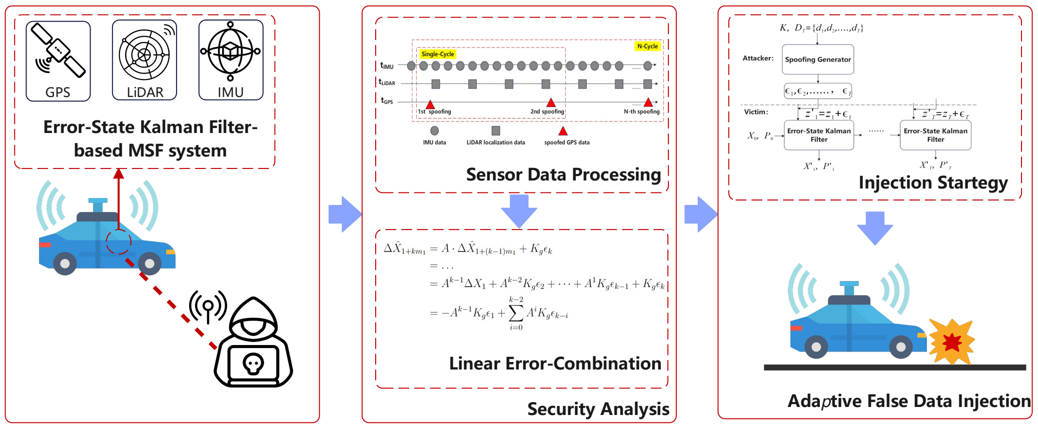
Graphical Abstract

Security Analysis and Adaptive False Data Injection against Multi-Sensor Fusion Localization for Autonomous Driving

Linqing Hu, Junqi Zhang, Jie Zhang, Shaoyin Cheng, Yuyi Wang, Weiming Zhang, Nenghai Yu


GPS
LiDAR
IMU
Error-State Kalman Filter-based MSF system
Single-Cycle
N-Cycle
t_{IMU}
t_{LiDAR}
t_{GPS}
1st spoofing
2nd spoofing
N-th spoofing
IMU data
LiDAR localization data
spoofed GPS data
Sensor Data Processing
$\Delta\hat{X}_{1+km_1} = A \cdot \Delta\hat{X}_{1+(k-1)m_1} + K_g\epsilon_k = \dots = A^{k-1}\Delta X_1 + A^{k-2}K_g\epsilon_2 + \cdots + A^1 K_g\epsilon_{k-1} + K_g\epsilon_k = -A^{k-1}K_g\epsilon_1 + \sum_{i=0}^{k-2} A^i K_g\epsilon_{k-i}$
Linear Error-Combination
Security Analysis
K, $D_T=\{d_1,d_2,\dots,d_T\}$
Attacker:
Spoofing Generator
$\epsilon_1,\epsilon_2,\dots,\epsilon_T$
Victim:
$z'_1=z_1+\epsilon_1$
$z'_T=z_T+\epsilon_T$
X_0, P_0
Error-State Kalman Filter
Error-State Kalman Filter
X'_1, P'_1
X'_T, P'_T
Injection Startegy
Adaptive False Data Injection

Highlights

Security Analysis and Adaptive False Data Injection against Multi-Sensor Fusion Localization for Autonomous Driving

Linqing Hu, Junqi Zhang, Jie Zhang, Shaoyin Cheng, Yuyi Wang, Weiming Zhang, Nenghai Yu

- Designed a new error analysis model for ESKF-based MSF in autonomous vehicles
- Discovered a linear error-combination vulnerability exploitable in ESKF systems
- Proposed an energy-efficient attack strategy using constrained optimization
- Achieved 9% less energy usage in simulations and 18% less in real-world datasets
- Validated method effectiveness in an autonomous vehicle with end-to-end testing

Security Analysis and Adaptive False Data Injection against Multi-Sensor Fusion Localization for Autonomous Driving

Linqing Hu[a,b], Junqi Zhang[a,b,*], Jie Zhang[c], Shaoyin Cheng[a,b], Yuyi Wang[d], Weiming Zhang[a,b,*], Nenghai Yu[a,b]

[a]*School of Cyber Science and Technology, University of Science and Technology of China, Hefei, 230026, Anhui, China*
[b]*Anhui Province Key Laboratory of Digital Security, Hefei, 230026, Anhui, China*
[c]*CFAR and IHPC, Agency for Science, Technology and Research (A*STAR), Singapore*
[d]*CRRC Zhuzhou Institute Co., Ltd., Zhuzhou, 412001, Hunan, China*
[e]*Tengen Intelligence Institute, Zhuzhou, 412000, Hunan, China*

Abstract

Multi-sensor Fusion (MSF) algorithms are critical components in modern autonomous driving systems, particularly in localization and AI-powered perception modules, which play a vital role in ensuring vehicle safety. The Error-State Kalman Filter (ESKF), specifically employed for localization fusion, is widely recognized for its robustness and accuracy in MSF implementations. While existing studies have demonstrated the vulnerability of ESKF to sensor spoofing attacks, these works have primarily focused on a black-box implementation, leading to an insufficient security analysis. Specifically, due to the lack of theoretical guidance in previous methods, these studies have consistently relied on exponential functions to fit attack sequences across all scenarios. As a result, the attacker had to explore an extensive parameter space to identify effective attack sequences, lacking the ability to adaptively generate optimal ones. This paper aims to fill this crucial gap by conducting a thorough security analysis of the ESKF model and presenting a simple ap-

[*]Corresponding authors.

Email addresses: `hlq2018@mail.ustc.edu.cn` (Linqing Hu), `zhangjunqi@mail.ustc.edu.cn` (Junqi Zhang), `zhang_jie@cfar.a-star.edu.sg` (Jie Zhang), `sycheng@ustc.edu.cn` (Shaoyin Cheng), `yuyiwang920@gmail.com` (Yuyi Wang), `zhangwm@ustc.edu.cn` (Weiming Zhang), `ynh@ustc.edu.cn` (Nenghai Yu)

proach for modeling injection errors in these systems. By utilizing this error modeling, we introduce a new attack strategy that employs constrained optimization to reduce the energy needed to reach the same deviation target, guaranteeing that the attack is both efficient and effective. This method increases the stealthiness of the attack, making it harder to detect. Unlike previous methods, our approach can dynamically produce nearly perfect injection signals without requiring multiple attempts to find the best parameter combination in different scenarios. Through extensive simulations and real-world experiments, we demonstrate the superiority of our method compared to state-of-the-art attack strategies. Our results indicate that our approach requires significantly less injection energy to achieve the same deviation target. Additionally, we validate the practical applicability and impact of our method through end-to-end testing on an AI-powered autonomous driving system.

Keywords:
Error-State Kalman Filter, Multi-Sensor Fusion, Security Analysis, Adaptive False Data Injection, Autonomous Driving, Sensor Spoofing, Attack Strategy

---

1. Introduction

The rapid development and deployment of commercial autonomous driving systems [1, 2, 3] have significantly advanced the capabilities of modern vehicles, enabling them to navigate complex environments with minimal human intervention. These systems are increasingly being integrated into everyday transportation, promising enhanced safety, efficiency, and convenience, such as Google's Waymo [1] and Baidu Apollo [2]. However, alongside these advancements, there is a growing recognition of the critical need to ensure the security and reliability of these systems [4, 5, 6]. Autonomous vehicles rely heavily on an array of sensors and sophisticated algorithms to accurately perceive their surroundings and make split-second decisions. Among these components, the localization module is particularly crucial, as it is responsible for determining the centimeter level position of the vehicle in real-time [7, 8]. Accurate localization is not only essential for safe and efficient navigation but also forms the backbone of many other decision-making processes within the vehicle including, but not limited to, the perception, planning, and control modules. Consequently, any compromise to this module could lead to serious consequences, potentially putting the safety of the vehicle and its occupants at risk.

While the security of perception systems, such as image-based attacks on cameras [9, 10, 11, 12, 13], has received considerable attention, the localization module's security has been comparatively underexplored [14]. This module is pivotal for the vehicle's navigation, providing high-precision and robust localization that is indispensable for the reliable functioning of autonomous driving systems. Despite its critical importance, existing security research has largely focused on other areas [11, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23], often overlooking the specific challenges and vulnerabilities associated with localization. As autonomous vehicles increasingly rely on precise localization to make real-time driving decisions, the potential consequences of compromised localization are severe, making this an urgent area for further investigation. The ability of an autonomous vehicle to maintain accurate positioning under various environmental conditions and potential threats is a key determinant of its overall safety and effectiveness.

To enhance the accuracy and robustness of localization, current autonomous driving systems often employ Multi-Sensor Fusion (MSF) techniques [24, 25, 26, 27, 28, 29, 30, 31]. MSF integrates data from multiple sensors, such as Inertial Measurement Units (IMU), Light Detection and Ranging (LiDAR), and Global Positioning Systems (GPS), to provide a comprehensive understanding of the vehicle's environment. This fusion of sensor data mitigates the weaknesses of individual sensors, enhancing the overall resilience of the localization process. By leveraging the complementary strengths of different sensors, MSF not only improves accuracy but also provides a more reliable estimate of the vehicle's position, even in challenging conditions where one or more sensors may be compromised or degraded. However, despite the robustness that MSF offers, recent research has demonstrated that these systems remain vulnerable to sensor spoofing attacks [14], particularly those targeting the fusion process itself. For example, GPS spoofing attacks have been shown to disrupt vehicle positioning effectively, revealing critical vulnerabilities in black-box MSF implementations like those used in Apollo [2].

One of the widely adopted models within MSF for localization is the Error-State Kalman Filter (ESKF) [32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44]. The ESKF is specifically designed to handle the nonlinearities present in the system and measurement models by estimating the error state rather than the full state. This approach provides several advantages, including improved numerical stability and the ability to handle small perturbations effectively. The ESKF fuses data from IMU, LiDAR, and GPS, among other sensors, to produce a highly accurate and robust estimate of the vehicle's position and orientation. Despite its effectiveness in handling sensor fusion, the security aspects of ESKF-based systems have not been thoroughly analyzed. This lack of analysis leaves a critical gap in understanding how adversarial attacks can exploit the ESKF's fusion process to compromise the vehicle's localization accuracy and reliability.

In related studies, Nashimoto [45] was the first to analyze in detail the behavior of fusion filter models under different sensor interference. He discovered that under certain observation injections, the filtering results could be misled. However, this study lacked a comprehensive security analysis. Subsequently, Shen [14] successfully implemented GPS sensor spoofing attacks on the MSF-based localization system in the commercial autonomous driving platform Apollo and introduced a specific attack strategy called FusionRipper. Due to the commercial value of the localization system, it was not open-sourced. Although Shen's paper also touched upon the security analysis of the localization model, it was still focused on the traditional Kalman Filter model, and the black-box nature of the system prevented a deeper understanding of internal error propagation. Building on Shen's work, Chang [46, 47] further improved the success rate of FusionRipper in different scenarios by utilizing scene classification techniques. Additionally, Chang provided a detailed derivation and analysis of error propagation within fusion filter models under sensor spoofing conditions. However, like Shen, Chang's analysis was still centered on traditional KF models. Furthermore, Baidu's research [24] indicates that the localization module used in the commercial Apollo autonomous driving system is based on an ESKF fusion filter model. Overall, while ESKF is widely regarded for its effectiveness in fusing data from diverse sensors, its security aspects have not been thoroughly analyzed. This gap in the literature underscores the need for a more detailed understanding of fusion model vulnerabilities and the development of targeted defense mechanisms. Specifically, due to the lack of theoretical guidance in previous methods, exponential functions were consistently used to model attack sequences across all scenarios. Consequently, attackers were forced to navigate an extensive parameter space to identify effective attack sequences, without the ability to adaptively generate optimal ones.

In this paper, we seek to bridge this gap by conducting a comprehensive analysis of ESKF-based localization systems within the context of autonomous driving security. We explore the error dynamics within the ESKF framework, identifying that errors can be represented as a linear combination of injected signals, particularly under steady-state conditions. This insight allows us to formulate a new class of attacks that are not only effective but also energy-efficient, providing a deeper understanding of how adversarial inputs can be optimized to manipulate the system. Our approach offers a more nuanced view of how ESKF-based systems can be compromised, going beyond the limitations of black-box models and providing a clearer pathway for developing robust countermeasures.

Building on this foundation, we propose a novel attack strategy that leverages constrained optimization techniques to minimize the energy required to achieve specific attack objectives. Unlike existing methods, such as the state-of-the-art FusionRipper, which depends on heuristic approaches for parameter selection and requires extensive retesting across different scenarios, our method adaptively computes optimal injection signals across various scenarios, ensuring consistent performance while reducing the need for manual tuning. This adaptability makes our approach more versatile and practical for real-world applications, where attack scenarios can vary significantly, and the ability to quickly and efficiently adapt to new conditions is critical.

We validate the effectiveness of our approach through simulations and real-world experiments, demonstrating that our method consistently outperforms FusionRipper in terms of both efficacy and efficiency. Specifically, our approach achieves similar or greater levels of disruption with approximately 9% less energy consumption in simulated environments and about 18% less in real-world scenarios. Finally, we conduct the end-to-end experiment on a real vehicle equipped with a commercially available AI-powered autonomous driving system to validate the effectiveness of our method.

Our contributions are summarized as follows:

- **Design of an Error Analysis Model:** We design a new error analysis model specifically for ESKF-based localization systems in autonomous vehicles, providing a detailed understanding of how errors propagate within these systems.

- **Discovery of a Linear Error-Combination:** We identify that the error in ESKF-based systems can be expressed as a linear combination of injection signals, particularly under steady-state conditions, revealing a key vulnerability that can be exploited by adversaries.

- **Proposition of an Energy-Efficient Attack Method:** We introduce a novel attack strategy that utilizes constrained optimization to minimize the energy required for achieving targeted localization errors, offering a more efficient alternative to existing methods.

- **Extensive Experimental Validation:** We conduct comprehensive evaluations through simulations, real-world datasets, and end-to-end testing on an AI-powered autonomous driving system. Our results demonstrate that our method outperforms the state-of-the-art Fusion-Ripper attack, requiring 9% less energy in simulations and 18% less in real-world scenarios while confirming its practical applicability in actual driving conditions.

2. Problem Formulation

2.1. Multi-Sensor Fusion

In existing AI-powered autonomous driving systems, the localization module predominantly employs multi-sensor fusion-based techniques, such as Apollo. Multi-sensor fusion systems combine position information from GNSS (Global Navigation Satellite System), position and orientation data from LiDAR (Light Detection and Ranging), and inertial estimates from IMU(Inertial Measurement Unit) to provide accurate state estimations such as position, velocity, and orientation [24], as shown in Figure 1. Compared to traditional navigation methods, multi-sensor fusion leverages redundant observations from independent sensors like GPS and LiDAR, enabling higher localization accuracy, such as centimeter-level precision, and improved robustness in the positioning results.

In multi-sensor fusion schemes, filtering methods based on the Kalman Filter (KF) are the dominant approach, with the Error-State Kalman Filter (ESKF) being the widely used model. Compared to the classical KF model, the ESKF divides the state variables into nominal states and error states. The nominal states are estimated without considering errors, while the error states are estimated independently and updated using the observations provided by the sensors. Subsequently, during each observation update, the error states are used to correct the nominal states, enabling high-precision state estimation.

Specifically, the overall estimation process of the ESKF consists of two steps: the prediction step and the update step.

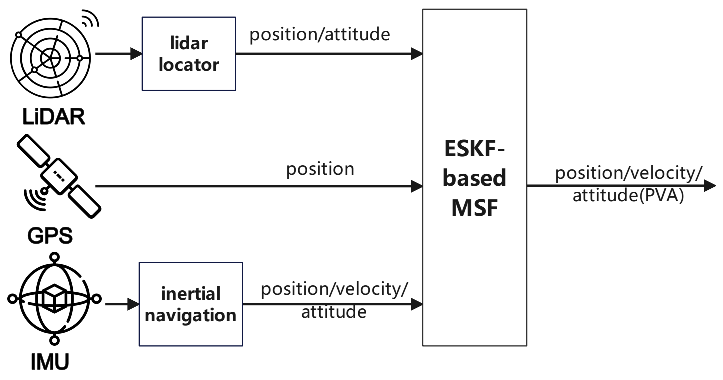


Figure 1: Multi-Sensor Fusion

Prediction step:. In the prediction step, the nominal states are propagated using inertial measurements to obtain a prior estimate of the pose, denoted as:

$$\check{X}_{k|k-1} = f(\hat{X}_{k-1}, u_{k-1}) \tag{1}$$

where $\hat{X}_{k|k-1}$ is the predicted state at time step k, $f(\cdot)$ represents the state transition function, and u_{k-1} is the control input. The error states are estimated using the prediction equations to derive the predicted state estimates and covariance estimates:

$$\delta\check{x}_{k|k-1} = F_{k-1}\delta\hat{x}_{k-1} + B_{k-1}w_k \tag{2}$$

$$\check{P}_{k|k-1} = F_{k-1}\hat{P}_{k-1}F_{k-1}^{\mathrm{T}} + B_{k-1}Q_kB_{k-1}^{T} \tag{3}$$

where $\delta\check{x}_{k|k-1}$ is the predicted error state at time step k, $\delta\hat{x}_{k-1}$ is the error state estimate at the previous time step $k-1$, w_k is the process noise, $\check{P}_{k|k-1}$ is the predicted covariance matrix, F_{k-1} is the state transition matrix, Q_k is the process noise covariance.

Update step:. During the update step, the error states first calculate the gain matrix based on the observations, then update error state estimate and covariance matrix:

$$K_k = \check{P}_{k|k-1}G_k^{\mathrm{T}}\left(G_k\check{P}_{k|k-1}G_k^{\mathrm{T}} + C_kR_kC_k^{\mathrm{T}}\right)^{-1} \tag{4}$$

$$\hat{P}_k = (I - K_kG_k)\,\check{P}_{k|k-1} \tag{5}$$

$$\delta\hat{x}_k = \delta\check{x}_{k|k-1} + K_k\left(z_k - G_k\delta\check{x}_{k|k-1}\right) \tag{6}$$

where K_k is the gain matrix, $\check{P}_{k|k-1}$ is the predicted covariance matrix from the prediction step, G_k is the observation matrix, R_k is the observation noise covariance matrix, $\hat{P}_k$ is the updated covariance matrix, $\delta\hat{x}_k$ is the updated error state estimate, $\delta\check{x}_{k|k-1}$ is the predicted error state from the prediction step, and z_k is the observation measurement at time step k.

Finally, the error states are used to correct the nominal states, resulting in the current state estimation.

$$\hat{X}_k = \check{X}_{k|k-1} + \delta\hat{x}_k, \tag{7}$$

A detailed explanation of the ESKF process is provided in Appendix A, along with a comprehensive notation table summarizing all symbols and their meanings used throughout this paper, which can be found in Appendix C.

2.2. Threat Model

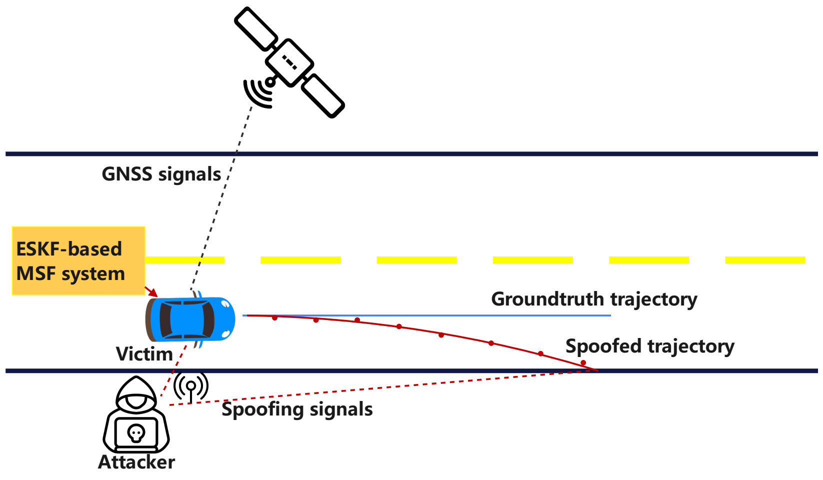


Figure 2: Attack Scenario

Scenario. We consider a sophisticated attack scenario in which a victim's autonomous driving system relies heavily on an ESKF-based fusion localization module for accurate state estimation. In this scenario, an attacker strategically injects spoofing signals into the sensor data, aiming to disrupt the output of the fusion model and cause a significant deviation in state estimation. This deviation could be large enough to compromise the vehicle's safe operation, potentially leading to critical failures or accidents, as illustrated in Figure 2. Such an attack poses a serious risk, especially in complex driving environments where precise localization is crucial.

Attacker Capabilities. In this scenario, it is assumed that the attacker has access to an autonomous driving system identical to that of the victim. This allows the attacker to reverse-engineer the Kalman gain matrix used in the victim's ESKF-based MSF localization system. Additionally, the attacker is equipped with a high-power GPS signal spoofer capable of overwhelming legitimate GPS signals. By utilizing this spoofer, the attacker can inject false data into the GPS receiver, contaminating the sensor measurements before they are processed by the fusion model. This capability allows the attacker to precisely control the spoofing signals, maximizing the potential disruption to the vehicle's state estimation process.

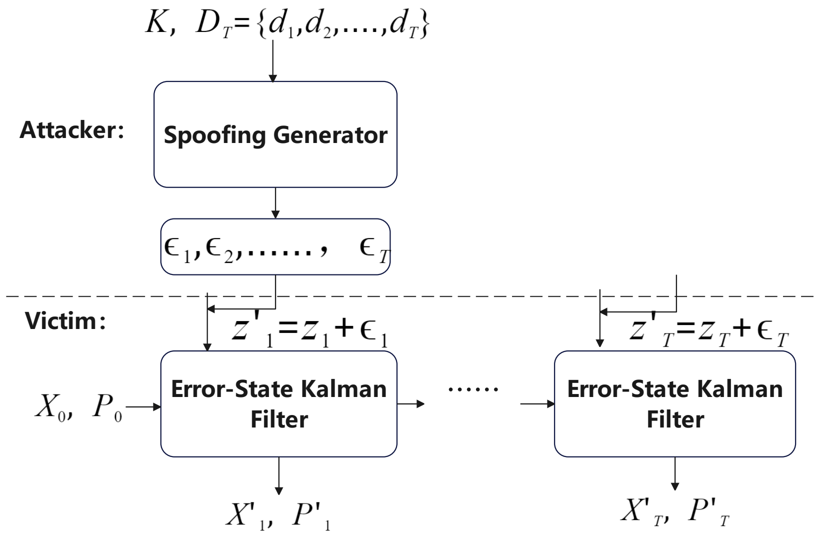

$K,\ D_T=\{d_1,d_2,\ldots,d_T\}$
Attacker:
Spoofing Generator
$\epsilon_1,\epsilon_2,\ldots,\ \epsilon_T$
Victim:
$z'_1=z_1+\epsilon_1$
$z'_T=z_T+\epsilon_T$
$X_0,\ P_0$
Error-State Kalman Filter
......
Error-State Kalman Filter
$X'_1,\ P'_1$
$X'_T,\ P'_T$

Figure 3: Injection Model

Attack Goal. The primary objective of the attacker is to manipulate the vehicle's state estimation by injecting a sequence of spoofing signals $\epsilon_1, \epsilon_2, \ldots, \epsilon_k$ at successive time steps. Let z_k represent the observation data at time step k, and z'_k represent the spoofed data at the same time step. The spoofing signal ϵ_k is injected such that $z'_k = z_k + \epsilon_k$, as shown in Figure 3. This manipulation results in a deviation ΔX_k between the state estimate before spoofing X_k and the state estimate after spoofing X'_k, where $\Delta X_k = |X_k - X'_k|$. The magnitude of ΔX_k is critical, as a sufficiently large deviation can lead to erroneous control decisions, potentially resulting in a collision or other safety-critical events. The attacker aims to ensure that $\Delta X_k \geq d_k$ for one or more time steps k within the time horizon T, where D_T represents the set of target deviations $\{d_1, d_2, \ldots, d_T\}$. Achieving this condition would signify a successful attack, as it compromises the vehicle's ability to operate safely.

3. Security Analysis of MSF Model Under Injection Attack

3.1. Preliminary Setup

Before we proceed with further analysis, it is necessary to clarify the key assumptions, including the subject of our analysis, the conditions under which the analysis is conducted, and the definitions of the symbols that will be used throughout the analysis.

Key Assumptions. We conduct a security analysis of the injection model mentioned in the previous section, focusing specifically on the three sensors: GPS, LiDAR, and IMU. Among these, the IMU has the highest frequency, followed by LiDAR, and GPS has the lowest frequency. This setup aligns with the sensor frequency configurations commonly used in commercial autonomous driving systems, such as Apollo. The injection process is carried out under the condition that the vehicle is already operating in a steady state.

Core Definitions. Assume that the IMU update period is T_{imu}, the GPS update period is T_{gps}, and the LiDAR update period is T_{lidar}. The initial state estimate is $(\check{X}_1, \check{P}_1)$, the gain matrix is K_i for $i = 1, 2, \ldots, k$, where $k \in \mathbb{N}^+$ and the state deviation is given by $\Delta\hat{X}_k = \Delta\check{X}_k - \Delta\delta\hat{x}_k$. In this analysis, variables with a hat symbol (e.g., $\hat{X}$) represent posterior information, while those with a check symbol (e.g., $\check{X}$) represent prior information.

Single Cycle: A single cycle is defined as the period from the end of the 1st GPS spoofing to the end of the 2nd GPS spoofing. During this cycle, there are a total of $m_1 = T_{gps}/T_{imu}$ IMU prediction updates and $m_2 = T_{gps}/T_{lidar}$ LiDAR updates. Between two consecutive LiDAR updates, there are $m_3 = T_{lidar}/T_{imu}$ IMU prediction updates.

N-Cycle: An N-cycle is defined as the cumulative period covering N consecutive single cycles. During an N-cycle, the process repeats N times the number of IMU prediction updates, LiDAR updates, and GPS spoofings that occur in a single cycle. The total state deviation over the N-cycle is analyzed by summing the deviations $\Delta\hat{X}_k$ over each individual cycle.

3.2. Analytical Framework

Our analysis approach begins by deriving the results for a single cycle and then extends these results to N-cycles, as shown in Figure 4.

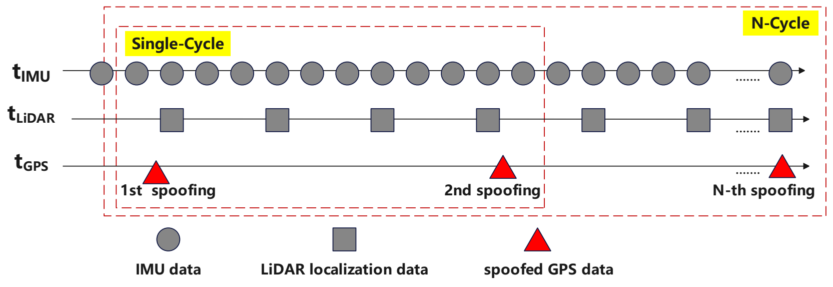


Figure 4: Sensor data processing

3.2.1. Single-Cycle State Deviation Analysis

1st GPS Injection. Since all other state variables remain consistent before and after the spoofing under the initial conditions, the state deviation is therefore attributed solely to the influence of the GPS injection update:

$$\begin{aligned}\Delta\hat{X}_1 &= 0 - \Delta\delta\hat{x}_1 \\ &= -K_1\epsilon_1\end{aligned} \tag{8}$$

From 1st GPS Injection to 1st LiDAR Update. First, consider the IMU inertial computation. Due to the assumption of a steady-state system, there are a total of $m_3 = T_{lidar}/T_{imu}$ iterations:

$$\check{X}_{1+m_3} = \hat{X}_1 + \begin{bmatrix} \hat{v}\cdot T_{imu}\cdot m_3 \\ 0_{12\times1} \end{bmatrix} \tag{9}$$

where $\hat{v}$ is the velocity of the state. Since the increment is fixed, there is no difference before and after injection, so the state deviation is:

$$\Delta\check{X}_{1+m_3} = \Delta\hat{X}_1 \tag{10}$$

Next, we calculate the error and update the state based on LiDAR observations. Since it is in steady-state, F_{k+1} is considered unchanged basically, we name it F, so are the B and w. Starting with the error calculation, based on the derivation from a single observation, the error is a fixed value:

$$\begin{aligned}\delta\check{x}_{1+m_3} &= F\delta\hat{x}_{m_3} + Bw \\ &= F^2\delta\hat{x}_{m_3-1} + F\cdot Bw + Bw \\ &= \dots \\ &= F^{m_3}\delta\hat{x}_1 + (F^{m_3-1} + \dots + F^1 + I)\cdot Bw \\ &= (F^{m_3-1} + \dots + F^1 + I)\cdot Bw\end{aligned} \tag{11}$$

Since the error is independent of the observations, it cancels out in the state deviation calculation before and after injection.

For the LiDAR observation update:

$$\delta\hat{x}_{1+m_3} = \delta\check{x}_{1+m_3} + K_{1+m_3}(z_{L1} - G_L\delta\check{x}_{1+m_3}) \tag{12}$$

where z_{L1} is the LiDAR's 1st measurement. Since the LiDAR observation does not contain the spoofing increment and is the same before and after spoofing:

$$\Delta\delta\hat{x}_{1+m_3} = K_{1+m_3}\Delta\check{X}_{1+m_3} \tag{13}$$

Finally, we calculate the state deviation. Based on the results from the previous Eq. (10)(13), we have:

$$\begin{aligned}\Delta\hat{X}_{1+m_3} &= \Delta\check{X}_{1+m_3} - \Delta\delta\hat{x}_{1+m_3} \\ &= (I - K_{1+m_3})\Delta\check{X}_{1+m_3} \\ &= (I - K_{1+m_3})\Delta\hat{X}_1\end{aligned} \tag{14}$$

From 1st LiDAR Update to m_2-th LiDAR Update. Based on the state deviation iterative equation provided in previous paragraph, we have:

$$\Delta\hat{X}_{1+km_3} = (I - K_{1+km_3})\Delta\hat{X}_{1+(k-1)m_3} \tag{15}$$

where k is the LiDAR's k-th measurement. Thus, after $m_2 = T_{gps}/T_{lidar}$ iterations, with Eq. (10) we obtain:

$$\begin{aligned}\Delta\hat{X}_{1+m_2\cdot m_3} &= \prod_{j=1}^{m_2}(I-K_{1+jm_3})\Delta\hat{X}_1 \\ &= -\prod_{j=1}^{m_2}(I-K_{1+jm_3})\cdot K_1\epsilon_1\end{aligned} \tag{16}$$

Note that the gain matrix from $1+m_3$ to $1+m_2\cdot m_3$ are all LiDAR update gain matrixs, while K_1 is the GPS observation update gain matrix.

2nd GPS Injection. Above, we reveal that:

- The state deviation caused by IMU inertial computation is independent of the number of iterations and is only related to the state deviation induced by the previous observation update.
- The error does not affect the state deviation, nor is it related to the number of iterations. It is only related to the state deviation induced by the previous observation update.

Therefore, even if the 2nd GPS spoofing does not coincide with the most recent LiDAR observation update (which is more realistic), there will be a brief period of IMU inertial computation and error calculation between the observation updates. However, based on points 1 and 2 above, we can conclude that these computations do not affect the state deviation, which means we only need to calculate the state deviation caused by the LiDAR updates between the two GPS injections.

To analyze the state deviation, we first need to consider the error update. The error update can be expressed as:

$$\delta\hat{x}'_{1+m_1} = \delta\check{x}_{1+m_1} + K_{1+m_1}(z'_{G2} - H_G\delta\check{x}_{1+m_1}) \tag{17}$$

where the z_{G2} is the GPS's second measurement. Given that $z'_{G2} = \check{X}_{1+m_1} - (z_{G2}+\epsilon_2)$, this equation simplifies to:

$$\delta\hat{x}'_{1+m_1} = [(I-K_{1+m_1}H_G)\delta\check{x}_{1+m_1} - K_{1+m_1}z_{G2}] + K_{1+m_1}\cdot(\check{X}_{1+m_1}-\epsilon_2) \tag{18}$$

Thus, the error deviation can be described by the following equation:

$$\Delta\delta\hat{x}_{1+m_1} = K_{1+m_1}\Delta\check{X}_{1+m_1} - K_{1+m_1}\epsilon_2 \tag{19}$$

Next, we consider the state deviation caused by the 2nd GPS injection. The state deviation after the 2nd GPS spoofing can be written as:

$$\begin{aligned}\Delta\hat{X}_{1+m_1} &= \Delta\check{X}_{1+m_1} - \Delta\delta\hat{x}_{1+m_1} \\ &= (I - K_{1+m_1})\Delta\check{X}_{1+m_1} + K_{1+m_1}\epsilon_2\end{aligned} \tag{20}$$

From the previous Eq. (16), we know that:

$$\begin{aligned}\Delta\check{X}_{1+m_1} &= \Delta\hat{X}_{1+m_2\cdot m_3} \\ &= -\prod_{j=1}^{m_2}(I - K_{1+jm_3})\cdot K_1\epsilon_1\end{aligned} \tag{21}$$

Substituting this into the equation for the state deviation, we obtain:

$$\begin{aligned}\Delta\hat{X}_{1+m_1} &= (I - K_{1+m_1})\Delta\check{X}_{1+m_1} + K_{1+m_1}\epsilon_2 \\ &= (I - K_{1+m_1})\cdot\left[\prod_{j=1}^{m_2}(I - K_{1+jm_3})\cdot\Delta\hat{X}_1\right] + K_{1+m_1}\epsilon_2 \\ &= (I - K_{1+m_1})\cdot\left[-\prod_{j=1}^{m_2}(I - K_{1+jm_3})\cdot K_1\epsilon_1\right] + K_{1+m_1}\epsilon_2\end{aligned} \tag{22}$$

It is important to note that the gain matrix K with a subscript containing m_1 is associated with the GPS observation updates, while those with a subscript containing m_3 correspond to the LiDAR observation updates.

3.2.2. Extending State Deviation to N-Cycle Injections

This section extends the analysis to state deviation across multiple cycles of injection, generalizing from single-cycle scenarios to N-cycle scenarios.

As shown in the previous section, the state deviation iterative equation is:

$$\begin{aligned}\Delta\hat{X}_{1+km_1} &= (I - K_{1+km_1})\Delta\check{X}_{1+km_1} + K_{1+km_1}\epsilon_k \\ &= (I - K_{1+km_1})\cdot\left[\prod_{j=1}^{m_2}(I - K_{1+jm_3})\cdot\Delta\hat{X}_{1+(k-1)m_1}\right] + K_{1+km_1}\epsilon_k\end{aligned} \tag{23}$$

For the convenience of notation, let all GPS gain matrices be denoted by K_g and all LiDAR gain matrices by K_l.

Thus, we have:

$$\begin{aligned}\Delta\hat{X}_{1+km_1} &= (I-K_g)\cdot\left[\prod_{j=1}^{m_2}(I-K_l)\cdot\Delta\hat{X}_{1+(k-1)m_1}\right]+K_g\epsilon_k \\ &= (I-K_g)\cdot(I-K_l)^{m_2}\cdot\Delta\hat{X}_{1+(k-1)m_1}+K_g\epsilon_k\end{aligned} \tag{24}$$

Let $A=(I-K_g)\cdot(I-K_l)^{m_2}$, then:

$$\begin{aligned}\Delta\hat{X}_{1+km_1} &= A\cdot\Delta\hat{X}_{1+(k-1)m_1}+K_g\epsilon_k \\ &= \dots \\ &= A^{k-1}\Delta X_1+A^{k-2}K_g\epsilon_2+\cdots+A^1K_g\epsilon_{k-1}+K_g\epsilon_k \\ &= -A^{k-1}K_g\epsilon_1+\sum_{i=0}^{k-2}A^iK_g\epsilon_{k-i}\end{aligned} \tag{25}$$

It can be observed that, since the matrix coefficients are constants, the deviation is a linear combination of the injection sequences, making it a concise equation for further analysis.

4. Optimized Injection Attack Methodology

In this section, we will demonstrate how to leverage the findings from the previous section and propose a constraint-based optimization method for injection attacks.

4.1. Fusion Strategy

While the Error-State Kalman Filter (ESKF) fusion model is widely adopted, there are different kinds of practical fusion strategies for implementing the fusion process. In this section, we conduct a quantitative analysis based on one of the most commonly used fusion strategies, which is also adopted by the commercial localization system, Apollo_Shenlan[48].

Consider the state vector:

$$X=\begin{bmatrix}p\\v\\\theta\\b_a\\b_g\end{bmatrix} \tag{26}$$

where p represents the 3D position, v represents the 3D velocity, θ represents the 3D orientation, and b_a and b_g are the accelerometer and gyroscope biases.

In this analysis, LiDAR provides the position $p_{\text{lidar}} = [x, y, z]^T$ and orientation $\theta_{\text{lidar}} = [\text{roll}, \text{pitch}, \text{yaw}]^T$ information, while GPS provides the position $p_{\text{gps}} = [x, y, z]^T$ information for state estimation updates. Therefore, each update observation z is represented as:

$$z = \begin{bmatrix} p_{lidar} \\ \theta_{lidar} \\ p_{gps} \end{bmatrix} \tag{27}$$

Since the actual deviation generally considers lateral or longitudinal shifts in the plane, the injection only affects the position in the x and y directions. Assuming the injection is perpendicular to the vehicle's direction of travel, the k-th injection vector is given by $\epsilon_k = [\epsilon_k^{gps}\sin(\theta_z), -\epsilon_k^{gps}\cos(\theta_z), 0]^T$, where θ_z is the rotation angle of z-direction. Thus, the update observation z under injection is:

$$z' = \begin{bmatrix} p_{lidar} \\ \theta_{lidar} \\ p_{gps} + \epsilon \end{bmatrix} \tag{28}$$

Next, we will quantitatively derive and analyze the deviation according to the filtering steps of the ESKF.

Inertial Prediction. Consider the impact of velocities v_x and v_y on the position state variables, with all state variables defined in the N-frame. Given that the vehicle is in a steady operating state, changes in orientation and velocity can be approximately neglected over short periods of time. Thus, we have:

$$\check{X}_{k+1} = \hat{X}_k + \begin{bmatrix} v_x \cdot T \\ v_y \cdot T \\ 0_{13\times1} \end{bmatrix} \tag{29}$$

$$\Delta\check{X}_{k+1} = \Delta\hat{X}_k + \begin{bmatrix} \Delta\hat{X}_{k(41)} \cdot T \\ \Delta\hat{X}_{k(51)} \cdot T \\ 0_{13\times1} \end{bmatrix} \tag{30}$$

where the subscript notation such as $\hat{X}_{k(41)}$ refers to the element in the 4th row and 1st column of the state vector $\hat{X}$, T is the time interval before the next observation update.

Error Prediction Iterative Calculation. The error prediction is calculated as $\delta\check{x}_k = F\delta\check{x}_{k-1} + B\omega$, as shown in Eq. (11). Although F and B are time-varying matrices, their impact is considered negligible for the overall accuracy. This is because the initial error is set to zero in the program and remains unchanged throughout the process.

Error Update Calculation. The observation matrix G and the update observation error ϵ are given by:

$$G = \begin{bmatrix} I_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_3 & 0 & 0 \\ I_3 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \epsilon = \begin{bmatrix} 0_{6\times1} \\ \epsilon^{gps}\sin(\theta_z) \\ -\epsilon^{gps}\cos(\theta_z) \\ 0 \end{bmatrix} \tag{31}$$

where ϵ^{gps} is the GPS measurement injection error.

The error update is calculated as:

$$\begin{aligned} \delta\hat{x}_{k+1} &= \delta\check{x}_{k+1} + K_{15\times9}\left[\check{X}^9_{k+1} - (z_{k+1} + \epsilon_{k+1}) - G_{9\times15}\delta\check{x}_{k+1}\right] \\ &= \delta\check{x}_{k+1} - K_{15\times9}(\epsilon_{k+1} + G_{9\times15}\delta\check{x}_{k+1}) + K_{15\times9}(\check{X}^9_{k+1} - z_{k+1}) \end{aligned} \tag{32}$$

where the subscripts 15×9 and superscripts 9 indicate the dimensions of the matrices and vectors, respectively. These notations are used to facilitate the analysis of the equations.

The error deviation is then:

$$\Delta\delta\hat{x}_{k+1} = K_{15\times9}(\Delta\check{X}^9_{k+1} - \epsilon_{k+1}) \tag{33}$$

Error Correction. The corrected state is calculated as:

$$\hat{X}_{k+1} = \check{X}^{15}_{k+1} - \delta\hat{x}^9_{k+1} \tag{34}$$

The state deviation is then:

$$\Delta\hat{X}_{k+1} = \Delta\check{X}^{15}_{k+1} - \Delta\delta\hat{x}^9_{k+1} \tag{35}$$

Error Iterative Equation. Substituting the results from the Inertial Prediction and Error Update Calculation into the Error Correction, we obtain:

$$\begin{aligned} \Delta\hat{X}_{k+1} &= \Delta\check{X}^{15}_{k+1} - \Delta\delta\hat{x}^9_{k+1} \\ &= \Delta\hat{X}_k + \begin{bmatrix} \Delta\hat{X}_{k(41)} \cdot T \\ \Delta\hat{X}_{k(51)} \cdot T \\ 0_{13\times1} \end{bmatrix} - K_{15\times9}(\Delta\check{X}^9_{k+1} - \epsilon_{k+1}) \end{aligned} \tag{36}$$

where $\Delta \check{X}^{9}_{k+1}$ also applies to Eq. (30) but is somewhat special, which will be discussed in the next section.

The derivation above is based on the ESKF algorithm analysis in Section 3.2. The process depends on the fusion strategy, such as the Apollo_Shenlan[48] used here. For different fusion strategies, only the observation variables change, while the analysis remains the same and is not limited to a specific strategy.

4.2. Constrained Optimization

Based on the fusion strategy and analysis results presented in Section 4.1, a linear relationship can be established between the error deviation and the injection magnitude. For the error deviation to pose a significant threat to the vehicle's safety, it must reach a certain threshold. Simultaneously, the energy required to construct the injection should be minimized to avoid detection and maintain a covert attack. To address these conditions, we can formulate constraint conditions to describe the deviation magnitude and construct an objective function aimed at minimizing the injection energy. By applying constraint optimization methods to the established error deviation relationship, the optimized injection magnitude can be obtained. The problem is formulated as follows:

Problem. Consider a target with the ESKF model (Section 2.1) and a measurement injection model (Section 2.2). Assume the target knows the gain matrix K. Find a sequence of injection signal inputs $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_T\}$, that achieves a desired separation d_t between X'_t and X_t at step t. Such that,

$$\begin{array}{ll} \textit{minimize} & \sum_{t=1}^{T} \gamma_t \cdot \|\epsilon_t\|_p^p \\ \textit{subject to,} & \\ & \|X_t - X'_t\|_p^p \geq d_t^p, \quad \forall t \end{array} \tag{37}$$

where $\gamma_t \in \mathbb{R}^+$ is a weighting parameter, and $T \in \mathbb{Z}^+$ is the optimization horizon.

State Error Deviation. Consider the injection signal $\epsilon_k = [\epsilon_k \sin(\theta_z), -\epsilon_k \cos(\theta_z), 0]^T$. Based on Eq. 36, the state error deviation variables can be recursively determined as follows. Detailed derivations are provided in the Appendix B.

$$\begin{aligned} \Delta \hat{X}_{k+1(11)} = {} & (1 - K_{11} - K_{17}) \Delta \hat{X}_{k(11)} + (1 - K_{11}) T \cdot \Delta \hat{X}_{k(41)} - \\ & K_{17} \epsilon_{k+1} \sin(\theta_z) \end{aligned} \tag{38}$$

Algorithm 1 Optimized Injection Attack Algorithm

```
1: Input: $K$: Kalman gain matrix; $T$: Time step; $\theta_z$: yaw angle; $D_t$: Desired deviation vector at time $t$
2: Output: Optimal injection sequences $\epsilon^*$, state deviations $\Delta\hat{X}_{k(11)}$, $\Delta\hat{X}_{k(21)}$, $\Delta\hat{X}_{k(41)}$, $\Delta\hat{X}_{k(51)}$, and minimum energy $\sum_{t=1}^{k} \|\epsilon_t\|_2$
3: Initialize deviation values to zero: $\Delta\hat{X}_{0(11)} \leftarrow 0$, $\Delta\hat{X}_{0(21)} \leftarrow 0$, $\Delta\hat{X}_{0(41)} \leftarrow 0$, $\Delta\hat{X}_{0(51)} \leftarrow 0$
4: Extract Kalman gains from $K$:
5: $K_{11} \leftarrow K[1,1]$, $K_{17} \leftarrow K[1,7]$
6: $K_{22} \leftarrow K[2,2]$, $K_{28} \leftarrow K[2,8]$
7: $K_{41} \leftarrow K[4,1]$, $K_{47} \leftarrow K[4,7]$, $K_{45} \leftarrow K[4,5]$
8: $K_{52} \leftarrow K[5,2]$, $K_{58} \leftarrow K[5,8]$, $K_{54} \leftarrow K[5,4]$
9: for $i = 0$ to $k-1$ do
10:     Compute the next state deviations:
11:     $\Delta\hat{X}_{i+1(11)} \leftarrow (1 - K_{11} - K_{17})\Delta\hat{X}_{i(11)} + (1 - K_{11})T \cdot \Delta\hat{X}_{i(41)} - K_{17}\epsilon_{i+1}\sin(\theta_z)$
12:     $\Delta\hat{X}_{i+1(21)} \leftarrow (1 - K_{22} - K_{28})\Delta\hat{X}_{i(21)} + (1 - K_{22})T \cdot \Delta\hat{X}_{i(51)} + K_{28}\epsilon_{i+1}\cos(\theta_z)$
13:     $\Delta\hat{X}_{i+1(41)} \leftarrow (-K_{41} - K_{47})\Delta\hat{X}_{i(11)} + (1 - K_{41}T - K_{44}) \cdot \Delta\hat{X}_{i(41)} - K_{45} \cdot \Delta\hat{X}_{i(51)} - K_{47}\epsilon_{i+1}\sin(\theta_z)$
14:     $\Delta\hat{X}_{i+1(51)} \leftarrow (-K_{52} - K_{58})\Delta\hat{X}_{i(21)} + (1 - K_{52}T - K_{55}) \cdot \Delta\hat{X}_{i(51)} - K_{54} \cdot \Delta\hat{X}_{i(41)} + K_{58}\epsilon_{i+1}\cos(\theta_z)$
15:     Apply constraint:
16:     if $\| \begin{bmatrix} \Delta\hat{X}_{i+1(11)} \\ \Delta\hat{X}_{i+1(21)} \end{bmatrix} \| \geq D_t$ then
17:         Continue with the next step
18:     else
19:         Adjust $\epsilon_{i+1}$ to satisfy the constraint
20:     end if
21: end for
        Obtain $\epsilon^* = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_k\}$
22: Return the optimal injection sequence $\epsilon^*$, the final state deviations $\Delta\hat{X}_{k(11)}$, $\Delta\hat{X}_{k(21)}$, $\Delta\hat{X}_{k(41)}$, $\Delta\hat{X}_{k(51)}$, and minimum energy $\sum_{t=1}^{k} \|\epsilon_t\|_2$
```

$$\Delta \hat{X}_{k+1(21)} = (1 - K_{22} - K_{28})\Delta \hat{X}_{k(21)} + (1 - K_{22})T \cdot \Delta \hat{X}_{k(51)} + K_{28}\epsilon_{k+1}\cos(\theta_z) \tag{39}$$

$$\Delta \hat{X}_{k+1(41)} = (-K_{41} - K_{47})\Delta \hat{X}_{k(11)} + (1 - K_{41}T - K_{44}) \cdot \Delta \hat{X}_{k(41)} - K_{45} \cdot \Delta \hat{X}_{k(51)} - K_{47}\epsilon_{k+1}\sin(\theta_z) \tag{40}$$

$$\Delta \hat{X}_{k+1(51)} = (-K_{52} - K_{58})\Delta \hat{X}_{k(21)} + (1 - K_{52}T - K_{55}) \cdot \Delta \hat{X}_{k(51)} - K_{54} \cdot \Delta \hat{X}_{k(41)} + K_{58}\epsilon_{k+1}\cos(\theta_z) \tag{41}$$

Optimized Injection Attack Algorithm. Consider the injection signal $\epsilon = [\epsilon_k \sin(\theta_z), -\epsilon_k \cos(\theta_z), 0]^T$, with the objective function defined as the L2-norm of the injection magnitude when $p = 2$, which is the energy cacluation. Setting the weight params γ_t to 1, we derive the optimized injection attack algorithm 1.

5. Experiments

In this section, we first introduce the experimental setup, including the simulation and real-world datasets. Following that, we outline the experimental design, beginning with the validation of our model's accuracy, followed by comparative experiments against the current state-of-the-art method, FusionRipper, under the same deviation targets. Finally, we conduct end-to-end injection tests on an AI-powered autonomous driving system to demonstrate the effectiveness of our approach.

5.1. Setup

Environment. The experiments were conducted on a system with the following specifications: Ubuntu 18.04 LTS operating system, an Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz, a GeForce GTX 1080 Ti GPU, and 32GB of memory. The autonomous driving system used in our experiments is Shenlan_MSF[48], which replaces the black-box Apollo_MSF[2] with a white-box approach. This modification allows for transparent analysis, making it more trustworthy and providing the high precision typical of commercial systems.

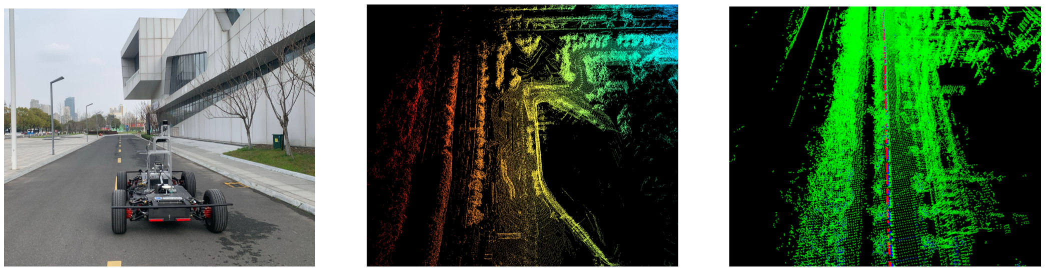

(a) Vehicle (b) PointCloud Map (c) Sensors Trajectory

Figure 5: Real-world data collection environment

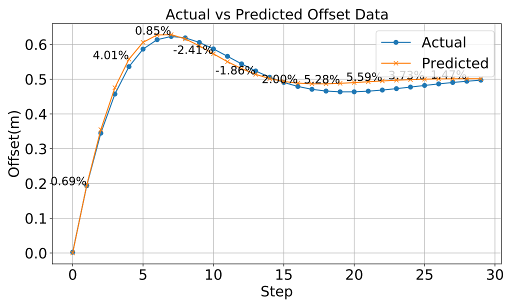
Datasets. For the datasets, we utilized the GNSS-INS-SIM[49] simulation library to generate IMU and GPS simulation data. The specific trajectory used was a constant-speed straight-line trajectory with 5.0 m/s velocity and 90° rotation of z-direction. Since this simulation library does not provide LiDAR data, we converted ground truth values into LiDAR positioning data, theoretically enhancing positioning accuracy and further validating the effectiveness of the injection attack method. Real-world data were collected using an actual vehicle equipped with the autonomous driving system on a straight road segment. The system recorded environmental data and system state information to create the real-world dataset.

5.2. Verification of Model Accuracy

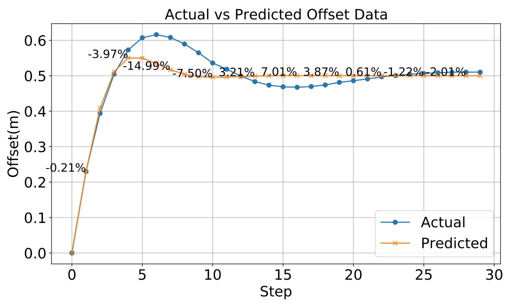
To verify the model's accuracy, we compare the predicted offset with the actual observed offset and calculate the percentage error to assess the model's fit, guiding future injection attacks. Since our approach differs from FusionRipper[14], only testing our method's accuracy is necessary for optimizing the attack, without needing a comparison with FusionRipper.

5.2.1. Constant Injection Attack

In the constant injection verification, we set the injection amount to $\epsilon = 1$ m and tested the model on both simulated and real datasets. During the tests, the actual offset was used as a baseline to calculate the percentage error of the predicted offset. As shown in Figure 6, in 30 injection tests on the simulated dataset, the error between the actual and predicted offsets was kept within 6%. On the real dataset, however, the maximum error reached 15%, which is understandable given the additional noise and other influencing factors present in real-world data. Notably, the error was controlled within

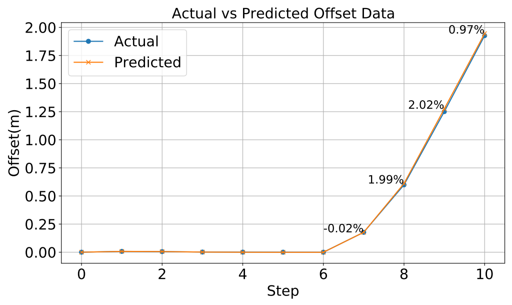


(a) sim dataset

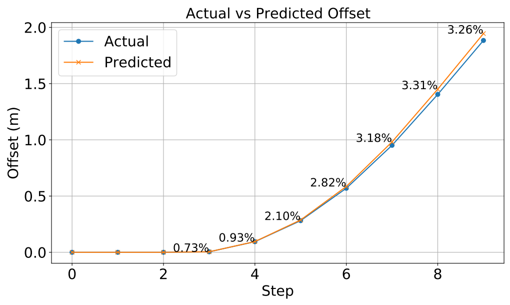


(b) real dataset

Figure 6: Constant injection test



(a) sim dataset



(b) real dataset

Figure 7: Incremental injection test

Condition	Type	Data Type	1	2	3	4	5	6	7	8	9	10
Constant	Sim	Actual Data	0.194	0.345	0.457	0.536	0.586	0.614	0.623	0.619	0.606	0.587
		Predicted Data	0.195	0.355	0.476	0.558	0.606	0.627	0.628	0.616	0.596	0.573
		Error Rate(%)	0.67	2.97	3.97	3.93	3.32	2.19	0.85	0.49	1.63	2.41
	Real	Actual Data	0.230	0.394	0.505	0.573	0.607	0.616	0.608	0.590	0.565	0.536
		Predicted Data	0.229	0.409	0.512	0.550	0.550	0.534	0.517	0.504	0.498	0.496
		Error Rate(%)	0.21	3.81	1.33	3.97	9.39	13.28	14.99	14.44	11.84	7.54
Increment	Sim	Actual Data	0.007	0.005	0.001	0.001	0.000	0.000	0.177	0.599	1.251	1.926
		Predicted Data	0.007	0.006	0.001	0.000	0.001	0.001	0.177	0.611	1.276	1.945
		Error Rate(%)	1.14	9.80	0.00	0.00	0.00	0.00	0.02	2.00	2.03	0.97
	Real	Actual Data	0.000	0.000	0.000	0.004	0.094	0.281	0.569	0.951	1.404	1.884
		Predicted Data	0.000	0.000	0.000	0.004	0.095	0.287	0.585	0.982	1.450	1.945
		Error Rate(%)	0.00	0.00	0.00	0.72	0.93	2.11	2.83	3.19	3.27	3.28

Table 1: Comparison of Simulated and Real Data under Constant and Increment Injection

4% for the first four injections and within 2% for the final offset, indicating that the accuracy before and after the injections is acceptable.

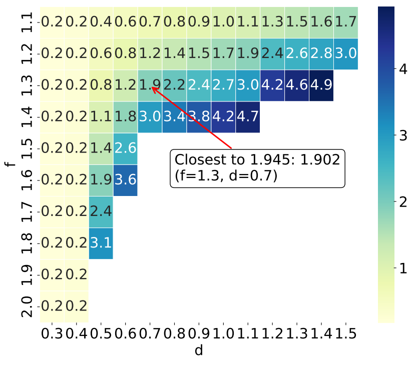
5.2.2. Incremental Injection Attack

For the incremental injection spoofing, we followed the thresholds set in the FusionRipper paper for different road scenarios, choosing the intermediate threshold of 1.945 meters as our target offset. We then applied the constrained optimization algorithm proposed in Section 4.2, using the `scipy` [50] library to solve for the optimal injection amounts, which provides a suite of optimization algorithms. As shown in Figure 7, in 10 injection tests on the simulated dataset, the error between the actual and predicted offsets was controlled to within 2%. On the real dataset, the error remained within 4%.

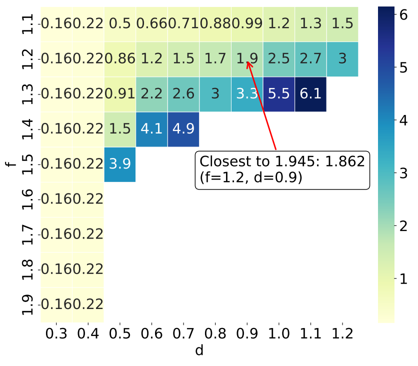
The results from both constant and incremental injection attack demonstrate that the model maintains a reasonable level of accuracy overall. Whether dealing with simple constant injections or optimized injections, the model achieves satisfactory performance.

5.3. Injection Attack Method Comparison

In this section, we will compare the required injection energy for achieving the same target offset between the current best attack method, FusionRipper, and our proposed method.



(a) sim dataset



(b) real dataset

Figure 8: Optimal params combination analysis

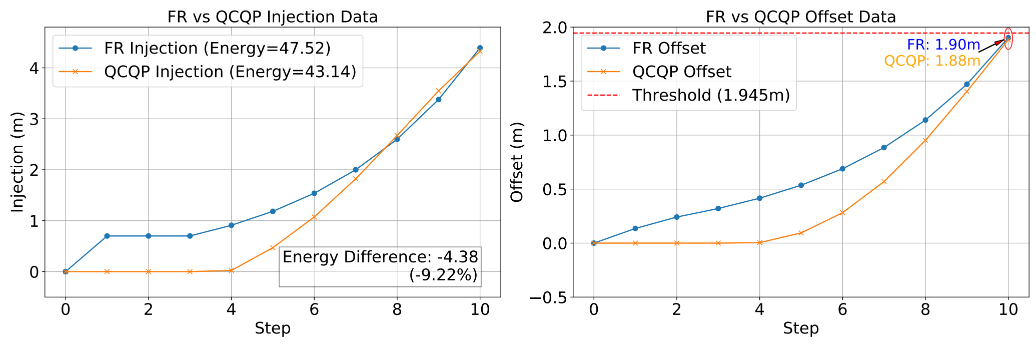
5.3.1. *Optimal Attack Params*

The FusionRipper method divides the entire attack process into two phases. In the first phase, a constant injection of d is applied until the offset exceeds a predetermined threshold of 0.295 meters, at which point the process transitions into the second phase. The second phase employs exponential deception, using the formula $d \cdot f^i$, where f is an exponential growth parameter. Each subsequent injection builds on the previous one, multiplying by f to amplify the resulting offset impact.

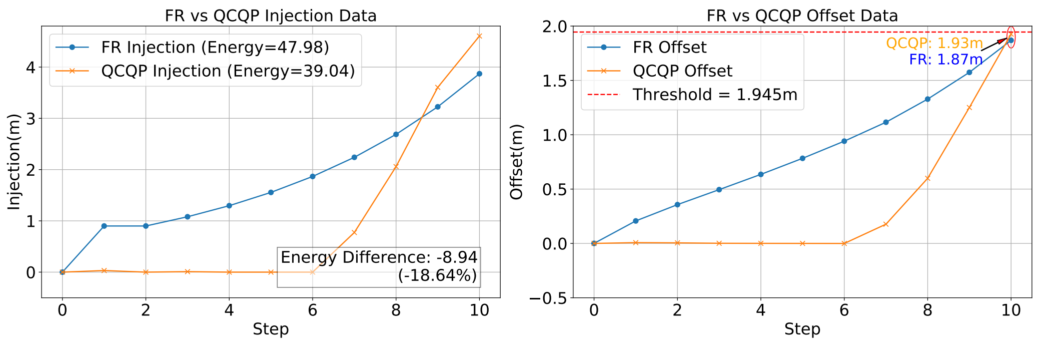
In our study, we also set the offset target to 1.945 meters. However, unlike the constraint optimization approach, which directly generates the optimal injection sequence, the FusionRipper method requires iterative testing to determine the optimal parameters d and f. Following the methodology outlined in the FusionRipper paper, we explored a range of parameters: for the simulated dataset, as shown in Figure 8, d values ranged from 0.3 to 1.4 and f values from 1.1 to 1.9, both with a step size of 0.1. Similarly, for the real dataset, d values ranged from 0.3 to 1.2 and f values from 1.1 to 1.9, also with a step size of 0.1. As illustrated in the figure, the optimal parameter combinations for the simulated and real datasets were found to be $d = 0.7, f = 1.3$ and $d = 0.9, f = 1.2$, respectively.

5.3.2. *Attack Methods Comparison*

As shown in Figure 9, both in the simulated and real datasets, the attack method implemented via `QCQP` consistently outperforms `FusionRipper`



(a) sim dataset



(b) real dataset

Figure 9: Comparison of FusionRipper and QCQP

in terms of achieving greater state estimation offsets with lower energy consumption. By setting the same offset target that is sufficient to cause the vehicle to deviate from the road, potentially leading to an accident, the `QCQP` approach reduces energy consumption by approximately 9% in the simulated dataset, while in the real dataset, the energy savings increase to around 18%. Specifically, the `QCQP` method achieves slightly larger offsets compared to `FusionRipper`, indicating its effectiveness in disrupting the vehicle's localization system.

Moreover, the ability of `QCQP` to achieve these results with reduced energy expenditure highlights its optimization capability, which is crucial for real-world applications where energy resources may be limited. The consistent performance across both datasets further validates the robustness of the `QCQP` method, demonstrating its applicability in varied scenarios, from controlled simulations to complex real-world environments.

5.4. Attack Effectiveness

In Sections 5.2 and 5.3, the experiments are designed to demonstrate the model's accuracy and the effectiveness of the attacks in detail, but with limited experimental scenarios. Therefore, we conduct a more extensive set of tests in this section. The scenario remains straight-line, as it aligns better with the steady-state attack assumptions. The main objective is to evaluate the model's accuracy and attack effects on simulation and real-world datasets at different velocities.

5.4.1. Experimental Setup

We collect sensor data at 2, 5, 7, and 10 m/s velocities for both the simulation and real-world datasets. The other experimental settings remain the same as before, and the specific details can be found in Section 5.1. Additionally, the experimental process for each dataset is the same as in Sections 5.2 and 5.3.

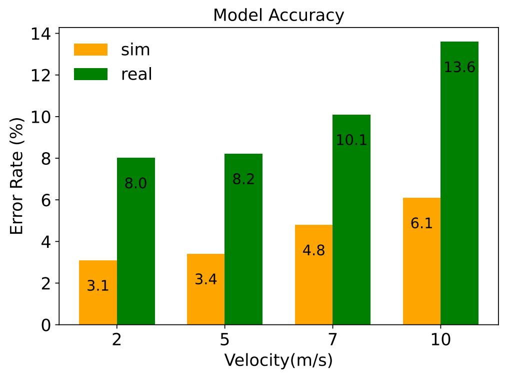
5.4.2. Results

As shown in Figure 10, (a) represents the average error percentage of the model at the first 10 points. It can be observed that at low speeds, such as 2 and 5 m/s, the accuracy difference is minimal. However, as the speed increases, such as at 10 m/s, the model's accuracy begins to decline. (b) shows the model's energy rate at different speeds. At low speeds, the energy difference is also slight, but as the speed increases, the energy rate starts to decrease. This indicates that the model's prediction performance deteriorates in high-speed scenarios, and the error increases. Nevertheless, it still maintains a certain level of performance.

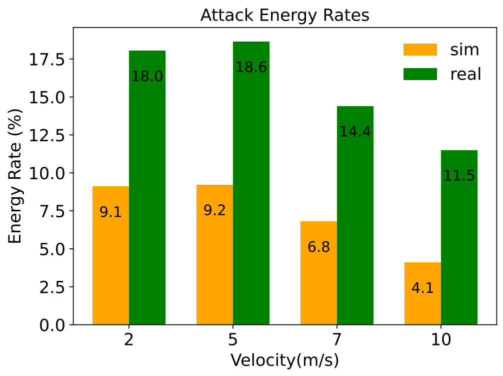
5.5. Attack Robustness

5.5.1. Inaccuracy Sources and Modeling

We build on the work[14] to model localization and uncertainty errors, which can be attributed to three main factors: localization error σ_1 due to the attacker's self-localization, distance measurement error σ_2 from LiDAR sensors, and GPS receiver error σ_3, representing the discrepancy between the intended and actual GPS positions. These errors are modeled using a combined normal distribution $N(0, 0.058^2)$, which represents the total position error σ_{pos}. Additionally, the measurement uncertainty σ_{var} is set to 0.008 based on real-world data.



(a) model accuracy



(b) attack energy

Figure 10: Comparison under varying velocity

5.5.2. Experimental Setup

We use the error distributions mentioned above to assess the robustness of our method, incorporating localization errors from [14]. For each GPS input, localization errors are sampled from $N(0, \sigma^2_{\text{pos}})$, with directions uniformly distributed between 0 and 360 degrees, while measurement uncertainties are drawn from $N(0, \sigma^2_{\text{var}})$. Additionally, we evaluate the impact of $2\times$ and $3\times$ error magnitudes to further test robustness.

5.5.3. Results

As shown in Figure 11, as the injected error increases, the energy rate of our method decreases relative to FusionRipper. Without error, the energy rates are 9.22% and 18.64%, respectively. As the error gradually increases, the energy rates decrease from 8.90% and 17.45% to 4.05% and 13.04%, respectively.

5.6. Ablation Study

Since our safety analysis derivation considers the impact of both position and velocity on vehicle state estimation, an ablation experiment is needed to further illustrate the importance of additional state variables in the modeling process. Here, we assume that the influence of velocity, particularly longitudinal and lateral velocity, is disregarded in the state estimation error model. We then compare the error between the actual vehicle position and the position estimated by the model.

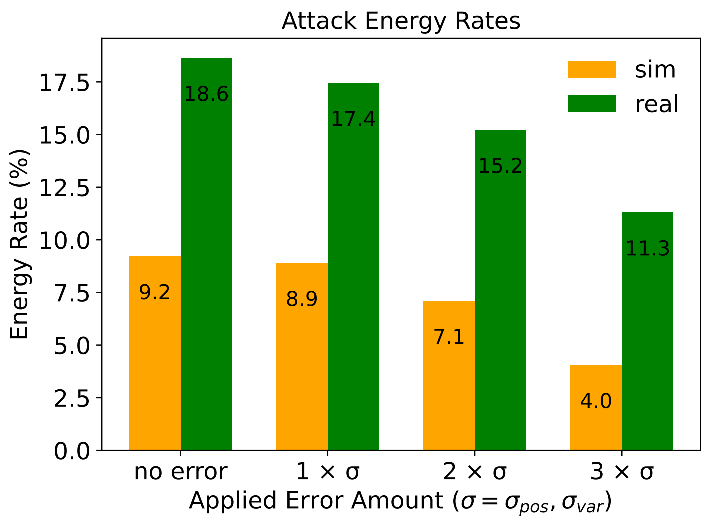


Figure 11: Attack energy rates under varying injection inaccuracies

5.6.1. Experimental Setup

This experiment uses a simulated dataset, with the experimental setup consistent with Sections 5.1 and 5.2. However, in the modeling process, the influence of the velocity variable is not considered. Following the same method as in Section 4.1, we obtain the following model:

$$\Delta\hat{X}_{k+1(11)} = (1 - K_{11} - K_{17})\Delta\hat{X}_{k(11)} - K_{17}\epsilon_{k+1}\sin(\theta_z) \tag{42}$$

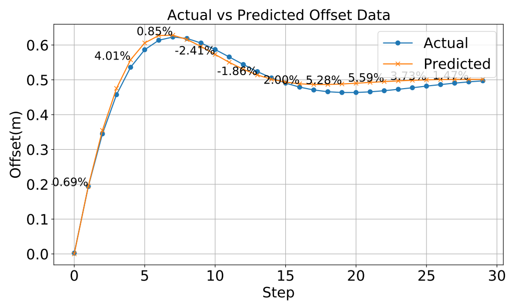
$$\Delta\hat{X}_{k+1(21)} = (1 - K_{22} - K_{28})\Delta\hat{X}_{k(21)} + K_{28}\epsilon_{k+1}\cos(\theta_z) \tag{43}$$

$$\Delta\hat{X}_{k+1(41)} = (-K_{41} - K_{47})\Delta\hat{X}_{k(11)} - K_{47}\epsilon_{k+1}\sin(\theta_z) \tag{44}$$

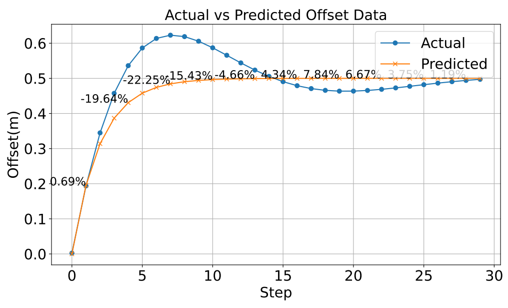
$$\Delta\hat{X}_{k+1(51)} = (-K_{52} - K_{58})\Delta\hat{X}_{k(21)} + K_{58}\epsilon_{k+1}\cos(\theta_z) \tag{45}$$

5.6.2. Comparison of Model Accuracy

As shown in Figure 12, we calculated the model's prediction error for the first 10 data points. The results indicate that the average prediction error is 3.41% when incorporating speed and 24.94% without it. This significant difference demonstrates the critical role of the speed variable in enhancing model completeness, which, in turn, influences the accuracy of state estimation predictions.



(a) model considering velocity

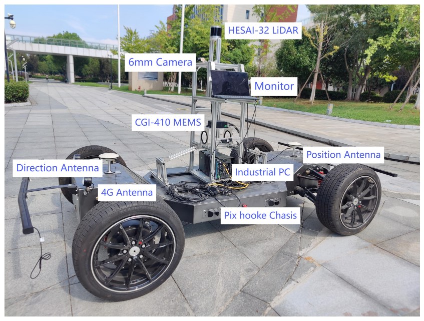


(b) model without considering velocity

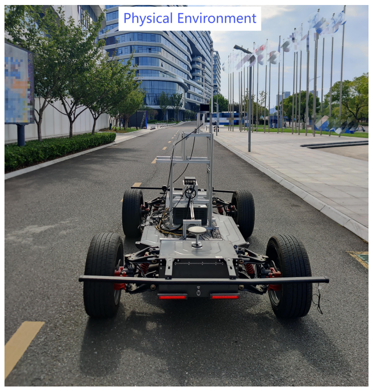
Figure 12: Comparison of different model accuracy

5.7. *End-to-end Experiment for AI-powered Autonomous Driving System*

In the experiments above, we did not consider the vehicle's actual planning, control modules, and dynamic feedback. In this section, we will conduct end-to-end injection attacks on a real vehicle equipped with an AI-powered autonomous driving system to validate the effectiveness of this method.



(a) Vehicle hardware

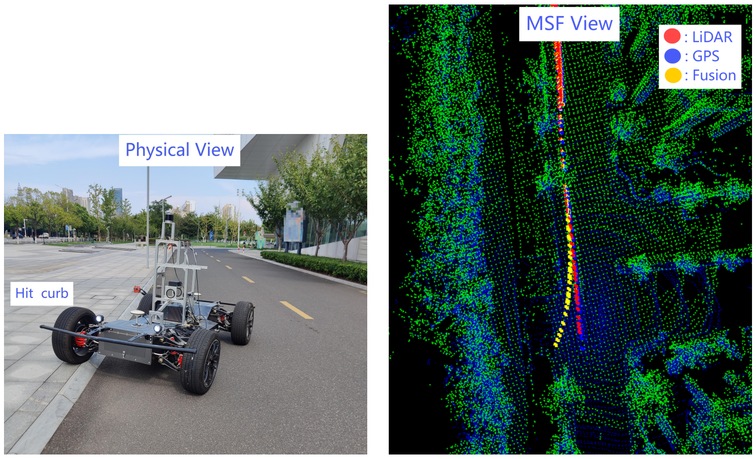


(b) Physical environment

Figure 13: Vehicle hardware and experiment environment

Experiment Setup. Our test vehicle is based on a Pix Hooke chassis equipped with the Apollo_Shenlan [48] autonomous driving system. The Apollo_Shenlan system is based on Apollo 6.0 edu [51] version, with the black-box MSF system replaced by Shenlan_MSF. Other hardware includes a 32-line LiDAR, a Huace CGI-410 INS, and a Nuvo-8111 industrial PC with an Intel Core i9-9900K CPU, NVIDIA RTX 3060 GPU, 32GB RAM, and a 1TB SSD, as depicted in Figure 13.

We selected a straight road and activated the full suite of autonomous driving modules, which included perception, localization, prediction, planning, routing, and control. To enable the autonomous driving, we set a specific target endpoint for the vehicle to reach. During the driving process, we strategically injected attack vectors aimed at modifying the GPS observation data, which in turn altered the vehicle's localization results. These manipulated localization results subsequently propagated through the downstream modules, ultimately leading to changes in the vehicle's throttle and steering control, thereby impacting its overall trajectory.



(a) Physical View (b) MSF View

Figure 14: End-to-end experiment for autonomous driving vehicle

Results. As shown in Figure 14, during the normal autonomous driving process, the injection attack successfully caused the vehicle to deviate from its intended trajectory, resulting in the vehicle colliding with the curb. This outcome clearly demonstrates the effectiveness of our method in exploiting system vulnerabilities, leading to significant disruptions as predicted. The ability to induce such a failure in a realistic setting underscores the practical applicability and impact of our approach on autonomous driving systems.

6. Limitations And Defense Discussions

6.1. Limitations

Error Model Constraints. Our research primarily focuses on the error-state Kalman filter algorithm. Due to the inherent complexity of the model, we rely on a steady-state assumption to filter and minimize potential errors, aiming to capture the primary impact of the injection-induced vehicle state deviations. Future work could explore model analysis under different states to enhance robustness and adaptability.

Testing Scenario Limitations. Due to the assumptions underlying our model analysis, our attack scenarios were only validated and tested in straight-line environments. This limitation means that the security analysis and attack methods have not been tested in more complex scenarios, such as those involving turns. This will serve as a starting point for future work.

6.2. Defense Discussions

System-Level Cross-Validation. GPS spoofing attacks can cause noticeable position discrepancies between LiDAR-based locators and MSF-based locators, as the manipulated GPS signals lead to false position estimates. By comparing the position estimates from both systems and validating the discrepancies against a predefined threshold, such attacks can be detected. If the discrepancy exceeds the threshold, it serves as an indicator of a GPS spoofing attack, prompting the system to take corrective actions.

Algorithm-Level State Monitoring. Monitoring changes in the MSF system's internal states over time helps understand its response to GPS spoofing attacks. Additionally, strengthening the detection of malicious sensor signals, such as spoofed GPS, before the fusion process can improve system robustness by excluding unreliable data and ensuring more accurate fusion outcomes.

7. Related Work

Sensor Spoofing and AD Security. Autonomous vehicles rely on various sensor data for tasks such as perception and localization. Attackers can forge sensor data to interfere with the autonomous system's estimation of the vehicle's current state, thereby posing a threat to vehicle safety. Existing research has addressed sensor spoofing across IMU, LiDAR, and GPS systems.

In the realm of IMU spoofing, two main types of attacks are identified. Trippel[52] exposed the susceptibility of MEMS accelerometers to malicious acoustic interference, leading to compromised linear and angular velocity data. Ji[53] manipulating IMU data to disrupt a vehicle's target detection functionality, specifically targeting the system's anomaly detection mechanisms. Similar to the studies on LiDAR deception, research on IMU spoofing predominantly focuses on attacks against individual sensors and does not address the challenges in scenarios involving the fusion of multiple sensors.

In the realm of LiDAR spoofing, Cao[11] developed a method for attackers to synchronize a photodiode with a LiDAR, creating deceptive points in the point cloud. Tu[54] explored the creation of adversarial 3D objects to mislead LiDAR systems. These objects, however, are noticeable due to their unique shapes and placements. Zhu[18] focused on identifying crucial adversarial positions in physical space, aiming to deceive LiDAR systems more efficiently. Jin[55] esigned a physical laser attack against LiDAR-based 3D object detection. Li[56] explores the vulnerability of LiDAR point cloud processing in autonomous vehicles to adversarial attacks through GPS-based trajectory spoofing, demonstrating that minor trajectory perturbations can lead to significant detection failures in 3D object detection systems, without the need to tamper with raw LiDAR data. These studies primarily concentrate on single-sensor deception strategies targeting LiDAR in autonomous driving systems, overlooking the complexities involved in multisensor fusion positioning tasks that incorporate LiDAR.

In the realm of GPS spoofing, Shen [14] was the first to successfully carry out GPS spoofing attacks on an MSF-based localization system and proposed an attack strategy. Building on Shen's approach, Chang [47] improved the success rate of attacks using FusionRipper by incorporating scene classification techniques. Zeng [57] introduced an idea that leveraged GPS spoofing to manipulate IMU data under similar scenarios, successfully misleading vehicle localization. Beyond autonomous vehicles, there has also been research on GPS spoofing attacks targeting drones and robots. For instance, Sathaye [58] successfully employed a GPS transmitter to spoof and manipulate the drone in a real-world scenario. Su[59] proposes a stealthy GPS spoofing strategy aimed at manipulating the trajectory of unmanned aerial vehicles (UAVs), effectively altering their flight paths without detection.

Security Analysis on Multi-Sensor Fusion. Nashimoto[45] explored the vulnerabilities of an Attitude and Heading Reference System (AHRS) under signal injection attacks, demonstrating significant security risks in systems that fuse data from multiple sensors, notably in inclination measurements. This work suggests new directions for bolstering the security of sensor fusion systems. Shen[14] developed FusionRipper, a technique for identifying and exploiting vulnerabilities in LiDAR-based ESKF systems, combining theoretical analysis with simulation experiments to pinpoint critical weaknesses, such as LiDAR locator uncertainty and ESKF initial state uncertainty. Chang[46] found that the sensor update frequency significantly affects the success of GPS spoofing attacks, corroborating FusionRipper's premises. However, vulnerabilities were deemed more critical in steady states, indicating the IMU's limited role in initiating takeover effects.

False Data Injection. Several studies have explored the vulnerability of Kalman filters to such attacks. Zhang[60] proposed a strategy for injecting fake data into Kalman filters using a constrained optimization problem. Li[61] developed an optimal attack strategy maximizing estimation error while ensuring stealthiness. Guo[62] investigated fake data injection attacks on multi-sensor systems, proposing an optimal strategy to maximize the degradation of estimation performance. Huo[63] studied these attacks on unstable and stable systems, designing strategies to degrade state estimation performance. Furthermore, Bonitz[64] proposed an attack view based on reinforcement learning to mislead the Kalman filter's output by observation spoofing.

8. Conclusion

This paper presents a comprehensive security analysis of Error-State Kalman Filter (ESKF) based multi-sensor fusion systems for autonomous driving, addressing a critical gap in the existing literature. We introduced a novel error analysis model specifically designed for ESKF-based localization systems, revealing that errors can be represented as a linear combination of injected signals under steady-state conditions. Leveraging this insight, we developed an energy-efficient attack method using constrained optimization techniques to minimize the required injection energy while achieving targeted localization errors. Our experiments, conducted on both simulated and real-world datasets, demonstrate the superiority of our approach compared to the state-of-the-art FusionRipper method. Our optimized injection attack achieves similar or greater levels of disruption while consuming approximately 9% less energy in simulated environments and 18% less in real-world scenarios. Furthermore, we validated the practical applicability of our method through end-to-end testing on an AI-powered autonomous driving system, successfully causing the vehicle to deviate from its intended path and collide with a curb.

These findings highlight the vulnerability of current ESKF-based fusion systems to sophisticated attacks and underscore the need for robust defense mechanisms. Future work should focus on developing countermeasures to detect and mitigate such energy-efficient injection attacks, ensuring the security and reliability of autonomous driving systems. Additionally, extending this analysis to other sensor fusion algorithms and exploring the impact of attacks on different driving scenarios could provide valuable insights for enhancing overall system resilience.

Appendix A. Error-State Kalman Filter Process

First, we introduce the error-state equation:

$$\delta\dot{p} = \delta v \tag{A.1}$$

$$\delta\dot{v} = -R_t\left[a_t - b_{a_t}\right]_\times \delta\theta + R_t\left(n_a - \delta b_a\right) \tag{A.2}$$

$$\delta\dot{\theta} = -\left[\omega_t - b_{\omega_t}\right]_\times \delta\theta + n_\omega - \delta b_\omega \tag{A.3}$$

$$\delta\dot{b}_a = n_{b_a} \tag{A.4}$$

$$\delta\dot{b}_\omega = n_{b_\omega} \tag{A.5}$$

where p is the position vector, v is the velocity vector, θ is the attitude error vector, b_{a_t} is the accelerometer bias, b_{ω_t} is the gyroscope bias, R_t is the rotation matrix at time t, a_t is the specific force vector at time t, ω_t is the angular velocity vector at time t, n_a is the accelerometer noise, n_ω is the gyroscope noise, n_{b_a} is the accelerometer bias noise, n_{b_ω} is the gyroscope bias noise, $[x]_\times$ is the skew-symmetric matrix of vector x, and δ denotes small changes or errors in the respective variables.

Let $\delta x = \begin{bmatrix} \delta p \\ \delta v \\ \delta\theta \\ \delta b_a \\ \delta b_\omega \end{bmatrix}, \quad w = \begin{bmatrix} n_a \\ n_\omega \\ n_{b_a} \\ n_{b_\omega} \end{bmatrix}$, the error equation can then be written in the general form of the state equation:

$$\delta\dot{x} = F_t \delta x + B_t w \tag{A.6}$$

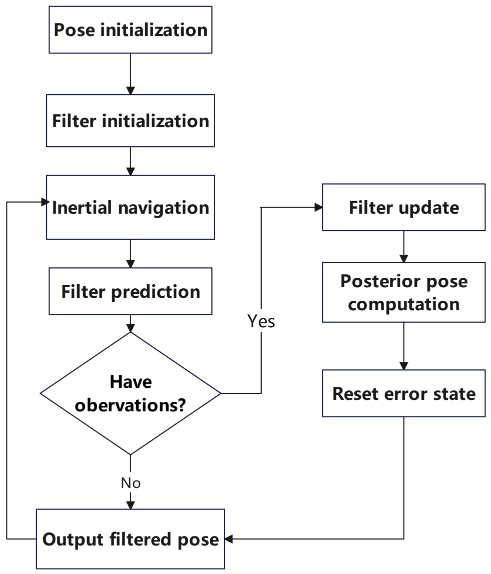


Figure A.15: ESKF processing flowchart

where

$$F_t = \begin{bmatrix} 0 & I_3 & 0 & 0 & 0 \\ 0 & 0 & -R_t\left[\overline{a}_t\right]_\times & -R_t & 0 \\ 0 & 0 & -\left[\overline{\omega}_t\right]_\times & 0 & -I_3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0, \end{bmatrix}, B_t = \begin{bmatrix} 0 & 0 & 0 & 0 \\ R_t & 0 & 0 & 0 \\ 0 & I_3 & 0 & 0 \\ 0 & 0 & I_3 & 0 \\ 0 & 0 & 0 & I_3 \end{bmatrix}$$

Then, we introduce the ESKF processing flowchart, as shown in Figure A.15.

Next we explain the most important parts:

Inertial navigation.

$$\check{R}_k = \hat{R}_{k-1}\left(I + \frac{\sin\phi}{\phi}(\phi\times) + \frac{1-\cos\phi}{\phi^2}(\phi\times)^2\right) \tag{A.7}$$

$$\check{v}_k = \hat{v}_{k-1} + \left(\frac{\check{R}_k\overline{a}_k + \hat{R}_{k-1}\overline{a}_{k-1}}{2} - g\right)(t_k - t_{k-1}) \tag{A.8}$$

$$\hat{p}_k = \check{p}_{k-1} + \frac{\check{v}_k + \hat{v}_{k-1}}{2}(t_k - t_{k-1}) \tag{A.9}$$

where $\check{R}_k$ is the predicted rotation matrix, $\hat{R}_{k-1}$ is the estimated rotation matrix at the previous time step, I is the identity matrix, ϕ is the rotation vector, $\check{v}_k$ is the predicted velocity, $\hat{v}_{k-1}$ is the estimated velocity at the previous time step, $\overline{a}_k$ is the specific force at time k, g is the gravity vector, t_k and t_{k-1} are the time steps, $\hat{p}_k$ is the estimated position at time k, and $\check{p}_{k-1}$ is the predicted position at the previous time step.

Filter prediction.

$$\delta\check{x}_k = F_{k-1}\delta\hat{x}_{k-1} + B_{k-1}w_k \tag{A.10}$$

$$\check{P}_k = F_{k-1}\hat{P}_{k-1}F_{k-1}^{\mathrm{T}} + B_{k-1}Q_kB_{k-1}^{T} \tag{A.11}$$

where $\delta\check{x}_k$ is the predicted state error, F_{k-1} is the state transition matrix at the previous time step, $\delta\hat{x}_{k-1}$ is the state error estimate at the previous time step, B_{k-1} is the input matrix at the previous time step, w_k is the process noise, $\check{P}_k$ is the predicted error covariance matrix, $\hat{P}_{k-1}$ is the error covariance matrix estimate at the previous time step, and Q_k is the process noise covariance matrix.

Filter update.

$$K_k = \check{P}_kG_k^{\mathrm{T}}\left(G_k\check{P}_kG_k^{\mathrm{T}} + C_kR_kC_k^{T}\right)^{-1} \tag{A.12}$$

$$\hat{P}_k = \left(I - K_kG_k\right)\check{P}_k \tag{A.13}$$

$$\delta\hat{x}_k = \delta\check{x}_k + K_k\left(y_k - G_k\delta\check{x}_k\right) \tag{A.14}$$

where K_k is the Kalman gain matrix, $\check{P}_k$ is the predicted error covariance matrix, G_k is the measurement matrix, C_k is the measurement matrix for the measurement noise, R_k is the measurement noise covariance matrix, $\hat{P}_k$ is the updated error covariance matrix, $\delta\hat{x}_k$ is the updated state estimate error, $\delta\check{x}_k$ is the predicted state error, and y_k is the measurement vector at time k.

Posterior pose computation.

$$\hat{p}_k = \check{p}_k - \delta\hat{p}_k \tag{A.15}$$

$$\hat{v}_k = \check{v}_k - \delta\hat{v}_k \tag{A.16}$$

$$\hat{R}_k = \check{R}_k\left(I - \left[\delta\hat{\theta}_k\right]_\times\right) \tag{A.17}$$

$$\hat{b}_{a_k} = \check{b}_{a_k} - \delta\hat{b}_{a_k} \tag{A.18}$$

$$\hat{b}_{\omega_k} = \check{b}_{\omega_k} - \delta\hat{b}_{\omega_k} \tag{A.19}$$

where $\hat{p}_k$ is the updated position, $\check{p}_k$ is the predicted position, $\delta\hat{p}_k$ is the position estimate error, $\hat{v}_k$ is the updated velocity, $\check{v}_k$ is the predicted velocity, $\delta\hat{v}_k$ is the velocity estimate error, $\hat{R}_k$ is the updated rotation matrix, $\check{R}_k$ is the predicted rotation matrix, and $\delta\hat{\theta}_k$ is the rotation vector error. $\hat{b}_{a_k}$ is the updated accelerometer bias, $\check{b}_{a_k}$ is the predicted accelerometer bias, and $\delta\hat{b}_a$ is the accelerometer bias estimate error. $\hat{b}_{\omega_k}$ is the updated gyroscope bias, $\check{b}_{\omega_k}$ is the predicted gyroscope bias, and $\delta\hat{b}_{\omega_k}$ is the gyroscope bias estimate error.

Reset error state.

$$\delta\hat{x}_k = 0 \tag{A.20}$$

where $\delta\hat{x}_k$ represents the state estimate error, which is set to zero.

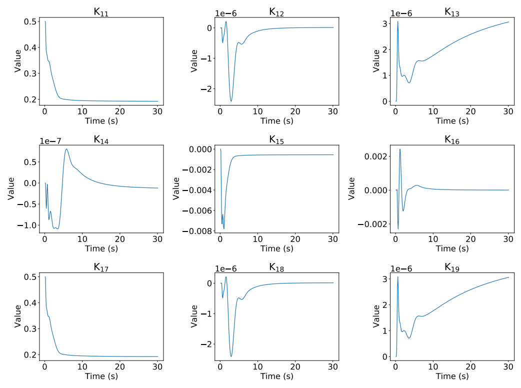
Appendix B. Fusion Strategy Derivation

Based on the steady-state assumption, the main elements of the K matrix are assumed to be approximately constant. To validate this assumption, we plotted the elements of the gain matrix K, as shown in Figure B.16. From the figure, it is evident that some elements are too small to be considered significant, while the main elements, such as K_{11} and K_{22}, are approximately constant.

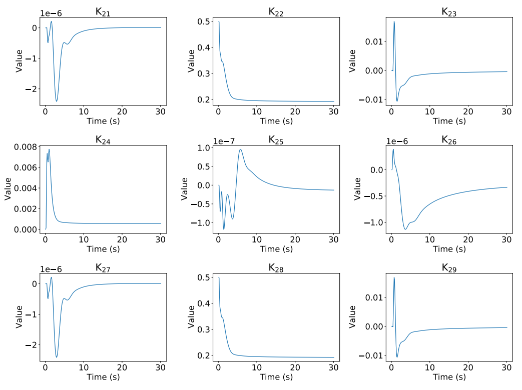
Building on the analysis presented above, we propose a refined approach to approximate the complex calculation processes typically involved in sensor fusion systems. By selectively focusing on the key elements of the Kalman gain matrix K and deliberately ignoring the elements with negligible impact, our method enhances computational efficiency without compromising accuracy. This strategic simplification is particularly crucial for real-time applications in autonomous driving systems, where computational resources are at a premium.

The high accuracy and reliability of our proposed method have been thoroughly validated in Section 5.2. Through extensive simulations and real-world testing scenarios, we have demonstrated that our approach provides a robust theoretical foundation for further development of injection attack strategies. These strategies aim to exploit vulnerabilities in sensor fusion systems to test and improve their resilience against adversarial conditions.

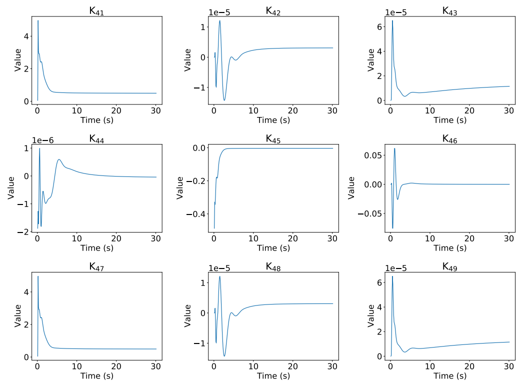
Regarding the fusion strategy discussed in Section 4.1, we have chosen to simplify the model by focusing solely on the states that directly influence the



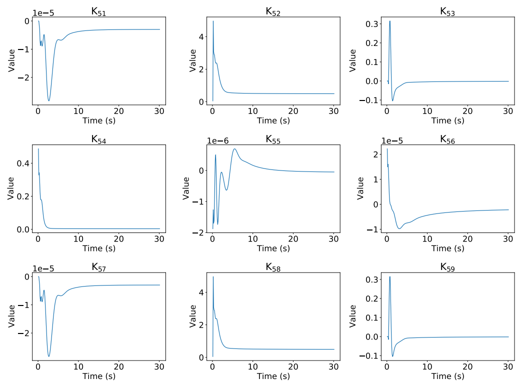
(a) row 1



(b) row 2



(c) row 4



(d) row 5

Figure B.16: K matrix in the steady state

vehicle's position. Specifically, we concentrate on the velocity components in the x and y directions ($\Delta\hat{X}_{k(41)}$ and $\Delta\hat{X}_{k(51)}$), as well as the positional state variables in the x and y directions ($\Delta\hat{X}_{k(11)}$ and $\Delta\hat{X}_{k(21)}$). This model simplification allows us to streamline the Kalman filter calculations, thus enhancing the system's overall efficiency and responsiveness.

The detailed derivation of our simplified model is provided below, illustrating how each selected state variable is updated through the filtering process. The focus is on ensuring that the core dynamics of the system are captured, while extraneous details that do not significantly affect the outcome are excluded. This approach not only clarifies the underlying mathematical structure but also aids in the practical implementation and optimization of the sensor fusion algorithm.

X-direction Position. The position in the X-direction is updated by incorporating both the predicted motion and the correction based on sensor measurements. This update is crucial for maintaining an accurate trajectory. Position $\Delta\hat{X}_{k+1(11)}$ is affected directly by the velocity of x and y direction $\Delta\hat{X}_{k(41)}$, $\Delta\hat{X}_{k(51)}$, thus the complete update process is as follows:

$$\begin{aligned}
\Delta\hat{X}_{k+1(11)} &= \Delta\hat{X}_{k(11)} + \Delta\hat{X}_{k(41)} \cdot T - K_{15\times 9} \begin{bmatrix} \Delta\hat{X}_{k(11)} + \Delta\hat{X}_{k(41)} \cdot T \\ \Delta\hat{X}_{k(21)} + \Delta\hat{X}_{k(51)} \cdot T \\ \Delta\hat{X}_{k(31)} \\ \Delta\hat{X}_{k(41)} \\ \Delta\hat{X}_{k(51)} \\ \Delta\hat{X}_{k(61)} \\ \Delta\hat{X}_{k(71)} + \epsilon_{k+1}\sin(\theta_z) \\ \Delta\hat{X}_{k(81)} - \epsilon_{k+1}\cos(\theta_z) \\ \Delta\hat{X}_{k(91)} \end{bmatrix} \\
&= \Delta\hat{X}_{k(11)} + \Delta\hat{X}_{k(41)} \cdot T - K_{15\times 9} \begin{bmatrix} \Delta\hat{X}_{k(11)} + \Delta\hat{X}_{k(41)} \cdot T \\ \Delta\hat{X}_{k(21)} + \Delta\hat{X}_{k(51)} \cdot T \\ 0 \\ \Delta\hat{X}_{k(41)} \\ \Delta\hat{X}_{k(51)} \\ 0 \\ \Delta\hat{X}_{k(11)} + \epsilon_{k+1}\sin(\theta_z) \\ \Delta\hat{X}_{k(21)} - \epsilon_{k+1}\cos(\theta_z) \\ 0 \end{bmatrix} \\
&= (1 - K_{11} - K_{17})\Delta\hat{X}_{k(11)} + (1 - K_{11})T \cdot \Delta\hat{X}_{k(41)} - \\
&\quad K_{17}\epsilon_{k+1}\sin(\theta_z) \qquad \text{(B.1)}
\end{aligned}$$

Y-direction Position. For the Y-direction, similar principles apply:

$$\begin{aligned}
\Delta\hat{X}_{k+1(21)} &= (1 - K_{22} - K_{28})\Delta\hat{X}_{k(21)} + (1 - K_{22})T \cdot \Delta\hat{X}_{k(51)} + \\
&\quad K_{28}\epsilon_{k+1}\cos(\theta_z) \qquad \text{(B.2)}
\end{aligned}$$

X-direction Velocity. Updating the velocity in the X-direction involves processing the position and velocity at last step:

$$\begin{aligned}
\Delta\hat{X}_{k+1(41)} &= \Delta\hat{X}_{k(41)} - K_{15\times 9}\begin{bmatrix} \Delta\hat{X}_{k(11)} + \Delta\hat{X}_{k(41)}\cdot T \\ \Delta\hat{X}_{k(21)} + \Delta\hat{X}_{k(51)}\cdot T \\ 0 \\ \Delta\hat{X}_{k(41)} \\ \Delta\hat{X}_{k(51)} \\ 0 \\ \Delta\hat{X}_{k(71)} + \epsilon_{k+1}\sin(\theta_z) \\ \Delta\hat{X}_{k(81)} - \epsilon_{k+1}\cos(\theta_z) \\ 0 \end{bmatrix} \\
&= \Delta\hat{X}_{k(41)} - K_{15\times 9}\begin{bmatrix} \Delta\hat{X}_{k(11)} + \Delta\hat{X}_{k(41)}\cdot T \\ \Delta\hat{X}_{k(21)} + \Delta\hat{X}_{k(51)}\cdot T \\ 0 \\ \Delta\hat{X}_{k(41)} \\ \Delta\hat{X}_{k(51)} \\ 0 \\ \Delta\hat{X}_{k(11)} + \epsilon_{k+1}\sin(\theta_z) \\ \Delta\hat{X}_{k(21)} - \epsilon_{k+1}\cos(\theta_z) \\ 0 \end{bmatrix} \\
&= (-K_{41} - K_{47})\Delta\hat{X}_{k(11)} + (1 - K_{41}T - K_{44})\cdot\Delta\hat{X}_{k(41)} - \\
&\quad K_{45}\cdot\Delta\hat{X}_{k(51)} - K_{47}\epsilon_{k+1}\sin(\theta_z) \qquad \text{(B.3)}
\end{aligned}$$

Y-direction Velocity. For the Y-velocity, similar principles apply:

$$\begin{aligned}
\Delta\hat{X}_{k+1(51)} &= (-K_{52} - K_{58})\Delta\hat{X}_{k(21)} + (1 - K_{52}T - K_{55})\cdot\Delta\hat{X}_{k(51)} - \\
&\quad K_{54}\cdot\Delta\hat{X}_{k(41)} + K_{58}\epsilon_{k+1}\cos(\theta_z) \qquad \text{(B.4)}
\end{aligned}$$

Appendix C. List of Symbols

The following symbols are used throughout this paper to describe the mathematical models and algorithms in the analysis of error-state Kalman filters for autonomous systems. The definitions provided aim to facilitate a clearer understanding of the mathematical formulations and their applications.

Symbol	Description	Units
$\check{}$	Prior state	-
$\hat{}$	Posterior state	-
δx	Estimate error state	-
δp	Position error state	meters (m)
δv	Velocity error state	meters per second (m/s)
$\delta \theta$	Attitude error state	radians (rad)
δb_a	Accelerometer bias error state	meters per second squared (m/s^2)
δb_ω	Gyroscope bias error state	radians per second (rad/s)
R_t	Rotation matrix at time t	-
a_t	Specific force vector at time t	meters per second squared (m/s^2)
ω_t	Angular velocity vector at time t	radians per second (rad/s)
b_{a_t}	Accelerometer bias at time t	meters per second squared (m/s^2)
b_{ω_t}	Gyroscope bias at time t	radians per second (rad/s)
n_a, n_ω	Accelerometer and gyroscope noise	-
$[x]_\times$	Skew-symmetric matrix of vector x	-

Table C.2: Symbol definitions used in the Error-State Kalman Filter.

Symbol	Description	Units
$\check{}$	Prior state	-
$\hat{}$	Posterior state	-
ΔX_k	Offset of state estimate at step k	-
$\Delta \hat{X}_{k(11)}$	Offset in X-direction position estimate	meters (m)
$\Delta \hat{X}_{k(21)}$	Offset in Y-direction position estimate	meters (m)
$\Delta \hat{X}_{k(41)}$	Offset in X-direction velocity estimate	meters per second (m/s)
$\Delta \hat{X}_{k(51)}$	Offset in Y-direction velocity estimate	meters per second (m/s)
$K_{15\times 9}$	Kalman gain matrix with dimensions 15×9	-
K_{ij}	Element of the Kalman gain matrix at row i, column j	-
d_t	Target deviation at time t	meters(m)
T	Time interval between updates	seconds (s)
θ_z	Rotation angle of z-direction	radians (rad)
ϵ_k	Inject signals at step k	meters(m)

Table C.3: Symbol definitions used in the Security Analysis.

References

[1] Waymo, Waymo - fully autonomous driving technology, `https://waymo.com/`, accessed: Aug. 2024 (2024).

[2] Baidu, Apollo autonomous driving platform, `https://github.com/ApolloAuto/apollo`, accessed: Aug. 2024 (2024).

[3] Autoware Foundation, Autoware, `https://autoware.org/`, accessed: Aug. 2024 (2024).

[4] Y. Deng, T. Zhang, G. Lou, X. Zheng, J. Jin, Q.-L. Han, Deep learning-based autonomous driving systems: A survey of attacks and defenses, IEEE Transactions on Industrial Informatics 17 (12) (2021) 7897–7912.

[5] J. Shen, N. Wang, Z. Wan, Y. Luo, T. Sato, Z. Hu, X. Zhang, S. Guo, Z. Zhong, K. Li, et al., Sok: On the semantic ai security in autonomous driving, arXiv preprint arXiv:2203.05314 (2022).

[6] D. Yu, S. Lee, R.-H. Hsu, J. Lee, Ensuring end-to-end security with fine-grained access control for connected and autonomous vehicles, IEEE Transactions on Information Forensics and Security (2024).

[7] T. G. Reid, S. E. Houts, R. Cammarata, G. Mills, S. Agarwal, A. Vora, G. Pandey, Localization requirements for autonomous vehicles, arXiv preprint arXiv:1906.01061 (2019).

[8] J. Levinson, M. Montemerlo, S. Thrun, Map-based precision vehicle localization in urban environments., in: Robotics: science and systems, Vol. 4, Atlanta, GA, USA, 2007, pp. 121–128.

[9] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, D. Song, Robust physical-world attacks on deep learning visual classification, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 1625–1634.

[10] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, T. Kohno, Physical adversarial examples for object detectors, in: 12th USENIX workshop on offensive technologies (WOOT 18), 2018.

[11] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, 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.

[12] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, K. Chen, Seeing isn't believing: Towards more robust adversarial attack against real world object detectors, in: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, 2019, pp. 1989–2004.

[13] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. A. Chen, M. Liu, B. Li, Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks, in: 2021 IEEE symposium on security and privacy (SP), IEEE, 2021, pp. 176–194.

[14] J. Shen, J. Y. Won, Z. Chen, Q. A. Chen, Drift with devil: Security of {Multi-Sensor} fusion based localization in {High-Level} autonomous driving under {GPS} spoofing, in: 29th USENIX security symposium (USENIX Security 20), 2020, pp. 931–948.

[15] S. Kochanthara, T. Singh, A. Forrai, L. Cleophas, Safety of perception systems for automated driving: A case study on apollo, ACM Transactions on Software Engineering and Methodology 33 (3) (2024) 1–28.

[16] J. Liu, J.-M. Park, "seeing is not always believing": detecting perception error attacks against autonomous vehicles, IEEE Transactions on Dependable and Secure Computing 18 (5) (2021) 2209–2223.

[17] J. Sun, Y. Cao, Q. A. Chen, Z. M. Mao, Towards robust {LiDAR-based} perception in autonomous driving: General black-box adversarial sensor attack and countermeasures, in: 29th USENIX Security Symposium (USENIX Security 20), 2020, pp. 877–894.

[18] Y. Zhu, C. Miao, T. Zheng, F. Hajiaghajani, L. Su, C. Qiao, Can we use arbitrary objects to attack lidar perception in autonomous driving?, in: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021, pp. 1945–1960.

[19] Y. Zhu, C. Miao, H. Xue, Y. Yu, L. Su, C. Qiao, Malicious attacks against multi-sensor fusion in autonomous driving, in: Proceedings of

the 30th Annual International Conference on Mobile Computing and Networking, 2024, pp. 436–451.

[20] Y. Chen, Y. Huai, S. Li, C. Hong, J. Garcia, Misconfiguration software testing for failure emergence in autonomous driving systems, Proceedings of the ACM on Software Engineering 1 (FSE) (2024) 1913–1936.

[21] S. Kim, M. Liu, J. J. Rhee, Y. Jeon, Y. Kwon, C. H. Kim, Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022, pp. 1753–1767.

[22] G. Lou, Y. Deng, X. Zheng, M. Zhang, T. Zhang, Testing of autonomous driving systems: where are we and where should we go?, in: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2022, pp. 31–43.

[23] S. Tang, Z. Zhang, Y. Zhang, J. Zhou, Y. Guo, S. Liu, S. Guo, Y.-F. Li, L. Ma, Y. Xue, et al., A survey on automated driving system testing: Landscapes and trends, ACM Transactions on Software Engineering and Methodology 32 (5) (2023) 1–62.

[24] G. Wan, X. Yang, R. Cai, H. Li, Y. Zhou, H. Wang, S. Song, Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes, in: 2018 IEEE international conference on robotics and automation (ICRA), IEEE, 2018, pp. 4670–4677.

[25] A. Soloviev, Tight coupling of gps, laser scanner, and inertial measurements for navigation in urban environments. in 2008 ieee/ion position, location and navigation symposium (pp. 511–525) (2008).

[26] J. Kelly, G. S. Sukhatme, Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration, The International Journal of Robotics Research 30 (1) (2011) 56–79.

[27] Z. Tao, P. Bonnifait, V. Fremont, J. Ibanez-Guzman, Mapping and localization using gps, lane markings and proprioceptive sensors, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 406–412.

[28] Y. Gao, S. Liu, M. M. Atia, A. Noureldin, Ins/gps/lidar integrated navigation system for urban and indoor environments using hybrid scan matching algorithm, Sensors 15 (9) (2015) 23286–23302.

[29] J. K. Suhr, J. Jang, D. Min, H. G. Jung, Sensor fusion-based low-cost vehicle localization system for complex urban environments, IEEE Transactions on Intelligent Transportation Systems 18 (5) (2016) 1078–1086.

[30] M. Schreiber, H. Königshof, A.-M. Hellmund, C. Stiller, Vehicle localization with tightly coupled gnss and visual odometry, in: 2016 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2016, pp. 858–863.

[31] F. de Ponte Müller, Survey on ranging sensors and cooperative techniques for relative positioning of vehicles, Sensors 17 (2) (2017) 271.

[32] I. Brigadnov, A. Lutonin, K. Bogdanova, Error state extended kalman filter localization for underground mining environments, Symmetry 15 (2) (2023) 344.

[33] N. Davari, A. Gholami, Variational bayesian adaptive kalman filter for asynchronous multirate multi-sensor integrated navigation system, Ocean Engineering 174 (2019) 108–116.

[34] S. Fakoorian, A. Santamaria-Navarro, B. T. Lopez, D. Simon, A.-a. Agha-mohammadi, Towards robust state estimation by boosting the maximum correntropy criterion kalman filter with adaptive behaviors, IEEE Robotics and Automation Letters 6 (3) (2021) 5469–5476.

[35] Z. Li, Y. Zhang, Constrained eskf for uav positioning in indoor corridor environment based on imu and wifi, Sensors 22 (1) (2022) 391.

[36] A. Mikov, A. Panyov, V. Kosyanchuk, I. Prikhodko, Sensor fusion for land vehicle localization using inertial mems and odometry, in: 2019 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL), IEEE, 2019, pp. 1–2.

[37] A. I. Mourikis, S. I. Roumeliotis, A multi-state constraint kalman filter for vision-aided inertial navigation, in: Proceedings 2007 IEEE international conference on robotics and automation, IEEE, 2007, pp. 3565–3572.

[38] S. Piperakis, D. Kanoulas, N. G. Tsagarakis, P. Trahanias, Outlier-robust state estimation for humanoid robots, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2019, pp. 706–713.

[39] S. I. Roumeliotis, G. S. Sukhatme, G. A. Bekey, Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization, in: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), Vol. 2, IEEE, 1999, pp. 1656–1663.

[40] J. Sola, Quaternion kinematics for the error-state kf, Laboratoire dAnalyse et dArchitecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Toulouse, France, Tech. Rep (2012) 35.

[41] J. Sola, Quaternion kinematics for the error-state kalman filter, arXiv preprint arXiv:1711.02508 (2017).

[42] A. Soliman, G. A. Ribeiro, A. Torres, M. Rastgaar, Error-state kalman filter for online evaluation of ankle angle, in: 2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), IEEE, 2022, pp. 225–231.

[43] Y. Yin, J. Zhang, M. Guo, X. Ning, Y. Wang, J. Lu, Sensor fusion of gnss and imu data for robust localization via smoothed error state kalman filter, Sensors 23 (7) (2023) 3676.

[44] W. Zhen, S. Zeng, S. Soberer, Robust localization and localizability estimation with a rotating laser scanner, in: 2017 IEEE international conference on robotics and automation (ICRA), IEEE, 2017, pp. 6240–6245.

[45] S. Nashimoto, D. Suzuki, T. Sugawara, K. Sakiyama, Sensor con-fusion: Defeating kalman filter in signal injection attack, in: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, 2018, pp. 511–524.

[46] J. Chang, L. Zhang, L.-T. Hsu, B. Xu, F. Huang, D. Xu, Analytic models of a loosely coupled gnss/ins/lidar kalman filter considering update frequency under a spoofing attack, IEEE Sensors Journal 22 (23) (2022) 23341–23355.

[47] J. Chang, F. Huang, L. Zhang, D. Xu, L.-T. Hsu, Selection of areas for effective gnss spoofing attacks to a vehicle-mounted msf system based on scenario classification models, IEEE Transactions on Vehicular Technology 72 (11) (2023) 14645–14655.

[48] S. 2017, Apollo_shenlan, https://github.com/shenlan2017/Apollo_ShenLan, accessed: Aug. 2024 (2024).

[49] Aceinna, Gnss-ins-sim: An open-source gnss/ins simulation platform, https://github.com/Aceinna/gnss-ins-sim, accessed: Aug. 2024 (2024).

[50] SciPy, Scipy - scientific computing tools for python, https://scipy.org/, accessed: Aug. 2024.

[51] ApolloAuto, Apollo: An open autonomous driving platform, https://github.com/ApolloAuto/apollo/tree/v6.0_edu, accessed: Aug. 2024 (2024).

[52] T. Trippel, O. Weisse, W. Xu, P. Honeyman, K. Fu, Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks, in: 2017 IEEE European symposium on security and privacy (EuroS&P), IEEE, 2017, pp. 3–18.

[53] X. Ji, Y. Cheng, Y. Zhang, K. Wang, C. Yan, W. Xu, K. Fu, Poltergeist: Acoustic adversarial machine learning against cameras and computer vision, in: 2021 IEEE Symposium on Security and Privacy (SP), IEEE, 2021, pp. 160–175.

[54] J. Tu, M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng, R. Urtasun, Physically realizable adversarial examples for lidar object detection, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 13716–13725.

[55] Z. Jin, X. Ji, Y. Cheng, B. Yang, C. Yan, W. Xu, Pla-lidar: Physical laser attacks against lidar-based 3d object detection in autonomous vehicle, in: 2023 IEEE Symposium on Security and Privacy (SP), IEEE, 2023, pp. 1822–1839.

[56] Y. Li, C. Wen, F. Juefei-Xu, C. Feng, Fooling lidar perception via adversarial trajectory perturbation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 7898–7907.

[57] K. C. Zeng, S. Liu, Y. Shu, D. Wang, H. Li, Y. Dou, G. Wang, Y. Yang, All your {GPS} are belong to us: Towards stealthy manipulation of road navigation systems, in: 27th USENIX security symposium (USENIX security 18), 2018, pp. 1527–1544.

[58] H. Sathaye, M. Strohmeier, V. Lenders, A. Ranganathan, An experimental study of {GPS} spoofing and takeover attacks on {UAVs}, in: 31st USENIX security symposium (USENIX security 22), 2022, pp. 3503–3520.

[59] J. Su, J. He, P. Cheng, J. Chen, A stealthy gps spoofing strategy for manipulating the trajectory of an unmanned aerial vehicle, IFAC-PapersOnLine 49 (22) (2016) 291–296.

[60] Z. Zhang, L. Zhou, P. Tokekar, Strategies to inject spoofed measurement data, arXiv preprint arXiv:1809.04756 (2018).

[61] Y.-G. Li, G.-H. Yang, Optimal stealthy false data injection attacks in cyber-physical systems, Information Sciences 481 (2019) 474–490.

[62] Y. Guo, M. Wu, K. Tang, J. Tie, X. Li, Covert spoofing algorithm of uav based on gps/ins-integrated navigation, IEEE Transactions on Vehicular Technology 68 (7) (2019) 6557–6564.

[63] J.-R. Huo, X.-J. Li, False data injection attacks on sensors against state estimation in cyber-physical systems, Journal of the Franklin Institute 360 (9) (2023) 6110–6130.

[64] D. A. Bonitz, Using reinforcement learning to spoof a monitored kalman filter, Ph.D. thesis, Monterey, CA; Naval Postgraduate School (2022).