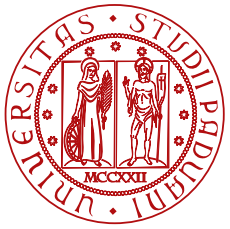


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

University of Padova
Department of Information Engineering
Master Thesis in Control System Engineering

Architectural Comparison of Kalman Filter-Based Multi-Sensor Localization for a Duckiebot

Master Candidate

Luca Callegaro

Student ID 2103660

Supervisor

Prof. Angelo Cenedese

University of Padova

Co-supervisor

Dott. Riccardo Antonello

University of Padova

Academic Year: 2025/2026
23/04/2026

To my parents
and friends

Abstract

Accurate localization is a fundamental requirement for autonomous mobile robots, as it directly impacts navigation performance and system reliability. In low-cost robotic platforms, localization must rely on onboard sensors such as wheel encoders, inertial measurement units, and vision-based systems, which are inherently affected by noise, drift, and limited observability. These challenges motivate the use of sensor fusion techniques to obtain reliable state estimates over time.

This thesis investigates how different Kalman filter architectures influence localization performance for a unicycle-model mobile robot. Rather than proposing a novel estimation algorithm, this work focuses on the role of architectural design choices in sensor fusion. Standalone filtering baselines, centralized sensor fusion, cascaded filtering schemes, and error-state Kalman filter architectures are analyzed within a unified estimation framework, with particular attention to accuracy, robustness, and fault tolerance when fusing heterogeneous sensor measurements.

The considered architectures are implemented in a MATLAB/Simulink simulation environment that enables consistent modeling of robot kinematics, sensor characteristics, and noise processes. Wheel encoder and inertial measurements provide high-rate motion information, while vision-based localization using AprilTag detections acts as an external correction source to mitigate long-term drift. The performance of each filtering approach is evaluated under realistic operating conditions using quantitative error and consistency metrics.

Simulation results are further validated through experimental tests on a Duckietown robotic platform, demonstrating the practical relevance of architectural choices in Kalman filter-based localization for real-world mobile robots.

Contents

List of Figures

List of Tables

List of Acronyms

IMU Inertial Measurement Unit

KF Kalman Filter

EKF Extended Kalman Filter

ESKF Error-State Kalman Filter

DDR Differential Drive Robot

NIS Normalized Innovation Squared

ROS Robot Operating System

RMSE Root-Mean-Square Error

SLAM Simultaneous Localization and Mapping

Notation

The main symbols used throughout this thesis are summarized below. Unless otherwise specified, all variables are expressed in discrete time and refer to quantities evaluated at time step k.

State and Estimation Variables

Symbol	Description	Unit
q_k	Robot state vector	–
x_k, y_k	Robot position in world frame	m
θ_k	Robot heading angle	rad
b_k	Gyroscope bias	rad/s
$\hat{q}_k$	Estimated state	–
P_k	State covariance matrix	–
K_k	Kalman gain	–

Motion and Input Quantities

Symbol	Description	Unit
u_k	Input vector	–
Δs_k	Incremental traveled distance	m
$\Delta s_L, \Delta s_R$	Wheel displacements	m
$\Delta\theta_k$	Heading increment	rad
T_s	Sampling interval	s
v	Linear velocity	m/s
ω	Angular velocity	rad/s

Noise and Covariance Terms

Symbol	Description	Unit
w_k	Process noise	–
v_k	Measurement noise	–
Q_k	Process noise covariance	–
R_k	Measurement noise covariance	–
σ_s	Translational noise std	m
σ_θ	Rotational noise std	rad
σ_b	Bias random-walk std	rad/s
σ_ϕ	Bearing noise std	rad
σ_ρ	Range noise std	m

1

Introduction

1.1 Motivation

Accurate localization is a fundamental requirement for autonomous mobile robots, as it directly affects their ability to navigate, perceive the environment, and execute control tasks safely and reliably. In many practical applications, especially in educational and research platforms such as Duckietown, localization is achieved using low-cost sensors operating in structured but imperfect environments. Such constraints introduce significant challenges for state estimation, primarily due to the non-negligible impact of sensor noise and the inherent partial observability of the system.

Duckietown is an open-source educational and research ecosystem designed to enable hands-on experimentation with autonomous driving concepts using low-cost robotic platforms [1]. Its reference robot, the Duckiebot, integrates wheel encoders, an inertial measurement unit, and a monocular camera within a constrained computational and sensing budget. These characteristics make Duckietown a representative testbed for studying localization and sensor fusion strategies under realistic resource and sensing limitations, while still operating in a structured environment with known landmarks.

Figure 1.1: Duckiebot mobile robot used as a reference platform in this thesis, equipped with wheel encoders, IMU, and a monocular camera.

Mobile robots such as the Duckiebot (Figure 1.1), which this work focuses on, are commonly modeled using the unicycle kinematic model [2], which captures the essential nonholonomic behavior of differential-drive platforms. While this model is simple and computationally efficient, it is inherently nonlinear and sensitive to modeling errors and disturbances such as wheel slip, actuator saturation, and uneven surfaces. As a result, localization based on just a single sensor is often insufficient to guarantee reliable performance over time.

Sensor fusion techniques aim to overcome these limitations by combining information from multiple heterogeneous sensors, such as wheel encoders, inertial measurement units, and vision-based systems. Among the available estimation methods, Kalman filtering and its nonlinear extensions have become a standard solution due to their solid theoretical foundation and real-time applicability [2, 3].

However, beyond the choice of the filtering algorithm itself, the architecture adopted to fuse sensor information plays a crucial role in determining localization accuracy, robustness, and computational efficiency [4]. Centralized, standalone, and cascaded Kalman-based schemes offer different trade-offs in terms of modularity, fault tolerance, and sensitivity to modeling assumptions.

In tightly coupled robotic systems, where multiple sensors observe the same underlying state variables, architectural design becomes particularly critical. Although measurement noises may be modeled as independent, the estimation errors induced by different sensing modalities are inherently correlated through the shared state dynamics. If such cross-correlations are neglected, especially in multi-stage filtering schemes, the resulting estimator may become statistically inconsistent and overconfident [3].

While Kalman filtering itself is extensively studied, comparatively fewer works provide a systematic experimental evaluation of how different fusion architectures affect consistency and performance under identical sensing conditions on low-cost unicycle robots.

1.2 Problem Statement

This thesis addresses the problem of state estimation for a unicycle-model mobile robot equipped with multiple sensors, focusing on how different Kalman filter architectures influence localization performance. The primary challenge lies in accurately estimating the robot pose in the presence of nonlinear dynamics, noisy measurements, and imperfect sensor models.

Rather than proposing a new estimation algorithm, this work investigates how different configurations of Kalman filters, fused in a centralized manner, or combined through a cascaded filtering framework, affect localization accuracy and robustness. The comparison is performed through both simulation and experimental evaluation. Simulation allows controlled analysis of modeling assumptions and noise processes, while real-world experiments on the Duckiebot platform provide validation under practical sensing limitations, timing irregularities, and real sensor imperfections.

To this end, a detailed simulation framework is developed using MATLAB/Simulink, which allows precise modeling of robot kinematics, sensor characteristics, and noise processes. The Duckiebot platform is then used exclusively as a hardware testbed to validate the conclusions drawn from simulation, ensuring that the results remain relevant to real-world robotic systems.

1.3 Objectives

The main objectives of this thesis are:

- To model the kinematics of a unicycle-model mobile robot for localization purposes

- To derive realistic measurement models for the sensors available on the Duckiebot platform
- To design and implement multiple Kalman filter architectures, including standalone, centralized, and cascaded approaches
- To quantitatively compare localization performance using standardized error metrics
- To validate simulation results through experimental tests on real robotic hardware

The main contributions of this thesis are:

- A unified MATLAB-based evaluation framework for comparing Kalman filter architectures under identical sensing conditions and real recorded datasets.
- A detailed statistical consistency analysis using NIS metrics and covariance-bound validation.
- The implementation and experimental validation of centralized, cascaded, error-state and hybrid-cascaded EKF architectures for unicycle-model localization.
- An experimental comparison over multi-lap trajectories with motion-capture ground truth, highlighting the impact of architectural design on estimation accuracy and robustness.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 introduces the theoretical background required for this work, including the unicycle kinematic model and Kalman filtering techniques. Chapter 3 describes the system modeling, including robot dynamics and sensor measurement models. Chapter 4 presents the different Kalman filter architectures considered in this study. Chapter 5 describes the software implementation, discretization strategy, and numerical considerations adopted in the filter realization. Chapter 6 discusses the simulation results and comparative analysis. Chapter 7 presents experimental validation using the Duckiebot platform. Finally, Chapter 8 concludes the thesis and outlines possible directions for future work.

2

Background

This chapter introduces the theoretical foundations required for the development and comparison of Kalman filter-based localization architectures. The focus is placed on probabilistic state estimation for nonlinear mobile robots, with particular attention to the unicycle kinematic model and multisensor fusion strategies relevant to the Duckiebot platform.

2.1 Mobile Robot Kinematics

Mobile robots operating on planar surfaces are commonly modeled using the unicycle kinematic model [2, 5], which captures the essential motion constraints of differential-drive platforms.

The Duckiebot is a Differential Drive Robot (DDR) with three wheels: two fixed and one omnidirectional wheel; henceforth the distance between the fixed wheels is called l and the radius of each wheel is called r. The sagittal axis can be defined as the line passing through the center of gravity of the unicycle, perpendicular to the axis of rotation of the wheels and aligned with the direction of motion. Figure 2.1 shows a simple model of the Duckiebot. The Duckiebot kinematics are modeled by the unicycle model, in which the robot configuration is described by the state vector

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.1}$$

where x and y indicate the Cartesian coordinates of the Duckiebot position and θ

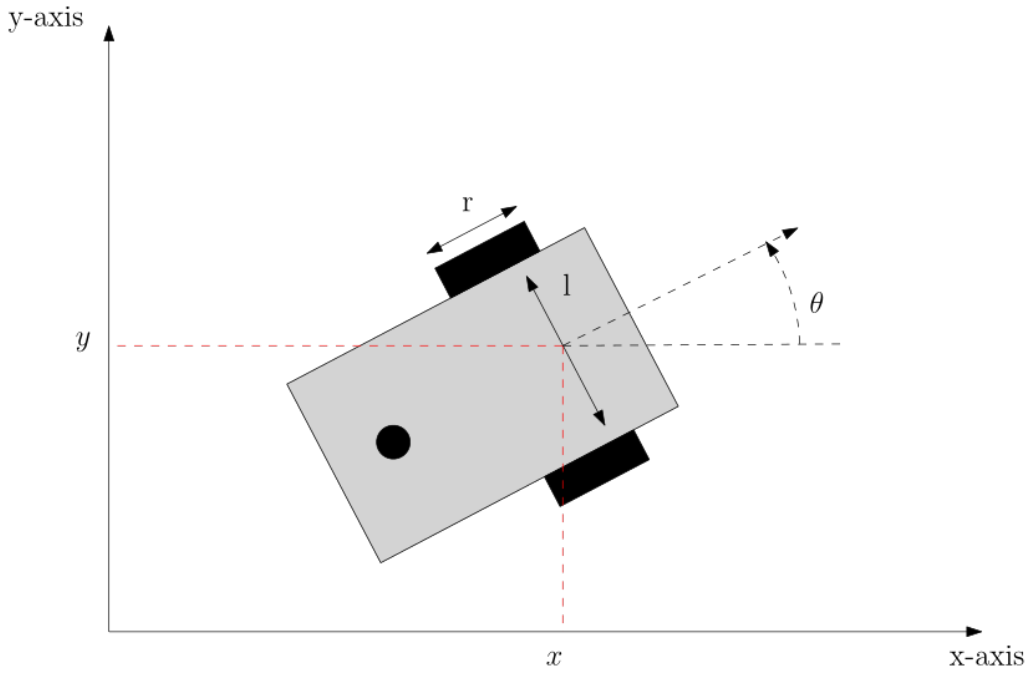


Figure 2.1: Duckiebot model

the yaw angle of the robot, measured counterclockwise, with respect to the x-axis of the world frame, while the system inputs are v and ω, namely, the magnitude of the linear and angular velocities, respectively. This system is subject to a pure-rolling constraint and, using this constraint in Pfaffian form [5, 6], the kinematic equations can be written as:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.2}$$

that can be rewritten as:

$$\dot{x} = v\cos\theta, \qquad \dot{y} = v\sin\theta, \qquad \dot{\theta} = \omega. \tag{2.3}$$

In discrete time, assuming piecewise-constant velocities over a sampling interval T_s (which may vary between updates in practical implementations), the model can be approximated as

$$x_{k+1} = x_k + \Delta s\cos(\theta_k) \tag{2.4}$$

$$y_{k+1} = y_k + \Delta s\sin(\theta_k) \tag{2.5}$$

$$\theta_{k+1} = \theta_k + \Delta\theta \tag{2.6}$$

where Δs and $\Delta\theta$ represent the incremental displacement and rotation, typically

obtained from wheel encoder measurements.
Due to the nonlinear dependence on θ, this model requires nonlinear estimation techniques when used for localization. Furthermore, real-world effects such as wheel slip, uneven surfaces, and actuator imperfections introduce modeling errors that accumulate over time.

2.2 Sensors for Localization

Reliable localization cannot be achieved using dead reckoning alone, as errors grow unbounded without external corrections. For this reason, mobile robots typically rely on multisensor fusion.

In these experiments, the Duckiebot platform integrates three main sensing modalities: wheel encoders, an Inertial Measurement Unit (IMU), and a monocular camera detecting AprilTag landmarks. These sensing modalities provide complementary information: wheel encoders offer high-rate relative motion estimates, the IMU supplies short-term rotational dynamics, and vision-based observations provide absolute position corrections. Their different noise characteristics, update rates, and failure modes motivate the need for a principled fusion framework.

2.2.1 Wheel Encoders

Wheel encoders provide incremental motion estimates by measuring wheel rotations. For a differential-drive robot, the left and right wheel displacements Δs_L and Δs_R yield

$$\Delta s = \frac{\Delta s_L + \Delta s_R}{2}, \tag{2.7}$$

$$\Delta \theta = \frac{\Delta s_R - \Delta s_L}{l}, \tag{2.8}$$

where l denotes the wheelbase. Encoder-based odometry provides high-rate motion updates but suffers from drift due to wheel radius uncertainty and slippage.

2.2.2 Inertial Measurements

The Inertial Measurement Unit (IMU) gyroscope measures the angular velocity of the robot. The measurement can be modeled as

$$\omega_m = \omega + b + n_\omega, \tag{2.9}$$

where b represents a slowly varying bias and n_ω is zero-mean Gaussian noise [3]. Bias estimation is critical, as uncompensated bias leads to significant heading drift.

2.2.3 Vision-Based Landmark Observations

The monocular camera provides external corrections through the detection of AprilTags placed at known positions in the environment [7]. For a tag located at world coordinates (x_t, y_t), the robot observes a bearing ϕ and range ρ:

$$z = \begin{bmatrix} \phi \\ \rho \end{bmatrix}. \tag{2.10}$$

The corresponding nonlinear measurement model is

$$\phi = \text{atan2}(y_t - y,\ x_t - x) - \theta, \tag{2.11}$$

$$\rho = \sqrt{(x_t - x)^2 + (y_t - y)^2}. \tag{2.12}$$

Landmark observations provide absolute corrections that prevent long-term drift, but are limited by camera field of view, occlusions, and detection noise.

Although measurement noises are commonly modeled as independent, all sensing modalities observe the same underlying robot state. As a result, estimation errors become inherently correlated through the shared state dynamics. This aspect plays a crucial role in the design of multisensor fusion architectures, particularly in multi-filter configurations where such correlations are not explicitly represented.

2.3 Bayesian State Estimation

Localization can be formulated as a recursive probabilistic inference problem [2, 8]. The objective is to estimate the posterior distribution

$$p(q_k \mid z_{1:k}), \tag{2.13}$$

where $z_{1:k}$ denotes the sequence of measurements up to time k. Under the Markov assumption, the system state at time k depends only on the previous state q_{k-1} and the control input u_{k-1}, while measurements are conditionally independent given the

current state. These assumptions allow the posterior to be computed recursively through two fundamental steps: prediction and correction.

The prediction step propagates the previous posterior through the motion model,

$$p(q_k \mid z_{1:k-1}) = \int p(q_k \mid q_{k-1}, u_{k-1})\, p(q_{k-1} \mid z_{1:k-1})\, dq_{k-1}, \tag{2.14}$$

which corresponds to marginalization of the prior distribution. The correction step incorporates the new measurement using Bayes' rule,

$$p(q_k \mid z_{1:k}) \propto p(z_k \mid q_k)\, p(q_k \mid z_{1:k-1}). \tag{2.15}$$

This recursive structure forms the foundation of Bayesian filtering. Exact computation of these integrals is generally intractable for nonlinear systems, motivating the use of approximated filtering techniques such as the Kalman filter and its nonlinear extensions.

2.4 The Kalman Filter

For linear systems of the form

$$q_{k+1} = Fq_k + Gu_k + w_k, \tag{2.16}$$

$$z_k = Hq_k + v_k, \tag{2.17}$$

with Gaussian noise $w_k \sim \mathcal{N}(0, Q)$ and $v_k \sim \mathcal{N}(0, R)$, the Kalman filter computes the minimum variance estimate [3, 9]. Under the assumptions of linear dynamics and Gaussian noise, the posterior distribution remains Gaussian at every time step. Consequently, it is fully characterized by its mean and covariance, which are propagated recursively by the filter. The Kalman filter is optimal in the minimum mean-square error sense when the system model and noise statistics are correctly specified, i.e., when they accurately represent the true underlying system. The prediction step is given by

$$\hat{q}_{k|k-1} = F\hat{q}_{k-1}, \tag{2.18}$$

$$P_{k|k-1} = FP_{k-1}F^T + Q. \tag{2.19}$$

The correction step is

$$K_k = P_{k|k-1}H^T \left(HP_{k|k-1}H^T + R\right)^{-1}, \tag{2.20}$$

$$\hat{q}_k = \hat{q}_{k|k-1} + K_k(z_k - H\hat{q}_{k|k-1}), \tag{2.21}$$

$$P_k = (I - K_k H)P_{k|k-1}. \tag{2.22}$$

In practical implementations, the Joseph stabilized covariance update form

$$P_k = (I - K_k H_k)P_{k|k-1}(I - K_k H_k)^T + K_k R K_k^T \tag{2.23}$$

is often preferred to preserve symmetry and positive semi-definiteness under finite-precision arithmetic [3]. This stabilized form is adopted in the implementations presented in this thesis.

The Kalman gain K_k determines the relative weighting between the predicted state and the measurement innovation. Intuitively, it balances model confidence, represented by $P_{k|k-1}$, against measurement uncertainty, represented by R. Higher process uncertainty increases the reliance on measurements, while higher measurement noise leads the filter to trust the prediction more strongly. Despite its optimality under linear-Gaussian assumptions, the Kalman filter is sensitive to modeling inaccuracies and incorrect noise covariance tuning. In practical robotic systems, model errors, unmodeled dynamics, and correlated sensor noise may degrade consistency. These limitations motivate the use of nonlinear extensions and careful architectural design in multisensor fusion systems.

2.5 Extended Kalman Filter

Mobile robot localization is inherently nonlinear due to the unicycle dynamics and landmark observation models. The Extended Kalman Filter (EKF) extends the Kalman filter framework to nonlinear systems [10] described by

$$q_{k+1} = f(q_k, u_k) + w_k, \tag{2.24}$$

$$z_k = h(q_k) + v_k. \tag{2.25}$$

At each time step, the EKF approximates the nonlinear functions $f(\cdot)$ and $h(\cdot)$ by first-order Taylor expansions around the current state estimate. The resulting Jacobian matrices

$$F_k = \frac{\partial f}{\partial q}, \qquad H_k = \frac{\partial h}{\partial q}, \tag{2.26}$$

define a locally linear model used for covariance propagation and measurement update.

The validity of this approximation depends on both the magnitude of the state uncertainty and the degree of system nonlinearity. When uncertainty remains small, the linear model is typically accurate; however, in the presence of strong nonlinearities or poor initialization, approximation errors may become significant.

Since the linearization is performed about the estimated state rather than the true state, modeling errors influence both the state update and the covariance propagation. This may lead to inconsistency, where the filter becomes overconfident and underestimates the actual estimation error. Such effects are particularly relevant in mobile robotics, where heading uncertainty and nonlinear range–bearing observations strongly affect estimation performance.

For differential-drive platforms, nonlinearities arise in the motion model through the orientation-dependent kinematics, and in the observation model through trigonometric measurement relationships. Accurate Jacobian computation and appropriate noise modeling are therefore essential to maintain numerical stability and reliable performance.

In multisensor fusion architectures involving multiple update stages, these approximation effects may interact with architectural design choices, potentially influencing global consistency and robustness. The impact of such design decisions is analyzed in Chapter 4.

2.6 Error-State Kalman Filtering

An alternative formulation is the Error-State Kalman Filter (ESKF) [11, 12], in which the nominal state is propagated using the full nonlinear model, while the filter estimates only a small perturbation around it:

$$q = \hat{q} \oplus \delta q. \tag{2.27}$$

where $\oplus$ denotes the composition operator between the nominal state and the error

state. In this work, since all state variables are represented in Euclidean coordinates, this operation reduces to standard vector addition, i.e., $q = \hat{q} + \delta q$.

Here, $\hat{q}$ represents the nominal state obtained through nonlinear integration, and δq denotes a small error state defined in a locally linear space. Instead of directly filtering the full state, the Kalman filter operates on this error representation, which is assumed to remain small.

The key idea is that linearization is performed around the nominal trajectory rather than around the true state estimate itself. Because the error state is typically close to zero, the linear approximation is often more accurate than in a standard EKF formulation. After each update, the estimated error δq is injected into the nominal state, and the error state is reset to zero.

This structure improves numerical stability and consistency, particularly for orientation variables and inertial sensor bias estimation. For this reason, the ESKF is widely adopted in inertial navigation and visual–inertial odometry systems [12, 13].

However, the separation between nominal and error states introduces additional implementation complexity and requires careful handling of state injection and covariance reset operations. The practical implications of adopting this formulation within different multisensor fusion architectures are discussed in Chapter 4.

2.7 Consistency Metrics: Innovation and NIS

To evaluate filter consistency, the innovation is defined as

$$y_k = z_k - \hat{z}_k, \tag{2.28}$$

where $\hat{z}_k$ denotes the predicted measurement obtained from the current state estimate. The innovation represents the discrepancy between actual and expected sensor observations.
Its covariance is given by

$$S_k = H_k P_{k|k-1} H_k^T + R, \tag{2.29}$$

which accounts for both propagated state uncertainty and measurement noise.
The Normalized Innovation Squared (NIS) is defined as

$$\text{NIS}_k = y_k^T S_k^{-1} y_k, \tag{2.30}$$

corresponding to the squared Mahalanobis distance of the innovation [3]. Intuitively, the NIS quantifies how large the measurement residual is relative to the uncertainty predicted by the filter.

Under correct modeling assumptions and properly tuned noise covariances, the innovation sequence follows a zero-mean Gaussian distribution with covariance S_k. Consequently, the NIS is chi-square distributed with degrees of freedom equal to the measurement dimension, enabling statistical consistency testing through hypothesis verification [3, 14].

Confidence bounds derived from the corresponding chi-square distribution define acceptance regions for statistically consistent innovations. Values consistently exceeding these bounds indicate filter overconfidence or modeling mismatch, whereas systematically small values suggest overly conservative uncertainty estimates. Such tests are commonly interpreted as innovation-based gating procedures for outlier detection based on the Mahalanobis distance [2].

For this reason, the NIS provides a principled statistical framework for estimator validation, tuning, and measurement consistency assessment.

2.8 Multisensor Fusion Architectures

Beyond the choice of filter, the architecture used to fuse sensor information strongly influences robustness and modularity.

2.8.1 Centralized Fusion

In a centralized EKF, all sensor measurements update a single shared state vector. This approach naturally accounts for cross-correlations between sensors and is optimal under Gaussian assumptions. However, the state dimension grows with the number of estimated variables, increasing computational cost and reducing modularity. Moreover, faults or modeling errors in one sensing modality may affect the entire estimation process.

2.8.2 Cascaded and Two-Stage Filtering

In cascaded architectures, multiple filters operate sequentially, where the output of one stage provides inputs or priors to another. Such architectures improve modularity and fault isolation, enabling independent tuning of subsystems. However, unless

cross-correlations between stages are explicitly modeled, the overall system may become statistically inconsistent [4].

2.8.3 Hybrid Architectures

Hybrid schemes combine cascaded estimation with partial feedback between stages, aiming to balance modularity and global consistency. By reintroducing selected correlations or shared information, hybrid approaches attempt to mitigate inconsistency while preserving architectural flexibility.

The architectural choice directly influences estimation consistency, computational complexity, robustness to sensor faults, and scalability to additional sensing modalities. While centralized approaches offer theoretical optimality under ideal assumptions, distributed and hybrid schemes provide practical advantages in modular robotic systems. A systematic comparison of these architectures under identical modeling and experimental conditions constitutes the core contribution of this thesis and is presented in Chapter 4.

3

System Modeling

This chapter presents the mathematical models adopted for state estimation. The objective is to define a consistent probabilistic description of the robot dynamics and sensor measurements, independently of the specific filtering architecture used for estimation.

3.1 State Representation

The robot state is defined as

$$q_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \\ b_k \end{bmatrix}, \tag{3.1}$$

where x_k and y_k denote the planar position of the robot in the world frame, θ_k represents the heading angle, and b_k is the gyroscope bias.

The inclusion of the bias state is necessary to compensate for slowly varying offsets in the inertial measurement unit (IMU). Without explicit bias estimation, small angular velocity offsets would accumulate over time, resulting in unbounded heading drift [3, 12].

3.2 Wheel Encoder Modeling

Wheel encoders measure incremental rotations of the left and right wheels. Let $\Delta s_{L,k}$ and $\Delta s_{R,k}$ denote the incremental displacements of the left and right wheels during the sampling interval T_s.

For a differential-drive robot, these increments yield

$$\Delta s_k = \frac{\Delta s_{L,k} + \Delta s_{R,k}}{2}, \tag{3.2}$$

$$\Delta \theta_{\text{enc},k} = \frac{\Delta s_{R,k} - \Delta s_{L,k}}{l}, \tag{3.3}$$

where Δs_k denotes the incremental forward displacement of the robot, $\Delta \theta_{\text{enc},k}$ is the encoder-based heading increment, and l is the wheelbase.

In this work, encoder-derived quantities Δs_k and $\Delta \theta_{\text{enc},k}$ are treated as inputs to the motion model. Their uncertainty is not handled through a separate measurement update, but instead incorporated into the process noise covariance. This modeling choice reflects the high update rate of encoder data and their direct role in state propagation.

3.3 Discrete-Time Motion Model

Using encoder-derived increments, the robot pose is propagated according to the unicycle kinematic model [5]. The sampling interval T_s is computed from encoder timestamps and may vary between consecutive updates due to asynchronous sensor measurements. All discrete-time quantities are therefore evaluated using the actual measured sampling interval.

A midpoint integration scheme is adopted to improve numerical accuracy:

$$x_{k+1} = x_k + \Delta s_k \cos\left(\theta_k + \frac{\Delta \theta_k}{2}\right), \tag{3.4}$$

$$y_{k+1} = y_k + \Delta s_k \sin\left(\theta_k + \frac{\Delta \theta_k}{2}\right), \tag{3.5}$$

$$\theta_{k+1} = \theta_k + \Delta \theta_k, \tag{3.6}$$

where the heading increment is defined as

$$\Delta\theta_k = \Delta\theta_{\text{enc},k} - b_k T_s. \tag{3.7}$$

Here, $\Delta\theta_{\text{enc},k}$ denotes the heading increment obtained from wheel encoder measurements, while b_k represents the gyroscope bias. Although the state propagation is driven by encoder-derived increments, the bias term is introduced to ensure consistency with the IMU measurement model. In particular, the gyroscope provides angular velocity measurements affected by bias, and the estimation of b_k relies on the discrepancy between encoder-based and inertial angular velocity estimates. This allows compensating for slowly varying bias effects that would otherwise lead to heading drift.

The gyroscope bias is modeled as a discrete-time random walk with diffusion proportional to the sampling interval [3]:

$$b_{k+1} = b_k + w_{b,k}, \qquad w_{b,k} \sim \mathcal{N}(0, \sigma_b^2 T_s). \tag{3.8}$$

This formulation ensures that the bias variance grows proportionally with elapsed time, preserving consistency under variable sampling intervals.

The complete state evolution can therefore be expressed compactly as

$$q_{k+1} = f(q_k, u_k) + w_k, \tag{3.9}$$

where the input vector is

$$u_k = \begin{bmatrix} \Delta s_k \\ \Delta\theta_{\text{enc},k} \end{bmatrix}, \tag{3.10}$$

and w_k represents zero-mean Gaussian process noise.

3.4 Linearization of the Motion Model

For Extended Kalman filtering, the Jacobian of the motion model with respect to the state is required. Denoting

$$F_k = \left.\frac{\partial f}{\partial q}\right|_{q_k, u_k},$$

the Jacobian matrix associated with the discrete-time model is

$$F_k = \begin{bmatrix} 1 & 0 & -\Delta s_k \sin\left(\theta_k + \frac{\Delta\theta_k}{2}\right) & 0 \\ 0 & 1 & \Delta s_k \cos\left(\theta_k + \frac{\Delta\theta_k}{2}\right) & 0 \\ 0 & 0 & 1 & -T_s \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.11}$$

3.5 Process Noise Modeling

Process noise accounts for wheel slip, encoder quantization effects, unmodeled dynamics, and bias drift. It is assumed Gaussian [3]:

$$w_k \sim \mathcal{N}(0, Q_k). \tag{3.12}$$

The covariance matrix is chosen diagonal:

$$Q_k = \text{diag}\left(\sigma_s^2, \sigma_s^2, \sigma_\theta^2, \sigma_b^2 T_s\right)., \tag{3.13}$$

reflecting the simplifying assumption that translational, rotational, and bias disturbances are mutually independent.

In practice, the process noise covariance is recomputed at each time step. The translational component σ_s is chosen proportional to the traveled distance Δs_k, while the rotational component σ_θ depends on the encoder-derived heading increment:

$$\sigma_s = \alpha_s |\Delta s_k|, \tag{3.14}$$

$$\sigma_\theta = \alpha_\theta + \beta_\theta |\Delta\theta_{\text{enc},k}|, \tag{3.15}$$

where α_s, α_θ, and β_θ are tuning parameters.

The bias diffusion term scales with the sampling interval T_s, yielding $Q_{b,k} = \sigma_b^2 T_s$.

3.6 IMU Measurement Model

The gyroscope measures the angular velocity of the robot about its vertical axis. The measurement is modeled as

$$z_{\text{imu},k} = \omega_k + b_k + v_{\text{imu},k}, \tag{3.16}$$

where ω_k is the true angular velocity and $v_{\text{imu},k}$ is zero-mean Gaussian noise [3]:

$$v_{\text{imu},k} \sim \mathcal{N}(0, R_{\text{imu}}). \tag{3.17}$$

In the adopted discrete-time formulation, the angular velocity used for prediction is approximated from encoder-derived heading increments:

$$\hat{\omega}_k = \frac{\Delta\theta_{\text{enc},k}}{T_s}. \tag{3.18}$$

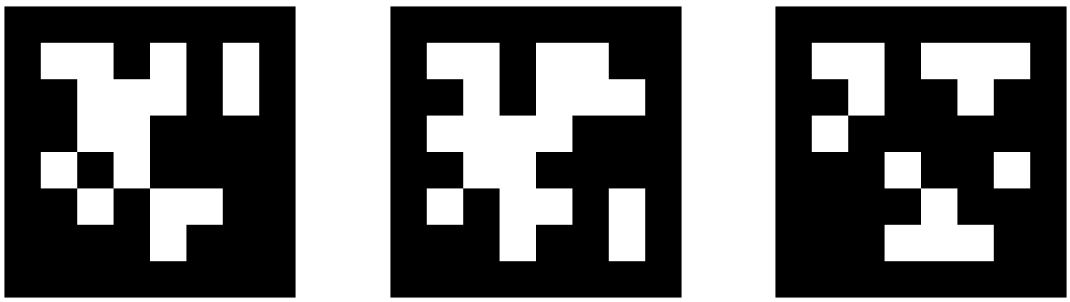
The predicted IMU measurement therefore becomes

$$\hat{z}_{\text{imu},k} = \frac{\Delta\theta_{\text{enc},k}}{T_s} + b_k. \tag{3.19}$$

This formulation enables direct estimation of the bias state through the discrepancy between encoder-based and gyroscope-based angular velocity estimates.

3.7 AprilTag Measurement Model

Absolute pose corrections are obtained through detection of AprilTag landmarks [7] placed at known world coordinates (x_t, y_t). Some example of AprilTags are reported in Figure 3.1.



(a) Number 0 (b) Number 1 (c) Number 2

Figure 3.1: Examples of AprilTags

The camera provides range–bearing measurements:

$$z_{\text{tag},k} = \begin{bmatrix} \phi_k \\ \rho_k \end{bmatrix}, \tag{3.20}$$

where ϕ_k is the bearing angle and ρ_k is the range to the detected landmark. The nonlinear measurement model is

$$\phi_k = \text{atan2}(y_t - y_k,\, x_t - x_k) - \theta_k, \tag{3.21}$$

$$\rho_k = \sqrt{(x_t - x_k)^2 + (y_t - y_k)^2}. \tag{3.22}$$

Measurement noise is assumed Gaussian:

$$v_{\text{tag},k} \sim \mathcal{N}(0, R_{\text{tag}}), \tag{3.23}$$

where R_{tag} is diagonal, reflecting independent bearing and range uncertainties.

Only landmarks within the camera field of view generate measurements. Data association is assumed known, as landmark identities are provided directly by the AprilTag detection algorithm.

3.8 Linearization of the Measurement Model

For EKF updates, the Jacobian of the AprilTag measurement model with respect to the state is required.

Let

$$\Delta x = x_t - x_k, \quad \Delta y = y_t - y_k, \quad r^2 = \Delta x^2 + \Delta y^2.$$

The measurement Jacobian is

$$H_k = \begin{bmatrix} \frac{\Delta y}{r^2} & -\frac{\Delta x}{r^2} & -1 & 0 \\ -\frac{\Delta x}{r} & -\frac{\Delta y}{r} & 0 & 0 \end{bmatrix}. \tag{3.24}$$

3.9 Observability Considerations

In the absence of landmark observations, the system becomes locally unobservable in position and heading, leading to unbounded drift due to encoder and bias uncertainty [13, 15]. AprilTag measurements provide absolute corrections that restore observability of the pose. Bias observability is achieved through the discrepancy between encoder-derived and IMU-measured angular velocity.

3.10 Modeling Assumptions

The modeling framework adopted in this chapter relies on the following assumptions:

- The robot operates on a planar surface and follows the unicycle kinematic model.
- Encoder-derived increments are treated as inputs, with their uncertainty incorporated into process noise.
- IMU and camera measurement noises are zero-mean Gaussian and mutually independent.
- Landmark positions are known and fixed.
- Data association is assumed correct.

These assumptions define a consistent probabilistic system model that forms the basis for the filter architectures presented in the next chapter.

4

Kalman Filter Architectures

This chapter presents the different Kalman Filter (KF) architectures developed and implemented for multisensor localization of the Duckiebot platform. All architectures operate on the same probabilistic system model described in Chapter 3, and use identical noise parameters and sensor data. Consequently, performance differences arise exclusively from architectural design choices rather than from modeling assumptions [4, 16].

The architectures considered are:

- A centralized Extended Kalman Filter (EKF)
- A cascaded dual-filter architecture
- An Error-State Kalman Filter (ESKF)
- A hybrid partially coupled architecture

Each structure differs in how the system state is partitioned and how cross-correlations between state components are represented.

4.1 Centralized Extended Kalman Filter

4.1.1 State and Covariance Structure

The centralized architecture estimates the full state vector jointly:

$$q_k = \begin{bmatrix} x_k & y_k & \theta_k & b_k \end{bmatrix}^T, \tag{4.1}$$

with covariance matrix

$$P_k \in \mathbb{R}^{4\times4}. \tag{4.2}$$

All cross-correlations between position, heading, and bias states are explicitly maintained within the covariance matrix.

4.1.2 Prediction Step

Wheel encoder ticks are first converted into incremental wheel displacements:

$$\Delta s_L = \Delta \text{ticks}_L \cdot \frac{2\pi r}{N_{\text{enc}}}, \tag{4.3}$$

$$\Delta s_R = \Delta \text{ticks}_R \cdot \frac{2\pi r}{N_{\text{enc}}}, \tag{4.4}$$

where r is the wheel radius and N_{enc} the encoder resolution.

The incremental robot motion is then computed as

$$\Delta s_k = \frac{\Delta s_L + \Delta s_R}{2}, \tag{4.5}$$

$$\Delta\theta_{\text{enc},k} = \frac{\Delta s_R - \Delta s_L}{l}. \tag{4.6}$$

The bias-corrected heading increment is defined as:

$$\Delta\theta_k = \Delta\theta_{\text{enc},k} - b_k T_s. \tag{4.7}$$

Using midpoint integration, the predicted state becomes

$$x_{k+1} = x_k + \Delta s_k \cos\left(\theta_k + \frac{\Delta\theta_k}{2}\right), \tag{4.8}$$

$$y_{k+1} = y_k + \Delta s_k \sin\left(\theta_k + \frac{\Delta\theta_k}{2}\right), \tag{4.9}$$

$$\theta_{k+1} = \theta_k + \Delta\theta_k, \tag{4.10}$$

$$b_{k+1} = b_k + w_{b,k}. \tag{4.11}$$

The corresponding Jacobian matrix is

$$F_k = \begin{bmatrix} 1 & 0 & -\Delta s_k \sin\left(\theta_k + \frac{\Delta\theta_k}{2}\right) & 0 \\ 0 & 1 & \Delta s_k \cos\left(\theta_k + \frac{\Delta\theta_k}{2}\right) & 0 \\ 0 & 0 & 1 & -T_s \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.12}$$

The process noise covariance is chosen adaptively as

$$\sigma_s = \alpha_s |\Delta s_k|, \tag{4.13}$$

$$\sigma_\theta = \alpha_\theta + \beta_\theta |\Delta\theta_{\text{enc},k}|, \tag{4.14}$$

leading to

$$Q_k = \text{diag}\left(\sigma_s^2, \sigma_s^2, \sigma_\theta^2, \sigma_b^2 T_s\right). \tag{4.15}$$

σ_s is modeled as proportional to the traveled distance, reflecting the fact that translational uncertainty mainly arises from motion-dependent effects such as wheel slip and encoder quantization.

In contrast, the rotational component σ_θ includes both a constant term and a motion-dependent term. The constant term α_θ accounts for baseline uncertainty due to factors such as sensor noise and bias estimation errors, while the proportional term captures motion-induced uncertainty associated with encoder-based heading increments.

The numerical values of the coefficients are reported in Chapter 5.

In the following measurement updates, consistency is assessed using the Normalized Innovation Squared (NIS), introduced in Section 2.7. The NIS is computed for each sensor modality and used to monitor filter consistency and support measurement validation.

4.1.3 IMU Update

The gyroscope provides an angular velocity measurement

$$z_k = \omega_{\text{imu},k}. \tag{4.16}$$

In the implemented formulation, the predicted measurement is computed using the encoder-derived heading increment:

$$\hat{z}_k = \frac{\Delta\theta_{\text{enc},k}}{T_s} + \hat{b}_{k|k-1}. \tag{4.17}$$

The encoder-derived angular velocity is used as an approximation of the true angular rate. Consequently, encoder uncertainty—arising from factors such as wheel slip, encoder quantization, and unmodeled kinematic effects—is implicitly absorbed into the effective measurement noise term. This assumes that such errors remain small relative to gyroscope bias drift over the considered time scale.

The innovation is therefore

$$y_k = z_k - \hat{z}_k. \tag{4.18}$$

Since the measurement depends only on the bias state, the Jacobian matrix is

$$H_{\text{imu}} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.19}$$

Although the measurement equation is directly sensitive only to the bias state, the heading and position states are indirectly corrected through the cross-covariance terms contained in P_k.

The innovation covariance is

$$S_k = H_{\text{imu}} P_{k|k-1} H_{\text{imu}}^T + R_{\text{imu}}, \tag{4.20}$$

and the Kalman gain is

$$K_k = P_{k|k-1} H_{\text{imu}}^T S_k^{-1}. \tag{4.21}$$

The state estimate is updated as

$$\hat{q}_k = \hat{q}_{k|k-1} + K_k y_k, \tag{4.22}$$

followed by the covariance update

$$P_k = (I - K_k H_{\text{imu}}) P_{k|k-1}. \tag{4.23}$$

The NIS associated with the IMU update is computed as

$$\text{NIS}_k = y_k^T S_k^{-1} y_k, \tag{4.24}$$

and is used to monitor the statistical consistency of the inertial measurements and to verify the correctness of the assumed noise model.

In implementation, a numerically stabilized covariance update may be adopted to preserve symmetry and positive definiteness.

The IMU update is applied only after at least one encoder motion increment has been processed. This ensures consistency between encoder-based and inertial angular velocity estimates and avoids bias updates in the absence of motion.

4.1.4 AprilTag Update

Absolute pose corrections are obtained from AprilTag landmark detections. For a detected tag located at known world coordinates (x_t, y_t), the camera provides range–bearing measurements

$$z_k = \begin{bmatrix} \phi_k \\ \rho_k \end{bmatrix}. \tag{4.25}$$

Given the predicted state estimate $\hat{q}_{k|k-1} = \begin{bmatrix} \hat{x}_{k|k-1} & \hat{y}_{k|k-1} & \hat{\theta}_{k|k-1} & \hat{b}_{k|k-1} \end{bmatrix}^T$, the expected measurement is

$$\hat{z}_k = \begin{bmatrix} \text{atan2}(y_t - \hat{y}_{k|k-1},\ x_t - \hat{x}_{k|k-1}) - \hat{\theta}_{k|k-1} \\ \sqrt{(x_t - \hat{x}_{k|k-1})^2 + (y_t - \hat{y}_{k|k-1})^2} \end{bmatrix}. \tag{4.26}$$

Defining

$$\Delta x = x_t - \hat{x}_{k|k-1}, \qquad \Delta y = y_t - \hat{y}_{k|k-1}, \qquad r^2 = \Delta x^2 + \Delta y^2,$$

the Jacobian matrix with respect to the state is

$$H_{\text{tag},k} = \begin{bmatrix} \frac{\Delta y}{r^2} & -\frac{\Delta x}{r^2} & -1 & 0 \\ -\frac{\Delta x}{r} & -\frac{\Delta y}{r} & 0 & 0 \end{bmatrix}. \tag{4.27}$$

The measurement is independent of the bias state, as reflected by the zero entries in the fourth column of $H_{\text{tag},k}$. The innovation is computed as

$$y_k = z_k - \hat{z}_k, \tag{4.28}$$

with angular residual normalized to the interval $(-\pi, \pi]$.

The innovation covariance is

$$S_k = H_{\text{tag},k} P_{k|k-1} H_{\text{tag},k}^T + R_{\text{tag}}, \tag{4.29}$$

and the Kalman gain is

$$K_k = P_{k|k-1} H_{\text{tag},k}^T S_k^{-1}. \tag{4.30}$$

The state estimate is updated as

$$\hat{q}_k = \hat{q}_{k|k-1} + K_k y_k, \tag{4.31}$$

followed by covariance update

$$P_k = (I - K_k H_{\text{tag},k}) P_{k|k-1}. \tag{4.32}$$

The corresponding NIS is computed as

$$\text{NIS}_k = y_k^T S_k^{-1} y_k. \tag{4.33}$$

The NIS is used to assess the statistical consistency of the filter and to support the tuning of noise covariance parameters. In particular, its empirical distribution is compared against the theoretical chi-square bounds to verify whether the assumed process and measurement noise levels are consistent with the observed estimation errors.

In addition, for vision-based measurements, the NIS is used to perform chi-square gating, enabling the rejection of outlier observations.

For a two-dimensional measurement, NIS_k follows a chi-square distribution with two degrees of freedom under correct noise assumptions. Measurements with NIS_k exceeding the 99% confidence threshold ($\chi^2_{2,0.99} = 9.21$) are rejected to mitigate the effect of outliers.

Additional geometric and numerical validity checks are performed prior to the update step to avoid singular configurations and ill-conditioned covariance matrices.

4.1.5 Structural Characteristics

Because all state components are estimated jointly, cross-covariance terms between bias and pose states are naturally preserved.

This joint formulation ensures that heading and bias states are indirectly corrected through their cross-correlation with position states [3]. While statistically consistent under correct modeling assumptions, the centralized structure increases computational complexity and tightly couples all sensing modalities within a single estimator.

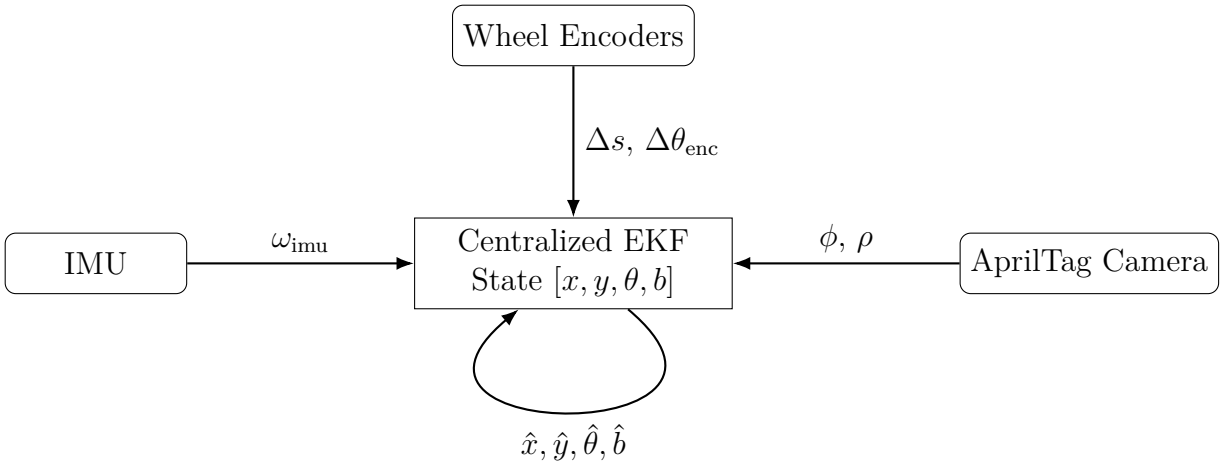


Figure 4.1: Centralized EKF architecture. Encoders drive state propagation, while IMU and vision provide measurement updates.

4.2 Cascaded Architecture

4.2.1 State Decomposition

In the cascaded approach, the full state is partitioned into two independent filters:

$$q_k^{(1)} = b_k, \tag{4.34}$$

$$q_k^{(2)} = \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^T. \tag{4.35}$$

The corresponding covariance matrices are

$$P_k^{(1)} \in \mathbb{R}^{1\times 1}, \qquad P_k^{(2)} \in \mathbb{R}^{3\times 3}. \tag{4.36}$$

The chosen state partition reflects a deliberate architectural trade-off between simplicity and information coupling. In the cascaded structure, the first filter is dedicated exclusively to gyroscope bias estimation, while the second filter estimates the full pose, including heading.

This choice is motivated by the role of the bias as a slowly varying sensor parameter that can be estimated independently from the robot pose. By isolating the bias into a dedicated scalar filter, the estimation problem is significantly simplified, and the IMU measurement model becomes linear and directly observable.

In contrast, the heading state θ_k is retained within the pose filter because it is strongly coupled with the planar motion dynamics. The evolution of x_k and

y_k depends explicitly on the heading through the nonlinear motion model, and separating θ_k from the position states would break this natural coupling, leading to a less coherent propagation model.

As a result, the cascaded architecture preserves the coupling between position and heading within a single filter, while treating the bias as an external parameter that influences the system through feedforward correction.

An alternative partitioning, where θ_k and b_k are jointly estimated in the same filter, is considered in the hybrid architecture presented later in this chapter. That formulation reintroduces partial coupling between heading and bias estimation, at the cost of increased complexity.

Unlike the centralized formulation, cross-covariance terms between bias and pose states are not represented.

4.2.2 Bias Estimation Filter

The first filter estimates the gyroscope bias independently. The bias is modeled as a random walk:

$$b_{k+1} = b_k + w_{b,k}, \qquad w_{b,k} \sim \mathcal{N}(0, Q_b). \tag{4.37}$$

The covariance propagation reduces to

$$P^{(1)}_{k|k-1} = P^{(1)}_{k-1|k-1} + Q_b. \tag{4.38}$$

The IMU measurement is reformulated as

$$z_k = \omega_{\text{imu},k} - \frac{\Delta\theta_{\text{enc},k}}{T_s}, \tag{4.39}$$

which yields the linear measurement model

$$\hat{z}_k = \hat{b}_{k|k-1}. \tag{4.40}$$

The innovation is

$$y_k = z_k - \hat{z}_k, \tag{4.41}$$

with innovation covariance

$$S_k = P^{(1)}_{k|k-1} + R_{\text{imu}}. \tag{4.42}$$

The Kalman gain becomes

$$K_k = \frac{P^{(1)}_{k|k-1}}{S_k}. \tag{4.43}$$

The update equations reduce to

$$\hat{b}_{k|k} = \hat{b}_{k|k-1} + K_k y_k, \tag{4.44}$$
$$P^{(1)}_{k|k} = (1 - K_k) P^{(1)}_{k|k-1}. \tag{4.45}$$

4.2.3 Pose Estimation Filter

The second filter estimates the pose using encoder-based propagation and landmark updates.

The bias estimate $\hat{b}_k$ from the first filter is used during heading propagation:

$$\Delta\theta_k = \Delta\theta_{\text{enc},k} - \hat{b}_{k|k} T_s. \tag{4.46}$$

To account for bias uncertainty in heading propagation, the heading variance is inflated as

$$P^{(2)}_{33,k|k-1} = P^{(2)}_{33,k|k-1} + T_s^2 P^{(1)}_{k|k}. \tag{4.47}$$

The landmark update is performed only on the pose state. The measurement Jacobian is

$$H_{\text{tag},k} \in \mathbb{R}^{2\times 3}, \tag{4.48}$$

and the update affects only x_k, y_k, θ_k.

Unlike the centralized architecture, the bias state is not corrected during landmark updates, since no cross-covariance terms are maintained.

4.2.4 Information Flow and Limitations

The cascaded structure introduces a one-way information flow:

- The bias estimate influences pose propagation.
- Bias uncertainty inflates heading variance.
- Landmark updates correct only the pose filter.
- No feedback from pose corrections to bias estimation exists.

This structural simplification approximates statistical coupling through variance injection rather than explicit cross-covariance tracking. Its impact on consistency depends on the relative magnitude of bias uncertainty and motion excitation.

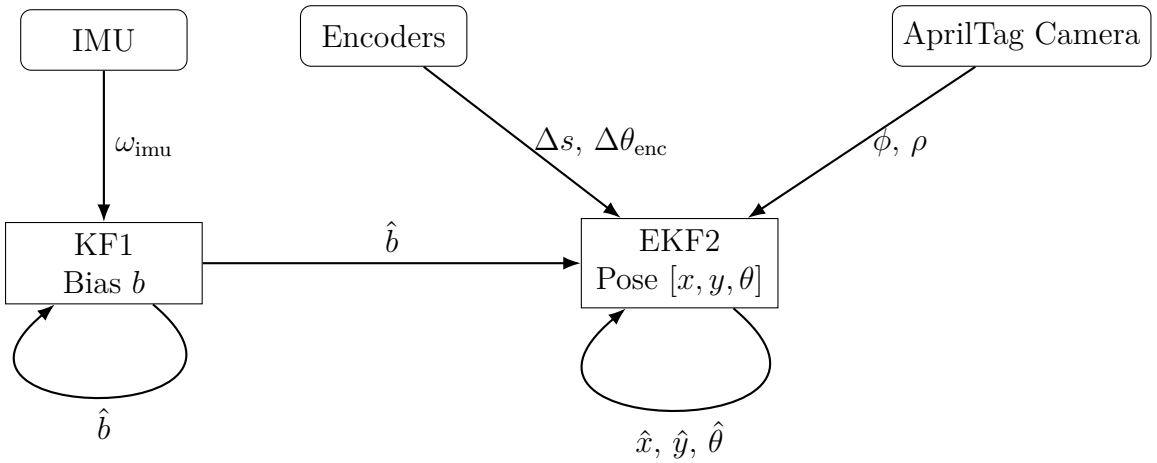


Figure 4.2: The cascaded architecture exhibits one-way information flow: bias affects pose propagation, while pose corrections do not influence bias estimation.

4.3 Error-State KF

4.3.1 Nominal and Error State Definition

In the Error-State KF formulation, the state estimate is decomposed into:

- A nominal state $\hat{q}_k$, propagated using the full nonlinear model.
- A small error state δq_k, estimated using a linearized model.

The nominal state is defined as

$$\hat{q}_k = \begin{bmatrix} \hat{x}_k & \hat{y}_k & \hat{\theta}_k & \hat{b}_k \end{bmatrix}^T, \tag{4.49}$$

while the error state is

$$\delta q_k = \begin{bmatrix} \delta x_k & \delta y_k & \delta\theta_k & \delta b_k \end{bmatrix}^T. \tag{4.50}$$

The total state is related by

$$q_k = \hat{q}_k + \delta q_k, \tag{4.51}$$

since all variables are represented in Euclidean coordinates.

4.3.2 Nominal State Propagation

The nominal state is propagated using the nonlinear motion model:

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + \Delta s_k \cos\left(\hat{\theta}_{k-1|k-1} + \frac{\Delta\theta_k}{2}\right), \tag{4.52}$$

$$\hat{y}_{k|k-1} = \hat{y}_{k-1|k-1} + \Delta s_k \sin\left(\hat{\theta}_{k-1|k-1} + \frac{\Delta\theta_k}{2}\right), \tag{4.53}$$

$$\hat{\theta}_{k|k-1} = \hat{\theta}_{k-1|k-1} + \Delta\theta_k, \tag{4.54}$$

with

$$\Delta\theta_k = \Delta\theta_{\text{enc},k} - \hat{b}_{k-1|k-1} T_s. \tag{4.55}$$

4.3.3 Error Covariance Propagation

The error covariance is propagated using the linearized system dynamics:

$$P_{\delta q,k|k-1} = F_k P_{\delta q,k-1|k-1} F_k^T + Q_k, \tag{4.56}$$

Although the covariance propagation equation has the same form as in the centralized EKF, the two formulations differ in the point around which the system is linearized and in the quantity being estimated.

In the centralized EKF, the nonlinear motion and measurement models are directly linearized with respect to the full state vector q_k, and the filter propagates and updates the state estimate itself.

In contrast, the Error-State KF linearizes the system dynamics with respect to the error state δq_k, defined as the deviation from the nominal trajectory $\hat{q}_k$. The linearization is therefore performed around the nominal state, which follows the nonlinear model, while the filter estimates only small corrections.

As a result, although the propagation equations appear structurally similar, the centralized EKF operates on the full state, whereas the Error-State KF operates on a locally linear error representation. This distinction improves the validity of the

linear approximation, since the error state remains small even when the nominal trajectory evolves nonlinearly.

In the considered planar localization problem, the practical differences between the centralized EKF and the Error-State KF are limited. Since all states are represented in Euclidean form and no manifold constraints are involved, the error-state formulation reduces to an additive correction of the nominal state.

As a result, the Jacobians are evaluated at the same operating point and both filters yield nearly identical numerical behavior. The main distinction therefore lies in the interpretation of the estimated quantities rather than in the structure of the resulting equations.

4.3.4 IMU Error Update

For the IMU update, the innovation is directly computed as the residual between the IMU measurement and its nominal prediction:

$$y_k = \omega_{\text{imu},k} - \left(\frac{\Delta\theta_{\text{enc},k}}{T_s} + \hat{b}_{k|k-1} \right). \tag{4.57}$$

The measurement Jacobian in the error-state space is

$$H_{\text{imu}} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.58}$$

The Kalman gain is computed as

$$K_k = P_{\delta q,k|k-1} H_{\text{imu}}^T S_k^{-1}, \tag{4.59}$$

with

$$S_k = H_{\text{imu}} P_{\delta q,k|k-1} H_{\text{imu}}^T + R_{\text{imu}} \tag{4.60}$$

4.3.5 AprilTag Error Update

The landmark innovation is computed using the nominal state:

$$y_k = z_k - \hat{z}_k(\hat{q}_k). \tag{4.61}$$

The measurement update is performed entirely in the error-state space, leaving the nominal state unchanged until correction injection. The Jacobian is identical to the one derived in Chapter 3, evaluated at the nominal state $\hat{q}_k$.

The innovation covariance is computed as

$$S_k = H_k P_{\delta q,k|k-1} H_k^T + R_k, \tag{4.62}$$

and the Kalman gain is

$$K_k = P_{\delta q,k|k-1} H_k^T S_k^{-1}. \tag{4.63}$$

4.3.6 Injection and Reset

After computing the correction $\delta q_k = K_k y_k$, the nominal state is updated as

$$\hat{q}_k = \hat{q}_k + \delta q_k. \tag{4.64}$$

The error state is then reset to zero:

$$\delta q_k = 0. \tag{4.65}$$

The covariance is updated prior to injection using the Joseph stabilized form

$$P_{\delta q,k|k} = (I - K_k H_k) P_{\delta q,k|k-1} (I - K_k H_k)^T + K_k R_k K_k^T. \tag{4.66}$$

Since an additive error representation is adopted, no additional covariance transformation is required after injection.

4.3.7 Structural Characteristics

The ESKF preserves the full state covariance and therefore maintains all cross-correlations between position, heading, and bias states, similarly to the centralized EKF.

However, the separation between nominal and error states offers several advantages:

- Linearization is performed around the nominal trajectory, which typically remains close to the true state.
- The error state remains small, improving the validity of first-order approximations [10, 12].
- Numerical stability is improved, particularly for orientation variables.
- Bias estimation is naturally integrated within the same covariance structure.

In contrast to the cascaded architecture, the ESKF does not approximate cross-correlation through variance inflation, but explicitly represents it within the error covariance matrix.

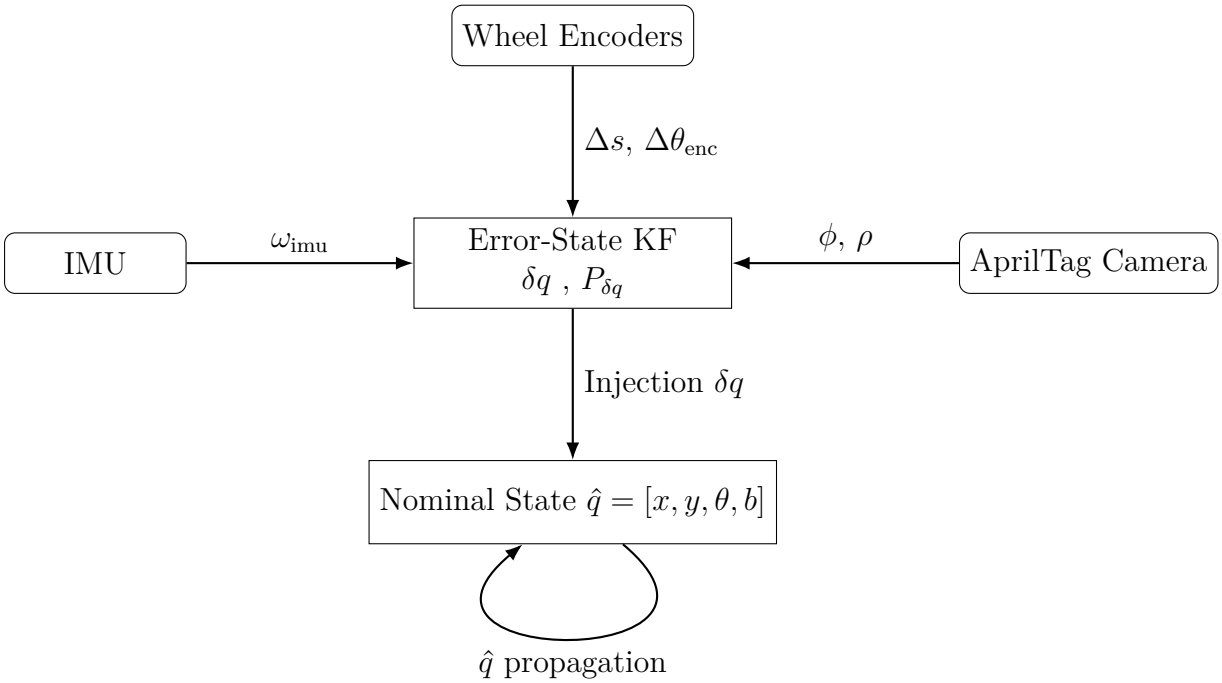


Figure 4.3: Error-State KF architecture. Measurements update the error-state, which is injected into the nominal state estimate. Full covariance is maintained in error-state form.

4.4 Hybrid Architecture

4.4.1 State Partitioning

In the hybrid architecture, the full state vector is partitioned into two interacting filters:

$$q_k^{(1)} = \begin{bmatrix} \theta_k \\ b_k \end{bmatrix}, \qquad q_k^{(2)} = \begin{bmatrix} x_k \\ y_k \end{bmatrix}. \tag{4.67}$$

The corresponding covariance matrices are

$$P_k^{(1)} \in \mathbb{R}^{2\times 2}, \qquad P_k^{(2)} \in \mathbb{R}^{2\times 2}. \tag{4.68}$$

Unlike the centralized architecture, cross-covariance terms between planar position and heading–bias states are not explicitly maintained. However, information exchange between the two filters is achieved through uncertainty injection and feedback updates.

4.4.2 Heading–Bias Filter

The first filter estimates heading and gyroscope bias using encoder propagation and IMU measurements.

Prediction

From encoder-derived increments, the heading is propagated as

$$\hat{\theta}_{k|k-1} = \hat{\theta}_{k-1|k-1} + \Delta\theta_{\text{enc},k} - \hat{b}_{k-1|k-1} T_s. \tag{4.69}$$

This formulation follows directly from the bias-compensated angular increment model introduced in equation 3.7, where the encoder-based heading increment is used as a reference and the gyroscope bias is explicitly estimated and removed to ensure consistency with the IMU measurement model.

The bias is modeled as a random walk, capturing slow time variations:

$$b_{k+1} = b_k + w_{b,k}, \qquad w_{b,k} \sim \mathcal{N}(0, Q_b). \tag{4.70}$$

In the prediction step, this leads to

$$\hat{b}_{k|k-1} = \hat{b}_{k-1|k-1}, \tag{4.71}$$

The linearized state transition matrix is

$$F_k^{(1)} = \begin{bmatrix} 1 & -T_s \\ 0 & 1 \end{bmatrix}. \tag{4.72}$$

The covariance propagation follows

$$P_{k|k-1}^{(1)} = F_k^{(1)} P_{k-1|k-1}^{(1)} (F_k^{(1)})^T + Q_k^{(1)}, \tag{4.73}$$

where $Q_k^{(1)}$ models heading process noise and bias drift.

IMU Update

The gyroscope provides the measurement

$$z_k = \omega_{\text{imu},k}. \tag{4.74}$$

The predicted measurement is

$$\hat{z}_k = \frac{\Delta\theta_{\text{enc},k}}{T_s} + \hat{b}_{k|k-1}. \tag{4.75}$$

The innovation becomes

$$y_k = z_k - \hat{z}_k. \tag{4.76}$$

The measurement Jacobian is

$$H_k^{(1)} = \begin{bmatrix} 0 & 1 \end{bmatrix}. \tag{4.77}$$

The innovation covariance and Kalman gain are

$$S_k = H_k^{(1)} P_{k|k-1}^{(1)} (H_k^{(1)})^T + R_{\text{imu}}, \tag{4.78}$$

$$K_k = P_{k|k-1}^{(1)} (H_k^{(1)})^T S_k^{-1}. \tag{4.79}$$

The state and covariance are updated as

$$\hat{q}_{k|k}^{(1)} = \hat{q}_{k|k-1}^{(1)} + K_k y_k, \tag{4.80}$$

$$P_{k|k}^{(1)} = (I - K_k H_k^{(1)}) P_{k|k-1}^{(1)}. \tag{4.81}$$

4.4.3 Position Filter

The second filter estimates planar position using heading provided by the first filter.

Prediction

Using the heading estimate $\hat{\theta}_{k|k}$ from EKF-1, the planar motion model is

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + \Delta s_k \cos(\hat{\theta}_{k|k}) \tag{4.82}$$

$$\hat{y}_{k|k-1} = \hat{y}_{k-1|k-1} + \Delta s_k \sin(\hat{\theta}_{k|k}). \tag{4.83}$$

The state transition Jacobian is

$$F_k^{(2)} = I_2. \tag{4.84}$$

Heading Uncertainty Injection

Since heading uncertainty is not jointly represented, its effect on planar motion must be approximated.

Let

$$\sigma_\theta^2 = P_{k|k}^{(1)}(1,1). \tag{4.85}$$

The additional covariance contribution induced by heading uncertainty is approximated as

$$Q_{\theta,k} = \Delta s_k^2 \sigma_\theta^2 I_2. \tag{4.86}$$

The covariance propagation therefore becomes

$$P_{k|k-1}^{(2)} = F_k^{(2)} P_{k-1|k-1}^{(2)} (F_k^{(2)})^T + Q_k^{(2)} + Q_{\theta,k}, \tag{4.87}$$

where $Q_k^{(2)}$ models translational process noise.

4.4.4 AprilTag Update

Landmark observations update only the planar position state.

For a tag located at (x_t, y_t), the predicted measurement is

$$\hat{z}_k = \begin{bmatrix} \operatorname{atan2}(y_t - \hat{y}_{k|k-1},\ x_t - \hat{x}_{k|k-1}) - \hat{\theta}_{k|k} \\ \sqrt{(x_t - \hat{x}_{k|k-1})^2 + (y_t - \hat{y}_{k|k-1})^2} \end{bmatrix}. \tag{4.88}$$

The Jacobian with respect to $[x_k, y_k]^T$ is

$$H_k^{(2)} = \begin{bmatrix} \frac{\Delta y}{r^2} & -\frac{\Delta x}{r^2} \\ -\frac{\Delta x}{r} & -\frac{\Delta y}{r} \end{bmatrix}. \tag{4.89}$$

with

$$\Delta x = x_t - \hat{x}_{k|k-1}, \quad \Delta y = y_t - \hat{y}_{k|k-1}, \quad r^2 = \Delta x^2 + \Delta y^2. \tag{4.90}$$

The innovation covariance and gain are

$$S_k = H_k^{(2)} P_{k|k-1}^{(2)} (H_k^{(2)})^T + R_{\text{tag}}, \tag{4.91}$$

$$K_k = P_{k|k-1}^{(2)} (H_k^{(2)})^T S_k^{-1}. \tag{4.92}$$

The updated state and covariance are

$$y_k = z_k - \hat{z}_k \tag{4.93}$$

$$\hat{q}^{(2)}_{k|k} = \hat{q}^{(2)}_{k|k-1} + K_k y_k, \tag{4.94}$$

$$P^{(2)}_{k|k} = (I - K_k H^{(2)}_k) P^{(2)}_{k|k-1}. \tag{4.95}$$

4.4.5 Heading Feedback from Vision

Since landmark observations provide both range and bearing information, they also contain implicit information about the robot heading. This information can be exploited to improve the heading estimate through a feedback correction.

An equivalent heading measurement is obtained by combining the estimated robot position with the observed bearing:

$$\theta_{\text{cam}} = \text{atan2}(y_t - \hat{y}_{k|k},\ x_t - \hat{x}_{k|k}) - \phi_k. \tag{4.96}$$

This quantity represents the heading that is consistent with the current position estimate and the observed landmark direction. The corresponding innovation is defined as

$$y^{\text{fb}}_k = \theta_{\text{cam}} - \hat{\theta}_{k|k}. \tag{4.97}$$

The feedback is incorporated as an additional scalar measurement affecting only the heading component of the first filter. The associated measurement model is linear, with Jacobian

$$H^{(1)}_{\text{fb}} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{4.98}$$

Using this model, a standard Kalman correction step is applied to the heading–bias filter. This update is performed after the IMU correction, resulting in a sequential multi-sensor update at the same time step.

This feedback mechanism introduces a partial coupling between the two filters: although cross-covariances are not explicitly maintained, vision-based position updates indirectly improve heading estimation. As a result, the hybrid architecture recovers part of the information flow that would otherwise be lost in a fully decoupled formulation.

4.4.6 Structural Implications

The hybrid architecture introduces bidirectional information flow:

- Heading uncertainty affects planar position propagation through covariance injection.
- Vision-based position updates indirectly correct heading through feedback.

However, the full joint covariance between $[x, y]$ and $[\theta, b]$ is not explicitly represented. Cross-correlations are approximated rather than maintained directly.

Compared to the cascaded architecture, the hybrid scheme improves consistency by incorporating partial feedback. Although the hybrid architecture does not explicitly maintain the full joint covariance between position and heading–bias states, the combination of uncertainty injection and vision-based feedback allows it to approximate cross-correlation effects in a structured manner [16].

The resulting estimation behavior can approach that of centralized architectures while preserving modular decomposition.

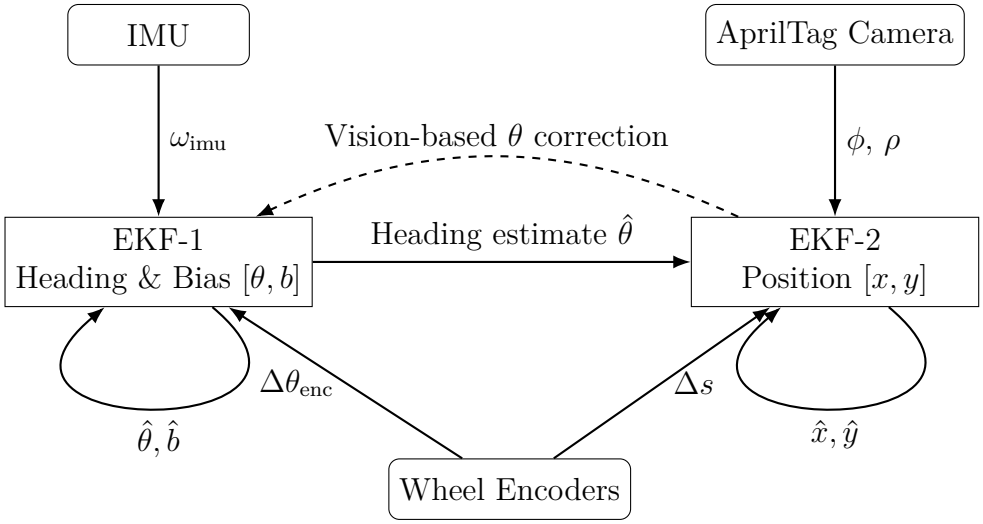


Figure 4.4: Hybrid-Cascaded EKF architecture. Heading–bias and position are estimated in separate filters. Heading uncertainty influences position propagation, while vision-based updates provide partial feedback to heading estimation. Full joint covariance is not maintained.

4.5 Comparative Structural Analysis

4.5.1 Covariance Structure Comparison

The considered architectures differ primarily in how cross-correlations between position, heading, and bias states are represented:

- Centralized EKF: Full joint covariance matrix is maintained.

- Cascaded EKF: No explicit cross-covariance between sub-filters; coupling is approximated through variance injection.
- ESKF: Full covariance maintained in error-state form.
- Hybrid-Cascaded EKF: Partial coupling through uncertainty injection and feedback, without full joint covariance tracking.

The centralized EKF and ESKF explicitly preserve all statistical dependencies, while partitioned architectures rely on approximations.

4.5.2 Computational Complexity

The centralized EKF and ESKF propagate a 4×4 covariance matrix at each step.

Partitioned architectures reduce matrix dimensionality to 2×2 blocks, decreasing computational cost. However, this reduction is achieved at the expense of neglecting or approximating cross-correlations.

4.5.3 Expected Consistency Behavior

Architectures that neglect explicit cross-covariance terms may, in principle, be more sensitive to modeling inaccuracies [4], particularly when strong dependencies exist between sub-states.

However, when uncertainty injection and feedback mechanisms are properly designed, partitioned architectures can achieve consistency levels comparable to fully centralized formulations.

The hybrid architecture introduces bidirectional information exchange between sub-filters, which mitigates the loss of explicit cross-correlation. Although full joint covariance tracking is not performed, carefully tuned uncertainty propagation and feedback correction can preserve statistical consistency in practice.

The practical implications of these structural differences are evaluated experimentally in Chapter 6 and Chapter 7.

Table 4.1: Structural comparison of the considered Kalman filter architectures.

Architecture	**Joint Cov.**	**Cross-Corr. Handling**	**State Dim.**
Centralized EKF	Yes	Explicit	4
Cascaded EKF	No	Variance injection	$1 + 3$
ESKF	Yes (error)	Explicit	4
Hybrid-Cascaded EKF	No	Partial (feedback)	$2 + 2$

5

Model Implementation

This chapter describes the practical implementation of the probabilistic model and the different Kalman filter architectures presented in Chapter 4. While Chapter 3 introduced the mathematical formulation, the focus here is on engineering choices, parameter selection, data handling, and numerical considerations that influence estimator behavior.

5.1 Software Framework

5.1.1 Development Environment

The complete localization pipeline was implemented using a combination of Robot Operating System (ROS) based data acquisition and offline MATLAB evaluation.

Sensor data were collected on the Duckiebot platform using ROS nodes, while all Kalman filter architectures were implemented and executed offline in MATLAB. This separation allowed identical datasets to be reused across all filter variants, ensuring a fair architectural comparison.

The implementation follows a modular structure:

- ROS nodes handle sensor acquisition and preprocessing.
- A unified logging node aggregates all measurements into a single topic.
- MATLAB scripts implement the different Kalman filter architectures.

All filters share identical motion and measurement models and differ only in state representation and covariance structure.

5.1.2 ROS-Based Sensor Acquisition

Wheel encoder ticks, IMU measurements, AprilTag detections, and Vicon ground-truth data were collected using dedicated ROS nodes.

AprilTag Detection Node A custom ROS node was developed for AprilTag detection. Compressed camera images are undistorted using intrinsic calibration parameters and processed using the `apriltag` library. Detected tag poses are estimated in the camera frame and transformed to the robot base frame using a fixed extrinsic calibration matrix.

Each detection includes:

- Tag identifier
- 3D relative translation
- Quaternion orientation
- Detection confidence metrics

Unified Sensor Logger To simplify offline processing, a dedicated logging node aggregates all sensor measurements into a single topic (`/raw_sensor_log`) using a standardized numeric format.

Each message contains:

- Timestamp (seconds)
- Measurement type identifier
- Measurement values

The following measurement types are recorded:

- Left encoder ticks
- Right encoder ticks
- IMU angular velocity (ω_z)
- AprilTag relative position
- Vicon ground-truth pose

This unified logging strategy guarantees strict timestamp consistency across all architectures during offline replay.

5.1.3 Offline Data Processing

Recorded ROS bag files are loaded into MATLAB for filter evaluation.

During preprocessing:

- Messages are sorted by timestamp.
- Time intervals are computed from consecutive measurements.
- Encoder ticks are converted to metric displacements.
- Angular quantities are normalized to $(-\pi, \pi]$.

All architectures process the same measurement sequence without modification, ensuring that performance differences arise solely from structural estimator design.

5.2 Dataset Description

5.2.1 Experimental Platform

Experiments were conducted using a Duckiebot DB21J platform, a small-scale differential-drive robot developed within the Duckietown project for autonomous driving research and education [17]. A picture of the experimental platform is shown in Fig. 5.1.

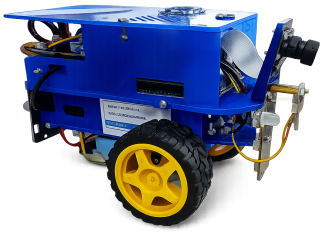


Figure 5.1: Duckiebot DB21J platform used in the experiments.

The robot is a differential-drive vehicle equipped with wheel encoders, an onboard IMU (MPU-9250), and a monocular forward-facing camera.

The main geometric parameters of the robot are:

- Wheel radius: approximately $r \approx 0.03\,\mathrm{m}$
- Wheel baseline: approximately $l \approx 0.10\,\mathrm{m}$

The camera is rigidly mounted on the chassis. Intrinsic and extrinsic calibration parameters were obtained from the standard camera calibration procedure.

Wheel Encoders Each wheel is equipped with an absolute encoder providing $N_{\text{enc}} = 135$ ticks per revolution. Encoder measurements are published at approximately 30 Hz.

Inertial Measurement Unit The robot is equipped with an MPU-9250 IMU. Only the angular velocity around the vertical axis (ω_z) is used for state estimation. The IMU operates at approximately 30 Hz. No additional onboard filtering was applied to the gyroscope measurements.

Camera and AprilTags The monocular camera operates at approximately 30 Hz. AprilTag detections use the `tag36h11` family with a tag side length of 0.12 m.

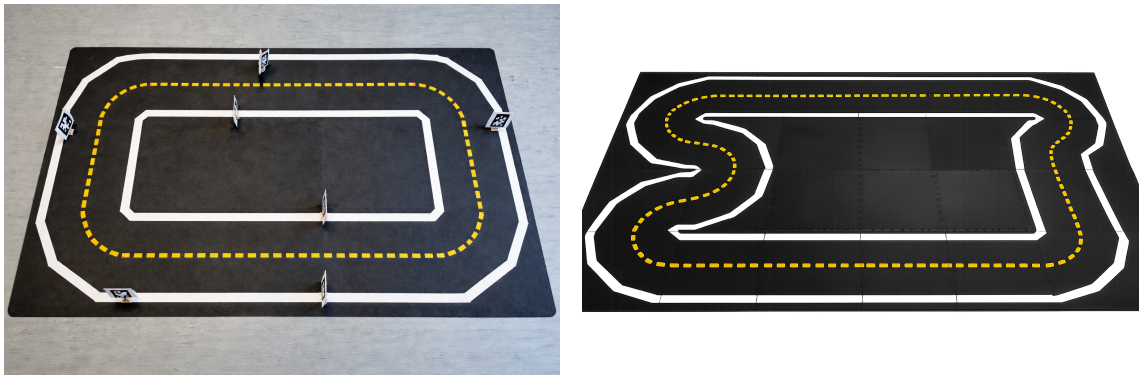
Detected tag poses are expressed relative to the robot base frame using a fixed extrinsic transformation. Tag world coordinates were obtained using the Vicon motion capture system.

5.2.2 Recorded Experiments

Two track configurations were designed to evaluate estimator performance under different levels of geometric complexity. Photographs of the experimental setups are shown in Figure 5.2a and Figure 5.2b.

- **Easy track:** primarily composed of straight segments connected by smooth 90° turns.
- **Hard track:** includes tighter curvature sections, multiple direction changes, and longer straight segments.

The easy track contained 10 AprilTags, while the hard track contained 11. All tags were rigidly fixed in the environment, and their world coordinates were measured using the Vicon motion capture system to ensure accurate landmark positioning.



(a) Easy track (b) Hard track

Figure 5.2: Tracks used for localization experiments

Geometric Characteristics The easy configuration consists of two long straight segments connected by two shorter straights and four smooth 90° turns with an approximate radius of 0.30 m.

The hard track introduces:

- Increased curvature variation
- Consecutive directional changes
- Longer straight-line segments

These characteristics amplify heading uncertainty growth and increase sensitivity to bias estimation errors, making the scenario more challenging for loosely coupled architectures.

Experimental Protocol For each track configuration, multiple runs were recorded. Each run consisted of three complete laps of the circuit.

The easy track was primarily employed during the development phase for filter tuning and parameter adjustment, owing to its simpler geometry and repeatable motion profile. After parameter tuning was finalized, both track configurations were used for performance evaluation.

From the recorded datasets, a representative run for each track was selected for detailed analysis. Selection was based on data completeness and stable sensor operation, ensuring absence of packet loss or sensor dropout.

Motion Profile The robot operated at a maximum linear speed of approximately 0.3 m/s. Velocity was controlled by a lane-following controller, leading to moderate speed variations along the trajectory.

Straight segments primarily test encoder integration performance, while curved sections stress heading estimation and bias correction. The hard track, in particular, generates sustained angular motion, which highlights differences in cross-covariance handling between filter architectures.

Ground Truth System Ground-truth pose measurements were obtained using a Vicon motion capture system operating at 100 Hz.

Vicon data were used exclusively for:

- Initialization of the robot's initial pose
- Quantitative evaluation of estimation accuracy

Ground truth was not used as a measurement input to any of the Kalman filter architectures.

Sensor Synchronization and Latency Considerations All sensor data were timestamped within ROS. Initial experiments revealed noticeable latency when transmitting camera data over WiFi. To mitigate this effect:

- AprilTag detection resolution was reduced
- A region of interest was applied for lane detection
- All critical nodes were executed onboard the Duckiebot

After these adjustments, latency effects were considered negligible for the purposes of state estimation.

All datasets were recorded in an indoor laboratory environment under artificial lighting conditions.

5.3 Discretization and Numerical Considerations

5.3.1 Sampling Time Handling

All sensor measurements are timestamped within ROS and logged into a unified dataset. During offline evaluation, timestamps are sorted in ascending order and processed sequentially.

The sampling interval T_s is computed dynamically from consecutive timestamps:

$$T_s = t_k - t_{k-1}. \tag{5.1}$$

Therefore, a variable time-step formulation is adopted. This avoids imposing an artificial fixed-rate discretization and ensures consistency with the actual sensor timing.

Encoder and IMU measurements are typically available at approximately 30 Hz. The camera provides images at 30 Hz; however, AprilTag detections are event-driven and occur at a lower and irregular rate, depending on visibility and processing time.

Vicon ground-truth data are recorded at 100 Hz and are used exclusively for evaluation and initialization.

State propagation is triggered by encoder increments, while measurement updates (IMU or AprilTag) are applied asynchronously whenever new sensor data become available.

5.3.2 State Propagation

State propagation is performed using the discrete-time motion model defined in Chapter 3. The same formulation is used across all architectures to ensure a fair comparison.

Encoder increments provide linear and angular motion estimates, while bias correction is applied to the heading update. Midpoint integration is employed to reduce discretization error during curved motion.

5.3.3 Angle Normalization

All angular states and residuals are normalized to the interval $(-\pi, \pi]$ after each update.

Angle wrapping is applied to:

- The heading state θ_k
- Innovation terms involving angular measurements
- Vision-based bearing residuals

This prevents artificial discontinuities at $\pm\pi$ and avoids spurious large innovations.

5.3.4 Covariance Update Form

For improved numerical stability, the covariance matrix is updated using the Joseph stabilized form [10]:

$$P_k \leftarrow (I - K_k H_k) P_k (I - K_k H_k)^T + K_k R_k K_k^T. \tag{5.2}$$

This formulation guarantees symmetry and positive semi-definiteness of the covariance matrix under finite numerical precision.

The Joseph form is used consistently across all filter architectures to ensure a fair comparison.

5.4 Process Noise Modeling

Process noise modeling plays a central role in balancing estimator responsiveness and consistency. In this work, process noise is used to capture unmodeled effects such as wheel slip, encoder quantization, uneven ground contact, and residual modeling errors that cannot be explicitly represented in the deterministic motion equations.

Rather than assuming constant noise levels, adaptive noise models are adopted to reflect the fact that uncertainty grows with motion magnitude. This choice is particularly important for differential-drive robots, where odometry error accumulation is strongly correlated with traveled distance and angular motion [2, 5].

5.4.1 Translational and Rotational Noise

The translational and rotational process noise standard deviations are defined as

$$\sigma_s = \alpha_s |\Delta s_k|, \tag{5.3}$$

$$\sigma_\theta = \alpha_\theta + \beta_\theta |\Delta \theta_{\text{enc},k}|. \tag{5.4}$$

Rationale for Adaptive Scaling The translational uncertainty σ_s is modeled as proportional to the traveled distance Δs_k. This reflects the empirical observation that odometry errors grow with motion, due to effects such as wheel slip, surface irregularities, and finite encoder resolution. When the robot is stationary or moves very slowly, the translational process noise naturally approaches zero.

Similarly, the rotational uncertainty σ_θ increases with the magnitude of the encoder-derived heading increment. Curved motion and sustained rotation amplify sensitivity to wheel asymmetries and encoder mismatch, motivating a motion-dependent rotational noise model.

The constant term α_θ accounts for residual angular uncertainty even during very small or zero encoder rotations, capturing unmodeled disturbances and discretization effects.

Units and Consistency All angular quantities are expressed in radians. Consequently, σ_θ, α_θ, and $\Delta\theta_{\text{enc},k}$ are expressed in radians, while σ_s and α_s are expressed in meters. This ensures dimensional consistency within the covariance matrix and avoids implicit unit conversions during filter execution.

Interpretation of Coefficients The coefficient α_s determines the rate at which translational uncertainty grows per unit distance traveled. Larger values increase robustness to unmodeled wheel slip at the cost of reduced short-term precision.

The coefficient α_θ represents a baseline angular uncertainty, while β_θ controls the sensitivity of rotational uncertainty to encoder-derived heading changes. Together, these parameters regulate how aggressively heading uncertainty is injected during

turning maneuvers, which is critical for maintaining consistency in landmark-based updates.

The selected values were tuned empirically to achieve statistically consistent behavior across all architectures, as verified through NIS analysis [3, 14].

5.4.2 Bias Noise

The gyroscope bias is modeled as a random walk:

$$b_{k+1} = b_k + w_{b,k}, \qquad w_{b,k} \sim \mathcal{N}(0, \sigma_b^2 T_s). \tag{5.5}$$

This model captures slow temporal variations in the IMU bias caused by temperature changes, sensor aging, and electronic drift. The bias noise standard deviation σ_b is assumed constant and expressed in radians per second.

A small value of σ_b enforces temporal smoothness of the bias estimate, preventing rapid fluctuations that would otherwise be absorbed from measurement noise. At the same time, allowing nonzero bias process noise ensures that long-term drift can be tracked and compensated.

The chosen value of σ_b represents a compromise between bias adaptability and estimator stability, and was selected to yield consistent IMU innovation statistics across both centralized and partitioned architectures.

5.5 Measurement Noise Modeling

Measurement noise covariance matrices determine the relative weight assigned to sensor observations during the correction step. Accurate noise modeling is essential to ensure statistical consistency, as verified through innovation analysis.

5.5.1 IMU Noise

The gyroscope measurement noise covariance is modeled as

$$R_{\text{imu}} = \sigma_\omega^2, \tag{5.6}$$

where σ_ω represents the standard deviation of angular velocity noise around the vertical axis.

The value of σ_ω was selected based on a combination of:

- Manufacturer specifications of the MPU-9250 gyroscope,
- Empirical observation of stationary sensor data,
- Innovation consistency analysis.

Since the IMU measurement model compares the gyroscope reading to the encoder-derived angular velocity estimate, R_{imu} implicitly accounts for both gyroscope noise and residual encoder discretization effects.

The final value of σ_ω was tuned to ensure that the NIS statistic for IMU updates remained consistent with a chi-square distribution with one degree of freedom. Experimental results show that the empirical mean of the IMU NIS is close to unity, indicating well-balanced measurement weighting.

5.5.2 AprilTag Measurement Noise

The AprilTag measurement covariance matrix is defined as

$$R_{\text{tag}} = \begin{bmatrix} \sigma_\phi^2 & 0 \\ 0 & \sigma_\rho^2 \end{bmatrix}, \tag{5.7}$$

where σ_ϕ is the bearing standard deviation (in radians) and σ_ρ is the range standard deviation (in meters).

Bearing Variance The bearing uncertainty σ_ϕ captures errors arising from:

- Pixel quantization,
- Camera intrinsic calibration errors,
- Extrinsic calibration uncertainty,
- Small pose estimation inaccuracies in the AprilTag detector.

Bearing measurements are particularly sensitive to small image distortions and tag detection jitter. A moderate angular noise level was therefore adopted to avoid overconfidence in heading corrections.

Range Variance The range uncertainty σ_ρ reflects errors in estimating the 3D position of the tag relative to the camera. Range accuracy depends on:

- Tag size,
- Distance to the landmark,
- Camera resolution,
- Perspective effects.

Since range estimation degrades with distance, a constant variance approximation was adopted, representing an average operating distance within the track.

Independence Assumptions The covariance matrix is chosen diagonal, assuming independence between bearing and range noise. While small correlations may exist in practice due to projective geometry coupling, empirical innovation analysis indicated that a diagonal model provides adequate statistical consistency.

The selected values of σ_ϕ and σ_ρ were validated through chi-square consistency testing based on the Normalized Innovation Squared (NIS). A detailed description of the validation and gating procedure is provided in Section 5.7.

The empirical AprilTag NIS statistics fall within the expected confidence bounds, confirming appropriate tuning.

5.6 Initialization Strategy

Proper initialization is essential for reliable filter convergence, particularly in nonlinear state estimation problems.

Initial Pose Estimate

The initial robot pose (x_0, y_0, θ_0) was set using Vicon ground-truth measurements at the beginning of each experimental run. This ensures consistent alignment between estimated and reference trajectories for evaluation purposes.

Ground truth was used exclusively for initialization and performance evaluation, and was never incorporated as a measurement input to the Kalman filter architectures.

Initial Bias Estimate

The gyroscope bias was initialized to zero:

$$b_0 = 0. \tag{5.8}$$

This choice reflects the absence of prior knowledge regarding the bias magnitude. The bias state is allowed to converge naturally through IMU updates during the initial motion phase.

Initial Covariance Matrices

Initial covariance matrices were selected to represent moderate uncertainty:

- Position variance: $\sigma_x^2 = \sigma_y^2 = (0.15\,\mathrm{m})^2$
- Heading variance: $\sigma_\theta^2 = (20^\circ)^2$
- Bias variance: $\sigma_b^2 = (5^\circ)^2$

These values ensure that early measurements are appropriately weighted without causing numerical instability.

Sensitivity Considerations

Additional experiments with perturbed initial conditions confirmed that all architectures converge reliably under reasonable initialization errors. The selected initial covariances therefore provide a robust compromise between rapid convergence and conservative uncertainty representation.

5.7 Measurement Validation and Gating

Reliable state estimation requires rejecting inconsistent or physically implausible measurements. In this work, innovation-based statistical gating is employed to ensure consistency across all filter architectures.

5.7.1 Normalized Innovation Squared

For each measurement update, the innovation vector is defined as

$$y_k = z_k - \hat{z}_k, \tag{5.9}$$

with innovation covariance

$$S_k = H_k P_{k|k-1} H_k^T + R_k. \tag{5.10}$$

The NIS statistic is computed as

$$\text{NIS}_k = y_k^T S_k^{-1} y_k. \tag{5.11}$$

Under correct modeling assumptions and properly tuned noise covariances, the NIS follows a chi-square distribution with degrees of freedom equal to the measurement dimension [2, 3].

In this work, the NIS is primarily used to tune the process and measurement noise covariance matrices. Specifically, the empirical distribution of the NIS is compared against the theoretical chi-square distribution to assess statistical consistency. The noise parameters in Q_k and R_k are adjusted such that the NIS values fall within the expected confidence bounds.

IMU Updates The IMU measurement is one-dimensional. Therefore, the NIS is compared against a chi-square distribution with one degree of freedom:

- 95% confidence bound: $\chi^2_{1,0.95} = 3.84$
- 99% confidence bound: $\chi^2_{1,0.99} = 6.63$

These thresholds are used primarily to assess the statistical consistency of the filter, i.e., to verify whether the magnitude of the innovation is compatible with the predicted covariance. This information is used to tune the process and measurement noise parameters rather than to strictly reject measurements.

Systematically large NIS values indicate underestimated noise, while consistently small values suggest overestimated uncertainty.

AprilTag Updates AprilTag measurements are two-dimensional (bearing and range). The NIS therefore follows a chi-square distribution with two degrees of freedom:

- 95% confidence bound: $\chi^2_{2,0.95} = 5.99$
- 99% confidence bound: $\chi^2_{2,0.99} = 9.21$

The 99% confidence threshold is used as the gating criterion for landmark updates. Measurements exceeding $\chi^2_{2,0.99}$ are rejected as statistical outliers based on innovation gating principles [3].

5.7.2 Outlier Rejection

Outlier rejection is implemented through a combination of statistical and geometric validation.

NIS-Based Rejection The NIS is used both for offline consistency analysis, guiding the tuning of Q_k and R_k, and for online measurement validation through statistical gating. These two uses operate at different levels: the former evaluates the overall filter behavior, while the latter enables rejection of individual inconsistent observations.

For each AprilTag detection, the NIS value is computed. If

$$\text{NIS}_k > 9.21, \tag{5.12}$$

the measurement is discarded and no update is applied. This prevents large residuals from corrupting the state estimate, particularly in cases of partial occlusion or incorrect pose estimation.

Geometric Validity Checks Prior to performing the update, additional geometric checks are applied:

- Minimum range threshold to avoid singular configurations
- Positive depth constraint in the camera frame
- Verification that the innovation covariance S_k is positive definite and sufficiently well-conditioned to ensure stable matrix inversion

These safeguards prevent ill-conditioned updates in degenerate geometric situations, such as extremely close landmarks or near-collinearity.

Handling Missing Detections AprilTag measurements are event-driven and may be absent during extended portions of the trajectory. When no landmark detection is available, the filter relies solely on propagation and IMU updates.

Since the process noise model accounts for motion-dependent uncertainty, state covariance naturally increases during periods without external corrections. Once new landmark observations become available, the covariance contracts through measurement updates.

This behavior ensures graceful degradation in the absence of vision, while preserving statistical consistency.

5.8 Architecture-Specific Implementation Details

Although the architectures share the same underlying motion and measurement models, their implementations differ in how state variables are partitioned, how covariance is handled, and how information is exchanged between subsystems.

The theoretical formulations were presented in the previous chapter; here, the focus is on the key implementation differences relevant to practical deployment.

5.8.1 Centralized EKF

The centralized architecture maintains a single joint state vector

$$q_k = [x_k,\, y_k,\, \theta_k,\, b_k]^T,$$

with a full 4×4 covariance matrix.

State propagation is performed using encoder-derived motion increments, while IMU and AprilTag measurements directly update the joint state. All cross-correlations between position, heading, and bias states are explicitly represented in the covariance matrix.

Since the full covariance is maintained, no additional variance injection or approximation mechanisms are required. All measurement updates are applied directly to the same state vector.

5.8.2 Cascaded EKF Architecture

In the cascaded implementation, the bias and pose states are estimated by two independent filters:

- KF-1: bias state b_k
- EKF-2: pose state $[x_k, y_k, \theta_k]^T$

The bias filter uses IMU measurements to estimate the gyroscope offset independently from the pose estimator.

During pose propagation, the bias estimate $\hat{b}_k$ is used to correct the heading increment. To account for bias uncertainty, the heading variance in EKF-2 is inflated using

$$P_\theta \leftarrow P_\theta + T_s^2 P_k^{(1)},$$

where $P_k^{(1)}$ is the bias variance.

Landmark updates modify only the pose filter. No explicit cross-covariance terms are maintained between bias and pose states, resulting in a one-way information flow from bias to pose.

5.8.3 Error-State Kalman Filter

The Error-State Kalman Filter separates the state into:

- A nominal state propagated using the full nonlinear model,
- An error state estimated through linearized dynamics.

Measurement updates are performed in the error-state space. After each correction, the estimated error is injected into the nominal state, and the error state is reset to zero.

The covariance matrix is maintained in error-state form and updated using the Joseph stabilized formulation. Since the state representation is additive, no additional covariance transformation is required after injection.

Compared to the centralized EKF, the ESKF improves numerical stability by keeping the linearized error state small, while preserving the full joint covariance structure.

5.8.4 Hybrid-Cascaded EKF Architecture

The hybrid implementation partitions the state into two interacting filters:

- EKF-1: heading and bias $[\theta_k, b_k]^T$
- EKF-2: planar position $[x_k, y_k]^T$

Encoder increments are split between the filters: heading propagation is handled by EKF -1, while translational propagation uses the heading estimate to update position in EKF-2.

Since full joint covariance is not maintained, heading uncertainty is injected into the position filter through an additive covariance term proportional to $\Delta s_k^2 \, P_{\theta\theta}$.

AprilTag measurements update the position filter directly. Additionally, a vision-based heading estimate derived from range–bearing observations provides a feedback correction to EKF-1, introducing bidirectional coupling between the filters.

This partial feedback mechanism differentiates the hybrid approach from the purely cascaded architecture and improves consistency without maintaining a full joint covariance.

5.9 Computational Considerations

Although the primary focus of this thesis is on estimation consistency and architectural design, computational aspects are briefly discussed for completeness.

State Dimension per Architecture

The state dimension differs across architectures:

- Centralized EKF: 4 states $[x, y, \theta, b]$
- Cascaded EKF: $1 + 3$ states (bias and pose in separate filters)
- Error-State KF: 4 states (error representation)
- Hybrid-Cascaded EKF: $2 + 2$ states (heading–bias and position)

The largest covariance matrix propagated in any architecture is therefore 4×4. Partitioned architectures operate on smaller 2×2 or 3×3 matrices.

Matrix Inversion Size

The computationally dominant operations in Kalman filtering are matrix multiplications and inversion of the innovation covariance matrix.

In this work:

- IMU updates require inversion of a 1×1 matrix.
- AprilTag updates require inversion of a 2×2 matrix.

Even in the centralized formulation, no large matrix inversions are required. Therefore, computational complexity differences between architectures remain negligible.

Runtime Performance

All filters were implemented in MATLAB and executed offline. For the recorded datasets (approximately three laps per run), execution time per architecture was well below real-time requirements.

Given the small state dimension and low measurement dimensionality, the computational load is minimal. The dominant cost arises from matrix multiplications of size at most 4×4, which are trivial for modern processors.

Real-Time Feasibility

From a theoretical standpoint, all considered architectures are suitable for real-time deployment on embedded platforms such as the Jetson NANO used in the Duckiebot.

Since encoder and IMU updates occur at approximately 30 Hz, and AprilTag detections are event-driven at comparable or lower rates, the computational burden of state estimation remains significantly below available processing capacity.

Therefore, architectural differences in this study are motivated by statistical and structural considerations rather than computational constraints.

5.10 Summary

This chapter presented the practical implementation details of the probabilistic motion and measurement models, as well as the different Kalman filter architectures.

All architectures share:

- Identical motion and measurement models
- The same process and measurement noise parameters
- The same initialization strategy
- The same dataset and timestamp sequence

Consequently, performance differences observed in the following chapters can be attributed solely to architectural design choices, specifically to how state partitioning and covariance structure affect information flow and uncertainty representation.

Careful attention was given to numerical stability, including variable sampling time handling, angle normalization, and Joseph-form covariance updates.

Measurement validation through NIS-based gating ensures statistical consistency and robustness against outliers.

With these implementation details established, the next chapters evaluate and compare the estimation performance of the considered architectures through simulation and experimental validation.

6

Simulation Results

This chapter presents a systematic comparative evaluation of the Kalman filter architectures introduced in Chapter 4. All filters are assessed using identical motion and measurement models, shared noise parameters, and the same recorded datasets, ensuring that observed differences arise exclusively from architectural design choices.

Two track configurations are considered. The *easy track* is primarily used for parameter tuning and verification of statistical consistency. Its moderate curvature and frequent landmark visibility provide a controlled environment for adjusting process and measurement noise parameters, validating innovation behavior, and ensuring stable filter operation.

The *hard track*, characterized by frequent turns, longer straight segments, and sustained angular motion, is used for unbiased performance evaluation. All noise parameters determined during tuning on the easy track are kept unchanged for the hard-track analysis. This separation prevents overfitting and guarantees a fair architectural comparison.

Performance is evaluated using both accuracy and consistency metrics. Accuracy is quantified through position and heading Root-Mean-Square Error (RMSE) with respect to Vicon ground truth.

Consistency refers to the statistical agreement between the predicted estimation uncertainty and the observed estimation errors. It is assessed using the Normalized Innovation Squared (NIS), whose empirical distribution is compared against theoretical chi-square bounds, together with outlier statistics at 95% and 99% confidence levels [3, 14].

The objective of this chapter is therefore twofold:

- To verify statistical consistency of each architecture under controlled tuning

conditions.

- To compare localization accuracy and robustness under more demanding geometric excitation.

The results presented in the following sections provide insight into how covariance structure, state partitioning, and cross-correlation handling affect estimation performance in practice.

6.1 Evaluation Metrics

To enable a rigorous and objective comparison of the considered filter architectures, performance is evaluated using both *accuracy* and *statistical consistency* metrics.

Accuracy metrics quantify the deviation from ground truth, while consistency metrics assess whether the estimated covariance correctly represents the actual estimation uncertainty.

6.1.1 Position Root-Mean-Square Error

Position accuracy is evaluated using the planar RMSE with respect to Vicon ground truth:

$$\mathrm{RMSE}_{pos} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[(x_k - x_k^{gt})^2 + (y_k - y_k^{gt})^2\right]}. \tag{6.1}$$

Here, (x_k, y_k) denotes the estimated position at time step k, (x_k^{gt}, y_k^{gt}) the corresponding ground-truth position, and N the total number of samples.

This metric provides a global measure of translational accuracy over the entire trajectory.

6.1.2 Heading Root-Mean-Square Error

Heading accuracy is evaluated using the angular root-mean-square error:

$$\mathrm{RMSE}_{\theta} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\mathrm{wrap}\left(\theta_k - \theta_k^{gt}\right)^2}. \tag{6.2}$$

The operator $\mathrm{wrap}(\cdot)$ normalizes angular differences to the interval $(-\pi, \pi]$, preventing artificial discontinuities at $\pm\pi$.

Heading RMSE is reported in degrees for improved interpretability.

6.1.3 Normalized Innovation Squared (NIS)

Statistical consistency is evaluated using the Normalized Innovation Squared (NIS):

$$\text{NIS}_k = y_k^T S_k^{-1} y_k, \tag{6.3}$$

where y_k is the innovation and S_k its covariance.

Under correct modeling assumptions, NIS_k follows a chi-square distribution with degrees of freedom equal to the measurement dimension ν [2, 3].

In this work:

- $\nu = 1$ for IMU updates
- $\nu = 2$ for AprilTag updates

For a statistically consistent filter, the expected mean value satisfies [14]

$$\mathbb{E}[\text{NIS}] \approx \nu. \tag{6.4}$$

Systematically larger values indicate overconfidence, while consistently smaller values suggest conservative uncertainty estimates.

6.1.4 Confidence Bounds and Outlier Rate

To evaluate innovation consistency over time, NIS values are compared against chi-square confidence thresholds.

For a chosen confidence level α, the acceptance region is defined as

$$\text{NIS}_k \leq \chi^2_{\nu,\alpha}, \tag{6.5}$$

where $\chi^2_{\nu,\alpha}$ denotes the upper α-quantile of the chi-square distribution [3] with ν degrees of freedom.

Outlier percentage is defined as

$$\text{Outlier Rate} = \frac{\#\{\text{NIS}_k > \chi^2_{\nu,\alpha}\}}{\text{Total Measurements}}. \tag{6.6}$$

For a consistent filter, approximately a fraction $(1 - \alpha)$ of the total number of innovations is expected to lie outside the confidence region.

6.1.5 Consistency Envelope Verification

In addition to innovation-based metrics, state consistency is evaluated by comparing estimation errors with predicted covariance bounds.

For each state component, the estimation error is plotted together with $\pm 2\sigma$ bounds derived from the covariance matrix [18]:

$$\pm 2\sqrt{P_{ii,k}}. \tag{6.7}$$

A consistent estimator should exhibit error trajectories that remain predominantly within these bounds without systematic divergence or excessive conservativeness.

6.1.6 Interpretation Framework

It is important to emphasize that low RMSE alone does not guarantee statistical consistency. An estimator may achieve small average error while underestimating uncertainty, leading to overconfident covariance predictions.

Therefore, both accuracy and consistency metrics are jointly analyzed in the following sections to provide a comprehensive evaluation of the considered filter architectures.

6.2 Sensor Models

The simulations rely on three sensing modalities: wheel encoders, gyroscope measurements, and vision-based AprilTag observations. All sensor models are assumed to be affected by zero-mean Gaussian noise.

Wheel Encoder Model Wheel encoder measurements provide incremental wheel displacements, which are converted into linear and angular motion increments:

$$\Delta s_k = \frac{\Delta s_L + \Delta s_R}{2}, \tag{6.8}$$

$$\Delta\theta_{\text{enc},k} = \frac{\Delta s_R - \Delta s_L}{l}. \tag{6.9}$$

Encoder noise is implicitly modeled within the process noise covariance, which scales with the traveled distance and rotation magnitude.

Gyroscope Model The gyroscope provides angular velocity measurements modeled as:

$$\omega_k = \dot{\theta}_k + b_k + n_{\omega,k}, \tag{6.10}$$

where b_k is a slowly varying bias and $n_{\omega,k}$ is zero-mean Gaussian noise with variance:

$$n_{\omega,k} \sim \mathcal{N}(0, \sigma_\omega^2), \quad \sigma_\omega = 17^\circ/\text{s}. \tag{6.11}$$

The bias is modeled as a random walk process.

AprilTag Measurement Model Each AprilTag detection provides relative bearing and range measurements:

$$z_k = \begin{bmatrix} \phi_k \\ \rho_k \end{bmatrix} = \begin{bmatrix} \arctan2(y_t - y_k, x_t - x_k) - \theta_k \\ \sqrt{(x_t - x_k)^2 + (y_t - y_k)^2} \end{bmatrix} + n_k, \tag{6.12}$$

where (x_t, y_t) denotes the landmark position.

Measurement noise is modeled as Gaussian:

$$n_k \sim \mathcal{N}(0, R_k), \quad R_k = \begin{bmatrix} \sigma_\phi^2 & 0 \\ 0 & \sigma_\rho^2 \end{bmatrix}, \tag{6.13}$$

with

$$\sigma_\phi = 1^\circ \tag{6.14}$$

$$\sigma_\rho = 0.015\,\text{m}. \tag{6.15}$$

6.3 Dead-Reckoning Baseline

Before comparing the different Kalman filter architectures, it is instructive to evaluate the performance of pure dead-reckoning. This baseline corresponds to state propagation using only encoder-derived motion increments and gyroscope measurements, without any landmark-based correction.

In this configuration, the state is propagated according to the discrete-time unicycle model described in Chapter 3, using encoder-derived increments for translational motion and gyroscope measurements for rotational motion.

The gyroscope bias is estimated using the encoder–IMU discrepancy, as described in the bias estimation model. However, no landmark-based (AprilTag) measurement updates are incorporated.

Consequently, although short-term rotational consistency is improved through bias estimation, the system remains purely odometry-based and lacks any external absolute correction mechanism.

6.3.1 Trajectory Drift

Figure 6.1 shows the dead-reckoning trajectory on the hard track in simulation, overlaid with the reference trajectory.

While the initial segment of the trajectory approximately follows the ground truth, drift accumulates progressively. The most significant deviations occur during curved segments, where heading errors induced by gyroscope bias propagate into large position errors.

After three complete laps of the 12 m track, the dead-reckoning estimate exhibits a substantial global displacement error, demonstrating the unbounded nature of purely integrated motion estimates [2, 5]. Moreover, the heading evolution comparison in Figure 6.2 highlights the substantial drift.

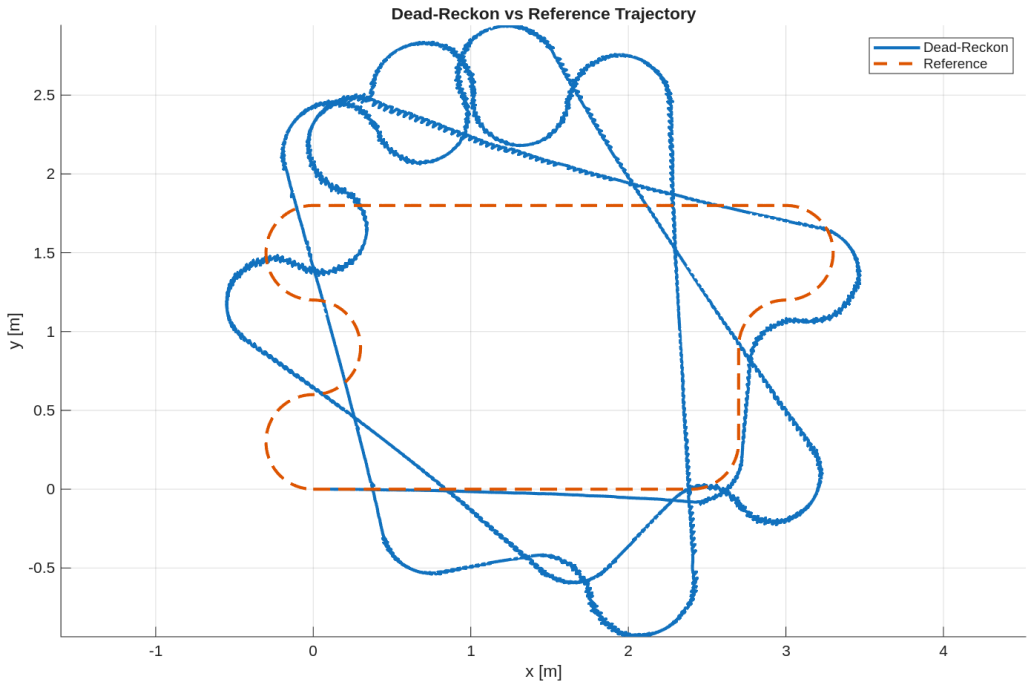


Figure 6.1: Dead-reckoning trajectory in simulation

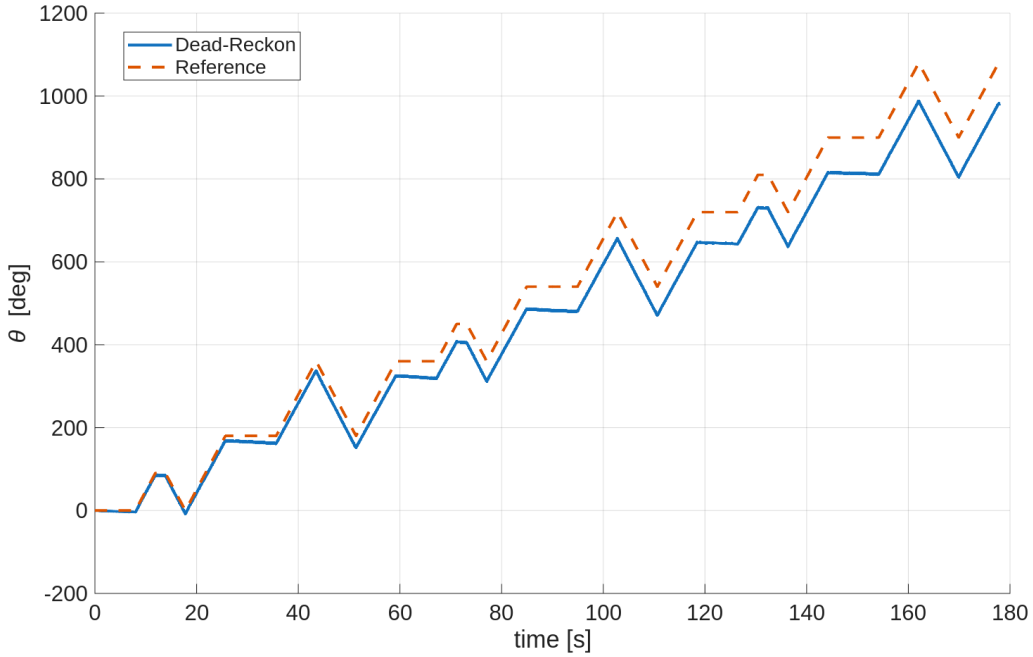
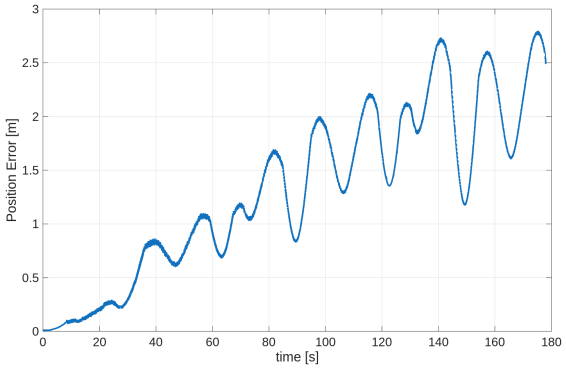
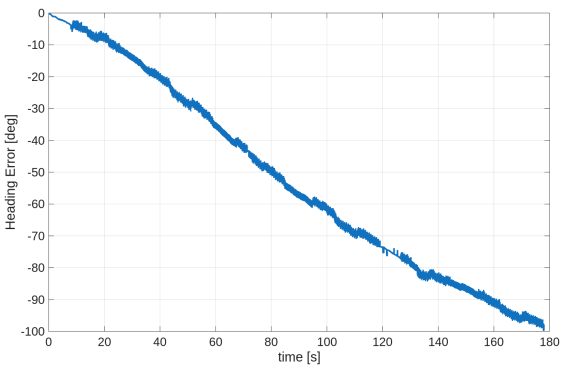


Figure 6.2: Dead-reckoning heading in simulation



(a) Dead-reckoning position error in simulation



(b) Dead-reckoning heading error in simulation

Figure 6.3: Dead-reckoning errors in the hard simulation scenario.

6.3.2 Error Growth

Figure 6.3 reports the position and heading errors over time.

Position error increases monotonically, particularly during sustained turning motion. Heading error accumulates due to residual bias estimation error, encoder noise, and integration effects, resulting in systematic orientation drift.

Unlike the Kalman-based architectures, dead-reckoning lacks a correction mechanism capable of reducing accumulated error. Consequently, uncertainty grows without bound.

6.3.3 Quantitative Comparison

Table 6.1 summarizes the RMSE obtained for dead-reckoning on the hard track.

Table 6.1: Dead-reckoning RMSE on hard track.

Method	Position RMSE [m]	Heading RMSE [deg]
Dead-Reckoning	1.532 m	60.55 deg

The magnitude of these errors is significantly larger than those obtained with landmark-assisted Kalman filtering, as shown in the following sections.

6.3.4 Discussion

The results confirm that encoder–IMU integration alone is insufficient for reliable long-term localization. Although short-term heading consistency is improved through bias estimation, residual errors in bias and encoder measurements accumulate over time, producing substantial global drift.

In particular, heading errors generated during curved motion translate directly into large position deviations. Since no absolute reference is available, these errors cannot be corrected once accumulated.

This baseline highlights the fundamental limitation of purely integrated motion estimation and motivates the introduction of landmark-based updates. By incorporating AprilTag observations, the Kalman filter architectures are able to periodically correct both heading and position, transforming an open-loop integration process into a bounded-error estimation problem.

6.4 Parameter Tuning and Consistency Verification on Easy Track

The easy track configuration was used to tune process and measurement noise parameters and to verify statistical consistency of the implemented filter architectures under moderate geometric excitation.

This track contains smooth 90° turns, frequent landmark visibility, and limited sustained angular motion. Such conditions provide a controlled environment for adjusting noise parameters without excessive nonlinear effects.

The tuning of process and measurement noise parameters was performed using an iterative procedure based on innovation consistency.

Initial values for the covariance matrices Q_k and R_k were selected based on prior empirical knowledge. The filters were then executed on the easy-track dataset, and the resulting Normalized Innovation Squared (NIS) statistics were analyzed.

The tuning objective was to achieve statistical consistency, i.e., to ensure that the empirical distribution of the NIS matched the theoretical chi-square distribution with appropriate degrees of freedom. In particular, the mean NIS was adjusted to be close to the expected value ν, and the percentage of samples exceeding the 95% and 99% confidence bounds was monitored.

Process noise parameters were increased when the filter exhibited overconfidence (NIS too large), and decreased when the filter was overly conservative (NIS too small). Similarly, measurement noise parameters were tuned to balance responsiveness and robustness of the updates.

This procedure was repeated until stable and statistically consistent behavior was obtained for all considered filter architectures. Once stable and statistically consistent behavior was achieved, these parameters were fixed and subsequently applied unchanged to the hard-track evaluation.

6.4.1 Trajectory Consistency

Figure 6.4 shows the estimated trajectories overlaid with the reference trajectory in simulation. All architectures closely follow the ground-truth trajectory, with no observable divergence or instability. No systematic drift is observed, confirming that bias estimation and encoder-based propagation operate correctly under nominal conditions.

The landmark positions are defined in a global reference frame and remain constant throughout the simulation. Their spatial distribution is chosen to ensure sufficient visibility along the track, with tags placed near curves and along straight segments to provide regular geometric constraints.

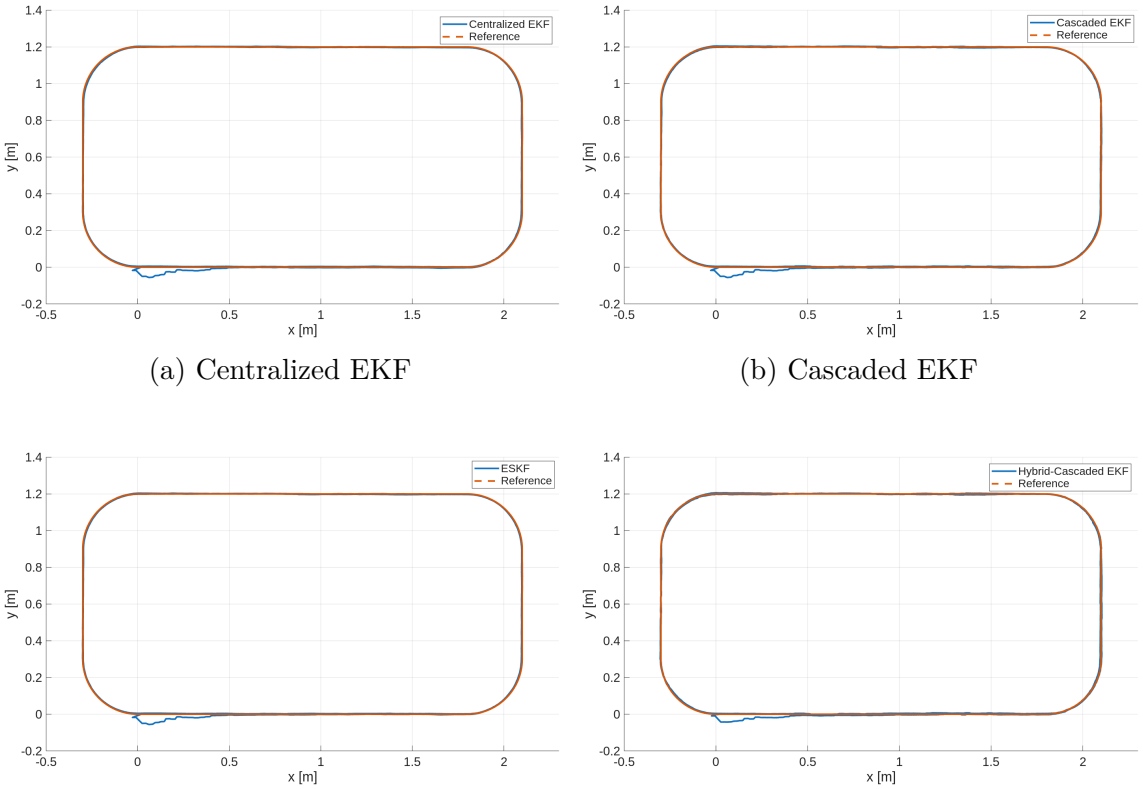
At each time step, landmark observations are generated based on the relative pose between the robot and each visible tag. A landmark is considered visible if it lies within the camera field of view and in front of the robot.

For each visible landmark, bearing and range measurements are computed as:

$$\phi_k = \arctan2(y_t - y_k, x_t - x_k) - \theta_k, \tag{6.16}$$

$$\rho_k = \sqrt{(x_t - x_k)^2 + (y_t - y_k)^2}. \tag{6.17}$$

This procedure emulates the output of a camera-based AprilTag detection system, where each detection provides relative bearing and distance information with limited field of view and measurement uncertainty.



(a) Centralized EKF

(b) Cascaded EKF

(c) ESKF

(d) Hybrid-Cascaded EKF

Figure 6.4: Estimated trajectories on the easy track

6.4.2 Position and Heading Errors

Figures 6.5 and 6.6 present the planar position and heading error over time with respect to the reference trajectory. More detailed plots are visible in Fig A.1 in Appendix A.1.

Position errors remain bounded within a few centimeters throughout the trajectory. Heading errors are small and show rapid correction following landmark updates.

No architecture exhibits sustained error growth, indicating stable interaction between propagation and measurement updates.

Root-mean-square errors for the easy track are summarized in Table 6.2.

Differences between architectures remain minor under these conditions, as expected in a scenario with frequent external corrections.

Table 6.2: RMSE on easy track

Architecture	Position RMSE [m]	Heading RMSE [deg]
Centralized EKF	0.006	0.96
Cascaded EKF	0.006	0.92
ESKF	0.006	0.89
Hybrid-Cascaded EKF	0.006	0.74

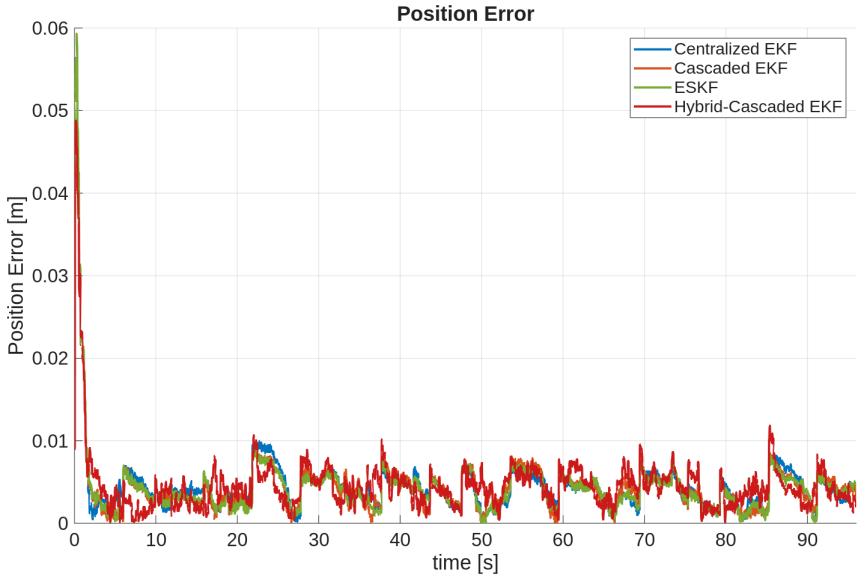


Figure 6.5: Planar position errors with respect to the reference trajectory for all filter architectures on the easy track.

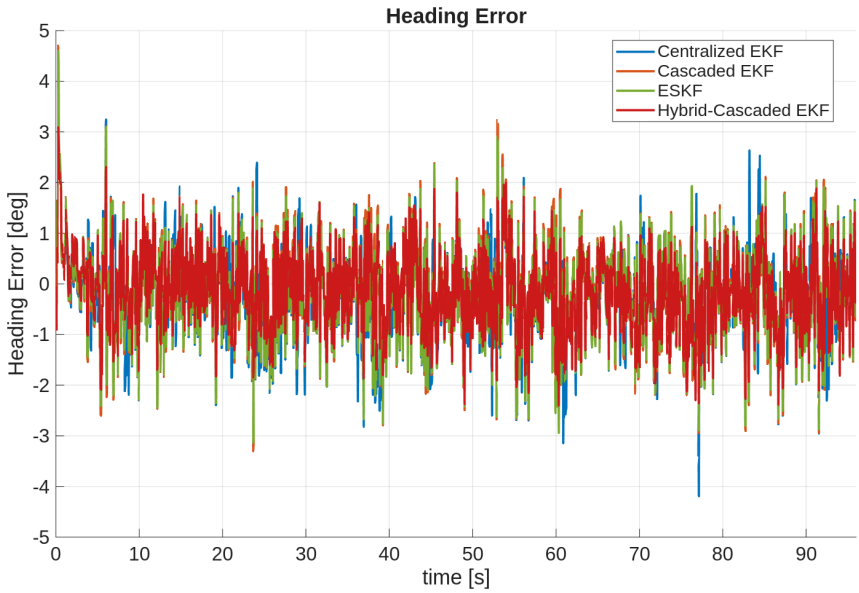


Figure 6.6: Heading errors with respect to the reference trajectory for all filter architectures on the easy track.

6.4.3 Innovation Consistency

Statistical consistency was evaluated using the Normalized Innovation Squared (NIS), whose mean values are summarized in Table 6.3.

Figure A.2 in Appendix A.1 shows the NIS of both AprilTag and IMU updates, together with 95% and 99% chi-square confidence bounds.

Table 6.3: Mean NIS on easy track.

Architecture	IMU Mean NIS ($\nu = 1$)	Tag Mean NIS ($\nu = 2$)
Centralized EKF	0.99	1.41
Cascaded EKF	0.99	1.44
ESKF	0.99	1.49
Hybrid-Cascaded EKF	0.99	1.06

For a consistent estimator, the expected mean NIS equals the measurement dimension ν. For the IMU ($\nu = 1$), observed mean values are very close to unity, indicating accurate calibration of the yaw-rate measurement noise.

For the AprilTag updates ($\nu = 2$), the centralized, cascaded, and ESKF architectures exhibit mean NIS values around 1.4–1.5, which remain below the theoretical expectation of 2. This indicates mild conservativeness in the measurement covariance modeling [14].

The hybrid architecture exhibits a substantially lower mean NIS (about 1.06), reflecting a more conservative innovation covariance. This behavior arises from the decoupled structure and the explicit incorporation of heading uncertainty into the position update, which inflates the effective measurement covariance.

Outlier percentages at 99% confidence level are reported in Table 6.4.

Table 6.4: Outlier percentage on easy track.

Architecture	IMU > 99%	Tag > 99%
Centralized EKF	1.18%	0.11%
Cascaded EKF	1.21%	0.10%
ESKF	1.18%	0.10%
Hybrid-Cascaded EKF	1.21%	0.10%

The IMU outlier rates closely match the theoretical expectation of 1%, confirming accurate modeling of yaw-rate measurement uncertainty.

For AprilTag updates, the observed outlier percentages remain significantly below the theoretical value. This behavior indicates conservative innovation covariance

estimates, leading to fewer rejected measurements than statistically expected. Such conservativeness improves robustness at the expense of slightly reduced correction aggressiveness.

6.4.4 Covariance Envelope Verification

To further assess estimator consistency, the estimation errors were compared with the corresponding $\pm 2\sigma$ covariance envelopes, as shown in Fig. A.3 in Appendix A.1.

For all architectures, both position and heading errors remain consistently bounded within the predicted uncertainty limits, confirming the absence of divergence or covariance underestimation.

The covariance envelopes generally exceed the observed estimation errors, indicating a moderately conservative uncertainty prediction. This behavior is particularly evident in the heading component, where the predicted variance remains larger than the empirical error.

Such conservativeness is desirable in practical robotic localization, as it improves robustness against modeling inaccuracies and prevents overconfident state estimation.

6.4.5 Discussion

Under the easy-track conditions, all considered architectures demonstrate stable operation, accurate bias estimation, and statistically consistent innovation behavior.

The centralized, cascaded, and ESKF formulations yield nearly identical position accuracy, confirming their theoretical equivalence under planar motion and frequent landmark observations. In this regime, external corrections dominate the estimation process, limiting the influence of architectural differences on translational performance, as commonly observed in landmark-based localization systems [2].

The hybrid-cascaded architecture achieves accuracy comparable to the fully coupled formulations and exhibits the lowest heading RMSE among the evaluated filters. The results demonstrate that architectural decoupling alone does not imply performance degradation. When uncertainty exchange between subsystems is properly accounted for, simplified formulations can recover the accuracy and consistency of fully coupled estimation.

In contrast, pure dead-reckoning exhibits substantial drift, highlighting the fundamental necessity of landmark-based updates for bounded-error localization.

Overall, the easy-track scenario confirms that all Kalman-based architectures provide reliable and statistically consistent localization when frequent observations

are available. The tuned noise parameters obtained from this dataset are therefore retained unchanged for the more demanding hard-track evaluation presented in the next section.

6.5 Performance Evaluation on Hard Track

The hard-track configuration is used to evaluate localization performance under more demanding motion and sensing conditions. In contrast to the easy track, this scenario includes prolonged turning maneuvers, longer prediction intervals between landmark observations, and reduced geometric excitation uniformity.

All process and measurement noise parameters correspond exactly to those tuned on the easy track and are applied here without modification. This ensures that the following results represent an unbiased comparison of the considered filter architectures rather than dataset-specific tuning.

6.5.1 Trajectory Comparison

Fig 6.7 presents the estimated trajectories together with the reference trajectory, while Fig 6.8 highlights the estimated heading evolutions of the filters, overlaid with the reference one.

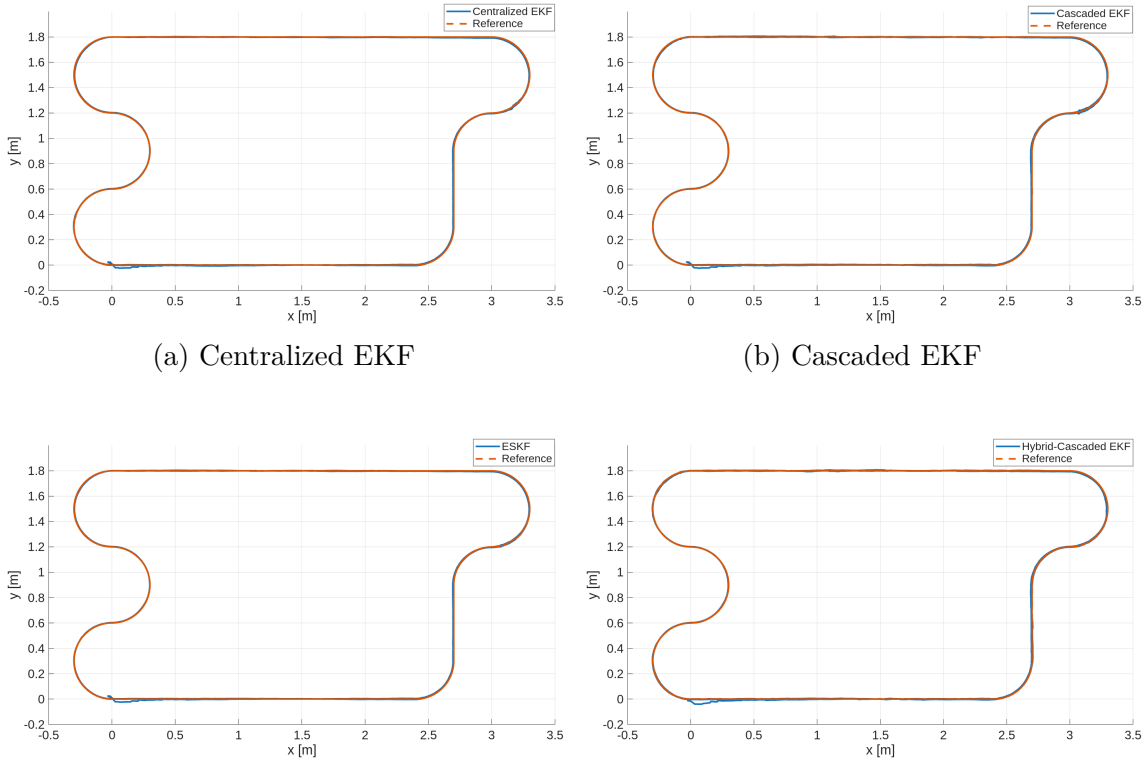
All Kalman-based architectures successfully maintain bounded localization error throughout the experiment despite the increased motion complexity. Small deviations become visible during extended curved segments, where heading uncertainty accumulates between successive landmark observations.

No estimator divergence is observed, confirming stable operation under sustained nonlinear motion.

6.5.2 Position and Heading Errors

Table 6.5 summarizes the obtained accuracy metrics. Detailed planar position and heading error plots with respect to ground truth are shown in Fig. A.4 in Appendix A.2. Compared to the easy-track scenario, larger transient errors appear due to longer propagation phases and reduced measurement availability.

Nevertheless, all Kalman-based architectures maintain bounded errors throughout the trajectory. Heading deviations occur primarily during aggressive turning maneuvers, where uncertainty in rotational motion directly affects position propagation.



(a) Centralized EKF

(b) Cascaded EKF

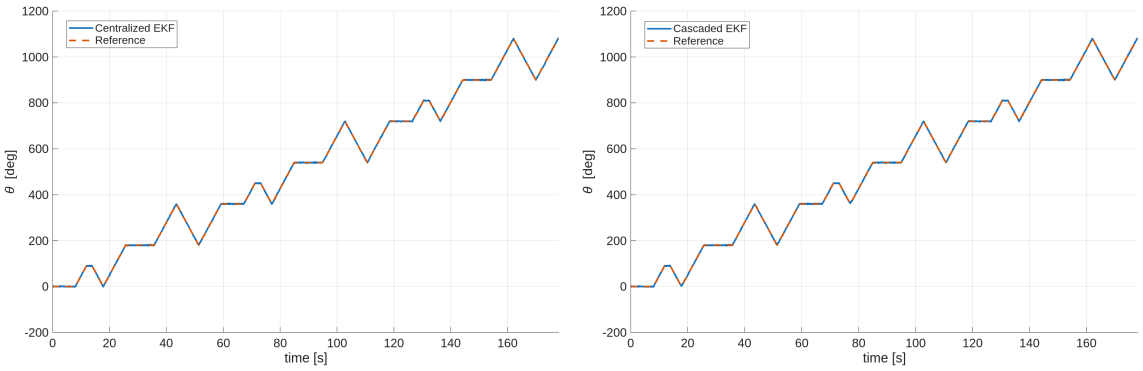
(c) ESKF

(d) Hybrid-Cascaded EKF

Figure 6.7: Estimated trajectories on the hard track

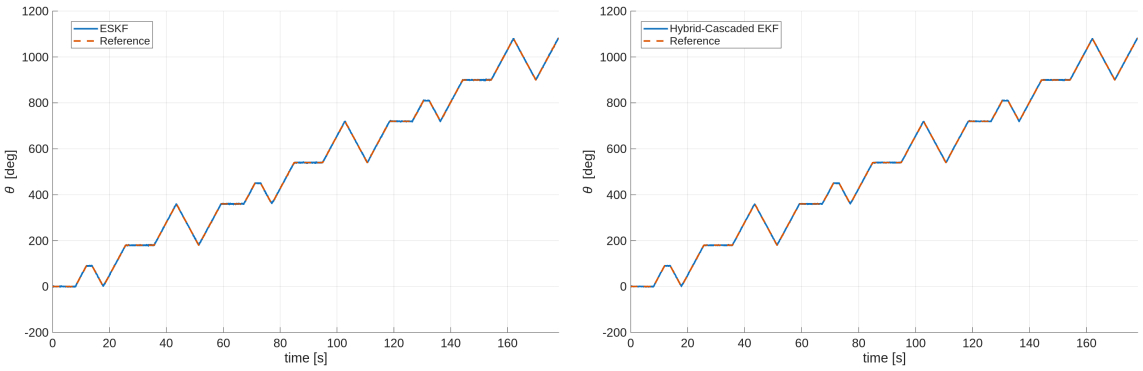
Table 6.5: RMSE on hard track

Architecture	Position RMSE [m]	Heading RMSE [deg]
Centralized EKF	0.005	0.99
Cascaded EKF	0.005	1.00
ESKF	0.005	0.96
Hybrid-Cascaded EKF	0.006	0.75



(a) Centralized EKF

(b) Cascaded EKF



(c) ESKF

(d) Hybrid-Cascaded EKF

Figure 6.8: Estimated heading evolution on the hard track

6.5.3 Innovation Consistency

Statistical consistency on the hard track is evaluated using the Normalized Innovation Squared (NIS) without any retuning of noise parameters.

Table 6.6 reports the innovation statistics for both IMU and AprilTag updates. A detailed representation of AprilTag and IMU NIS can be seen in Figure A.5 in Appendix A.2.

Despite the increased motion complexity, observed innovation statistics remain close to theoretical expectations, indicating that noise parameters identified during the easy-track tuning phase generalize well to more challenging operating conditions.

Table 6.6: Mean NIS on simulated hard track.

Architecture	IMU Mean NIS ($\nu = 1$)	Tag Mean NIS ($\nu = 2$)
Centralized EKF	0.99	1.301
Cascaded EKF	0.99	1.278
ESKF	0.99	1.351
Hybrid-Cascaded EKF	0.99	1.01

Outlier percentages at 99% confidence level are reported in Table 6.7.

Table 6.7: Outlier percentage on hard track.

Architecture	IMU > 99%	Tag > 99%
Centralized EKF	0.79%	0.24%
Cascaded EKF	0.77%	0.17%
ESKF	0.79%	0.17%
Hybrid-Cascaded EKF	0.77%	0.06%

Similarly to the results obtained on the easy track, the IMU outlier rates closely match the theoretical expectation of 1%: in the same way, the observed outlier percentages of the AprilTag updates remain significantly below the theoretical value, indicating conservativeness in the estimates.

6.5.4 Covariance Envelope Verification

Estimator consistency was further assessed by comparing state estimation errors with the corresponding $\pm 2\sigma$ covariance envelopes, as illustrated in Fig. A.6 in Appendix A.2.

For all architectures, estimation errors remain largely contained within predicted uncertainty bounds, demonstrating consistent covariance propagation even during sustained nonlinear motion.

Temporary increases in estimation error are accompanied by corresponding covariance growth, confirming appropriate uncertainty adaptation during prediction-dominated phases.

6.5.5 Discussion

The hard-track experiment highlights practical differences between the considered estimation architectures that were not observable under easy-track conditions.

While all Kalman-based formulations maintain bounded localization error, differences emerge in the handling of heading uncertainty and its impact on position estimation during prolonged prediction intervals.

The centralized EKF provides fully coupled uncertainty propagation, yielding stable performance but requiring higher computational complexity and tighter numerical conditioning.

The cascaded and error-state formulations achieve comparable accuracy while reducing state coupling, confirming that state partitioning does not inherently degrade estimation performance when cross-correlations are properly handled.

The hybrid-cascaded architecture demonstrates robustness comparable to the fully coupled filters while maintaining a modular structure. The explicit incorporation of heading uncertainty into the position update enables reliable operation during extended turning motion without introducing instability.

These results indicate that architectural simplifications can be introduced without sacrificing localization accuracy, provided that uncertainty propagation between subsystems is carefully modeled.

Although the hard-track scenario introduces increased geometric complexity and longer prediction intervals, slightly improved accuracy metrics are occasionally observed compared to the easy-track configuration. This effect arises from the richer motion excitation generated by sustained turning maneuvers. Continuous angular motion improves gyroscope bias observability [13, 15], enabling more accurate bias estimation and consequently reducing long-term heading drift. Since heading errors directly propagate into position estimation, improved bias convergence may compensate for reduced measurement availability, leading to comparable or marginally lower RMSE values despite the increased trajectory complexity.

6.6 Overall Architectural Comparison

The experimental results obtained across both easy and hard track configurations enable a comprehensive comparison of the considered Kalman filter architectures.

Under moderate operating conditions with frequent landmark observations, all architectures exhibit nearly identical performance. In this regime, measurement updates dominate estimation behavior [8], limiting the influence of covariance structure and state representation.

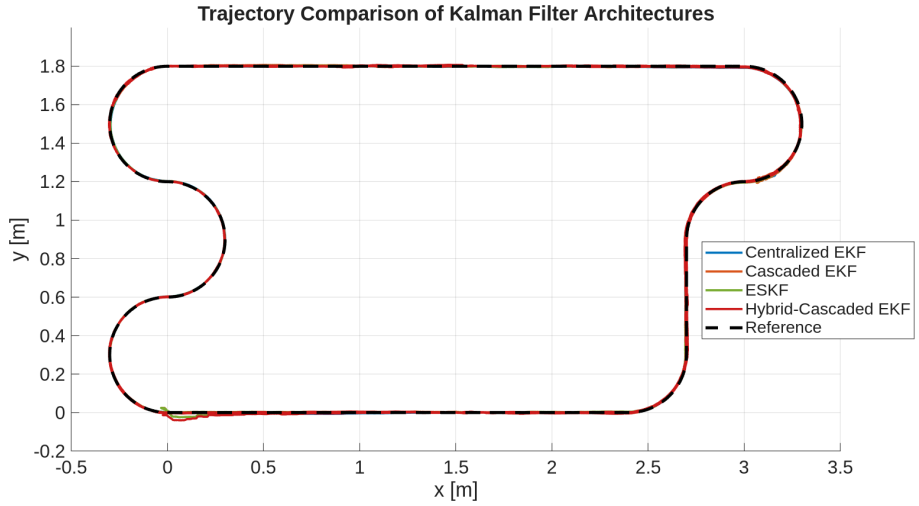


Figure 6.9: Comparison of estimated trajectories obtained with the different Kalman filter architectures in simulation.

Although trajectory accuracy remains largely unchanged, increased motion complexity highlights differences in how each architecture propagates and corrects uncertainty during prolonged prediction phases. Filters capable of accurately propagating heading uncertainty maintain improved consistency during extended prediction intervals, reducing transient position errors.

The hybrid-cascaded formulation achieves performance comparable to centralized estimation while offering improved modularity and reduced implementation complexity. This demonstrates that careful uncertainty coupling between decoupled subsystems can recover the benefits of fully joint estimation.

As it can be seen in Figures 6.9 and 6.10, the results confirm that estimator architecture primarily affects robustness and consistency rather than nominal accuracy, emphasizing the importance of covariance modeling in practical robotic localization systems.

The comparative analysis presented in this chapter demonstrates that statistically consistent localization can be achieved using multiple Kalman filter formulations,

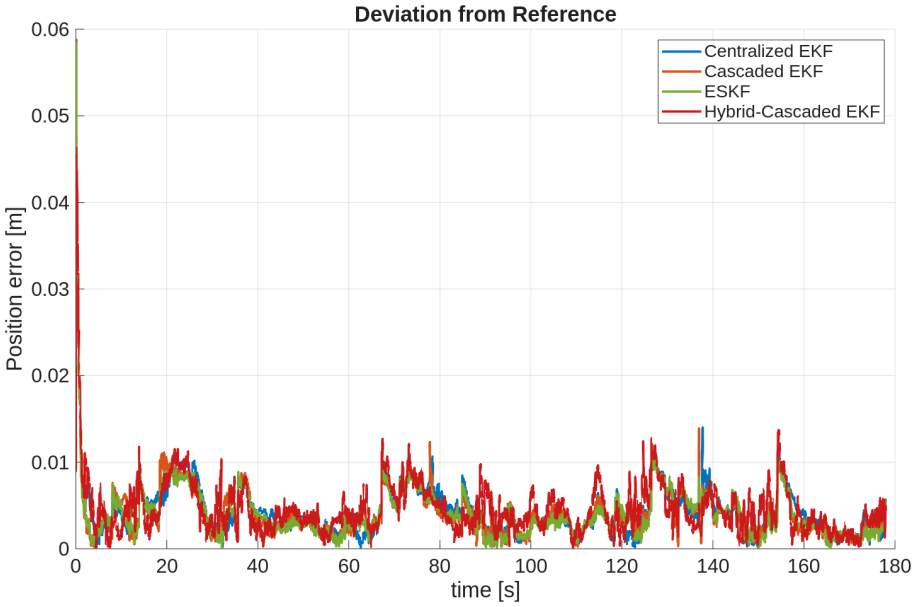


Figure 6.10: Position error with respect to the reference trajectory for all estimator architectures. After the initial convergence phase, all filters maintain millimeter-level accuracy with comparable steady-state performance, confirming that estimator architecture primarily influences robustness rather than nominal accuracy.

each characterized by different structural trade-offs.

The implications of these findings and directions for future development are discussed in the following chapter.

7

Experimental Validation

This chapter presents the experimental validation of the localization architectures previously analyzed in simulation. All estimators were deployed on a physical Duckiebot operating in a Duckietown environment in order to evaluate localization performance under real sensing conditions, actuation uncertainty, and asynchronous sensor measurements.

The objective of the experiments is to verify whether the accuracy and statistical consistency observed in simulation are preserved when the estimators operate on real-world data without parameter retuning. All process and measurement noise parameters correspond exactly to those identified during simulation tuning, presented in Section 6.2, and were applied unchanged throughout the experimental evaluation.

7.1 Experimental Platform

7.1.1 Duckiebot Hardware Platform

Experiments were conducted using a standard Duckiebot platform, a small-scale differential-drive autonomous robot designed for education and research within the Duckietown ecosystem [1]. The robot integrates low-cost sensing and embedded computation, making it representative of resource-constrained autonomous systems.

The platform is equipped with:

- incremental wheel encoders providing odometric motion measurements,
- an onboard inertial measurement unit (IMU) measuring angular velocity,
- a monocular forward-facing camera used for visual landmark detection,

- an embedded single-board computer executing perception and control software while recording all sensor measurements for subsequent offline processing.

The robot follows a differential-drive kinematic configuration consistent with the motion model introduced in Chapter 3. Encoder measurements provide incremental wheel displacements used for state propagation, while gyroscope measurements enable online estimation of heading bias.

Figure 7.1 shows the estimated gyroscope bias during a representative experimental run. The bias rapidly converges during the initial motion phase and remains approximately bounded throughout the trajectory. Small fluctuations are expected due to the interaction between encoder-derived angular velocity and gyroscope measurements during turning maneuvers.

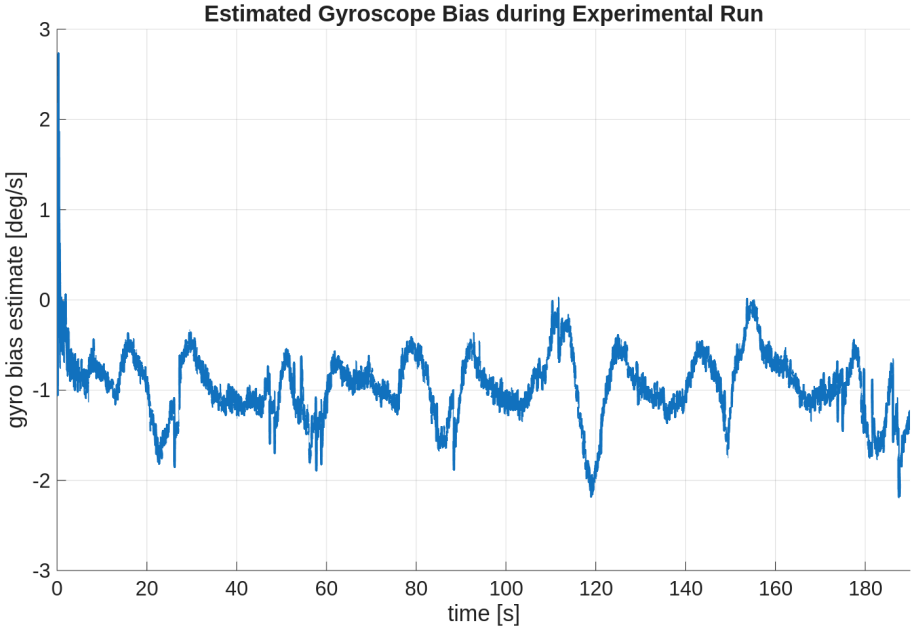


Figure 7.1: Estimated gyroscope bias during a representative experimental run on the Duckiebot.

7.1.2 Visual Landmark Infrastructure

Absolute pose corrections are obtained through AprilTag landmarks placed at known positions along the Duckietown track. Landmark detection is performed using the AprilTag visual fiducial system [7], which provides robust identification and relative pose estimation from monocular camera images. Figure 7.2 shows the Apriltags markers on the easy track.

Each detected tag supplies range and bearing information relative to the robot frame, enabling periodic correction of accumulated odometric drift. This measurement

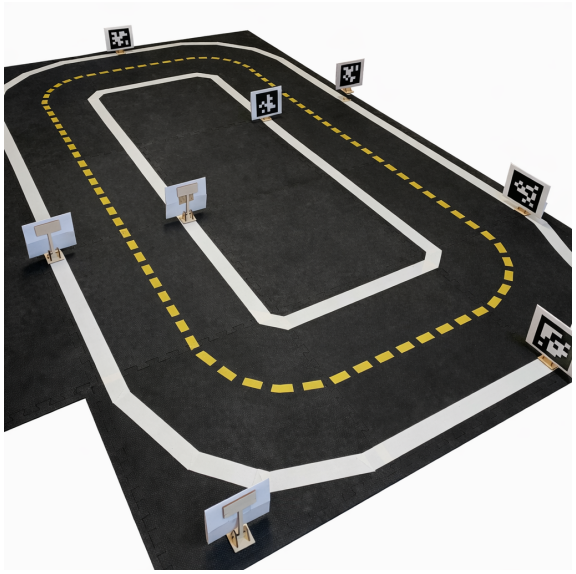


Figure 7.2: Easy track with Apriltags

configuration directly corresponds to the landmark observation model adopted in the simulation framework.

7.1.3 Ground-Truth System

To enable quantitative performance evaluation, ground-truth robot pose was obtained using an external motion-capture system. The system provides high-accuracy measurements of planar position and heading, allowing computation of localization errors and consistency metrics equivalent to those used in simulation.

Ground-truth data were time-synchronized with onboard sensor measurements and recorded together within ROS bag files. State estimation was subsequently performed offline by replaying the recorded measurements in chronological order, enabling direct comparison between estimated trajectories and ground truth.

7.1.4 Experimental Conditions

All experiments were performed under fully autonomous operation. Robot motion along the track was generated using a custom lane-following controller developed specifically for this work.

The controller processes monocular camera images to estimate the relative pose of the vehicle with respect to lane centroids and generates linear and angular velocity commands accordingly. This approach enables continuous excitation of both translational and rotational dynamics while maintaining repeatable experimental

conditions across estimator architectures.

In addition to the lane-following module, dedicated software nodes were implemented for wheel velocity control and AprilTag-based landmark detection. Together, these components provide the sensing and control pipeline required for real-time localization experiments.

Only the hard-track configuration was considered experimentally. As demonstrated in Chapter 6, easy-track scenarios primarily serve parameter tuning purposes in simulation, whereas the hard track provides a more representative evaluation of estimator robustness under prolonged prediction intervals and sustained turning motion.

All sensor measurements collected during the experiments were recorded and later processed offline in MATLAB. Localization was performed by replaying the time-ordered sensor data, preserving the original asynchronous measurement timing.

All estimators employ the same models and noise parameters identified during simulation tuning, without modification based on experimental data. Consequently, observed performance differences can be attributed exclusively to estimator architecture rather than experiment-specific parameter adjustment. Offline replay ensures identical input data for all estimator architectures, enabling a fair and fully reproducible comparison.

7.2 Experimental Software Architecture

Experimental data acquisition and robot operation were implemented through a modular software architecture based on the Robot Operating System (ROS).

Three custom ROS nodes were developed to provide autonomous motion, low-level actuation control, and visual landmark detection. Together, these components generate the synchronized multi-sensor dataset used for offline localization analysis.

Unlike the simulation framework, state estimation is not executed onboard the robot. Instead, all sensor measurements and control signals are recorded into ROS bag files and subsequently processed offline in MATLAB, where the localization algorithms described in previous chapters are applied.

7.2.1 Lane-Following Node

Autonomous motion along the Duckietown track is achieved through a custom lane-following node developed specifically for this work.

The node processes images acquired from the onboard monocular camera to estimate the relative lateral displacement and heading error of the robot with respect to lane markings. Based on these quantities, linear and angular velocity commands are generated using a feedback control law.

The objective of the controller is not high-precision tracking, but rather the generation of repeatable trajectories containing both straight segments and sustained turning maneuvers. Such motion ensures sufficient excitation of the robot dynamics, which is necessary for meaningful evaluation of estimator observability and bias convergence properties.

7.2.2 Wheel Control Node

A dedicated wheel-control node converts velocity commands produced by the lane-following module into motor actuation signals.

The node implements closed-loop wheel velocity regulation using encoder feedback, compensating for actuator nonlinearities and wheel asymmetries. This layer ensures stable execution of commanded motions while providing incremental wheel displacement measurements required for odometric state propagation.

Encoder measurements are published as time-stamped ROS messages and recorded together with all other sensor data.

7.2.3 AprilTag Detection Node

Visual landmark observations are obtained through a dedicated AprilTag detection node.

Camera images are processed using the AprilTag detection algorithm [7], which provides identification of visible landmarks together with relative pose estimates. Detected tags are converted into range and bearing measurements consistent with the observation model introduced in Chapter 3.

Each detection is published with an associated timestamp and landmark identifier, enabling direct reconstruction of the measurement sequence during offline estimator execution.

7.2.4 Data Recording and Offline Processing

All sensor measurements and ground-truth data are recorded during experiments using ROS bag files. The recorded dataset includes:

- wheel encoder measurements,

- IMU angular velocity measurements,
- AprilTag landmark observations,
- motion-capture ground-truth pose.

After data collection, ROS bag files are imported into MATLAB, where measurements are reordered according to their timestamps and replayed sequentially. Localization is then performed offline using the Kalman filter architectures presented in Chapters 3 and 4.

This procedure guarantees that all estimators operate on identical measurement sequences while preserving the asynchronous timing characteristics of the real robotic system. Fig. 7.3 illustrates the ROS architecture used for the experiments.

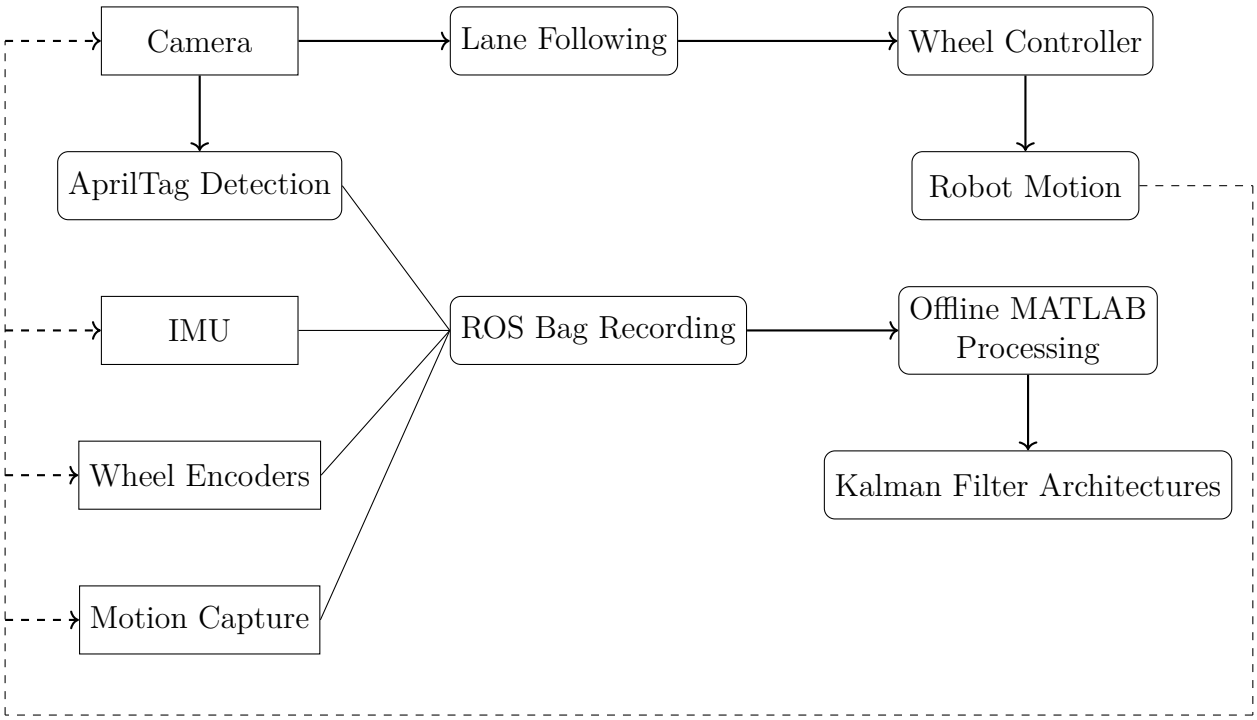


Figure 7.3: Experimental data-flow architecture used for real-world localization validation. Sensor measurements and control signals are recorded through ROS and subsequently processed offline in MATLAB for estimator evaluation.

7.3 Experimental Methodology

This section describes the experimental protocol adopted to evaluate the localization architectures under real-world operating conditions.

7.3.1 Dataset Acquisition

Experiments were conducted on the hard-track configuration described in Chapter 5. The robot was operated under fully autonomous lane-following control for multiple runs, each lasting approximately 180 seconds.

During each run, all available sensor measurements were recorded in ROS bag files. No manual intervention was performed during data collection, and all runs were executed under comparable environmental conditions.

7.3.2 Offline Estimator Execution

Localization was performed offline by importing recorded ROS bag files into MATLAB and replaying the time-ordered sensor measurements sequentially.

The asynchronous timing structure of the real system was preserved, ensuring that prediction and correction steps occur according to the original measurement timestamps. This procedure guarantees that all estimator architectures process identical input data under identical temporal conditions.

No parameter retuning was performed for experimental validation. Process and measurement noise parameters correspond exactly to those identified during simulation tuning (Chapter 6). Consequently, performance differences observed experimentally can be attributed solely to estimator architecture rather than to experiment-specific calibration.

7.3.3 Evaluation Metrics

Performance evaluation follows the same criteria adopted in the simulation study in order to enable direct comparison between simulated and real-world results.

The following metrics are computed:

- planar position error with respect to motion-capture ground truth,
- heading error,
- Root Mean Square Error (RMSE) for position and heading,
- Normalized Innovation Squared (NIS) consistency statistics for both IMU and landmark measurements.

These metrics allow assessment of both estimation accuracy and statistical consistency under real sensing conditions.

7.4 Performance Evaluation

7.4.1 Trajectory Comparison

Figure 7.4 shows the estimated trajectories obtained with the four Kalman filter architectures during a representative experimental run on the hard-track configuration. The selected run exhibits error statistics close to the mean values reported in Table 7.1, and is therefore representative of typical estimator behavior.

All architectures maintain bounded localization error throughout the experiment despite real-world disturbances, wheel slip, and sensor noise. The estimated trajectories closely follow the motion-capture reference, with no evidence of divergence or instability.

Small deviations are visible during prolonged turning segments, where heading uncertainty temporarily propagates into position error between successive landmark observations. These deviations are rapidly corrected once visual measurements become available, confirming stable interaction between prediction and update phases under real sensing conditions.

Overall, the strong overlap between estimated and ground-truth trajectories indicates that the architectural differences primarily influence consistency and robustness rather than nominal path tracking capability.

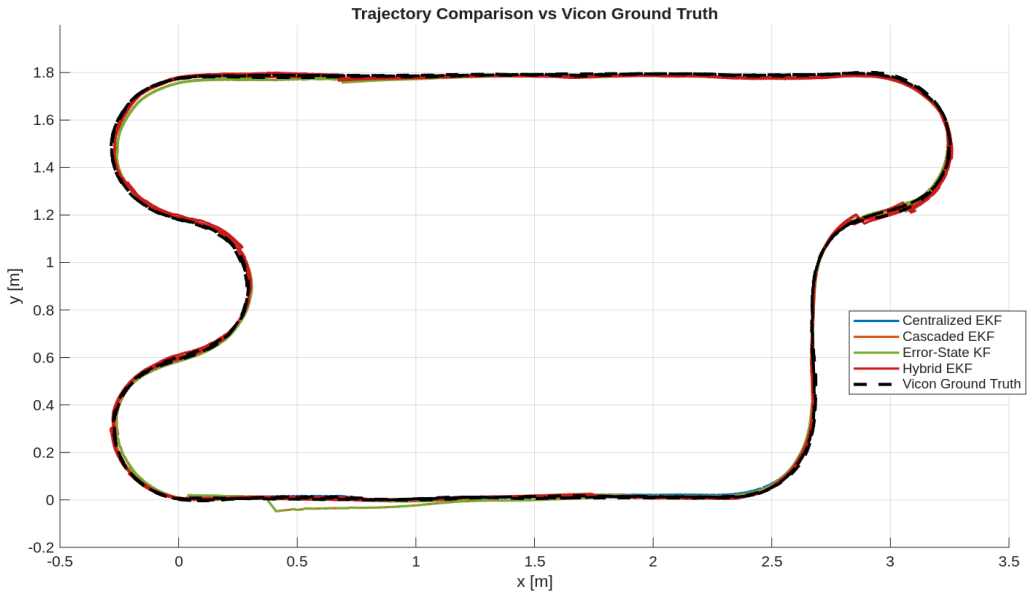


Figure 7.4: Comparison of estimated trajectories obtained with the different Kalman filter architectures with the Duckiebot.

7.4.2 Position and Heading Errors

The quantitative accuracy results obtained over four independent experimental runs are summarized in Table 7.1. For each architecture, the reported values correspond to the mean RMSE across runs together with the associated standard deviation, which reflects sensitivity to initial conditions and measurement variability.

Table 7.1: Experimental RMSE over four runs (mean ± std).

Architecture	Position RMSE [m]	Heading RMSE [deg]
Centralized EKF	0.0208 ± 0.0047	2.54 ± 0.45
Cascaded EKF	0.0187 ± 0.0026	2.35 ± 0.37
ESKF	0.0176 ± 0.0037	2.24 ± 0.41
Hybrid EKF	0.0156 ± 0.0015	2.06 ± 0.48

The Hybrid EKF achieves the lowest average position and heading RMSE, with a mean planar error of 1.56 cm and a heading error of 2.06°. The reduced standard deviation observed for the Hybrid architecture indicates improved robustness to variations in initial conditions and measurement noise across repeated experiments.

The ESKF and Cascaded formulations exhibit comparable performance, with slightly larger mean errors but moderate run-to-run variability. In contrast, the Centralized EKF shows the highest standard deviation in position error, suggesting greater sensitivity to initial state offsets and accumulated numerical coupling effects under real-world disturbances.

It is important to note that, although differences between architectures are measurable, all filters maintain centimeter-level position accuracy and heading errors below 3°, demonstrating stable and bounded localization performance under real sensing conditions.

The experimental results preserve the architectural trends identified in simulation. While absolute errors increase due to real-world disturbances, the relative behavior of the considered formulations remains consistent.

Figures 7.5 and 7.6 show the time evolutions of the position and heading errors of the four filter architectures on the Duckiebot.

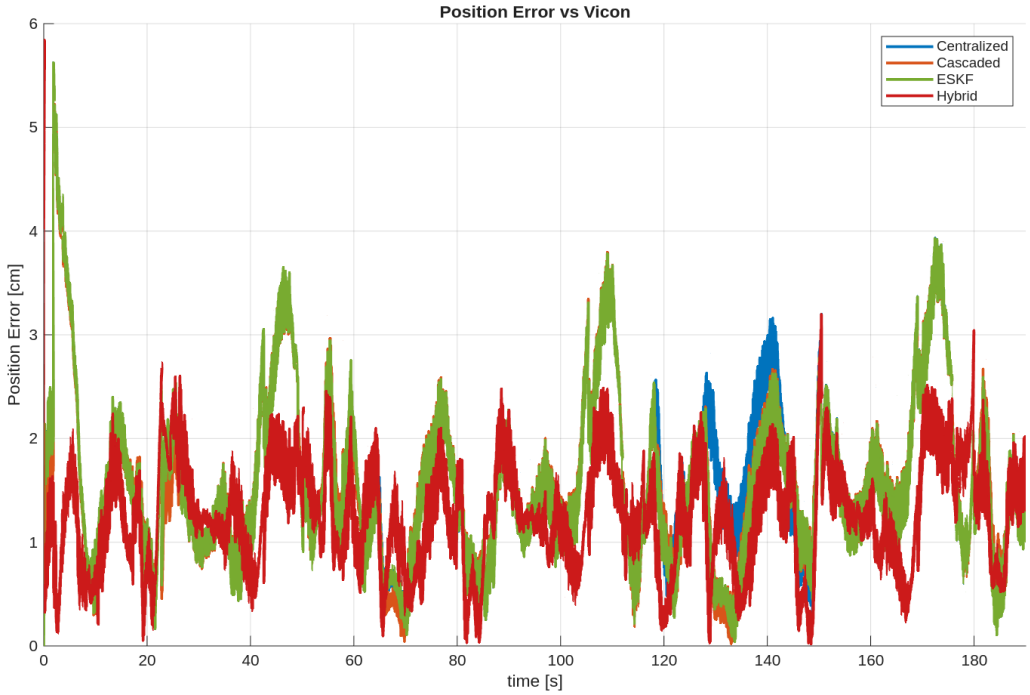


Figure 7.5: Position error with respect to the reference trajectory for all estimator architectures with the Duckiebot.

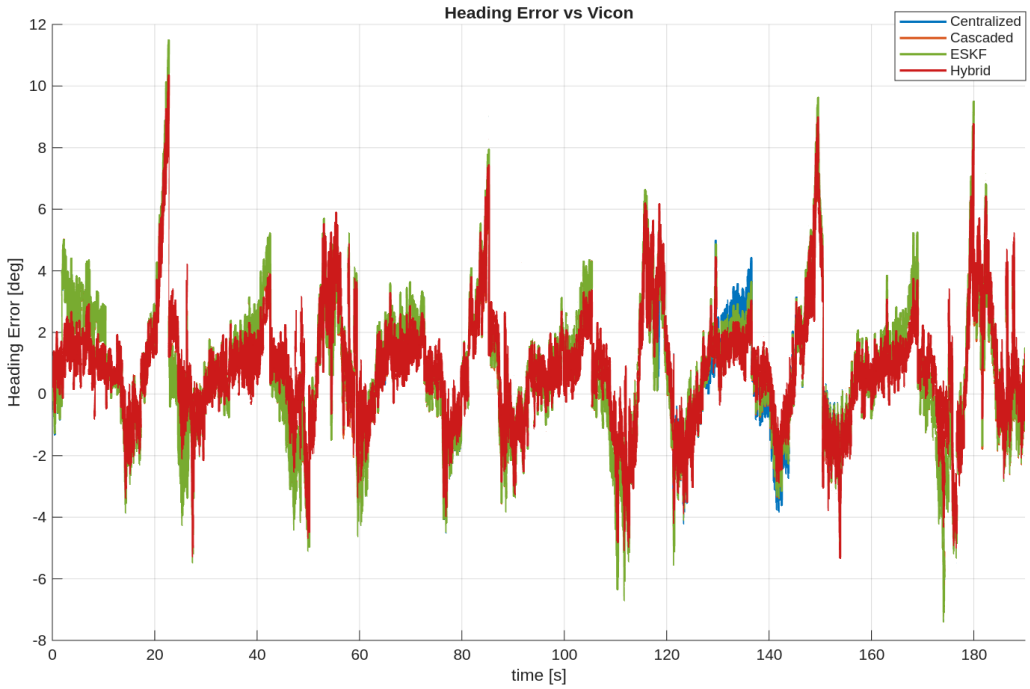


Figure 7.6: Heading error with respect to the reference trajectory for all estimator architectures with the Duckiebot.

7.4.3 Innovation Consistency Analysis

Statistical consistency of the estimators under real sensing conditions is evaluated using the Normalized Innovation Squared (NIS) for both IMU and AprilTag updates. The same chi-square confidence bounds adopted in simulation are used here, without parameter retuning.

Tables 7.2 and 7.3 report the mean NIS values and the percentage of innovations exceeding the 99% confidence threshold across the experimental runs.

Table 7.2: Mean NIS with the Duckiebot.

Architecture	IMU Mean NIS ($\nu = 1$)	Tag Mean NIS ($\nu = 2$)
Centralized EKF	0.90	1.132
Cascaded EKF	0.90	1.080
ESKF	0.89	1.019
Hybrid-Cascaded EKF	0.89	1.014

Table 7.3: Outlier percentage with the Duckiebot.

Architecture	IMU $> 99\%$	Tag $> 99\%$
Centralized EKF	0.47%	1.33%
Cascaded EKF	0.51%	0.67%
ESKF	0.47%	0.60%
Hybrid-Cascaded EKF	0.47%	1.46%

For a statistically consistent estimator with $\nu = 1$ degree of freedom, the expected mean NIS equals 1 and approximately 1% of the innovations are expected to exceed the 99% confidence bound.

For the IMU updates, the observed mean values remain close to unity, while the measured outlier percentages are slightly below the theoretical expectation. This indicates mildly conservative modeling of angular velocity noise under real experimental conditions. Although occasional large innovation spikes are visible in the time histories, this behavior is characteristic of the heavy-tailed nature of the $\chi^2(1)$ distribution and does not indicate estimator divergence.

The fact that innovation peaks occur simultaneously across all architectures suggests that these events are primarily driven by real sensor disturbances or transient modeling mismatch rather than architectural deficiencies.

For AprilTag updates ($\nu = 2$), the observed mean NIS values remain below the theoretical expectation of 2, again indicating slightly conservative covariance

modeling. Outlier percentages at the 99% confidence level remain close to the theoretical value of 1%, with small deviations across architectures. This behavior confirms that landmark measurement uncertainty is reasonably modeled despite real-world perception noise and occasional detection inaccuracies.

Compared to simulation, slightly larger innovation variability is observed, reflecting unmodeled disturbances such as wheel slip, camera latency, and actuator nonlinearities. Nevertheless, innovation statistics remain bounded and close to theoretical predictions, demonstrating that noise parameters identified in simulation generalize effectively to real-world operation.

Figures 7.7 and 7.8 show the time evolution of the IMU and AprilTag NIS for the Hybrid-Cascaded architecture during a representative experimental run. Corresponding plots for the remaining architectures are provided in Figure B.2 in Appendix B.

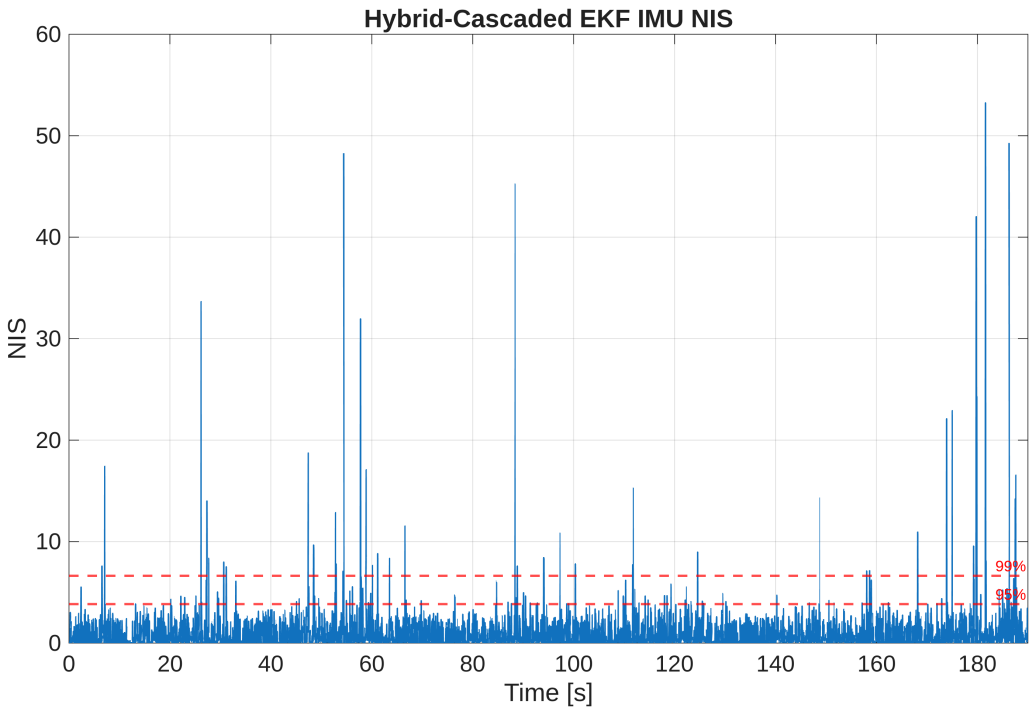


Figure 7.7: Hybrid-Cascaded EKF IMU NIS time evolution on the Duckiebot

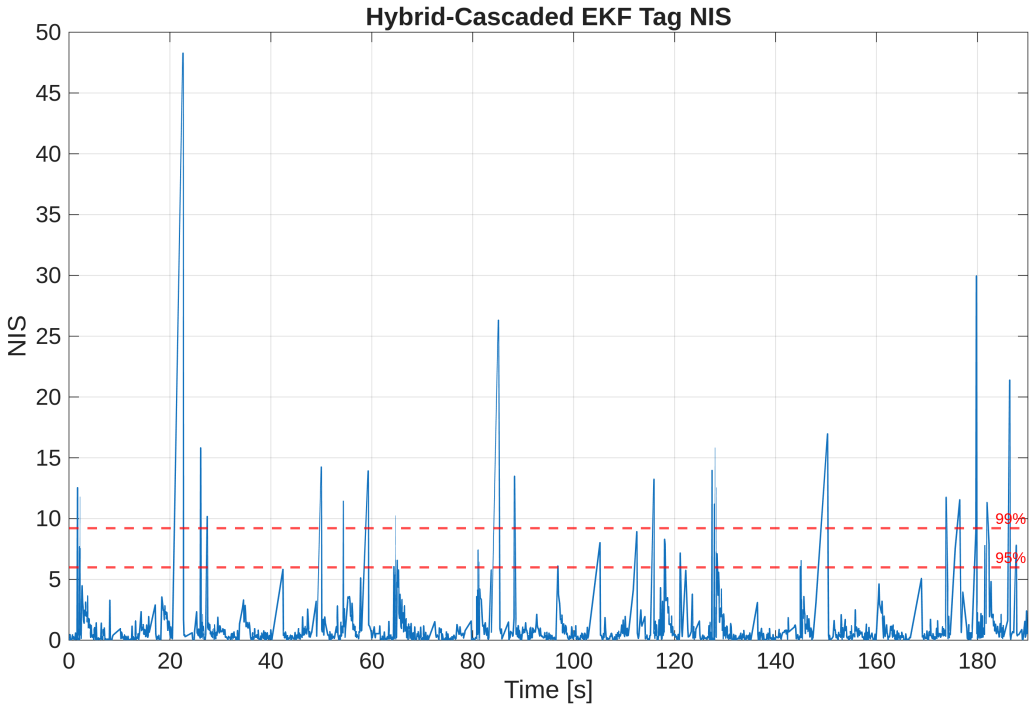


Figure 7.8: Hybrid-Cascaded EKF AprilTag NIS time evolution on the Duckiebot

7.4.4 Covariance Envelope Verification

In addition to innovation statistics, estimator consistency can be evaluated by comparing the true estimation error with the covariance predicted by the filter. For a statistically consistent estimator, the true error should remain bounded within the corresponding covariance envelope with high probability.

Figure B.3 in Appendix B illustrates this comparison for both planar position error and heading error for all filter architectures, Figure 7.9 reports the consistency results obtained with the Hybrid-Cascaded EKF architecture. The blue curves represent the true estimation error computed with respect to the motion-capture ground truth, while the dashed curves correspond to the predicted 2σ covariance bound derived from the state covariance matrix.

For all architectures, the position error remains well bounded by the predicted uncertainty envelope during most of the experiment. Short transient peaks occur primarily during initialization and during segments with limited landmark visibility, where prediction uncertainty temporarily increases before being corrected by subsequent visual measurements. These peaks remain consistent with the expected probabilistic interpretation of the 2σ bound.

A similar behavior is observed for the heading error. The predicted covariance successfully captures the overall magnitude of the orientation uncertainty, and the true

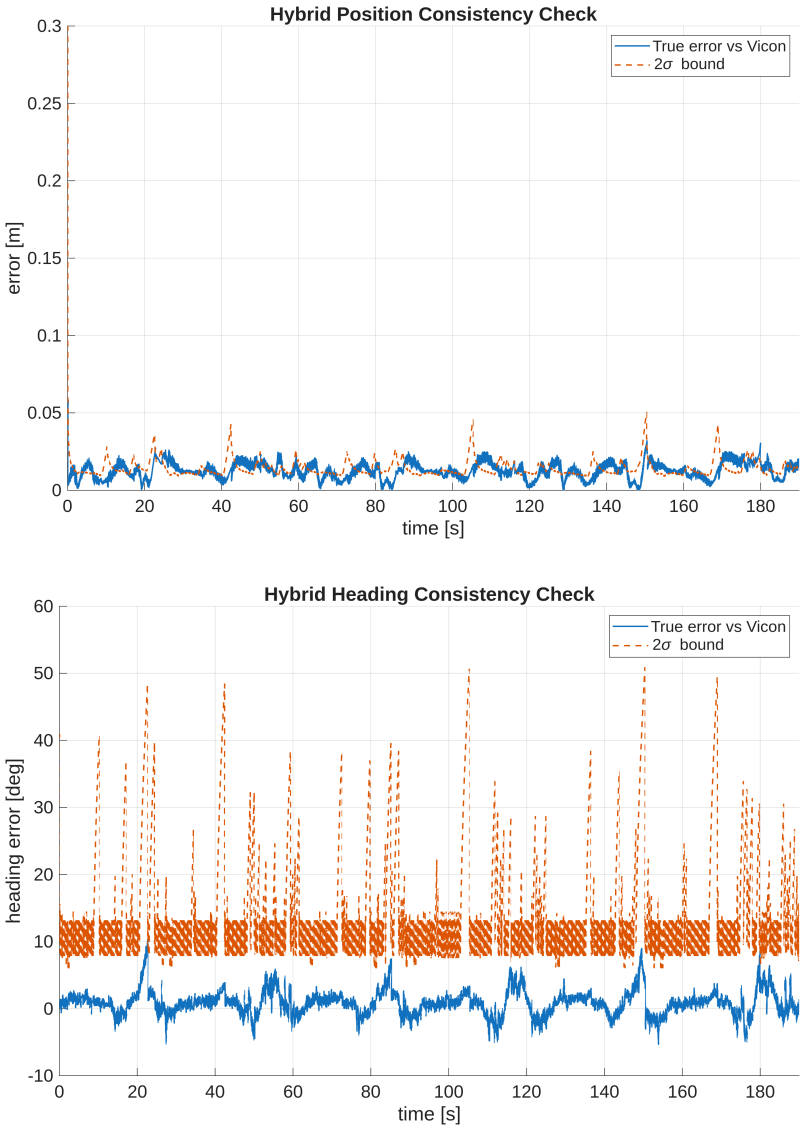


Figure 7.9: Position (top) and heading (bottom) consistency of the Hybrid-Cascaded EKF with the Duckiebot.

error remains largely contained within the predicted envelope. Occasional deviations occur during rapid turning maneuvers, where modeling inaccuracies and encoder noise can temporarily increase the orientation error before subsequent landmark updates reduce the uncertainty.

Across all architectures, the agreement between predicted covariance and true estimation error indicates that the filters provide a realistic estimate of their own uncertainty. This confirms that the noise parameters identified in simulation remain valid under real-world operating conditions and that the estimators maintain statistical consistency when deployed on the physical Duckiebot platform.

7.5 Robustness Experiments

7.5.1 Landmark dropout region

To evaluate the robustness of the localization algorithms under limited perception conditions, an additional experiment was conducted in which AprilTag measurements were artificially disabled in a portion of the environment. In particular, landmark updates were ignored whenever the robot position satisfied $x < 0.5\,\mathrm{m}$, creating a region where the estimator must rely exclusively on wheel odometry and IMU measurements.

Figure 7.10 shows the resulting trajectory obtained with the Hybrid-Cascaded EKF, starting from $(0, 0)$ and proceeding anticlockwise along the track, compared with the motion-capture ground truth. Figure 7.11 shows the relative position and heading errors. When the robot enters the landmark-deprived region, the estimator operates in pure prediction mode and the trajectory gradually diverges from the reference path due to accumulated odometric and heading errors. This effect is particularly visible during the tight turns in the lower-left portion of the track.

Once the robot re-enters an area where AprilTag measurements become available again, the estimator rapidly corrects the accumulated drift through landmark updates and the trajectory converges back to the ground-truth path. This behavior confirms that, although visual measurements are essential to bound long-term drift, the filter remains stable and capable of recovering from temporary loss of external observations.

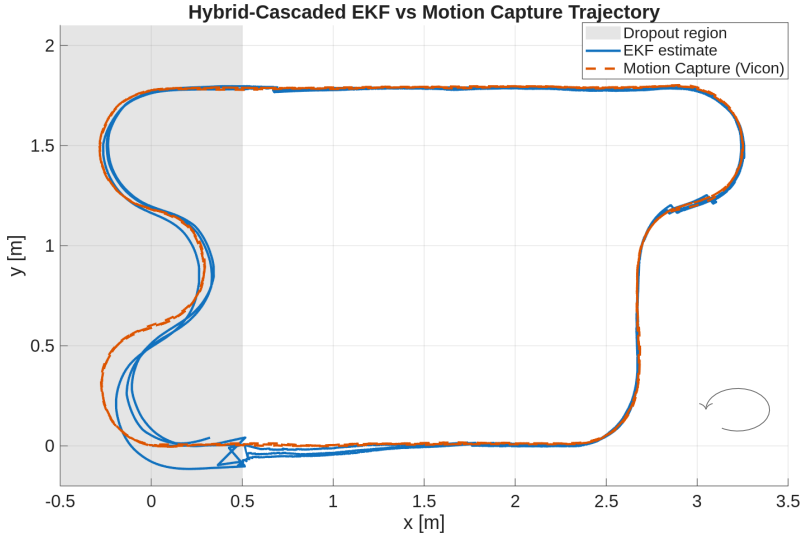


Figure 7.10: Trajectory estimated by the Hybrid–Cascaded EKF when AprilTag updates are disabled in the region $x < 0.5\,\mathrm{m}$ (shaded area). The gray arrow indicates the anticlockwise direction of motion.

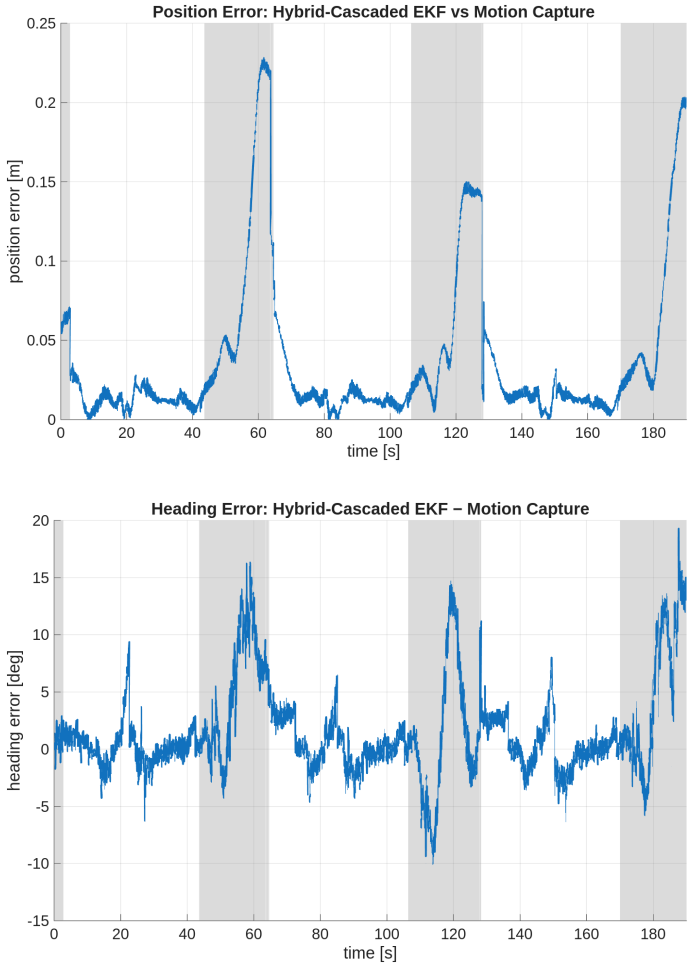


Figure 7.11: Position error (top) and heading error (bottom) of the Hybrid–Cascaded EKF with respect to the motion-capture ground truth during the landmark dropout experiment. The shaded region corresponds to the portion of the trajectory where AprilTag updates are disabled ($x < 0.5\,\mathrm{m}$).

The degradation in accuracy is reflected in the error statistics: the position RMSE increases from 1.56 cm in the nominal experiment to 6.2 cm when landmark measurements are disabled in the region $x < 0.5\,\mathrm{m}$, while the heading RMSE increases from 2.06° to 4.3°.

7.5.2 Reduced landmark density

To further evaluate estimator robustness, an additional experiment was performed in which the number of AprilTag landmarks available for localization was artificially reduced. Approximately half of the tags placed along the track were ignored during the estimator update phase, resulting in longer intervals between successive visual corrections.

Figures 7.12 and 7.13 show the trajectory obtained with the Hybrid-Cascaded EKF compared with the motion-capture ground truth under these conditions, and the relative position and heading errors. As expected, the reduced landmark density leads to slightly larger deviations from the reference trajectory, particularly in curved sections of the track where heading uncertainty propagates more strongly into position error.

Despite the reduced frequency of landmark observations, the estimator remains stable and the accumulated drift is corrected whenever a new landmark becomes visible. This experiment demonstrates that the localization framework maintains reliable performance even when the environment provides only sparse visual references.

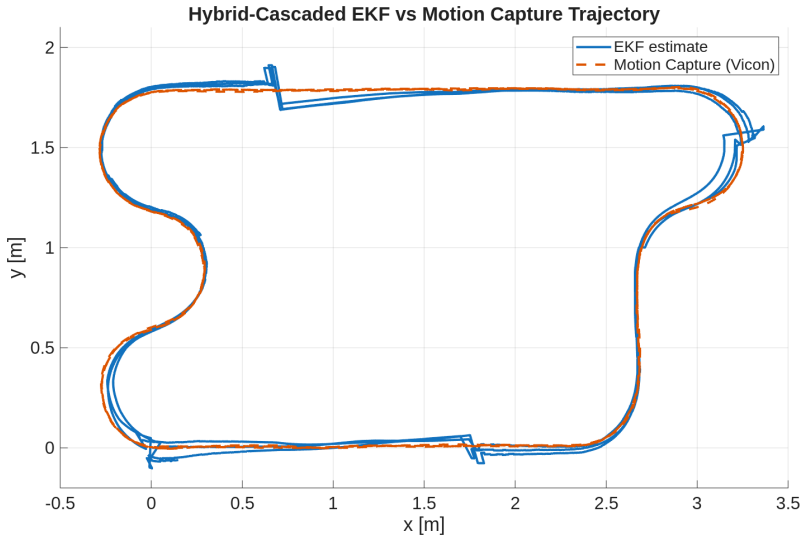


Figure 7.12: Estimated trajectory obtained with the Hybrid–Cascaded EKF when approximately half of the AprilTag landmarks available along the track are ignored during the update stage.

The quantitative error statistics confirm the trends observed in the figures: the position RMSE increases from 1.56 cm in the nominal experiment to 3.5 cm when the landmark density is reduced, while the heading RMSE increases from 2.06° to 3.04°.

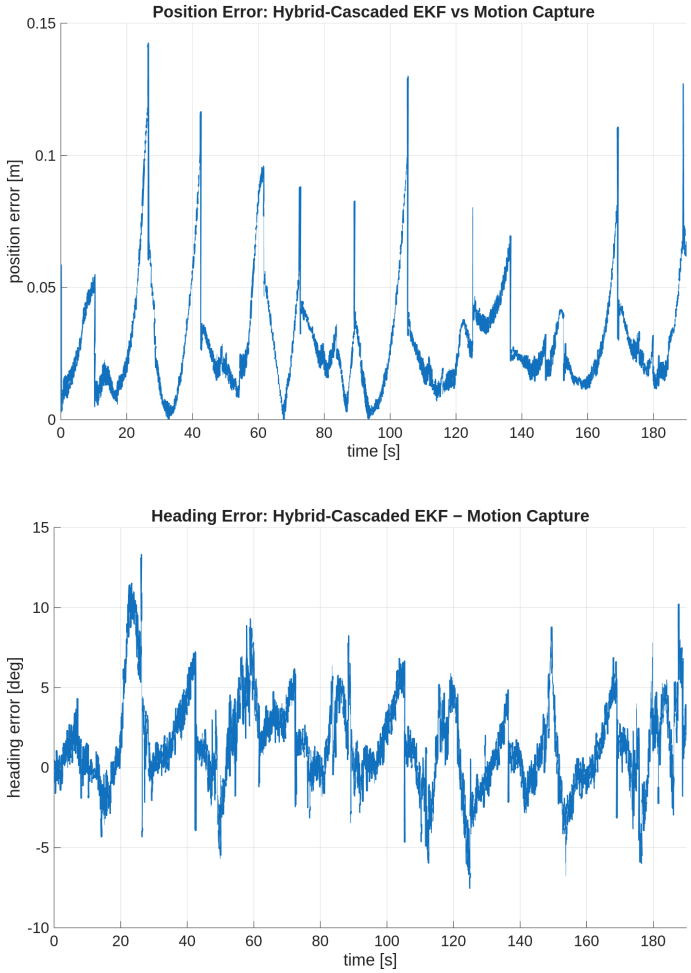


Figure 7.13: Position error (top) and heading error (bottom) of the Hybrid–Cascaded EKF with respect to the motion-capture ground truth when only half of the available AprilTag landmarks are used.

7.5.3 Camera measurement latency

Another robustness experiment was performed to evaluate the sensitivity of the localization framework to delayed visual measurements. In real robotic systems, camera-based perception pipelines often introduce non-negligible processing latency due to image acquisition, feature detection, and pose estimation. To reproduce this effect, artificial latency was introduced in the AprilTag measurements before they were processed by the estimator.

Specifically, each landmark observation was delayed by a fixed time offset before being applied to the filter update step. Three latency values were considered: 0 ms (nominal case), 100 ms, and 200 ms.

Figure 7.14 shows the resulting trajectories estimated by the Hybrid-Cascaded EKF under the different latency conditions, together with the motion-capture ground

truth. As the latency increases, the visual corrections are applied later with respect to the true robot pose, causing temporary discrepancies between the estimated and reference trajectories.

Figures 7.15 illustrate the corresponding position and heading errors. The delayed landmark updates lead to larger transient errors, particularly during turns where heading uncertainty propagates more strongly into position drift. When the corrections are eventually applied, the estimator gradually compensates for the accumulated error.

Despite the delayed perception feedback, the estimator remains stable for all tested latency values, demonstrating the robustness of the proposed architecture to moderate sensing delays.

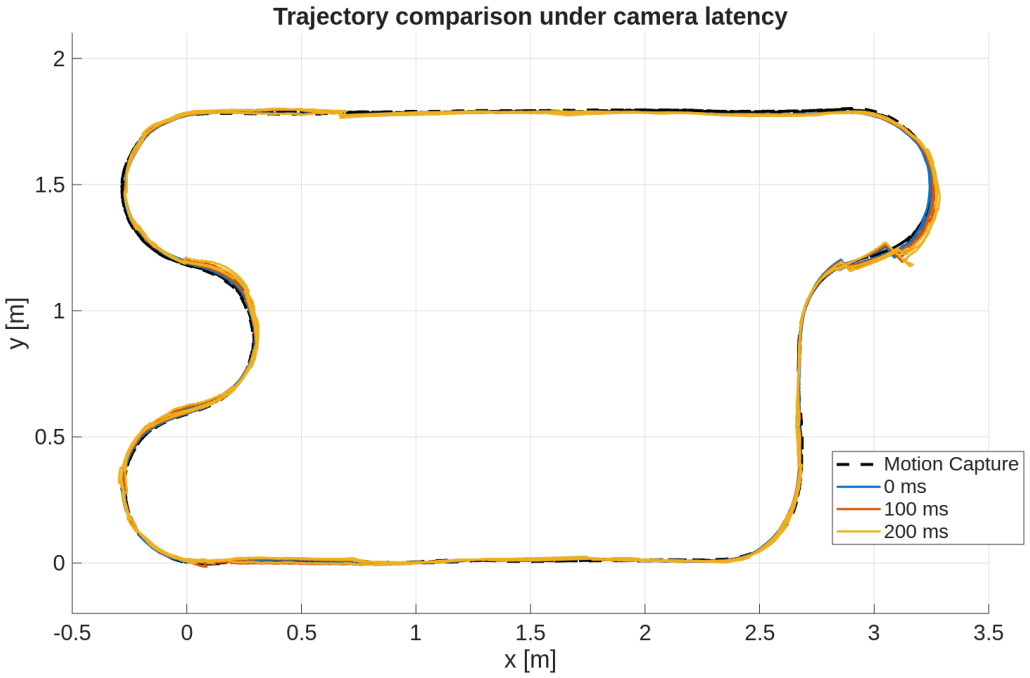


Figure 7.14: Estimated trajectories obtained with the Hybrid–Cascaded EKF under simulated camera latency of 0 ms, 100 ms, and 200 ms, compared with the motion-capture ground truth.

The quantitative results are summarized in Table 7.4. As expected, increasing perception latency leads to progressively larger estimation errors because landmark corrections are applied with increasing delay relative to the true robot pose. The position RMSE grows from 1.56 cm in the nominal case to 4.3 cm at 200 ms latency, while the heading RMSE increases from 2.06° to 5.11°. For larger delays, the error continues to increase approximately linearly, reaching 7.8 cm and 8.68° at 400 ms latency.

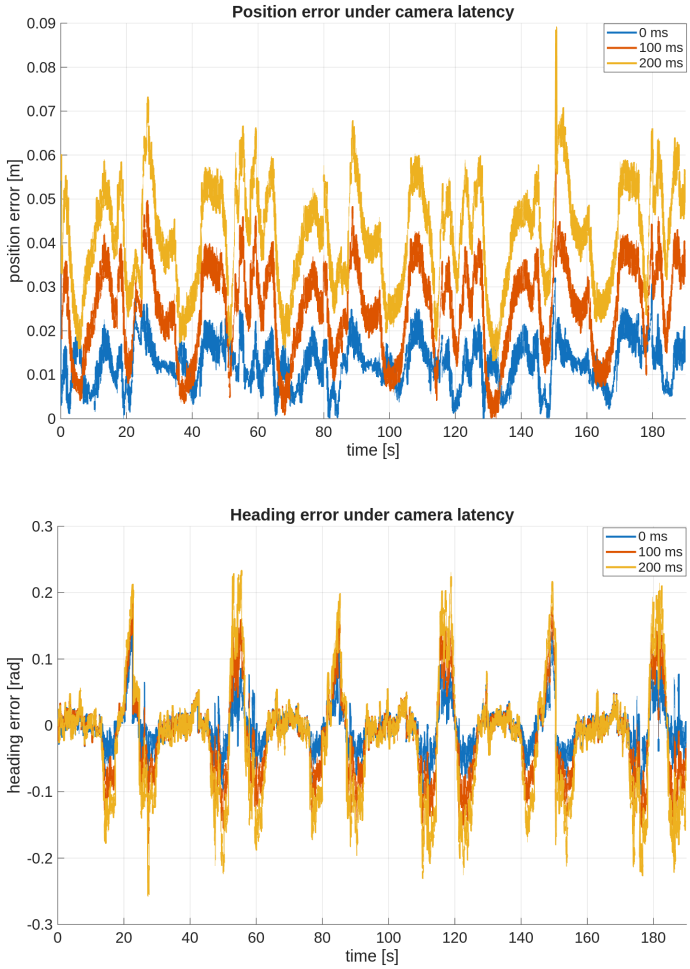


Figure 7.15: Position error (top) and heading error (bottom) of the Hybrid–Cascaded EKF with respect to the motion-capture ground truth under simulated camera latency.

Table 7.4: Position and heading RMSE obtained with the Hybrid–Cascaded EKF under different simulated camera latency values.

Camera latency [ms]	Position RMSE [cm]	Heading RMSE [deg]
0	1.56	2.06
100	2.7	3.38
200	4.3	5.11
300	6.0	6.88
400	7.8	8.68

These results indicate that the estimator remains stable under moderate perception delays (100–200 ms), although estimation accuracy gradually degrades as visual corrections are applied with increasing delay relative to the true robot pose.

7.6 Discussion

The experimental results confirm the main trends previously observed in the simulation study. All considered estimator architectures achieve stable and reliable localization when deployed on the physical Duckiebot platform, maintaining centimeter-level position accuracy and bounded heading errors despite real-world sensing disturbances, actuation uncertainty, and asynchronous measurements.

Among the evaluated approaches, the Hybrid-Cascaded formulation consistently achieves the lowest average RMSE and the smallest run-to-run variability. The ESKF and Cascaded architectures exhibit comparable performance, indicating that both formulations effectively mitigate the numerical coupling effects that can arise in fully centralized filters.

The statistical consistency analysis further confirms that the noise parameters identified in simulation remain valid under real sensing conditions. The measured NIS values remain close to their theoretical expectations and the covariance envelope verification demonstrates that the predicted uncertainty bounds provide a realistic description of the true estimation error. These results indicate that the modeling assumptions adopted in the simulation framework generalize effectively to real-world operation without requiring additional parameter tuning.

The robustness experiments provide further insight into the behavior of the estimator under degraded perception conditions. When landmark observations are temporarily unavailable, the estimator operates in pure prediction mode and odometric drift gradually increases the localization error. However, once visual measurements become available again, the accumulated error is rapidly corrected, demonstrating that the filter remains stable and capable of recovering from temporary loss of external observations.

Similarly, reducing the density of available landmarks leads to slightly larger deviations from the reference trajectory due to longer prediction intervals between visual corrections. Nevertheless, the estimator maintains bounded errors and continues to provide reliable localization even when only sparse visual information is available.

The latency experiments highlight the importance of timely perception updates in vision-based localization systems. As landmark measurements are applied with increasing delay relative to the true robot pose, the accuracy of the visual corrections gradually decreases, leading to larger position and heading errors. Despite this degradation, the estimator remains stable for moderate delays, indicating that the proposed architecture is robust to the processing latency typically encountered in

real camera-based perception pipelines.

Overall, the experimental validation confirms the conclusions drawn from the simulation analysis and demonstrates that the proposed localization framework performs reliably on a real robotic platform. The estimators maintain statistical consistency, exhibit robustness to perception degradations, and achieve accurate localization using only low-cost onboard sensors. These results highlight the practical suitability of the proposed architectures for resource-constrained mobile robots operating in real-world environments with imperfect and asynchronous sensing.

8 Conclusions

This thesis investigated the impact of estimator architecture on multi-sensor localization for a unicycle-model mobile robot. Several Kalman-filter-based formulations were analyzed and compared both in simulation and on a real Duckiebot platform operating in a Duckietown environment.

The work focused on understanding how different architectural choices in Kalman filtering influence estimation accuracy, statistical consistency, and robustness under realistic sensing conditions. In particular, centralized, cascaded, error-state, and hybrid-cascaded estimation architectures were implemented and evaluated using identical motion and measurement models.

8.1 Summary of the Work

The first part of the thesis introduced the theoretical background required for mobile robot localization, including the unicycle kinematic model and nonlinear Kalman filtering techniques. A detailed system model was then developed for the Duckiebot platform, including encoder-based odometry, inertial measurements, and landmark observations obtained through AprilTag detections.

Several estimation architectures were subsequently implemented, including centralized, cascaded, error-state, and hybrid-cascaded Kalman filters. Particular attention was given to the treatment of uncertainty propagation and cross-correlations between different sensor modalities.

A simulation framework was developed in MATLAB/Simulink in order to evaluate the estimators under controlled conditions. The simulation study allowed systematic

analysis of estimation accuracy, innovation statistics, and covariance consistency under both moderate and challenging motion scenarios.

Finally, the proposed localization architectures were validated experimentally on a real Duckiebot platform. Sensor data were collected using a ROS-based software architecture and processed offline to ensure identical measurement sequences for all estimators. The experimental evaluation confirmed the trends observed in simulation and demonstrated the practical feasibility of the proposed localization framework.

8.2 Main Findings

The comparative analysis conducted in this thesis highlights several important observations regarding the role of estimator architecture in multi-sensor robotic localization.

First, under conditions with frequent landmark observations, all considered Kalman filter architectures achieve similar localization accuracy. In this regime, measurement updates dominate the estimation process, limiting the influence of covariance structure and state representation.

Second, architectural differences become more apparent during extended prediction intervals or degraded sensing conditions. Filters capable of correctly propagating heading uncertainty and maintaining consistent covariance structure exhibit improved robustness and reduced transient errors.

Among the evaluated approaches, the Hybrid-Cascaded architecture achieved the lowest average localization error and the smallest run-to-run variability in experimental trials. The results indicate that a carefully designed modular architecture can achieve performance comparable to fully coupled estimators while improving implementation simplicity and numerical stability.

The statistical consistency analysis further demonstrated that the noise parameters identified in simulation generalize well to real-world operation. Innovation statistics remained close to their theoretical expectations and covariance envelope verification confirmed that the predicted uncertainty bounds accurately describe the true estimation error.

Finally, robustness experiments involving landmark dropout, reduced landmark density, and simulated camera latency confirmed that the localization framework maintains stable operation even under degraded perception conditions. Although estimation accuracy decreases when visual corrections are delayed or temporarily unavailable, the estimator remains stable and rapidly recovers once measurements

become available again.

8.3 Future Work

Several directions can be explored to further extend the work presented in this thesis.

A first natural extension concerns the relaxation of the assumption that landmark positions are known a priori. In the present work, AprilTag landmarks are placed at fixed, known locations within the environment and serve as external absolute references for the localization process. In many real-world scenarios, however, such prior knowledge may not be available.

Future work could therefore investigate Simultaneous Localization and Mapping (SLAM) approaches in which both the robot pose and the landmark positions are estimated jointly. In this context, the Kalman-filter-based architectures studied in this thesis could be extended to an EKF-SLAM framework or compared with graph-based formulations. Such an extension would allow the robot to operate in previously unknown environments while progressively building a map of visual landmarks.

Another promising direction concerns the treatment of delayed or out-of-sequence measurements. The latency experiments conducted in this thesis highlight the impact that delayed visual observations can have on estimation accuracy. Future research could therefore explore estimation techniques specifically designed to handle delayed measurements, such as state buffering or out-of-sequence update methods, which may significantly reduce the degradation observed under large perception delays.

Finally, the localization framework could be extended by incorporating additional sensing modalities. In particular, vision-based motion estimation techniques such as visual odometry or feature tracking could provide complementary information between landmark observations. Combining visual odometry with inertial and wheel encoder measurements would allow the system to maintain accurate localization even in environments where artificial landmarks are sparse or temporarily unavailable.

These extensions would further improve the robustness and generality of the localization framework and enable its application to a wider range of autonomous robotic platforms and environments.

A

Detailed Simulation Results

This appendix reports detailed per-architecture simulation results supporting the comparative analysis presented in Chapter 6. Individual estimator behaviors are shown separately to provide complete experimental documentation.

A.1 Simulation - Easy Track

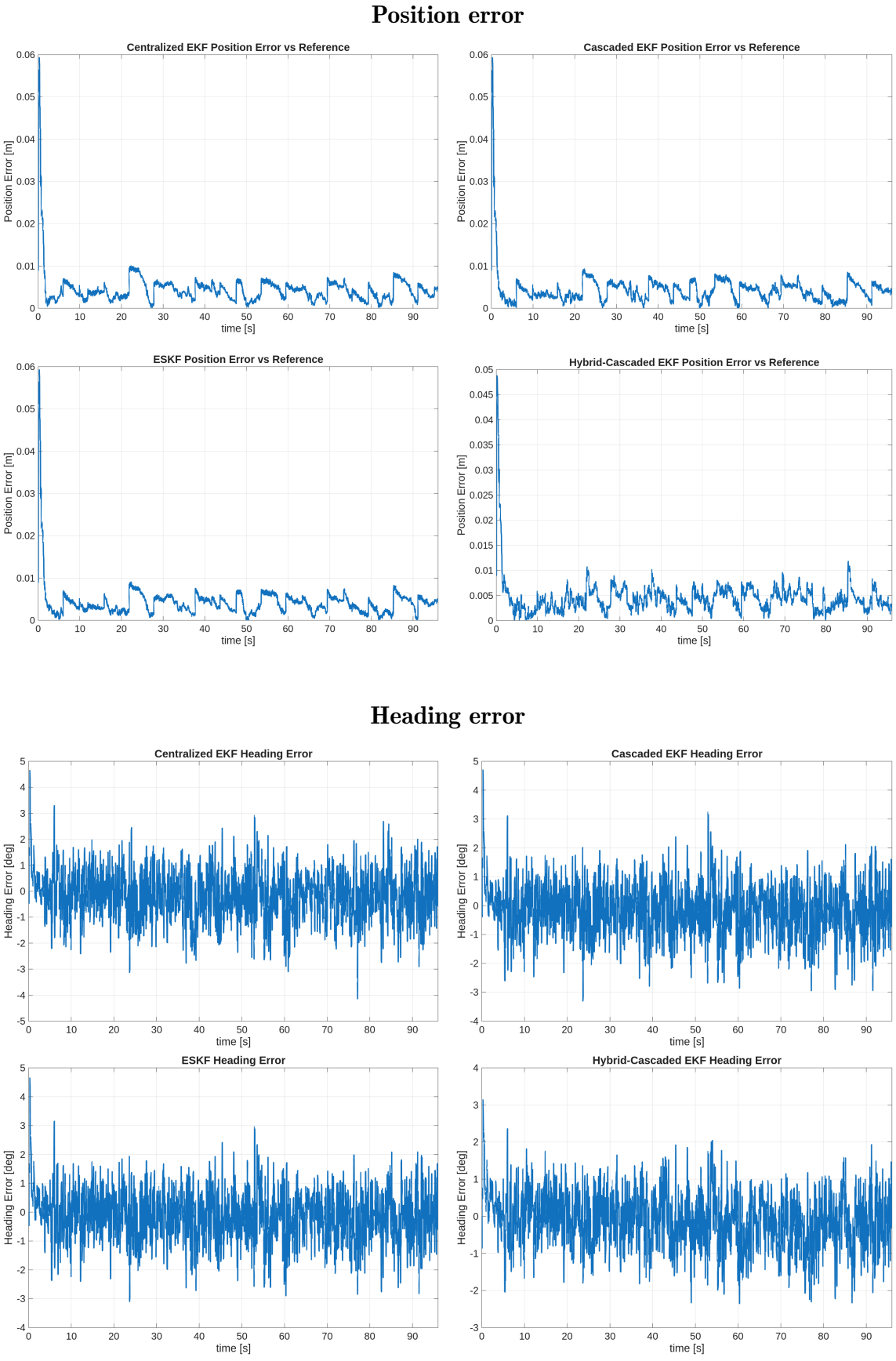


Figure A.1: Planar position errors (top) and heading errors (bottom) with respect to the reference trajectory for all filter architectures on the simulated easy track.

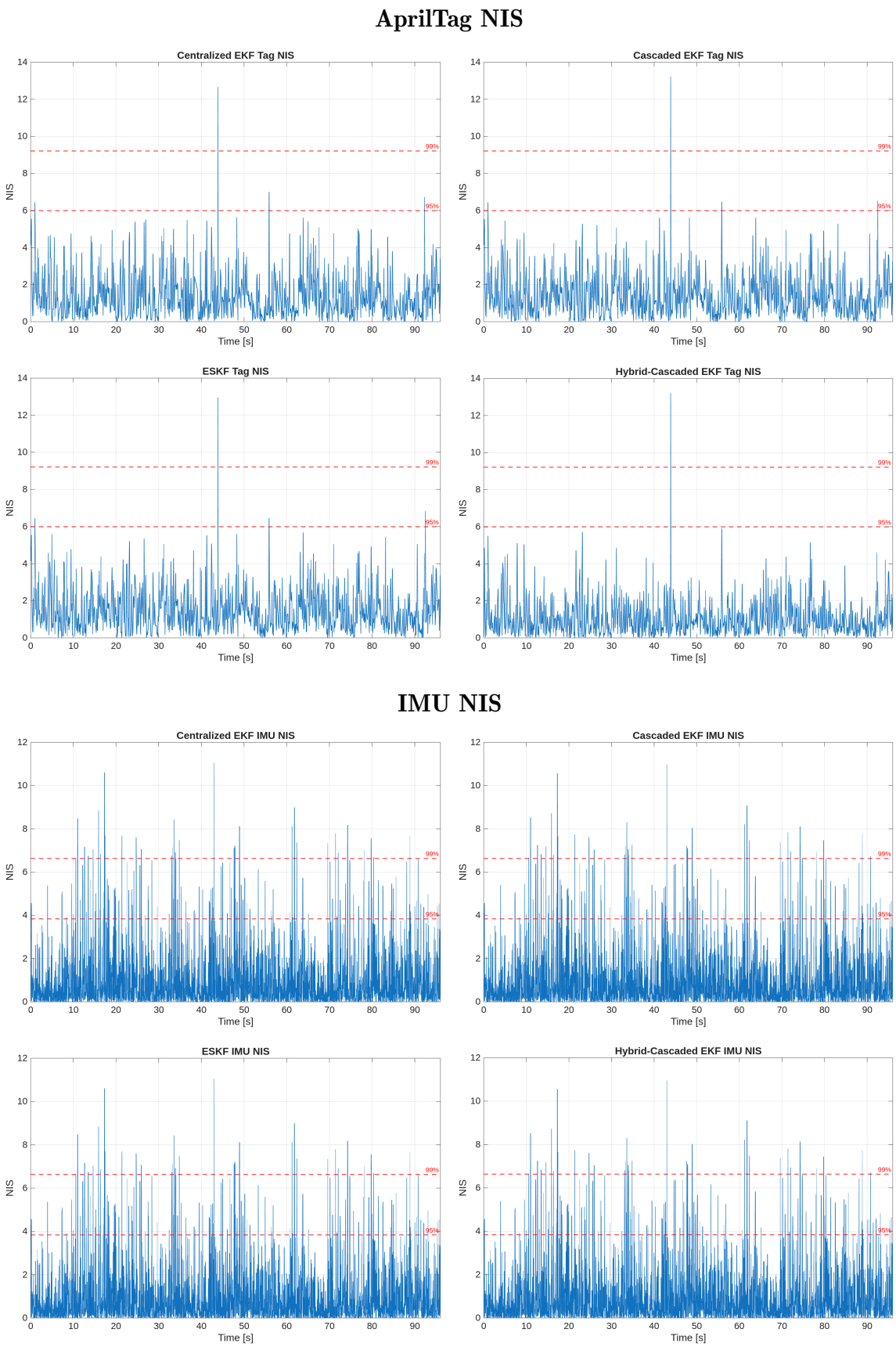


Figure A.2: AprilTag (top) and IMU (bottom) NIS for all filter architectures on the easy track.

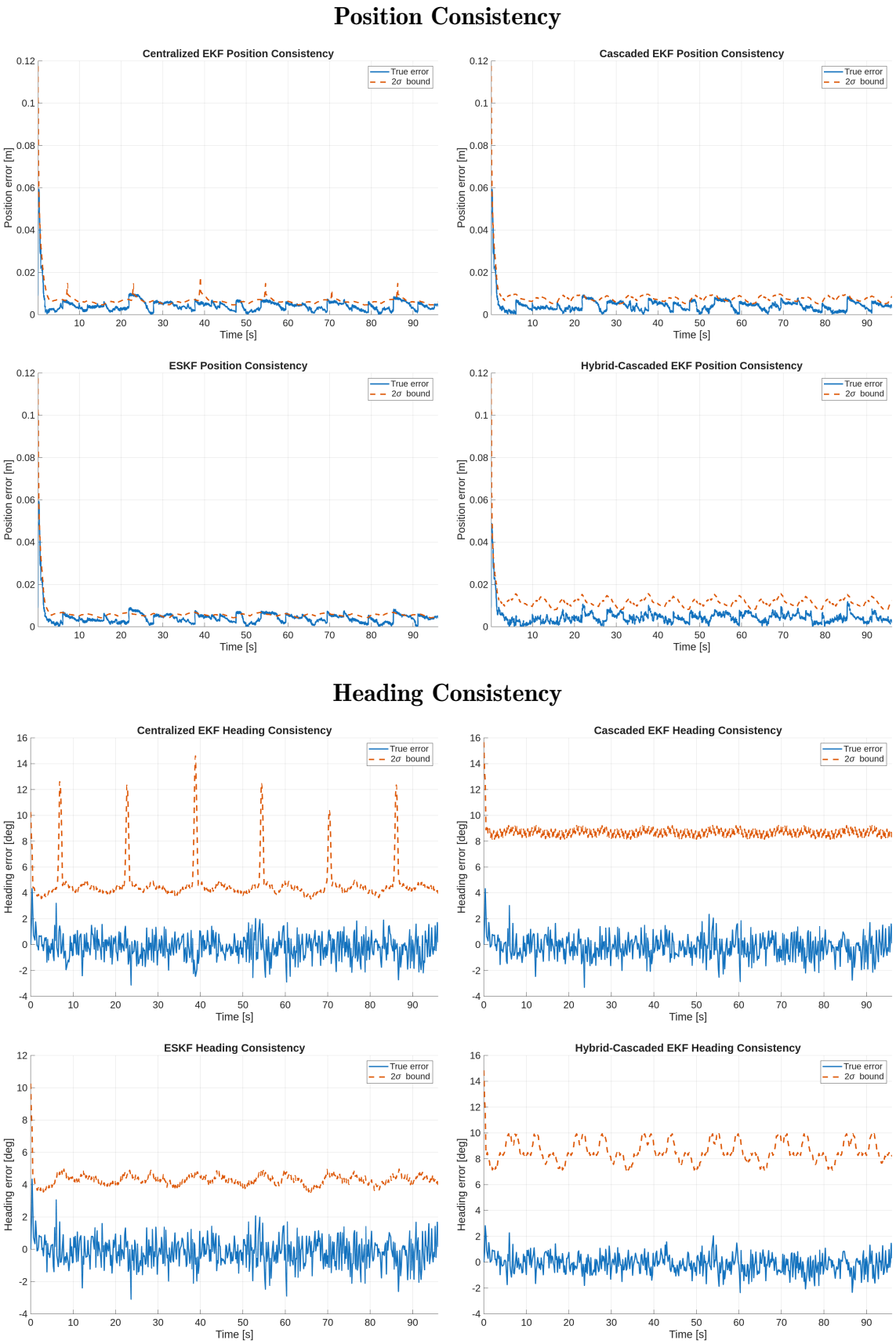


Figure A.3: Position (top) and heading (bottom) error consistency with respect to the predicted 2σ covariance envelope.

A.2 Simulation - Hard Track

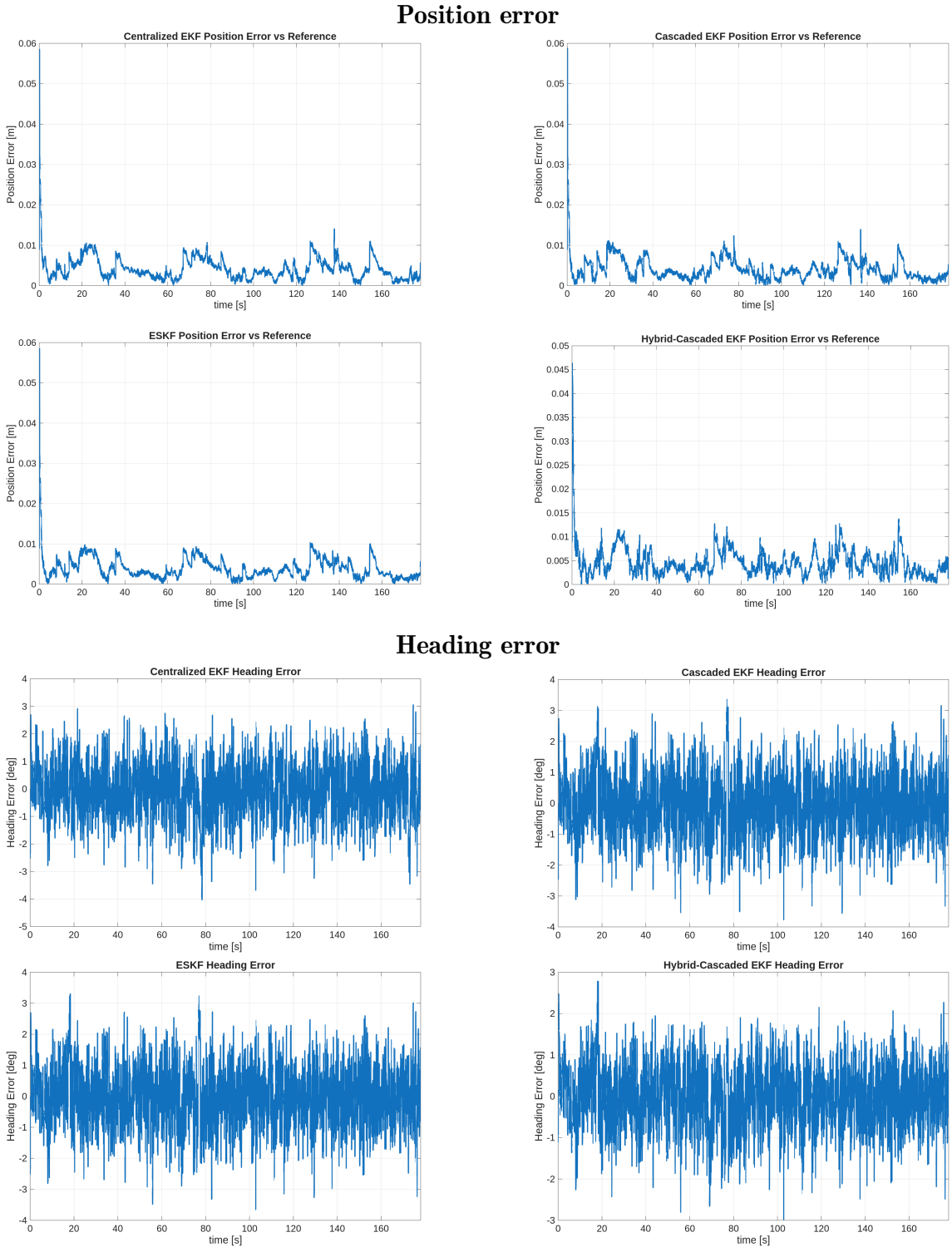


Figure A.4: Planar position errors (top) and heading errors (bottom) with respect to the reference trajectory for all filter architectures on the simulated hard track.

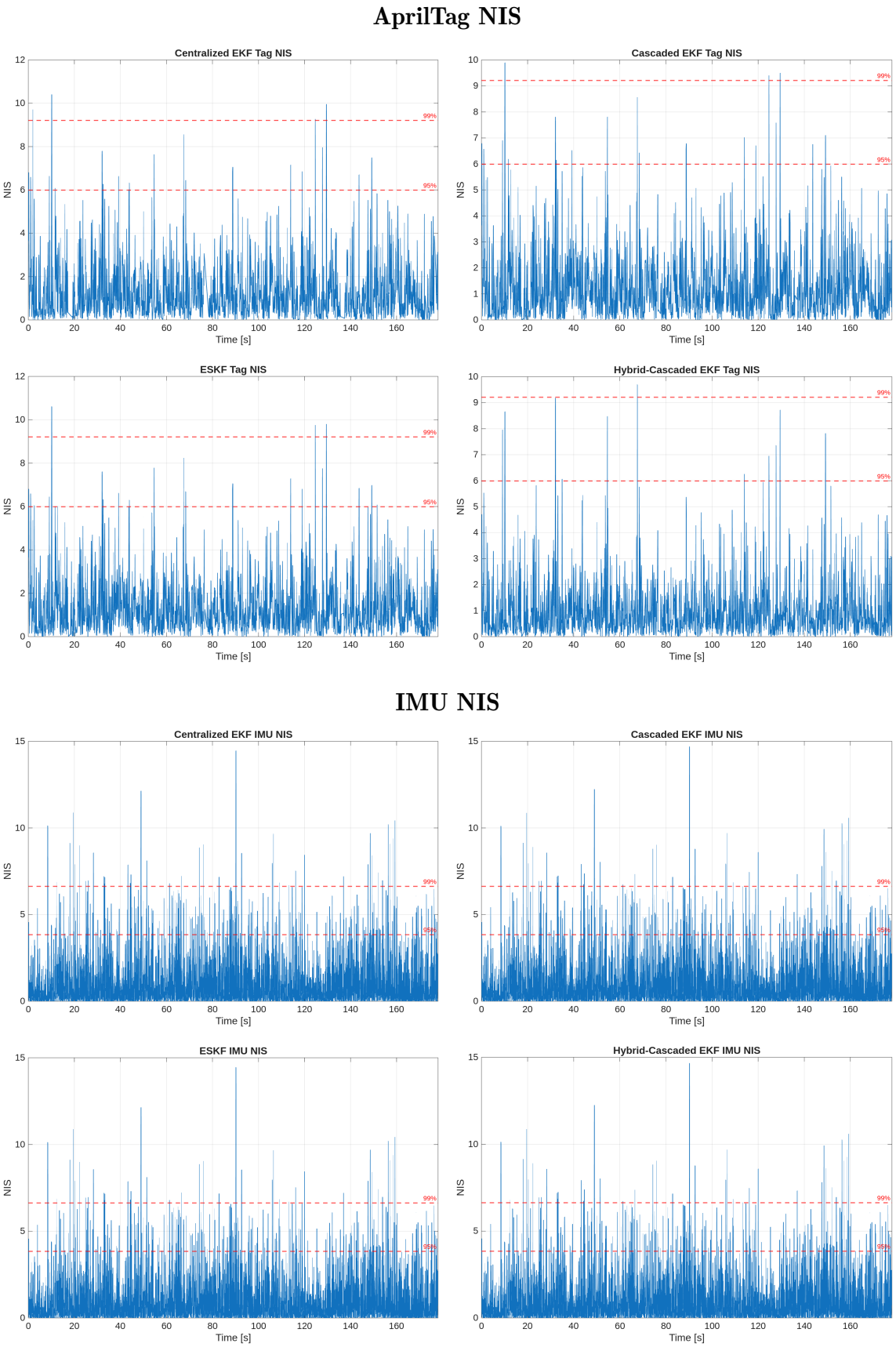


Figure A.5: AprilTag (top) and IMU (bottom) NIS for all filter architectures on the easy track.

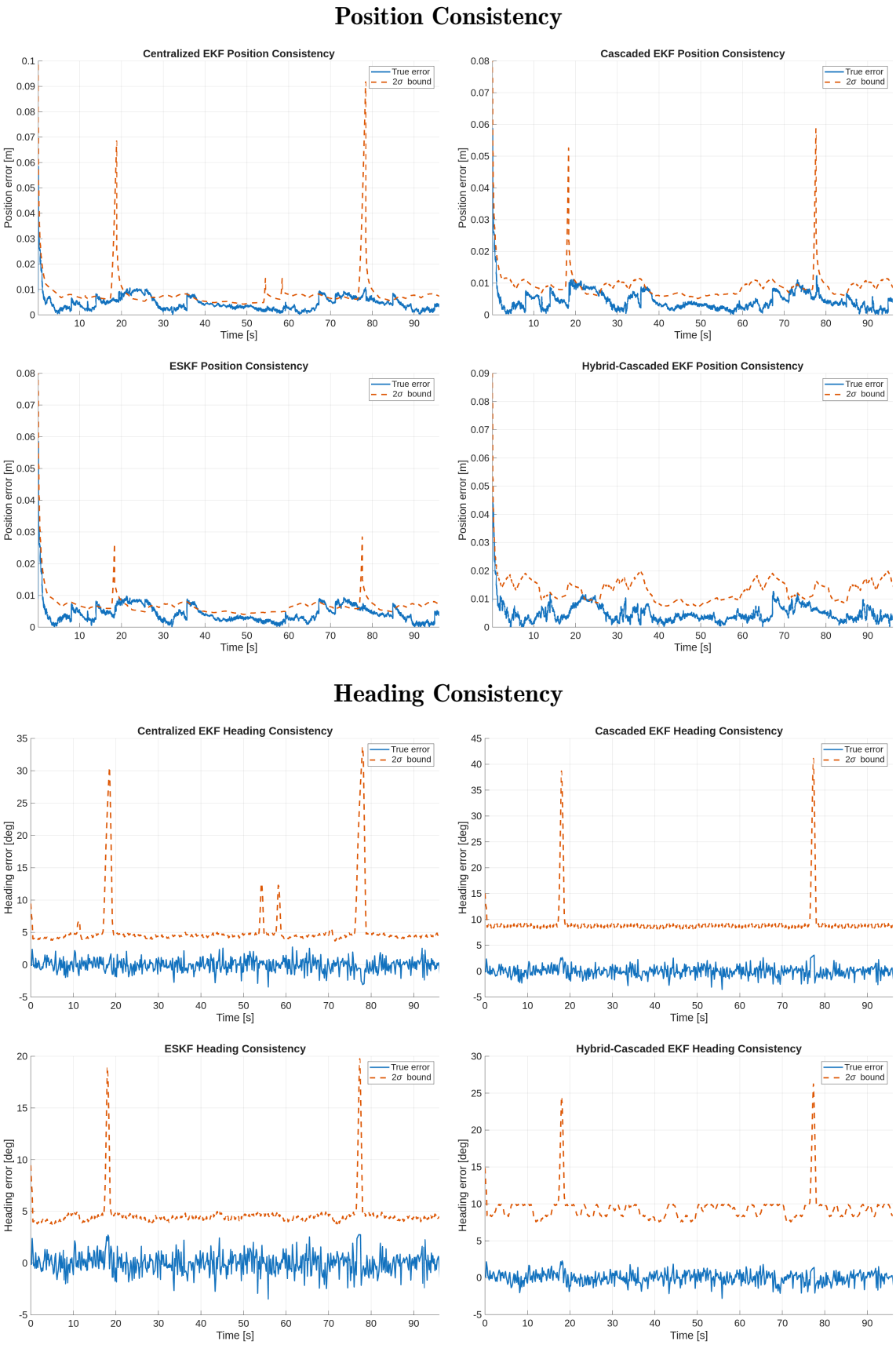


Figure A.6: Position (top) and heading (bottom) error consistency with respect to the predicted 2σ covariance envelope.

B

Detailed Experimental Results

This appendix reports detailed experimental results supporting the comparative analysis presented in Chapter 7. For completeness, the behavior of each localization architecture is illustrated individually using the experimental dataset recorded with the Duckiebot.

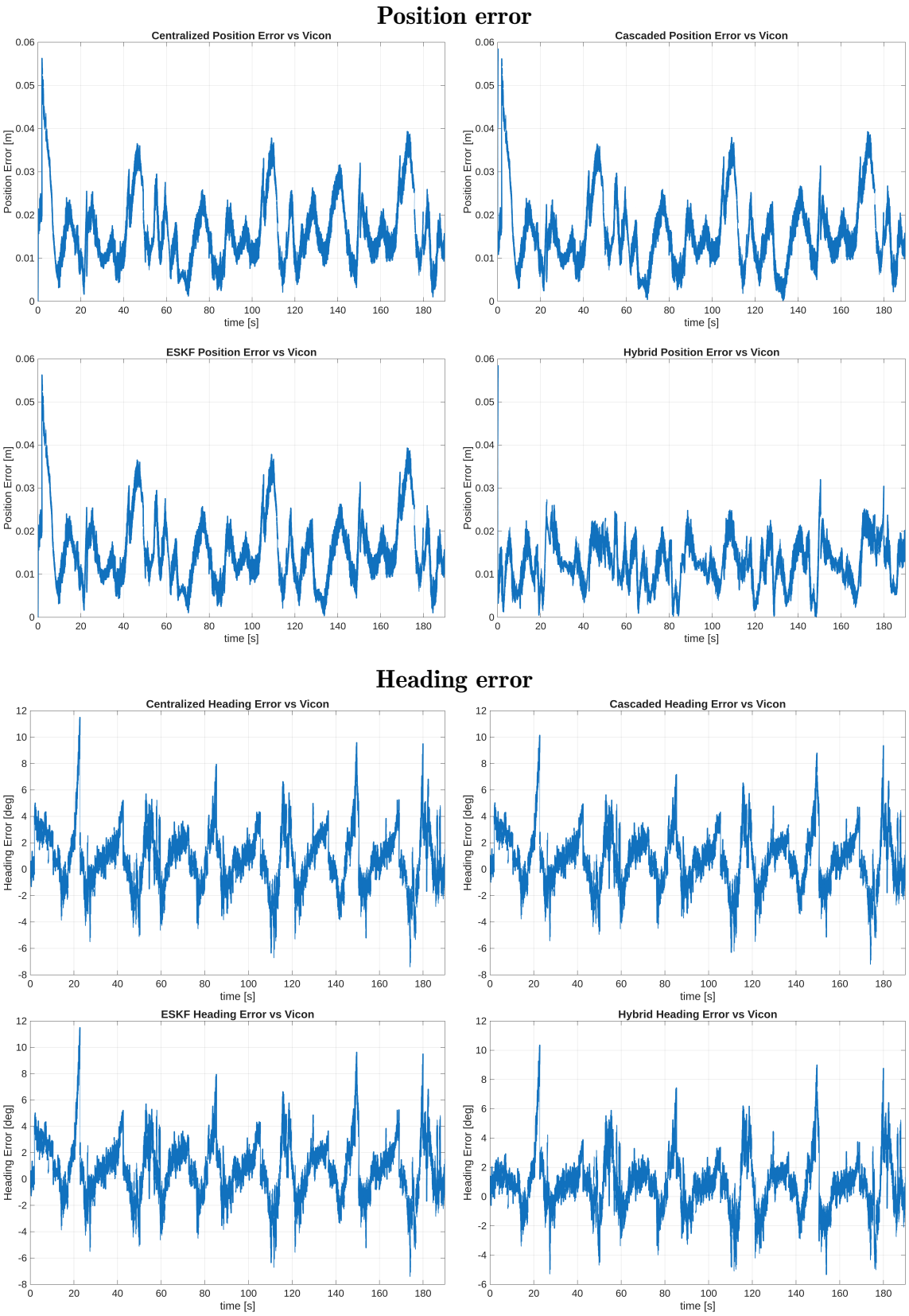


Figure B.1: Planar position errors (top) and heading errors (bottom) with respect to the reference trajectory for all filter architectures with the Duckiebot.

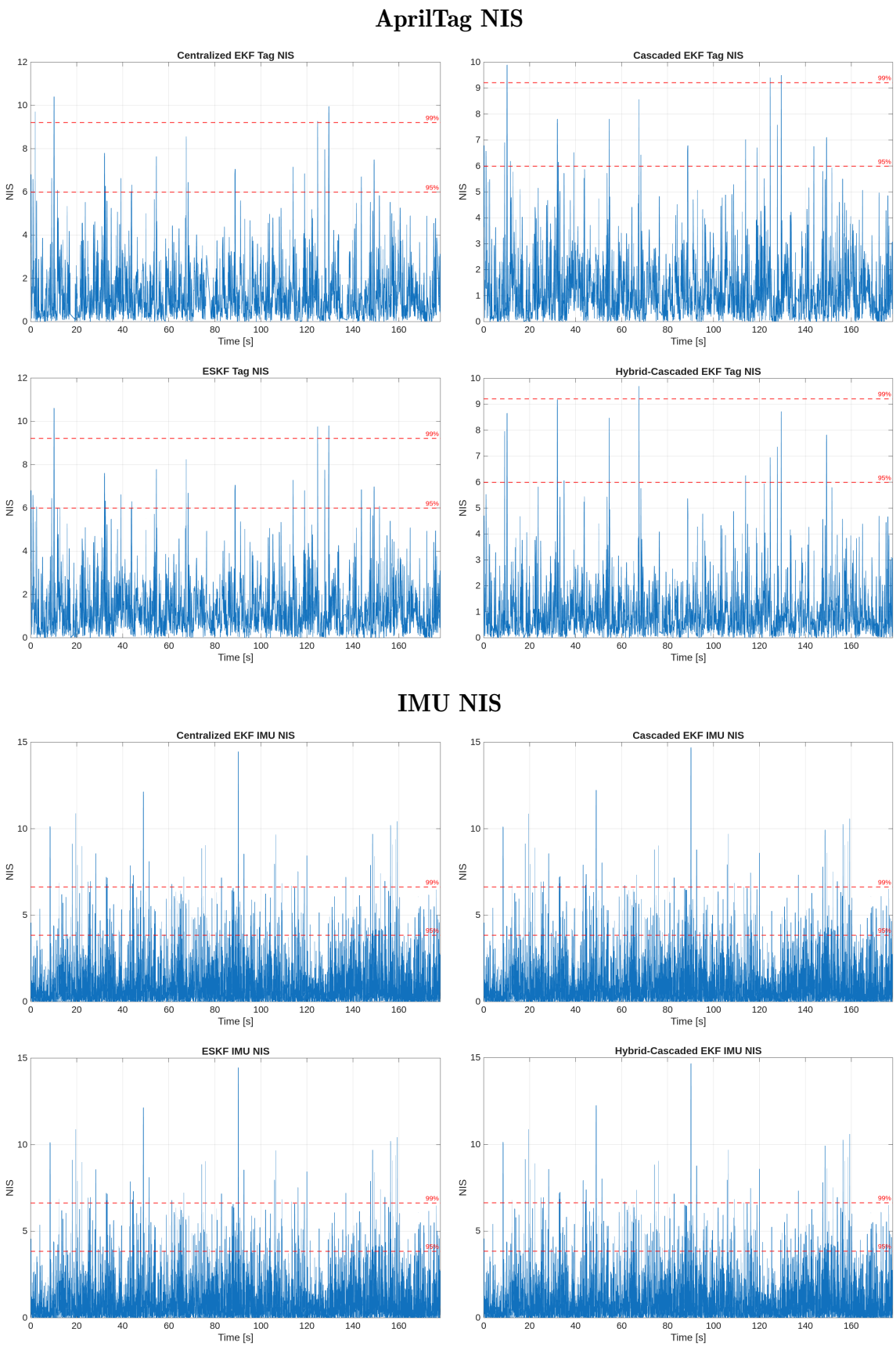


Figure B.2: AprilTag (top) and IMU (bottom) NIS for all filter architectures with the Duckiebot.

Figure B.3: Position (top) and heading (bottom) error consistency with respect to the predicted 2σ covariance envelope.

References

[1] Liam Paull et al. "Duckietown: an Open, Inexpensive and Flexible Platform for Autonomy Education and Research". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1497–1504.

[2] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[3] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, 2001.

[4] Neal A. Carlson and Michael P. Berarducci. "Federated Kalman Filter Simulation Results". In: *Navigation* 41.3 (1994), pp. 297–322.

[5] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. Springer, 2010.

[6] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[7] Edwin Olson. "AprilTag: A robust and flexible visual fiducial system". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 3400–3407.

[8] Simo Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[9] Rudolf E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.1 (1960), pp. 35–45.

[10] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*. Vol. 1. Academic Press, 1979.

[11] Joan Solà. "Quaternion kinematics for the error-state Kalman filter". In: *arXiv preprint arXiv:1711.02508* (2017).

[12] Anastasios I. Mourikis and Stergios I. Roumeliotis. “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation”. In: *IEEE International Conference on Robotics and Automation* (2007), pp. 3565–3572.

[13] Agostino Martinelli. “Observability Properties and Determination of the Absolute Scale in Vision-Based Structure from Motion”. In: *Foundations and Trends in Robotics* 1.3 (2012), pp. 139–209.

[14] X. Rong Li, Zhixin Zhao, and Vesselin P. Jilkov. “Performance Evaluation of Estimators with Applications to Tracking and Navigation”. In: *IEEE Transactions on Aerospace and Electronic Systems* 39.4 (2003), pp. 1377–1396.

[15] Sungckun Hong and Moon J. Lee. “Observability Analysis of Strapdown Inertial Navigation Systems”. In: *IEEE Transactions on Aerospace and Electronic Systems* 40.4 (2004), pp. 1328–1335.

[16] Xiaping Ma, Peimin Zhou, and Xiaoxing He. “Advances in Multi-Source Navigation Data Fusion Processing Methods”. In: *Mathematics* 13.9 (2025), p. 1485. DOI: 10.3390/math13091485.

[17] Duckietown Foundation. *Duckietown Documentation*. `https://docs.duckietown.com`. 2023.

[18] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes*. 4th ed. McGraw-Hill, 2002.