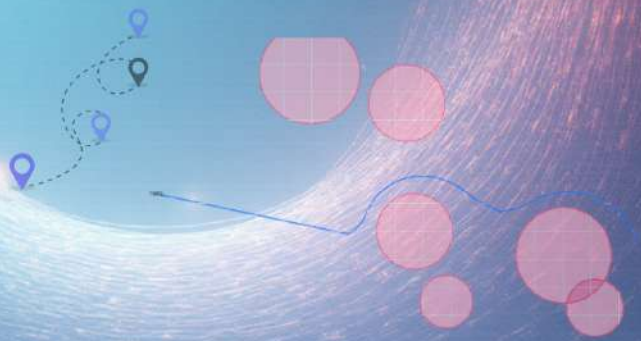
AI-Enhanced Navigation Systems

A Practitioner's Guide



By Dr. Barak Or, PhD

Edition 1.0 | January 2026

ArtificialGate

Contents

Edition and Scope

Practitioner Preview - Industrial Edition

This is an advanced technical booklet intended for engineers, system architects, and researchers working on real-world navigation systems.

It is not a tutorial, not a survey of machine learning techniques, and not a collection of end-to-end solutions. The material assumes prior familiarity with state estimation, probabilistic filtering, and multi-sensor fusion.

The focus of this booklet is architectural and conceptual: how learning-based models interact with classical estimation pipelines, where they provide genuine value, and where they introduce new risks.

All examples, discussions, and design principles are framed from an engineering perspective. The emphasis is on system behavior, consistency, failure modes, and long-term reliability - not on benchmark performance or isolated model accuracy.

This booklet is distributed as a limited conference edition and represents a subset of a larger ongoing book project. Some topics are intentionally introduced without full derivations or implementation details, in order to preserve focus on system-level understanding.

Preface

Artificial intelligence is increasingly embedded in navigation systems - yet navigation itself remains, at its core, a problem of state estimation under uncertainty.

Modern navigation systems operate under partial observability, imperfect sensing, and long-term drift. These constraints are structural, not incidental, and they do not disappear with the introduction of learning-based models.

The purpose of this booklet is to clarify the role of artificial intelligence within this reality.

Rather than treating AI as a replacement for classical navigation pipelines, this work adopts a principled engineering stance: learning-based methods are examined as structured components inside estimation systems, subject to the same requirements of consistency, bounded error, and failure awareness as any other system element.

This booklet is written for readers who already understand how navigation systems work. It is not intended for introductory study, nor for readers seeking end-to-end learning solutions. Instead, it targets practitioners and researchers who design, analyze, and deploy navigation systems that must operate reliably beyond laboratory

conditions.

Throughout the text, learning is discussed in terms of interfaces, assumptions, and system impact. The central question is not whether learning improves performance in isolated scenarios, but whether it preserves, or undermines, the long-term stability and reliability of the navigation system as a whole.

How to Read This Document

This document is structured to reflect how navigation systems are designed, analyzed, and validated in practice. It assumes that the reader is familiar with: state-space models, Kalman filtering and its variants, multi-sensor fusion, and basic probabilistic reasoning. These concepts are not reintroduced, but are used as a shared technical language. The early chapters establish a common estimation-centric foundation. Subsequent sections examine how learning-based components interact with this foundation, both constructively and destructively. Later chapters focus on architecture, failure awareness, and runtime behavior - aspects that often determine whether a system succeeds or fails in deployment. The chapters are intentionally connected. Concepts introduced in earlier sections are reused and expanded rather than repeated. Readers are encouraged to follow the document sequentially, particularly if they are less familiar with hybrid estimation–learning architectures. There are no "magic models" in this document. All learning-based components are discussed in terms of assumptions, limitations, and observable effects. Performance gains are considered secondary to stability, interpretability, and recoverability. If a design choice appears conservative, it is by intent. Navigation systems fail not when models are weak, but when systems are overconfident.

About the Author

Dr. Barak Or is an entrepreneur, researcher, and lecturer specializing in Artificial Intelligence, with a particular focus on Deep Learning and Machine Learning-based Inertial Sensing. He holds a PhD in machine learning and navigation systems, together with three additional degrees from the Technion - Israel Institute of Technology in aerospace engineering, economics and management. His work has led to multiple patents, peer-reviewed publications, and applied research projects in both academia and industry.

Since 2024, Dr. Or has served as the Academic Director of the AI and Deep Learning program at the Google and Reichman Tech School, where he designed and leads a comprehensive program for professional software developers. His teaching career began as a Teaching Assistant at the Technion and as a Physics Teacher at the Hebrew Reali School, later expanding into specialized AI training for engineers and researchers.

Dr. Or is the founder of ArtificialGate, an interactive online platform for AI education designed for corporate and technical audiences, and has also established several ventures in applied AI. These include MetaOr AI, a research and training lab; Alma Technologies, a deep-tech startup developing inertial sensor-based positioning solutions for GPS-denied environments; and AIME Systems, which

applies machine learning to healthcare and women's reproductive health. His work in these fields has secured competitive grants from the Israel Innovation Authority, the Ministry of Defense (MAFAT), and the Ministry of Economy and Industry, while also resulting in multiple patents in advanced navigation technologies.

Earlier in his career, Dr. Or was awarded the Gemunder Prize for research in satellite interception and placed second in the Technion's Security Innovations Competition.His academic background includes a Ph.D. in Machine Learning-based Inertial Navigation from the University of Haifa (2022), M.Sc. and B.Sc. in Aerospace Engineering from the Technion (2018, 2016) and a B.A. in Economics and Management (Cum Laude, 2016) from the Technion.

Through his combined work as researcher, educator, and entrepreneur, Dr. Or continues to advance the application of AI, while mentoring the next generation of engineers and data scientists.

Personal website: www.barakor.com.

PART I

Navigation as an Estimation System

Foundations, State, Drift, and Consistency

1 Navigation Is State Estimation

Navigation systems are often described in terms of sensors, perception pipelines, or algorithmic components. In operational systems, none of these elements define navigation.

Navigation is fundamentally the problem of estimating the evolving state of a physical system over time, under uncertainty, from incomplete, indirect, and noisy measurements.

This distinction is not semantic. It determines how navigation systems are designed, how failure manifests, and how learning-based components can be integrated without undermining reliability.

Sensors do not provide navigation. They provide constraints. Algorithms do not produce truth. They produce estimates conditioned on assumptions.

Artificial intelligence does not change this foundation. It may alter how specific relationships are modeled, but it does not change what is being estimated, nor the fact that estimation is recursive, uncertain, and time-coupled.

This chapter establishes an estimation-centric view of navigation. All subsequent discussion of learning-enhanced systems builds on this perspective.

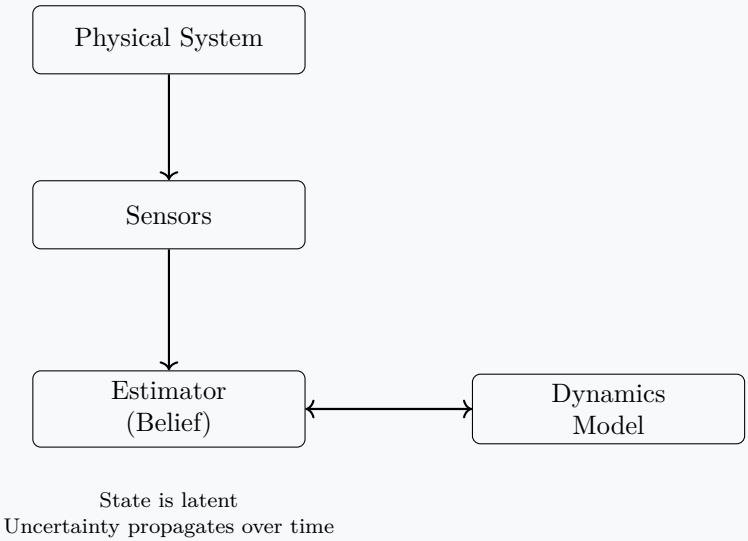


Figure 1.1: Navigation as recursive state estimation under uncertainty. Sensors provide constraints on a latent state. The estimator maintains belief through interaction with the system model.

1.1 Navigation Beyond Sensing

Sensors measure signals, not state. An inertial measurement unit outputs angular rates and specific forces. A camera produces pixel intensities. A GNSS receiver delivers pseudo-ranges and Doppler shifts.

None of these quantities correspond directly to navigation variables such as position, velocity, or orientation.

Navigation states are latent variables. They exist only within a mathematical framework that relates system dynamics, measurements, and uncertainty.

Formally, navigation systems operate within a state-space model:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \tag{1.1}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{1.2}$$

where $\mathbf{x}_k$ denotes the navigation state, $\mathbf{u}_k$ represents control or inertial inputs, $\mathbf{z}_k$ denotes measurements, and $\mathbf{w}_k$ and $\mathbf{v}_k$ are process and measurement noise terms.

The state $\mathbf{x}_k$ is never observed directly. It is inferred through recursive estimation, conditioned on assumptions embedded in $f(\cdot)$, $h(\cdot)$, and the associated noise models.

Treating navigation as a sensing problem implicitly assumes that improved measurements yield improved navigation. This assumption does not hold in the long term.

Navigation error is dominated not by instantaneous measurement quality, but by how errors propagate, correlate, and accumulate over time.

Time is the dominant dimension in navigation. Errors integrate. Biases compound. Modeling mismatches that are negligible over milliseconds become critical over minutes or hours.

This temporal coupling distinguishes navigation from static perception tasks. Any component introduced into a navigation system-learned or analytical-participates in this recursion and must be evaluated at the system level.

1.2 Canonical State Vectors

While navigation systems differ across platforms and applications, their state representations share a common structural pattern.

At the core lies the kinematic state: position $\mathbf{p}$, velocity $\mathbf{v}$, and orientation $\mathbf{R}$ (or an equivalent minimal representation).

A minimal navigation state may be written as

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} & \mathbf{v} & \boldsymbol{\theta} \end{bmatrix}^\top \tag{1.3}$$

where $\boldsymbol{\theta}$ denotes orientation parameters.

In operational systems, this state is insufficient.

Sensor imperfections introduce additional latent variables that directly influence how measurements are interpreted. As a result, practical navigation systems estimate augmented states of the form

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} & \mathbf{v} & \boldsymbol{\theta} & \mathbf{b}_{\text{imu}} & \mathbf{s} & \boldsymbol{\delta t} \end{bmatrix}^\top \tag{1.4}$$

where $\mathbf{b}_{\text{imu}}$ represents inertial sensor biases, $\mathbf{s}$ denotes scale or calibration parameters, and $\boldsymbol{\delta t}$ captures timing offsets between sensing modalities.

These additional states are not implementation artifacts. They represent the dominant sources of long-term navigation error.

Many of them evolve slowly and are only weakly observable. Their estimation depends on subtle interactions between system dynamics, motion, and measurement geometry. Ignoring these states does not eliminate their effect. It merely hides it.

Learning-based approaches often attempt to bypass explicit modeling of such states. While this may improve short-term accuracy, it removes explicit handles for uncertainty propagation and failure diagnosis.

Without explicit state representation, the system loses the ability to reason about what has degraded when performance deteriorates.

Explicit state modeling is therefore not an obstacle to learning

integration. It is a prerequisite for doing so safely.

1.3 Partial Observability as the Default

Full observability is the exception, not the rule, in navigation systems.

Most operational scenarios involve states that are partially observable, intermittently observable, or effectively unobservable over extended time intervals.

Observability is determined not only by sensor availability, but by motion, geometry, and environmental context. An inertial system without external reference cannot observe absolute position. A monocular vision system cannot observe scale without sufficient motion or additional assumptions. Bias states may become observable only under specific excitation conditions.

Formally, observability is a property of the system dynamics and measurement models. Learning cannot recover information that is not present in the measurements.

Motion-induced observability is therefore central to navigation. Certain states become observable only because the platform moves in ways that excite the system. Straight-line motion may render biases unobservable, while aggressive maneuvers reveal them.

This dependence on motion has two critical implications.

First, observability is context-dependent. A system that is well-conditioned in one operational regime may become ill-conditioned in another.

Second, learning-based models cannot violate observability constraints. They may exploit structure, but they cannot create information. Any system that appears to do so is either overfitting or

relying on unmodeled assumptions.

Ignoring partial observability leads to overconfident systems. Such systems may appear accurate under nominal conditions, yet fail abruptly when conditions change.

Reliable navigation architectures accept partial observability as the default. They are designed to detect it, manage it, and operate safely within it. This principle applies equally to classical estimators and to AI-enhanced systems. Navigation is not about eliminating uncertainty. It is about estimating state while knowing what cannot be known.

AI Implementation Notes

Learning should influence navigation systems only through local, bounded interfaces: residuals, uncertainty, or soft constraints. Learned state corrections without corresponding uncertainty updates break consistency by design. Treat learned outputs as correlated unless proven otherwise. When correlation cannot be modeled explicitly, their influence must be limited in magnitude or time.

2 Error-State Formulation in Practice

Operational navigation systems do not estimate the navigation state directly. Instead, they maintain a nominal trajectory and estimate deviations from it. This design choice is not a historical artifact. It reflects fundamental structural properties of navigation problems.

The error-state formulation separates nonlinear kinematics from uncertainty propagation. By doing so, it provides numerical stability, interpretability, and explicit control over how uncertainty evolves through time. These properties become essential once navigation systems operate at high rate, over long durations, or incorporate learning-based components.

This chapter explains why error-state filters dominate real systems, how error dynamics are modeled, and what a navigation filter actually estimates.

2.1 Full-State vs Error-State Filters

In a full-state formulation, the filter estimates the complete navigation state directly. The state at time k may be written as

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_k & \mathbf{v}_k & \boldsymbol{\theta}_k & \mathbf{b}_k \end{bmatrix}^\top \tag{2.1}$$

where $\mathbf{p}_k$ denotes position, $\mathbf{v}_k$ velocity, $\boldsymbol{\theta}_k$ orientation parameters, and $\mathbf{b}_k$ sensor-related states.

The full-state filter propagates both the mean and covariance of $\mathbf{x}_k$ through nonlinear dynamics. While conceptually straightforward, this approach encounters practical limitations in real navigation systems. Linearization errors accumulate, covariance propagation becomes fragile, and numerical conditioning degrades over long integration horizons, particularly in high-rate inertial pipelines.

The error-state formulation addresses these limitations by decomposing the problem into two coupled components: a nominal state and a small error state.

The nominal state $\bar{\mathbf{x}}_k$ is propagated deterministically using the full nonlinear dynamics. The filter then estimates the deviation

$$\delta\mathbf{x}_k = \begin{bmatrix} \delta\mathbf{p}_k & \delta\mathbf{v}_k & \delta\boldsymbol{\theta}_k & \delta\mathbf{b}_k \end{bmatrix}^\top \tag{2.2}$$

The true state is represented implicitly as

$$\mathbf{x}_k = \bar{\mathbf{x}}_k \oplus \delta\mathbf{x}_k \tag{2.3}$$

where $\oplus$ denotes composition on the appropriate state manifold.

This separation yields several advantages. First, the error state remains small by construction, which justifies linearization. Second, uncertainty propagation operates on deviations rather than absolute quantities. Third, numerical stability is preserved even under aggressive maneuvers and long-duration operation.

For these reasons, the error-state Kalman filter (ESKF) has become the dominant architecture in industrial navigation systems. Its prevalence is not theoretical. It is driven by operational robustness, debuggability, and predictable failure behavior.

2.2 Error Dynamics

The power of the error-state formulation lies in how it models the evolution of small deviations. Error dynamics describe how imperfections in sensing, modeling, and calibration translate into uncertainty growth over time.

Using continuous-time notation, the error-state dynamics may be expressed as

$$\dot{\delta\mathbf{x}}(t) = \mathbf{F}(t)\,\delta\mathbf{x}(t) + \mathbf{w}(t) \tag{2.4}$$

where $\mathbf{F}(t)$ is the error-state Jacobian and $\mathbf{w}(t)$ represents process noise.

Bias terms are among the most influential components of $\delta\mathbf{x}$. In inertial navigation, accelerometer and gyroscope biases are commonly modeled as random walks:

$$\dot{\mathbf{b}}(t) = \mathbf{w}_b(t) \tag{2.5}$$

This model reflects physical reality. Biases evolve slowly and unpredictably, but they do not remain constant. Even small bias errors integrate into significant position and velocity errors over time.

Scale factor and misalignment errors introduce structured distortion. Unlike noise, these errors systematically warp measurements, coupling sensor outputs directly to navigation error. Depending on the application, such effects are modeled as constants or slow random walks.

The key point is that these processes are not incidental. They dominate long-term error growth. Treating them as white noise is mathematically convenient, but operationally dangerous.

Learning-based models are sometimes proposed as replacements for explicit bias or calibration states. While learning may approximate certain error behaviors, removing explicit error states eliminates the estimator's ability to propagate uncertainty coherently. The system may appear accurate while becoming fundamentally inconsistent.

A disciplined integration strategy treats learning as an augmentation to error dynamics, never as a substitute for explicit error-state representation.

2.3 What the Filter Actually Estimates

A navigation filter does not estimate truth. It estimates a conditional probability distribution defined by its models and assumptions.

At each time step, the filter maintains:

- an estimated mean error state $\mathbb{E}[\delta \mathbf{x}_k]$,
- and an associated covariance matrix $\mathbf{P}_k$.

The mean represents the most likely deviation from the nominal trajectory under the assumed model. The covariance represents uncertainty under the same assumptions.

Both quantities are conditional. If the assumptions embedded in $\mathbf{F}$, the noise models, or the measurement models are violated, the covariance ceases to be meaningful - even if the mean appears accurate.

This distinction becomes critical when integrating AI components. Learning-based outputs often improve mean accuracy while silently invalidating uncertainty assumptions. When such outputs

are injected without corresponding uncertainty handling, the filter becomes overconfident.

Consistency, not accuracy, is therefore the defining property of a reliable navigation estimator. A consistent filter may be conservative, but it knows when it is uncertain. An inconsistent filter may look accurate while being fundamentally wrong.

Understanding what the filter estimates - and what it does not - is essential for system-level design. It determines whether errors are detectable, recoverable, and bounded.

The error-state formulation makes these properties explicit. It exposes where assumptions reside, how they propagate, and how they fail. This transparency is precisely what enables safe integration of learning-based components into navigation systems.

AI Implementation Notes

Learning should augment error dynamics, not replace them. Explicit error states must remain the authority for uncertainty propagation. Any learned correction that modifies the state without modifying covariance introduces inconsistency by construction.

3 Drift Is Structural, Not Accidental

Drift is often treated as an implementation flaw, a calibration error, or a limitation of specific sensors. In navigation systems, drift is none of these.

Drift is a structural consequence of estimating state under partial observability, imperfect models, and finite information. It arises even when sensors operate nominally and algorithms are implemented correctly.

Understanding drift as a structural phenomenon is essential. It reframes the role of sensors, estimators, and learning-based components. Most importantly, it clarifies what navigation systems can and cannot guarantee over long time horizons.

This chapter examines the dominant drift mechanisms across inertial, vision-based, and GNSS-based navigation, and explains why drift can never be eliminated - only bounded, corrected, or detected.

3.1 Inertial Drift Mechanisms

Inertial navigation systems estimate motion by integrating angular rates and specific forces over time. This integration process is inherently unstable with respect to small errors.

Let the inertial error state include position, velocity, attitude,

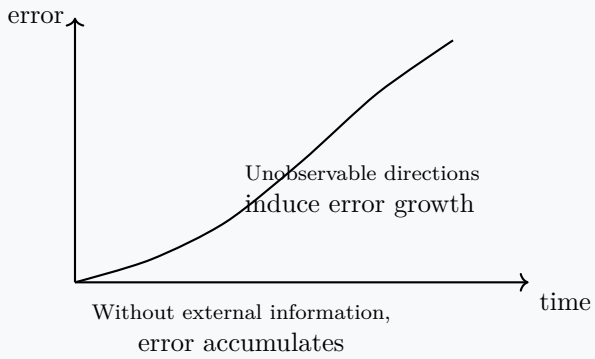


Figure 3.1: Structural drift in navigation systems. When parts of the state are unobservable, estimation error grows over time even under nominal operation.

and bias terms:

$$\delta\mathbf{x}(t) = \begin{bmatrix} \delta\mathbf{p}(t) & \delta\mathbf{v}(t) & \delta\boldsymbol{\theta}(t) & \delta\mathbf{b}(t) \end{bmatrix}^\top \tag{3.1}$$

Bias errors enter the dynamics through integration. Even when bias magnitudes are small, their effect grows unbounded with time. This behavior is not sensor-specific; it is a property of integration itself.

A simplified continuous-time relationship illustrates this effect:

$$\dot{\delta\mathbf{v}}(t) = \delta\mathbf{b}_a(t) \tag{3.2}$$

$$\dot{\delta\mathbf{p}}(t) = \delta\mathbf{v}(t) \tag{3.3}$$

A constant accelerometer bias produces linearly growing velocity error and quadratically growing position error. No estimator can prevent this growth without external information.

Calibration reduces initial bias uncertainty, but it does not elim-

inate bias evolution. Temperature effects, aging, and environmental conditions introduce slow variations that cannot be perfectly modeled.

Learning-based bias correction is sometimes proposed as a mitigation. Learning can improve short-term bias estimation by exploiting context such as temperature, vibration, or operating regime. However, learning does not change the structural instability of inertial integration.

AI can assist inertial navigation by: improving bias modeling, adapting uncertainty levels, or detecting abnormal drift behavior. It cannot remove the need for external correction or eliminate long-term divergence on its own.

Inertial drift is therefore unavoidable. The role of the estimator, classical or AI-enhanced, is not to eliminate drift, but to manage it through correction, detection, and bounded operation.

3.2 Vision Drift Mechanisms

Vision-based navigation systems infer motion and structure from image sequences. Unlike inertial systems, vision does not integrate signals directly. Nevertheless, vision-based navigation also exhibits drift. Vision drift arises from three structural sources.

First, scale ambiguity. In monocular systems, absolute scale is not observable without sufficient motion or additional assumptions. Errors in scale estimation propagate into position and velocity estimates over time.

Second, feature degradation. Changes in illumination, motion blur, texture scarcity, and viewpoint reduce measurement quality. As features degrade, measurement noise becomes correlated and

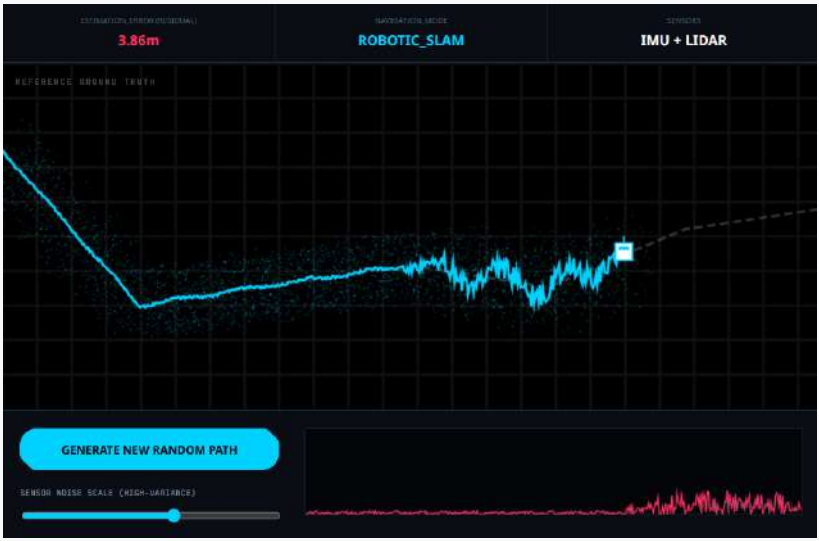


Figure 3.2: **Estimation Residuals Under Non-Gaussian Noise.** Residual behavior under increased sensor noise and stress. Classical filters remain stable only when uncertainty growth reflects true error statistics. Learned components must preserve this relationship to avoid silent divergence.

increasingly non-Gaussian.

Third, model inconsistency. Linearization errors, delayed updates, and correlation between past and current estimates introduce slow divergence.

These effects are cumulative. Individual visual measurements may remain locally accurate, yet long-term trajectories drift as small inconsistencies accumulate.

Loop closure and mapping mitigate drift by introducing global constraints. However, these mechanisms rely on environmental revisitability and reliable data association. In open or changing environments, such assumptions cannot be guaranteed.

Learning-based visual front-ends often improve local robustness. They enhance feature extraction, matching, and depth inference, and can significantly reduce short-term error. However, they do not eliminate the structural causes of drift.

AI improves vision-based navigation when it: enhances measurement quality, adapts uncertainty to context, or detects degradation in visual conditions. When learning introduces overconfidence or suppresses uncertainty growth, it accelerates silent divergence.

Vision drift is therefore not a failure of perception quality. It is a consequence of estimating motion and structure under partial observability with long-term correlation.

3.3 GNSS Drift Is Different

GNSS-based navigation is often assumed to be drift-free. This assumption is misleading.

GNSS errors are typically bounded, but they are not stationary. Multipath, occlusion, atmospheric effects, and receiver-specific biases introduce slowly varying errors that depend on environment and geometry.

Unlike inertial drift, GNSS errors may appear as: step changes, slow biases, or geometry-dependent distortions.

A critical distinction is that GNSS drift is externally driven. It depends on factors outside the estimator's control and may change abruptly.

Treating GNSS measurements as ground truth masks these effects. When GNSS biases are unmodeled, they propagate into fused navigation solutions and contaminate other state estimates.

Learning-based GNSS correction models are increasingly ex-

plored. They can capture environment-specific error patterns and improve robustness in challenging conditions. However, learned corrections must be accompanied by calibrated uncertainty estimates.

AI improves GNSS-based navigation when it: models context-dependent error behavior, detects degradation, or adapts trust dynamically. A learned correction without uncertainty is more dangerous than no correction at all.

GNSS drift is therefore different in character, but not in consequence. It must be detected, bounded, and managed - not assumed away.

3.4 Why Drift Never Disappears

Drift cannot be eliminated because it arises from missing information. Whenever parts of the state are unobservable, uncertainty must grow.

Formally, if a system lacks sufficient measurements to constrain all state components, the estimation error covariance grows along unobservable directions. Any estimator that does not reflect this growth is inconsistent by construction.

Learning does not change this fact. No model can infer information that is not present in the data. When learning appears to remove drift entirely, it is exploiting hidden assumptions or overfitting to restricted conditions.

Reliable navigation systems therefore adopt a different objective. They do not attempt to eliminate drift. They ensure that drift is bounded when possible, corrected when information becomes available, and detected before estimates lose validity.

This perspective has direct implications for how AI should be used in navigation systems. AI is most effective when it: improves er-

ror modeling, adapts uncertainty to context, or accelerates detection of degraded conditions.

AI is ineffective, and dangerous, when it suppresses uncertainty growth or replaces explicit estimation structure.

Drift is not an accident to be fixed. It is a structural property to be respected.

Only systems that accept this reality can integrate learning-based components without sacrificing long-term reliability.

Engineering Principle: Drift

Drift is a structural consequence of partial observability and finite information. It cannot be eliminated by better sensors, better models, or learning alone. AI improves navigation systems when it helps expose, bound, or detect drift - not when it hides it. Any component that suppresses uncertainty growth without adding information introduces overconfidence by design.

4 Accuracy, Consistency, and Boundedness

Navigation performance is often summarized by a single number, such as position error, velocity error, or orientation error. While convenient, this practice hides the properties that actually determine whether a navigation system is usable, safe, and reliable over time.

Accuracy alone is insufficient. Navigation systems rarely fail because they are inaccurate at a given instant. They fail because they become wrong without knowing it.

This chapter reframes navigation performance through three system-level properties: accuracy, consistency, and boundedness. Each captures a fundamentally different aspect of estimator behavior. Understanding their distinction is essential for designing navigation systems that remain reliable, particularly when learning-based components are introduced.

4.1 Accuracy as a Snapshot Metric

Accuracy measures deviation from truth at a specific time. Given the true state $\mathbf{x}_k^{\text{true}}$ and an estimated state $\hat{\mathbf{x}}_k$, accuracy can be expressed as

$$\mathbf{e}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k^{\text{true}}, \tag{4.1}$$

where $\mathbf{e}_k$ denotes the estimation error vector.

Accuracy is inherently a snapshot metric. It answers a narrow question: how close is the estimate to truth at this moment?

In short-duration experiments or tightly controlled benchmarks, accuracy is often sufficient. In long-duration navigation, however, it is not. Two systems with identical instantaneous accuracy may exhibit radically different long-term behavior. One may recover gracefully from degradation, while the other may drift silently until failure becomes irreversible.

Learning-based components tend to optimize accuracy directly. Neural models are typically trained to minimize error norms over representative datasets, which often yields impressive short-term performance under nominal conditions. However, optimizing accuracy in isolation ignores temporal coupling. A learned correction that improves accuracy at time k may invalidate uncertainty assumptions at time $k+1$. From a system perspective, such improvements can be actively harmful.

Accuracy alone does not reveal whether the system knows when it is wrong. For navigation systems, this distinction is decisive.

4.2 Consistency as a System Property

Consistency describes the relationship between estimated uncertainty and actual error. A navigation system is consistent if its reported uncertainty correctly reflects the statistics of its true estimation error.

Let $\mathbf{P}_k$ denote the estimated error covariance. Consistency requires that the error $\mathbf{e}_k$ be statistically compatible with $\mathbf{P}_k$. This relationship is commonly evaluated using normalized error measures.

For example, the normalized estimation error squared (NEES) is given by

$$\text{NEES}_k = \mathbf{e}_k \mathbf{P}_k^{-1} \mathbf{e}_k^\top. \tag{4.2}$$

A consistent estimator produces NEES values that remain within expected statistical bounds over time. An inconsistent estimator may appear accurate while being fundamentally overconfident.

Consistency is not a property of a single model or component. It emerges from the interaction between system dynamics, measurements, noise assumptions, and update logic. Classical navigation systems enforce consistency through explicit modeling and conservative assumptions. Their strength lies in predictability, while their weakness lies in rigidity, including fixed noise models, manual tuning, and limited adaptability to context.

Learning-based components can assist consistency if they are integrated correctly. AI can model context-dependent uncertainty, sensor quality, and residual behavior that classical models struggle to capture. For example, a learned trust model can adaptively scale measurement covariance based on environment or operating conditions. When learning modulates uncertainty rather than bypassing estimation, consistency can improve rather than degrade.

The central rule is simple: AI should adjust uncertainty, not replace it.

4.3 Bounded Error vs Minimal Error

Operational navigation systems prioritize bounded error over minimal error. The objective is not to achieve the smallest possible error under ideal conditions, but to remain within known and predictable limits under non-ideal ones.

Boundedness refers to the ability of a system to guarantee that error remains within acceptable bounds, given its assumptions. Minimal error focuses on reducing instantaneous deviation, often without regard to long-term behavior. These objectives are not equivalent.

Learning-based systems often excel at minimizing error on their training distribution. However, they frequently lack mechanisms to enforce boundedness under distribution shift. Classical estimators enforce boundedness through conservative uncertainty growth and explicit observability constraints. Their limitation is that they may sacrifice accuracy even when conditions are favorable.

Hybrid systems can combine these strengths. Learning can reduce error where data supports it, while the estimator maintains authority over uncertainty growth and bounding behavior.

Table 4.1 summarizes these trade-offs.

Table 4.1: Accuracy, consistency, and boundedness in classical and AI-enhanced navigation systems.

Property	**Classical Estimation**	**AI-Enhanced Estimation**
Accuracy	Conservative, stable	High under nominal conditions
Consistency	Explicitly enforced	At risk if uncertainty is bypassed
Boundedness	Guaranteed by design	Depends on integration discipline
Adaptability	Limited	High, context-dependent
Failure visibility	High	Low if confidence is uncalibrated

The pattern is clear: AI improves accuracy and adaptability, but boundedness must remain under estimator control.

4.4 Silent Inconsistency

The most dangerous failure mode in navigation systems is silent inconsistency. In this mode, the system continues to produce confident estimates while its internal assumptions are violated.

Silent inconsistency occurs when error grows, uncertainty does not, and no alarm is triggered. Learning-based components are particularly susceptible to this failure mode. Neural models often generalize poorly outside training conditions while maintaining confident outputs. When such outputs are injected into a navigation filter without corresponding uncertainty handling, the system becomes blind to its own degradation.

Detecting silent inconsistency requires internal signals. Innovation statistics, residual monitoring, and consistency tests provide early warning when assumptions no longer hold. AI can assist detection rather than undermine it. Learning-based monitors can identify anomalous residual patterns or shifts in innovation statistics. Crucially, these models must operate on estimator signals, not replace them.

The objective is not to eliminate inconsistency entirely, but to detect it before it becomes operationally significant.

Consistency therefore forms the bridge between accuracy and boundedness. Without consistency, accuracy is misleading and boundedness is illusory. Navigation systems that integrate AI successfully treat uncertainty as a first-class signal. They allow learning to improve performance while preserving the estimator's authority over trust, bounds, and failure awareness.

Engineering Principle: Performance

Accuracy measures closeness to truth. Consistency determines whether uncertainty can be trusted. Boundedness ensures that failure remains controllable. AI improves navigation systems when it strengthens consistency or accelerates detection of inconsistency, not when it merely optimizes pointwise accuracy.

References - Part I

1. R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1), 1960.
2. Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley, 2001.
3. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.
4. P. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, 2013.
5. S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

PART II

Where Learning Fits (and Where It Doesn't)

Safe Roles, Limits, and Integration Boundaries

5 Learning in a Structured Estimation World

The increasing availability of data and computational power has made learning-based models attractive for navigation systems. In isolation, this trend is unsurprising: learning excels at capturing complex patterns in large datasets. Navigation systems, however, are not generic prediction problems. They are structured estimation systems governed by physics, geometry, and observability constraints.

The central question is therefore not whether learning can improve navigation performance. The question is where learning can be introduced without violating the assumptions that make navigation systems reliable over long time horizons.

This chapter establishes a clear boundary for the role of learning in navigation. It examines what learning can represent better than classical models, what it fundamentally cannot recover, and how observability defines a hard limit on the benefits learning can provide.

5.1 What Learning Can Represent Better Than Physics

Classical navigation models rely on explicit mathematical descriptions of dynamics, sensors, and noise processes. These models are necessarily simplified. They capture dominant physical effects, but they struggle with complex, context-dependent phenomena that are difficult to parameterize analytically.

Learning-based models excel in precisely these regimes. They are particularly effective at representing relationships that are nonlinear, high-dimensional, and strongly dependent on operating context. In navigation systems, such relationships rarely appear in the nominal dynamics. Instead, they manifest primarily in error processes.

Examples include context-dependent sensor noise, environment-specific measurement degradation, and subtle correlations between residuals and operating conditions. These effects are real, repeatable, and operationally significant, yet they are poorly captured by fixed analytical models.

Classical estimation typically assumes measurement noise of the form

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k), \tag{5.1}$$

with the covariance $\mathbf{R}_k$ treated as constant or adjusted through heuristic tuning.

In practice, measurement uncertainty depends on latent variables such as motion regime, environment, sensor state, and platform dynamics. Learning-based models can approximate this dependence by mapping contextual features to uncertainty estimates,

$$\mathbf{R}_k = g_\theta(\mathbf{c}_k), \tag{5.2}$$

where $\mathbf{c}_k$ denotes contextual information and $g_\theta(\cdot)$ is a learned mapping.

The critical point is not the use of learning itself, but where it acts. In this formulation, learning operates on uncertainty rather than on state. The estimator retains full authority over state propagation and fusion, while learning refines how much trust is assigned to incoming information.

Learning is also effective at modeling residual structure. Given an innovation

$$\boldsymbol{\nu}_k = \mathbf{z}_k - h(\bar{\mathbf{x}}_k), \tag{5.3}$$

learning can capture systematic patterns in $\boldsymbol{\nu}_k$ that classical models treat as noise. These patterns often reflect environmental effects, sensor degradation, or unmodeled dynamics.

Such uses of learning align naturally with error-state formulations. Learning augments quantities the estimator already computes, rather than replacing the estimation process itself. When used this way, learning improves performance without undermining estimator structure.

5.2 What Learning Fundamentally Cannot Recover

Despite its expressive power, learning is constrained by information content. No model-learned or analytical-can recover information that is not present in the measurements. Learning-based navigation systems often appear to violate this principle in controlled experiments. In reality, they exploit hidden assumptions embedded in the data, such as fixed environments, restricted motion regimes, or correlations that do not generalize beyond the training distribution. When these assumptions are violated, performance degrades abruptly.

From an estimation perspective, learning cannot create observability. It cannot remove integration drift without external reference, and it cannot infer absolute quantities from relative measurements alone.

For example, no amount of learning can infer absolute position from inertial data in the absence of external information. A learned model may appear to do so by memorizing motion patterns or exploiting dataset-specific structure, but such behavior does not generalize and fails under distribution shift. Similarly, learning cannot replace uncertainty propagation. A neural network may output a state estimate, but without an associated, calibrated uncertainty representation, that estimate cannot be integrated safely

into a recursive estimator. State estimates without uncertainty are assertions, not beliefs.

These limitations are structural, not technological. They follow directly from the mathematics of estimation and do not disappear with larger models, more data, or improved training techniques. Recognizing what learning cannot do is as important as exploiting what it can. Systems that ignore these limits tend to exhibit silent inconsistency, overconfidence, and delayed failure detection-failure modes that are particularly dangerous in navigation.

5.3 The Observability Wall

Observability defines the boundary between what can and cannot be estimated. It is a property of system dynamics and measurement geometry, not of the model class used to process data.

For a system described by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k), \tag{5.4}$$

$$\mathbf{z}_k = h(\mathbf{x}_k), \tag{5.5}$$

certain directions in state space may be unobservable. Along these directions, uncertainty must grow over time.

Learning does not alter this fact. It may reduce apparent error temporarily by exploiting correlations, but it cannot remove the underlying lack of information. When learning appears to overcome observability limits, it is relying on assumptions that are not guaranteed to hold in operation.

The observability wall is therefore the point at which learning ceases to provide reliable benefit. Beyond this boundary, improvements in pointwise accuracy are typically accompanied by degraded consistency and increased overconfidence.

Effective navigation architectures respect this wall. They allow

learning to operate where information exists and enforce conservative behavior where it does not. Learning is permitted to refine uncertainty, residuals, and context-dependent behavior, but the estimator remains authoritative wherever information is fundamentally limited.

This leads to a clear design principle. Learning should enhance estimation where physics-based models are weak, but estimation must remain in control where information is scarce. Navigation systems that integrate learning successfully do not treat AI as a replacement for structure. They treat it as a tool that operates within structure, constrained by observability and uncertainty.

5.4 Semantic Reasoning and Foundation Models

Recent advances in foundation models have enabled navigation systems to reason not only about geometry, but also about semantic context. This marks a shift from navigation systems that answer only *where* the platform is, to systems that can also reason about *what* is present in the environment and how it should influence motion. Classical navigation pipelines based on EKF or factor graph formulations remain responsible for estimating state under physical constraints. They provide position, velocity, orientation, and uncertainty in a well-defined state space. Foundation models, by contrast, operate at a semantic level. They interpret scenes, objects, affordances, and instructions expressed in natural language. The value of this separation is architectural. Semantic models do not replace geometric estimation; they augment it by shaping the cost structure under which estimation and planning operate. Rather than producing state estimates directly, semantic reasoning modifies how the system evaluates risk, preference, and feasibility.

This interaction is naturally expressed through semantic cost-mapping. High-level semantic cues are translated into modifications

of the estimator's cost or uncertainty landscape, without violating the underlying physical model.

Example
A command such as *"Navigate to the loading dock, but avoid the wet floor"* cannot be resolved through geometry alone. A foundation model may recognize the semantic class *wet floor* from visual input and associate it with increased risk. The correct integration point is not a direct change in state, but an increase in cost or covariance associated with the corresponding region in the map, effectively creating a soft no-go zone. The estimator and planner then remain free to reason under uncertainty while respecting this semantic constraint.

The role of the practitioner is to ensure that semantic reasoning remains advisory rather than authoritative. Foundation models may suggest constraints, preferences, or risk modifiers, but they must not override the physical consistency enforced by the state-space model. Semantic outputs should be expressed as costs, priors, or uncertainty modulation-not as direct state corrections. When integrated in this manner, foundation models extend the expressive power of navigation systems without undermining their reliability. They enable context-aware behavior while preserving the estimator's role as the final arbiter of belief, uncertainty, and feasibility.

Engineering Principle: Learning

Learning improves navigation systems when it models error, uncertainty, and context-dependent behavior that physics-based models cannot capture easily. Learning fails when it attempts to replace observability, uncertainty propagation, or estimator structure. The estimator must remain authoritative wherever information is fundamentally limited.

6 Learned Error and Residual Models

The most effective role of learning in navigation systems is not state estimation, but error modeling. Rather than predicting where the system is, learning is best suited to modeling how and why estimation errors evolve.

This distinction is subtle but fundamental. State estimation is constrained by observability and physics. Error processes, by contrast, are shaped by sensor imperfections, environmental conditions, and unmodeled dynamics-domains where analytical models are often weak.

This chapter examines how learning can model bias dynamics, residual corrections, and context-dependent error behavior, and explains why residual learning generalizes better than direct state prediction.

6.1 Learning Bias Dynamics

Biases are among the dominant sources of long-term navigation error. In inertial systems, accelerometer and gyroscope biases introduce errors that grow through integration. Classically, these biases are modeled as random walks:

$$\dot{\mathbf{b}}(t) = \mathbf{w}_b(t) \tag{6.1}$$

where $\mathbf{b}(t)$ denotes the bias vector and $\mathbf{w}_b(t)$ is zero-mean noise.

This model is intentionally conservative. It reflects uncertainty rather than precise knowledge of bias evolution.

In reality, bias dynamics depend on latent variables such as temperature, vibration, aging, and operational regime. These dependencies are difficult to capture analytically.

Learning-based models can approximate bias dynamics by conditioning on contextual signals. A learned bias evolution model may be expressed as

$$\dot{\mathbf{b}}(t) = f_\theta(\mathbf{c}(t)) \tag{6.2}$$

where $\mathbf{c}(t)$ is a vector of contextual features and $f_\theta(\cdot)$ is a learned mapping.

Crucially, learning does not replace the bias state. It augments its dynamics. The bias remains an explicit part of the estimator state, preserving uncertainty propagation and observability structure.

When bias learning is embedded inside an error-state formulation, the estimator retains authority over how learned corrections influence the state. If learned predictions become unreliable, they can be downweighted or disabled without destabilizing the system.

This separation is essential. Learning that replaces bias states removes the estimator's ability to reason about long-term error growth.

6.2 Learning Residual Corrections

Residuals encode the mismatch between predicted and observed measurements. They represent the point at which model assumptions meet reality.

Given a nominal state $\bar{\mathbf{x}}_k$ and a measurement $\mathbf{z}_k$, the innovation (residual) is

$$\boldsymbol{\nu}_k = \mathbf{z}_k - h(\bar{\mathbf{x}}_k) \tag{6.3}$$

Classical estimators assume that $\boldsymbol{\nu}_k$ is zero-mean noise. In prac-

tice, residuals often exhibit structure: systematic bias, correlation with motion, or dependence on environment.

Learning-based residual models aim to capture this structure. A learned correction may be written as

$$\boldsymbol{\nu}_k^{\text{corr}} = \boldsymbol{\nu}_k - g_\theta(\mathbf{c}_k) \tag{6.4}$$

where $g_\theta(\cdot)$ predicts systematic residual components based on context $\mathbf{c}_k$.

This approach has two key advantages.

First, residuals are local quantities. Errors in learned residual corrections affect individual updates rather than the entire state trajectory.

Second, residuals are already part of the estimator's internal logic. Correcting them preserves the estimator's structure and uncertainty handling.

Learning residuals therefore improves estimation fidelity while maintaining estimator transparency. It does not entangle learning with state propagation or fusion logic.

6.3 Context-Dependent Error Behavior

Navigation errors are rarely stationary. Sensor performance depends on motion, environment, and operational conditions.

Classical models approximate this variability using fixed or heuristically tuned noise parameters. While robust, this approach sacrifices accuracy under favorable conditions and may underestimate uncertainty under adverse ones.

Learning enables context-dependent error modeling. Measurement uncertainty can be expressed as

$$\mathbf{R}_k = g_\theta(\mathbf{c}_k) \tag{6.5}$$

where $\mathbf{R}_k$ is the measurement covariance and $\mathbf{c}_k$ encodes contextual information such as speed, vibration, or environmental cues.

This formulation allows the estimator to adapt trust dynamically. Under favorable conditions, uncertainty contracts. Under degraded conditions, uncertainty expands.

Importantly, learning adjusts uncertainty, not state. The estimator remains responsible for fusion, propagation, and consistency.

Context-dependent modeling is particularly valuable in multimodal systems. It enables principled weighting between sensors whose reliability varies with operating regime.

When implemented correctly, learning improves robustness rather than undermining it.

6.4 Why Residual Learning Generalizes Better

Residual learning generalizes better than direct state prediction for structural reasons.

First, residuals live in a constrained space. They are small, bounded quantities whose statistics are shaped by estimator design. Learning in this space is easier and more stable than learning full state mappings.

Second, residuals are invariant to global reference frames. A residual model trained in one environment often transfers to another, while a state prediction model implicitly encodes environment-specific structure.

Third, residual learning preserves estimator feedback. If learned corrections degrade, the estimator detects inconsistency through residual statistics and adapts accordingly.

Direct state learning lacks these safeguards. Errors propagate silently through recursion, and failure is difficult to localize.

Table 6.1 summarizes this distinction.

Table 6.1: Residual learning versus direct state learning in navigation systems.

Aspect	**Residual Learning**	**State Learning**
Integration point	Inside estimator update	Outside estimator
Error impact	Local, bounded	Global, accumulative
Uncertainty handling	Preserved	Often absent
Generalization	High	Low
Failure detection	Explicit	Implicit or missing

Residual learning aligns with the philosophy of structured estimation. It leverages learning where physics is weak, while preserving estimator authority where structure is essential.

This design pattern recurs throughout successful AI-enhanced navigation systems. Learning does not replace estimation. It sharpens it.

AI Implementation Notes

Learning should be integrated based on its effect on consistency and boundedness, not accuracy alone. Any improvement in pointwise error that degrades uncertainty calibration introduces overconfidence by design.

7 Trust, Quality, and Measurement Weighting

Navigation systems rarely fail because sensors suddenly stop working. They fail because sensor quality degrades gradually, while the estimation system continues to trust incoming data as if nothing has changed. By the time failure becomes visible, the internal estimate is often already inconsistent.

Trust is therefore a central concept in navigation systems. It governs how measurements influence the estimate, how uncertainty evolves over time, and whether degradation is detected early or only after divergence has occurred.

This chapter examines how trust is handled in classical estimation, why static covariance tuning is insufficient in real systems, and how learning-based trust models can be integrated without violating estimator consistency.

7.1 Sensors Degrade Continuously

Real sensors rarely exhibit binary behavior. They do not transition cleanly from a fully operational state to complete failure. Instead, sensor quality degrades continuously as a function of environment, operating conditions, and time.

Common examples include GNSS degradation due to multipath or occlusion, vision degradation caused by lighting changes or mo-

tion blur, and inertial sensor bias variation driven by temperature, vibration, or aging.

A generic measurement model can be written as

$$\mathbf{z}_k = h(\bar{\mathbf{x}}_k) + \mathbf{v}_k, \tag{7.1}$$

where $\mathbf{v}_k$ represents measurement noise.

In practice, the statistics of $\mathbf{v}_k$ are not stationary. Both its variance and structure depend on latent conditions that are rarely modeled explicitly. Classical estimators implicitly assume that sensor quality is either acceptable or unacceptable: measurements are either trusted or rejected. This assumption does not reflect physical reality.

As a result, degraded measurements are often trusted longer than they should be. By the time a hard threshold is crossed, the estimator may already be overconfident and inconsistent. Recognizing that sensor quality is a continuous variable rather than a discrete state is therefore a prerequisite for robust navigation.

7.2 Limits of Classical Covariance Tuning

Classical navigation systems encode trust through the measurement covariance matrix $\mathbf{R}_k$. Measurement weighting is governed by the Kalman gain,

$$\mathbf{K}_k = \mathbf{P}_k \mathbf{H}_k^\top \left(\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^\top + \mathbf{R}_k \right)^{-1}. \tag{7.2}$$

In most systems, $\mathbf{R}_k$ is tuned offline. Engineers select conservative values intended to guarantee stability across a wide range of conditions. While this approach is robust, it has two fundamental limitations.

First, it is pessimistic. Under favorable conditions, the estimator underutilizes high-quality measurements and sacrifices achievable

accuracy. Second, it is brittle. When operating conditions deviate significantly from those assumed during tuning, $\mathbf{R}_k$ no longer reflects reality.

Manual retuning is impractical in dynamic environments, and rule-based adaptations rely on explicit indicators that are often unavailable or unreliable. As navigation systems incorporate more sensors and operate across broader regimes, static covariance tuning becomes a scalability bottleneck.

7.3 Learned Trust Scores

Learning-based models provide a mechanism to estimate trust dynamically. Instead of predicting state directly, learning predicts how much the system should trust a given measurement under current conditions.

Let a learned trust score be denoted by $\tau_k \in [0, 1]$, defined as

$$\tau_k = g_\theta(\mathbf{c}_k), \tag{7.3}$$

where $\mathbf{c}_k$ encodes contextual information such as motion regime, sensor diagnostics, or environmental cues.

Trust scores can modulate measurement uncertainty through

$$\mathbf{R}_k^{\text{eff}} = \frac{1}{\tau_k}\mathbf{R}_k. \tag{7.4}$$

When trust is high, effective uncertainty contracts. When trust is low, uncertainty expands. This formulation has several advantages. Trust becomes a continuous signal rather than a binary decision. Degradation is handled gracefully rather than abruptly. Most importantly, learning operates on uncertainty rather than on state, leaving the estimator responsible for fusion and propagation.

Learned trust scores are also interpretable. They can be moni-

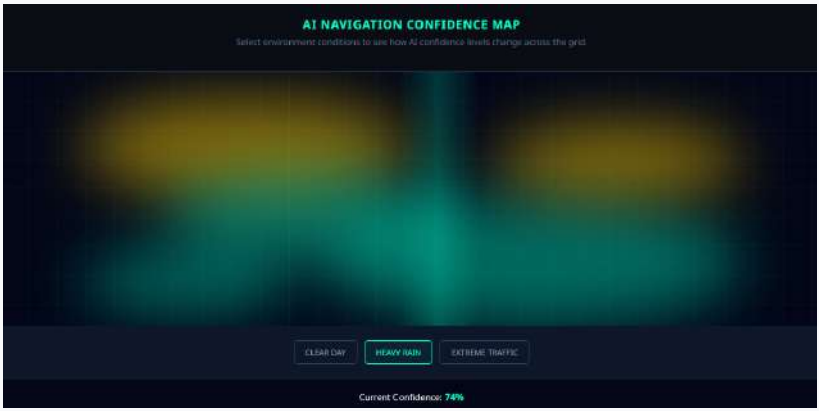


Figure 7.1: **AI Navigation Confidence Map.** Spatial confidence field estimated at runtime under varying environmental stress. Confidence reflects estimator trust and uncertainty modulation rather than geometric accuracy. Such fields are used to adapt sensor weighting and recovery behavior without modifying state estimates.

tored, logged, and audited as system-level signals. Learning-based trust estimation is particularly valuable when classical quality indicators are weak or ambiguous, as it can capture complex, nonlinear relationships between context and measurement reliability.

7.4 Injecting Trust Without Breaking Consistency

Injecting learned trust into a navigation system is deceptively simple. When done incorrectly, it can destroy estimator consistency.

The central rule is that trust must influence measurement weighting, not innovation structure or state updates. Learning should not alter the measurement model $h(\cdot)$, modify residuals directly, or inject state corrections outside the estimator.

Instead, learned trust should modulate covariance or gating thresholds. In this configuration, the estimator continues to compute

innovations, evaluate residual statistics, and propagate uncertainty according to its internal models. Learning becomes a second-order influence whose effect is bounded by estimator logic.

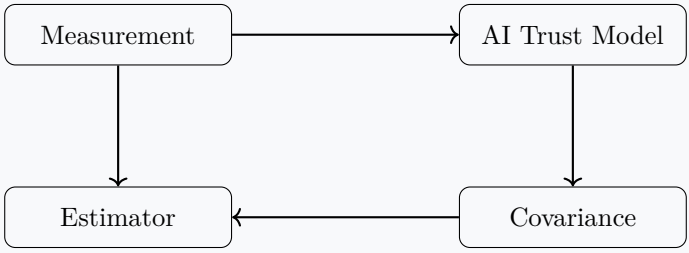


Figure 7.2: Learning observes measurements and context, modulates uncertainty, and never injects state directly. The estimator remains the sole authority for state updates.

When trust is injected correctly, learning improves both accuracy and robustness. When injected incorrectly, it produces systems that fail silently and unpredictably.

Table 7.1 summarizes safe and unsafe trust integration strategies.

Table 7.1: Trust integration strategies in navigation systems.

Strategy	**Effect**	**Risk**
Covariance scaling	Adaptive weighting	Low
Adaptive gating	Context-aware rejection	Low
Residual correction	Local influence	Moderate
Direct state correction	Global influence	High
End-to-end replacement	Opaque behavior	Critical

The principle is unambiguous. Trust is a first-class signal in

navigation systems. Learning can estimate it more flexibly than classical models, but the estimator must remain the final authority.

7.5 Explainability Beyond Black-Box Trust

In safety-critical navigation systems, trust cannot be implicit. Any mechanism that influences estimation must be inspectable, auditable, and attributable to observable signals. This requirement applies with particular force to learning-based components, whose internal logic is otherwise opaque.

Explainability in this context is not a visualization exercise. It is an engineering interface. The purpose of explainability is not to justify decisions after the fact, but to expose the signals that caused those decisions in a form that can be evaluated, tested, and acted upon.

In structured navigation systems, learning-based trust or weighting mechanisms must therefore provide interpretable outputs alongside their numerical effect. When a learned component downweights a sensor or measurement source, the system must expose *why* that decision was made in terms of estimator-relevant evidence.

This evidence may take the form of saliency measures, attention weights, residual pattern classification, or other structured diagnostics. The specific representation is less important than the principle: learned decisions must be traceable to observable inconsistencies between data and model assumptions.

Example
A learned trust model reduces the weight of GNSS measurements during operation.
Diagnostic signal: GNSS weight reduced to 0.15. *Explanatory context:* Detected pseudo-range residual patterns consistent with multipath interference and inconsistent with expected receiver dynamics.
The estimator does not act on the explanation directly. It acts on the adjusted uncertainty. The explanation exists to support monitoring, debugging, and certification, not to replace estimation logic.

The engineering value of explainability lies in its ability to separate *effect* from *cause*. The estimator responds to quantified uncertainty. The operator or developer inspects the explanatory interface to understand whether that response is justified.

Without such interfaces, learning-based components remain opaque. They may function correctly under nominal conditions, yet provide no basis for diagnosis when behavior changes. In contrast, interpretable trust mechanisms transform learning from an oracle into a diagnostic subsystem whose behavior can be validated over time.

Explainability, when designed as an estimator-facing interface rather than a visualization artifact, enables learning-based navigation systems to meet the requirements of long-term reliability, operational trust, and safety-critical transparency.

AI Implementation Notes

Learning should be used to estimate measurement quality and uncertainty, not to replace estimator logic. Trust must modulate covariance or gating, never state or residual structure directly. Any learned component that increases confidence without adding information degrades consistency, even if short-term accuracy improves.

8 What Not to Learn

As learning-based methods continue to demonstrate impressive results in controlled benchmarks, there is a growing temptation to expand their role beyond safe architectural boundaries. In navigation systems, this temptation is particularly dangerous.

Some learning formulations are not merely suboptimal. They are structurally incompatible with long-term reliability, estimator consistency, and failure awareness. Their failure modes are not obvious during development and often emerge only after prolonged operation or under distribution shift.

This chapter focuses explicitly on learning approaches that should *not* be used in operational navigation systems. The issue is not lack of accuracy. It is violation of core estimation principles. These approaches tend to erase uncertainty, entangle failure modes, and eliminate the signals required for diagnosis and recovery.

Understanding what not to learn is therefore a prerequisite for any serious integration of AI into navigation systems.

8.1 Direct State Regression

Direct state regression refers to learning a mapping from sensor data directly to the navigation state,

$$\hat{\mathbf{x}}_k = f_\theta(\mathbf{z}_{1:k}), \tag{8.1}$$

where $\mathbf{z}_{1:k}$ denotes a history of sensor measurements and $f_\theta(\cdot)$ is a learned model.

At first glance, this approach appears attractive. It bypasses explicit modeling, avoids linearization, and often produces low error on curated datasets. However, direct state regression violates fundamental estimation constraints.

First, it ignores observability. The model outputs a complete state vector even when parts of the state are unobservable. This creates an illusion of certainty where none exists and masks the growth of uncertainty along unobservable directions.

Second, it removes explicit uncertainty representation. Without a covariance or belief model, the estimate cannot be fused, gated, or validated. Any downstream system must treat the output as fact rather than belief.

Third, errors propagate globally. A single incorrect prediction contaminates the entire state estimate, with no mechanism for localization, isolation, or correction.

From a system perspective, direct state regression replaces structured uncertainty with opaque confidence. This tradeoff is unacceptable in navigation systems that must detect degradation, diagnose failure, and recover gracefully.

8.2 End-to-End Navigation Mappings

End-to-end navigation models extend direct regression by learning the entire navigation pipeline,

$$\hat{\mathbf{x}}_{1:k} = f_\theta(\mathbf{z}_{1:k}). \tag{8.2}$$

Such models implicitly entangle perception, estimation, and temporal reasoning. They may incorporate recurrent architectures, attention mechanisms, or transformer-based sequence models.

While expressive, end-to-end mappings introduce severe structural risks. When performance degrades, it becomes unclear whether the cause lies in sensing, perception, dynamics modeling, or temporal aggregation. Failure modes are opaque, and internal assumptions are inaccessible.

From an estimation standpoint, end-to-end navigation removes explicit state uncertainty, innovation statistics, and consistency checks. Without these signals, the system has no principled way to determine when it is wrong.

This absence of self-awareness is more dangerous than moderate inaccuracy. End-to-end navigation models may appear accurate while operating far outside their valid regime. As a result, recovery options are limited to retraining, heuristic resets, or full system restart.

End-to-end navigation therefore conflicts directly with the requirements of safety-critical and long-duration deployment.

8.3 Entangled Perception–Estimation Failures

One of the most subtle risks of aggressive learning integration is failure entanglement. Classical navigation systems enforce a strict separation: perception produces measurements, and estimation reasons about state under uncertainty. End-to-end learning collapses this separation. Figure 8.1 illustrates this entanglement.

When perception quality degrades, the system has no independent signal to detect or compensate. There is no residual, no innovation, and no statistical test indicating loss of validity. Failures therefore remain silent, and the system continues to output confident estimates while operating outside its valid regime.

This failure mode is particularly dangerous in long-duration missions and safety-critical applications, where delayed detection often results in irreversible divergence.

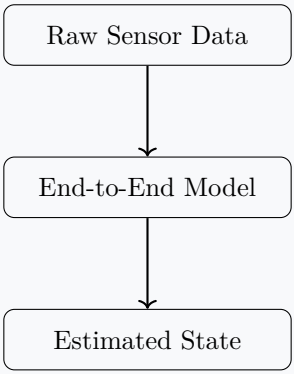


Figure 8.1: End-to-end navigation mapping. Perception and estimation are entangled, preventing uncertainty reasoning, failure isolation, and principled recovery.

8.4 System-Level Consequences

The learning approaches discussed above share a common trait: they erase system structure. By removing explicit estimation stages, they eliminate the very signals required for monitoring, validation, and recovery. Table 8.1 summarizes learning approaches that undermine navigation system reliability.

The conclusion is not that learning should be avoided. It is that learning must be constrained. Navigation systems succeed because they expose uncertainty, structure, and failure. Any learning approach that removes these properties is incompatible with reliable deployment. Knowing what *not* to learn is therefore as important as knowing what to learn.

Table 8.1: Learning approaches that undermine navigation system reliability.

Approach	**Primary Issue**	**Risk**
Direct state regression	Ignores observability and uncertainty	High
End-to-end navigation	Opaque internal structure and failure modes	Critical
State replacement without covariance	Overconfidence and silent divergence	High
Perception–estimation entanglement	No failure isolation or diagnostics	Critical

AI Implementation Notes

Learning must never replace estimator structure, uncertainty propagation, or observability constraints. Any learned component that outputs state without calibrated uncertainty, bypasses innovation statistics, or entangles perception with estimation introduces silent failure modes by design. In navigation systems, learning is safe only when it operates inside the estimator and leaves authority over belief and uncertainty unchanged.

9 Learning Interfaces That Survive Reality

After understanding what learning should model and what it should avoid, the remaining question is architectural: how should learning-based components be connected to a navigation system?

In practice, most failures of AI-enhanced navigation systems are not caused by weak models. They are caused by poor interfaces. A strong model connected incorrectly is more dangerous than a modest model connected safely.

This chapter focuses on learning interfaces that preserve estimator structure, maintain consistency, and enable failure detection. The central message is simple: in navigation systems, interfaces matter more than model choice.

9.1 Pseudo-Measurements

One of the safest and most widely applicable interfaces between learning and estimation is the pseudo-measurement.

A pseudo-measurement is a learned output that is injected into the estimator using exactly the same mathematical structure as a physical measurement. From the estimator's perspective, there is no distinction between a sensor measurement and a learned constraint-only the associated uncertainty differs.

A classical measurement model can be written as

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k. \tag{9.1}$$

A learned pseudo-measurement replaces $\mathbf{z}_k$ with a learned quantity $\mathbf{z}_k^{\mathrm{AI}}$,

$$\mathbf{z}_k^{\mathrm{AI}} = h_{\mathrm{AI}}(\mathbf{x}_k) + \mathbf{v}_k^{\mathrm{AI}}. \tag{9.2}$$

Crucially, the estimator does not know whether a measurement is physical or learned. It treats both as constraints with associated uncertainty and applies the same fusion, gating, and consistency logic.

This interface has several decisive advantages. First, it preserves estimator authority. The estimator remains fully responsible for state propagation, fusion, and uncertainty evolution. Second, it localizes failure. If a learned pseudo-measurement degrades, its influence is bounded by its covariance and does not contaminate the entire state trajectory. Third, it is reversible. Pseudo-measurements can be downweighted, disabled, or replaced without changing the estimator architecture.

Common uses of pseudo-measurements include learned bias estimates, learned velocity constraints, learned scale or alignment cues, and environment-dependent corrections. The pseudo-measurement pattern is therefore not a workaround or compromise. It is a first-class architectural primitive for safe integration of learning into navigation systems.

9.2 Learned Priors vs Learned Likelihoods

Learning can enter an estimation system in two fundamentally different ways: through priors or through likelihoods.

A learned prior modifies the belief over the state before mea-

surements are applied,

$$p(\mathbf{x}_k) \leftarrow p_{\mathrm{AI}}(\mathbf{x}_k). \tag{9.3}$$

A learned likelihood modifies how measurements constrain the state,

$$p(\mathbf{z}_k \mid \mathbf{x}_k) \leftarrow p_{\mathrm{AI}}(\mathbf{z}_k \mid \mathbf{x}_k). \tag{9.4}$$

This distinction has deep practical consequences. Learned priors are global. They bias the entire state estimate and influence all subsequent updates. If incorrect, their effect is widespread and difficult to isolate or reverse.

Learned likelihoods, by contrast, are local. They affect individual measurement updates and are naturally moderated by estimator logic. Their influence is bounded in time and space, and inconsistencies are often visible through innovation statistics.

For this reason, learned likelihoods are generally safer than learned priors. Examples include adaptive measurement covariance, context-dependent gating, and residual-based trust modulation. Learned priors may still be useful for initialization or coarse regularization, but they require extreme caution. Once injected, they continue to bias the system even when contradicted by incoming data.

A robust design principle therefore emerges: when in doubt, learn likelihoods, not priors.

9.3 Why Interfaces Matter More Than Model Choice

It is tempting to focus on model architecture: neural network depth, attention mechanisms, or training strategy. In navigation systems, these details are secondary.

What determines system reliability is how learning interacts with uncertainty, recursion, and failure detection. Figure 9.1 illustrates

the contrast between unsafe and safe integration paths.

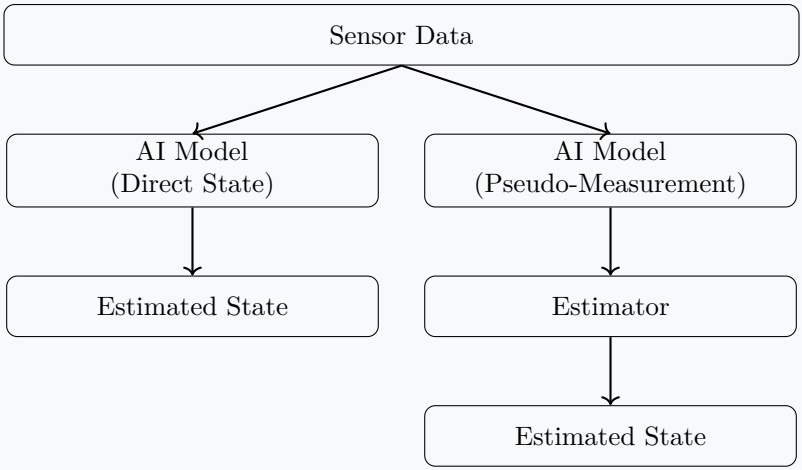


Figure 9.1: Model choice versus interface choice. Safe interfaces preserve estimator authority and failure isolation, while unsafe interfaces bypass uncertainty and diagnostics.

In the unsafe path, learning replaces estimation. Uncertainty, observability, and diagnostics are lost. In the safe path, learning augments estimation. Structure is preserved, and failure remains detectable.

Table 9.1 summarizes common learning interfaces and their system-level impact.

The conclusion is unambiguous. In navigation systems, robustness is an architectural property. A mediocre model with a safe interface will outperform a strong model with a fragile interface. Systems that survive reality are not those with the most advanced models, but those with the most disciplined integration.

Table 9.1: Learning interfaces and their system-level impact.

Interface	**System Effect**	**Risk**
Pseudo-measurements	Local, bounded influence	Low
Adaptive covariance	Improved consistency	Low
Learned likelihoods	Context-aware weighting	Low
Learned priors	Global bias	Moderate
Direct state injection	Estimator bypass	High

AI Implementation Notes

Learning should never bypass estimator structure. Safe integration routes learning through pseudo-measurements, likelihoods, or uncertainty modulation, where its influence is local, bounded, and observable. Any learned component that injects state directly, suppresses uncertainty growth, or removes innovation-based diagnostics introduces silent failure modes by design.

References - Part II

1. Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*. Academic Press, 1988.
2. X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4), 2003.
3. S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.
4. A. N. Bishop. Innovation-based fault detection in Kalman filters. *IEEE Control Systems Magazine*, 2007.
5. D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations (ICLR)*, 2017.
6. C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. *International Conference on Machine Learning (ICML)*, 2017.
7. M. Ovadia et al. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
8. A. Kendall and Y. Gal. What uncertainties do we need in Bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
9. A. Malinin and M. Gales. Predictive uncertainty estimation via prior networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
10. B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

PART III

Hybrid Navigation Architectures

Filters, Graphs, and Structured AI Integration

10 Filter-Based vs Graph-Based Systems

Modern navigation systems are typically built around one of two estimation paradigms: recursive filtering or graph-based optimization. Both are mathematically grounded, widely deployed, and capable of high accuracy. However, their system-level behavior differs in ways that become critical under real operating conditions.

This chapter contrasts filter-based and graph-based navigation architectures, with particular emphasis on how each paradigm interacts with learning-based components. The goal is not to declare a winner, but to expose the trade-offs that matter in deployed systems: latency, consistency, complexity, and failure behavior.

10.1 EKF and ESKF Pipelines

Filter-based navigation systems implement recursive Bayesian estimation. At each time step, the estimator maintains a belief over the current state and updates it incrementally as new measurements arrive.

In error-state formulations, the system propagates a nominal trajectory while estimating a small error state,

$$\delta \mathbf{x}_k = \begin{bmatrix} \delta \mathbf{p}_k & \delta \mathbf{v}_k & \delta \boldsymbol{\theta}_k & \delta \mathbf{b}_k \end{bmatrix}^\top . \tag{10.1}$$

Prediction and update follow the standard Kalman structure. During propagation, the error-state mean and covariance are predicted as

$$\delta \mathbf{x}_{k|k-1} = \mathbf{F}_{k-1}\, \delta \mathbf{x}_{k-1}, \tag{10.2}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\, \mathbf{P}_{k-1}\, \mathbf{F}_{k-1}^{\top} + \mathbf{Q}_{k-1}. \tag{10.3}$$

During the measurement update, the Kalman gain is computed as

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\, \mathbf{H}_k^{\top} \left(\mathbf{H}_k\, \mathbf{P}_{k|k-1}\, \mathbf{H}_k^{\top} + \mathbf{R}_k \right)^{-1}, \tag{10.4}$$

and the posterior error-state mean is updated according to

$$\delta \mathbf{x}_k = \delta \mathbf{x}_{k|k-1} + \mathbf{K}_k\, \boldsymbol{\nu}_k, \tag{10.5}$$

where $\boldsymbol{\nu}_k$ denotes the innovation.

The corresponding posterior covariance update is given by

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_{k|k-1}. \tag{10.6}$$

In practical navigation systems, the numerically robust Joseph form is often used,

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_{k|k-1}\, (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^{\top} + \mathbf{K}_k\, \mathbf{R}_k\, \mathbf{K}_k^{\top}, \tag{10.7}$$

to preserve symmetry and positive semi-definiteness of the covariance matrix.

Filter-based pipelines exhibit several defining properties. They are low-latency, as each update is local in time and computationally bounded. They expose explicit internal signals such as innovations, covariance growth, and consistency metrics, which are essential for failure detection and for safe integration of learning-based components. They are also inherently causal: the system estimates the present using the past, without revisiting history.

The primary limitation of filters lies in linearization and correlation handling. Once a measurement is fused, its effect cannot be revisited. Errors introduced early may persist even when contradictory information arrives later.

Learning integrates naturally into filter pipelines when it augments residuals, covariances, or bias dynamics. When learning attempts to replace the filter itself, consistency and observability guarantees are lost.

10.2 Factor Graph Formulations

Graph-based navigation systems formulate estimation as a global optimization problem. States are represented as nodes, and measurements are represented as factors connecting subsets of states.

Let the set of states be $\{\mathbf{x}_i\}$ and the set of measurements be $\{\mathbf{z}_j\}$. Estimation is posed as

$$\{\hat{\mathbf{x}}_i\} = \arg\min_{\{\mathbf{x}_i\}} \sum_j \left\| \mathbf{z}_j - h_j(\mathbf{x}_{\mathcal{C}(j)}) \right\|^2_{\mathbf{\Omega}_j}, \tag{10.8}$$

where $\mathbf{\Omega}_j$ is an information matrix and $\mathcal{C}(j)$ denotes the set of states involved in factor j.

Graph-based systems offer several advantages. They handle delayed measurements naturally, allow relinearization, and enable global consistency by revisiting past states when new constraints arrive. These properties make graphs attractive for vision-based navigation, mapping, and loop closure.

However, these benefits come at a cost. Latency grows with graph size. Consistency becomes implicit rather than explicit. Failure detection becomes more difficult because residuals are distributed across the graph rather than localized in time.

From a learning perspective, graph-based systems are powerful but fragile. Learning can influence factor weights, add learned

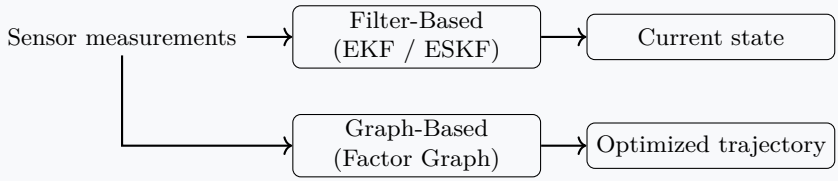


Figure 10.1: Filter-based and graph-based navigation process the same sensor measurements but produce fundamentally different outputs: filters estimate the current state through local recursion, while graph-based systems optimize over a trajectory.

factors, or propose additional constraints. However, an incorrect learned factor can propagate globally across the solution, contaminating large portions of the trajectory.

Unlike filters, graph-based systems rarely degrade gracefully. When they fail, they tend to fail through global inconsistency rather than localized divergence.

10.3 Latency, Consistency, and Complexity Trade-offs

The choice between filters and graphs is not primarily about accuracy. It is about how systems behave under stress. Figure 10.1 contrasts the two paradigms.

Table 10.1 summarizes the key system-level trade-offs.

A clear pattern emerges. Filter-based systems are conservative, interpretable, and resilient. Graph-based systems are expressive, flexible, and globally consistent when operating within their assumptions.

Hybrid systems often combine both approaches: filters for real-time estimation and graphs for background optimization or mapping. For AI-enhanced navigation, this distinction is critical. Learning components that affect global structure belong in graph-based systems only with extreme caution. Learning components that model

Table 10.1: System-level trade-offs between filter-based and graph-based navigation.

Aspect	**Filter-Based**	**Graph-Based**
Latency	Low, bounded	Higher, grows with graph size
Consistency signals	Explicit (covariance, innovation)	Implicit, distributed
Handling delayed data	Limited	Natural
Failure localization	Local	Global
AI integration	Safe for residuals and trust	Risky for global factors

error, trust, or residuals integrate more safely into filters. The architectural choice therefore precedes the learning choice. Models do not determine system behavior. Architectures do.

AI Implementation Notes

Learning integrates most safely inside filter-based pipelines when it augments residuals, uncertainty, or bias dynamics, where its influence is local, bounded, and observable. In graph-based systems, learning should be restricted to soft factors or adaptive weights and must never introduce hard global priors. Direct state correction or estimator replacement by learning breaks consistency and eliminates diagnosable failure modes by design.

11 Injecting Learning into Graphs and Filters

Once learning is constrained to safe roles-such as modeling residuals, estimating trust, or adapting uncertainty-the remaining challenge is architectural: how should learning-based components be connected to a navigation system.

In practice, many failures of AI-enhanced navigation systems are not caused by poor models, but by poor integration. The same learning component can behave safely or catastrophically depending on how it is injected into the estimator.

This chapter examines where learning integrates cleanly into filter-based and graph-based systems, where it violates core estimation assumptions, and how subtle correlation effects can lead to hidden coupling and long-term inconsistency.

11.1 Where Learning Fits Cleanly

Learning integrates cleanly when it respects the estimator's internal structure. In both filters and graphs, this structure is defined by explicit state variables, explicit uncertainty representation, and explicit update rules.

In filter-based systems, safe integration points include residual modeling, measurement covariance adaptation, and augmentation of bias dynamics. In all of these cases, learning operates on quantities

the estimator already computes, rather than replacing estimator logic.

Let the innovation be defined as

$$\boldsymbol{\nu}_k = \mathbf{z}_k - h(\bar{\mathbf{x}}_k). \tag{11.1}$$

A learned residual correction modifies this innovation locally,

$$\boldsymbol{\nu}_k^{\text{eff}} = \boldsymbol{\nu}_k - g_\theta(\mathbf{c}_k), \tag{11.2}$$

where $\mathbf{c}_k$ denotes contextual features.

Crucially, the filter still computes the Kalman gain, updates the covariance, and evaluates consistency. Learning influences the update, but does not replace it. If the learned correction degrades, its effect remains localized and bounded by estimator logic.

In graph-based systems, learning fits cleanly when it proposes *soft* constraints. A learned factor may be introduced with an explicit information matrix,

$$\left\| \mathbf{z}^{\text{AI}} - h(\mathbf{x}) \right\|^2_{\boldsymbol{\Omega}_{\text{AI}}}. \tag{11.3}$$

Here, learning contributes information, not belief. If the learned factor is weak or incorrect, its influence is moderated by optimization and competing constraints. The estimator retains global authority.

The common principle across both paradigms is locality. Learning should affect individual updates or factors, not the global belief directly.

11.2 Where Learning Breaks Assumptions

Learning breaks estimator assumptions when it bypasses uncertainty handling or violates independence structure.

In filter-based systems, this occurs when learning directly modi-

fies the state,

$$\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_k + f_\theta(\cdot), \tag{11.4}$$

without a corresponding update to the covariance.

Such injections create overconfidence by construction. The mean changes, but uncertainty does not. Consistency is lost even if accuracy improves temporarily.

In graph-based systems, assumptions break when learned components introduce hard constraints or global priors. A learned prior that biases multiple states simultaneously couples errors across the graph.

Unlike physical measurements, learned constraints often share training data, internal representations, or model parameters. Treating such constraints as independent violates the conditional independence assumptions that factor graphs rely on.

The result is optimistic solutions with underestimated uncertainty and limited failure observability.

Learning should therefore never replace state propagation, uncertainty propagation, or measurement likelihood structure. When learning bypasses these elements, failure becomes silent.

11.3 Correlation Leakage and Hidden Coupling

The most subtle failure mode of learning integration is correlation leakage.

Correlation leakage occurs when multiple learned outputs are treated as independent despite sharing common sources of information. Consider two learned corrections applied at successive time steps,

$$\boldsymbol{\nu}_k^{\mathrm{AI}}, \quad \boldsymbol{\nu}_{k+1}^{\mathrm{AI}}. \tag{11.5}$$

If both corrections are produced by the same model under similar context, they are statistically correlated. Treating them as

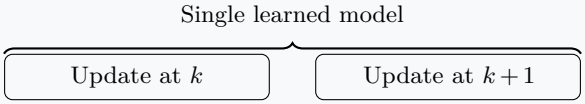


Figure 11.1: Correlation leakage: successive updates appear independent but share hidden dependence through a common learned model, violating independence assumptions.

independent updates leads to overconfident estimates.

In filters, this manifests as artificially shrinking covariance. In graphs, it manifests as overly stiff trajectories. The danger is that correlation leakage is invisible locally. Each update appears reasonable; inconsistency emerges only over time.

Learning-based systems are particularly susceptible to this effect because shared model parameters induce hidden coupling across updates.

Mitigating correlation leakage requires architectural discipline. Learned components should operate on short temporal windows, affect uncertainty explicitly, and be validated through innovation statistics rather than pointwise accuracy.

Figure 11.1 illustrates hidden coupling.

Correlation leakage is not a modeling error. It is an integration error. Robust navigation systems assume correlation unless proven otherwise.

AI Implementation Notes

Learning should be integrated only through estimator-native interfaces such as residuals, uncertainty, or soft factors. Any learned component that modifies state without corresponding uncertainty updates, introduces hard global constraints, or assumes independence without justification creates hidden coupling and silent inconsistency. In navigation systems, safe learning integration is defined by locality, bounded influence, and observability of failure.

12 Modular Degradation and Fallback

Reliable navigation systems are not designed to avoid failure. They are designed to continue operating when parts of the system fail. This distinction becomes critical once learning-based components are introduced.

Unlike classical models, learned components may fail silently, degrade gradually, or behave unpredictably outside their training regime. Their failure modes are often statistical rather than physical, and they do not necessarily produce explicit alarms when validity is lost.

For this reason, learning must be treated as a *non-essential* module. The navigation system must remain safe, bounded, and diagnosable even when learning is partially or fully disabled. Learning may improve performance, but it must never become a prerequisite for correctness.

This chapter focuses on architectural patterns that enable graceful degradation and controlled fallback in AI-enhanced navigation systems.

12.1 Disabling Learning at Runtime

A fundamental requirement for safe AI integration is the ability to disable learning at runtime.

Disabling learning does not mean stopping the navigation system. It means reverting the system to a conservative, fully model-based

operating mode whose behavior is well understood and bounded.

In filter-based systems, disabling learning typically involves removing learned residual corrections, freezing learned bias dynamics, and reverting measurement covariances or trust parameters to conservative defaults. If a learned correction enters the update as

$$\boldsymbol{\nu}_k^{\text{eff}} = \boldsymbol{\nu}_k - g_\theta(\mathbf{c}_k), \tag{12.1}$$

then runtime disabling corresponds to

$$\boldsymbol{\nu}_k^{\text{eff}} \leftarrow \boldsymbol{\nu}_k. \tag{12.2}$$

The estimator continues to function without architectural change. Uncertainty may grow more rapidly and performance may degrade, but consistency is preserved and failure remains observable.

In graph-based systems, disabling learning may involve removing learned factors, reducing their information weight, or preventing new learned constraints from being added to the graph. The defining requirement is reversibility. If learning cannot be removed cleanly without destabilizing the estimator, it is too deeply embedded.

Runtime disablement is therefore not an optional safety feature. It is a structural test of whether learning has been integrated correctly.

12.2 Conservative Fallback Modes

Fallback modes define how the system behaves once learning is degraded or disabled.

A conservative fallback mode is characterized by increased uncertainty, reduced reliance on weak or degraded sensors, and stricter gating thresholds. These modes are not designed to preserve accuracy. They are designed to preserve boundedness, predictability, and diagnosability.

From an estimation perspective, fallback corresponds to intentionally pessimistic assumptions. Noise covariances are inflated, trust is reduced, and updates become less aggressive. This behavior may appear overly conservative, but it ensures that the system remains self-aware and does not drift into silent inconsistency.

Learning may assist in *triggering* fallback by detecting anomalous patterns in residuals or innovation statistics. However, learning must not control fallback behavior itself. The decision to enter fallback must be based on estimator-native signals such as innovation inflation, covariance growth, or consistency violations.

Once in fallback, the system should remain stable even if learning remains disabled indefinitely. A fallback mode that relies on learning to function is not a fallback; it is a delayed failure.

12.3 Designing for Partial Failure

In real systems, failure is rarely total. Some learned components may remain valid while others degrade.

Designing for partial failure therefore requires strict modularity. Each learned component must have a well-defined interface, affect a limited part of the estimation pipeline, and be independently disableable without cascading effects.

For example, a learned trust model may be disabled while residual learning remains active, or vision-based learning may be disabled while inertial bias learning continues. The system must be able to absorb such changes without architectural restructuring.

Partial failure should lead to graceful performance degradation rather than abrupt collapse. This requires that learning-based components never become single points of failure and never introduce dependencies that the classical estimator cannot absorb.

Systems designed with this principle exhibit a clear hierarchy. Classical estimation defines the safety envelope and guarantees

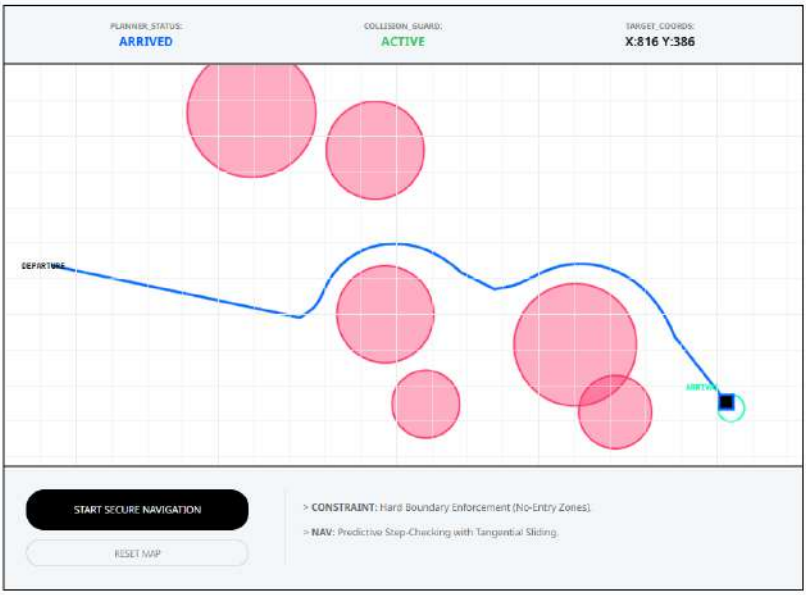


Figure 12.1: **Recovery-Oriented Autonomous Path Planning.** Navigation under hard safety constraints using classical potential fields augmented with AI-driven recovery logic. Estimation and safety constraints remain authoritative, while learning assists in escaping local minima and degraded regimes.

bounded, observable behavior. Learning operates strictly within that envelope, improving performance where valid and withdrawing cleanly when validity is lost.

12.4 Sustainable Path Planning

In many modern robotic systems, energy is as constraining as accuracy. Navigation decisions that ignore energetic cost may be optimal geometrically, yet suboptimal operationally. As a result, sustainable path planning has become a core requirement for long-duration

autonomy.

Classical path planning optimizes kinematic or geometric criteria such as distance, smoothness, or collision avoidance. Energy expenditure is typically treated indirectly or approximated with simple heuristics. In real environments, however, energetic cost depends strongly on terrain properties, aerodynamic effects, and operating conditions that are difficult to model analytically.

Learning-based models provide a practical mechanism for estimating these effects. Rather than replacing the planner or estimator, learning augments the cost model used during planning. The total navigation cost can be expressed as

$$C_{\text{total}} = C_{\text{kinematic}} + \int f_\theta(\text{terrain}, \text{wind}, \text{context})\, ds, \tag{12.3}$$

where $C_{\text{kinematic}}$ captures geometric and dynamic effort, and $f_\theta(\cdot)$ represents a learned estimate of environment-dependent energetic resistance.

In this formulation, learning does not predict state or control directly. It predicts cost. By estimating factors such as terrain friction, slope-dependent power loss, or aerodynamic drag, the system can evaluate routes not only by feasibility and safety, but also by expected energy consumption.

The architectural advantage of this approach is that energy optimization remains subordinate to estimation and planning structure. The estimator continues to provide state, uncertainty, and feasibility constraints. The planner incorporates learned energetic cost as an additional objective, allowing trade-offs between path length, risk, and endurance.

When integrated correctly, sustainable path planning improves operational longevity without compromising reliability. The system does not become energy-optimal by assumption, but energy-aware through accumulated experience. Learning captures persistent en-

vironmental resistance, while estimation and planning ensure that decisions remain physically consistent and reversible.

Sustainable navigation therefore extends the role of learning from perception and trust into long-horizon resource management. It does not change what navigation is. It changes what navigation optimizes for.

AI Implementation Notes

Learning must be designed as a removable, non-essential module. Every learned component must support clean runtime disablement and a conservative fallback mode. Fallback should be triggered by estimator signals, not AI confidence. If disabling learning destabilizes the system or removes observability, the integration is architecturally unsafe.

13 Why End-to-End Still Fails at Scale

End-to-end learning architectures remain attractive in navigation systems. They promise conceptual simplicity, reduced modeling effort, and impressive performance on curated benchmarks. In controlled settings, these promises are often fulfilled.

Yet despite repeated successes in laboratory experiments and short demonstrations, end-to-end navigation systems continue to fail when deployed at scale. These failures are not sporadic, nor are they merely the result of insufficient data or model capacity. They are structural.

This chapter explains why end-to-end navigation fails persistently in real systems, why these failures are difficult to mitigate even with extensive engineering effort, and why structured estimation architectures continue to dominate long-term, safety-critical deployments.

13.1 Generalization Limits

End-to-end navigation models learn a direct mapping from sensor histories to state estimates,

$$\hat{\mathbf{x}}_{1:k} = f_\theta(\mathbf{z}_{1:k}), \tag{13.1}$$

where the model implicitly absorbs perception, dynamics, and temporal reasoning into a single function.

Such models assume that the training data adequately spans the operational space. In real navigation systems, this assumption rarely holds. Operational environments vary continuously in sensor configuration, motion profiles, environmental conditions, and failure modes. Even small deviations from training conditions can induce large and unpredictable errors.

End-to-end models typically generalize by exploiting correlations rather than causality. They learn statistical regularities that are sufficient within the training distribution but fragile under distribution shift. When the correlation structure changes, performance degrades abruptly.

Unlike structured estimators, end-to-end models have no explicit mechanism to recognize extrapolation. They do not track observability, uncertainty growth, or consistency. As a result, they continue to produce confident outputs even when operating far outside their valid regime.

At scale, this behavior is unavoidable. As the diversity of operational conditions increases, the probability of encountering unseen or weakly represented scenarios approaches certainty. In such regimes, confidence becomes misleading and error becomes unbounded.

13.2 Debuggability and Certification

Navigation systems deployed in safety-critical or long-duration missions must be diagnosable. When performance degrades, engineers must be able to answer three questions: what failed, why it failed, and how to recover.

End-to-end systems do not provide these guarantees. Because perception, estimation, and temporal reasoning are entangled within a single model, errors cannot be localized. There are no explicit residuals, no innovation statistics, and no consistency metrics that indicate loss of validity.

From a certification perspective, this opacity is prohibitive. Certification frameworks depend on explicit assumptions, bounded behavior, and traceable failure modes. End-to-end models offer only empirical validation against finite datasets, with no principled way to reason about behavior outside those conditions.

Structured estimators, by contrast, expose internal signals that can be monitored, tested, and constrained. Even when augmented by learning, they retain interpretable interfaces that allow engineers to detect degradation and initiate recovery.

At scale, the cost of debugging dominates the cost of modeling. Systems that cannot be debugged cannot be certified. Systems that cannot be certified cannot be deployed.

13.3 Why Structure Wins Long-Term

The continued dominance of structured navigation architectures is not the result of conservatism or lack of innovation. It is the result of alignment with the physical and informational constraints of navigation.

Structured estimators explicitly encode causality, observability, uncertainty propagation, and failure awareness. These properties do not emerge automatically from data, regardless of model size or training effort.

Learning excels when applied within this structure. It improves residual modeling, trust estimation, and adaptation to context. When learning replaces structure, it removes the very signals required for robustness, diagnosis, and recovery.

At small scale, end-to-end systems may appear competitive. At large scale, structure becomes non-negotiable. This is not a limitation of current machine learning methods; it is a reflection of the nature of navigation itself.

Reliable navigation systems are not those that predict best under

ideal conditions. They are those that know when they are wrong, expose their uncertainty, and remain bounded when assumptions are violated.

AI Implementation Notes

End-to-end navigation should not be used at the system level in operational deployments. Learning should be integrated inside structured estimators to model residuals, trust, or context-dependent behavior. Any model that cannot expose uncertainty, innovation, or failure signals will not scale. Structure is not a constraint on AI; it is what allows AI to survive deployment.

References - Part III

1. Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley, 2001.
2. S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.
3. T. D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.
4. F. Dellaert and M. Kaess. Factor graphs for robot perception. *Foundations and Trends in Robotics*, 6(1–2), 2017.
5. M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6), 2008.
6. V. Indelman, S. Williams, M. Kaess, and F. Dellaert. Information-based inference for belief space planning. *International Journal of Robotics Research*, 34(14), 2015.
7. B. Or and I. Klein. A hybrid model and learning-based adaptive navigation filter. *IEEE Transactions on Instrumentation and Measurement*, 71, 2022.
8. B. Or and I. Klein. Learning vehicle trajectory uncertainty. *Engineering Applications of Artificial Intelligence*, 122, 2023.
9. J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, and L. Carlone. A survey on active simultaneous localization and mapping: State of the art and new frontiers. *IEEE Transactions on Robotics*, 2023.

PART IV

Reliability, Failure, and Runtime Awareness

Detection, Recovery, and Trust Under Uncertainty

14 Failure as Loss of Coherence

In classical engineering, failure is often treated as a discrete event. A component operates correctly until it fails, at which point it is reset, replaced, or shut down. This framing is appropriate for many physical subsystems, but it does not describe how navigation systems fail.

Navigation systems do not fail abruptly. They fail gradually. Failure manifests as a growing mismatch between the system model, the incoming data, and the estimator's internal belief. By the time a hard failure becomes visible, the system has often been operating in a degraded and increasingly inconsistent state for a significant period.

This chapter reframes failure not as a binary condition, but as a loss of coherence between estimation assumptions and reality. This perspective is essential for designing AI-enhanced navigation systems that remain reliable over long durations and under changing conditions.

14.1 Failure Is Not Binary

Navigation systems estimate state under uncertainty. As long as the estimator's assumptions remain approximately valid, performance degrades gracefully. When these assumptions are violated, degradation accelerates.

Failure therefore emerges as a continuum rather than an event.

Early in this process, accuracy may remain high and outputs may appear nominal. As degradation progresses, uncertainty becomes miscalibrated. Eventually, estimates diverge while remaining internally confident.

This progression is particularly dangerous in AI-enhanced systems. Learning-based components often maintain confident outputs even when operating outside their training regime. When such outputs are integrated without appropriate scrutiny, they accelerate the transition from mild degradation to silent failure.

From an estimation standpoint, failure begins when the underlying assumptions no longer hold: when the process model no longer reflects true system dynamics, when the measurement model no longer reflects sensor behavior, or when the uncertainty model no longer reflects actual error statistics. None of these conditions produce immediate alarms by default. They must be inferred from internal consistency signals.

Treating failure as binary encourages late detection and reactive response. Treating failure as gradual enables early intervention, bounded degradation, and controlled recovery.

14.2 Coherence Between Model, Data, and Uncertainty

Coherence refers to alignment between three elements: the system model, the incoming data, and the estimator's representation of uncertainty. When these elements are coherent, residuals behave as expected, uncertainty grows or contracts appropriately, and the system remains self-aware.

Loss of coherence occurs when this alignment breaks. For example, a learned residual correction may reduce pointwise error while invalidating covariance assumptions; a learned trust model may oversuppress uncertainty under previously unseen conditions; or a sensor

model may become outdated due to environmental change. In each case, the observable symptom is the same: error and uncertainty decouple.

Formally, let the estimation error be

$$\mathbf{e}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k^{\text{true}}, \tag{14.1}$$

and let $\mathbf{P}_k$ denote the estimated covariance. Coherence requires that the statistics of $\mathbf{e}_k$ remain compatible with $\mathbf{P}_k$. Loss of coherence occurs when this compatibility no longer holds.

AI does not inherently cause incoherence. Poor integration does. When learning modifies state estimates without corresponding updates to uncertainty, or modifies uncertainty without respecting estimator structure, coherence degrades silently. The system may appear to improve in accuracy while becoming fundamentally less reliable.

Reliable navigation systems therefore monitor coherence explicitly. They treat deviations between expected and observed residual behavior, innovation statistics, or covariance evolution as early indicators of failure. This shifts the goal of reliability. The objective is not to prevent all failure, but to detect loss of coherence early enough to respond before divergence becomes irreversible.

In AI-enhanced navigation systems, runtime awareness is not optional. It is the primary defense against confident failure. Systems that cannot observe their own loss of coherence cannot recover, regardless of model sophistication.

AI Implementation Notes

Failure in navigation systems should be treated as gradual loss of coherence, not as a binary event. Learning-based components must be integrated in ways that preserve alignment between residuals, uncertainty, and model assumptions. Any AI component that improves pointwise accuracy while degrading uncertainty calibration introduces systemic risk. Runtime monitoring should focus on detecting incoherence before divergence becomes visible.

15 Innovation, Residuals, and Detection

Reliable navigation systems do not rely on external alarms to detect failure. They rely on internal signals that reflect whether the estimator's assumptions remain valid.

Among these signals, innovations and residuals play a central role. They represent the point of contact between the model and reality. When learning-based components are introduced, these signals become even more critical: they are often the only way to detect when learning is no longer helping.

This chapter examines innovation statistics as runtime signals, how residual energy reveals gradual degradation, and how failures of learned components can be detected before they become catastrophic.

15.1 Innovation Statistics as Signals

In recursive estimation, the innovation quantifies the discrepancy between prediction and measurement. For a measurement $\mathbf{z}_k$, the innovation is given by

$$\boldsymbol{\nu}_k = \mathbf{z}_k - h(\bar{\mathbf{x}}_k), \tag{15.1}$$

where $\bar{\mathbf{x}}_k$ denotes the predicted nominal state.

Under correct modeling assumptions, $\boldsymbol{\nu}_k$ is a zero-mean random

variable whose covariance is

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k. \tag{15.2}$$

This relationship makes the innovation a powerful diagnostic signal. If the model, noise assumptions, and uncertainty are coherent, innovation statistics behave predictably.

A common scalar summary is the normalized innovation squared (NIS):

$$\mathrm{NIS}_k = \boldsymbol{\nu}_k \mathbf{S}_k^{-1} \boldsymbol{\nu}_k^\top. \tag{15.3}$$

When NIS values remain within expected bounds, the estimator is internally consistent. Persistent deviations indicate loss of coherence.

Innovation statistics are especially valuable in AI-enhanced systems. They provide a model-agnostic check on learned components. Regardless of how a correction was produced-analytical or learned-its validity is reflected in the innovation.

Learning should therefore never bypass innovation computation. If a learned component does not influence $\boldsymbol{\nu}_k$ or $\mathbf{S}_k$ in a traceable way, its failure cannot be detected.

15.2 Residual Energy Growth

While individual innovations capture instantaneous mismatch, long-term reliability depends on trends.

Residual energy aggregates innovation magnitude over time. A simple measure may be defined as

$$E_k = \sum_{i=k-W}^{k} \boldsymbol{\nu}_i \boldsymbol{\nu}_i^\top, \tag{15.4}$$

where W is a sliding window length.

Under stable conditions, residual energy fluctuates around a steady level. When assumptions degrade, residual energy grows systematically.

This growth often precedes visible divergence in state estimates. Accuracy may remain acceptable while residual energy trends upward.

Learning-based components frequently mask this effect. By reducing instantaneous error, they may delay visible failure, even as residual statistics drift.

For this reason, residual energy is a critical early-warning signal. It captures cumulative mismatch rather than pointwise error.

In practice, residual energy growth should be monitored alongside innovation normalization. The combination provides sensitivity to both abrupt and gradual degradation.

15.3 Detecting Learned Component Failure

Failures of learned components are rarely abrupt. They manifest as slow erosion of validity under distribution shift, sensor aging, or changes in operating regime.

Because learned models often produce confident outputs, their failure cannot be inferred from output magnitude alone.

Detection must therefore rely on estimator-level signals.

Three patterns are particularly indicative of learned component failure:

First, persistent inflation of innovation statistics after learning-based corrections are applied. If learning reduces raw residuals but increases normalized statistics, it is likely miscalibrated.

Second, divergence between error reduction and uncertainty calibration. When accuracy improves but covariance shrinks excessively, overconfidence is being introduced.

Third, temporal correlation in innovations. Repeated structure

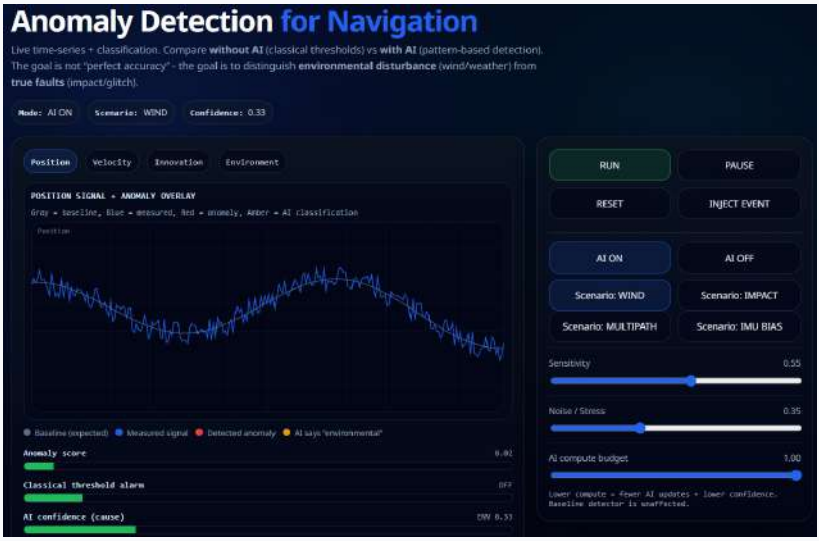


Figure 15.1: **Innovation-Based Anomaly Detection.** Measured navigation signals compared against expected estimator behavior. Smooth, correlated deviations indicate environmental disturbance, while impulsive mismatches suggest faults. Learning assists classification but does not replace innovation-based detection.

in $\boldsymbol{\nu}_k$ suggests hidden coupling often induced by shared learned models.

Detection strategies should focus on isolating the contribution of learning. This can be achieved by: temporarily disabling learned components, comparing residual statistics with and without learning, or monitoring innovation behavior conditional on learning activation.

Crucially, detection does not require ground truth. It relies entirely on internal consistency signals.

Learning that cannot be monitored through innovation and residual statistics cannot be trusted in deployment.

Figure 15.2: **Anomaly Classification and Confidence.** Classification confidence represents belief over anomaly cause rather than correctness. Overconfidence is mitigated by retaining residual statistics and estimator coherence checks as primary signals.

AI Implementation Notes

Always route AI effects through innovations and covariances. Monitor NIS and residual energy as primary runtime signals. If learning improves accuracy but degrades innovation statistics, disable it. A learned component that cannot be detected when it fails is unsafe.

16 Overconfidence Is the Real Enemy

In navigation systems, large errors are not the most dangerous failure mode. Overconfidence is. A system that knows it is uncertain can degrade gracefully. A system that is wrong while believing it is correct will fail silently and irreversibly.

Learning-based components tend to improve point estimates. At the same time, they often obscure or suppress uncertainty. This mismatch is subtle, difficult to detect externally, and devastating at scale.

This chapter explains why learning naturally hides uncertainty, how overconfidence manifests as a mismatch between error and covariance, and why rejecting confident but wrong outputs is a core requirement for reliable AI-enhanced navigation.

16.1 Why Learning Hides Uncertainty

Most learning models are trained to minimize a pointwise loss. They are optimized to predict the most likely output, not to represent uncertainty faithfully. Formally, a learned model often approximates

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y}} \; \mathbb{E}\big[\ell(\mathbf{y}, \mathbf{y}^{\text{true}})\big], \tag{16.1}$$

where uncertainty is implicit or ignored. Even when probabilistic outputs are used, the learned uncertainty reflects the training distribution, not the operational one. Out-of-distribution inputs therefore

produce confident but invalid predictions.

In navigation systems, this behavior is especially dangerous. Learning-based corrections may reduce instantaneous error while simultaneously invalidating uncertainty assumptions inside the estimator. The problem is not malicious. It is structural. Learning compresses information. Uncertainty, by definition, expands it. Without explicit architectural constraints, learning will always bias the system toward confidence.

16.2 Overconfidence as a Decision Failure

Overconfidence is not a failure of sensing, modeling, or learning. It is a failure of decision-making under uncertainty.

In structured navigation systems, detection of inconsistency often occurs before visible failure. Innovation statistics inflate. Residual energy trends upward. Internal coherence degrades. Yet many systems continue to act as if nothing is wrong.

This is the defining characteristic of overconfidence. The system has access to evidence that its assumptions are violated, but continues to trust its outputs and act on them.

From an architectural perspective, overconfidence occurs when estimates retain authority after the conditions that justified that authority have disappeared.

Learning-based components exacerbate this behavior. By improving pointwise accuracy, they delay visible degradation. By suppressing uncertainty, they reduce the urgency of response.

Overconfidence is therefore not a lack of information. It is the refusal to downgrade trust when information demands it. Reliable navigation systems are designed not only to detect inconsistency, but to change behavior when it appears. Without this transition, detection has no operational value.

16.3 Rejecting Confident Wrong Outputs

Reliable systems must be designed to reject confident but wrong information. This principle applies equally to physical sensors and to learned components. A confident measurement with invalid assumptions is more dangerous than a noisy one. Rejecting confident wrong outputs requires two capabilities.

First, the system must be able to detect overconfidence. This requires monitoring internal consistency signals, not just output magnitude or prediction variance. Second, the system must be willing to discard information. When learning-based outputs violate estimator expectations, they must be gated, downweighted, or disabled-even if they appear accurate. This behavior is counterintuitive. Engineers are often reluctant to discard high-quality predictions. In navigation systems, refusing such predictions is a sign of maturity, not weakness. The estimator must remain the final authority. Learning may suggest. Estimation decides. Systems that fail to reject confident wrong outputs do not fail because they lack intelligence, but because they lack discipline.

AI Implementation Notes

Assume learned outputs are overconfident by default. Never adjust state without adjusting covariance. Use innovation-based checks to detect confidence mismatch. If an AI output cannot be safely rejected at runtime, it is unsafe to deploy.

17 Recovery-Oriented Design

In many engineering disciplines, recovery is associated with physical failure. A component breaks, the system halts, and a recovery procedure is triggered. Navigation systems fail differently. In navigation, failure often occurs without any physical malfunction. Sensors continue to operate. Algorithms continue to produce outputs. Yet the system's internal belief drifts away from reality. For AI-enhanced navigation systems, this distinction is critical. Recovery must be designed not as a response to hardware failure, but as a response to loss of estimation coherence.

This chapter introduces recovery-oriented design as a core architectural principle and explains why recovery is fundamentally an estimation concept.

17.1 Recovery Without Physical Failure

In structured estimation systems, recovery is triggered when assumptions are violated, not when components stop functioning.

A navigation system may require recovery even when: sensor data continues to arrive, state estimates appear smooth, and accuracy metrics remain acceptable. The trigger for recovery is not error magnitude, but inconsistency.

Typical recovery triggers include: persistent innovation inflation, residual energy growth, or divergence between error behavior and uncertainty estimates. In AI-enhanced systems, recovery may be

required even when learning outputs remain confident. Indeed, confident learned outputs are often the cause of delayed recovery.

Recovery without physical failure therefore requires: continuous monitoring of estimator coherence, and explicit mechanisms for reverting or restructuring belief. This perspective shifts recovery from a rare exception to a normal operational mode.

17.2 Time-to-Restore Coherence

Traditional reliability metrics focus on time-to-failure. For navigation systems, time-to-recovery is often more important.

Time-to-restore coherence measures how quickly a system can return to a state where model assumptions, data, and uncertainty are aligned. This process may involve: disabling learned components, inflating uncertainty, reinitializing bias states, or switching to a conservative fallback mode. Formally, recovery is achieved when innovation statistics return to expected bounds and uncertainty once again reflects actual error. A system with fast recovery can tolerate frequent degradation. A system with slow recovery accumulates error and risk.

AI integration directly affects recovery time. Learning that is modular and reversible enables rapid recovery. Learning that is deeply entangled delays or prevents it. Recovery-oriented design therefore prioritizes: reversibility, locality of learned effects, and observability of internal state.

17.3 Why Recovery Is an Estimation Concept

Recovery is often treated as a control or operations problem. In navigation systems, it is fundamentally an estimation problem.

Recovery does not require knowing the true state. It requires restoring consistency between belief and uncertainty.

This distinction is essential. A system may recover without becoming accurate, as long as it becomes self-aware again.

Structured estimators support this process naturally. They expose internal signals that indicate when recovery has occurred.

End-to-end learning systems do not. Without explicit uncertainty and innovation signals, there is no principled notion of recovery.

For this reason, recovery-oriented design favors structured estimation architectures, even when augmented by learning.

Recovery is not about fixing the model. It is about restoring trust in the estimate.

AI Implementation Notes

Design recovery as a normal runtime mode, not an exception. Trigger recovery using estimator coherence signals, not output error. Ensure learned components can be disabled or weakened to accelerate recovery. If recovery cannot be defined without ground truth, the design is unsafe.

18 What This Enables (and What It Doesn't)

The integration of learning into navigation systems has reached a level of maturity that allows meaningful deployment in real-world systems. At the same time, persistent misconceptions continue to inflate expectations and obscure fundamental limitations.

This closing chapter clarifies what AI-enhanced navigation can realistically deliver, what remains unresolved at the engineering level, and which research directions are likely to matter in practice.

18.1 What AI-Enhanced Navigation Can Realistically Deliver

When integrated within structured estimation architectures, AI can significantly improve navigation performance.

Its most reliable contributions include: improved modeling of error processes, context-aware trust and uncertainty adaptation, and improved system behavior once degradation has been detected.

AI enables navigation systems to operate closer to their physical limits without sacrificing stability. It reduces conservatism where data supports it, and increases caution where uncertainty grows.

Crucially, AI enhances *system awareness*. It allows estimators to better understand when assumptions are violated, and to respond earlier than classical models alone would permit.

What AI does not deliver is certainty. It does not remove observability limits, eliminate drift, or replace the need for explicit uncertainty management.

AI-enhanced navigation is therefore not a new class of systems, but a refinement of existing ones.

18.2 Open Engineering Problems

Despite substantial progress, several engineering challenges remain open.

Robust modeling of correlation introduced by learned components is still poorly understood. Most deployed systems rely on conservative heuristics rather than principled solutions.

Runtime validation of learning under distribution shift remains an active challenge. Detecting when a learned component should be weakened or disabled without ground truth is difficult and system-dependent.

Finally, integrating learning across long time horizons without accumulating hidden coupling or overconfidence requires further architectural discipline.

These are not algorithmic gaps. They are system design problems.

18.3 Research Directions That Actually Matter

Future research in AI-enhanced navigation will be most impactful when it aligns with system-level needs.

Promising directions include: learning models that explicitly reason about uncertainty, architectures that expose internal failure signals, and recovery-aware learning that accelerates return to coherence.

Equally important is research that defines limits. Understanding when learning should *not* be used is as valuable as improving its performance.

Navigation systems evolve slowly for a reason. They operate under constraints that reward robustness over novelty.

The path forward is therefore not end-to-end learning, but deeper integration between learning and estimation.

Progress will come not from replacing structure, but from working within it.

18.4 Compute-Aware Navigation and Physical AI

As navigation systems increasingly incorporate learning-based components, computational constraints become a first-class design concern. Modern platforms operate under strict limits on power, latency, and thermal budget, particularly at the edge. Navigation architectures must therefore account not only for estimation accuracy, but also for the computational cost required to achieve it.

Compute-aware navigation treats inference as a resource that must be managed dynamically, rather than as a fixed-cost operation. The objective is not to maximize model fidelity at all times, but to allocate computational effort where it provides the greatest system-level benefit.

This leads naturally to the concept of adaptive inference. Different operational regimes impose different estimation demands. During nominal, steady-state motion, the navigation problem is often well-conditioned. In such regimes, lightweight learning models or reduced-order estimators are sufficient to maintain accuracy and consistency. Running high-capacity models continuously in these conditions wastes energy without improving reliability.

By contrast, during aggressive maneuvers, rapid dynamics, or partial sensor degradation, the estimation problem becomes more

challenging. Linearization error increases, observability may degrade, and uncertainty grows more rapidly. In these regimes, higher-fidelity models and more expensive inference may be justified, provided they remain integrated within the estimator's structure.

The key architectural requirement is that transitions between inference modes are driven by estimator-native signals rather than external heuristics. Changes in innovation statistics, covariance growth, or sensor availability should determine when additional computational resources are activated. Compute scaling must therefore be subordinate to estimation state, not the other way around.

From a system perspective, navigation efficiency can no longer be evaluated by accuracy alone. It must be understood as a trade-off between estimation quality and computational cost. Practical designs operate along a Pareto frontier in which accuracy, robustness, latency, and energy consumption are balanced explicitly.

Compute-aware navigation does not change the fundamentals of estimation. It reinforces them. By aligning inference complexity with estimation need, physical AI systems achieve improved endurance, predictable behavior, and graceful degradation under resource constraints. Learning remains a tool for improving performance, not a mandate to consume computation indiscriminately.

AI Implementation Notes

Use AI to reduce conservatism, not to eliminate uncertainty. Expect improved robustness and recovery, not guaranteed accuracy. Focus implementation effort on uncertainty, trust, and recovery signals. If a learned component cannot expose its limits, it is not ready for deployment.

The navigation engineer of 2026 is no longer just a filter designer; they are a Trust Architect. They must build systems that can explain their decisions, survive on limited edge power, and remain compliant

with global safety laws.

References - Part IV

1. Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley, 2001.
2. A. N. Bishop. Innovation-based fault detection in Kalman filters. *IEEE Control Systems Magazine*, 2007.
3. X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4), 2003.
4. S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.
5. R. Mehra. On the identification of variances and adaptive Kalman filtering. *IEEE Transactions on Automatic Control*, 15(2), 1970.
6. R. K. Mehra and J. Peschon. An innovations approach to fault detection and diagnosis in dynamic systems. *Automatica*, 7(5), 1971.
7. P. Maybeck. *Stochastic Models, Estimation, and Control*. Academic Press, 1979.
8. M. Basseville and I. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice Hall, 1993.