsystem*, to which we apply our runtime monitors. In the following, we briefly review the key aspects of the Apollo Auto perception system.

Apollo Auto Perception System

Apollo Auto's perception system is tasked with the detection and classification of obstacles and traffic lights.[5] The perception module is capable of using multiple cameras, radars, and LiDARs to recognize obstacles. There is a submodule for each sensor modality, that independently detects, classifies, and tracks obstacles. The results from each sub-module are then fused using a probabilistic sensor fusion algorithm.

Obstacle Detection. Obstacles such as cars, trucks, bicycles, are detected using an array of radars, LiDARs, and cameras. Each obstacle is represented by a 3D bounding-box in the world frame, the class of the object, a confidence score, together with other sensor-specific information (*e.g.,* the velocity of the obstacle). Each sensor is processed as follows:

Camera: The camera-based obstacle detection network is based on the monocular object detection SMOKE [151] and trained on the Waymo Open Dataset [152]. The network predicts 2D and 3D information about each obstacle, and then a post-processing step predicts the 3D bounding box of each obstacle by minimizing the reprojection error of available templates for the predicted obstacle class;

LiDAR: The LiDAR-based obstacle detection network, called Mask-Pillars is based on PointPillars [153], but enhanced with a residual attention module to improve detection in case of occlusion;

Radar: Apollo Auto uses directly the obstacles detections reported by the radar (assumed to have an embedded detector [154]), that are post-processed to be transformed to the world frame.

[5] Note that our monitors can be also applied to other perception-related subsystems, such as the localization and high-definition map subsystem, see [150] for an example.

Vehicle Configuration

The simulated vehicle is a Lincoln MKZ with one Velodyne VLS-128 LiDAR, one front-facing camera with a field-of-view of 50°, one front-facing telephoto camera (pointed 4° upwards) for traffic light detection and recognition, one Continental ARS 408-21 front-facing radar, GPS, and IMU.

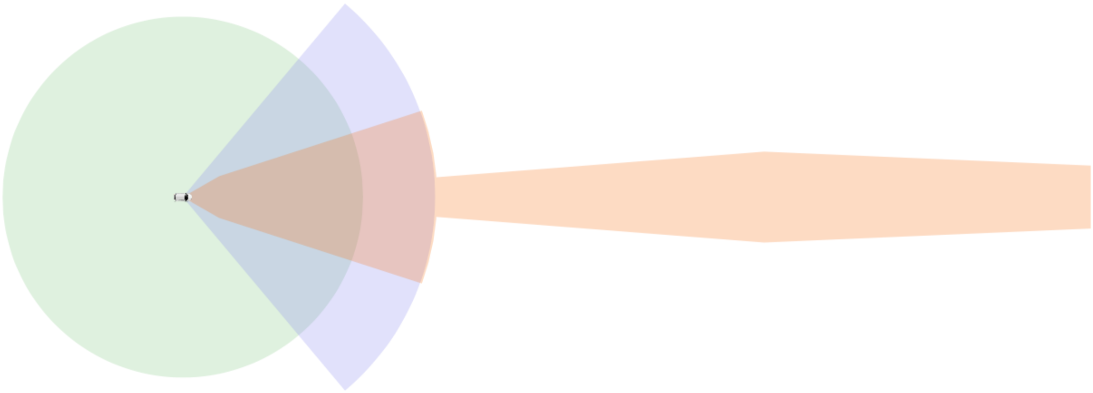


Figure 3-7: **Vehicle sensor field-of-view (FOV).** LiDAR FOV is shown in green, the camera FOV in blue and the radar FOV in orange.

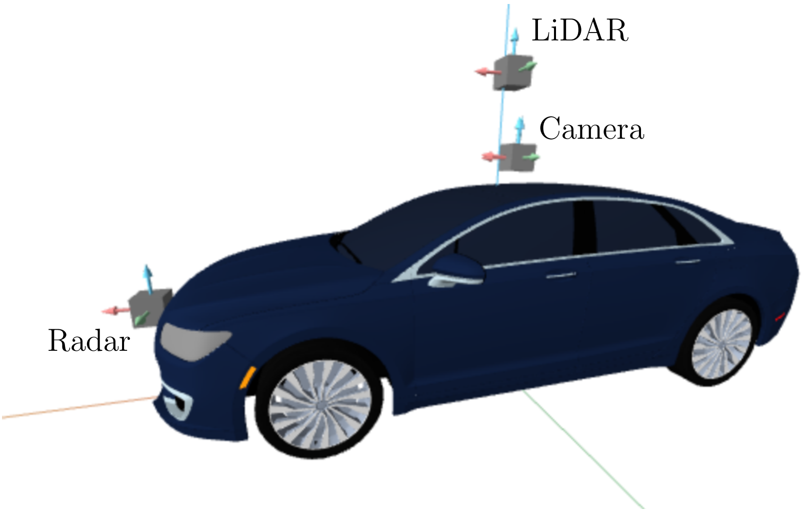


Figure 3-8: **Vehicle configuration.** LiDAR and Camera are mounted on the roof of the vehicle, while the radar is mounted on the front bumper.

We ran the Baidu's Apollo AV stack on a computer with an Intel i9-9820X

(4.1 GHz) processor, 64 GB of memory and two NVIDIA GeForce RTX 2080Ti. The simulator ran on a computer with 11th Generation Intel i7-11700F (4.8 GHz) processor, 16 GB of memory, and an NVIDIA GeForce RTX 3060. The two computers were connected using a Gigabit Ethernet cable.

3.5.2 Diagnostic Graph

We focused our attention on the obstacle detection pipeline. The system we aim to monitor, together with the failure modes considered, is shown in Fig. 3-9 The system is composed of four modules:

- Lidar-based Obstacle detector, based on a deep learning algorithm, subject to *out-of-distribution sample* failure mode;
- Camera-based Obstacle detector, based on a deep learning algorithm, subject to *out-of-distribution sample* failure mode;
- Radar-based Obstacle detector subject to *misdetection* failure mode;
- Sensor Fusion subject to *misassociation* failure mode.

Each module produces a set of detected obstacles. We identified three failure modes for each set of detected obstacles:

- *misdetection*: the module detected a ghost obstacle or is missing an obstacle in the scene;
- *misposition*: the module detected the obstacle correctly, but its position is incorrect (*i.e.,* more than 2.5m error in our tests);
- *misclassification*: the module detected the obstacle correctly but the obstacle's semantic class is incorrect.

We equipped the obstacle detection system with 18 diagnostic tests. For each pair of modules' outputs, namely (Lidar, Camera), (Radar, Camera), (Lidar, Sensor Fusion), (Radar, Sensor Fusion), (Lidar, Radar), and (Camera, Sensor Fusion), there is a test that compares the outputs to diagnose each of the output's failure modes (*i.e.*, misdetection, misposition, and misclassification). Intuitively, each test compares the two sets of obstacles coming from the corresponding modules, and if they are different, it reports if the inconsistency was due to a misdetection, misposition, or misclassification. Moreover, we included a priori relation between every module and its output. In particular, the modules are assumed to fail if their outputs have at least one active failure mode. In the probabilistic diagnostic graph we also added an a priori relation for each module's failure mode, indicating the prior probability of that failure mode being active.

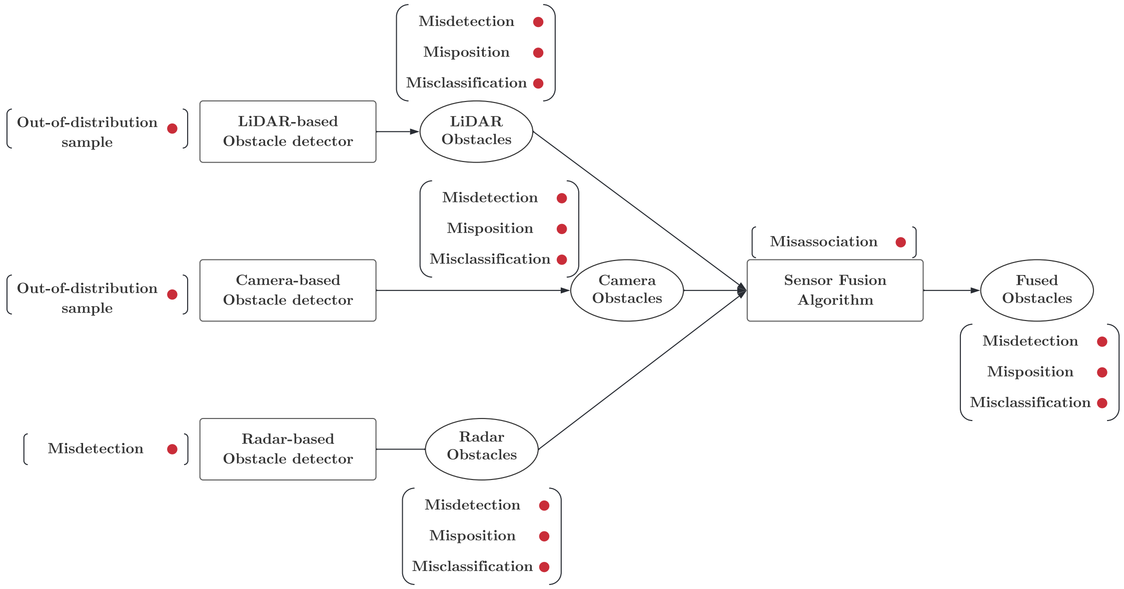


Figure 3-9: Perception system considered in our experiments. Modules are shown as rectangular blocks, outputs are shown as rounded boxes, while failure modes are denoted with red dots.

Diagnostic Tests

We now describe the logic for the diagnostic tests we implemented. Consider two sets of synchronized detected obstacles[6], say $\mathcal{A}$ and $\mathcal{B}$, produced by two modules, using some sensor data. Let Ω be the region defined by the intersection of both sensor fields

[6] By synchronized we mean that the two outputs are produced at the same time instant.

of view and a region of interest (*e.g.,* a region close to a drivable area[7]). Denote by $\mathcal{A}_\Omega$ and $\mathcal{B}_\Omega$ the set of obstacles restricted to the region Ω, namely $\mathcal{A}_\Omega \subseteq \mathcal{A}$ such that for each obstacle o in $\mathcal{A}$, o is in $\mathcal{A}_\Omega$ if and only if o is inside the region defined by Ω. The same relation holds for $\mathcal{B}_\Omega$. Then the diagnostic test checking for misdetections is defined as follows:

$$t_{\text{misdetection}} = \begin{cases} \text{FAIL} & \text{if } |\mathcal{A}_\Omega| \neq |\mathcal{B}_\Omega| \\ \text{PASS} & \text{otherwise} \end{cases}$$

Note that if the two sets of obstacles have a different cardinality —when restricted to the area co-visible by both sensors— it means that one of the two sets contains a ghost obstacle or one of the two sets is missing an obstacle. From a single test, we are not able to say which of the two sets is experiencing the misdetection, but we know at least one output did.

Let us now move our attention to the misposition failure mode. Let $\mathcal{C}$ be the set of *matched* obstacles, that is, a pair of obstacles (l, r) —with $l \in \mathcal{A}_\Omega$ and $r \in \mathcal{B}_\Omega$— is in $\mathcal{C}$, if l and r represent the same obstacles. A common approach for finding the set of matches is to select all the pairs that are closest to each other (*i.e.,* solving an assignment problem)[8]. The diagnostic test checking for mispositioned obstacles is defined as follows:

$$t_{\text{misposition}} = \begin{cases} \text{FAIL} & \exists (l, r) \in \mathcal{C} \text{ such that } |\text{pos}(l) - \text{pos}(r)| \geq \theta \\ \text{PASS} & \text{otherwise} \end{cases}$$

where $\text{pos}(\cdot)$ is the position of an obstacle and θ is an error threshold, chosen as $\theta = 2.5\,\text{m}$ in our experiments.

[7] In our experiments, the region of interest is the area within 5 meters from a drivable lane.

[8] We matched obstacles using a generalization of the Hungarian algorithm [155], with the cost of each match being the Euclidean distance between obstacles.

Finally, the test checking for misclassified obstacles is defined as follows:

$$t_{\text{misclassification}} = \begin{cases} \text{FAIL} & \exists (l, r) \in \mathcal{C} \text{ such that } \operatorname{cls}(l) \neq \operatorname{cls}(r) \\ \text{PASS} & \text{otherwise} \end{cases}$$

where cls$(\cdot)$ is the class of the obstacle, *i.e.,* the test fails if associated obstacles are assigned different semantic classes.

Temporal Diagnostic Graph

To build a temporal diagnostic graph we stack 2 regular diagnostic graphs into a temporal diagnostic graph. In the probabilistic case, each module failure mode is connected to its successive (in time) via a priori relationships, which represent the transition probability between states in consecutive time steps. No temporal a priori relations are added in the deterministic case. We also added temporal tests. The logic of the tests presented in Section 3.5.2 is applicable to temporal tests with small changes. In temporal tests, the sets $\mathcal{A}$ and $\mathcal{B}$ are not time-synchronized anymore (*e.g.,* they are obstacles detected by the same sensor at consecutive time stamps), therefore the position of each obstacle in each set must be adjusted for the distance the obstacle traveled between consecutive detections. To use the tests described earlier in the temporal domain we used the following approach. If the obstacle is equipped with an estimated velocity vector, since the time difference between detections is usually below 30 ms, we assume constant speed and integrate the speed over the time interval to find an approximate position of each obstacle. When the velocity is not available, we use the average speed of an obstacle (for a given obstacle's class) and adapt the misposition threshold θ to account for the uncertainty.

3.5.3 Fault Identification: Implementation Details

Deterministic Fault Identification. For the tests with the deterministic model, we assumed the Weaker-OR model for the diagnostic tests as described in Eq. (3.3). We used this model for both the regular diagnostic graph and the temporal diagnostic graph, and solved the optimization problem in Eq. (3.8) using Google OR-Tools [156] Integer Programming Solver.

Probabilistic Fault Identification. To perform probabilistic inference on the diagnostic graph, we transformed it into a factor graph and trained the potentials for each relation using the maximum margin learning algorithm described in Section 3.3.2 on the training dataset. We used the Hamming distance defined in Eq. (3.20) as the loss function $\mathcal{L}$. We set the regularization parameter to $\lambda = 10$; see [133].[9] For each diagnostic graph, we perform inference using the max-product algorithm for a fixed number of iteration (100 iterations). In our implementation, we use the *Grante* library [157] to perform learning and inference over the factor graph.

Graph-Neural-Network-based Fault Identification. In Section 3.3.3 we saw that a graph neural network requires a feature for each node in the graph to perform neural message passing. We now discuss how we set the feature vector for each node in the graph. Recall that the GNN uses a pairwise undirected graph, where a node is either a failure mode or a test outcome. The feature $\boldsymbol{x}_{t_k} \in \mathbb{R}^2$ for a test t_k is set as the one-hot encoding of the test outcome (*i.e.,* [1 0] if the test passed, [0 1] if it failed). For the failure mode nodes we do not have any measurable quantity at runtime; we therefore use the training dataset to compute the feature vectors. In particular the feature vector $\boldsymbol{x}_{f_i} \in \mathbb{R}^2$ for a failure mode f_i is computed as follow: let ρ_i be the empirical probability that f_i is ACTIVE, *i.e.,* $\rho_i = \frac{1}{|\mathcal{W}|}\sum_{(\boldsymbol{z},\boldsymbol{f})\in\mathcal{W}} \mathbb{1}[f_i = \text{ACTIVE}]$; then the feature vector is chosen as $\boldsymbol{x}_{f_i} = [1-\rho_i, \rho_i]^\mathsf{T}$. Intuitively, the feature describes the prior probability of the failure mode f_i's state.

We now discuss the architecture of the GNN. Our GNN is composed by a linear layer that embeds the feature vectors in $\mathbb{R}^{16}$, followed by a ReLU function. The output is then passed to a stack of graph convolution layers interleaved with ReLU activation functions. We tested four different graph convolution layers

- in the case of GCN, we stack 3 layers with 16 hidden channels each;
- in the case of GCNII, we stack 64 layers with 16 hidden channels each with

[9] In our experiment we noticed that the performance of the learning algorithm are not sensitive to the choice of λ.

$\alpha = 0.1$, $\beta = 0.4$;

- in the case of GIN, we stack 3 layers with 16 hidden channels each with the function $\zeta^{(k)}(\cdot)$ (*cf.* Eq. (3.19)) being a 2-layer perceptron for $k = 1, \ldots, 3$;

- in the case of GraphSAGE, we stack 3 (and 6 for temporal diagnostic graphs) layers with mean aggregator and 16 hidden channels each.

Finally, the readout function that converts the graph embedding to node labels is a linear layers followed by a softmax pooling. We perform an ablation of the different GNN architectures in Section 3.5.5.

We implemented the GNNs in PyTorch [141] and trained them on the training dataset for 100 epochs using the Adam optimizer. To reduce the amount of guesswork in choosing an initial learning rate, we used the learning rate finder available in the PyTorch Lightning library [158]. The procedure is based on [159]: the learning rate finder does a small training run where the learning rate is increased after each processed batch and the corresponding loss is logged. Then, the learning rate is chosen to be the point with the steepest negative gradient.

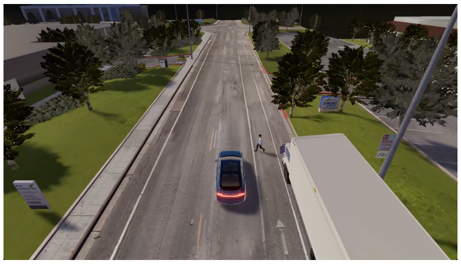
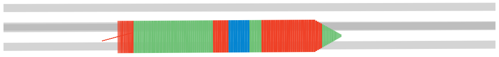
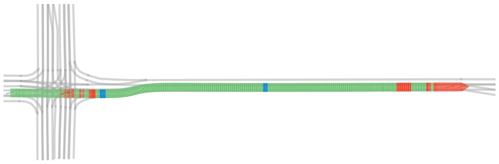
Baselines. We compared the proposed monitors against two simple baselines. In the first baseline (label: "Baseline"), whenever a diagnostic test returns FAIL, all failure modes in its scope are considered active. In the second baseline (label: "Baseline (w/rel. scores)"), modules are ordered by a reliability score defined by the system designer. In our experiments we considered the radar to be more reliable than the sensor fusion, which is more reliable than the LiDAR, which in turn is more reliable than the camera. When a diagnostic test fails, this second baseline labels all the failure modes in the test scope associated to the least reliable module (and its outputs) as ACTIVE. For example if a diagnostic test comparing camera and LiDAR obstacles returns FAIL, the failure modes associated with the camera are the ones that are labeled active because the camera is considered less reliable than the LiDAR. Both baselines label a module' failure modes as active if at least one of the module's outputs is failing.

3.5.4 Scenarios

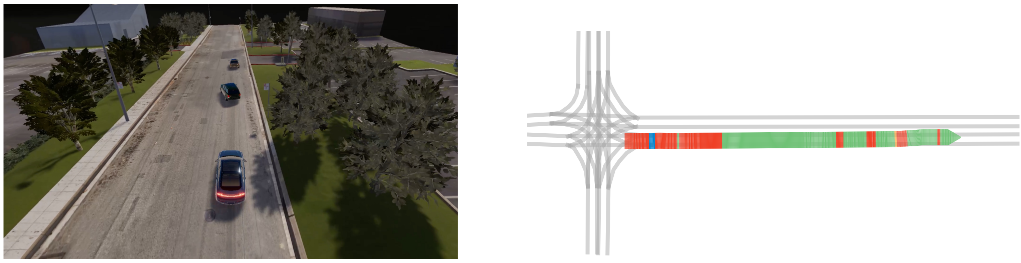
We designed a set of challenging scenarios to stress-test the Apollo Auto perception system. These scenarios were created using the LGSVL Simulator Visual Scenario Editor, which allows the user to create scenarios using a drag-and-drop interface. The vehicle behavior is tested on each scenario in a multitude of situations including different time of day (noon, 6 PM, 9 PM) or weather condition (rain and fog). The scenarios are described in Table 3.2.

Table 3.2: **Scenarios.** (Left) Snapshot of the scenario, (Right) Top-view of the trajectory, color-coded by fault detection results. The motion of the vehicle is represented by an arrow with the tail of the arrow representing the start location and the head of the arrow representing the stop location (the direction of motion is always left-to-right or bottom-to-top).

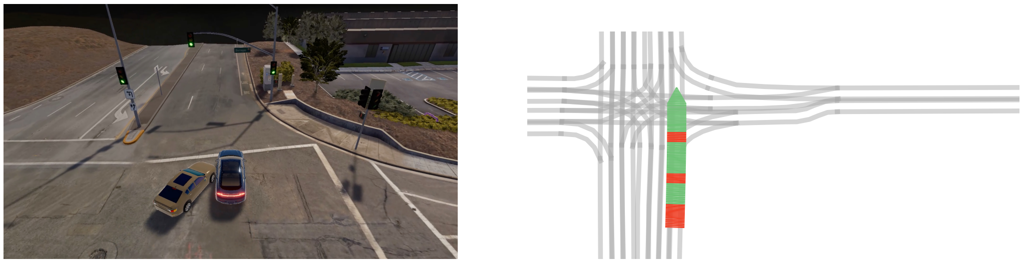
🟩 Fault-free (TN) 🟥 Fault Detected Correctly (TP)
🟦 False Alarm (FP) 🟧 Missed Fault (FN)

Scene	Fault Detection Results
	
Hidden Pedestrian. A pedestrian, initially occluded by a track parked on the right-hand side of the street, steps in front of the ego vehicle.	
	

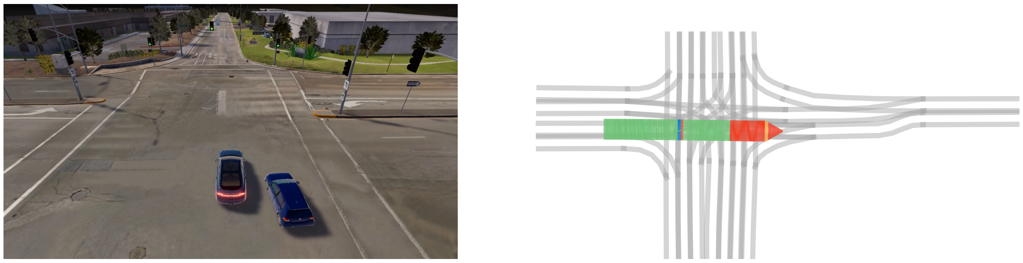
Overturned Truck. The ego vehicle encounters an overturned truck occupying the lane it is driving in. The scenario recreates an accident occurred in Taiwan where a Tesla hit an overturned truck on a highway [160].



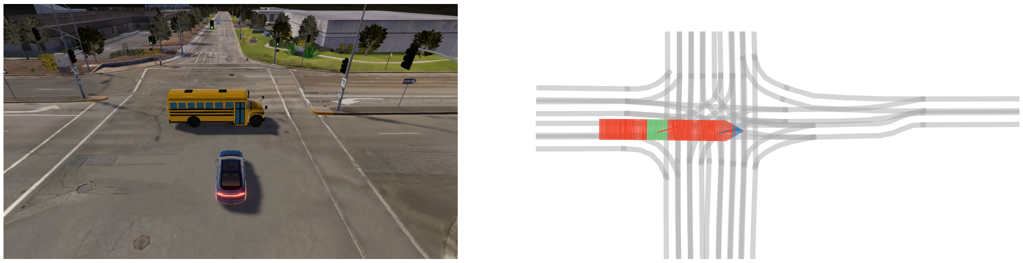
Stopped Vehicle. While driving, the car in front of the ego vehicle makes a lane change to avoid the stationary car that is in their lane. This leaves the ego vehicle with little to no time to react to the stationary car.



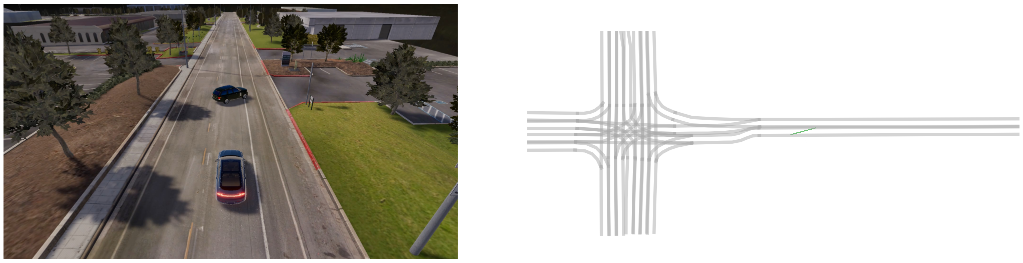
Cut Off Left. While driving in the right lane on a three-lane road, a vehicle from the left lane cuts the ego vehicle off.



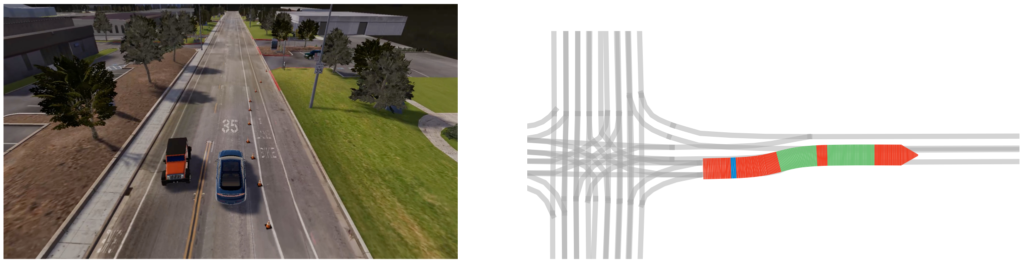
Cut Off Right. While driving in the left lane on a two-lane road, a vehicle from the right lane cuts the ego vehicle off while turning into a parking lot.



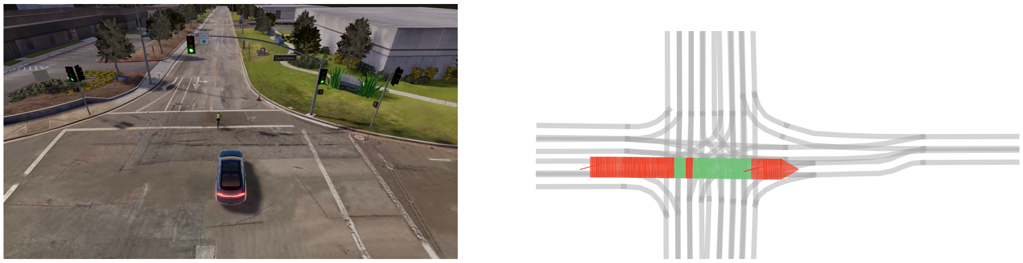
School Bus Intersection. The ego vehicle drives through an intersection. A school bus crosses the intersection coming from the left-hand side. As the ego vehicle crosses the intersection, a pedestrian steps into the intersection from the left-hand side.



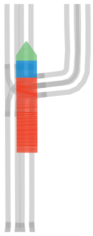
Car in Front. A car is still in front of the ego vehicle preventing it to move forward.



Cones in the Lane. The ego vehicle is driving on a lane partially delimited by traffic cones, while another vehicle is driving in the opposite lane. After passing traffic cones, another vehicle exits a parking lot and merges right in front of the ego vehicle.



Cyclist. The ego vehicle is stopped at an intersection and as it starts driving through the intersection, a cyclist enters the field of view from the left-hand side of the intersection and rides right in front of the ego vehicle.



Turkeys. While driving on a straight road, the ego vehicle must avoid a collision with two turkeys that suddenly walk in front of the ego vehicle.

Dataset generation

We executed the diagnostic tests described in Section 3.5.2 every 0.3 s, and used the corresponding test outcomes to perform fault identification. Time synchronization of the modules' output is achieved by pairing outputs that are closest in time to each other. Ground-truth labels for the outputs' failure modes are generated using the ground-truth detections provided by the simulator. In particular, to generate the label for each failure mode of an output, we used the three diagnostic tests described in Section 3.5.2 comparing the set of obstacles to the ground-truth detections. For a module m instead, since all modules have only one failure mode, the associated failure mode f_m is labeled as ACTIVE if and only if any failure mode if its output is ACTIVE. We collected 1650 regular diagnostic graphs from different deployments of the agent in the scenarios described in Table 3.2. The samples are randomly split them into 1320 (80%) training samples, 165 (10%) testing samples, and 165 validation samples. Of the 1320 samples used for training, 675 (51.13 %) contain a failure and 645 (48.86 %) do not. The dataset is therefore balanced for the purpose of training the diagnostic graph. To create the temporal diagnostic graph, we used a sliding window that stacks 2 consecutive regular diagnostic graphs into a single temporal

Algorithm	**Regular**			**Temporal**		
	All	Outputs	Modules	All	Outputs	Modules
Factor Graph	93.30	96.72	83.03	93.60	96.88	83.74
Deterministic	91.06	93.69	83.18	89.26	92.33	80.06
Baseline (w/rel. scores)	92.39	94.65	85.61	90.18	92.69	82.67
Baseline	84.85	89.09	72.12	83.90	87.73	72.39
GCN	92.27	96.01	81.06	91.79	96.06	78.99
GCNII	87.61	93.94	68.64	92.60	96.01	82.36
GIN	91.89	96.06	79.39	93.21	96.47	83.44
GraphSage	92.84	96.46	81.97	92.71	96.42	81.60

Table 3.3: **Fault identification accuracy.** Best accuracy is highlighted in green, second-best is highlighted in yellow.

diagnostic graph. Using this approach, we collected 1590 temporal diagnostic graphs, randomly split into 1272 (80%) training samples, 159 (10%) test samples, and 159 validation samples. As a result of the random splitting, both the temporal and regular diagnostic graph datasets may contain samples that are 0.3 s apart.

3.5.5 Fault Detection and Identification Results

We used three metrics to evaluate the performance for both the fault detection and identification problems:

Accuracy is the percentage of correctly detected (resp. identified) failures over the total number of samples;

Precision measures the percentage of correct identifications over the number of failures the fault identification system reported; a monitor achieves high precision if it has a low rate of false alarms;

Recall measures the percentage of correct identifications over the number of failures the system experienced; a monitor has high recall if it is able to catch a large fraction of failures occurring in the perception system;

Fault Identification Results

Table 3.3 reports the accuracy of all compared techniques, averaged across all test scenarios in Table 3.2. The first and fourth columns report the overall accuracy ("All")

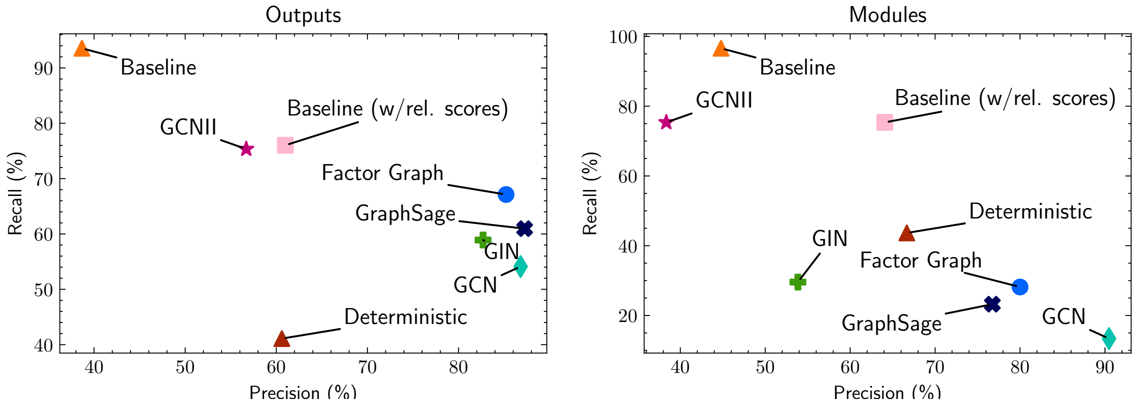


Figure 3-10: Precision/Recall for regular diagnostic graphs. (Left) Modules, (Right) Outputs.

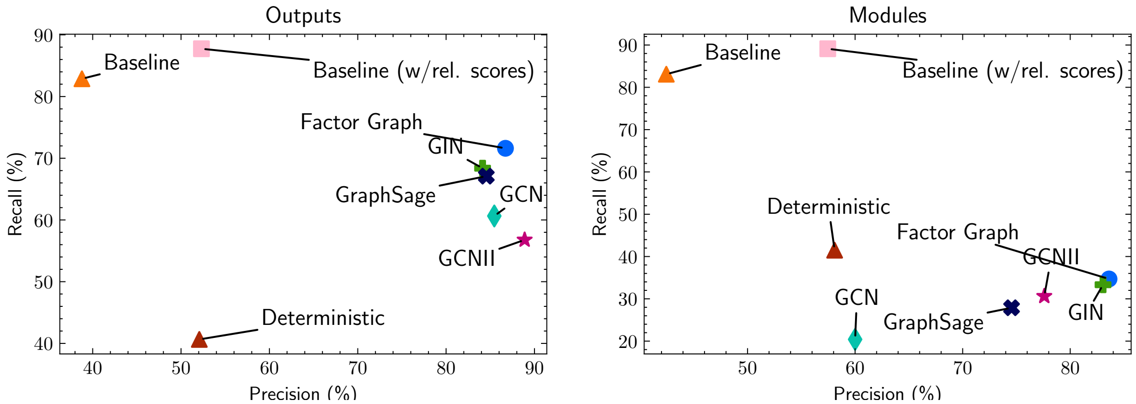


Figure 3-11: Precision/Recall for temporal diagnostic graphs. (Left) Modules, (Right) Outputs.

when using regular and temporal diagnostic graphs, respectively. The remaining columns report a breakdown of the accuracy in terms of modules and outputs. The overall accuracy results suggest that factor-graph-based probabilistic fault identification outperforms all other algorithms and achieves 96.72 % accuracy when using regular diagnostic graphs and 96.88 % with temporal diagnostic graphs. GNNs architectures achieve the second-best performance (GraphSAGE in the regular case, GIN in the temporal case). If we now look at the breakdown of the fault identification results between modules and outputs, we notice two trends. First, the factor graph still performs the best across the spectrum, but it is slightly slightly inferior than a baseline in the regular case. As we will see shortly, the baselines tend to make quite conservative decisions (*i.e.,* they tend to detect more failures than the ones actually present in the system), which increases accuracy (and recall) at the expense of precision. Second, output fault identification has higher accuracy than module fault identification; this is expected, since most of our tests directly involve outputs, while we can only indirectly infer module failures via the a priori relations. Note that the two statistics (output fault identification vs. module fault identification) are typically used for different purposes, as discussed in Remark 21.

Fig. 3-10 shows precision-recall trade-offs when using regular diagnostic graphs. Best results are near the top-right corner of each figure, where both precision and recall are high. The figure confirms that while the baselines have large recall (due to the fact that are conservative in detecting failure modes as active), their precision is relatively low (*i.e.,* they have a large number of false alarms). On the other side of the spectrum, GNN architectures (with the exception of GCNII) achieve high prediction (87.25 % for GraphSAGE) but low recall (60.96 % for GraphSAGE). The deterministic fault identification struggles to mark failure modes as active, achieving low precision and recall in the output space; this is due to the fact that it disregards PASS results (which do not even appear in the optimization Eq. (3.8)). Factor graph inference again achieves a reasonable trade-off, with 85.22 % precision and 67.12 % recall.

Fig. 3-11 shows precision-recall trade-offs when using temporal diagnostic graphs. Compared to the regular diagnostic graph we see a steep increase in precision in the output space. The best-performing model goes from around 90 % precision of the regular graph to 97 % of the temporal diagnostic graph.

PAC-Diagnosability. Fig. 3-12 and Fig. 3-13 show the PAC-Diagnosability bound defined in Eq. (3.23) for each of the compared techniques. The bound represents the number of fault identification mistakes each algorithm is expected to make with a given confidence (δ in Eq. (3.23)). The plots show that with high probability, most of the algorithms are expected to make less than 1 mistake in the fault identification (*i.e.,* false alarms or false negatives). The factor graph has the lowest bound of all methods in both the regular and temporal diagnostic graphs; the only exception is Fig. 3-12(right), where the baseline with reliability score has the lowest bound for module fault identification.

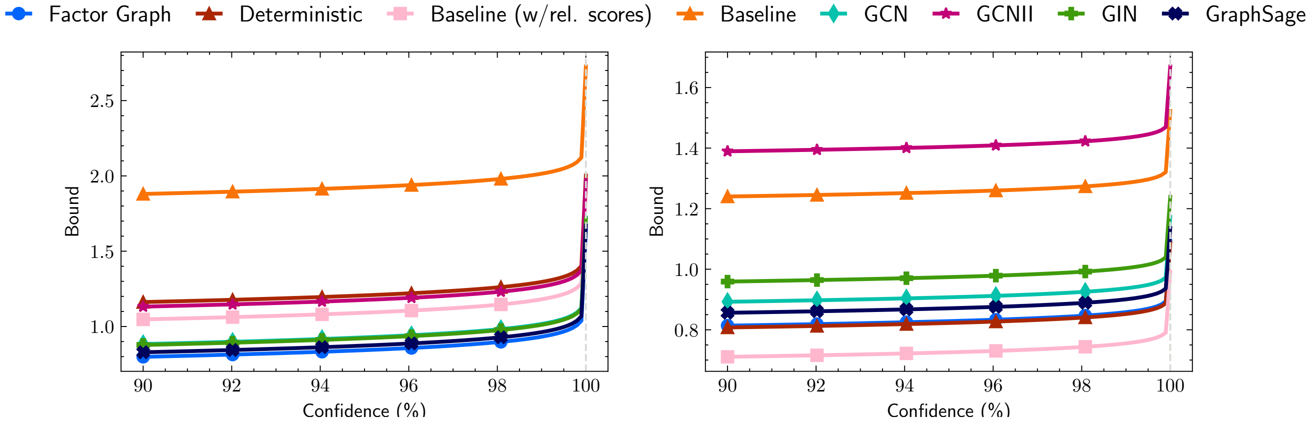


Figure 3-12: PAC-diagnosability bounds for regular diagnostic graphs. (Left) Modules, (Right) Outputs. Lower is better.

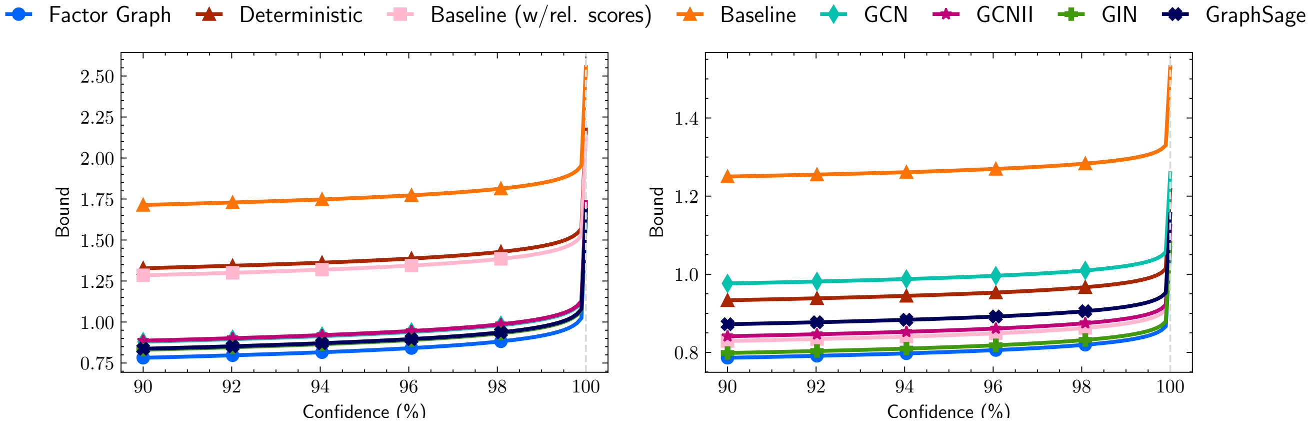


Figure 3-13: PAC-diagnosability bounds for temporal diagnostic graphs. (Left) Modules, (Right) Outputs. Lower is better.

κ-diagnosability. Let us now discuss the deterministic diagnosability of the perception system considered in our experiments (Fig. 3-9). If the tests behave as a Deterministic OR, the diagnostic graph used in our experiments is 5-diagnosable. This means that if there are up to 5 active failure modes the deterministic fault identification will be able to correctly identify them. If we instead assume the tests behave as a Weak-OR, which might fail when all the failure modes in its scope are active, the diagnostic graph is 3-diagnosable. It's worth noticing that this does not mean that if there are more than 3 (or 5) active failure modes the fault identification will surely fail, but rather that we do not have the guarantee that it will not make any mistake. When using Deterministic Weaker-OR tests, the diagnosability drops to zero, meaning that the fault identification guarantees vanish.

Extra diagnosability results. To show the effectiveness of the deterministic and probabilistic diagnosability we generated a random 4-diagnosable diagnostic graph with 10 independent failure modes and Weak-OR tests and collected the fault identification results (using the deterministic model) for every syndrome and every possible fault assignment. The results are shown in Fig. 3-14. The figure reports the average number of incorrect fault identification results (*i.e.,* the Hamming distance between the estimated and actual vector of active faults) for increasing number of active faults. The vertical dashed line represents the deterministic diagnosability value: by Definition 36, the fault identification is guaranteed to correctly identify the active failure modes provided that there are less than 4 active failure modes. In fact, from the plot we see that the fault identification algorithm does not make any mistake in the fault identification when there are less than 4 faults. The horizontal dashed line instead represents the probabilistic diagnosability value, in particular it is the ceiling of the bound in Eq. (3.23), computed with very high confidence ($1-1\times10^{-12}$). The bound guarantees that with high probability the average number of mistakes (the average Hamming distance) the fault identification algorithm is going to make is less that 2; this is again consistent with the numerical results.

Timing. The runtime of each method is shown in Table 3.4. All algorithms perform inference in less than 4 ms, except for GCNII which averages at around 20 ms. This is likely due to the fact that GCNII uses a deep architecture, which incurs an increased computational cost. The best performing algorithm, *i.e.,* the factor graph, can be executed in real-time as its runtime averages around 0.8 ms for regular graphs and 3.8 ms for temporal graphs.

Fault Detection Results

Recall that fault detection is the problem of deciding whether the system is working in normal conditions or whether at least a fault has occurred. Table 3.5 and Fig. 3-15 show accuracy, precision, and recall. Fig. 3-15 shows that most of the algorithms for

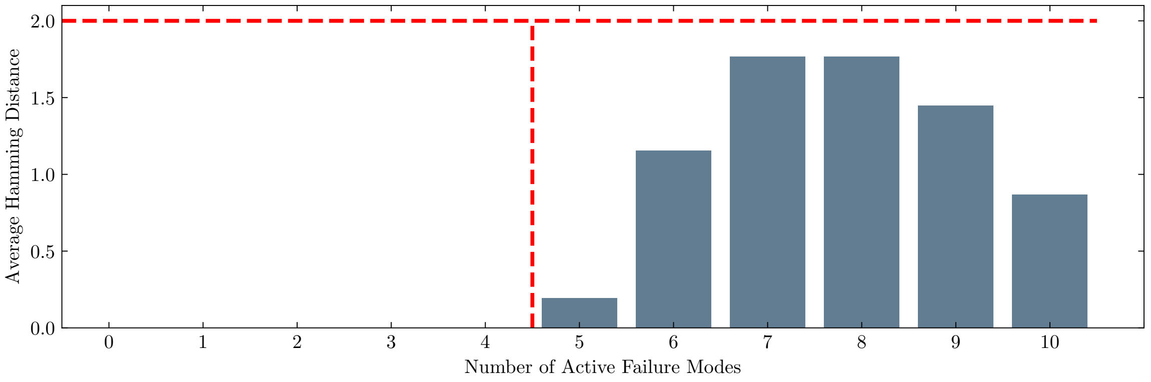


Figure 3-14: **PAC-diagnosability vs. κ-diagnosability.** Average Hamming distance between the estimated and actual vector $\boldsymbol{f}$ of fault states in a randomly generated 4-diagnosable diagnostic graph with 10 independent failure modes and Weak-OR tests. The vertical dashed line represents the deterministic diagnosability bound: if the system is experiencing less than 4 active failure modes, the fault identification is guaranteed to be correct (0 Hamming distance). The horizontal dashed line represents the ceiling of the PAC-diagnosability bound in Eq. (3.23): with very high probability the average number of mistakes (average Hamming distance) is less than the PAC-diagnosability bound.

inference presented In this thesis (as well as the baselines) attain similar performance with precision above 90 % and recall above 80 %; this confirms that fault detection is a somewhat easier problem compared to fault identification. Table 3.5 shows that the deterministic approach and the baselines do particularly well for fault detection: they both detect failure as soon as a single test fails, which makes their accuracy high. On the other hand, the factor graph approach may prefer explaining a failed

		Factor Graph	**Deterministic**	**Baseline (w/rel. scores)**	**Baseline**	**GCN**	**GCNII**	**GIN**	**GraphSage**
Regular	Avg.	0.79	3.25	0.10	0.10	0.63	19.88	0.48	0.59
	Std.	(0.17)	(0.14)	(0.06)	(0.06)	(0.01)	(0.10)	(0.01)	(0.02)
Temporal	Avg.	2.53	3.68	0.27	0.26	0.68	24.56	0.50	0.85
	Std.	(0.04)	(0.46)	(0.17)	(0.16)	(0.01)	(0.33)	(0.01)	(0.01)

Table 3.4: Average runtime (“Avg.”) and standard deviation (“Std.”) for fault identification, in milliseconds.

Algorithm	Regular			Temporal		
	All	Outputs	Modules	All	Outputs	Modules
Factor Graph	76.67	88.48	64.85	81.60	91.41	71.78
Deterministic	89.09	89.09	89.09	93.25	93.25	93.25
Baseline (w/rel. scores)	89.09	89.09	89.09	85.28	85.28	85.28
Baseline	89.09	89.09	89.09	85.28	85.28	85.28
GCN	71.82	86.06	57.58	80.06	90.18	69.94
GCNII	68.48	87.88	49.09	78.83	85.89	71.78
GIN	83.94	86.06	81.82	83.13	92.64	73.62
GraphSage	76.67	89.09	64.24	79.14	89.57	68.71

Table 3.5: **Fault detection accuracy.** Best accuracy is highlighted in green, second-best is highlighted in yellow.

test as a false alarm. Therefore, while factor graphs would be the go-to approach for fault identification, a simpler baseline approach suffices for fault detection.

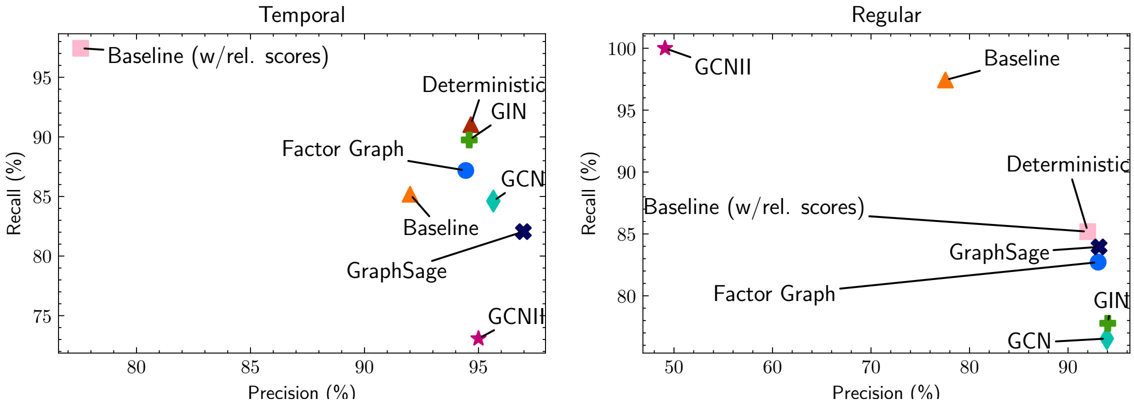


Figure 3-15: Fault detection in diagnostic graphs. (Left) Regular, (Right) Temporal.

The results of the fault identification experiments show that the factor graph can achieve the best accuracy for fault identification, and all the proposed approaches achieve similar performance for fault detection.

The choice between model-based (factor graph or deterministic factor graph) and deep-learning-based (graph neural networks) depends on the specific application. Model-based approaches have the advantage of being more interpretable, but the inference time increases with the number of failure modes (or timesteps), while deep-learning-based approaches have the advantage of having an almost constant inference time (*e.g.,* GCN and GIN), but are not interpretable. Deterministic diagnostic graphs and factor graphs have clear advantages when it is not possible to curate a dataset for the purpose of training a model, because the system designer can directly encode the expected system behavior. Finally, the deterministic diagnostic graph provides stronger guarantees (*i.e.,* deterministic diagnosability) compared to factor graphs and graph neural networks (*i.e.,* PAC-diagnosability).

Table 3.6 shows the results of fault identification for temporal diagnostic graphs for each scenario class. Similar to Table 3.3, we see that the factor graph is more likely to outperform the other in the output space. The deterministic diagnostic graph, on the other hand, is most likely to outperform the other approaches in terms of recall in module space, due to the fact that it conservatively estimates failure modes as active when a test fails (due to the specific choice of diagnostic tests used, *i.e.,* WeakerOR) and propagates the failure to modules. No clear pattern emerges from the scenario-based analysis that would justify choosing one graph neural network architecture over another, even with more information about the failure distribution.

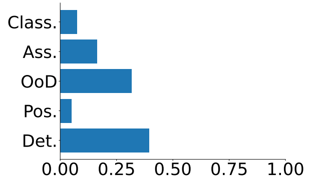
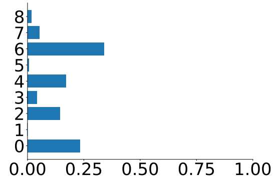
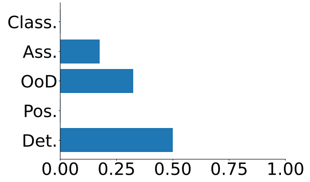
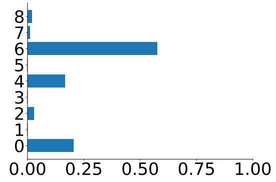
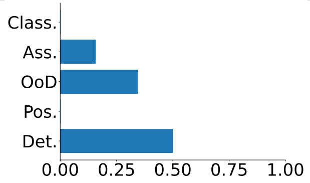
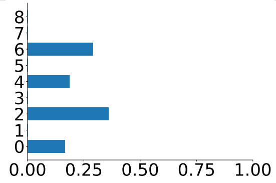
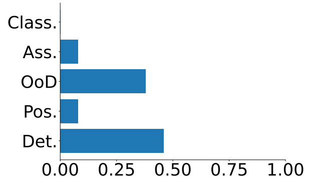
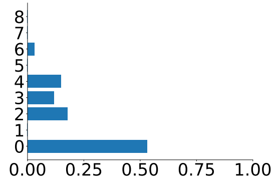
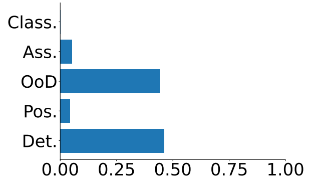
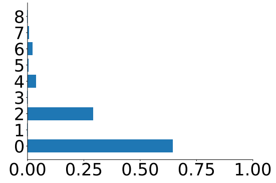
Scenario	**Alg.**	Modules **Prec.**	 **Rec.**	 **Acc.**	Outputs **Prec.**	 **Rec.**	 **Acc.**	**Tests**	**Failure Types**	**Scenario Failures**
Cyclist	FG	94.16	46.03	75.28	93.98	73.96	95.21	94.24 47.57 86.43		
	Det.	64.25	40.55	64.04	47.91	35.49	83.69			
	GCN	85.82	38.41	70.44	87.23	60.65	92.50			
	GCNII	90.51	45.40	74.17	92.77	64.50	93.69			
	GIN	89.74	44.44	73.62	90.94	68.34	94.01			
	SAGE	93.88	43.81	74.31	86.03	69.23	93.46			
Turkeys	FG	99.09	51.42	74.00	98.83	79.34	96.17	92.70 37.96 84.46		
	Det.	27.16	9.73	37.38	24.71	9.25	78.57			
	GCN	95.45	9.91	52.00	99.39	76.53	95.75			
	GCNII	99.04	48.58	72.50	98.84	79.81	96.25			
	GIN	99.08	50.94	73.75	97.13	79.34	95.92			
	SAGE	100.00	49.06	73.00	99.38	75.59	95.58			
School Bus Intersection	FG	35.14	3.35	58.95	85.79	42.01	91.36	90.01 46.62 88.74		
	Det.	82.13	43.26	73.47	80.86	43.00	91.02			
	GCN	51.16	11.34	60.29	70.97	39.69	89.81			
	GCNII	30.91	4.38	57.92	76.54	31.96	89.64			
	GIN	50.00	10.57	60.08	82.69	44.33	91.36			
	SAGE	33.33	2.06	59.26	78.79	53.61	91.91			
Hidden Pedestrian	FG	60.00	20.00	88.71	81.25	68.42	97.58	75.33 68.90 94.27		
	Det.	63.16	75.00	91.41	66.67	76.19	96.61			
	GCN	50.00	13.33	87.90	64.71	57.89	96.24			
	GCNII	33.33	20.00	85.48	73.33	57.89	96.77			
	GIN	50.00	20.00	87.90	84.21	84.21	98.39			
	SAGE	30.00	20.00	84.68	83.33	78.95	98.12			
Overturned Truck	FG	95.00	26.39	91.77	93.57	89.12	99.36	87.24 75.89 97.59		
	Det.	86.67	83.87	96.63	80.00	80.50	98.43			
	GCN	68.52	25.69	90.55	90.30	82.31	99.01			
	GCNII	88.64	27.08	91.62	93.43	87.07	99.29			
	GIN	73.33	22.92	90.62	93.18	83.67	99.16			
	SAGE	92.31	16.67	90.70	84.44	77.55	98.63			

Table 3.6: Algorithm performance breakdown by scenario type (Part 1 of 2) for temporal diagnostic graphs. The table shows Precision (Prec.), Recall (Rec.), and Accuracy (Acc.) for each algorithm and scenario type for both modules and outputs. The column **Tests** shows (from top to bottom) Precision, Recall and Accuracy of diagnostic tests. The column *Failure Types* shows the percentage of failures for each failure mode type, representing, from top to bottom, misclassification, mispositioning, out-of-distribution sample, misdetection, and misassociation. Finally, the *Scenario Failures* column reports the number of active failure modes that each sample (*i.e.*, diagnostic graph) has as a percentage of the total number of samples; the horizontal red line represents the average number of active failure modes. The best is highlighted in green, the second best in yellow.

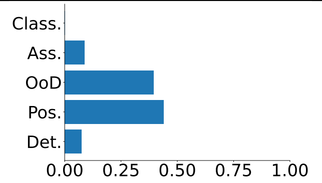
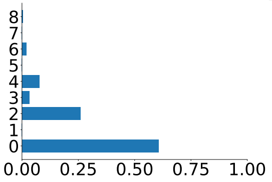
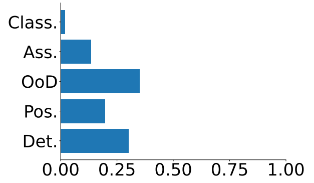
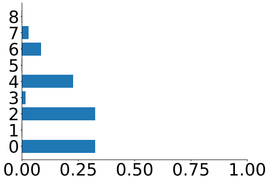
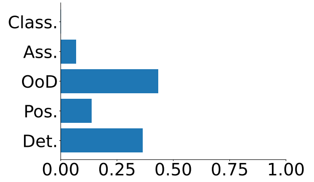
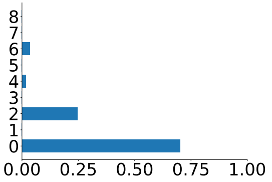
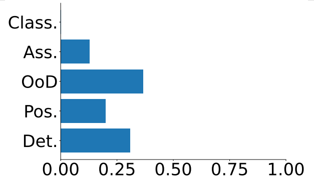
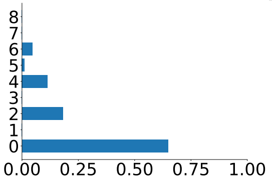
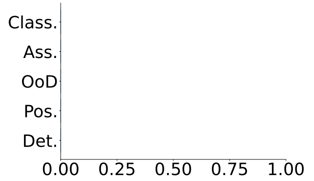
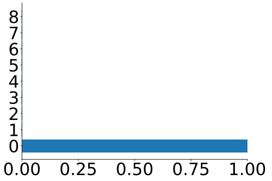
Scenario	Alg.	Modules			Outputs			Tests	Failures Types	Scenario Failures
		Prec.	Rec.	Acc.	Prec.	Rec.	Acc.			
Stopped Vehicle	FG	92.86	26.26	90.72	93.07	89.52	99.26	84.79 57.99 95.82		
	Det.	59.43	60.58	89.81	58.62	61.82	96.36			
	GCN	31.25	5.05	87.00	89.09	46.67	97.44			
	GCNII	0.00	0.00	87.00	75.00	8.57	95.92			
	GIN	0.00	0.00	87.25	88.10	70.48	98.31			
	SAGE	33.33	2.02	87.50	88.24	57.14	97.81			
Cones in the Lane	FG	28.57	8.82	69.92	50.00	34.25	90.49	57.96 46.87 85.30		
	Det.	38.75	40.26	65.58	28.85	36.59	84.78			
	GCN	40.00	20.59	70.70	59.52	34.25	91.54			
	GCNII	26.09	8.82	69.14	54.29	26.03	90.89			
	GIN	55.56	7.35	73.83	52.00	35.62	90.76			
	SAGE	33.33	8.82	71.09	68.00	46.58	92.84			
Cut Off Left	FG	48.15	34.21	91.14	69.09	100.00	98.71	78.47 75.44 97.58		
	Det.	57.45	69.23	92.79	56.25	69.23	97.52			
	GCN	33.33	2.63	91.14	89.29	65.79	98.79			
	GCNII	83.33	13.16	92.27	96.15	65.79	98.94			
	GIN	100.00	2.63	91.59	80.95	89.47	99.09			
	SAGE	80.00	21.05	92.73	75.76	65.79	98.41			
Cut Off Right	FG	73.53	56.82	91.57	77.36	91.11	98.39	85.23 58.55 95.69		
	Det.	63.04	59.18	89.24	57.14	56.00	95.83			
	GCN	78.95	34.09	90.06	91.67	48.89	97.49			
	GCNII	87.50	31.82	90.36	100.00	57.78	98.09			
	GIN	85.71	27.27	89.76	88.89	71.11	98.29			
	SAGE	71.43	34.09	89.46	87.50	62.22	97.89			
Car in Front	FG	–	–	100.00	–	–	100.00	– – 100.00		
	Det.	–	–	100.00	–	–	100.00			
	GCN	–	–	100.00	–	–	100.00			
	GCNII	–	–	100.00	–	–	100.00			
	GIN	–	–	100.00	–	–	100.00			
	SAGE	–	–	100.00	–	–	100.00			

Table 3.6: Algorithm performance breakdown by scenario type (Part 2 of 2) for temporal diagnostic graphs.

3.6 Extended Literature Review

This section reviews related work on runtime monitoring and AV safety assurance, spanning both industrial practice (Section 3.6.1) and academic research (Section 3.6.2).

3.6.1 State of Practice

The automotive industry currently uses four classes of methods to claim the safety of an AV [22], namely: miles driven, simulation, scenario-based testing, and disengagement. Each of these methods has well-known limitations. The *miles driven* approach is based on the statistical argument that if the probability of crashes per mile is lower in autonomous vehicles than for humans, then AVs are safer; however, such an analysis would require an impractical amount (*i.e.,* billions) of miles to produce statistically-significant results [22], [161].[10] The same approach can be made more scalable through *simulation*, but unfortunately creating a life-like simulator is an open problem, for some aspects even more challenging than self-driving itself. *Scenario-based* testing is based on the idea that if we can enumerate all the possible driving scenarios that could occur, then we can simply expose the AV (via simulation, closed-track testing, or on-road testing) to all these scenarios and, as a result, be confident that the AV will only make sound decisions. However, enumerating all possible corner cases (and perceptual conditions) is a daunting task. Finally, *disengagement* is defined as the moment when a human safety driver has to intervene in order to prevent a hazardous situation. However, while less frequent disengagements indicate an improvement of the AV behavior, they do not give evidence of the system safety.

An established methodology to ensure safety is to develop a *standard* that every manufacturer has to comply with. In the automotive industry, the standard ISO 26262 [162] is a risk-based safety standard that applies to electronic systems in production vehicles. A key issue is that ISO 26262 mostly focuses on electronic systems rather than algorithmic aspects, hence it does not readily apply to fully autonomous

[10] Moreover, the analysis should cover all representative driving conditions (*e.g.,* driving on a highway is easier than driving in urban environment) and should be repeated at every software update, quickly becoming impractical.

vehicles [163]. The recent ISO 21448 [29], which extends the scope of ISO 26262 to cover autonomous vehicles functionality, primarily considers mitigating risks due to unexpected operating conditions, and provides high-level considerations on best-practice for the development life-cycle. Both ISO 26262 and ISO/PAS 21448 are designed for self-driving vehicles supervised by a human [164]. Koopman and Wagner [165] propose a standard called UL 4600 [166] specifically designed for high-level autonomy (levels 4 and 5). This standard focuses on ensuring that a comprehensive safety case is created, but it is technology-agnostic, meaning that it requires evidence of system safety without prescribing the use of any specific approach or technology to achieve it.

3.6.2 State of the Art

Related work tries to tackle the problem of safety assurance using different strategies. **Formal methods** [167]–[175] have been used as a tool to study safety of autonomous systems. These approaches have been successful for decision systems, such as obstacle avoidance [176], road rule compliance [177], high-level decision-making [178], and control [114], [179], where the specifications are usually model-based and have well-defined semantics [180]. However, they are challenging to apply to perception systems, due to the complexity of modeling the physical environment [181], and the trade-off between evidence for certification and tractability of the model [182]. One common approach is finding an example where the system fails (*i.e., falsification*). Current approaches [183]–[185] consider high-level abstractions of perception [22], [128], [186] or rely on simulation to assert the true state of the world [183], [184], [187]. Other approaches focus on adversarial attacks for neural-network-based object detection [188]–[190]; these methods derive bounds on the magnitude of the perturbation that induces incorrect detection result, and are typically used off-line [191].

Previous works on **runtime fault detection and identification** focused on components of the perception system [192]. Miller *et al.* [34] propose a framework for quantifying false negatives in object detection. Out-of-distribution sample detection [193]–[196] is a popular technique for detecting failures due to shifts in the distribution of data in learning-based algorithms. For semantic segmentation, Besnier *et al.* [33] and Oberdiek *et al.* [197] propose an out-of-distribution detection mechanism, while Rahman *et al.* [198] propose a failure detection framework to identify pixel-level misclassifications. Lambert and Hays [199] propose cross-modality fusion algorithm to detect changes in high-definition map. Liu and Park [126] propose a methodology to analyze the consistency between camera image data and LiDAR data to detect perception errors. Sharma *et al.* [127] propose a framework for equipping any trained deep network with a task-relevant epistemic uncertainty estimate. Several GNSS/RTK integrity monitors have been proposed [31], [32], [200]–[203] to detect localization errors (the interested reader should refer to [204], [205] for a comprehensive survey). Another line of works leverages spatio-temporal information to detect failures. You *et al.* [206] use spatio-temporal information from motion prediction to verify 3D object detection results. Balakrishnan *et al.* [186], [207] propose the Timed Quality Temporal Logic (TQTL) to reason about desiderable spatio-temporal properties of a perception algorithm.

Kang *et al.* [208] use model assertions, which similarly place a logical constraint on the output of a module to detect anomalies. Fault-tolerant architectures [209] have been also proposed to detect and potentially recover from a faulty state, but these efforts mostly focus on implementing watchdogs and monitors for specific modules, rather than providing tools for system-level analysis and monitoring.

Fault identification and anomaly detection have been extensively studied **in other areas of engineering**. Bayesian networks, Hidden Markov Models [210], [211], and deep learning [212] have been used to enable fault identification, but mainly in industrial systems instrumented to detect component failures. Graph-neural networks have been used for anomaly detection (see [213] for a comprehensive survey). In this context, “anomaly detection is the data mining process that aims to identify the unusual patterns that deviate from the majorities in a dataset” [213]. In order to detect anomalies, objects (*i.e.,* nodes, edges, or sub-graphs) are usually represented by features that provide valuable information for anomaly detection, and when a feature considerably differs from the others (or the training data), the object is classified as anomalous. De Kleer and Williams [214] propose a methodology to detect failures by comparing observations with a predicted output. The dissimilarities are then used to search for potential failures that explain the measurements. The work assumes the availability of a model that predicts the behavior of the system, and —after collecting intermediate results of each component— it searches for the smallest set of failing components that explains the wrong measurements. Preparata, Metze, and Chien [44] study the problem of fault diagnosis in multi-processor systems, introducing the concept of diagnosability; their work is then extended by subsequent works [146], [215], [216]. Sampath *et al.* [217] propose the concept of diagnosability for discrete-event systems [218], [219]. The system is modeled as a finite-state machine, and is said to be diagnosable if and only if a fault can be detected after a finite number of events.

The present paper extends this literature in several directions. First, we take a black-box approach and remain agnostic to the inner workings of the perception system we aim to monitor (relaxing assumptions in related work [214]). Second, we develop a fault identification framework that reasons over the consistency of heterogeneous and potentially asynchronous perception modules (going beyond the homogeneous, synchronous, and deterministic framework in [44]). Third, the framework is applicable to complex real-world perception systems (not necessarily modeled as discrete-event systems [218], [219]). The present paper also extends our previous work on perception-system monitoring [150], which only focuses on the deterministic case and considers a simplified model.

Chapter 4

Task-Aware Perception Monitoring

In this chapter we leverage the fault detection and identification framework developed in the previous chapter to develop a task-aware runtime monitor. We start by defining the notion of task-aware risk (Section 4.1). We then propose an algorithm to estimate the failure risk by using statistical tool called copula (Sections 4.2 and 4.3). We demonstrate, through experiments, that our approach can accurately estimate the risk of perception failures (Section 4.4), and conclude the chapter with a discussion of related work (Section 4.5).

4.1 Risk Estimation Formulation

This section provides an overview of the building blocks of our task-aware perception monitor, which comprises of three components: perception failure detection and identification, plausible scene generator, and task-aware risk estimator. Moreover, the section formalizes the problem of task-aware risk estimation, which is the main focus of this paper. Throughout the rest of this section we will refer to the AV as the *ego vehicle* while any other agent is referred to as a *non-ego agent*.

Perception Failure Detection and Identification. We assume access to a perception failure detection and identification module, such as the algorithms presented in Chapter 3, that identifies the set of failure modes the perception system is experiencing. The perception system is composed of a set of modules, each of which is responsible for a specific task, e.g., object detection, localization, etc. Each module is subject to a finite set of failure modes. The perception failure detection and identification module computes a failure state vector $\boldsymbol{f}$, containing the relevant information about the active failures, that is, the set of active failure modes and the corresponding perception diagnostic information (such as intermediate detection results, raw sensor data, etc.).

Plausible Scene Generator. Before we assess the risk that the perception failure poses to the AV's motion plan, we need to understand the actual scene in which the AV is operating. To this end, we construct a new estimate of the surrounding scene using what we call a *plausible scene generator.* Let $\boldsymbol{x}_t \in \mathbb{X} \subseteq \mathbb{R}^n$ be an estimate of the world state at time t, and $\boldsymbol{f}$ the active perception failure modes provided by failure detection and identification discussed in Chapter 3, the plausible scene generator returns alternative plausible scenes in the form of a probability distribution $\zeta(\hat{\boldsymbol{x}}_t|\boldsymbol{x}_{0:t}, \boldsymbol{f})$ over the plausible world states $\hat{\boldsymbol{x}}_t \in \mathbb{X}$ at time t. We require the plausible scene generator to *support the actual world state* to be able to accurately assess the risk of the perception failure. We will provide more details about the plausible scene generator in Section 4.2.

Relative Scenario Risk. We are interested in understanding how much more risk does the ego's motion plan encounters in the generated plausible scenes $\zeta(\hat{\boldsymbol{x}}_t|\boldsymbol{x}_{0:t}, \boldsymbol{f})$ compared to the perceived scene $\boldsymbol{x}_{0:t}$. Let $\boldsymbol{x}^{\mathrm{e}}_{t:t+T}$ be the ego's motion plan generated by a planning module for a time horizon T. We assume the availability of a trajectory prediction module which provides a distribution $\psi(\boldsymbol{x}_{t:t+T}|\boldsymbol{x}_{0:t}, \boldsymbol{x}^{\mathrm{e}}_{t:t+T})$ on the future world state trajectories conditioned on the world state history and the ego's motion plan. The trajectory predictor ψ reasons about agent interactions and provides multimodal predictions that account for multiple agent intentions; in our experiments we use Trajectron++ [220] which is a state-of-the-art trajectory prediction model that satisfies these criteria. The approach we describe is agnostic to the choice of the planning and prediction modules. Let $c : \mathbb{X} \to \mathbb{R}^+$ be a *cost function* such that higher values imply riskier scenarios for the ego vehicle. Examples of such functions might be the distance between the ego vehicle and the closest non-ego agent or a surrogate (so that higher values imply shorter times) of the time-to-collision metric [221]. The distribution ψ on the future world states $\boldsymbol{x}_{t:t+T}$ induces a sequence of univariate distributions $\{\phi_{t+\tau}(c_{t+\tau}|\boldsymbol{x}_{0:t}, \boldsymbol{x}^{\mathrm{e}}_{t:t+T})\}_{\tau=1}^{T}$ over the predicted costs c for each time step in the future. In the rest of the paper, we will work with the predicted cost distribution $\phi_{t+\tau}$ for a particular τ and for a particular motion plan $\boldsymbol{x}^{\mathrm{e}}_{t:t+T}$. For the sake of notational compactness, we drop the explicit dependence of ϕ on $t+\tau$ and $\boldsymbol{x}^{\mathrm{e}}_{t:t+T}$ to express the predicted cost distribution as $\phi(c|\boldsymbol{x}_{0:t})$. Similarly, the plausible scene distribution $\zeta(\hat{\boldsymbol{x}}_t|\boldsymbol{x}_{0:t}, \boldsymbol{f})$ on the plausible world state $\hat{\boldsymbol{x}}_t$ induces the cost distribution $\phi\left(c|\boldsymbol{x}_{0:t}, \boldsymbol{f}\right)$.

We are now ready to formalize our task-aware notion of risk in the following definition.

Definition 45 (Relative Scenario Risk (RSR))**.** *Let $\boldsymbol{x}_{0:t}$ be the world state history for the perceived scene and let $\zeta(\hat{\boldsymbol{x}}_t|\boldsymbol{x}_{0:t}, \boldsymbol{f})$ be the distribution of the generated plausible scenes due to the perception faults $\boldsymbol{f}$, detected by a perception failure detection and identification module. Let $\phi_A := \phi(c|\boldsymbol{x}_{0:t})$ be the distribution of the costs for the perceived scene and $\phi_B := \phi\left(c|\boldsymbol{x}_{0:t}, \boldsymbol{f}\right)$ be the distribution of the costs for the plausible scenes. Let $\theta \in \mathbb{R}^+$ be the cost threshold that the planner desires to stay below. The* relative scenario risk *(RSR) between the plausible and the perceived scenes is then defined as:*

$$\hat{\mathcal{R}} : \theta \mapsto \underset{A\sim\phi_A, B\sim\phi_B}{Pr}(B > \theta \mid A \leq \theta). \tag{4.1}$$

For a given θ, the higher the RSR is, the further the plausible scene cost distribution ϕ_B is skewed toward higher costs, as illustrated in Fig. 1-2. Hence, larger values of $\hat{\mathcal{R}}$ imply that the plausible scenes are riskier than the perceived scene. Note that, in the general, ϕ_A and ϕ_B will not be independent since the underlying scene is largely the same.

The choice of θ defines a desired safety threshold: for instance, if the cost is the distance between agents, θ can be the smallest acceptable distance between the ego and the nearest agent. The choice of θ may be scenario dependent (e.g., the minimum distance might be different when driving on a highway vs. a traffic jam). To overcome this dependency, we take θ to capture the bulk of the probability mass in the perceived scene cost distribution (ϕ_A). To make this more concrete, let Φ_A be the marginal cumulative distribution function (CDF) of ϕ_A and Φ_B be the CDF of ϕ_B. Recall that the generalized inverse of a CDF Φ, here denoted by Φ^{-1}, is defined as:

$$\Phi^{-1}(p) := \inf \{c \in \mathbb{R} \ : \ \Phi(c) \geq p\} . \tag{4.2}$$

Then, we choose $\theta = \Phi_A^{-1}(p)$. This is equivalent to taking θ to be the maximum value of the risk cost, among the most common situations in the perceived scene. We call p the *risk aversion* parameter as it denotes the amount of risk the ego agent is willing to accept in its motion plan.

We can now define the p-quantile relative scenario risk.

Definition 46 (p-quantile Relative Scenario Risk)**.** *Let $\hat{\mathcal{R}}$ be the relative scenario risk in Definition 45. Let $p \in (0, 1)$ be the risk aversion parameter described above. Then, the p-quantile relative scenario risk (p-RSR) is defined as:*

$$\mathcal{R} : p \mapsto \hat{\mathcal{R}} \circ \Phi_A^{-1}(p). \tag{4.3}$$

Note that the definition above is simply restating Definition 45 in terms of the risk aversion parameter p.

Problem Statement. Given the perceived world state history $\boldsymbol{x}_{0:t}$, the perception module fault modes $\boldsymbol{f}$, and the risk aversion p, we want to estimate the p-quantile relative scenario risk $\mathcal{R}(p)$ in Definition 46. It is worth noting that this is a challenging problem because the distributions ϕ_A and ϕ_B are not independent, and we do not have an explicit analytical representation for them or their CDFs Φ_A, Φ_B; hence, we cannot analytically compute $\Phi_A^{-1}(p)$ or $\mathcal{R}(p)$. However, we can sample from these distributions independently. In particular, we take samples from these distributions in such a way that they are independent (each new sample does not depend on the previous one) and identically distributed (the underlying scene and the behavior of the agents are fixed). In the next section, we present a method to estimate $\mathcal{R}(p)$

using these samples.

4.2 Plausible Scene Generation

The goal of the plausible scene generator is to generate a probabilistic description of the plausible scenes that are consistent with the perceived scene and the active failure modes. Let $\boldsymbol{x}_t \in \mathbb{X} \subseteq \mathbb{R}^n$ be an estimate of the world state at time t, provided by the perception module (this is the "perceived scene"). We assume that the world state includes the ego vehicle's state $\boldsymbol{x}^{\mathrm{e}}$, non-ego agents' states $\boldsymbol{x}^{\mathrm{ne}}$, and map attributes $\boldsymbol{x}^{\mathrm{m}}$, e.g., lane lines, stop signs, traffic signals, etc. Given the perceived world state $\boldsymbol{x}_t$ from the perception module, and the active perception failure modes information $\boldsymbol{f}$ from the perception failure detection and identification, the plausible scene generator returns alternative plausible scenes in the form of a probability distribution $\zeta(\hat{\boldsymbol{x}}_t|\boldsymbol{x}_{0:t}, \boldsymbol{f})$ over the plausible world states $\hat{\boldsymbol{x}}_t \in \mathbb{X}$ at time t; we require the plausible scene generator to *support the actual world state*.

Our framework does not not assume a particular implementation of the plausible scene generator, which might depend on the perception system architecture. While there are some approaches that can be used for scenario generation [38], [39], [222], [223], the approach we adopt here is model-based approach that leverages the results of the perception failure identification method discussed in Chapter 3.

In particular, our plausible scene generator leverages the output of diagnostic tests in conjunction with the active failure modes information $\boldsymbol{f}$ provided by the Failure Identification module. Each failing diagnostic test t_i outputs a set of *candidates* $\mathcal{C}_i$ that are likely to cause the failure mode i. For example, consider the case where the radar-based detection module detects an obstacle in front of the ego-vehicle, but the same obstacle is missed by the camera-based detection module: in this case, the perception system could prioritize the camera detection and discard the radar detection as a false positive; therefore, the perceived scene would have no obstacle in front of the ego-vehicle. However, a tcomparing the two sensor modalities can detect the inconsistency between them and, taking spatiotemporal considerations into account, infer that the missing vehicle is the source of the failure.

In general the diagnostic test cannot distinguish between a missing obstacle and a ghost obstacle, but it can still provide useful information about the failure mode. The task of distinguishing the two cases is left to fault identification module. Once the fault identification module has identified the active failure modes (in our case, a missing obstacle in the camera detections), the plausible scene generator can then use the information about the candidate associated with the active failure to generate an alternative plausible scene that contains an obstacle as detected by the radar. Also, some of the diagnostic tests might fail to detect the failure mode in their scope as active (false negative), even though the failure identification module detects it as active. Therefore, plausible scene generator must also trace back the candidates associated with the active failure modes from other diagnostic tests that are connected in the diagnostic graph.

In general, the plausible scene may not be unique. For example, since the radar-based detection module may only be able to detect the position and the velocity of the missing obstacle, but not its class (e.g., car, pedestrian, etc.), such a detected failure may give rise to a *probability distribution* over plausible scenes $\zeta(\hat{\boldsymbol{x}}_t|\boldsymbol{x}_{0:t}, \boldsymbol{f})$ (e.g., the uniform distribution over the missing obstacle's class). Similarly, this distribution $\zeta(\hat{\boldsymbol{x}}_t|\boldsymbol{x}_{0:t}, \boldsymbol{f})$ can also model the uncertainty in the radar position and velocity measurements. Assuming that at least a module in the perception pipeline computed (even partially) the correct result, the distribution of generated scenes will support the actual scene.[1] It is worth noting that since the plausible scene generator produces a probabilistic distribution, as opposed to a deterministic scenario, re-planning can be computationally infeasible or overly conservative because the distribution can support arbitrarily many plausible scenes.

The plausible scene generator is used in this paper supports the real scene 96.24 % of the time, it correctly remove the closest ghost obstacle 99.77 % of the time, and it correctly adds the closest missing obstacle 91.56 % of the time. This proves that the

[1] This approach can also be generalized to early-fusion and middle-fusion perception systems as spatio-temporal information across frames can provide useful diagnostic information regardless of the perception architecture.

plausible scene generator supports the actual scene in even in challenging scenarios.

4.3 Estimating Relative Risk

In this section, we use *copulas* [224], a statistical tool used to model tail dependencies between distributions, to provide an algorithm to estimate the RSR defined in Definition 46.

4.3.1 Introduction to Copulas

To model the dependence between the two univariate distributions ϕ_A and ϕ_B, we use the concept of *copula* (for a more detailed introduction see [224, Chapter 1]). Copulas are tools for modeling the dependence of multiple random variables, and the name "copula" was chosen to emphasize the way in which a copula couples a joint distribution function to its univariate marginals. We make this mathematically precise in the following definition.

Definition 47 (Copula [45]). *A d-dimensional copula* $C : [0,1]^d \to [0,1]$ *is a function defined on a d-dimensional unit cube* $[0,1]^d$ *that satisfies the following:*

1. $C(u_1, \dots, u_{i-1}, 0, u_{i+1}, \dots, u_d) = 0$ *for any* u_i*,* $i \in \{1, \dots, d\}$*,*

2. $C(1, \dots, 1, u, 1, \dots, 1) = u$ *for any* $u \in [0,1]$ *in any position, and*

3. C *is d-non-decreasing.*[2]

These three properties ensure that the copula behaves like a joint distribution function. To gain intuition, consider each u_i to be a probability in the range $[0,1]$. The first condition says that if the probability of the event associated to u_i is zero, then, regardless of the probability of the other events, the joint probability of all events happening at the same time is zero. Conversely, if all events are sure to occur except one, then the probability of the joint event is the probability of the single

[2] That is, for each hyper-rectangle $B = \prod_{i=1}^{d}[x_i, y_i] \subseteq [0,1]^d$, the C-volume of B is non-negative: $\int_B dC(u) = \sum_{z \in \prod_{i=1}^{d}\{x_i, y+i\}} (-1)^{N(\boldsymbol{z})} C(\boldsymbol{z}) \geq 0$, where $N(\boldsymbol{z}) = \#\{k : z_k = x_k\}$

non-sure event. Finally, the last condition imposes the copula to be non-decreasing in each component.

Sklar's theorem [225], presented next, provides the theoretical foundation for the application of copulas together with the conditions for the existence (and uniqueness) of the copula. Note that In this thesis, we only require the existence of the copula, but we include the uniqueness conditions as well below for the sake of completeness.

Theorem 48 (Sklar's theorem [225])**.** *Let $\Phi(x_1, \ldots, n_d)$ be a joint distribution function, and let Φ_i, $i = 1, \ldots, d$ be the marginal distributions. Then, there exists a copula $C : [0, 1]^d \to [0, 1]$ such that for all $x_1, \ldots, x_d$ in $[-\infty, +\infty]$*

$$\Phi(x_1, \ldots, x_d) = C(\Phi_1(x_1), \ldots, \Phi_d(x_d)). \tag{4.4}$$

Moreover, if the marginals are continuous, then C is unique; otherwise, C is uniquely determined on $\mathrm{Range}\,\Phi_1 \times \cdots \times \mathrm{Range}\,\Phi_d$ where $\mathrm{Range}\,\Phi_i$ denotes the range (image) of Φ_i.

The importance of copulas in the study of multivariate distributions is emphasized by Sklar's theorem, which shows, firstly, that all multivariate distributions can be expressed in terms of copulas, and secondly, that copulas may be used to construct multivariate distribution functions from univariate ones. The latter point is particularly important for us because, as we noted in the previous section, we cannot sample from the joint distribution $\Pr(A, B)$, but we can sample from the marginals $\Pr(A)$ and $\Pr(B)$.

4.3.2 Estimating p-RSR using copulas

Let A, B be random variables drawn from ϕ_A and ϕ_B with CDFs Φ_A and Φ_B, respectively (notation introduced in Definition 45); as a quick reminder, ϕ_A is the cost distribution of the perceived scene and ϕ_B of the plausible scene. Let's assume for a moment that we can estimate the copula relating ϕ_A and ϕ_B. Since the copula $C(A, B)$ contains the information on the dependence structure between (A, B), we can use it to measure the tail dependency between the two distributions. Hence, using the definition of conditional probability and Eq. (4.4) from Theorem 48, we can express p-RSR in Eq. (4.1) as follows:[3]

$$
\begin{aligned}
\mathcal{R}(p) &= \Pr(B > \Phi_A^{-1}(p) \mid A \le \Phi_A^{-1}(p)) \\
&= 1 - \Pr(B \le \Phi_A^{-1}(p) \mid A \le \Phi_A^{-1}(p)) \\
&= 1 - \frac{\Pr(A \le \Phi_A^{-1}(p), B \le \Phi_A^{-1}(p))}{\Pr(A \le \Phi_A^{-1}(p))} \\
&= 1 - \frac{C(\Phi_A \circ \Phi_A^{-1}(p), \Phi_B \circ \Phi_A^{-1}(p))}{\Phi_A \circ \Phi_A^{-1}(p)} \\
&= 1 - \frac{C(p, \Phi_B \circ \Phi_A^{-1}(p))}{p}.
\end{aligned}
\tag{4.5}
$$

Unfortunately, we do not have access to the explicit expression of the two CDFs Φ_A and Φ_B and the copula C, therefore, $\mathcal{R}$ cannot be computed analytically. In what follows, we will provably bound $\mathcal{R}$ by constructing empirical estimates $\Phi_A^{(n)}$ and $\Phi_B^{(n)}$ of the CDFs Φ_A and Φ_B, respectively, with n i.i.d. samples from both, ϕ_A and ϕ_B.

Theorem 49 (PAC bound on p-RSR)**.** *Let $\{A_i\}_{i=1}^n$ and $\{B_i\}_{i=1}^n$ be n i.i.d. samples from CDFs Φ_A and Φ_B, respectively. Let*

$$
\Phi_A^{(n)}(A) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}[A_i \le A], \qquad \Phi_B^{(n)}(B) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}[B_i \le B]
$$

be empirical estimates of Φ_A and Φ_B, respectively. Let the risk aversion parameter $p \in (0,1)$ be as described in Section 4.1 and let $\alpha \in (0,1)$. Then, with probability at least $1-\alpha$:

$$
1 - \frac{\min\{p, \bar{v}(p,\alpha,n)\}}{p} \le \mathcal{R}(p) \le 1 - \frac{\max\{p + \underline{v}(p,\alpha,n) - 1, 0\}}{p},
$$

where

$$
\begin{aligned}
\underline{v}(p,\alpha,n) &= \Phi_B^{(n)} \circ \left[\Phi_A^{(n)} + \epsilon(\alpha,n)\right]^{-1}(p) - \epsilon(\alpha,n) \\
\bar{v}(p,\alpha,n) &= \Phi_B^{(n)} \circ \left[\Phi_A^{(n)} - \epsilon(\alpha,n)\right]^{-1}(p) + \epsilon(\alpha,n)
\end{aligned}
$$

[3] For notational brevity, we are dropping the distributions from which the random variables A and B are drawn from under the probability sign Pr.

and $\epsilon(\alpha, n) = \sqrt{\ln(2/\alpha)/(2n)}$.

Both bounds are sharp in the sense that they can be attained. In particular, the lower bound is attained when ϕ_A and ϕ_B are *perfectly positively dependent* in the sense that B is almost surely a strictly increasing function of A. Conversely, the upper bound is attained when ϕ_A and ϕ_B are *perfectly negatively dependent*, meaning that B is almost surely a strictly decreasing function of A.

The PAC bounds in Theorem 49 are tractable to compute and allow us to estimate p-RSR $\mathcal{R}(p)$ at runtime. In particular the assumption on i.i.d. samples is not restrictive: as we noted in Section 4.1, our problem formulation allows i.i.d. samples. Also we do not assume a particular copula, but only its existence, which is proved by Sklar's theorem.

4.3.3 Triggering Safety Maneuvers

With the results presented in Theorem 49, we have a way to measure whether the plausible scene, compared to the perceived scene, exposes the ego-vehicle to an unwanted risk in terms of probability. However, if the probability is low, it may be detrimental to the ego-vehicle's performance to initiate safety maneuvers to mitigate the risk. In the following, we design a detection algorithm (Algorithm 5) that can be used to detect whether the system is likely to experience a high-risk situation that can be used to trigger safety maneuvers.

Consider a *risk threshold* $\gamma \in (0, 1)$ that denotes high-risk situations. If the lower bound on $\mathcal{R}(p)$ in Theorem 49 is above γ, it means that with probability at least $1-\alpha$, the current scene indeed corresponds to a high-risk situation (in the sense of p-RSR). In such a case, Algorithm 5 detects a task-relevant failure (line 5), which can be used to trigger a safety maneuver. This reasoning can easily be extended to consider multiple thresholds for multiple criticality levels, each associated with different mitigation strategies or different driving scenarios (*e.g.,* highway, urban driving, pick-up/drop-off, etc.).

Algorithm 5 can be summarized as follows: after identifying a failure and gener-

Algorithm 5: p-RSR Detection Algorithm

```
Input: The state $\boldsymbol{x}_{0:t}$, the faults $\boldsymbol{f}$, the cost metric $c$, the risk aversion $p$, the
       confidence level $1-\alpha$, and the risk threshold $\gamma$.
Output: TRUE if critical scenario, FALSE otherwise.
1 $\{A_i\}_{i=1}^n \sim \phi(c|\boldsymbol{x}_{0:t})$, $\{B_i\}_{i=1}^n \sim \phi(c|\boldsymbol{x}_{0:t}, \boldsymbol{f})$ ;
2 $\Phi_A^{(n)}(A) \leftarrow {}^1\!/_n \sum_{i=1}^n \mathbb{1}[A_i \leq a]$ ;
3 $\Phi_B^{(n)}(B) \leftarrow {}^1\!/_n \sum_{i=1}^n \mathbb{1}[B_i \leq b]$ ;
4 if $\min\{p, \bar{v}(p,\alpha,n)\} < p(1-\gamma)$ then
5     return TRUE
6 else
7     return FALSE
8 end
```

ating the plausible scene, the algorithm samples the two scenes (line 1) and estimates the empirical CDFs (lines 2-3). It then returns TRUE if the lower bound in Theorem 49 is above the risk threshold γ, or FALSE otherwise (line 4). The algorithm steps are depicted in Fig. 4-1.

The algorithm has four parameters: the risk aversion $p \in (0,1)$, the risk threshold $\gamma \in (0,1)$, the number of samples $n \in \mathbb{N}$, and the confidence level $1-\alpha \in (0,1)$. The risk aversion p measures the risk tolerance of the ego-vehicle in terms of quantiles of the perceived scene risk distribution. The ego-vehicle is expected to behave safely (*e.g.,* avoid collisions) in situations where the risk cost is below the p-quantile, so higher values indicate higher risk tolerance. Our risk metric p-RSR represents the probability that the plausible scene is riskier than the perceived scene. If this probability is significant, *i.e.,* above the risk threshold γ, the algorithm will classify the scene as high risk. To estimate p-RSR we use the PAC bounds in Theorem 49, which requires choosing a desired confidence level $1-\alpha$ and the number n of predicted cost samples. Clearly, n should be as large as possible to provide tighter bounds, but this is limited by the computational budget of the system. If the values of p, γ, and $1-\alpha$ are close to 1 the algorithm will be imprudent, *i.e.,* it will classify most of the scenes as low risk; on the other hand, if these values are small, the algorithm will be overly prudent, *i.e.,* it will classify most of the scenes as high risk. In our experiments, we found that the values of p, γ, and $1-\alpha$ in the range $[0.9, 0.99]$ provide a good trade-off

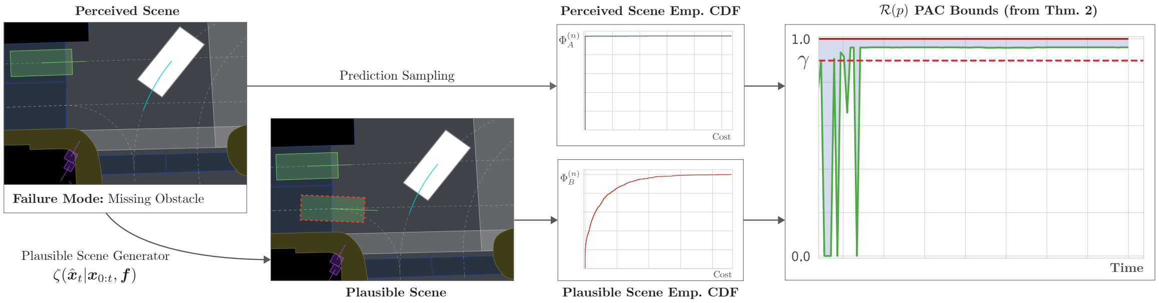

Perceived Scene
Failure Mode: Missing Obstacle
Prediction Sampling
Plausible Scene Generator
$\zeta(\hat{\boldsymbol{x}}_t|\boldsymbol{x}_{0:t}, \boldsymbol{f})$
Plausible Scene
Perceived Scene Emp. CDF
$\Phi_A^{(n)}$
Cost
Plausible Scene Emp. CDF
$\Phi_B^{(n)}$
Cost
$\mathcal{R}(p)$ PAC Bounds (from Thm. 2)
1.0
γ
0.0
Time

Figure 4-1: **Depiction of Algorithm 5.** The perceived scene, which is subject to a missed-obstacle failure, is processed by the plausible scene generator which produces the plausible scene. The two scenes are sampled and the empirical CDFs of the costs are estimated. The perceived scene empirical CDF has a low risk since the only vehicle in the scene is stationary, since it is giving the ego vehicle the right-of-way. However, the plausible scene has a higher risk since the ego vehicle is now in a collision path with a moving vehicle. The two CDFs are used to compute the PAC bounds in Theorem 49. The solid red line represent the upper bound, the solid green line the lower bound, while the dashed red line represent the risk threshold γ. Whenever the lower bound is above γ, the algorithm labels the scenario as high risk.

between prudence and imprudence for the task of detecting collision-prone situations.

4.4 Experimental Results

In this section, we compare the performance of our task-aware risk estimator against various other baselines. Our experiments were conducted on a desktop computer with an `Intel i9-10980XE` 4.7 GHz CPU (36 cores) and an `NVIDIA GeForce RTX 3090` GPU.

4.4.1 Dataset

We tested the proposed approach using the publicly available NuPlan dataset [46]. To test the risk estimation we implement a fault injection mechanism into a NuPlan scenario. We considered 4 classes of failures, namely, *Misdetection*, *Missed Obstacle*, *Ghost Obstacle*, and *Mislocalization*. Each of these classes is further divided into various subclasses.

Misdetection. A misdetection represents an error in the estimation of one of the agents/objects around the vehicle. We consider:

1. *Orientation*: the ego perception system estimates the wrong orientation of the agent;
2. *Size*: the ego perception estimates the wrong size of an agent;
3. *Velocity*: the ego perception estimates the wrong velocity (both direction and/or magnitude) of the agent; and finally
4. *Traffic Light*: the ego perception estimates the wrong status for the traffic light.

All misdetection subtypes (except traffic light) are subject to noise, that can vary across scenarios. For example, an orientation misdetection might offset the vehicle heading with a Gaussian distribution with mean $\pi/6$ and standard deviation of 0.1.
Ghost Obstacle. The ego perception system wrongly detects an obstacle that does not exist (i.e., a ghost obstacle). The ghost obstacle can be:

1. *in-path*, if it lays on the ego trajectory, or
2. *not in-path*, if it is not on the ego trajectory.

Missing Obstacle. The ego perception system fails to detect an agent; the missed obstacle can be:

1. *in-path*, if it is on the ego trajectory, or
2. *not in-path*, if it is not on the ego trajectory.

Mislocalization. The ego perception system fails to localize itself in the map. Each failure mode can be:

1. *static* if the failure persists for the entire duration of a scenario (20 s), or
2. *dynamic*, if it randomly appears/disappears over time.

In our experiments, a dynamic failure mode appears with probability 0.25 and lasts at least 1 s before disappearing.

We manually designed 100 realistic scenarios for evaluation, each with at least one common failure mode typically found in autonomous vehicles; see Table 3.2 for a breakdown of the failures across scenarios. Examples of such realistic scenarios include ghost obstacles, flickering detections, misdetection of a pedestrian crossing the road,incorrect orientation/velocity estimation of a vehicle with the right-of-way, misdetection of the traffic light with incoming traffic, etc.

4.4.2 Implementation Details

We implemented all the components of the proposed approach in Python. As mentioned in Section 4.1, the proposed approach is planner agnostic. In our experiments, we used the Intelligent-Driver Model (IDM) planner [226], [227] provided in the NuPlan-devkit [228]. The planner is designed to move toward the goal, following the lane, while avoiding collisions with the leading agent in front of the ego vehicle. For the non-ego-prediction module, we instead used Trajectron++ [220].

To create high-risk situations, such as collisions, we use the closed-loop capability provided by NuPlan. Closed-loop simulations enable the ego vehicle and other agents to deviate from what was originally recorded in the dataset by the expert driver. In our simulations, each vehicle also behaves according to the IDM policy [226], [227]. However, due to a limitation of the NuPlan simulator, pedestrians and bicycles follow the original trajectory recorded in the dataset (open-loop).

Plausible Scene Generation

The primary goal of the experimental evaluation in this section is to focus on the task-relevant risk-estimation. We use a plausible scene generation method that, given the perception failure mode, proposes a plausible scene by corrupting the ground-truth information (*i.e.,* velocity, size, orientation or location of the agents) with Gaussian noise. In particular, we add zero-mean Gaussian noise with standard deviation 0.2m to the position, 0.1rad to the heading, and 0.1m/s to the velocity magnitude and direction. This approach yields similar results compared to the plausible scene generation presented in Section 4.2 and is also motivated by the following common scenario. Consider a perception system with two sensor modalities, *e.g.,* camera and radar, and a sensor fusion algorithm. Suppose, without loss of generality, that the sensor fusion is misdetecting the velocity of an agent due to a camera-based detection error, while the radar is fault-free. Once the fault detection and identification module recognizes the camera as the cause for the wrong perceived scene, the plausible scene generator could use a Kalman filter to track the radar detections (non-failing sensor modality) to propose a plausible velocity of the vehicle. Since the Kalman filter produces a Gaussian estimate of the uncertainty, the velocity of the vehicle is also Gaussian. This logic can be extended to other failure modes (*e.g.,* missing vehicle), and the plausible scene generator used In this thesis emulates it.

Baselines

We tested our approach against two baselines, one based on the Hamilton-Jacobi (HJ) reachability analysis [229] and another based on the collision probability. **HJ Reachability.** The core idea of HJ-Reachability is computing a set of target states that agents reason about either seeking or avoiding collision within a fixed time horizon. There are two types of agent reactions in HJ-reachability, namely, *collision-seeking* (min) and *collision-avoiding* (max). The idea behind the approach presented in [229] is to compensate for the lack of information about the perception failure by considering the conservative case in which both the agent and the ego vehicle are in a situation where the preferred actions are collision-seeking, namely the *min-min* strategy. For each agent in the scene, the HJ-reachability computes a value function (in our case based on the signed distance between the two bounding boxes), where its zero-sublevel set indicates the existence of a set of control inputs that lead to a collision. The value function is pre-computed offline, and at runtime we perform look-ups, making this approach extremely fast. Since there are multiple agents in the scene, we compute the value function for each agent and then we take the minimum value over the whole scene, if this value is smaller than zero, we say that the scene is high-risk. **Collision Probability.** The second baseline uses the trajectory prediction module to compute the probability of collision with any agents in the scene. Analogous to our proposed approach, this baseline uses the plausible scene to estimate the risk. It samples the trajectories using the trajectory prediction module in both the perceived scene and the plausible scene, and if the collision probability in the plausible scene is greater than in the perceived scene, and the former is above the threshold γ, we say that the scene is high-risk. The key difference between our approach and the collision probability baseline is that the latter does not capture the dependency between the perceived and the plausible scene. Both baselines, Collision Probability and HJ-Reachability, use absolute risk thresholds that do not adapt to the scenarios. In contrast, p-RSR measures the shift in the risk distribution due to the perceptual error between the perceived and plausible scenes, implicitly adapting the risk threshold to the scenario. For example, suppose a correctly detected vehicle cuts into the ego vehicle's lane (high risk), but the speed of a distant cyclist (low risk) is underestimated. Both baselines consider this a high risk scenario because the correctly perceived vehicle is making a risky maneuver, even though the perception error is a low risk. Since the failure does not significantly shift the risk distribution, our approach correctly classifies it as low risk.

Algorithm	Parameters	F1 Score	Accuracy	Precision	Recall	Alarm-to-Collision [s]		Runtime [s]	
						Average	Median	Average	Median
Momentum-Shaped Distance (Proposed)	$p = 0.90, \gamma = 0.9, \alpha = 0.1$	0.70	0.80	0.55	**0.96**	5.28	3.60	0.29	0.2
	$p = 0.95, \gamma = 0.9, \alpha = 0.1$	0.79	0.88	0.68	**0.96**	5.18	3.60		
	$p = 0.99, \gamma = 0.9, \alpha = 0.1$	**0.86**	**0.93**	**0.81**	0.92	4.72	3.03		
Time-To-Collision (Proposed)	$p = 0.90, \gamma = 0.9, \alpha = 0.1$	0.70	0.80	0.55	**0.96**	4.61	2.45	0.26	0.17
	$p = 0.95, \gamma = 0.9, \alpha = 0.1$	0.76	0.86	0.65	0.92	4.16	2.05		
	$p = 0.99, \gamma = 0.9, \alpha = 0.1$	0.79	0.90	0.79	0.79	4.13	2.25		
Collision Probability	$\gamma = 0.90$	0.40	0.32	0.26	**0.96**	6.53	4.95	0.25	0.17
	$\gamma = 0.95$	0.41	0.33	0.26	**0.96**	6.31	4.95		
	$\gamma = 0.99$	0.43	0.41	0.28	0.92	4.33	3.48		
HJ-Reachability	$\gamma = 0$	0.39	0.28	0.24	**0.96**	**7.10**	**5.75**	**0.01**	**0.01**

Table 4.1: **Risk estimation results.** The proposed approach outperforms the other methods in all metrics. In particular, it exhibits the highest precision while being on par with the most conservative methods in terms of recall.

		Predicted	
		High-Risk	Low-Risk
Actual	High-Risk	22 True Positive	2 False Negative
	Low-Risk	5 False Positive	71 True Negative

Table 4.2: **Momentum-Shaped Distance Confusion Matrix.** The proposed approach is able to detect both high-risk and low-risk scenarios reliably, with very few misclassifications. Among the misclassifications, the majority are false positives, which is a desirable property for a safety monitor.

4.4.3 Results

In our experiments we use $n = 1000$ samples from each scene, the confidence $1-\alpha$ and the risk threshold γ are set to 0.9. We tested several values of risk aversion, namely, $p = \{0.9, 0.95, 0.99\}$, reported in Table 4.1. We tested the time-to-collision [221] and the Momentum-Shaped Distance cost metrics (described in Appendix J) to assess the risk. As mentioned before, the time-to-collision computes the time before a collision happens between two actors if their speeds and orientations remain the same. The Momentum-Shaped Distance instead computes the distance between the bounding boxes of two actors, taking into account the relative velocity and orientation of the two actors. The two metrics are described in greater details in Appendix J. We consider a scenario to be high-risk if there is a collision. This also allows us to compute the *Alarm-To-Collision* metric, which measures the time between the first alarm raised by the perception monitor and the actual collision.

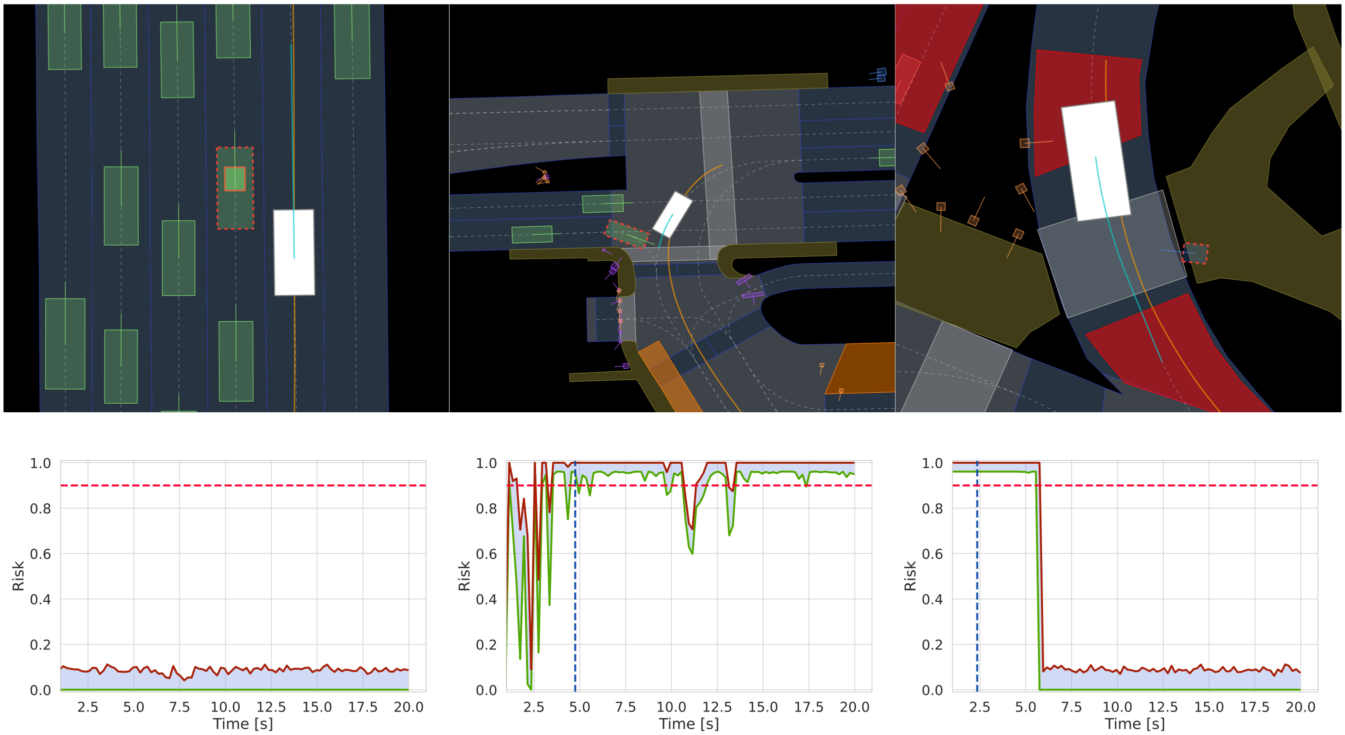
Table 4.1 reports the results averaged across the 100 scenarios. As mentioned above, we consider a scenario to be high-risk if there is a collision (ground-truth label), and report the F1 score, precision, recall, accuracy, and the *Alarm-To-Collision* metric. The proposed approach outperforms both baselines, *i.e.,* HJ-Reachability and collision probability, in terms of F1 Score, precision, recall, and accuracy. In particular, our approach outperforms all others when using the Momentum-Shaped Distance with risk aversion 0.99. Thus, while the baselines achieve similar performance on evidently risky situations (similar recall), our approach demonstrates greater finesse in subtle situations, increasing precision (and hence the F1 score). Besides the relevant classification results, it anticipates the collision by an average of 4.72 s, giving the AV enough time to take risk mitigation actions. HJ-Reachability has the fastest runtime (recall that the value function is precomputed and, at runtime, the approach simply uses a lookup-table), but also the most conservative; it exhibits a high recall (on par with our approach) but the lowest precision, accuracy, and F1 Score.

Table 4.2 reports the confusion matrix for the Momentum-Shaped Distance metric. The table shows that our approach is able to detect both high-risk and low-risk scenarios reliably, with very few misclassifications. It is worth noting that several of the false positives result from situations where there is a failure associated with an agent that is close to the ego vehicle, but the ego vehicle does not collide with it.

Other Runtime Considerations. The Momentum-Shaped Distance with risk aversion 0.99 averages a runtime of 0.29 s and a median of 0.2 s. The bottleneck of the proposed approach is the trajectory sampling, which is performed by the trajectory prediction module, in our case Trajectron++ [220]. From Fig. 4-3 we can see that the trajectory prediction module takes 0.22 s on average, roughly 75 % of the total runtime. This limitation can be easily overcome by using a faster trajectory prediction module, such as PredictionNet [230], which is two orders of magnitude faster than Trajectron++ [230]. Moreover, the proposed approach can be easily parallelized, as the cost computation can be computed in parallel for each agent in the scene and the two scenes can be batched into a single query for the prediction network. With the suggested implementation improvements, we expect to achieve significantly faster runtimes.

4.5 Related Work

We discuss prior art for all of the three stages of our perception monitoring scheme, *i.e.,* perception failure detection, plausible scene generation, and task-relevant risk estimation.

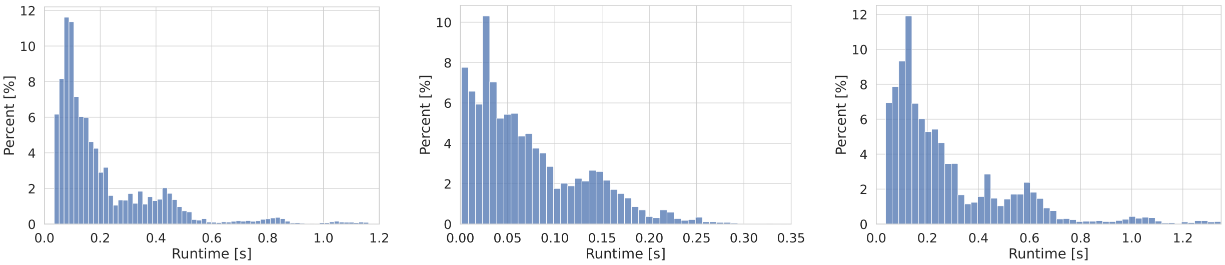


(a) Misdetection (Size)

(b) Missing Obstacle

(c) Missing Pedestrian

Figure 4-2: **Examples of scenarios and the associated estimated risk.** The top row shows the scenario, where the ego vehicle is represented as a white box, other vehicles as green boxes, and pedestrians as blue boxes. A dashed red line indicates the ground truth position and size of an agent, a solid line instead the one perceived by the ego perception system. The bottom row shows the estimated risk for the corresponding scenario in the top row. The horizontal dashed line represents the risk threshold γ. The red solid line represents the risk upper bound while the green line represents the risk lower bound. The vertical dashed blue line represents the time of the collision. It is worth noting that in our simulations, the behavior of the ego vehicle and the non-ego agents does not change after a collision, *i.e.,* the simulation continues running until the end of the scenario.



(a) Prediction Runtime

(b) Risk Estimation Runtime

(c) Total Runtime

Figure 4-3: **Timing breakdown for the proposed approach using Momentum-Shaped Distance.** The prediction runtime averages at 0.22 s (median runtime: 0.14 s). The risk estimation runtime averages at 0.07 s (median runtime: 0.06 s). The total runtime averages at 0.29 s (median runtime: 0.2 s).

Perception Failure Detection and Identification. Autonomous vehicles rely on onboard *perception systems* to provide situational awareness and inform the onboard decision-making and control systems. Reliability of the perception system is critical for safe operation of AVs. While it is desirable for the perception system to be fault-free under any conditions, it is hard to guarantee it [37]; therefore, detection and identification of failures in the perception system at runtime have gained increasing attention. The problem of fault detection and identification is studied in [40] where the authors proposed a system-level framework for online monitoring of the perception system of an AV. Besides failure detection, the framework in [40] also identifies, at runtime, the failure modes that the system is experiencing, from an a priori known list of failures. Other approaches in the literature include spatio-temporal information from motion prediction to assess 3D object detection [206], Timed Quality Temporal Logic (TQTL) to reason about desirable spatio-temporal properties of a perception algorithm [186], [207], or detect anomalies by placing logical-constraint model assertions [208]. Since perception systems are composed of multiple modules, failure detection for specific submodules has also received attention. Previous works focused on object detection [34], semantic segmentation [33], [198], localization [31], [32], out-of-distribution (OOD) detection [127], [231], or changes in high-definition map [199]. All of the above works focus on detecting and identifying failures in the perception system, but do not assess their impact on the AV's motion plan.

Plausible Scene Generation. While there is limited prior work on explicit plausible scene generation, many works in the literature reason about plausible alternative scenes in order to detect or avoid failures. Indeed, failure detection methods often use spatio-temporal inconsistencies between different sensor modalities, where each modality implicitly proposes a plausible scene. You *et al.* [206] use historical information from a number of previous 3D scenes to predict a plausible scene, that is then compared with the perceived scene to detect errors. Similarly, [126] correlates camera images and LIDAR point clouds to detect missing (or ghost) obstacles. Beside obstacle detection, previous work also tackled the mislocalization error. Li *et al.* [232] use particle filters to retain multiple likely positions of the AV. Furthermore, the literature on planning under occlusions contains both model-based [233]–[235] and learning based approaches [38], [39], [223]. These works augment the scene to include possible missing (occluded) obstacles in the scene. The same techniques can be used to generate plausible scenes whenever the failure detection does not provide enough information about the plausible scene.

Risk Assessment. There are several risk assessment techniques in the literature [221], [236]. One approach to "measure" the risk is to monitor the deterioration of the cost of the motion plan, as was done in [237]–[239]. Another approach is to use a *criticality metric*, such as Time To Collision (TTC), which computes the time before a collision happens between two actors if their speeds and orientations remain the same. There is a large variety of criticality metrics in the literature, and the interested reader is referred to [221] for a comprehensive overview; however, these metrics generally only assess whether a scenario is dangerous, while assuming that the inferred scene from perception is correct. A recent approach uses a neural network to classify the risk level using labeled data [240]. However, learning-based methods suffer from lack of OOD robustness, and are usually less interpretable and lack guarantees. Other works on perception-aware risk assessment, such as [241], which proposes risk-ranked recall for object detection systems, and [242], which develops perception-uncertainty-based risk envelopes, do not reason about the future actions of other agents. Topan *et al.* [229] do account for the future actions of other agents for perception failures by computing “inevitable” collision sets (ICS) via Hamilton-Jacobi (HJ) reachability analysis [243], and flagging a situation as unsafe if another agent is close to entering the ICS [244]–[246]. However, [229] assumes the worst-case scenario will occur, *i.e.,* the AV and other road agents will try to collide with each other, and does not consider the interactions between multiple agents, resulting in over-conservatism. In this thesis, we estimate the risk to the AV’s motion plan in the presence of a perception failure while accounting for future actions of other agents by leveraging a trajectory prediction network [220]. Furthermore, our approach is accompanied by PAC bounds and is run-time capable.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Conclusions

In this thesis, we have presented a toolkit that combines advances in outlier-robust estimation, runtime monitoring of perception systems, and risk assessment of autonomous vehicle perception failures, offering contributions to the field of safe and reliable robotic perception.

In Chapter 2 we investigated fundamental computational limits and general-purpose algorithms for outlier-robust estimation. We proved that, in the worst-case, outlier-robust estimation is inapproximable even in quasi-polynomial time. We presented two robust algorithms, ADAPT and GNC, and established convergence results and connections between the corresponding formulations. We proposed the first minimally tuned algorithms, ADAPT-MinT and GNC-MinT. These algorithms offer a new paradigm for resilient life-long estimation, being robust not only against outliers but also against unknown inlier noise statistics. We provided interpretations of maximum consensus and truncated least squares estimation.

In Chapter 3 we proposed a novel approach to fault detection and identification for perception systems. Toward this goal, we formalized the concept of *diagnostic tests*, a generalization of runtime monitors, that return diagnostic information about the presence of failure modes. We then introduced the concept of *diagnostic graph*, as a structure for organizing diagnostic information and its relations with the monitored perception system. We then provided a set of deterministic, probabilistic, and learning-based algorithms that use diagnostic graphs to perform fault detection and identification. In addition to the algorithms, we investigated fundamental limits and provided deterministic and probabilistic guarantees on the fault detection and identification results. These include results on the maximum number of faults that can be correctly identified in a given perception system as well as PAC-bounds on the number of mistakes our fault identification algorithms are expected to make.

Finally, in Chapter 4 we proposed a novel approach to risk assessment of autonomous vehicles perception failures. The risk assessment, together with the fault detection and identification module, allowed us to build a *task-aware perception monitor* that not only identifies the failure modes that perception system is experiencing, but also quantifies the risk that the AV faces in the current scene. We achieved this by first identifying the perception failure mode, followed by synthesizing plausible alternatives for the current scene, and then assessing how much more risk the AV faces in the plausible scene as compared to the perceived one. We formalized the notion of task-aware risk as the *p-quantile relative scenario risk*, and then developed an algorithm to estimate it using i.i.d. samples. Additionally, we provide PAC bounds for our risk estimate which ensure the correctness of our algorithm with high probability.

5.1 Future Work

The research described in this thesis paves the way for several avenues of future research.

The diagnostic graph is a powerful tool for organizing the diagnostic information of a perception system. However, in this thesis, we only used the diagnostic graph to detect and identify faults in a single perception system. In recent years, we saw the emergence of *vehicle-to-vehicle* (V2V) [247]–[249] and *vehicle-to-everything* (V2X) [250], [251] communication technology. This technology enables vehicles to communicate with each other and with the infrastructure, and it is expected to play a key role in the development of autonomous vehicles. Therefore, it is a logical next step to extend the diagnostic graph to a distributed setting, where multiple vehicles and infrastructure nodes can share their diagnostic information. We can envision an augmented diagnostic graph where multiple diagnostic graphs represent the perception system of collaborating vehicles, each connected to the other according to the communication network topology. This opens up the possibility of developing distributed fault detection and identification algorithms, where the additional shared information can improve the accuracy of the fault detection and identification process.

In the experimental sections of this thesis, we assumed that all the diagnostic tests are computed by the perception system at all times. However, sometimes diagnostic tests can be computationally expensive, and it might be desirable to compute them only when needed. One interesting direction is to develop a fault identification algorithm that uses a minimal number of diagnostic tests to identify the active failure modes in the most common cases with sufficient accuracy (diagnosability can be used to guide the selection of the diagnostic tests to request), but in cases where the algorithm is undecided it selectively requests additional diagnostic tests to resolve the ambiguity. This approach can significantly reduce the computational cost of the fault identification process, while maintaining the same (or improve) level of accuracy.

In this work, we estimated the relative scenario risk using a single metric: the momentum-shaped distance or time-to-collision. Our primary goal was to gauge the potential for collision, thus we utilized metrics that capture that risk. However, it's important to note that the relative scenario risk encompasses a broader notion of risk beyond just collision likelihood. Specifically, this metric captures the influence of perception failures on decision-making processes in driving scenarios, as indicated by the selected metric - for example, the risk of collision. A promising direction for future research lies in the development of a comprehensive set of risk metrics that address various concerns relevant to the decision-making layer. Such concerns could include, for instance, collision risk, passenger comfort, or the likelihood of traffic rule violations. Implementing our task-aware risk estimation algorithm alongside diverse metrics could effectively assess the impact of perception failures on these different aspects. Consequently, the decision-making layer could leverage this detailed risk analysis to adapt its behavior to the current situation. Moreover, this work did not explore how to use the risk estimate to inform the decision making process.

We acknowledge the critical importance of advancing decision-making algorithms within autonomous systems to effectively leverage risk-aware perception systems. Currently, our efforts have been concentrated on understanding and quantifying various risk factors. However, the actual development of a risk-aware decision-making algorithm that can dynamically respond to these quantified risks remains an ambitious and vital goal. This development, crucial for enhancing the safety and reliability of autonomous systems, is designated as a key area of future work. We envision an algorithm that not only comprehends the multifaceted nature of risk but also adeptly navigates through it, thereby ensuring safer and more effective decision-making processes in real-time scenarios.

The plausible scene generator is a key component of our risk estimation algorithm. In this thesis, we developed a model-based algorithm that generates plausible scenes by leveraging the candidates produced by the diagnostic tests and by sampling unknown features (*e.g.,* obstacle class) from pre-defined distributions. However, this approach is limited by the ability of the diagnostic test to produce candidates and by the ability of the system designer to model the uncertainty of the unknown features. While the approach is sufficient for many critical scenarios, the development of generative models [38], [222] opens up the possibility of developing a data-driven plausible scene generator that can generate plausible scenes in a more general setting.

Finally, the integration of our algorithms with real-world datasets and other perception subsystems will enhance the practical applicability of our framework. While this research was mainly motivated by safety concerns in autonomous vehicles, the proposed framework can be applied to other perception systems, such the ones used fod manipulation, industrial automation, and mobile robotics. This work does not explore the integration of our algorithms with other perception systems, but we believe that this is a promising direction for future research.

Safety in autonomous systems is a complex problem, and our work is only a small step towards a more comprehensive solution.

Appendix A

Proofs from Chapter 2

A.1 Proof of Proposition 2

We prove that any optimal solution to MC (Eq. (1.2)) is also an optimal solution to Eq. (2.3), and vice versa. To argue this, we use the method of contradiction.

First, assume $(\boldsymbol{x}_{\mathsf{MC}}, \mathcal{O}_{\mathsf{MC}})$ is an optimal solution to MC but not to Eq. (2.3), *i.e.,* there exists an optimal solution $(\boldsymbol{x}_{eq.Eq.\ (2.3)}, \mathcal{O}_{eq.Eq.\ (2.3)})$ to Eq. (2.3) such that $|\mathcal{O}_{eq.Eq.\ (2.3)}| < |\mathcal{O}_{\mathsf{MC}}|$ and $\prod_{i\,\in\,\mathcal{M}\setminus\mathcal{O}_{eq.Eq.\ (2.3)}} u(r(\boldsymbol{y}_i, \boldsymbol{x}_{eq.Eq.\ (2.3)}), \epsilon) \quad > 0$. But the latter inequality implies $r(\boldsymbol{y}_i, \boldsymbol{x}_{eq.Eq.\ (2.3)}) \leq \epsilon$ for all $i \in \mathcal{M} \setminus \mathcal{O}_{eq.Eq.\ (2.3)}$ (since the uniform distribution has non-zero probability only in $[0, \epsilon]$), and, as a result, $(\boldsymbol{x}_{eq.Eq.\ (2.3)}, \mathcal{O}_{eq.Eq.\ (2.3)})$ is feasible in MC and, yet, $|\mathcal{O}_{eq.Eq.\ (2.3)}| < |\mathcal{O}_{\mathsf{MC}}|$, which contradicts optimality.

Now assume $(\boldsymbol{x}_{eq.Eq.\ (2.3)}, \mathcal{O}_{eq.Eq.\ (2.3)})$ is a solution to Eq. (2.3) but not to MC, *i.e.,* there exist a solution $(\boldsymbol{x}_{\mathsf{MC}}, \mathcal{O}_{\mathsf{MC}})$ to MC such that $|\mathcal{O}_{\mathsf{MC}}| < |\mathcal{O}_{eq.Eq.\ (2.3)}|$ and $r(\boldsymbol{y}_i, \boldsymbol{x}_{\mathsf{MC}}) \leq \epsilon$ for all $i \in \mathcal{M}\setminus\mathcal{O}_{\mathsf{MC}}$. But the latter implies that $\prod_{i\,\in\,\mathcal{M}\setminus\mathcal{O}_{\mathsf{MC}}} u(r(\boldsymbol{y}_i, \boldsymbol{x}_{\mathsf{MC}}), \epsilon) = \epsilon^{-|\mathcal{M}\setminus\mathcal{O}_{\mathsf{MC}}|} > 0$, and, as a result, $(\boldsymbol{x}_{\mathsf{MC}}, \mathcal{O}_{\mathsf{MC}})$ is feasible for Eq. (2.3) and, yet, $|\mathcal{O}_{\mathsf{MC}}| < |\mathcal{O}_{eq.Eq.\ (2.3)}|$, which again contradicts optimality.

A.2 Proof of Proposition 3

The proof follows from taking the log of Eq. (2.4).

A.3 Proof of Proposition 4

Assuming a known number of outliers $|\mathcal{O}| = |\mathcal{O}^\circ|$, the TLS formulation in Eq. (2.2) becomes

$$\min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M},\, |\mathcal{O}| = |\mathcal{O}^\circ|}} \sum_{i \in \mathcal{M} \setminus \mathcal{O}} r^2(\boldsymbol{y}_i, \boldsymbol{x}) \;+\; \epsilon^2 |\mathcal{O}^\circ|, \tag{A.1}$$

where $\epsilon^2|\mathcal{O}^\circ|$ becomes a constant and is irrelevant for the optimization. It can be now seen that taking the $-\log(\cdot)$ of the objective in Eq. (2.5) leads to the same optimization as in Eq. (A.1).

A.4 Proof of Proposition 5

Since $\boldsymbol{x}^{\circ}$ is feasible in Eq. (2.7) and $\prod_{i \in \mathcal{M}} \hat{g}(r(\boldsymbol{y}_i, \boldsymbol{x}^{\circ})) > 0$ (since $r(\boldsymbol{y}_i, \boldsymbol{x}^{\circ}) \leq \alpha$ for any $i \in \mathcal{M}$), for any optimal solution $\boldsymbol{x}$ to Eq. (2.7), it also holds true that $\prod_{i \in \mathcal{M}} \hat{g}(r(\boldsymbol{y}_i, \boldsymbol{x})) > 0$, and, as a result, $r(\boldsymbol{y}_i, \boldsymbol{x}) \leq \alpha$ for any $i \in \mathcal{M}$. Therefore, after simplifying constants, Eq. (2.7) is equivalent to

$$\max_{\boldsymbol{x} \in \mathcal{X}} \quad \prod_{i \in \mathcal{M}} \max \left\{ e^{-r^2/2}, e^{-\epsilon^2/2} \right\}, \tag{A.2}$$

which is equivalent to

$$\max_{\boldsymbol{x} \in \mathcal{X}} \quad \sum_{i \in \mathcal{M}} \max \left\{ \, -r^2/2 \; , -\epsilon^2/2 \, \right\}, \tag{A.3}$$

since maximizing the objective function in Eq. (A.2) is equivalent to maximizing the log of it. Finally, Eq. (A.3) is equivalent to $\max_{\boldsymbol{x} \in \mathcal{X}} \ \sum_{i \in \mathcal{M}} \min \{r^2, \epsilon^2\}$, which is equivalent to TLS.

A.5 Proof of Theorem 6

Denote by $(\boldsymbol{x}_{\mathsf{G-TLS}}, \mathcal{O}_{\mathsf{G-TLS}})$ any optimal solution to G-TLS. We first prove $(\boldsymbol{x}_{\mathsf{G-TLS}}, \mathcal{O}_{\mathsf{G-TLS}})$ is feasible to G-MC (*i.e.*, $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{G-TLS}}}, \boldsymbol{x}_{\mathsf{G-TLS}}) \|_\infty \leq \epsilon$), and, then, prove $(\boldsymbol{x}_{\mathsf{G-TLS}}, \mathcal{O}_{\mathsf{G-TLS}})$ is actually an optimal solution to G-MC.

To prove $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{G-TLS}}}, \boldsymbol{x}_{\mathsf{G-TLS}}) \|_\infty \leq \epsilon$, first observe

$$\begin{aligned} |\mathcal{M} \setminus \mathcal{O}_{\mathsf{G-TLS}}| \cdot \| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{G-TLS}}}, \boldsymbol{x}_{\mathsf{G-TLS}}) \|_\infty^2 + \\ \epsilon^2 |\mathcal{O}_{\mathsf{G-TLS}}| \leq \epsilon^2 |\mathcal{M}|, \end{aligned} \tag{A.4}$$

since ϵ^2 is the value of G-TLS's objective function for $\mathcal{O} = \mathcal{M}$ (given any $\boldsymbol{x} \in \mathcal{X}$), while $(\boldsymbol{x}_{\mathsf{G-TLS}}, \mathcal{O}_{\mathsf{G-TLS}})$ is an optimal solution to G-TLS. Now, assume $\|\boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{G-TLS}}}, \boldsymbol{x}_{\mathsf{G-TLS}})\|_\infty > \epsilon$. Then, the value of G-TLS's objective function at $(\boldsymbol{x}_{\mathsf{G-TLS}}, \mathcal{O}_{\mathsf{G-TLS}})$ is strictly more than $\epsilon^2|\mathcal{M}|$, which contradicts Eq. (A.4). Hence, $\| \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{G-TLS}}}, \boldsymbol{x}_{\mathsf{G-TLS}}) \|_\infty \leq \epsilon$, and, as a result, $(\boldsymbol{x}_{\mathsf{G-TLS}}, \mathcal{O}_{\mathsf{G-TLS}})$ is feasible to G-MC.

We now prove $\mathcal{O}_{\mathsf{G-TLS}}$ is also optimal for G-MC. Assume by contradiction $\mathcal{O}_{\mathsf{G-TLS}}$ is not optimal for G-MC. Then, $|\mathcal{O}_{\mathsf{G-MC}}| < |\mathcal{O}_{\mathsf{G-TLS}}|$ (or, equivalently, $|\mathcal{O}_{\mathsf{G-MC}}|+1 \leq |\mathcal{O}_{\mathsf{G-TLS}}|$), since $\mathcal{O}_{\mathsf{G-MC}}$ is optimal. Since also $\|\boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{G-MC}}}, \boldsymbol{x}) \|_\infty < \epsilon$, the following hold:

$$\|\boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{G-MC}}}, \boldsymbol{x}) \|_\infty^2 + \epsilon^2 |\mathcal{O}_{\mathsf{G-MC}}| < \tag{A.5}$$

$$\epsilon^2 + \epsilon^2 |\mathcal{O}_{\mathsf{G-MC}}| = \tag{A.6}$$

$$\epsilon^2 (|\mathcal{O}_{\mathsf{G-MC}}|+1) \leq \epsilon^2 |\mathcal{O}_{\mathsf{G-TLS}}| \leq \tag{A.7}$$

$$|\mathcal{M} \setminus \mathcal{O}_{\mathsf{G-TLS}}| \cdot \|\boldsymbol{r}(\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}_{\mathsf{G-TLS}}}, \boldsymbol{x}) \|_\infty^2 + \epsilon^2 |\mathcal{O}_{\mathsf{G-TLS}}|. \tag{A.8}$$

Comparing Eq. (A.5) and Eq. (A.8), we notice that $\mathcal{O}_{\mathsf{G-MC}}$ achieves a better cost in G-TLS, contradicting the optimality of $\mathcal{O}_{\mathsf{G-TLS}}$.

A.6 Proof of Theorem 7

To prove the theorem, consider the following problem:

$$\min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M}}} \parallel \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \backslash \mathcal{O}}, \boldsymbol{x}) \parallel_2^2 \quad \text{s.t.} \quad |\mathcal{O}| = |\mathcal{O}_{\mathsf{MTS}}|. \tag{A.9}$$

Note that $\mathcal{O}_{\mathsf{MTS}}$ is feasible for MTS, hence the optimal objective of Eq. (A.9) is smaller than τ^2.

Now consider the Lagrangian of Eq. (A.9):

$$\begin{aligned} l(\epsilon) &\triangleq \min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M}}} \parallel \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \backslash \mathcal{O}}, \boldsymbol{x}) \parallel_2^2 + \epsilon^2 \left(|\mathcal{O}| - |\mathcal{O}_{\mathsf{MTS}}|\right) \\ &= f_{\mathsf{TLS}}(\epsilon) - \epsilon^2 |\mathcal{O}_{\mathsf{MTS}}|. \end{aligned} \tag{A.10}$$

By weak duality [252]:

$$f_{\mathsf{TLS}}(\epsilon) - \epsilon^2 |\mathcal{O}_{\mathsf{MTS}}| \leq \parallel \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \backslash \mathcal{O}_{Eq.\ (A.9)}}, \boldsymbol{x}_{Eq.\ (A.9)}) \parallel_2^2, \tag{A.11}$$

where $(\boldsymbol{x}_{Eq.\ (A.9)}, \mathcal{O}_{Eq.\ (A.9)})$ is an optimal solution to Eq. (A.9). Since $\parallel \boldsymbol{r}(\boldsymbol{y}_{\mathcal{M} \backslash \mathcal{O}_{Eq.\ (A.9)}}, \boldsymbol{x}_{Eq.\ (A.9)}) \parallel_2 \leq \tau$, then

$$f_{\mathsf{TLS}}(\epsilon) - \epsilon^2 |\mathcal{O}_{\mathsf{MTS}}| \leq \tau^2, \tag{A.12}$$

From the inequality Eq. (A.12) it follows:

- if $\tau^2 = r_{\mathsf{TLS}}^2(\epsilon)$, then Eq. (A.12) implies $|\mathcal{O}_{\mathsf{TLS}}| \leq |\mathcal{O}_{\mathsf{MTS}}|$; since $(\boldsymbol{x}_{\mathsf{TLS}}, \mathcal{O}_{\mathsf{TLS}})$ is also feasible for Eq. (G-MC), $|\mathcal{O}_{\mathsf{TLS}}| = |\mathcal{O}_{\mathsf{MTS}}|$, and $(\boldsymbol{x}_{\mathsf{TLS}}, \mathcal{O}_{\mathsf{TLS}})$ is also a solution to MTS.

- if $\tau^2 > r_{\mathsf{TLS}}^2(\epsilon)$, then $|\mathcal{O}_{\mathsf{TLS}}| \geq |\mathcal{O}_{\mathsf{MTS}}|$, since $(\boldsymbol{x}_{\mathsf{TLS}}, \mathcal{O}_{\mathsf{TLS}})$ is feasible in Eq. (G-MC).

- if $\tau^2 < r_{\mathsf{TLS}}^2(\epsilon)$, then $|\mathcal{O}_{\mathsf{TLS}}| < |\mathcal{O}_{\mathsf{MTS}}|$, since $(\boldsymbol{x}_{\mathsf{TLS}}, \mathcal{O}_{\mathsf{TLS}})$ is infeasible in Eq. (G-MC).

A.7 Proof of Theorem 12

We prove the theorem based on the inapproximability of the *variable selection* problem, reviewed in A.7. In particular, we first prove the inapproximability of $\mathsf{G\text{-}MC}$, by proving the inapproximability of MTS and MC (A.8 and A.9, respectively). Then, we prove the inapproximability of $\mathsf{G\text{-}TLS}$, by proving the inapproximability of TLS (A.10). For all cases we consider a linear measurement model, which results in residuals of the form:

$$r(\boldsymbol{y}_i, \boldsymbol{x}) = |y_i - \boldsymbol{a}_i^{\mathsf{T}}\boldsymbol{x}|,$$

for all $i \in \mathcal{M}$, where y_i is scalar, and $\boldsymbol{a}_i$ is a column vector.

Preliminary Definitions and Results

We present the *variable selection* problem, recall a known result on its inapproximability in even quasi-polynomial time, and review results that we will subsequently use for the proof of Theorem 12. We use the standard notation $\|\boldsymbol{x}\|_0$ to denote the number of non-zero elements in $\boldsymbol{x}$.

Problem 3 (Variable selection)**.** *Assume a matrix $\boldsymbol{U} \in \mathbb{R}^{\phi \times m}$, a vector $\boldsymbol{z} \in \mathbb{R}^{\phi}$, and a non-negative scalar ξ. Find a vector $\boldsymbol{d} \in \mathbb{R}^m$ that solves the optimization problem*

$$\min_{\boldsymbol{d} \in \mathbb{R}^m} \quad \| \boldsymbol{d} \|_0, \quad \text{s.t.} \quad \| \boldsymbol{U}\boldsymbol{d} - \boldsymbol{z} \|_2 \leq b. \tag{A.13}$$

The following lemma describes inapproximable instances of *variable selection* even in quasi-polynomial time.

Lemma 50 (Inapproximability of Variable Selection in Quasi-polynomial Time [253, Proposition 6])**.** *For any $\delta \in (0,1)$, unless $\mathsf{NP} \notin \mathsf{BPTIME}(m^{\mathsf{poly}\log m})$, there exist*

- *a function $q_1(m) = 2^{\Omega(\log^{1-\delta} m)}$,*
- *a polynomial $p_1(m) = O(m)$,*
- *a polynomial $\xi(m)$,*

- *a polynomial* $\phi(m)$*,*
- *and a zero-one matrix* $\boldsymbol{U} \in \mathbb{R}^{\phi(m)\times m}$*,*

such that, for large enough m*, no quasi-polynomial algorithm finds a* $\boldsymbol{d} \in \mathbb{R}^m$ *distinguishing the mutually-exclusive cases:*[1]

S$_1$. *There exists a vector* $\boldsymbol{d} \in \mathbb{R}^m$ *such that* $\boldsymbol{U}\boldsymbol{d} = \mathbf{1}_{\phi(m)}$ *and* $\| \boldsymbol{d} \|_0 \leq p_1(m)$*.*

S$_2$. *For any* $\boldsymbol{d} \in \mathbb{R}^m$*, if* $\| \boldsymbol{U}\boldsymbol{d} - \mathbf{1}_{\phi(m)} \|_2^2 \leq \xi(m)$*, then* $\| \boldsymbol{d} \|_0 \geq p_1(m)q_1(m)$*.*

The observation holds true even if the algorithm knows that $\boldsymbol{U}\boldsymbol{d} = \mathbf{1}_{\phi(m)}$ *is feasible for some* $\boldsymbol{y} \in \mathbb{R}^m$*, where* $\boldsymbol{y}$ *itself is unknown to the algorithm but* $\| \boldsymbol{y} \|_0$ *is known.*

In the next section, we use the inapproximability of variable selection to prove that MTS is inapproximable. Towards this goal, we prove two intermediate results.

We start with the following optimization problem and prove that it is also inapproximable:

$$\min_{\boldsymbol{d} \in \mathbb{R}^m} \quad \| \boldsymbol{d} \|_0, \quad \text{s.t.} \quad \boldsymbol{U}\boldsymbol{d} = \mathbf{1}_{\phi(m)}. \tag{A.14}$$

Proof that Eq. (A.14) is inapproximable: It suffices to set $b = 0$ in Eq. (A.13), and then apply Lemma 50. □

Given Eq. (A.14)'s inapproximability, we now prove the inapproximability of the optimization problem

$$\min_{\substack{\boldsymbol{d} \in \mathbb{R}^m \\ \boldsymbol{x} \in \mathbb{R}^n}} \quad \| \boldsymbol{d} \|_0, \text{ s.t. } \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}, \tag{A.15}$$

for an appropriate class of matrices $\boldsymbol{A}$.

Proof that Eq. (A.15) is inapproximable: Given the inapproximable instances of Eq. (A.14) (see Lemma 50), consider the instances for Eq. (A.15) where (i) $\boldsymbol{y}$ is any solution to $\boldsymbol{U}\boldsymbol{y} = \mathbf{1}_{\phi(m)}$ (because of Lemma 50, such a $\boldsymbol{y}$ exists), and (ii) $\boldsymbol{A}$ is a matrix in $\mathbb{R}^{m\times n}$, where $n = m - \text{rank}(\boldsymbol{U})$, such that the columns of $\boldsymbol{A}$ span the null space of $\boldsymbol{U}$ ($\boldsymbol{U}\boldsymbol{A} = \mathbf{0}$). Any such instance is constructed in polynomial time in

[1] If m is large enough, then $q_1(m) > 1$ (since $q_1(m) = 2^{\Omega(\log^{1-\delta} m)}$, where $\delta \in (0, 1)$), and, as a result, S$_1$ and S$_2$ are mutually exclusive.

m, since solving a system of equations and finding eigenvectors that span a matrix's null space happen in polynomial time.

We now prove the following statements are indistinguishable, where we consider $\xi'(m) \triangleq \phi^{-2.5}(m)\xi(m)$:

- S_1'. *There exist* $\boldsymbol{d} \in \mathbb{R}^m$ *and* $\boldsymbol{x} \in \mathbb{R}^n$ *such that* $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}$ *and* $|| \ \boldsymbol{d} \ ||_0 \leq p_1(m)$.
- S_2'. *For any* $\boldsymbol{d} \in \mathbb{R}^m$ *and* $\boldsymbol{x} \in \mathbb{R}^n$, *if* $||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{d}||_2^2 \leq \xi'(m)$, then $|| \ \boldsymbol{d} \ ||_0 \geq p_1(m)q_1(m)$.

To this end, we prove that (i) if S_1 is true (which is for any feasible $\boldsymbol{d}$ in Eq. (A.14)), then S_1' also is, and (ii) if S_2 is true, then also S_2' is. Therefore, no quasi-polynomial time algorithm can distinguish S_1' and S_2', since the opposite would contradict that S_1 and S_2 are indistinguishable. In particular:

a) Proof that when S_1 is true then S_1' also is Since $\boldsymbol{U}\boldsymbol{y} = \boldsymbol{U}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{U}\boldsymbol{d}$ implies that $\boldsymbol{1}_{\phi(m)} = \boldsymbol{U}\boldsymbol{d}$, if S_1 is true, then S_1' also is; moreover, $\boldsymbol{x}$ is the unique solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} - \boldsymbol{d}$ ($\boldsymbol{x}$ is unique since $\boldsymbol{A}$ is full column rank).

b) Proof that when S_2 is true then S_2' also is Assume $\boldsymbol{d} \in \mathbb{R}^m$ and $\boldsymbol{x} \in \mathbb{R}^n$ such that $|| \ \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{d} \ ||_2^2 \leq \xi'(m)$ and $|| \ \boldsymbol{d} \ ||_0 < p_1(m)q_1(m)$. If $||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{d}||_2^2 \leq \xi'(m)$, then $|| \ \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{d} \ ||_1^2 \leq [\phi(m)]^{0.5} \ \xi'(m)$, due to norms' equivalence. Hence, $|| \ \boldsymbol{U} \ ||_1^2 \ ||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{d}||_1^2 \leq || \ \boldsymbol{U} \ ||_1^2 \ [\phi(m)]^{0.5} \ \xi'(m)$, which implies $|| \ \boldsymbol{U}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{d}) \ ||_1^2 \leq || \ \boldsymbol{U} \ ||_1^2 \ [\phi(m)]^{0.5} \ \xi'(m)$, *i.e.*, $|| \ \boldsymbol{1}_{\phi(m)} - \boldsymbol{U}\boldsymbol{d} \ ||_1^2 \leq || \ \boldsymbol{U} \ ||_1^2 \ [\phi(m)]^{0.5} \ \xi'(m)$, and as a result $|| \ \boldsymbol{1}_{\phi(m)} - \boldsymbol{U}\boldsymbol{d} \ ||_1^2 \leq \phi(m)^{2.5} \ \xi'(m)$, where the last holds true because $\boldsymbol{U}$ is a zero-one matrix. Consequently, $||\boldsymbol{1}_{\phi(m)} - \boldsymbol{U}\boldsymbol{d}||_2^2 \leq [\phi(m)]^{2.5} \ \xi'(m)$, due to norms' equivalence. Finally, due to $\xi'(m)$'s definition, $[\phi(m)]^{2.5} \ \xi'(m) = \xi(m)$; thus, $|| \ \boldsymbol{1}_{\phi(m)} - \boldsymbol{U}\boldsymbol{d} \ ||_2^2 \leq \xi(m)$. Overall, there exist $\boldsymbol{d}$ such that $|| \ \boldsymbol{1}_{\phi(m)} - \boldsymbol{U}\boldsymbol{d} \ ||_2^2 \leq \xi(m)$ and $|| \ \boldsymbol{d} \ ||_0 < p_1(m)q_1(m)$, which contradicts S_2. □

A.8 Proof that MTS is Inapproximable

We use the notation:

- $\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} \triangleq \{y_i\}_{i\,\in\,\mathcal{M}\setminus\mathcal{O}}$, *i.e.*, $\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}}$ is the stack of all measurements $i \in \mathcal{M} \setminus \mathcal{O}$;
- $\boldsymbol{d}_{\mathcal{M}\setminus\mathcal{O}} \triangleq \{d_i\}_{i\,\in\,\mathcal{M}\setminus\mathcal{O}}$, *i.e.*, $\boldsymbol{d}_{\mathcal{M}\setminus\mathcal{O}}$ is the stack of all noises $i \in \mathcal{M} \setminus \mathcal{O}$;
- $\boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}} \triangleq \{\boldsymbol{a}_i^{\mathsf{T}}\}_{i\,\in\,\mathcal{M}\setminus\mathcal{O}}$, *i.e.*, $\boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}$ is the matrix with rows the row-vectors $\boldsymbol{a}_i^{\mathsf{T}}$, $i \in \mathcal{M} \setminus \mathcal{O}$.

The MTS problem in Eq. (2.1) now takes the form

$$\min_{\substack{\mathcal{O}\,\subseteq\,\mathcal{M} \\ \boldsymbol{x}\,\in\,\mathbb{R}^n}} \quad |\mathcal{O}|, \quad \text{s.t.} \quad \|\, \boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} - \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x} \,\|_2^2 \leq \tau^2. \tag{A.16}$$

To prove Eq. (A.16)'s inapproximability, we first consider an inapproximable instance of Eq. (A.15), and in Eq. (A.16) let $\mathcal{M} = \{1, 2, \dots, m\}$ and $\tau^2 = \xi'(m)$. Then, we prove the following statements are indistinguishable:

S_1''. *There exist $\mathcal{O} \subseteq \mathcal{M}$ and $\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} = \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x}$ and $|\mathcal{O}| \leq p_1(m)$.*

S_2''. *For any $\mathcal{O} \subseteq \mathcal{M}$ and $\boldsymbol{x} \in \mathbb{R}^n$, if $\|\, \boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} - \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x} \,\|_2^2 \leq \xi'(m)$, then $|\mathcal{O}| \geq p_1(m)q_1(m)$.*

To this end, we prove that (i) if S_1' is true, then S_1'' also is, and (ii) if S_2' is true, then also S_2'' is. In more detail:

a) Proof that if S_1' is true then S_1'' also is Assume S_1' is true and let $\mathcal{O} = \{i \;\text{ s.t. }\; d_i \neq 0, \quad i \in \mathcal{M}\}$. Then, $\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} = \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x}$, since $\boldsymbol{d}_{\mathcal{M}\setminus\mathcal{O}} = 0$ and $|\mathcal{O}| = \|\, \boldsymbol{d} \,\|_0 \leq p_1(m)$.

b) Proof that if S_2' is true then S_2'' also is Assume $\mathcal{O} \subseteq \mathcal{M}$ and $\boldsymbol{x} \in \mathbb{R}^n$ such that $\|\, \boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} - \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x} \,\|_2^2 \leq \xi'(m)$ and $|\mathcal{O}| < p_1(m)q_1(m)$. Let $\boldsymbol{d}_{\mathcal{M}\setminus\mathcal{O}} = 0$, and $\boldsymbol{d}_{\mathcal{O}} = \boldsymbol{y}_{\mathcal{O}} - \boldsymbol{A}_{\mathcal{O}}\boldsymbol{x}$. Then, $\|\, \boldsymbol{d} \,\|_0 = |\mathcal{O}| < p_1(m)q_1(m)$ and $\|\, \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{d} \,\|_2^2 = \|\, \boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} - \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x} \,\|_2^2 \leq \xi'(m)$, which contradicts S_2'.

A.9 Proof that MC is Inapproximable

The proof proceeds along the same line of MTS's proof. We use the same notation used in Appendix A.8.

We first consider an inapproximable instance of Eq. (A.15), and in Eq. (1.2) set $\epsilon^2 = \xi'(m)$. We then prove that the following statements are indistinguishable:

- S_1'''. *There exist $\mathcal{O} \subseteq \mathcal{M}$ and $\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} = A_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x}$ and $|\mathcal{O}| \leq p_1(m)$.*
- S_2'''. *For any $\mathcal{O} \subseteq \mathcal{M}$ and $\boldsymbol{x} \in \mathbb{R}^n$, if $||\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} - \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x}||_\infty^2 \leq \xi'(m)$, then $|\mathcal{O}| \geq p_1(m)q_1(m)$.*

To this end, we prove that (i) if S_1'' is true, then S_1''' also is, and (ii) if S_2'' is true, then also S_2''' is. Specifically:

a) Proof that if S_1' is true then S_1''' also is Assume S_1' is true and let $\mathcal{O} = \{i \ \text{ s.t. } \ d_i \neq 0, \quad i \in \mathcal{M}\}$. Then, $\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} = \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x}$, since $\boldsymbol{d}_{\mathcal{M}\setminus\mathcal{O}} = 0$ and $|\mathcal{O}| = ||\ \boldsymbol{d}\ ||_0 \leq p_1(m)$.

b) Proof that if S_2' is true then S_2''' also is Consider $\mathcal{O} \subseteq \mathcal{M}$ and $\boldsymbol{x} \in \mathbb{R}^n$ such that $||\ \boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} - \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x}\ ||_1^2 \leq \xi'(m)$ and $|\mathcal{O}| < p_1(m)q_1(m)$. Let $d_{\mathcal{M}\setminus\mathcal{O}} = 0$, and $\boldsymbol{d}_{\mathcal{O}} = \boldsymbol{y}_{\mathcal{O}} - \boldsymbol{A}_{\mathcal{O}}\boldsymbol{x}$. Then, $||\ \boldsymbol{d}\ ||_0 = |\mathcal{O}| < p_1(m)q_1(m)$ and $||\ \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{d}\ ||_1^2 \leq ||\ \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{d}\ ||_2^2 = ||\ \boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} - \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x}\ ||_2^2 \leq \xi'(m)$, where the first inequality holds due to the norms' equivalence, while the latter inequality contradicts S_2'.

A.10 Proof that TLS problem is Inapproximable

We prove the inapproximability of Eq. (TLS) by using the inapproximability of Eq. (A.16). To this end, we use the notation in A.8, along with the notation

$$f(\boldsymbol{x},\boldsymbol{w}) \triangleq \sum_{i \,\in\, \mathcal{M}} \min_{w_i \,\in\, \{0,1\}} \left[w_i \left(y_i - \boldsymbol{a}_i^\mathsf{T}\boldsymbol{x}\right)^2 + (1-w_i)\,\epsilon^2 \right].$$

Consider an inapproximable instance of Eq. (A.16), and in Eq. (TLS) set $\epsilon^2 = {}^1/_{p_1(m)}$. We prove the following are indistinguishable:

$\bar{\mathrm{S}}_1$. *There exist* $\boldsymbol{w} \in \{0,1\}^m$ *and* $\boldsymbol{x} \in \mathbb{R}^n$ *such that* $f(\boldsymbol{x},\boldsymbol{w}) \leq 1$ *and* $\|\ \boldsymbol{w}\ \|_0 \leq p_1(m)$.

$\bar{\mathrm{S}}_2$. *For any* $\boldsymbol{w} \in \{0,1\}^m$ *and* $\boldsymbol{x} \in \mathbb{R}^n$*, if* $f(\boldsymbol{x},\boldsymbol{w}) \leq \xi'(m)$*, then* $\|\ \boldsymbol{w}\ \|_0 \geq p_1(m)q_1(m)$.

To this end, we prove that (i) if S_1'' is true, then $\bar{\mathrm{S}}_1$ also is, and (ii) if S_2'' is true, then also $\bar{\mathrm{S}}_2$ is. Specifically:

a) Proof that if $\mathbf{S}_1''$ is true then $\bar{\mathbf{S}}_1$ also is Assume S_1'' is true and let $w_i = 1$ for all $i \in \mathcal{O}$, and 0 otherwise. Then, $||\boldsymbol{w}||_0 = |\mathcal{O}| \leq p_1(m)$, and $f(\boldsymbol{x},\boldsymbol{w}) = |\mathcal{O}|\epsilon^2 \leq p_1(m)\epsilon^2 = 1$.

b) Proof that if $\mathbf{S}_2''$ is true then $\bar{\mathbf{S}}_2$ also is Assume $\boldsymbol{w} \in \{0,1\}^m$ and $\boldsymbol{x} \in \mathbb{R}^n$ such that $f(\boldsymbol{x},\boldsymbol{w}) \leq \xi'(m)$ and $\|\ \boldsymbol{w}\ \|_0 < p_1(m)q_1(m)$. Let $\mathcal{O} = \{i \text{ s.t. } w_i = 1\}$, and as a result, $|\mathcal{O}| < p_1(m)q_1(m)$. Since $f(\boldsymbol{x},\boldsymbol{w}) \leq \xi'(m)$ and $f(\boldsymbol{x},\boldsymbol{w}) = \|\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} - \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x}\|_2^2$, it holds true that $\|\boldsymbol{y}_{\mathcal{M}\setminus\mathcal{O}} - \boldsymbol{A}_{\mathcal{M}\setminus\mathcal{O}}\boldsymbol{x}\|_2^2 \leq \xi'(m)$, which contradicts S_2''.

A.11 Proof of Theorem 15

The proof follows by taking $t \to +\infty$ (or, equivalently $\mu^{(t)} \to +\infty$) in Eq. (2.12). In more detail, it suffices to observe that $\lim_{t\to+\infty} \mu^{(t-1)} \,/\, (\mu^{(t-1)}{+}1) = 1$, $\lim_{t\to+\infty} (\mu^{(t-1)}{+}1)/\mu^{(t-1)} = 1$, and $\lim_{t\to+\infty} (\epsilon\sqrt{\mu^{(t-1)}(\mu^{(t-1)}+1)}/r_i^{(t)} - \mu^{(t-1)}) = 1/2$. In particular, the latter is true since $\lim_{t\to+\infty} (\epsilon\sqrt{\mu^{(t-1)}(\mu^{(t-1)}+1)}/r_i^{(t)} - \mu^{(t-1)}) = \lim_{t\to+\infty}[\epsilon\sqrt{\mu^{(t-1)}+1}/(\sqrt{\mu^{(t-1)}}r_i^{(t)})$ $1]/(1/\mu^{(t-1)})$, where now L'Hôspital's rule implies the latter is equal to

$$\lim_{t\to+\infty} \frac{\frac{d}{d\mu^{(t-1)}}\left(\frac{\epsilon\sqrt{\mu^{(t-1)}+1}}{r_i^{(t)}\sqrt{\mu^{(t-1)}}} - 1\right)}{\frac{d}{d\mu^{(t-1)}}\left(\frac{1}{\mu^{(t-1)}}\right)} =$$

$$\lim_{t\to+\infty} \frac{\epsilon}{r_i^{(t)}} \frac{\frac{\sqrt{\mu^{(t-1)}}}{2\sqrt{\mu^{(t-1)}+1}} - \frac{\sqrt{\mu^{(t-1)}+1}}{2\sqrt{\mu^{(t-1)}}}}{\mu^{(t-1)}\frac{-1}{(\mu^{(t-1)})^2}} =$$

$$\lim_{t\to+\infty} \frac{\epsilon}{r_i^{(t)}} \frac{\frac{-1}{2\sqrt{\mu^{(t-1)}}\sqrt{\mu^{(t-1)}+1}}}{\mu^{(t-1)}\frac{-1}{(\mu^{(t-1)})^2}} = \frac{1}{2},$$

where to derive the last equation we also took into account that $\lim_{t\to+\infty}\epsilon/r_i^{(t)} = 1$ (since the domain of $\epsilon\sqrt{\mu^{(t-1)}(\mu^{(t-1)}+1)}/r_i^{(t)} - \mu^{(t-1)}$, with respect to $r_i^{(t)}$, becomes the set $\{\epsilon\}$ for $t \to +\infty$).

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Alternative Justification for TLS

Proposition 51 (Weibull Distribution Leads to TLS)**.** *Assume* $r(\boldsymbol{y}_i, \boldsymbol{x}^\circ) \leq \epsilon$ *for any* $i \in \mathcal{M} \backslash \mathcal{O}^\circ$*. If* $r(\boldsymbol{y}_i, \boldsymbol{x}^\circ)$ *is a Weibull random variable for each* $i \in \mathcal{M}$*, with* cumulative probability distribution $\text{Weib}(r) \triangleq 1 - \exp\left(-r^2/2\right)$*, then* TLS *is equivalent to the maximum likelihood estimator*

$$\max_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \mathcal{O} \subseteq \mathcal{M}}} \prod_{i \in \mathcal{M} \backslash \mathcal{O}} \left[1 - \text{Weib}(r(\boldsymbol{y}_i, \boldsymbol{x}))\right] \prod_{i \in \mathcal{O}} \left[1 - \text{Weib}(\epsilon)\right]. \tag{B.1}$$

Broadly speaking, the Weibull distribution is commonly used in statistics to model the probability of an outcome's *failure* when the failure depends on sub-constituent failures: *e.g.,* a chain breaks if any of its rings breaks [254]. Similarly, an outlier-robust estimate "breaks" if measurements are misclassified as inliers instead of outliers and vice versa, and if the inliers' residuals are unnecessarily large:

- if a measurement i is classified as an outlier ($i \in \mathcal{O}$), then $1 - \text{Weib}(\epsilon)$ models the probability of a *successful* estimation given that i's residual is *at least* ϵ;
- if a measurement i is classified as an inlier ($i \in \mathcal{M} \setminus \mathcal{O}$), then $1 - \text{Weib}(r(\boldsymbol{y}_i, \boldsymbol{x}))$ models the probability of a *successful* estimation given that i's residual is at least $r(\boldsymbol{y}_i, \boldsymbol{x})$ *but not more than* ϵ: indeed, if $r(\boldsymbol{y}_i, \boldsymbol{x}) > \epsilon$, then Eq. (B.1) classifies i as an outlier, so to maximize the joint probability likelihood, since $1 - \text{Weib}(r(\boldsymbol{y}_i, \boldsymbol{x})) < 1 - \text{Weib}(\epsilon)$. Therefore, for all $i \in \mathcal{M} \setminus \mathcal{O}$, $r(\boldsymbol{y}_i, \boldsymbol{x}) \leq \epsilon$.

In summary, Eq. (B.1) aims to find $(\boldsymbol{x}, \mathcal{O})$ that maximize the probability of the estimator's success, and, particularly, it does so by forcing the inliers' $r(\boldsymbol{y}_i, \boldsymbol{x})$ to be as small as possible, since indeed $1 - \text{Weib}(r(\boldsymbol{y}_i, \boldsymbol{x})) \rightarrow 1$ when $r(\boldsymbol{y}_i, \boldsymbol{x}) \rightarrow 0$.

Proof of Proposition 51

The proof is derived by taking the $-\log(\cdot)$ of the objective function in Eq. (B.1), resulting in the TLS cost in Eq. (2.2).

Appendix C

Bound for Pose Graph Optimization

Let us consider the following SLAM problem (Pose Graph Optimization):

$$\min_{\boldsymbol{T}_i \in \mathrm{SE}(d)} \sum_{(i,j)\in\mathcal{E}_o} \|\boldsymbol{T}_j - \boldsymbol{T}_i \bar{\boldsymbol{T}}_{ij}\|_\Omega^2 + \sum_{(i_k,j_k)\in\mathcal{E}_{lc}} \|\boldsymbol{T}_{j_k} - \boldsymbol{T}_{i_k} \bar{\boldsymbol{T}}_k\|_\Omega^2 \tag{C.1}$$

where $\mathcal{E}_o$ are the (reliable) odometry edges, while $\mathcal{E}_{lc}$ are the (possibly unreliable) loop closures. Note that we indexed the loop closures using $k = 1, \ldots, |\mathcal{E}_{lc}|$, such that the k-th loop closure connects poses (i_k, j_k).

Let us define the set function:

$$f(\mathcal{S}) = \min_{\boldsymbol{T}_i \in \mathrm{SE}(d)} \sum_{(i,j)\in\mathcal{E}_o} \|\boldsymbol{T}_j - \boldsymbol{T}_i \bar{\boldsymbol{T}}_{ij}\|_\Omega^2 + \sum_{(i_k,j_k)\in\mathcal{E}_{lc}\setminus\mathcal{S}} \|\boldsymbol{T}_{j_k} - \boldsymbol{T}_{i_k} \bar{\boldsymbol{T}}_k\|_\Omega^2 \tag{C.2}$$

and it's normalized version:

$$\bar{f}(\mathcal{S}) = f(\emptyset) - f(\mathcal{S}) \tag{C.3}$$

which is also positive ($\bar{f}(\mathcal{S}) \geq 0, \forall \mathcal{S} \subseteq \mathcal{E}_{lc}$) and non-decreasing ($\bar{f}(\mathcal{S}_1) \leq \bar{f}(\mathcal{S}_2), \forall \mathcal{S}_1 \subseteq \mathcal{S}_2 \subseteq \mathcal{E}_{lc}$, since $f(\mathcal{S}_1) \geq f(\mathcal{S}_2), \forall \mathcal{S}_1 \subseteq \mathcal{S}_2 \subseteq \mathcal{E}_{lc}$). Eventually, we would like to solve:

$$\max_{|\mathcal{S}|=\beta} \bar{f}(\mathcal{S}) \tag{C.4}$$

which looks for the set $\mathcal{S}$ that makes $f(\mathcal{S})$ as small as possible.

Finally, define the marginal gain:

$$\bar{f}(\{s\}|\mathcal{S}) \triangleq \bar{f}(\mathcal{S} \cup \{s\}) - \bar{f}(\mathcal{S}) = f(\mathcal{S}) - f(\mathcal{S} \cup \{s\}) \tag{C.5}$$

Our goal now is to compute a good and computationally-inexpensive upper-bound for $\bar{f}(\{s\}|\mathcal{S})$, which can be used in the Lazy greedy algorithm. In particular:

- we want to avoid computing $f(\mathcal{S} \cup \{s\})$ at each iteration of the greedy.

- we can leverage the knowledge of $f(\mathcal{S})$ which has been computed at the previous iteration.

We also remark that computing an upper-bound for $\bar{f}(\{s\}|\mathcal{S})$ for a fixed $f(\mathcal{S})$ is the same as computing a lower-bound for $f(\mathcal{S} \cup \{s\})$.

Therefore in the following we compute a lower bound for $f(\mathcal{S} \cup \{s\})$. For this purpose, we note that the original cost function can be written as:

$$\min_{\boldsymbol{T}_i \in \mathrm{SE}(d)} \sum_{(i_k, j_k) \in \mathcal{E}_{lc}} \left(\|\boldsymbol{T}_{j_k} - \boldsymbol{T}_{i_k} \bar{\boldsymbol{T}}_k\|_\Omega^2 + \sum_{(i,j) \in \mathcal{E}_o^k} \frac{1}{n_{ij}(\mathcal{E}_{lc})} \|\boldsymbol{T}_j - \boldsymbol{T}_i \bar{\boldsymbol{T}}_{ij}\|_\Omega^2 \right) \tag{C.6}$$

where $n_{ij}(\mathcal{E}_{lc})$ is the number of loops the odometry edge is involved in, within the graph with loop closures $\mathcal{E}_{lc}$. Similarly:

$$f(\mathcal{S}) = \min_{\boldsymbol{T}_i \in \mathrm{SE}(d)} \sum_{(i_k,j_k)\in\mathcal{S}} \left(\sum_{(i,j)\in\mathcal{E}_o^k} \frac{1}{n_{ij}(\mathcal{E}_{lc})} \|\boldsymbol{T}_j - \boldsymbol{T}_i \bar{\boldsymbol{T}}_{ij}\|_\Omega^2 \right) \tag{C.7}$$

$$+ \sum_{(i_k,j_k)\in\mathcal{E}_{lc}\setminus\mathcal{S}} \left(\|\boldsymbol{T}_{j_k} - \boldsymbol{T}_{i_k}\bar{\boldsymbol{T}}_k\|_\Omega^2 + \sum_{(i,j)\in\mathcal{E}_o^k} \frac{1}{n_{ij}(\mathcal{E}_{lc})} \|\boldsymbol{T}_j - \boldsymbol{T}_i \bar{\boldsymbol{T}}_{ij}\|_\Omega^2 \right) \tag{C.8}$$

$$\geq \tag{C.9}$$

$$\sum_{(i_k,j_k)\in\mathcal{S}} \min_{\boldsymbol{T}_i \in \mathrm{SE}(d)} \left(\sum_{(i,j)\in\mathcal{E}_o^k} \frac{1}{n_{ij}(\mathcal{E}_{lc})} \|\boldsymbol{T}_j - \boldsymbol{T}_i \bar{\boldsymbol{T}}_{ij}\|_\Omega^2 \right) \tag{C.10}$$

$$+ \sum_{(i_k,j_k)\in\mathcal{E}_{lc}\setminus\mathcal{S}} \min_{\boldsymbol{T}_i \in \mathrm{SE}(d)} \left(\|\boldsymbol{T}_{j_k} - \boldsymbol{T}_{i_k}\bar{\boldsymbol{T}}_k\|_\Omega^2 + \sum_{(i,j)\in\mathcal{E}_o^k} \frac{1}{n_{ij}(\mathcal{E}_{lc})} \|\boldsymbol{T}_j - \boldsymbol{T}_i \bar{\boldsymbol{T}}_{ij}\|_\Omega^2 \right) \tag{C.11}$$

$$= \tag{C.12}$$

$$\sum_{(i_k,j_k)\in\mathcal{E}_{lc}\setminus\mathcal{S}} \min_{\boldsymbol{T}_i \in \mathrm{SE}(d)} \left(\|\boldsymbol{T}_{j_k} - \boldsymbol{T}_{i_k}\bar{\boldsymbol{T}}_k\|_\Omega^2 + \sum_{(i,j)\in\mathcal{E}_o^k} \frac{1}{n_{ij}(\mathcal{E}_{lc})} \|\boldsymbol{T}_j - \boldsymbol{T}_i \bar{\boldsymbol{T}}_{ij}\|_\Omega^2 \right) \tag{C.13}$$

Therefore we can compute the following (independent) quantities, for each $s \in \mathcal{E}_{lc}$:

$$b_k \doteq \min_{\boldsymbol{T}_i \in \mathrm{SE}(d)} \left(\|\boldsymbol{T}_{j_k} - \boldsymbol{T}_{i_k}\bar{\boldsymbol{T}}_k\|_\Omega^2 + \sum_{(i,j)\in\mathcal{E}_o^k} \frac{1}{n_{ij}(\mathcal{E}_{lc})} \|\boldsymbol{T}_j - \boldsymbol{T}_i \bar{\boldsymbol{T}}_{ij}\|_\Omega^2 \right) \tag{C.14}$$

And the desired lower bound for $f(\mathcal{S} \cup \{s\})$

$$f(\mathcal{S} \cup \{s\}) \geq \sum_{k \in \mathcal{S}\cup\{s\}} b_k \tag{C.15}$$

Note that

- b_k must be computed only once, at the beginning of the greedy
- each computation of b_k involves an optimization over a single cycle, which is faster than optimizing over the entire graph.

We also expect that outliers have large b_k since they do not agree with the odometry and produce large errors even along the cycle they create in the graph, while inliers have small b_k since they mostly agree with the odometry.

Appendix D

Routines for parameter-free algorithms

D.1 ClustersSeparation algorithm

ADAPT-MinT's subroutine ClustersSeparation is presented in Algorithm 6. Therein, for any real-vector $\boldsymbol{z} \in \mathbb{R}^l$ such that $z_i \geq 0$, and for all $i = 1, 2, \ldots, l$, $\operatorname{diam}(\boldsymbol{z}) \triangleq \sum_{i=1}^{l} |z_i - \operatorname{mean}(\boldsymbol{z})|^2$, and $\operatorname{mean}(\boldsymbol{z}) = \frac{1}{l}\sum_{i=1}^{l} z_i$; *i.e.,* diam captures the cumulative deviation of all z_i from their mean —their "centroid"— and, as such, can be interpreted as a diameter.

Algorithm 6: ClustersSeparation (ADAPT-MinT's subroutine).

```
Input: A real-valued vector $\boldsymbol{r} \in \mathbb{R}^l$.
Output: Centroids' distance that separates two clusters of entries in $\boldsymbol{r}$.
$\boldsymbol{z} = \operatorname{sort}(r_1, r_2, \ldots, r_l)$; // increasing order
$i = \arg\min_{j \in \{1,2,\ldots,l-1\}} \operatorname{diam}(\boldsymbol{z}_{1:j}) + \operatorname{diam}(\boldsymbol{z}_{j+1:\mathrm{end}})$;
$c_{\mathrm{left}} = \operatorname{mean}(\boldsymbol{z}_{1:i})$;   $c_{\mathrm{right}} = \operatorname{mean}(\boldsymbol{z}_{i+1:\mathrm{end}})$;
return $c_{right} - c_{left}$.
```

D.2 Chi2Fit **algorithm**

Chi2Fit is presented in Algorithm 7. Chi2Fit scores the fit of the empirical distribution of the residuals to the Gamma$(d/2, 2\sigma^2)$ distribution, which is equivalent to the desired χ^2 with degree of freedom d and variance σ^2. Since ϵ is unknown, the true variance of the residuals' error is also unknown. For this reason, in Chi2Fit's Line 1 an unbiased estimator for the variance is employed [255]. Then, Line 2 uses the Cramér–von Mises test to score the fit.

Algorithm 7: Chi2Fit (GNC-MinT's subroutine).

```
Input: Real-valued vector $\boldsymbol{r} \in \mathbb{R}^n$;
       $\chi^2$ distribution's degrees of freedom $d > 0$.
Output: Similarity statistic of $\chi^2$ distribution with empi- rical distribution
        of $\boldsymbol{r}$'s squared elements.
1 $\sigma^2 = \frac{1}{(n-1)d} \sum_{i=1}^{n} r_i^2$;
2 $s = \text{CramerVonMises}(\boldsymbol{r}, \text{Gamma}(\frac{d}{2}, 2\sigma^2))$;    // $\text{Gamma}(\frac{d}{2}, 2\sigma^2)$ = $\chi^2$ with
    degree of freedom $d$ and variance $\sigma^2$
3 return $s$.
```

D.3 Bisection **algorithm**

GNC-MinT's subroutine Bisection is presented in Algorithm Algorithm 8. Bisection aims to improve the inlier noise ϵ initial guess trying when the noise upper bound $NoiseUpBnd$ scores a bad χ^2 fit. The algorithm initialize the upper and lower bound for ϵ with resp., $NoiseUpBnd$ and $NoiseLowBnd$ and evaluates the fitness score after the outlier rejection (lines 1-2). Until the score it not close to the target value γ, it performs a binary search, stopping if (i) the fitness score does not improve across iterations (line 7), or (iii) the spread $(a-b)/a$ is too small (line 10). It then returns the next ϵ to test.

Algorithm 8: Bisection (GNC-MinT's subroutine).

```
   Input: Measurements $\boldsymbol{y}_i$, $\forall i \in \mathcal{M}$; $NoiseUpBnd \geq 0$;
          $NoiseLowBnd \geq 0$;Target critical value $\gamma$
   Output: Initial guess for $\epsilon$
 1 $a = c = NoiseUpBnd$; $b = NoiseLowBnd$;
 2 $(\boldsymbol{x}, \mathcal{I}) = \mathrm{GNC}(\boldsymbol{y}, c)$; $s = \mathrm{Chi2Fit}(\boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}}, \boldsymbol{x}), d)$;
 3 $e = s/\gamma$;
 4 while $e > 1.1$ or $e < 0.9$ do
 5     $c = (a+b)/2$;
 6     $(\boldsymbol{x}, \mathcal{I}) = \mathrm{GNC}(\boldsymbol{y}, c)$; $s = \mathrm{Chi2Fit}(\boldsymbol{r}(\boldsymbol{y}_{\mathcal{I}}, \boldsymbol{x}), d)$;
 7     if $|s/\gamma - e| < 1 \times 10^{-3}$ then break;
 8     else if $s/\gamma > 1.1$ then $a = c$ ;
 9     else if $s/\gamma > 0.9$ then $b = c$ ;
10     if $(a-b)/a < 0.1$ then break;
11     $e = s/\gamma$;
12 end
13 return $\max_{i \in \mathcal{I}} \{ r(\boldsymbol{y}_i, \boldsymbol{x}) \ s.t. \ r(\boldsymbol{y}_i, \boldsymbol{x}) < c \}$.
```

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix E

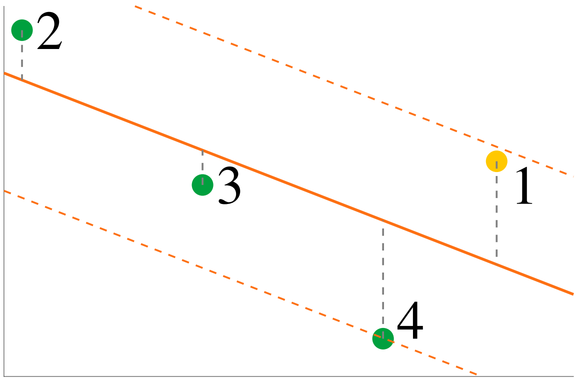
Limitations of ADAPT and ADAPT-MinT

We discuss failure modes of ADAPT and ADAPT-MinT. While ADAPT self-correction mechanism often improve the estimation sometimes it can harm the estimation, this can happen if ADAPT converged to the wrong estimate (thus the outliers residuals look like inliers) or if there are too many outliers and some of them fit the current estimation.

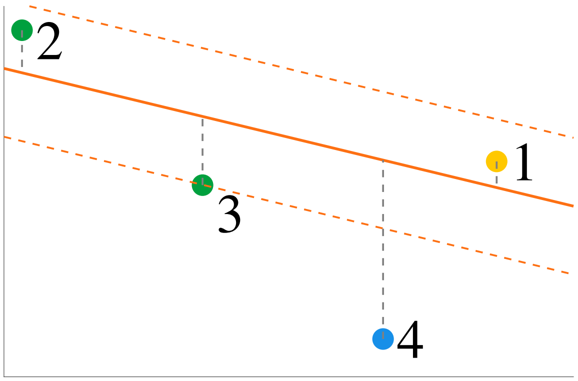
Inaccurate τ and θ. If τ and θ are set too low, lower than their true values, ADAPT will typically over-reject measurements. Conversely, if they are set too high, ADAPT is more likely to return sets containing outliers. Both scenarios can result to less accurate estimates.

Adversarial Outliers. ADAPT (and similarly GNC) can fail due to adversarial outliers. In Fig. E-1, we present such a scenario for a problem of linear regression, where there are three inliers (points 2-4) and one outlier (point 1). For appropriate *ThrDiscount*, ADAPT first rejects the inlier point 4, moving the new estimate (based on points 1-3) towards the outlier 1. Then, ADAPT rejects point 3, and, then, terminates. Conversely, if *ThrDiscount* $< 1/\alpha$, in one iteration ADAPT will reject both measurements (1 and 2), in the next iteration the remaining inliers (now in the set of estimated inliers) will move the estimate closer to 1 that will be re-included in the set of inliers through the self-correction mechanism.

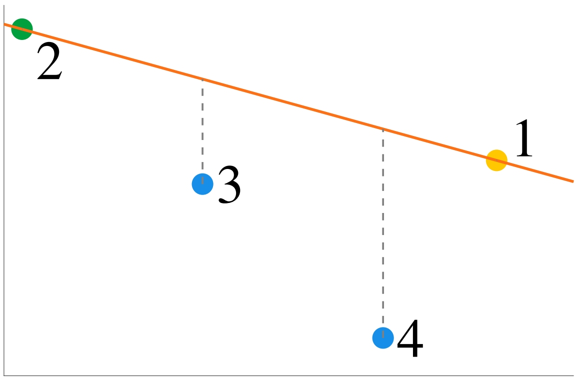
High Measurement Noise. High measurement noise can cause the cluster separation δ, used in ADAPT-MinT, to oscillate more than the chosen *ConvergThr*,



(a) Initilization



(b) Step 1



(c) Step 2

Figure E-1: A single outlier (point 1) leads ADAPT to the wrong solution.

thus making the algorithm to reject more measurements than the true number of outliers.

Appendix F

Limitations of GNC and GNC-MinT

We analyze failure modes of GNC and GNC-MinT.

Inaccurate ϵ. If ϵ is chosen lower than the real inlier threshold, then GNC can reject more measurements than the true number of outliers. Instead, if ϵ is too high, then GNC tends to reject less measurements, keeping outliers as inliers. Both scenarios can result to less accurate estimates.

Non-Gaussian Measurement Noise. If the residual's distribution is not close to a Gaussian, the χ^2 fitness score may not accurately indicate the presence of outliers. Thus, GNC-MinT may return less accurate estimates.

Arbitrarily Low *NoiseUpBnd*, and Arbitrarily Large *NoiseLowBnd*. If $\textit{NoiseLowBnd} - \textit{NoiseUpBnd}$ is unnecessarily large, then GNC-MinT, trying to find the true but unknown inlier threshold, will explore more ϵ values, and, as a result, it will run for longer time. Also, GNC-MinT stops as soon as the fitness score becomes worse (*cf.* GNC-MinT's lines 13-17). This point, however, may correspond to a local minima (thinking of the fitness score as a function of the inlier threshold guess). Therefore, if the *NoiseUpBnd* is unnecessarily high, there is a higher probability GNC-MinT stops prematurely.

Finally, GNC-MinT can under- or over-estimate the true inlier threshold: suppose at some iteration t, $\epsilon^{(t)} > \epsilon^{\circ}$, ϵ° being the true inlier threshold, and that the set $\mathcal{I}^{(t)}$ still contains few outliers but all the residuals are smaller than ϵ°, *i.e.,*

$\max\limits_{i \in \mathcal{I}^{(t)}} \{r(\boldsymbol{y}_i, \boldsymbol{x}^{(t)}) \text{ s.t. } r(\boldsymbol{y}_i, \boldsymbol{x}^{(t)}) < \epsilon^{(t)}\} < \epsilon^{\circ}$, then GNC-MinT will underestimate the inlier threshold ϵ.

Appendix G

Limitation of the greedy algorithm

A failure mode of the causal greedy is that it can never correct mistakes made in earlier steps. An example where this behavior leads to a wrong estimate (while ADAPT is able to correct the mistake) is shown in Fig. G-1. The example depict a line fitting problem where three points (1, 3 and 5) are inliers and two points (2 and 4) are outliers.

Let us first follow the execution of the greedy algorithm. The Greedy initialize its initial estimate $\boldsymbol{x}^{(0)}$ using all available measurements (iteration 0). In the first iteration, the Greedy removes point 1 (an inlier) permanently from the set of estimated inliers because it has the biggest residual, and computes the new estimate $\boldsymbol{x}^{(1)}$ using all remaining points $(2, \ldots, 5)$. In the second iteration, removes permanently point 5 (another inlier) from the set of estimated inliers and computes the new estimate $\boldsymbol{x}^{(2)}$. In the last iteration, the last ground truth inlier (point 3) has the biggest residual, therefore the greedy algorithm removes it from set of estimated inliers converging to the set of two outliers as the estimated set of inliers. Since the fit of the two points is below the inlier threshold, it stops. ADAPT on the other hand initialize its estimate using all available measurements and set the inlier threshold $\epsilon^{(0)} = \sqrt{0.99} \cdot r(\boldsymbol{y}_1, \boldsymbol{x}^{(0)})$ since 1 is the point with the biggest residual. In the first iteration, ADAPT removes all points above the current threshold, in our example it removes point 1 (an inlier) and 2 (an outlier) from the set of estimated inliers as their residuals is bigger than $\epsilon^{(0)}$. It than computes the new estimate $\boldsymbol{x}^{(1)}$ and decrease the inlier threshold to

$\epsilon^{(1)} = \sqrt{0.99} \cdot r(\boldsymbol{y}_4, \boldsymbol{x}^{(1)})$ (point 4 has the biggest residual). In the second iteration point 1 is within the current inlier threshold $\epsilon^{(1)}$ so it is included in the set of estimated inliers while the two outliers (point 2 and point 4) are above the threshold, in other words it recovered from a wrong choice made in past iterations, and the new estimate $\boldsymbol{x}^{(2)}$ is computed using the correct set of inliers thus estimating the correct line. ADAPT then decrease the inlier threshold $\epsilon^{(2)}$ to a value that is below the τ and stops.

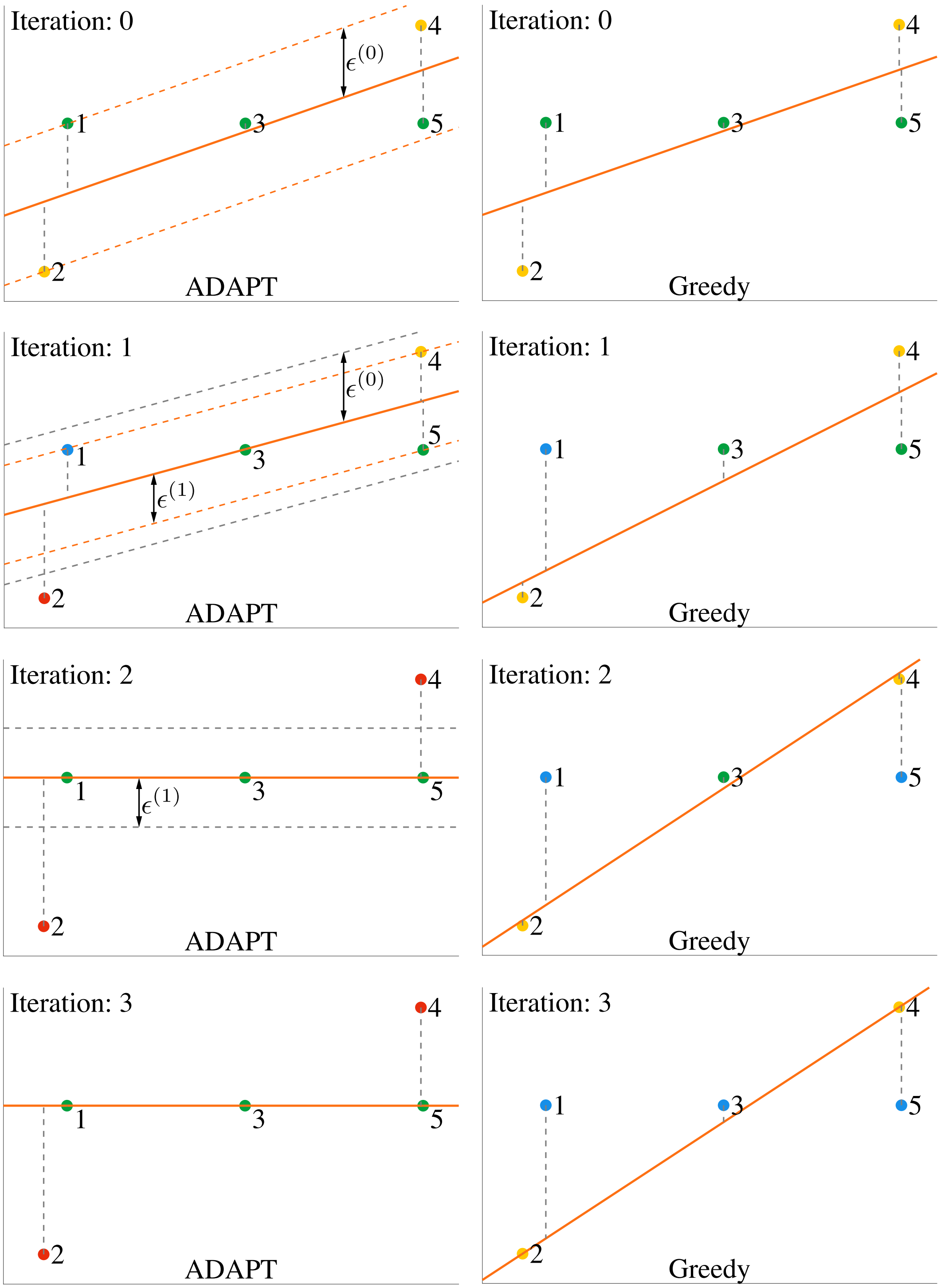
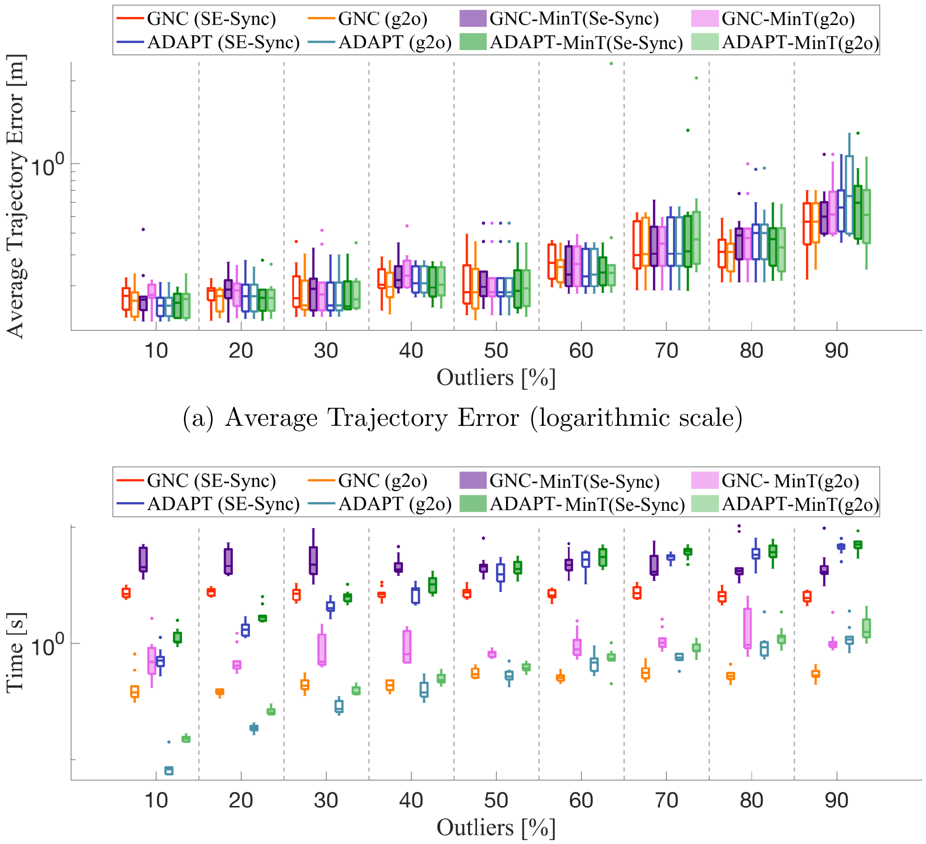


Figure G-1: An example of Greedy vs. ADAPT. The Greedy algorithm fails to recover the true solution while the ADAPT algorithm is able to recover from a wrong choice made in past iterations.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix H

Pose Graph: g2o vs. SE-Sync



(a) Average Trajectory Error (logarithmic scale)

(b) Running Time (logarithmic scale)

Figure H-1: Average Trajectory Error and Running time of the proposed algorithms on 2D SLAM (Grid) with two different non-minimal solvers: g2o and SE-Sync. The average performance is comparable while g2o offers a better running time.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix I

Proof of Theorem 49

Before proving Theorem 49, we introduce two useful results. Let's start by noticing that the risk function in Eq. (4.5) relies on the copula, and the CDFs of A and B, namely Φ_A and Φ_B. Therefore, to provide the performance bound on the risk function $\mathcal{R}$, we first need to bound the copula value. To this end, we can use the well-known Fréchet–Hoeffding copula bounds.

Theorem 52 (Fréchet–Hoeffding Theorem ([45], Theorem 2.2.3))**.** *For any copula $C : [0,1]^d \to [0,1]$ and any $(u_1, \dots, u_d) \in [0,1]^d$, the following bounds hold:*

$$W(u_1, \dots, u_d) \leq C(u_1, \dots, u_d) \leq M(u_1, \dots, u_d), \tag{I.1}$$

where

$$W(u_1, \dots, u_d) = \max \left\{ 1 - d + \sum_{i=1}^{d} u_i, 0 \right\},$$
$$M(u_1, \dots, u_d) = \min \left\{ u_1, \dots, u_d \right\}.$$

In this thesis, we are interested in the bi-dimensional copula, so we can apply Theorem 52 to a copula $C(p, v)$ and obtain:

$$\max \left\{ p + v - 1, 0 \right\} \leq C(p, v) \leq \min \left\{ p, v \right\}.$$

In particular, for the risk estimation, we take $v = \Phi_B \circ \Phi_A^{-1}(p)$. However, we do not have access to the explicit expression of the two CDFs Φ_A and Φ_B; therefore, we need to estimate them empirically. The following theorem provides estimation bounds on the empirical CDFs.

Theorem 53 (Dvoretzky–Kiefer–Wolfowitz Confidence Interval [256], [257])**.** *Let Φ be the CDF of an unknown distribution, and let $\Phi^{(n)}$ the empirical CDF computed using n i.i.d. samples from Φ, then, with probability at least $1-\alpha$,*

$$\Phi^{(n)}(x) - \epsilon(n,\alpha) \leq \Phi(x) \leq \Phi^{(n)}(x) + \epsilon(n,\alpha), \tag{I.2}$$

where $\epsilon(n,\alpha) = \sqrt{\ln(2/\alpha)/(2n)}$.

We use Theorem 53 to estlablish bounds on $\Phi_B \circ \Phi_A^{-1}(p)$ in the next lemma.

Lemma 54. *Let A, B be two random variables with CDFs Φ_A and Φ_B, respectively. Let $\Phi_A^{(n)}$ and $\Phi_B^{(n)}$ be the empirical CDFs estimated using n i.i.d. samples from Φ_A and Φ_B, respectively. Then, with probability at least $1-\alpha$ we have:*

$$\underline{v}(p,\alpha,n) \leq \Phi_B \circ \Phi_A^{-1}(p) \leq \bar{v}(p,\alpha,n),$$

where

$$\begin{aligned}\bar{v}(p,\alpha,n) &= \Phi_B^{(n)} \circ \left[\Phi_A^{(n)} - \epsilon(\alpha,n)\right]^{-1}(p) + \epsilon(\alpha,n)\\ \underline{v}(p,\alpha,n) &= \Phi_B^{(n)} \circ \left[\Phi_A^{(n)} + \epsilon(\alpha,n)\right]^{-1}(p) - \epsilon(\alpha,n)\end{aligned}$$

and $\epsilon(\alpha,n) = \sqrt{\ln(2/\alpha)/(2n)}$.

Proof. We are interested in the quantity $\Phi_B \circ \Phi_A^{-1}(p)$. Notice that $\Phi_A^{(n)}$ and $\Phi_B^{(n)}$ are increasing functions. From Theorem 53 we know that $\underline{x} \leq \Phi_A^{-1}(p) \leq \bar{x}$ (see Fig. I-1), where

$$\begin{aligned}\bar{x}(p,\alpha,n) &= \left[\Phi_A^{(n)} - \epsilon(\alpha,n)\right]^{-1}(p)\\ \underline{x}(p,\alpha,n) &= \left[\Phi_A^{(n)} + \epsilon(\alpha,n)\right]^{-1}(p)\end{aligned} \tag{I.3}$$

Similarly, we have $\underline{v} \leq \Phi_B \circ \Phi_A^{-1}(p) \leq \bar{v}$ where

$$\begin{aligned}\bar{v}(p,\alpha,n) &= \Phi_B^{(n)} \circ \bar{x}(p,\alpha,n) + \epsilon(\alpha,n)\\ \underline{v}(p,\alpha,n) &= \Phi_B^{(n)} \circ \underline{x}(p,\alpha,n) - \epsilon(\alpha,n)\end{aligned} \tag{I.4}$$

Substituting Eq. (I.3) into Eq. (I.4) we complete the proof □

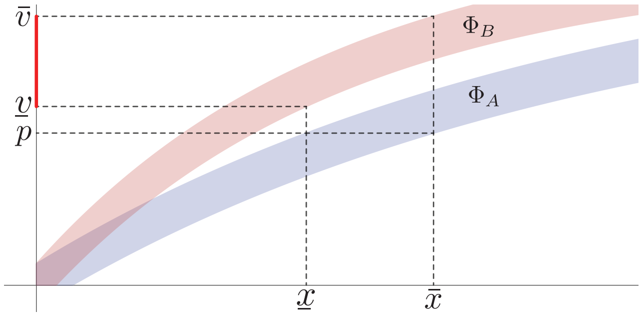


Figure I-1: CDF composition bounds. The shaded regions represent the the confidence intervals for the two CDFs, *i.e.*, $\Phi^{(n)}(x) - \epsilon(n, \alpha) \leq \Phi(x) \leq \Phi^{(n)}(x) + \epsilon(n, \alpha)$. The blue and orange regions represent Φ_A and Φ_B respectively.

We are now ready to prove Theorem 49.

Proof of Theorem 49. From Theorem 52 we know that

$$\max\{p + v - 1, 0\} \leq C(p, v) \leq \min\{p, v\},$$

where $v = \Phi_B \circ \Phi_A^{-1}(p)$. We can use Lemma 54 to bound v, obtaining

$$\max(p + \underline{v} - 1, 0) \leq C(p, v) \leq \min(p, \bar{v}). \tag{I.5}$$

Since $\mathcal{R}(p) = 1 - {}^{C(p, \Phi_B \circ \Phi_A^{-1}(p))}/_{p}$, we have $C(p, \Phi_B \circ \Phi_A^{-1}(p)) = p\,(1 - \mathcal{R}(p))$. Substituting this into Eq. (I.5) completes the proof. □

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix J

Risk Estimation Cost Functions

Let $\boldsymbol{x}_e$, $\boldsymbol{v}_e$ be the position and velocity of the ego vehicle. Similarly, let $\boldsymbol{x}_a$, $\boldsymbol{v}_a$ be the position and velocity of any non-ego agent. Moreover, let ν be a term used to penalize the violation of traffic rules, such as driving on the wrong side of the road, or driving in the opposite direction of the traffic, or crossing an intersection with a red traffic light.

The time-to-collision cost is defined as:

$$c_{\text{TTC}} = 1 - \max_{a \in \text{Agents}} \min \left\{ \frac{TTC(x_e, x_a, v_e, v_a)}{m}, 1 \right\} + \nu,$$

where m represent a maximum value for the TTC function, which outputs the time until a collision between the ego vehicle and another agent occurs (assuming constant velocity at their current heading direction), or is infinite if no collision occurs. In our experiments we use $m = 3$. This cost function takes values in $[0, 1]$, and higher values indicate smaller Time-To-Collisions, thus higher risk.

The Momentum Shape Distance is instead defined as:

$$\begin{aligned} c_{\text{MSD}} &= \max_{a \in \text{Agents}} e^{\epsilon \delta / 2} + \nu, \\ \delta &= \big((\boldsymbol{x}_{a,\|} - \boldsymbol{x}_{e,\|})(\boldsymbol{v}_{a,\|} - \boldsymbol{v}_{e,\|}) \big)^2 + \left((\boldsymbol{x}_{a,\perp} - \boldsymbol{x}_{e,\perp})(\boldsymbol{v}_{a,\perp} - \boldsymbol{v}_{e,\perp}) \right)^2 \end{aligned}$$

were we used the subscript $\|$ and $\perp$ to denote the projection of a vector along the parallel and perpendicular direction to the ego vehicle's heading, respectively. The scaling factor ϵ weighs the importance of an agent: in our experiments we set $\epsilon = 0.5$ when the agent is a vehicle, and $\epsilon = 1$ when the agent is a pedestrian.

Appendix K

Proof of Lemma 37

We prove "κ-diagnosability $\Rightarrow$ syndrome$(\mathcal{A}_1) \cap$ syndrome$(\mathcal{A}_2) = \emptyset$" and its reverse implication below. In both, we define $\mathcal{X} = \{\mathcal{A} \subseteq \{1, \ldots, N_f\} \mid |\mathcal{A}| \leq \kappa\}$ to be the set of subsets of $\{1, \ldots, N_f\}$ of cardinality no larger than κ.

$\Rightarrow$ Suppose $\mathcal{D}$ is κ-diagnosable. Suppose by contradiction that there exists a syndrome $\boldsymbol{z}$ such that $\boldsymbol{z} \in$ syndrome$(\mathcal{A}_1) \cap$syndrome$(\mathcal{A}_2)$, with $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{X}$ and $\mathcal{A}_1 \neq \mathcal{A}_2$. Since $\boldsymbol{z} \in$ syndrome$(\mathcal{A}_1)$ and $\boldsymbol{z} \in$ syndrome$(\mathcal{A}_2)$, we are unable to say if the syndrome $\boldsymbol{z}$ is produced by the set of active failure modes is $\mathcal{A}_1$ or $\mathcal{A}_2$, contradicting the definition of κ-diagnosability of $\mathcal{D}$.

$\Leftarrow$ Call $\mathcal{Y} = \bigcup_{\mathcal{A} \in \mathcal{X}}$ syndrome$(\mathcal{A})$ the set of all possible syndromes assuming there are less than κ active failure modes. From the assumptions we know that any two $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{X}$ have syndrome$(\mathcal{A}_1) \cap$ syndrome$(\mathcal{A}_2) = \emptyset$, which means that we can uniquely map a syndrome to any set $\mathcal{A}$. This is exactly the definition of κ-diagnosability.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix L

Proof of Theorem 38

The assumption on the cardinality allows us to transform our general diagnostic graph into an undirected graph akin to the one used in [146], [150]. Then, the conditions *(i)*, *(ii)* and *(iii)* are a straightforward application of Theorem 2 in [146] to the resulting graph.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix M

Proof of Theorem 39

Let $\boldsymbol{z}$ be a syndrome for the temporal diagnostic graph $\mathcal{D}^{[K]}$, generated by a set of active failure mode $\mathcal{A}$, such that $|\mathcal{A}|= m \leq \min_{i\in\{1,\ldots,K\}} \kappa(\mathcal{D}^{(i)})$. Clearly, each element of $\mathcal{A}$ is a variable node of one of the regular diagnostic graphs $\mathcal{D}^{(1)},\ldots,\mathcal{D}^{(K)}$ that compose $\mathcal{D}^{[K]}$, therefore we can split $\mathcal{A}$ into the variables nodes of each regular diagnostic graph, obtaining $\mathcal{A}^{(1)},\ldots,\mathcal{A}^{(K)}$ (these sets are non-overlapping and are such that $\cup_{i=1}^{K}\mathcal{A}^{(i)} = \mathcal{A}$). Similarly, we can project the syndrome $\boldsymbol{z}$ into K sub-syndromes $\boldsymbol{z}^{(1)},\ldots,\boldsymbol{z}^{(K)}$ each containing only the test outcomes of the corresponding regular diagnostic graphs (notice that doing the projection we lose the temporal tests, if any). By construction $|\mathcal{A}^{(i)}| \leq m$ for each $i = 1,\ldots,K$. From the assumption, we know that each sub-graph $\mathcal{D}^{(i)}$ is m-diagnosable. Therefore, each sub-graph $\mathcal{D}^{(i)}$ will be able to correctly identify the set of active failure modes $\mathcal{A}^{(i)}$ from the syndrome $\boldsymbol{z}^{(i)}$. This means that $\mathcal{D}^{[K]}$ is at least m-diagnosable, concluding the proof.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix N

Proof of Theorem 42

For each sample $(\boldsymbol{z}^{(i)}, \boldsymbol{f}^{(i)})$ in $\mathcal{W}$, the result of each Hamming distance will less or equal than N_f. From the Hoeffding's inequality we have that

$$\Pr\left[|h_{\mathrm{dist}F}(\Psi_{\mathcal{D}}) - \hat{h}_{\mathcal{W}}(\Psi_{\mathcal{D}})|\geq \epsilon\right] \leq 2\mathrm{exp}\left(-\frac{2\epsilon^2|\mathcal{W}|}{{N_f}^2}\right) \tag{N.1}$$

Setting the right-hand side of Eq. (N.1) to be equal to δ and solving for ϵ yields:

$$\epsilon = N_f\sqrt{\frac{\log(2/\delta)}{2|\mathcal{W}|}} \tag{N.2}$$

After setting the right-hand side to δ, Eq. (N.1) can be rewritten as:

$$\Pr\left[|h_{\mathrm{dist}F}(\Psi_{\mathcal{D}}) - \hat{h}_{\mathcal{W}}(\Psi_{\mathcal{D}})|\leq \epsilon\right] \geq \delta \tag{N.3}$$

Combining Eq. (N.2) and Eq. (N.3) and removing the absolute value we get:

$$\Pr\left[h_{\mathrm{dist}F}(\Psi_{\mathcal{D}}) - \hat{h}_{\mathcal{W}}(\Psi_{\mathcal{D}}) \leq N_f\sqrt{\frac{\log(2/\delta)}{2|\mathcal{W}|}}\right] \geq \delta \tag{N.4}$$

from which the result easily follows.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix O

Proof of Corollary 43

Let $\gamma = h_{\mathrm{dist}F}(\Psi_{\mathcal{D}})$ and $p = 1 - \delta$, substituting into Eq. (3.23), and solving for p yield the result.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix P

Using Fault Detection to Prevent Accidents

Here we show how fault detection and identification can be effectively used to prevent dangerous situations. To this aim, we developed an additional scenario (not included in Table 3.2) where a deer crosses the road while the ego vehicle cruises on a straight road (Fig. P-1).

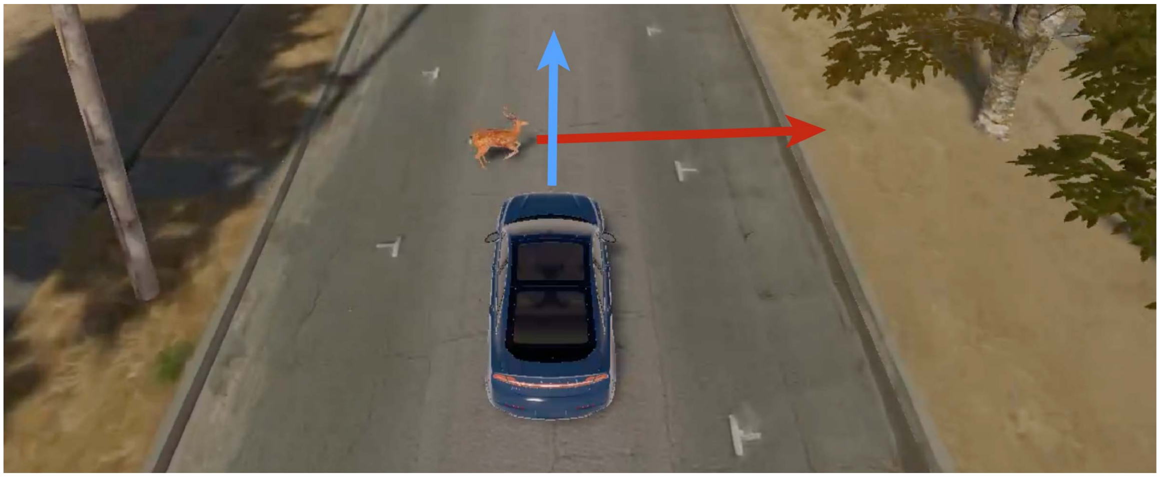


Figure P-1: Example scenario involving a deer crossing the road in front of the ego vehicle.

The scenario is novel to the identification algorithm, *i.e.,* not used for training, test, or validation. The results of the failure identification are shown in Fig. P-2, where we used the probabilistic fault identification. Initially, the monitor detects no

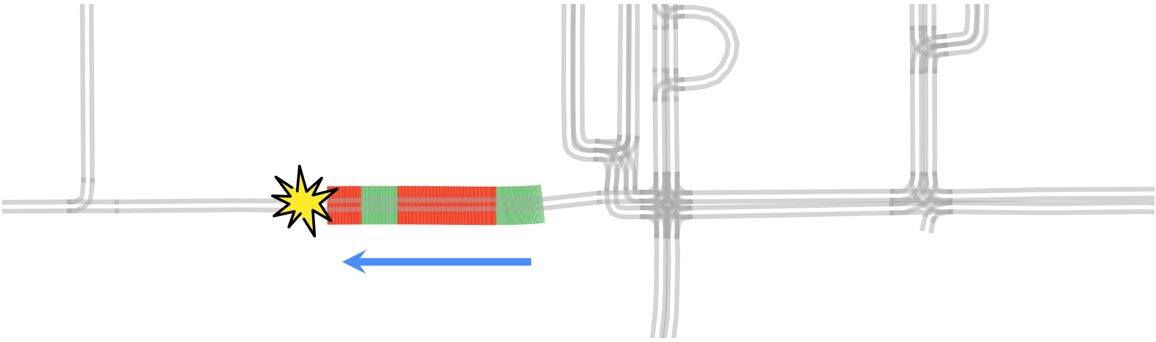


Figure P-2: Fault identification results for the example scenario in Fig. P-1. The car travels from right to left. Initially, the monitor detects no failure (rightmost, green section). As the ego vehicle gets closer to the obstacle, the LiDAR-based and camera-based obstacle detectors fail to detect the deer while the radar-based obstacle detector correctly locates the obstacle; as a result the fault identification/detection triggers an alarm (red sections).

failure (rightmost green section). As the ego vehicle gets closer to the undetected obstacle, the radar detects the obstacle but the camera does not. The inconsistency between the two sets of obstacles causes the test between camera and radar to return FAIL. Given the test's outcomes, the factor graph correctly detects and identifies the failure, triggering an alarm (rightmost red section). As the ego vehicle gets even closer, the deer goes out of the field-of-view of the radar while entering the LiDAR field-of-view. For a few meters, both camera and LiDAR fail to detect the deer Fig. P-3, but since it is out of the field-of-view of the radar, the diagnostic test fails to report the failure[1]. As the obstacle re-enters the field-of-view of the radar, the diagnostic test again returns FAIL, signaling the presence of a failure.

The first alarm is raised 7.19 s before the collision, flagging the camera misdetection as an active failure mode. Before the collision, the AV has a speed of 8.43 m/s. The car can reach a maximum deceleration of $6\,\mathrm{m\,s^{-2}}$. As result, the car would need 1.4 s to come to a complete stop. We note that after detecting the fault, for a short interval of time the monitor detects no failure: this is due to the fact that the deer goes out of the radar field-of-view, and no other obstacle detector is capable of detecting it, thus lacking redundancy to diagnose the failure; see the visualization and

[1] This could be solved by improving the logic of the diagnostic test; for instance, it could predict that —while the obstacle moved outside the field-of-view— it is unlikely it disappeared.

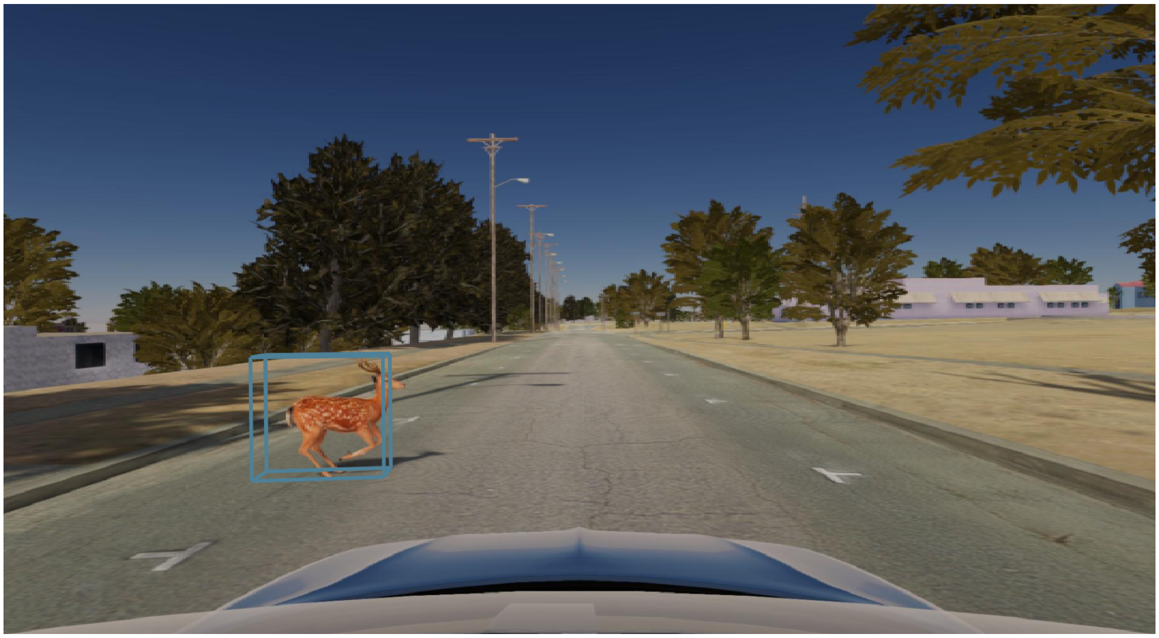


Figure P-3: Camera Image for the scenario in Fig. P-1. Blue bounding box is the ground truth detection. The camera fails to detect the deer crossing the road (misdetection failure).

explanation in Fig. P-4.

To gather statistical evidence of the effectiveness of the fault detection, we run the same scenario 10 times at different times of the day (sun, twilight, and night) and different weather conditions (including fog and rain). The probabilistic fault detection approach never raised false alarms in these tests, and the average time between the alarm and the collision was 7.54 s. The car traveled at an average speed of 6.16 m/s, requiring 1.03 s to come to a complete stop. The fault identification exhibited an average accuracy of 93.75 %.

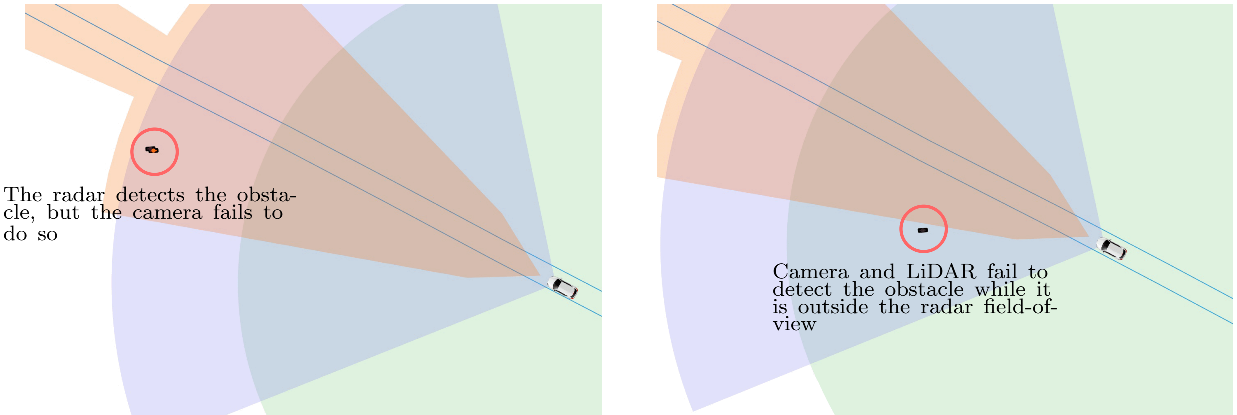

The radar detects the obstacle, but the camera fails to do so
Camera and LiDAR fail to detect the obstacle while it is outside the radar field-of-view

Figure P-4: Two snapshots from the example scenario of Fig. P-1. Shaded areas represent the sensor field-of-view (FOV): green, blue, and orange represent the LiDAR, camera, and radar FOVs, respectively. On the left, the deer is outside the LiDAR FOV (so the LiDAR obstacle detector is not supposed to detect the obstacle); the radar detects the obstacle, while the camera fails to detect it even if it is inside its FOV. Since the corresponding diagnostic test fails, our monitors can detect the failure. On the right, the deer is outside the radar FOV; in this case, both the camera and the LiDAR fail to detect the obstacles (even though it is within their FOVs), hence no diagnostic test fails and our monitor fails to detect the fault.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016, arxiv preprint: 1606.05830, ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2624754.

[2] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and Certifiable Point Cloud Registration," *IEEE Trans. Robotics*, vol. 37, no. 2, pp. 314–333, 2020.

[3] H. Yang and L. Carlone, "In perfect shape: Certifiably optimal 3D shape reconstruction from 2D landmarks," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Arxiv version: 1911.11924, 2020.

[4] S. Choi, Q. Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *cvpr*, 2015, pp. 5556–5565.

[5] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *icra*, IEEE, 2015, pp. 2174–2181.

[6] H. Maron, N. Dym, I. Kezurer, S. Kovalsky, and Y. Lipman, "Point registration via efficient convex relaxation," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–12, 2016.

[7] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, "Functional maps: A flexible representation of maps between shapes," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–11, 2012.

[8] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, IEEE, 2007, pp. 225–234.

[9] M. A. Audette, F. P. Ferrie, and T. M. Peters, "An algorithmic overview of surface registration techniques for medical imaging," *Med. Image Anal.*, vol. 4, no. 3, pp. 201–217, 2000.

[10] D. Rosen, L. Carlone, A. Bandeira, and J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the Special Euclidean group," *Intl. J. of Robotics Research*, 2018.

[11] D. Lowe, "Object recognition from local scale-invariant features," in *Intl. Conf. on Computer Vision (ICCV)*, 1999, pp. 1150–1157.

[12] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3d point cloud matching with smoothed densities," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5545–5554.

[13] T.-J. Chin, Z. Cai, and F. Neumann, "Robust fitting in computer vision: Easy or hard?" In *European Conf. on Computer Vision (ECCV)*, 2018.

[14] T. J. Chin and D. Suter, "The maximum consensus problem: Recent algorithmic advances," *Synthesis Lectures on Computer Vision*, vol. 7, no. 2, pp. 1–194, 2017.

[15] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, 1981.

[16] Á. P. Bustos and T. J. Chin, "Guaranteed outlier removal for point cloud registration with correspondences," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 40, no. 12, pp. 2868–2882, 2018.

[17] P. Huber, *Robust Statistics.* John Wiley & Sons, New York, NY, 1981.

[18] M. J. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *Intl. J. of Computer Vision*, vol. 19, no. 1, pp. 57–91, 1996.

[19] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[20] N. Sunderhauf and P. Protzel, "Towards a robust back-end for pose graph SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 1254–1261.

[21] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013.

[22] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *ArXiv*, vol. abs/1708.06374, 2017.

[23] EASA and Daedalean, *Concepts of Design Assurance for Neural Networks*, 2020.

[24] B. Chen, J. Cao, A. Parra, and T.-J. Chin, "Satellite pose estimation with deep landmark regression and nonlinear pose refinement," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, IEEE, 2019, pp. 2816–2824.

[25] G. Silberg, R. Wallace, G. Matuszak, J. Plessers, C. Brower, and D. Subramanian, "Self-driving cars: The next revolution," *White paper, KPMG LLP & Center of Automotive Research*, vol. 9, no. 2, pp. 132–146, 2012.

[26] American Automobile Association, *Active driving assistance system performance*, `https://newsroom.aaa.com/asset/active-driving-assistance-system-performance-may-2022/`, 2022.

[27] *Waymo and Cruise self-driving cars took over San Francisco streets at record levels in 2021*, `https://www.businessinsider.com/self-driving-car-accidents-waymo-cruise-tesla-zoox-san-francisco-2022-1`, Accessed: 2022-05-14.

[28] R. Salay, R. Queiroz, and K. Czarnecki, "An analysis of iso 26262: Using machine learning safely in automotive software," *arXiv preprint arXiv:1709.02435*, 2017.

[29] ISO Standard, *Road vehicles — safety of the intended functionality*, ISO/PAS 21448:2019(en), 2019.

[30] Aptiv, Audi, B. Apollo, *et al.*, *Safety First for Automated Driving*, 2019. [Online]. Available: `https://www.daimler.com/innovation/case/autonomous/safety-first-for-automated-driving-2.html`.

[31] H. Jing, Y. Gao, S. Shahbeigi, and M. Dianati, "Integrity monitoring of gnss/ins based positioning systems for autonomous vehicles: State-of-the-art and open challenges," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[32] O. A. Hafez, G. D. Arana, M. Joerger, and M. Spenko, "Quantifying robot localization safety: A new integrity monitoring method for fixed-lag smoothing," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3182–3189, 2020.

[33] V. Besnier, A. Bursuc, D. Picard, and A. Briot, "Triggering failures: Out-of-distribution detection by learning from local adversarial attacks in semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 701–15 710.

[34] D. Miller, P. Moghadam, M. Cox, M. Wildie, and R. Jurdak, "What's in the black box? the false negative mechanisms inside object detectors," *arXiv preprint arXiv:2203.07662*, 2022.

[35] P. Antonante, D. I. Spivak, and L. Carlone, "Monitoring and diagnosability of perception systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 168–175. DOI: `10.1109/IROS51168.2021.9636497`.

[36] M. S. Ramanagopal, C. Anderson, R. Vasudevan, and M. Johnson-Roberson, "Failing to learn: Autonomously identifying perception failures for self-driving cars," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3860–3867, 2018.

[37] D. Bogdoll, M. Nitsche, and J. M. Zöllner, "Anomaly detection in autonomous driving: A survey," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4488–4499.

[38] F. Christianos, P. Karkus, B. Ivanovic, S. V. Albrecht, and M. Pavone, "Planning with occluded traffic agents using bi-level variational occlusion models," *arXiv preprint arXiv:2210.14584*, 2022.

[39] M. Itkina, Y.-J. Mun, K. Driggs-Campbell, and M. J. Kochenderfer, "Multi-agent variational occlusion inference using people as sensors," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4585–4591.

[40] P. Antonante, H. Nilsen, and L. Carlone, "Monitoring of perception systems: Deterministic, probabilistic, and learning-based fault detection and identification," *arXiv preprint: 2205.10906*, 2022.

[41] V. Tzoumas, P. Antonante, and L. Carlone, "Outlier-robust spatial perception: Hardness, general-purpose algorithms, and guarantees," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Extended arxiv version: 1903.11683, 2019.

[42] S. Arora and B. Barak, *Computational complexity: A modern approach.* Cambridge University Press, 2009.

[43] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone, "Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 1127–1134, 2020, arXiv preprint:1909.08605 (with supplemental material), , ICRA Best paper award in Robot Vision.

[44] F. P. Preparata, G. Metze, and R. T. Chien, "On the connection assignment problem of diagnosable systems," *IEEE Transactions on Electronic Computers*, no. 6, pp. 848–854, 1967.

[45] R. B. Nelsen, *An introduction to copulas.* Springer Science & Business Media, 2007.

[46] K. T. e. a. H. Caesar J. Kabzan, "Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," in *Proceedings of CVPR ADP3 workshop*, 2021.

[47] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection.* John Wiley & Sons, New York, NY, 1987.

[48] P. Lajoie, S. Hu, G. Beltrame, and L. Carlone, "Modeling perceptual aliasing in SLAM via discrete-continuous graphical models," *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[49] H. Yang and L. Carlone, "A quaternion-based certifiably optimal solution to the Wahba problem with outliers," in *Intl. Conf. on Computer Vision (ICCV)*, (Oral Presentation, accept rate: 4%), Arxiv version: 1905.12536, 2019.

[50] National Institute of Standards and Technology (NIST), *Table of the standard normal distribution.* [Online]. Available: `https://www.itl.nist.gov/div898/handbook/eda/section3/eda3671.htm`.

[51] National Institute of Standards and Technology (NIST), *Table of the chi-square distribution.* [Online]. Available: `https://www.itl.nist.gov/div898/handbook/eda/section3/eda3674.htm`.

[52] G. Nemhauser, L. Wolsey, and M. Fisher, “An analysis of approximations for maximizing submodular set functions – I,” *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[53] C. Zach, “Robust bundle adjustment revisited,” in *European Conf. on Computer Vision (ECCV)*, 2014, pp. 772–787.

[54] H. Mobahi and J. W. Fisher, “On the link between gaussian homotopy continuation and convex envelopes,” in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, Springer, 2015, pp. 43–56.

[55] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. US Government printing office, 1948, vol. 55.

[56] Y. Xiang, R. Mottaghi, and S. Savarese, “Beyond PASCAL: A benchmark for 3d object detection in the wild,” in *IEEE Winter Conf. on Appl. of Computer Vision*, IEEE, 2014, pp. 75–82.

[57] J. Briales and J. Gonzalez-Jimenez, “Convex Global 3D Registration with Lagrangian Duality,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[58] K. Khoshelham, “Closed-form solutions for estimating a rigid motion from plane correspondences extracted from point clouds,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 78–91, 2016.

[59] Y.-L. Lin, V. I. Morariu, W. H. Hsu, and L. S. Davis, “Jointly optimizing 3D model fitting and fine-grained classification,” in *European Conf. on Computer Vision (ECCV)*, 2014.

[60] V. Ramakrishna, T. Kanade, and Y. Sheikh, “Reconstructing 3D human pose from 2D image landmarks,” in *European Conf. on Computer Vision (ECCV)*, 2012.

[61] X. Zhou, M. Zhu, S. Leonardos, and K. Daniilidis, “Sparse representation for 3D shape estimation: A convex relaxation approach,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 39, no. 8, pp. 1648–1661, 2017.

[62] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, “Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 4597–4604.

[63] L. Carlone, D. Rosen, G. Calafiore, J. Leonard, and F. Dellaert, “Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions,” in *iros*, PDF: `https://arxiv.org/abs/1506.00746`, Code: `https://www.bitbucket.org/lucacarlone/pgo3d-duality-opencode`, Datasets: `https://lucacarlone.mit.edu/datasets/`, Supplemental Material: `https://arxiv.org/abs/1506.00746`, 2015, pp. 125–132.

[64] L. Carlone, G. Calafiore, C. Tommolillo, and F. Dellaert, "Planar pose graph optimization: Duality, optimal solutions, and verification," *IEEE Trans. Robotics*, vol. 32, no. 3, pp. 545–565, 2016.

[65] L. Carlone, R. Aragues, J. Castellanos, and B. Bona, "A fast and accurate approximation for planar pose graph optimization," *Intl. J. of Robotics Research*, vol. 33, no. 7, pp. 965–987, 2014.

[66] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise consistent measurement set maximization for robust multi-robot map merging," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 2916–2923.

[67] P. Meer, D. Mintz, A. Rosenfeld, and D. Y. Kim, "Robust regression methods for computer vision: A review," *Intl. J. of Computer Vision*, vol. 6, no. 1, pp. 59–70, Apr. 1991.

[68] C. Stewart, "Robust parameter estimation in computer vision," *SIAM Review*, vol. 41, no. 3, pp. 513–537, 1999. DOI: 10.1137/S0036144598345802. eprint: https://doi.org/10.1137/S0036144598345802.

[69] M. Bosse, G. Agamennoni, and I. Gilitschenski, "Robust estimation and applications in robotics," *Foundations and Trends in Robotics*, vol. 4, no. 4, pp. 225–269, 2016, ISSN: 1935-8253. DOI: 10.1561/2300000047.

[70] D. Barath, J. Noskova, M. Ivashechkin, and J. Matas, "Magsac++, a fast, reliable and accurate robust estimator," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1304–1312.

[71] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.

[72] A. Chatterjee and V. M. Govindu, "Efficient and robust large-scale rotation averaging," in *Intl. Conf. on Computer Vision (ICCV)*, 2013, pp. 521–528.

[73] Q. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European Conf. on Computer Vision (ECCV)*, Springer, 2016, pp. 766–782.

[74] J. T. Barron, "A general and adaptive robust loss function," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4331–4339.

[75] N. Chebrolu, T. Läbe, O. Vysotska, J. Behley, and C. Stachniss, "Adaptive robust kernels for non-linear least squares problems," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2240–2247, 2021.

[76] J. Bazin, Y. Seo, R. Hartley, and M. Pollefeys, "Globally optimal inlier set maximization with unknown rotation and focal length," in *European Conf. on Computer Vision (ECCV)*, 2014, pp. 803–817.

[77] R. Hartley and F. Kahl, "Global optimization through rotation space search," *Intl. J. of Computer Vision*, vol. 82, no. 1, pp. 64–79, 2009.

[78] Y. Zheng, S. Sugimoto, and M. Okutomi, "Deterministically maximizing feasible subsystem for robust model fitting with unit norm constraint," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1825–1832.

[79] H. Li, "Consensus set maximization with guaranteed global optimality for robust geometry estimation," in *Intl. Conf. on Computer Vision (ICCV)*, 2009, pp. 1074–1080.

[80] P. Speciale, D. P. Paudel, M. R. Oswald, T. Kroeger, L. V. Gool, and M. Pollefeys, "Consensus maximization with linear matrix inequality constraints," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 5048–5056. DOI: 10.1109/CVPR.2017.536.

[81] T. Chin, Y. H. Kee, A. Eriksson, and F. Neumann, "Guaranteed outlier removal with mixed integer linear programs," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 5858–5866. DOI: 10.1109/CVPR.2016.631.

[82] G. Izatt, H. Dai, and R. Tedrake, "Globally optimal object pose estimation in point clouds with mixed-integer programming," in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, 2017.

[83] J. Yang, H. Li, and Y. Jia, "Optimal essential matrix estimation via inlier-set maximization," in *European Conf. on Computer Vision (ECCV)*, Springer, 2014, pp. 111–126.

[84] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A globally optimal solution to 3D ICP point-set registration," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 38, no. 11, pp. 2241–2254, Nov. 2016, ISSN: 0162-8828.

[85] O. Enqvist, E. Ask, F. Kahl, and K. Åström, "Robust fitting for multiple view geometry," in *European Conf. on Computer Vision (ECCV)*, Springer, 2012, pp. 738–751.

[86] C. Olsson, O. Enqvist, and F. Kahl, "A polynomial-time bound for matching and registration with outliers," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2008, pp. 1–8.

[87] H. Yang and L. Carlone, "One ring to rule them all: Certifiably robust geometric perception with outliers," in *Conf. on Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 18 846–18 859.

[88] L. Carlone and G. Calafiore, "Convex relaxations for pose graph optimization with outliers," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 2, pp. 1160–1167, 2018.

[89] H. Yang and L. Carlone, "A polynomial-time solution for robust registration with extreme outlier rates," in *Robotics: Science and Systems (RSS)*, 2019.

[90] A. P. Bustos, T.-J. Chin, F. Neumann, T. Friedrich, and M. Katzmann, "A practical maximum clique algorithm for matching with pairwise constraints," *arXiv preprint arXiv:1902.01534*, 2019.

[91] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 2, 1992.

[92] R. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Citeseer, 2009, pp. 3212–3217.

[93] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *Intl. Conf. on Computer Vision (ICCV)*, 2019, pp. 8958–8966.

[94] C. S. Chen, Y. P. Hung, and J. B. Cheng, "RANSAC-based DARCES: A new approach to fast automatic registration of partially overlapping range images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, no. 11, pp. 1229–1234, 1999.

[95] K. Arun, T. Huang, and S. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 5, pp. 698–700, Sep. 1987.

[96] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Amer.*, vol. 4, no. 4, pp. 629–642, Apr. 1987.

[97] J. C. Bazin, Y. Seo, and M. Pollefeys, "Globally optimal consensus set maximization through rotation search," in *Asian Conference on Computer Vision*, Springer, 2012, pp. 539–551.

[98] X. Zhou, M. Zhu, G. Pavlakos, S. Leonardos, K. G. Derpanis, and K. Daniilidis, "MonoCap: Monocular human motion capture using a CNN coupled with a geometric prior," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 41, no. 4, pp. 901–914, 2018.

[99] L. Kneip, H. Li, and Y. Seo, "UPnP: An optimal o(n) solution to the absolute pose problem with universal applicability," in *European Conf. on Computer Vision (ECCV)*, Springer, 2014, pp. 127–142.

[100] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 8, pp. 930–943, 2003.

[101] L. Ferraz, X. Binefa, and F. Moreno-Noguer, "Very fast solution to the pnp problem with algebraic outlier rejection," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 501–508.

[102] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," in *Robotics: Science and Systems (RSS)*, Jul. 2012.

[103] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[104] C. H. Tong and T. D. Barfoot, "Batch heterogeneous outlier rejection for feature-poor slam," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 2630–2637.

[105] C. H. Tong and T. D. Barfoot, "Evaluation of heterogeneous measurement outlier rejection schemes for robotic planetary surface mapping," *Acta Astronautica*, vol. 88, pp. 146–162, 2013.

[106] Y. Latif, C. D. C. Lerma, and J. Neira, "Robust loop closing over time.," in *Robotics: Science and Systems (RSS)*, 2012.

[107] G. Lee, F. Fraundorfer, and M. Pollefeys, "Robust pose-graph loop-closures with expectation-maximization," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[108] L. Wang and A. Singer, "Exact and stable recovery of rotations for robust synchronization," *Information and Inference: A Journal of the IMA*, vol. 30, 2013.

[109] F. Arrigoni, B. Rossi, P. Fragneto, and A. Fusiello, "Robust synchronization in SO(3) and SE(3) via low-rank and sparse matrix decomposition," *Comput. Vis. Image Underst.*, 2018.

[110] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[111] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME, Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[112] I. Diakonikolas, G. Kamath, D. Kane, J. Li, A. Moitra, and A. Stewart, "Robust estimators in high-dimensions without the computational intractability," *SIAM Journal on Computing*, vol. 48, no. 2, pp. 742–864, 2019.

[113] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[114] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 11, pp. 2715–2729, 2013.

[115] L. Liu, T. Li, and C. Caramanis, "High dimensional robust estimation of sparse models via trimmed hard thresholding," *arXiv preprint: 1901.08237*, 2019.

[116] P. Rousseeuw and M. Hubert, "Robust statistics for outlier detection," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 73–79, 2011.

[117] T. Zhang, "Adaptive forward-backward greedy algorithm for learning sparse representations," *IEEE Trans. on Information Theory*, vol. 57, no. 7, pp. 4689–4708, 2011.

[118] J. Liu, P. C. Cosman, and B. D. Rao, "Robust linear regression via ℓ_0 regularization," *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 698–713, 2018.

[119] S. Mishra, Y. Shoukry, N. Karamchandani, S. Diggavi, and P. Tabuada, "Secure state estimation against sensor attacks in the presence of noise," *IEEE Trans. on Control of Network Systems*, vol. 4, no. 1, pp. 49–59, 2017.

[120] E. Aghapour, F. Rahman, and J. Farrell, "Outlier accommodation by risk-averse performance-specified linear state estimation," in *2018 IEEE Conference on Decision and Control*, IEEE, 2018, pp. 2310–2315.

[121] H. Yang and L. Carlone, "Certifiably optimal outlier-robust geometric perception: Semidefinite relaxations and scalable global optimization," *IEEE Trans. Pattern Anal. Machine Intell.*, 2022.

[122] J. Yang, M. Ward, and J. Akhtar, "The development of safety cases for an autonomous vehicle: A comparative study on different methods," SAE Technical Paper, Tech. Rep., 2017.

[123] R. Yan, S. J. Dunnett, and L. M. Jackson, "Reliability modelling of automated guided vehicles by the use of failure modes effects and criticality analysis, and fault tree analysis," in *5th student conference on operational research (SCOR 2016)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[124] P. Antonante, D. Spivak, and L. Carlone, "Monitoring and diagnosability of perception systems," *arXiv preprint: 2011.07010*, 2020.

[125] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan kaufmann, 1988.

[126] J. Liu and J.-M. Park, ""seeing is not always believing": Detecting perception error attacks against autonomous vehicles," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2209–2223, 2021.

[127] A. Sharma, N. Azizan, and M. Pavone, "Sketching curvature for efficient out-of-distribution detection for deep neural networks," in *Uncertainty in Artificial Intelligence*, PMLR, 2021, pp. 1958–1967.

[128] A. Dokhanchi, H. B. Amor, J. Deshmukh, and G. Fainekos, "Evaluating perception systems for autonomous vehicles using quality temporal logic," in *Intl. Conf. on Runtime Verification (RV)*, 2018.

[129] L. A. Wolsey, *Integer programming.* John Wiley & Sons, 2020.

[130] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of constraint programming.* Elsevier, 2006.

[131] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques.* The MIT Press, 2009.

[132] S. E. Shimony, "Finding maps for belief networks is np-hard," *Artificial intelligence*, vol. 68, no. 2, pp. 399–410, 1994.

[133] S. Nowozin and C. H. Lampert, *Structured learning and prediction in computer vision.* Now publishers Inc, 2011, vol. 6.

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

[135] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[136] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Intl. Conf. on Learning Representations (ICLR)*, Apr. 2017.

[137] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[138] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning*, PMLR, 2020, pp. 1725–1735.

[139] W. L. Hamilton., R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems (NIPS)*, Dec. 2017, pp. 1025–1035.

[140] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" In *Intl. Conf. on Learning Representations (ICLR)*, May 2019.

[141] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[142] Z. Zhang, D. Lyu, P. Arcaini, L. Ma, I. Hasuo, and J. Zhao, "Falsifai: Falsification of ai-enabled hybrid control systems guided by time-aware coverage criteria," *IEEE Transactions on Software Engineering*, 2022.

[143] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, "Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," *arXiv preprint arXiv:2106.11810*, 2021.

[144] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in *In proceedings of IEEE Symposium Series on Computational Intelligence*, IEEE, 2015, pp. 159–166.

[145] A. Sengupta and A. T. Dahbura, "On self-diagnosable multiprocessor systems: Diagnosis by the comparison approach," *IEEE Transactions on Computers*, vol. 41, no. 11, pp. 1386–1396, 1992.

[146] S. L. Hakimi and A. T. Amin, "Characterization of connection assignment of diagnosable systems," *IEEE Transactions on Computers*, vol. 100, no. 1, pp. 86–88, 1974.

[147] LG, *LGSVL Simulator*. [Online]. Available: `https://www.lgsvlsimulator.com`.

[148] Baidu, *Apollo Auto*. [Online]. Available: `https://apollo.auto/`.

[149] Baidu, *Apollo Auto*. [Online]. Available: `https://github.com/ApolloAuto/apollo`.

[150] P. Antonante, D. Spivak, and L. Carlone, "Monitoring and diagnosability of perception systems," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[151] Z. Liu, Z. Wu, and R. Tóth, "SMOKE: Single-stage monocular 3d object detection via keypoint estimation," *arXiv preprint arXiv:2002.10111*, 2020.

[152] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

[153] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[154] *Google's self-driving startup Waymo is introducing fully driverless rides to San Francisco*, `https://www.continental-automotive.com/getattachment/5430d956-1ed7-464b-afa3-cd9cdc98ad63/ARS408-21_datasheet_en_170707_V07.pdf.pdf`, Accessed: 2022-05-15.

[155] D. F. Crouse, "On implementing 2d rectangular assignment algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1679–1696, 2016.

[156] Google, *Google OR-Tools.* [Online]. Available: `https://developers.google.com/optimization`.

[157] *Grante Library for Inference and Estimation on Discrete Factor Graph Model*, `http://www.nowozin.net/sebastian/grante/`, Accessed: 2022-05-15.

[158] W. Falcon *et al.*, *Pytorch lightning*, `https://github.com/PytorchLightning/pytorch-lightning`, 2019.

[159] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 2017, pp. 464–472.

[160] The Guardian, *Tesla driver dies in first fatal crash while using autopilot mode*, `www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk`, Accessed: 2022-05-15, 2016.

[161] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[162] ISO Standard, *Road vehicles – functional safety*, ISO 26262-1:2011, 2011.

[163] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE Int. J. Trans. Safety*, vol. 4, no. 1, 2016.

[164] F. Concas, J. K. Nurminen, T. Mikkonen, and S. Tarkoma, *Validation Frameworks for Self-Driving Vehicles: A Survey.* Springer, 2021, pp. 197–212.

[165] P. Koopman, U. Ferrell, F. Fratrik, and M. Wagner, "A safety standard approach for fully autonomous vehicles," in *International Conference on Computer Safety, Reliability, and Security*, Springer, 2019, pp. 326–332.

[166] Underwriters Laboratories, *ANSI/UL 4600 Standard for Safety for the Evaluation of Autonomous Products.* [Online]. Available: `https://ul.org/UL4600`.

[167] F. Ingrand, "Recent trends in formal validation and verification of autonomous robots software," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 321–328.

[168] A. Desai, T. Dreossi, and S. Seshia, "Combining model checking and runtime verification for safe robotics," in *International Conference on Runtime Verification*, Springer, 2017, pp. 172–189.

[169] B. Hoxha and G. Fainekos, "Planning in dynamic environments through temporal logic monitoring," in *AAAI Workshop: Planning for Hybrid Systems*, vol. 16, 2016, p. 12.

[170] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 1481–1488.

[171] S. Dathathri and R. Murray, "Decomposing GR(1) games with singleton liveness guarantees for efficient synthesis," *arXiv*, vol. abs/1709.07094, 2017.

[172] S. Ghosh, D. Sadigh, P. Nuzzo, V. Raman, A. Donzé, A. L. Sangiovanni-Vincentelli, S. S. Sastry, and S. A. Seshia, "Diagnosis and repair for synthesis from signal temporal logic specifications," in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '16, ACM, 2016, pp. 31–40.

[173] W. Li, L. Dworkin, and S. A. Seshia, "Mining assumptions for synthesis," in *Ninth ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMPCODE2011)*, 2011, pp. 43–50.

[174] W. Li, D. Sadigh, S. Sastry, and S. Seshia, "Synthesis for human-in-the-loop control systems," in *Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2014.

[175] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[176] S. Mitsch, K. Ghorbal, D. Vogelbacher, and A. Platzer, "Formal verification of obstacle avoidance and navigation of ground robots," *The International Journal of Robotics Research*, vol. 36, no. 12, pp. 1312–1340, 2017.

[177] N. Roohi, R. Kaur, J. Weimer, O. Sokolsky, and I. Lee, "Self-driving vehicle verification towards a benchmark," *arXiv preprint arXiv:1806.08810*, 2018.

[178] R. C. Cardoso, M. Farrell, M. Luckcuck, A. Ferrando, and M. Fisher, "Heterogeneous verification of an autonomous curiosity rover," pp. 353–360, 2020.

[179] S. Jha, V. Raman, D. Sadigh, and S. Seshia, "Safe autonomy under perception uncertainty using chance-constrained temporal logic," *Journal of Automated Reasoning*, vol. 60, pp. 43–62, 2017.

[180] M. Foughali, B. Berthomieu, S. Dal Zilio, P.-E. Hladik, F. Ingrand, and A. Mallet, "Formal verification of complex robotic systems on resource-constrained platforms," in *2018 IEEE/ACM 6th International FME Workshop on Formal Methods in Software Engineering (FormaliSE)*, 2018, pp. 2–9.

[181] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Toward verified artificial intelligence," *Communications of the ACM*, vol. 65, no. 7, pp. 46–55, 2022.

[182] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–41, 2019.

[183] T. Dreossi, D. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. Seshia, "VERIFAI: A toolkit for the design and analysis of artificial intelligence-based systems," *ArXiv:1902.04245*, 2019.

[184] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: A language for scenario specification and scene generation," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019, pp. 63–78.

[185] K. Leahy, E. Cristofalo, C. Vasile, A. Jones, E. Montijano, M. Schwager, and C. Belta, "Control in belief space with temporal logic specifications using vision-based localization," *Intl. J. of Robotics Research*, vol. 38, Apr. 2019.

[186] A. Balakrishnan, A. G. Puranic, X. Qin, A. Dokhanchi, J. V. Deshmukh, H. Ben Amor, and G. Fainekos, "Specifying and evaluating quality metrics for vision-based perception systems," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019, pp. 1433–1438.

[187] T. Dreossi, S. Ghosh, A. Sangiovanni-Vincentelli, and S. Seshia, "Systematic testing of convolutional neural networks for autonomous driving," *ArXiv*, vol. abs/1708.03309, 2017.

[188] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2267–2281.

[189] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Attacking vision-based perception in end-to-end autonomous driving models," *Journal of Systems Architecture*, vol. 110, p. 101 766, 2020.

[190] H. Delecki, M. Itkina, B. Lange, R. Senanayake, and M. J. Kochenderfer, "How do we fail? stress testing perception in autonomous vehicles," *arXiv preprint arXiv:2203.14155*, 2022.

[191] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, vol. 6, pp. 14 410–14 430, 2018.

[192] Q. M. Rahman, P. Corke, and F. Dayoub, "Run-time monitoring of machine learning for robotic perception: A survey of emerging trends," *IEEE Access*, vol. 9, pp. 20 067–20 075, 2021.

[193] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *arXiv preprint arXiv:2110.11334*, 2021.

[194] S. Mohseni, M. Pitale, J. Yadawa, and Z. Wang, "Self-supervised learning for generalizable out-of-distribution detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 5216–5223.

[195] J. Nitsch, M. Itkina, R. Senanayake, J. Nieto, M. Schmidt, R. Siegwart, M. J. Kochenderfer, and C. Cadena, "Out-of-distribution detection for automotive perception," in *In proceedings of IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 2938–2943.

[196] R. Sinha, A. Sharma, S. Banerjee, T. Lew, R. Luo, S. M. Richards, Y. Sun, E. Schmerling, and M. Pavone, "A system-level view on out-of-distribution data in robotics," *arXiv preprint arXiv:2212.14020*, 2022.

[197] P. Oberdiek, M. Rottmann, and G. A. Fink, "Detection and retrieval of out-of-distribution objects in semantic segmentation," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition workshops*, 2020, pp. 328–329.

[198] Q. M. Rahman, N. Sünderhauf, P. Corke, and F. Dayoub, "Fsnet: A failure detection framework for semantic segmentation," 2, vol. 7, 2022, pp. 3030–3037. DOI: `10.1109/LRA.2022.3143219`.

[199] J. Lambert and J. Hays, "Trust, but verify: Cross-modality fusion for hd map change detection," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[200] D. Knowles and G. Gao, "Euclidean distance matrix-based rapid fault detection and exclusion," *NAVIGATION: Journal of the Institute of Navigation*, vol. 70, no. 1, 2023, ISSN: 0028-1522. DOI: `10.33012/navi.555`. eprint: `https://navi.ion.org/content/70/1/navi.555.full.pdf`. [Online]. Available: `https://navi.ion.org/content/70/1/navi.555`.

[201] M. Joerger, S. Pullen, and R. Capua, "Development of gnss augmentation integrity messaging standards for automotive applications," in *Navigation Conference*, vol. 2021, 2021.

[202] H. Jiang, T. Li, D. Song, and C. Shi, "An effective integrity monitoring scheme for gnss/ins/vision integration based on error state ekf model," *IEEE Sensors Journal*, vol. 22, no. 7, pp. 7063–7073, 2022.

[203] A. El-Mowafy and N. Kubo, "Integrity monitoring of vehicle positioning in urban environment using rtk-gnss, imu and speedometer," *Measurement Science and Technology*, vol. 28, no. 5, p. 055 102, 2017.

[204] F. A. C. de Oliveira, F. S. Torres, and A. García-Ortiz, "Recent advances in sensor integrity monitoring methods-a review," *IEEE Sensors Journal*, 2022.

[205] X. Wang, C. Toth, and D. Grejner-Brzezinska, "A survey on integrity monitoring of gnss navigation for ground vehicles," in *In proceedings of the International Technical Meeting of the Satellite Division of The Institute of Navigation*, 2021, pp. 2591–2601.

[206] C. You, Z. Hau, and S. Demetriou, "Temporal consistency checks to detect lidar spoofing attacks on autonomous vehicle perception," in *Proceedings of the 1st Workshop on Security and Privacy for Mobile AI*, 2021, pp. 13–18.

[207] A. Balakrishnan, J. Deshmukh, B. Hoxha, T. Yamaguchi, and G. Fainekos, "Percemon: Online monitoring for perception systems," in *International Conference on Runtime Verification*, Springer, 2021, pp. 297–308.

[208] D. Kang, D. Raghavan, P. Bailis, and M. Zaharia, "Model assertions for debugging machine learning," in *NIPS*, 2018.

[209] A. Santamaria-Navarro, R. Thakker, D. D. Fan, B. Morrell, and A.-a. Aghamohammadi, *Towards resilient autonomous navigation of drones*, 2020. eprint: `2008.09679`.

[210] B. Cai, L. Huang, and M. Xie, "Bayesian networks in fault diagnosis," *IEEE Transactions on industrial informatics*, vol. 13, no. 5, pp. 2227–2240, 2017.

[211] A. Abdollahi, K. R. Pattipati, A. Kodali, S. Singh, S. Zhang, and P. B. Luh, "Probabilistic graphical models for fault diagnosis in complex systems," in *Principles of Performance and Reliability Modeling and Evaluation*, Springer, 2016, pp. 109–139.

[212] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi, "Applications of machine learning to machine fault diagnosis: A review and roadmap," *Mechanical Systems and Signal Processing*, vol. 138, p. 106 587, 2020.

[213] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[214] J. De Kleer and B. C. Williams, "Diagnosing multiple faults," *Artificial intelligence*, vol. 32, no. 1, pp. 97–130, 1987.

[215] K. Bhat, "Algorithms for finding diagnosability level and t-diagnosis in a network of processors," in *Proceedings of the ACM'82 conference*, 1982, pp. 164–168.

[216] A. T. Dahbura, "System-level diagnosis: A perspective for the third decade," in *Concurrent Computations*, Springer, 1988, pp. 411–434.

[217] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on automatic control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[218] J. Zaytoon and S. Lafortune, "Overview of fault diagnosis methods for discrete event systems," *Annual Reviews in Control*, vol. 37, no. 2, pp. 308–320, 2013.

[219] T. M. Tuxi, L. K. Carvalho, E. V. Nunes, and A. E. da Cunha, "Diagnosability verification using ltl model checking," *Discrete Event Dynamic Systems*, pp. 1–35, 2022.

[220] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *European Conference on Computer Vision*, Springer, 2020, pp. 683–700.

[221] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Neurohr, C. Gutenkunst, and E. Böde, "Criticality metrics for automated driving: A review and suitability analysis of the state of the art," *Archives of Computational Methods in Engineering*, vol. 30, no. 1, pp. 1–35, 2023.

[222] J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, *et al.*, "Scene transformer: A unified architecture for predicting future trajectories of multiple agents," in *International Conference on Learning Representations*, 2021.

[223] Y. Han, J. Banfi, and M. Campbell, "Planning paths through unknown space by imagining what lies therein," in *Proceedings of Conference on Robot Learning (CoRL)*, 2021, pp. 905–914.

[224] H. Joe, *Dependence modeling with copulas.* CRC press, 2014.

[225] M. Sklar, "Fonctions de repartition an dimensions et leurs marges," *Publications de l'Institut de Statistique de l'Université de Paris*, vol. 8, pp. 229–231, 1959.

[226] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, p. 1805, 2000.

[227] S. Albeaik, A. Bayen, M. T. Chiri, X. Gong, A. Hayat, N. Kardous, A. Keimer, S. T. McQuade, B. Piccoli, and Y. You, "Limitations and improvements of the intelligent driver model (IDM)," *SIAM Journal on Applied Dynamical Systems*, vol. 21, no. 3, pp. 1862–1892, 2022.

[228] Motional, *NuPlan-devkit.* [Online]. Available: `https://github.com/motional/nuplan-devkit`.

[229] S. Topan, K. Leung, Y. Chen, P. Tupekar, E. Schmerling, J. Nilsson, M. Cox, and M. Pavone, "Interaction-dynamics-aware perception zones for obstacle detection safety evaluation," in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1201–1210. DOI: `10.1109/IV51971.2022.9827409`.

[230] A. Kamenev, L. Wang, O. B. Bohan, I. Kulkarni, B. Kartal, A. Molchanov, S. Birchfield, D. Nistér, and N. Smolyanskiy, "Predictionnet: Real-time joint probabilistic traffic prediction for planning, control, and simulation," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8936–8942.

[231] D. Hendrycks, S. Basart, M. Mazeika, M. Mostajabi, J. Steinhardt, and D. Song, "Scaling out-of-distribution detection for real-world settings," *arXiv preprint arXiv:1911.11132*, 2019.

[232] F. Li, P. Bonnifait, and J. Ibañez-Guzmán, "Map-aided dead-reckoning with lane-level maps and integrity monitoring," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 1, pp. 81–91, 2018.

[233] E. Galceran, E. Olson, and R. M. Eustice, "Augmented vehicle tracking under occlusions for decision-making in autonomous driving," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 3559–3565.

[234] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 249–265, 2020.

[235] P. Zechel, R. Streiter, K. Bogenberger, and U. Göhner, "Pedestrian occupancy prediction for autonomous vehicles," in *Proceedings of IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 230–235.

[236] J. Dahl, G. R. de Campos, C. Olsson, and J. Fredriksson, "Collision avoidance: A literature review on threat-assessment techniques," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 101–113, 2018.

[237] A. Farid, D. Snyder, A. Z. Ren, and A. Majumdar, "Failure prediction with statistical guarantees for vision-based robot control," *arXiv preprint arXiv:2202.05894*, 2022.

[238] A. Farid, S. Veer, and A. Majumdar, "Task-driven out-of-distribution detection with statistical guarantees for robot learning," in *Proceedings of Conference on Robot Learning (CoRL)*, 2021, pp. 970–980.

[239] A. Farid, S. Veer, B. Ivanovic, K. Leung, and M. Pavone, "Task-relevant failure detection for trajectory predictors in autonomous vehicles," *arXiv preprint arXiv:2207.12380*, 2022.

[240] N. Agarwal and Y.-T. Chen, "Risk perception in driving scenes," in *Proceedings of Workshop on Machine Learning for Autonomous Driving at NeurIPS*, 2022.

[241] A. Bansal, J. Singh, M. Verucchi, M. Caccamo, and L. Sha, "Risk ranked recall: Collision safety metric for object detection systems in autonomous vehicles," in *Proceedings of Mediterranean Conference on Embedded Computing (MECO)*, 2021, pp. 1–4.

[242] J. Bernhard, P. Hart, A. Sahu, C. Schöller, and M. G. Cancimance, "Risk-based safety envelopes for autonomous vehicles under perception uncertainty," in *Proceedings of IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 104–111.

[243] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on automatic control*, vol. 50, no. 7, pp. 947–957, 2005.

[244] K. Leung, E. Schmerling, M. Zhang, M. Chen, J. Talbot, J. C. Gerdes, and M. Pavone, “On infusing reachability-based safety assurance within planning frameworks for human–robot vehicle interactions,” *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1326–1345, 2020.

[245] K.-C. Hsu, A. Z. Ren, D. P. Nguyen, A. Majumdar, and J. F. Fisac, “Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees,” *arXiv preprint arXiv:2201.08355*, 2022.

[246] H. Hu, K. Nakamura, and J. F. Fisac, “Sharp: Shielding-aware robust planning for safe and efficient human-robot interaction,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5591–5598, 2022.

[247] M. N. Ahangar, Q. Z. Ahmed, F. A. Khan, and M. Hafeez, “A survey of autonomous vehicles: Enabling communication technologies and challenges,” *Sensors*, vol. 21, no. 3, 2021, ISSN: 1424-8220. DOI: `10.3390/s21030706`. [Online]. Available: `https://www.mdpi.com/1424-8220/21/3/706`.

[248] S. Darbha, S. Konduri, and P. R. Pagilla, “Benefits of v2v communication for autonomous and connected vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1954–1963, 2019. DOI: `10.1109/TITS.2018.2859765`.

[249] J. Cui, G. Sabaliauskaite, L. S. Liew, F. Zhou, and B. Zhang, “Collaborative analysis framework of safety and security for autonomous vehicles,” *IEEE Access*, vol. 7, pp. 148 672–148 683, 2019. DOI: `10.1109/ACCESS.2019.2946632`.

[250] C. Jung, D. Lee, S. Lee, and D. H. Shim, “V2x-communication-aided autonomous driving: System design and experimental validation,” *Sensors*, vol. 20, no. 10, 2020, ISSN: 1424-8220. DOI: `10.3390/s20102903`. [Online]. Available: `https://www.mdpi.com/1424-8220/20/10/2903`.

[251] L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs, “Enhancements of v2x communication in support of cooperative autonomous driving,” *IEEE Communications Magazine*, vol. 53, no. 12, pp. 64–70, 2015. DOI: `10.1109/MCOM.2015.7355568`.

[252] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[253] D. Foster, H. Karloff, and J. Thaler, “Variable selection is hard,” in *Conference on Learning Theory (COLT)*, 2015, pp. 696–709.

[254] W. Weibull, “A statistical distribution function of wide applicability,” *applmech*, vol. 18, pp. 293–297, 1951.

[255] B. W. Bolch, “The teacher’s corner: More on unbiased estimation of the standard deviation,” *The American Statistician*, vol. 22, no. 3, pp. 27–27, 1968.

[256] A. Dvoretzky, J. Kiefer, and J. Wolfowitz, “Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator,” *The Annals of Mathematical Statistics*, pp. 642–669, 1956.

[257] P. Massart, "The tight constant in the dvoretzky-kiefer-wolfowitz inequality," *The Annals of Probability*, pp. 1269–1283, 1990.