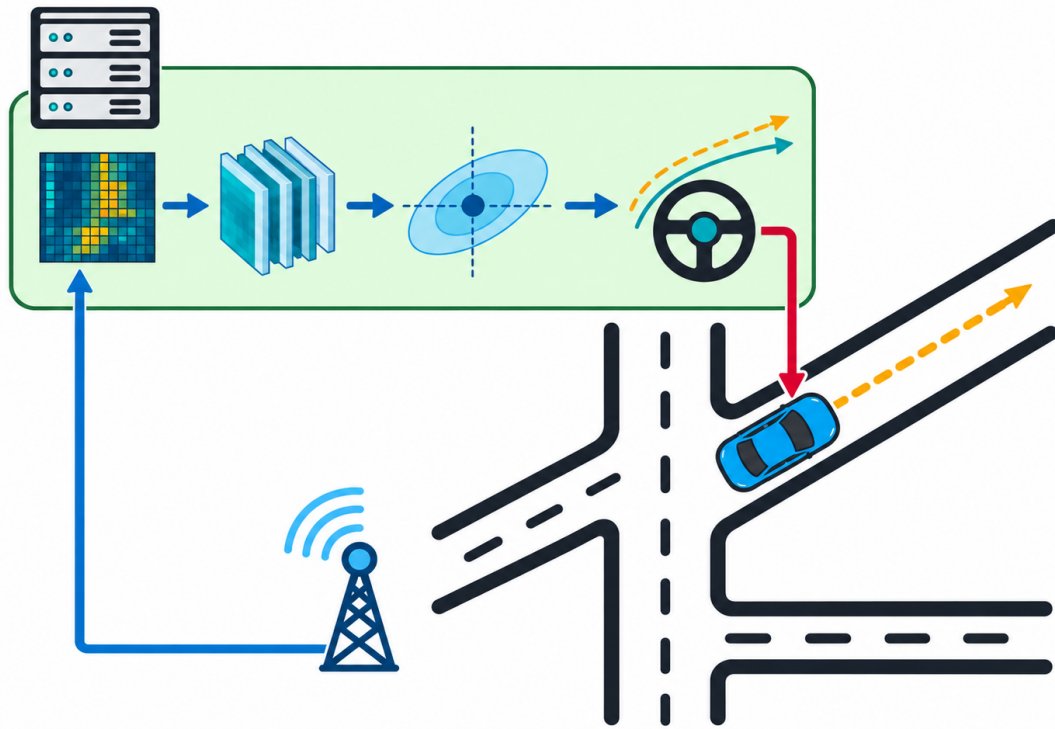




CHALMERS
UNIVERSITY OF TECHNOLOGY



Semantic Communication for Closed-loop Sensing-Communication-Control Systems

Master's thesis in Information and Communication Technology, MPICT

JINGCHENG WANG

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

MASTER'S THESIS 2026

Semantic Communication for Closed-loop Sensing-Communication-Control Systems

JINGCHENG WANG



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026

Semantic Communication for Closed-loop Sensing-Communication-Control Systems
JINGCHENG WANG

© JINGCHENG WANG, 2026.

Supervisor: Guangjin Pan, Department of Electrical Engineering
Examiner: Henk Wymeersch, Department of Electrical Engineering

Master's Thesis 2026
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Acknowledgements, dedications, and similar personal statements in this thesis, reflect the author's own views.

Cover: Concept illustration created by the author, depicting wireless sensing and feedback control.

Typeset in L^AT_EX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Abstract

This thesis investigates channel state information (CSI) as a sensing input for position estimation and simulated closed-loop control, motivated by the Level-2 interpretation of semantic communication. It evaluates task-relevant state extraction rather than a modeled semantic transmission link.

A simulation pipeline combines Sionna RT channel generation in a street-canyon scene, a ResNet-based estimator, Kalman filtering, and linear quadratic regulator (LQR) control. Complex CSI is represented by real and imaginary components across antennas and subcarriers. The neural model predicts two-dimensional position and coordinate-wise variance. Position predictions supply measurements to the filter, while the controller acts on the filtered state. The learned covariance is evaluated separately; the main closed-loop experiments use fixed diagonal covariance. On a random held-out test split of 2000 samples within the simulated scene, the estimator achieves a mean Euclidean localization error of 0.857 m, an RMSE of 1.204 m, and a median error of 0.673 m. Joint antenna-subcarrier masking yields an accuracy plateau: retaining 56.25% of the full CSI coefficients gives the lowest observed mean error, 0.626 m. This does not establish bitrate or energy savings. Predicted covariance is weakly correlated with localization errors and fails the reported consistency check, limiting its direct use in Kalman filtering.

The fixed-covariance sweep demonstrates tracking sensitivity to measurement confidence. In one paired seed-0 rollout per setting, a variance of 1 m^2 per coordinate yields the lowest true-state LQR cost, 750.27, and tracking RMSE, 0.683 m, among the tested settings. These findings characterize within-scene localization, uncertainty limitations, and covariance-sensitive tracking. Ideal interfaces, one scene and trajectory, and missing controller controls restrict conclusions about communication benefits, generalization, and the contribution of individual pipeline components. Calibrated uncertainty and broader evaluation are needed before deployment.

Keywords: Semantic Communication, Localization, Uncertainty Calibration, Kalman Filtering, LQR Control.

Acknowledgements

I would like to express my sincere gratitude to my examiner, Henk Wymeersch, and my supervisor, Guangjin Pan. I am deeply grateful to them for offering me the opportunity to carry out this master's thesis, for proposing and shaping the thesis topic, and for their generous support and guidance throughout the project.

I would also like to thank all the professors and teaching assistants I have met during my studies at Chalmers University of Technology in Sweden, whose teaching, feedback, and encouragement have helped me grow both academically and personally. I am equally thankful to my friends and classmates: Weirui Zhao, Yizhou Wang, Ci Gao, Zhe Du, Zhenlin Sun, Zile Ou, Tingting Zhao, Wei Zhang and all the other friends I met in Sweden, for their companionship, support, and many memorable moments during this journey.

Finally, I would like to thank everyone who has supported me along the way. This thesis marks not only the completion of a research project, but also an important chapter of my study and life at Chalmers.

Jingcheng Wang, Gothenburg, 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis, listed in alphabetical order:

AI	Artificial Intelligence
ANEES	Average Normalized Estimation Error Squared
BS	Base Station
CDF	Cumulative Distribution Function
CSI	Channel State Information
CUDA	Compute Unified Device Architecture
GPU	Graphics Processing Unit
ISAC	Integrated Sensing and Communication
KF	Kalman Filter
L-2	Level-2 semantic communication
L-3	Level-3 semantic communication
LQR	Linear Quadratic Regulator
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MIMO	Multiple-Input Multiple-Output
MSE	Mean Squared Error
NEES	Normalized Estimation Error Squared
NIS	Normalized Innovation Squared
NLL	Negative Log-Likelihood
OFDM	Orthogonal Frequency-Division Multiplexing
P90	90th percentile
ReLU	Rectified Linear Unit
ResNet	Residual Network
RMSE	Root Mean Squared Error
RSS	Received Signal Strength
RT	Ray Tracing
UE	User Equipment

Nomenclature

Below is the nomenclature of indices, sets, parameters, variables, and evaluation measures used throughout this thesis.

Indices

i	Sample index in the generated or held-out dataset
k	Discrete control time-step index in the closed-loop trajectory
t	Backward Riccati-recursion index for the finite-horizon LQR controller
m	Antenna-element index
n	Subcarrier index
ℓ	Propagation-path index in the ray-tracing channel model
d	Coordinate index, usually x or y
a	Discrete sensing-resource action index

Sets and Domains

$\mathcal{D}$	Dataset of CSI tensors and corresponding position labels
$\mathcal{B}$	Mini-batch of training samples
$\mathcal{E}$	Ray-tracing environment or propagation scene
$\mathcal{M}_a$	Selected antenna-index set for a resource action
$\mathcal{M}_f$	Selected subcarrier-index set for a resource action
$\{1, \dots, A\}$	Set of available sensing-resource actions
$\mathbb{R}$	Set of real-valued quantities
$\mathbb{C}$	Set of complex-valued quantities

Parameters

M	Number of antenna elements in the full CSI tensor
N	Number of subcarriers, or the number of evaluation samples when used in metric definitions
N_s	Number of supervised samples used in the generic learning formulation
K	Number of generated samples in the method chapter dataset notation
A	Number of discrete sensing-resource actions
K_a^a	Number of active antenna elements selected by resource action a
K_a^f	Number of active subcarriers selected by resource action a
ρ_a	Fraction of retained antenna-subcarrier coefficients under resource action a
f_n	Frequency of subcarrier n
$L(\mathbf{p})$	Number of propagation paths for UE position $\mathbf{p}$
E_0	Reference energy used in sample-wise CSI normalization
γ_i	Sample-wise CSI normalization factor
Δt	Discrete simulation time step
$\mathbf{s}_p$	Position scale used for label normalization and variance conversion
$\mathbf{u}_{\min}, \mathbf{u}_{\max}$	Lower and upper bounds used to clip the acceleration command
$\mathbf{Q}$	Process-noise covariance in the Kalman filter
$\mathbf{Q}_{\text{lqr}}$	LQR state-tracking cost matrix
$\mathbf{R}_{\text{lqr}}$	LQR control-effort cost matrix

Channel and Learning Variables

$\mathbf{p}_i$	Three-dimensional UE position of sample i
$\mathbf{p}_k$	Horizontal UE or agent position at control step k
x_i, y_i, z_i	Cartesian components of sample position $\mathbf{p}_i$
$x_{\text{BS1}}, y_{\text{BS1}}$	Horizontal coordinates of the first base station
$H_{m,n}$	Complex channel coefficient at antenna element m and subcarrier n
$\mathbf{H}_i$	Complex frequency-domain CSI matrix for sample i
$\mathbf{H}_k$	CSI observation generated at closed-loop control step k
$\mathbf{X}_i$	Real-valued two-channel CSI tensor formed from real and imaginary channel components
$\widetilde{\mathbf{X}}_i$	Normalized CSI tensor before resource masking

$\bar{\mathbf{X}}_i$	Normalized and resource-masked CSI tensor passed to the neural estimator
$\mathbf{y}_i$	Two-dimensional relative UE position label with respect to BS1
$\tilde{\mathbf{y}}_i$	Normalized position label
$\hat{\mathbf{y}}_i$	Neural-network prediction of the relative UE position
$\hat{\mathbf{y}}_i^{\text{phys}}$	Predicted position converted back to physical units
$\hat{\mathbf{y}}_i^{\text{fc}}$	Position prediction produced by the fully connected baseline estimator
$\hat{\mathbf{Y}}$	Batch of predicted relative positions
$\hat{\mathbf{l}}_i$	Predicted log-variance vector for sample i
$\hat{\sigma}_i^2$	Predicted coordinate-wise variance vector
$\hat{\mathbf{V}}$	Batch of predicted coordinate-wise variances
$\mathbf{r}_i$	Feature vector extracted by the adapted ResNet backbone
$g_\theta(\cdot)$	Adapted residual-network feature extractor with parameters θ
$f_\theta(\cdot)$	Neural localization estimator mapping CSI to position and uncertainty outputs
$h_\phi(\cdot)$	Fully connected baseline estimator with parameters ϕ
$W_\mu, \mathbf{b}_\mu$	Weight matrix and bias vector of the position head
$W_\sigma, \mathbf{b}_\sigma$	Weight matrix and bias vector of the log-variance head
$\Phi_{\text{RT}}(\cdot)$	Ray-tracing channel-generation mapping from position to CSI

State Estimation and Control Variables

$\mathbf{x}_k$	Four-dimensional dynamic state $[x_k, y_k, v_{x,k}, v_{y,k}]^T$
$\hat{\mathbf{x}}_{k k-1}$	Predicted Kalman-filter state before incorporating the measurement at step k
$\hat{\mathbf{x}}_{k k}$	Filtered Kalman-filter state after incorporating the measurement at step k
$\mathbf{x}_k^{\text{ref}}$	Reference state tracked by the controller
$\mathbf{x}_k^{\text{true}}$	Simulated true state used for closed-loop evaluation
$\mathbf{u}_k$	Two-dimensional acceleration control input
$\mathbf{z}_k$	CSI-based position measurement supplied to the Kalman filter
$\mathbf{w}_k$	Process noise in the state-transition model
$\mathbf{v}_k$	Measurement noise in the position-observation model
$\mathbf{F}$	Discrete-time state-transition matrix

$\mathbf{B}$	Control-input matrix
$\mathbf{H}$	Linear measurement matrix selecting the position components of the state
$\mathbf{P}_{k k-1}$	Predicted state-estimation covariance
$\mathbf{P}_{k k}$	Updated state-estimation covariance
$\mathbf{R}_k^{\text{KF}}, \mathbf{R}_k$	Measurement covariance used by the Kalman filter
$\mathbf{R}_k^{\text{learned}}$	Candidate learned covariance in physical units, evaluated offline
$\mathbf{K}_k$	Kalman gain or, in the LQR section, the time-varying feedback gain; the meaning is determined by context
$\mathbf{P}_t$	Riccati matrix used to compute the finite-horizon LQR feedback gain
$\mathbf{e}_k$	LQR tracking-error state based on the filtered estimate and the reference state
J	Finite-horizon LQR objective
c_k	Instantaneous LQR cost

Losses and Evaluation Measures

$\mathcal{L}_{\text{MSE}}$	Mean squared error loss in the generic localization formulation
$\mathcal{L}_{\text{NLL}}$	Gaussian negative log-likelihood loss in the generic uncertainty formulation
$\mathcal{L}_{\text{pos}}$	Position-regression loss used in the first training stage
$\mathcal{L}_{\text{var}}$	Variance-head negative log-likelihood loss used in the second training stage
e_i	Two-dimensional localization error for evaluation sample i
$e_{i,d}$	Coordinate-wise normalized localization error
d_i	Euclidean localization error expressed in physical units
e_k^{track}	True trajectory tracking error at control step k
MAE	Mean absolute localization error
RMSE	Root mean squared localization error
$\text{RMSE}_{\text{track}}$	Root mean squared tracking error over a closed-loop trajectory
Σ_i	Predicted diagonal covariance used for NEES evaluation
NEES_i	Normalized estimation error squared for sample i
ANEES	Average normalized estimation error squared over all evaluation samples

Contents

List of Acronyms	ix
Nomenclature	xi
List of Figures	xvii
List of Tables	xix
1 Introduction	1
1.1 Overview	1
1.1.1 Problem Context	1
1.1.2 Motivation	1
1.1.3 Limitations of Existing Approaches	2
1.1.4 Thesis Approach	4
1.1.5 Research Questions	4
1.1.6 Contributions	5
1.2 Related Work	6
2 Preliminaries	9
2.1 Problem Setting	9
2.2 Wireless Channel State Information	10
2.3 Ray-Tracing-Based Channel Simulation Using Sionna	11
2.4 Learning-Based Position Estimation Model	12
2.5 Closed-Loop Sensing-Communication-Control	13
3 Method	17
3.1 Method Overview	17
3.2 Channel and Dataset Generation	18
3.3 Model Building and Training	19
3.3.1 CSI Representation and Preprocessing	20
3.3.2 Neural Position Estimator	22
3.3.3 Structural Baseline Estimator	23
3.3.4 Training Objective	25
3.4 Closed-loop Control Method	27
3.4.1 Kalman Filter	28
3.4.2 LQR Control	29
3.4.3 Feedback Loop	30

4	Experimental Evaluation	33
4.1	Evaluation Goals	33
4.2	Datasets	34
4.3	Evaluation Measures	36
4.4	Implementation Details	38
4.5	Experiments	40
4.5.1	Localization Accuracy	40
4.5.2	Spatial Error Analysis	42
4.5.3	Uncertainty Analysis	43
4.5.4	Sensitivity to Sensing-Resource Allocation	45
4.5.5	Closed-loop Control and Measurement-Variance Sensitivity	47
4.6	Societal, Ethical, and Sustainability Considerations	49
5	Conclusion and Future Work	53
5.1	Conclusion	53
5.2	Limitations	54
5.3	Future Work	55
	Bibliography	59
A	Detailed Experimental Configuration	I
A.1	Channel Data and Preprocessing	I
A.2	Split, Training, and Checkpoint Selection	II
A.3	Evaluation and Plotting	IV
A.4	Closed-loop Configuration and Provenance	IV
B	Code and Data Reproduction Guide	VII
B.1	Repository, Environment, and Required Artifacts	VII
B.2	Dataset and Split Preparation	VIII
B.3	Offline Evaluation and Figure Generation	VIII
B.4	Retraining and Control Simulations	X

List of Figures

3.1	Simulated feedback loop for CSI-based tracking	18
3.2	Offline data-generation and training workflow	24
4.1	Empirical CDF of the two-dimensional localization error on the held-out test split. The curve shows the fraction of test samples whose localization error is below a given threshold.	42
4.2	Empirical localization-error CDFs for the ResNet and FCNN on the same held-out positions and resource masks. The horizontal axis is logarithmic.	43
4.3	Spatial map of mean localization error	44
4.4	Spatial map of predicted total variance	45
4.5	NEES diagnostics on the held-out test split	46
4.6	Localization error across CSI resource configurations	47
4.7	ANEES across CSI resource configurations	48
4.8	Closed-loop trajectory at the lowest observed tracking RMSE	49
4.9	True-state LQR cost across fixed covariances	50
4.10	Tracking RMSE across fixed covariances	51

List of Tables

4.1	Dataset configuration used in the experimental evaluation.	35
4.2	Main settings for the reported ResNet training and control implementation.	39
4.3	Localization accuracy of the trained CSI-based position estimator on the held-out test split.	41
4.4	Localization comparison on the same 2000 held-out test positions and resource masks. All errors are in metres. The models use unequal training budgets and one training run each.	41
4.5	Uncertainty diagnostics for the variance head on the held-out test split.	44
4.6	Localization accuracy and uncertainty calibration for the eight sensing-resource configurations, evaluated on the same held-out test split of 2000 samples with a single trained model.	46
4.7	Fixed-covariance closed-loop performance	48
A.1	Settings in the inspected channel generator. These values document the code configuration; a historical generator-version manifest is unavailable.	II
B.1	Mapping between thesis evidence and reproduction entry points. . . .	IX

1

Introduction

1.1 Overview

1.1.1 Problem Context

Integrated sensing and communication (ISAC) is a key direction for future wireless systems, where the same wireless infrastructure is expected to support both data transmission and environment sensing. Instead of treating communication and sensing as two independent functions, ISAC aims to exploit the coupling between radio signals, propagation channels, and the physical state of the environment. This is particularly relevant for autonomous and cyber-physical systems, where wireless networks may need to sense the state of a target, exchange task-relevant information, and support control decisions in a closed loop.

This thesis considers an ISAC-inspired simulation of CSI-based position estimation and closed-loop control. A ray tracer generates the channel frequency response associated with the target position. The CSI is supplied directly to a neural estimator, whose position output is passed to a Kalman filter. An LQR controller computes the control input from the filtered state, and that input is applied to the simulated target dynamics. These modules exchange data through ideal interfaces. The implementation does not include a sensing-node encoder, a transmitted compressed message, a separate receiver decoder, or a modeled transport link for measurements and commands.

Channel state information (CSI) provides a natural observation modality for this setting. The wireless channel depends on the position of the target, the geometry of the environment, the antenna configuration, and the propagation paths between transmitter and receiver. As a result, CSI contains information about the spatial state of the target. The central problem studied in this thesis is therefore how to use CSI as a sensing observation for position estimation in an ISAC-inspired closed-loop simulation.

1.1.2 Motivation

Classical communication systems are commonly designed around signal reconstruction or bit-level reliability. However, in many closed-loop applications, the receiver does not need a perfect reconstruction of the raw sensing observation. Instead, it needs the information that is relevant for a downstream task, such as estimating the target position or choosing a control action. This motivates a semantic communication perspective, where the transmitted representation is optimized according to the

meaning it has for the task rather than according to reconstruction fidelity alone. The project description separates semantic compression in a closed-loop system into three levels. Level 1 focuses on reconstructing the observation, Level 2 focuses on estimating the state, and Level 3 focuses directly on control. The latter two are abbreviated L-2 and L-3 in the remainder of this thesis. The present work uses L-2 as a functional interpretation of extracting position information from CSI rather than reconstructing the observation. It implements the state-estimation and downstream-control parts of that interpretation, with ideal data exchange between modules. A learned communication encoder–decoder and direct L-3 optimization of the final control objective are outside the implemented scope.

Studying CSI-based state estimation is valuable because measurement quality and the confidence assigned to measurements affect feedback control. This thesis evaluates localization under selected CSI inputs and tracking under fixed measurement covariances. These experiments provide a preliminary task-level sensing study motivated by L-2 semantic communication; they do not establish communication efficiency, the effect of message compression, or control gains over alternative estimators and controllers.

1.1.3 Limitations of Existing Approaches

Research on semantic and task-oriented communication, learned CSI representations, integrated sensing and communication, and wireless networked control, which is reviewed in detail in Section 1.2, shows substantial progress toward communication systems that are aware of tasks, context, and physical environments. Nevertheless, the existing methods remain only partially aligned with the closed-loop sensing-communication-control problem considered in this thesis. This subsection states the five gaps that motivate the thesis; the individual works behind each gap are discussed in Section 1.2. A first limitation concerns the objective used in many semantic and task-oriented communication studies. Foundational and survey works argue that future systems need to move beyond bit-level reliability and optimize communication according to meaning or task utility [2, 3, 4, 5]. Concrete neural semantic communication systems further show that learned transceivers can improve semantic recovery or source reconstruction under channel impairments [14, 7]. However, these objectives are usually formulated around reconstructing a message, preserving semantic content for human interpretation, or improving inference accuracy at the receiver. They do not directly address the case where the useful semantic variable is a physical state estimate whose error propagates through a Kalman filter and changes the cost of a feedback controller.

A second limitation is that much of the CSI representation-learning literature is reconstruction-oriented. For example, CsiNet demonstrates that neural networks can exploit channel structure to compress and recover massive-MIMO CSI more effectively than conventional compressive-sensing baselines [8]. This is important for reducing feedback overhead in high-dimensional antenna systems, whose potential and practical challenges are well established in massive-MIMO studies [16, 17]. Yet a low channel-reconstruction error is not necessarily equivalent to preserving the information needed for localization or control. Components of the channel that

dominate reconstruction metrics may be weakly related to the target position, while small errors in spatially informative components can have a disproportionate effect on state estimation. Therefore, reconstruction quality alone is an incomplete criterion for the CSI-based state-estimation task studied in this thesis.

A third limitation appears in CSI-based localization. Classical and modern wireless-positioning studies show that radio measurements can carry strong spatial information, especially when wide bandwidths and multiple antennas provide time, angle, and frequency selectivity [21, 22]. Deep CSI fingerprinting methods further demonstrate that neural networks can learn useful location-dependent features from CSI amplitude and phase [25, 26, 11]. Channel charting extends this idea by learning low-dimensional radio maps from CSI and reducing the dependence on explicit position labels [28]. These methods are directly relevant because they establish CSI as a useful sensing modality. Their limitation for this thesis is that localization is normally evaluated as a standalone prediction problem. The estimator is rarely placed inside a dynamic feedback loop, and the reported localization error is not translated into tracking error, control effort, or accumulated LQR cost.

A fourth limitation concerns the separation between ISAC signal processing and control. ISAC surveys and joint communication-radar signal-processing overviews clarify why future wireless systems benefit from reusing spectrum, waveforms, and infrastructure for both communication and sensing [9, 10]. However, these works usually organize the problem around sensing metrics, communication metrics, waveform design, beamforming, and receiver processing. The downstream controller is often outside the main evaluation loop. Conversely, wireless networked control studies analyze how packet loss, delay, scheduling, and data-rate constraints influence stability and control performance [38, 39, 13]. In that line of work, the wireless link is mainly treated as a constrained transport channel between sensors, controllers, and actuators. The channel itself is not typically used as the observation from which the controlled object’s position is inferred.

Finally, existing approaches often do not make uncertainty a first-class link between learning-based sensing and control. Kalman filtering provides a principled mechanism for combining a predicted state with a noisy measurement through covariance propagation [37]. In a learned CSI-based localization system, however, the measurement covariance cannot be assumed to be known a priori: it depends on the environment, the selected antenna and subcarrier resources, and the estimator’s own errors. If this covariance is underestimated, the filter may over-trust noisy neural measurements; if it is overestimated, the filter may underuse informative CSI observations. This thesis therefore focuses on a narrower but concrete gap: learning CSI-to-position estimates in a sensing and control simulation motivated by L-2 semantic communication and evaluating localization uncertainty offline and the effect of fixed measurement covariance on Kalman-filtered LQR tracking. The contribution connects simulated wireless sensing observations to state estimation and covariance-sensitive tracking; communication transport and its costs are outside the evaluation.

1.1.4 Thesis Approach

This thesis develops and evaluates a CSI-to-position estimation pipeline for a simulated closed-loop ISAC scenario. A ray-tracing environment generates channel observations for known target positions. The CSI is represented as real and imaginary components over antenna and subcarrier dimensions, and a ResNet-based estimator predicts the two-dimensional position and coordinate-wise variance. The position estimates are used as measurements in a Kalman filter, whose filtered state drives an LQR controller. The variance output is evaluated separately for statistical consistency and informativeness. Because the reported learned covariance is poorly calibrated, the main closed-loop experiments use fixed diagonal measurement covariances to study how measurement confidence affects tracking and control cost. The main focus is CSI-based state estimation motivated by the L-2 interpretation. The thesis studies the relationship between wireless channel observations, learned CSI features, localization accuracy, and the information retained by antenna-subcarrier masks. It also evaluates predicted uncertainty, because the covariance assigned to a position measurement changes the behavior of the Kalman filter and controller. The main closed-loop experiments use fixed diagonal measurement covariances; the learned variance is assessed separately and is not demonstrated as a reliable closed-loop confidence signal.

The implementation builds on Sionna for channel generation and Python-based deep learning tools for model training and evaluation. The channel generation step provides controlled CSI observations from known target positions, while the learning step maps these observations to position estimates. This makes it possible to evaluate the localization performance of the proposed approach, to analyze how different antenna and subcarrier resource choices affect the quality of the estimated state, and to measure how fixed covariance choices affect closed-loop tracking error and true control cost.

1.1.5 Research Questions

This thesis investigates CSI-based position estimation and its use in a simulated Kalman-filter and LQR control pipeline. The evaluation focuses on a single ray-traced street-canyon scene, with a random held-out split for offline localization and the specified simulation conditions for closed-loop tracking. The study addresses four research questions:

1. **RQ1 – Localization accuracy:** How accurately can the CSI-based neural estimator localize a UE within the simulated street-canyon scene, and how do localization errors vary across the evaluated area?
2. **RQ2 – CSI availability:** How does localization accuracy change across jointly selected antenna and subcarrier configurations under the adopted normalization and masking procedure?
3. **RQ3 – Uncertainty quality:** To what extent is the predicted diagonal position covariance statistically consistent with the observed localization errors, and how informative is it about spatial and resource-dependent variations in those errors?
4. **RQ4 – Closed-loop sensitivity:** How does the choice of fixed measurement

covariance affect tracking performance and true control cost in the simulated CSI-based Kalman-filter and LQR loop, and how does the estimated control cost compare with the true cost?

These questions address localization, predicted uncertainty, and feedback tracking within the evaluated scene. The resource study measures the effect of retaining subsets of CSI coefficients, with antenna and subcarrier counts varied jointly. It does not separate their individual effects or establish reductions in transmitted bitrate, latency, or energy consumption. An FCNN localization comparison uses the same test inputs, but unequal training budgets and one run per architecture limit attribution of the observed difference to architecture. Oracle, unfiltered, and model-only control comparisons are not reported, so the evaluation does not establish gains over those control alternatives.

1.1.6 Contributions

The contributions are empirical findings obtained using the ray-tracing, neural-estimation, filtering, and control pipeline. Each finding addresses one of the research questions within the scope of the reported experiments:

- **Characterization of within-scene localization accuracy (RQ1).** On the random held-out split of 2000 samples, with resource masks sampled from the eight configurations, the estimator achieves a mean Euclidean localization error of 0.857 m and an RMSE of 1.204 m (Table 4.3). The spatial analysis and error CDF reveal nonuniform accuracy and a sparse tail reaching 20.7 m (Figs. 4.3 and 4.1). The FCNN localization baseline has a mean Euclidean error of 2.033 m on the same test positions and resource masks (Table 4.4); unequal training budgets and one run per model limit architectural attribution. These findings describe interpolation within the simulated scene rather than generalization to new regions or environments.
- **Characterization of the accuracy plateau under joint CSI masking (RQ2).** Evaluation of the same test samples across eight joint antenna-subcarrier masks shows diminishing accuracy gains at larger configurations (Table 4.6). The 24×300 configuration retains 56.25% of the coefficients in the full 32×400 CSI tensor and yields the lowest observed mean Euclidean error, 0.626 m. Small differences between neighboring configurations do not establish a uniquely optimal allocation. The finding applies to the adopted normalization and masking procedure.
- **Evidence of limitations in learned position covariance (RQ3).** The learned covariance fails the reported ANEES consistency check: an ANEES of 1.351 lies below the interval $[1.913, 2.089]$ under the assumed two-dimensional Gaussian error model (Table 4.5). Weak spatial correlations and nearly constant predicted standard deviation across resource configurations further limit its value as a relative confidence signal (Tables 4.5 and 4.6). This negative result motivates calibration before direct use as a Kalman-filter measurement covariance.
- **Quantification of closed-loop sensitivity to fixed covariance (RQ4).** The fixed-variance sweep shows that tracking performance depends on the

covariance supplied to the Kalman filter. In a single paired seed-0 rollout per setting, a variance of 1 m^2 per position coordinate yields the lowest true LQR cost, 750.27, and tracking RMSE, 0.683 m, among the tested values (Table 4.7). The sweep also shows that estimated cost can decrease while true cost increases, so the controller’s internal cost alone is insufficient to assess realized tracking. These observations do not establish a generally optimal covariance setting.

1.2 Related Work

Semantic and task-oriented communication. Semantic communication revisits the classical communication objective by asking which information is useful for the receiver’s task, rather than only whether bits or source samples are reconstructed accurately. Calvanese Strinati and Barbarossa frame this shift as a move beyond Shannon-style reliable transmission toward semantic and goal-oriented communication for 6G systems [2]. Broader surveys by Luo et al. and Yang et al. organize semantic communication around semantic representation, knowledge bases, learning-based transceivers, and application domains [3, 4]. Gündüz et al. further emphasize that context and task utility belong in the communication design objective [5]. Xie et al. give one concrete deep-learning semantic communication architecture, showing how a transmitter and receiver can be optimized around semantic recovery rather than symbol recovery [14]. Task-oriented edge-inference systems, such as the multi-device cooperative inference approach of Shao et al., also demonstrate that communication resources can be assigned according to inference value [6]. Deep joint source-channel coding provides another important learning-based communication baseline because it optimizes the physical-layer transmission of images end to end [7]. These works motivate the L-2 view adopted in this thesis: the relevant semantic content is not a reconstructed CSI tensor or image, but the position information needed by the estimator and controller. This motivation is implemented as CSI-based position estimation and evaluation within a simulated control loop. Unlike the cited learned transceivers, the implementation does not transmit a learned representation over a modeled communication link.

Integrated sensing and communication and channel modeling. ISAC provides the wireless-system context in which the same radio infrastructure supports both communication and environmental sensing. Liu et al. survey ISAC as a dual-functional wireless-network paradigm for 6G, while Zhang et al. review signal-processing techniques for joint communication and radar sensing [9, 10]. These works establish the relevance of radio signals as both information carriers and sensing probes, but they usually focus on waveform, beamforming, radar processing, or network design rather than on learned CSI-to-state mappings inside a feedback-control loop. The channel observations used in this thesis are naturally connected to OFDM and massive MIMO, where frequency selectivity and large antenna arrays provide high-dimensional spatial information [15, 16, 17]. Standardized channel models such as 3GPP TR 38.901 define geometry, antenna, path-loss, and spatial-consistency assumptions for frequencies from 0.5 to 100 GHz [18]. Ray-tracing datasets and simulators make this geometry-specific channel structure usable for

machine-learning experiments. DeepMIMO provides a ray-tracing-based dataset framework for mmWave and massive-MIMO learning tasks [12], and Sionna RT offers a differentiable ray tracer for radio propagation modeling [19, 20]. Since the implementation in this thesis uses Sionna RT to generate complex channel frequency responses in a street-canyon scene, these works are directly related to the data-generation pipeline. The difference is that the generated channel is not used for beam prediction or channel reconstruction, but as the input to a state estimator for closed-loop control.

CSI-based localization and channel representations. Wireless localization has a long signal-processing history based on received signal strength, time of arrival, angle of arrival, and statistical estimation bounds [21]. Modern cellular and mmWave systems strengthen this connection because wide bandwidths and large antenna arrays make the channel more informative for positioning, especially in vehicular and infrastructure-supported scenarios [22]. CSI fingerprinting uses the spatial and frequency selectivity of the channel as a location-dependent feature. FIFS is an early example showing that fine-grained CSI can outperform coarser RSS fingerprints by exploiting subcarrier and antenna diversity [23]. DeepFi and its journal extension use deep networks to learn location fingerprints from CSI amplitude and phase [24, 25]. PhaseFi focuses on calibrated CSI phase and shows that phase information can also support deep-learning-based indoor localization [26]. Li et al. combine deep learning with subcarrier filtering in a multi-level CSI fingerprinting scheme [27]. Hsieh et al. further evaluate deep learning with RSS and CSI for indoor localization [11]. Channel charting takes a related but more self-supervised direction by learning a low-dimensional radio map from CSI without requiring explicit position labels for every sample [28]. These localization studies are central to the thesis because they show that CSI contains rich position information. However, most of them stop at localization accuracy. This thesis extends the viewpoint by asking how the learned CSI-to-position estimate and its measurement uncertainty affect Kalman filtering and LQR tracking in a dynamic loop.

Learning architectures for high-dimensional CSI. The CSI tensor in this thesis has real and imaginary components over antenna and subcarrier dimensions, which makes the estimator a structured high-dimensional regression problem. Fully connected networks provide a simple baseline but ignore much of the local structure across antennas and subcarriers. Convolutional models are attractive because adjacent antennas and neighboring subcarriers often contain correlated channel information, and residual networks make deeper convolutional feature extractors easier to optimize [29]. Sequence models provide another possible view of CSI, where subcarriers can be processed as an ordered sequence; LSTM networks were introduced to address long-range dependencies in sequential data [30]. Attention-based models and Transformers provide a more flexible mechanism for relating distant tokens [31]. Vision Transformers show how patch-based tokenization can be used for two-dimensional structured inputs [32]. The reported localization experiments compare the residual convolutional estimator with a fully connected baseline using the same test-input protocol. Unequal training budgets and one run per model limit conclusions about the cause and repeatability of the observed difference. Recurrent and Transformer models remain alternatives for future comparison; no results for those

architectures are reported.

Uncertainty-aware learning and filtering. Closed-loop control is sensitive not only to the mean position estimate but also to the confidence assigned to that estimate. Bayesian and approximate Bayesian deep-learning methods address this by estimating predictive uncertainty. Gal and Ghahramani interpret dropout as approximate Bayesian inference and use it to extract model uncertainty from neural networks [33]. Kendall and Gal distinguish epistemic and aleatoric uncertainty and show how input-dependent observation noise can be learned for regression-like perception tasks [34]. Deep ensembles provide a simple and scalable alternative for predictive uncertainty estimation [35], and calibration studies show that modern neural networks can be overconfident unless their predictive distributions are checked or adjusted [36]. These ideas motivate evaluating the learned variance as a candidate Kalman-filter measurement covariance; the main closed-loop experiments instead use fixed covariances because the learned output fails the reported consistency check. Kalman’s original filtering framework shows how prediction, measurement update, and covariance propagation can be combined for optimal linear state estimation under Gaussian assumptions [37]. In the present work, the neural network supplies the position measurement. Its variance head implements a candidate uncertainty interface evaluated offline; the reported feedback experiments supply fixed measurement covariance separately. The variance-sweep experiments therefore test a practical question raised by uncertainty-aware learning: if the covariance passed to the filter is too small or too large, the controller may either overreact to noisy measurements or underuse informative observations.

Wireless control and closed-loop objectives. Networked control studies make clear that communication decisions can change stability and control cost. Tatikonda and Mitter analyze control under communication constraints and characterize rate requirements for different control objectives [38]. Nair et al. survey feedback control under data-rate constraints, connecting information flow to stabilization and performance [39]. Park et al. review wireless network design for control systems, including reliability, delay, scheduling, and co-design issues [13]. These works are highly relevant because they shift evaluation from communication fidelity to control performance. However, they usually consider communication as the link between sensors, controllers, and actuators, whereas this thesis uses the wireless channel itself as the sensing source. Recent work on active-inference-enabled closed-loop ISAC moves closer to the present motivation by studying sensing, communication, and control over long horizons [1]. This thesis remains more modest: it does not solve the full joint L-3 control-oriented semantic communication problem. Instead, it implements the state-extraction part of the L-2 motivation by learning position from CSI and evaluating covariance sensitivity inside a Kalman-filtered LQR simulation. The physical propagation model supplies sensing features; measurement and command transport constraints are not modeled.

2

Preliminaries

This chapter introduces the technical background required to understand the thesis method and experiments. The goal is to make the later method chapter readable without mixing problem formulation, wireless-channel notation, learning objectives, and feedback-control equations in one place. The chapter first defines the closed-loop localization problem, then introduces CSI as the wireless sensing observation, explains the ray-tracing channel model used to generate data, describes learning-based position estimation from CSI, and finally summarizes the Kalman-filtered LQR loop used to connect estimation quality to control performance.

2.1 Problem Setting

This thesis studies CSI-based state estimation in a simulated closed loop. The target can be interpreted as a vehicle or mobile user equipment (UE), and a ray-traced wireless channel supplies the sensing observation. At each time step, the target has a physical state that includes its planar position and, for the control loop, its velocity. The state is written as

$$\mathbf{x}_k = [p_{x,k}, p_{y,k}, v_{x,k}, v_{y,k}]^T, \quad (2.1)$$

where $p_{x,k}$ and $p_{y,k}$ denote the two-dimensional position of the target, and $v_{x,k}$ and $v_{y,k}$ denote its velocity components at time step k .

The wireless observation is the channel state information (CSI) associated with the target position. In the setting considered here, the CSI is represented by a complex channel frequency response over multiple antenna elements and OFDM subcarriers. For a target position $\mathbf{p}_k = [p_{x,k}, p_{y,k}]^T$, the sensing process produces a channel observation

$$\mathbf{H}_k = g(\mathbf{p}_k, \mathcal{E}) + \mathbf{n}_k, \quad (2.2)$$

where $g(\cdot)$ denotes the environment-dependent wireless propagation mapping, $\mathcal{E}$ represents the fixed propagation environment, and $\mathbf{n}_k$ represents measurement noise and modeling uncertainty. The central difficulty is that $g(\cdot)$ is highly nonlinear because the received channel depends on geometry, multipath propagation, antenna layout, and frequency selectivity. As a result, the position is not directly observed; it must be inferred from the structure of the CSI.

The problem is motivated by the L-2 interpretation of semantic communication. Level 1 targets observation reconstruction, Level 2 targets state estimation, and Level 3 targets the final control objective. Here, L-2 describes the function of

extracting position information from CSI, rather than an implemented semantic transport link. The learning problem is to construct an estimator

$$\hat{\mathbf{p}}_k = f_\theta(\mathbf{X}_k), \quad (2.3)$$

where $\mathbf{X}_k$ is the representation of simulated CSI supplied directly to the estimator and f_θ is a neural network parameterized by θ . In the system studied in this thesis, $\mathbf{X}_k$ is formed from the real and imaginary parts of selected CSI entries over antenna and subcarrier dimensions. This representation allows the model to use both spatial and frequency-domain channel structure while also making it possible to study the effect of retaining selected CSI coefficients under the adopted masking procedure.

The estimated position is then used as the measurement input to the downstream state estimator and controller. In the closed-loop part of the thesis, the target follows a discrete-time dynamical model, the state estimate is updated using a Kalman filter, and an LQR controller computes the control input from the estimated state. The role of the learning-based CSI estimator is therefore not only to minimize a standalone localization error, but also to provide measurements that are useful for closed-loop tracking. This distinction is important: two estimators with similar average position error may lead to different control behavior if their errors have different variance, bias, or temporal structure.

The scope of the thesis is deliberately restricted. The wireless channel is generated from a fixed ray-tracing scene, and the target height is fixed so that the primary unknown is the two-dimensional position. The channel observation is generated from position rather than velocity, so Doppler effects are not part of the current channel model. The study evaluates position information in CSI and fixed-covariance sensitivity in a Kalman-filtered LQR loop. The modeled radio propagation generates the sensing observation; it does not carry an encoded state message between the learning model and controller. Those modules exchange values directly, without modeled bitrate, quantization, packet loss, or transport delay.

2.2 Wireless Channel State Information

Channel state information describes how a transmitted radio signal is transformed by the wireless propagation environment before it is received. In an OFDM system, the channel can be represented in the frequency domain at a set of subcarrier frequencies. For the antenna index $m \in \{1, \dots, M\}$ and subcarrier index $n \in \{1, \dots, N\}$, the complex channel frequency response is denoted by

$$H_{m,n} = H_m(f_n) \in \mathbb{C}, \quad (2.4)$$

where f_n is the frequency of the n th subcarrier. This representation is useful for the thesis because different target positions lead to different path lengths, angles, and attenuation patterns, which change the amplitude and phase of $H_{m,n}$ across both antenna and frequency dimensions. Wideband OFDM therefore provides delay-related information, while the antenna array provides spatial information. These two forms of diversity are the main reason CSI can be used as a wireless sensing feature for localization [15, 22].

The complex channel is represented by separating its real and imaginary components. For one channel observation, the input tensor is written as

$$\mathbf{X} = \begin{bmatrix} \text{Re}\{\mathbf{H}\} \\ \text{Im}\{\mathbf{H}\} \end{bmatrix} \in \mathbb{R}^{2 \times M \times N}, \quad (2.5)$$

where M is the number of antenna elements and N is the number of subcarriers. This real-valued tensor form is compatible with standard neural-network layers while still preserving the phase-sensitive information contained in the complex CSI. The thesis uses this tensor as the observation from which the two-dimensional target position is estimated.

Before the CSI is given to the learning model, the channel is normalized by its total energy, so that the total input energy is identical for every sample. This is important for the thesis because the neural network then learns position-dependent channel structure rather than only fitting large variations in absolute received power. To avoid stating the same operation twice with different notation, the exact form of this normalization is defined once in Eq. (3.5) of Section 3.3.1, together with the rest of the preprocessing pipeline.

The thesis also studies a resource-limited setting, in which only part of the antenna-subcarrier grid is used for one measurement. Because the way that selection is made is a design choice of this thesis rather than standard background, it is defined in the method chapter, in Eq. (3.6) and Eq. (3.7). What matters here is the underlying question it raises: the system does not need to preserve every channel coefficient if a smaller representation retains the position-relevant information required by the downstream estimator and controller.

2.3 Ray-Tracing-Based Channel Simulation Using Sionna

The thesis uses ray-tracing-based simulation to generate controllable CSI observations for different target positions. Ray tracing is appropriate in this setting because the wireless channel depends strongly on the geometry of the environment, including line-of-sight paths, reflected paths, diffracted paths, antenna placement, and object layout. Standardized channel models such as 3GPP TR 38.901 formalize many of these geometry-dependent propagation assumptions for modern wireless systems [18]. Dataset and simulator frameworks such as DeepMIMO and Sionna RT further show how ray-tracing-generated channels can be used for learning-based wireless experiments [12, 19].

For a receiver position $\mathbf{p}$, the simulator computes a set of propagation paths between the transmitter and receiver. A simplified frequency-domain expression for the channel at subcarrier frequency f_n is

$$H_m(f_n; \mathbf{p}) = \sum_{\ell=1}^{L(\mathbf{p})} a_{m,\ell}(\mathbf{p}) \exp(-j2\pi f_n \tau_\ell(\mathbf{p})), \quad (2.6)$$

where $L(\mathbf{p})$ is the number of propagation paths, $a_{m,\ell}(\mathbf{p})$ is the complex path coefficient observed at antenna element m , and $\tau_\ell(\mathbf{p})$ is the delay of path ℓ . This

equation is not used as a hand-designed localization algorithm in the thesis. Instead, it explains why the generated CSI changes with position: moving the target changes path lengths, angles, and interactions with the environment, which changes the complex response over antennas and subcarriers.

This thesis uses Sionna RT to generate the channel data. The simulated scene, the array geometry, the carrier and bandwidth settings, the receiver configuration and the base-station placement are properties of the experimental setup rather than of the propagation model itself, and they are therefore specified in Section 4.2 and Section 4.4. What matters for the present chapter is the structure that the simulator produces: for every UE position it returns a channel frequency response over a fixed grid of antenna elements and subcarriers, so that the learning problem is defined on a consistent observation format, and the labels are expressed relative to the serving base station.

The generated channel is a channel frequency response rather than a Doppler-dependent time-varying channel. This is consistent with the thesis scope stated in the proposal: the sensing observation is generated from position, while velocity appears in the downstream state-space model used for tracking and control. This separation keeps the preliminary wireless-sensing problem focused. The channel model supplies position-dependent CSI snapshots, and the control model later explains how a sequence of such position measurements affects closed-loop behavior.

2.4 Learning-Based Position Estimation Model

The state-estimation objective motivated by L-2 is formulated as supervised position estimation from CSI. A dataset consists of pairs

$$\mathcal{D} = \left\{ \left(\tilde{\mathbf{X}}^{(i)}, \mathbf{p}^{(i)} \right) \right\}_{i=1}^{N_s}, \quad (2.7)$$

where $\tilde{\mathbf{X}}^{(i)}$ is the normalized CSI tensor for sample i , $\mathbf{p}^{(i)} \in \mathbb{R}^2$ is the corresponding two-dimensional position label, and N_s is the number of samples. The learning model is a parameterized function f_θ that maps CSI to a position estimate,

$$\hat{\mathbf{p}}^{(i)} = f_\theta \left(\tilde{\mathbf{X}}^{(i)} \right). \quad (2.8)$$

A basic training objective is the mean squared position error,

$$\mathcal{L}_{\text{MSE}}(\theta) = \frac{1}{N_s} \sum_{i=1}^{N_s} \left\| \mathbf{p}^{(i)} - f_\theta \left(\tilde{\mathbf{X}}^{(i)} \right) \right\|_2^2. \quad (2.9)$$

This supervised loss implements the state-estimation objective motivated by the L-2 interpretation: predict position from CSI rather than reconstruct CSI. It does not optimize a transmitted message or include communication cost or closed-loop control cost in training.

Several neural architectures can be used for this mapping. A fully connected model treats the CSI tensor as a flattened feature vector and provides a simple baseline. An LSTM model treats the subcarrier dimension as an ordered sequence, using each

subcarrier as a feature vector formed from the antenna and real-imaginary dimensions; this is motivated by the use of recurrent networks for sequential dependencies [30]. The Transformer variant uses convolutional patch extraction followed by self-attention, reflecting the broader use of attention mechanisms and patch-based representations for structured data [31, 32]. The main model used in the current uncertainty-aware pipeline is a ResNet-style convolutional network. Residual networks are relevant because they allow deeper convolutional feature extractors to be optimized more reliably [29]. In this thesis, the convolutional viewpoint is natural because nearby antennas and subcarriers often contain correlated spatial-frequency information.

For the ResNet-based model, the network predicts both a mean position and a position variance. Let

$$(\hat{\boldsymbol{\mu}}^{(i)}, \hat{\boldsymbol{\sigma}}^{2(i)}) = f_{\theta}(\tilde{\mathbf{X}}^{(i)}), \quad (2.10)$$

where $\hat{\boldsymbol{\mu}}^{(i)} \in \mathbb{R}^2$ is the predicted position mean and $\hat{\boldsymbol{\sigma}}^{2(i)} \in \mathbb{R}_+^2$ is the predicted coordinate-wise variance. The variance is produced by predicting log-variance values and exponentiating them, which ensures that the variance is positive. Under a diagonal Gaussian observation model, the negative log-likelihood loss used for variance learning is

$$\mathcal{L}_{\text{NLL}}(\theta) = \frac{1}{2N_s} \sum_{i=1}^{N_s} \sum_{d=1}^2 \left(\log \hat{\sigma}_d^{2(i)} + \frac{(p_d^{(i)} - \hat{\mu}_d^{(i)})^2}{\hat{\sigma}_d^{2(i)}} \right), \quad (2.11)$$

where d indexes the two position coordinates. This uncertainty-aware formulation is important because the downstream Kalman filter requires a measurement covariance. The learned variance offers a candidate interface between the neural CSI estimator and probabilistic state estimation, subject to calibration. In this thesis it is evaluated offline; the reported closed-loop experiments use a separately specified fixed covariance. Calibration-related work in deep learning shows why such predictive uncertainty must be treated carefully, since inaccurate confidence estimates can be harmful even when the mean prediction error is small [34, 36].

2.5 Closed-Loop Sensing-Communication-Control

The final technical ingredient needed for the thesis is the closed-loop model that connects wireless position estimates to control performance. The target state is modeled with a discrete-time constant-velocity system,

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k, \quad (2.12)$$

where $\mathbf{x}_k$ is the state, $\mathbf{u}_k \in \mathbb{R}^2$ is the control input, and $\mathbf{w}_k$ is process noise. With sampling time Δt , the state-transition and input matrices are

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}. \quad (2.13)$$

The observation model extracts position from the state,

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad (2.14)$$

where $\mathbf{z}_k$ is the position measurement supplied by the neural CSI estimator and $\mathbf{v}_k$ is measurement noise. The filter interface can accept a learned or fixed measurement covariance. In all reported closed-loop experiments, $\mathbf{z}_k$ is the position estimated from CSI and the covariance is fixed at $r\mathbf{I}_2$ within each rollout. The learned variance head is assessed separately in offline diagnostics.

The Kalman filter combines the motion model with the CSI-based position measurement. Given a previous estimate $\hat{\mathbf{x}}_{k-1}$ and covariance $\mathbf{P}_{k-1}$, the prediction step is

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_{k-1}, \quad \mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}, \quad (2.15)$$

where $\mathbf{Q}$ is the process-noise covariance. The measurement update is

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T (\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}_k^{\text{KF}})^{-1}, \quad (2.16)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}), \quad \mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}) \mathbf{P}_{k|k-1}, \quad (2.17)$$

where $\mathbf{R}_k^{\text{KF}}$ is the measurement covariance. These equations show how the selected measurement covariance weights CSI-based observations in the closed loop. A learned covariance could enter through this interface, but the reported experiments supply a fixed covariance. If $\mathbf{R}_k^{\text{KF}}$ is too small, the filter over-trusts the CSI-based measurement; if it is too large, the filter relies too much on prediction. This is the theoretical basis for evaluating the effect of measurement variance on tracking performance [37].

The control input is computed using an LQR objective. For a reference state $\mathbf{x}_k^{\text{ref}}$, the instantaneous quadratic cost is

$$c_k = (\mathbf{x}_k - \mathbf{x}_k^{\text{ref}})^T \mathbf{Q}^{\text{LQR}} (\mathbf{x}_k - \mathbf{x}_k^{\text{ref}}) + \mathbf{u}_k^T \mathbf{R}^{\text{LQR}} \mathbf{u}_k, \quad (2.18)$$

where $\mathbf{Q}^{\text{LQR}}$ penalizes tracking error and $\mathbf{R}^{\text{LQR}}$ penalizes control effort. The feedback law has the form

$$\mathbf{u}_k = -\mathbf{K}^{\text{LQR}} (\hat{\mathbf{x}}_k - \mathbf{x}_k^{\text{ref}}), \quad (2.19)$$

where $\mathbf{K}^{\text{LQR}}$ is obtained from the discrete-time Riccati recursion. The weighting matrices $\mathbf{Q}^{\text{LQR}}$ and $\mathbf{R}^{\text{LQR}}$ set the trade-off between tracking accuracy and control effort, and their choice also determines which state components are penalized at all; the values used in the experiments are reported in Section 4.4. This closed-loop formulation follows the broader networked-control view that communication and sensing quality is judged by its effect on control performance rather than only by signal-level fidelity [38, 39, 13].

Together, the CSI model, learning-based estimator, Kalman filter, and LQR controller define the technical background needed for the rest of the thesis. The later method chapter specifies how the channel data are generated, how the neural models are trained, and how the closed-loop experiments are performed. The preliminary

material in this chapter explains why those components are connected: CSI carries position information, the neural estimator predicts position measurements, the Kalman filter weights that state estimate according to measurement uncertainty, and the LQR controller turns the resulting state estimate into a measurable tracking cost.

3

Method

3.1 Method Overview

This thesis addresses the problem of estimating the position of a moving agent from wireless channel observations and using this estimate in a closed-loop control setting. The central idea is that the channel state information (CSI) contains task-relevant spatial information about the agent, even though the raw channel response is high-dimensional and complex-valued. Instead of treating the wireless observation only as a signal to be reconstructed, the method uses it as the input to a localization pipeline whose output is directly useful for tracking and control.

The method combines offline supervised learning with simulated feedback control. In the offline stage, a ray-tracing-based wireless environment is used to generate channel observations for sampled user equipment (UE) positions. This step creates pairs of positions and CSI observations under a controlled sensing geometry. Second, the complex-valued CSI is converted into a representation that can be processed by a neural network. The representation preserves the antenna and subcarrier structure of the channel, since these dimensions carry spatial and frequency-dependent information that is relevant for localization. Third, a neural position estimator is trained to predict the UE position from the CSI representation. In this thesis, the main estimator is based on a ResNet-34 backbone adapted to CSI input. Its performance is evaluated using localization error measures such as the root mean squared error (RMSE) and the cumulative distribution function (CDF) of the position error.

The trained localization model is then connected to the feedback loop shown in Fig. 3.1. At each step, ray-traced CSI generated from the simulated target position is processed by the trained network. Its position estimate is passed directly to a Kalman filter, which combines the measurement with a motion-model prediction. An LQR controller computes a control input from the filtered state, and the input is applied directly to the simulated target dynamics. The main closed-loop experiments use fixed diagonal measurement covariances; the learned variance head is evaluated separately. All module interfaces are ideal data transfers. The simulation does not implement pilot-based CSI acquisition, feedback signaling, message quantization, an encoder–transport–decoder chain, or measurement and command packet loss or latency.

The chapter first describes channel and dataset generation (Section 3.2), then CSI representation, neural estimation, and training (Section 3.3). Section 3.4 presents the Kalman-filter and LQR feedback loop.

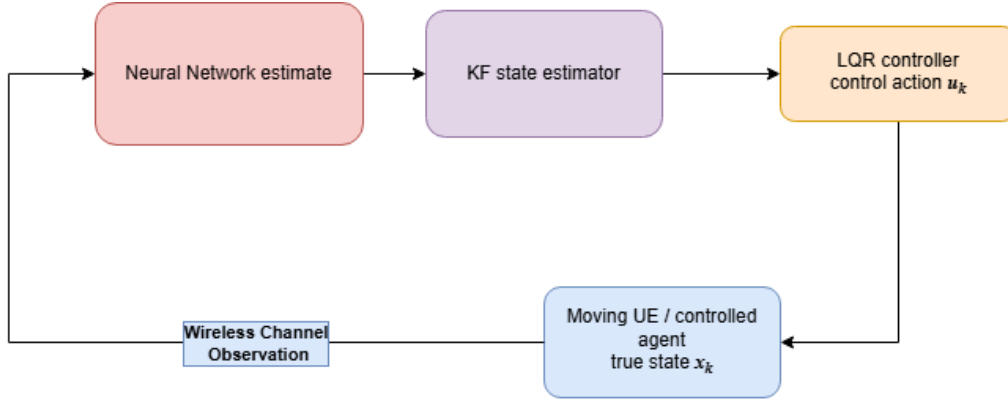


Figure 3.1: Simulated feedback loop for CSI-based tracking. A neural network maps wireless channel observations to position measurements, a Kalman filter estimates the state, and an LQR controller applies a control input to the moving UE. The resulting UE position determines the channel observation at the next step.

3.2 Channel and Dataset Generation

This section describes how the position-channel dataset used in this thesis is generated. The purpose of this part of the method is to create paired samples between physical user equipment (UE) positions and the corresponding wireless channel observations. These pairs are required because the localization model is trained in a supervised manner: the input is the channel state information (CSI), and the target is the UE position.

Let the position of the UE in sample i be denoted by

$$\mathbf{p}_i = [x_i, y_i, z_i]^T. \quad (3.1)$$

In this thesis, the UE height is fixed, so the localization target is the horizontal position of the UE. For each sampled position, the UE is placed in the ray-tracing scene and the propagation paths from the base stations to the UE are computed. The resulting paths are then converted into a frequency-domain channel response. For each sampled position the ray tracer returns the propagation paths, which are converted into the frequency-domain response of Eq. (2.6), evaluated per antenna element and subcarrier and written here as $h_{m,n}(\mathbf{p}_i)$ for UE position $\mathbf{p}_i$. This is what makes the channel informative for localization: changing the UE position changes the path lengths, delays, phases and gains, which produces a position-dependent CSI pattern across antennas and subcarriers.

The CSI sample for one base station is represented as

$$\mathbf{H}_i \in \mathbb{C}^{M \times N}, \quad (3.2)$$

where M is the number of antenna elements and N is the number of subcarriers. Since the learning model operates on real-valued tensors, this complex matrix is converted into the two-channel real tensor $\mathbf{X}_i \in \mathbb{R}^{2 \times M \times N}$ of Eq. (2.5), which keeps the spatial-frequency structure of the CSI while making it suitable as input to standard neural network layers.

The generated dataset can therefore be written as

$$\mathcal{D} = \{(\mathbf{X}_i, \mathbf{y}_i)\}_{i=1}^K, \quad (3.3)$$

where K is the number of generated samples. The target label is defined as the two-dimensional UE position relative to the first base station,

$$\mathbf{y}_i = [x_i, y_i]^T - [x_{\text{BS1}}, y_{\text{BS1}}]^T. \quad (3.4)$$

Using a base-station-centered coordinate frame makes the prediction target consistent with the single-base-station localization setting used by the learning pipeline. In the data generation procedure, UE positions are sampled within a predefined region of the scene. For each sampled position, the ray-tracing solver computes the channel frequency response for the configured base-station geometry, antenna array, carrier frequency, and subcarrier grid. The generated samples are stored as UE positions together with the corresponding CSI observations. Although channel responses are generated for two base stations, the localization model in this thesis primarily uses the CSI from the first base station. The second base-station channel is retained for possible extensions to multi-base-station localization.

This channel and dataset generation stage provides the data foundation for the learning method. The next section describes how the CSI tensor is preprocessed and how the neural position estimator learns the mapping from $\mathbf{X}_i$ to the relative position label $\mathbf{y}_i$.

3.3 Model Building and Training

This section describes the learning component of the proposed method. After the channel and dataset generation stage, each sample consists of a CSI observation and a corresponding relative UE position label. The purpose of the model-building stage is to define how this CSI observation is represented as neural-network input, how the position estimator maps this input to a localization output, and how the estimator is trained.

The first step is CSI representation and preprocessing. The generated channel response is complex-valued and structured over antenna and subcarrier dimensions. Before it can be used by the neural network, it is represented as real-valued input by separating the real and imaginary parts. The channel is also normalized so that the model learns position-dependent CSI patterns rather than relying on arbitrary differences in signal magnitude. This step prepares the data for the estimator while preserving the spatial-frequency structure introduced in Section 3.2.

The second step is neural position estimation. The estimator learns a nonlinear mapping from the preprocessed CSI tensor to the UE position in the base-station-centered coordinate frame. In this thesis, the main model is based on a residual convolutional network, which is suitable for extracting local and hierarchical features from the antenna-subcarrier representation. The position output supplies the localization estimate. An additional variance head implements a candidate uncertainty interface for filtering and control, but its predictions are evaluated only in the

offline diagnostics; the reported closed-loop experiments specify a fixed measurement covariance separately.

The third step is the training objective. The estimator is optimized as a supervised regression model using the generated position-channel pairs. The loss function measures the discrepancy between the predicted position and the ground-truth relative position label. This objective connects the dataset generation stage to the final localization task: the model is not trained to reconstruct the full CSI, but to preserve and extract the information that is useful for position estimation.

The following subsections explain these three parts in order: CSI representation and preprocessing, the neural position estimator, and the training objective.

3.3.1 CSI Representation and Preprocessing

The input to the localization model is the channel frequency response generated in Section 3.2. For each UE position, the channel observed at one base station is a complex-valued matrix $\mathbf{H}_i \in \mathbb{C}^{M \times N}$, where M denotes the number of antenna elements and N denotes the number of subcarriers. The antenna dimension captures spatial variations across the array, while the subcarrier dimension captures frequency-dependent variations caused by delay and multipath propagation. Both dimensions are therefore kept in the learning representation instead of flattening the CSI at the preprocessing stage.

Since standard neural network layers operate on real-valued tensors, the complex CSI is represented by its real and imaginary parts as defined in Eq. (2.5). Written per sample, this gives $\mathbf{X}_i \in \mathbb{R}^{2 \times M \times N}$, where the first index selects the real or the imaginary component of the channel coefficient at antenna element m and subcarrier n . Keeping the two components separately preserves both amplitude- and phase-related information, which matters because localization depends on how the channel changes across antennas and frequencies.

A sample-wise normalization is then applied to reduce the effect of arbitrary channel scale. Let $\|\mathbf{X}_i\|_F^2$ denote the total squared energy of the real-valued CSI tensor. The normalized input is written as

$$\widetilde{\mathbf{X}}_i = \gamma_i \mathbf{X}_i, \quad \gamma_i = \sqrt{\frac{E_0}{\|\mathbf{X}_i\|_F^2}}, \quad (3.5)$$

where $E_0 = 2M = 64$ is the reference energy for the $2 \times 32 \times 400$ input. This is the normalization implemented for the reported experiments and assumes nonzero sample energy; the code has no explicit zero-energy guard. Applying one positive scale factor to all coefficients preserves their relative antenna-subcarrier pattern within each sample, while removing the original total-energy differences between samples. The choice of E_0 sets the input scale; it does not imply that training would be invariant to a different scale. Section 2.2 refers to this definition.

The preprocessing also supports resource-aware localization by masking parts of the CSI tensor. This is motivated by closed-loop ISAC settings in which the sensing quality depends on the allocated wireless resources. Related closed-loop ISAC formulations model the channel measurement as the input to a localization model and

allow sensing resources, such as subcarrier allocation, to affect the observation quality [1]. In this thesis, this idea is represented by selecting only a subset of antennas and subcarriers before the CSI is passed to the localization model.

Let $\mathcal{M}_a \subseteq \{1, \dots, M\}$ denote the selected antenna indices and $\mathcal{M}_f \subseteq \{1, \dots, N\}$ denote the selected subcarrier indices. The masked CSI tensor is

$$\bar{\mathbf{X}}_i[c, m, n] = \widetilde{\mathbf{X}}_i[c, m, n] \mathbf{1}_{\{m \in \mathcal{M}_a\}} \mathbf{1}_{\{n \in \mathcal{M}_f\}}, \quad (3.6)$$

for channel index $c \in \{0, 1\}$. In practice, the selected antennas and subcarriers are centered within the full antenna-subcarrier grid, while all unselected entries are set to zero. This keeps the tensor size fixed for the neural network and allows different sensing-resource configurations to be represented by different masks.

The masking in Eq. (3.6) is instantiated through a discrete resource action $a \in \{1, \dots, A\}$ with $A = 8$. Each action selects an antenna count K_a^a and a subcarrier count K_a^f jointly, so that the spatial and frequency dimensions are reduced together rather than independently,

$$\begin{aligned} (K_a^a, K_a^f) \in \{ & (4, 50), (8, 100), (12, 150), (16, 200), \\ & (20, 250), (24, 300), (28, 350), (32, 400) \}. \end{aligned} \quad (3.7)$$

For a given action, the index sets used in Eq. (3.6) are the centred blocks

$$\mathcal{M}_a = \left\{ \frac{M-K_a^a}{2} + 1, \dots, \frac{M+K_a^a}{2} \right\}, \quad \mathcal{M}_f = \left\{ \frac{N-K_a^f}{2} + 1, \dots, \frac{N+K_a^f}{2} \right\}, \quad (3.8)$$

and the fraction of retained channel coefficients is

$$\rho_a = \frac{K_a^a K_a^f}{MN}, \quad (3.9)$$

which ranges from $\rho_1 = 0.016$ for the smallest configuration to $\rho_8 = 1$ for the full-resource case. The action index is therefore a scalar description of the sensing-resource budget spent on one localization measurement.

Selecting a centred block instead of a random subset is deliberate, because the two masked dimensions have different physical meanings. Retaining K_a^a contiguous antenna elements at the centre of the array is equivalent to operating a smaller uniform linear array, which reduces the effective aperture and therefore coarsens the angular resolution of the observation. Retaining K_a^f contiguous subcarriers at the centre of the frequency grid reduces the occupied bandwidth, which coarsens the delay resolution. A resource action thus does not remove arbitrary entries of the CSI matrix; it emulates a receiver that is configured with fewer antennas and a narrower band, which is the quantity a resource allocator in a closed-loop ISAC system would actually control.

During training and validation, one of the eight actions is drawn uniformly whenever a sample is loaded. Zero-filling keeps the input shape $2 \times M \times N$ fixed, allowing one network to be trained across all eight configurations. In the mixed-resource test evaluation, a seed-42 random sequence assigns one action to each ordered test sample; the FCNN uses the same sequence. The resource sweep in Section 4.5.4 instead applies each fixed action to all test samples. These protocols assess average

performance under mixed masks and performance conditional on a particular mask, respectively.

One consequence of the ordering of the preprocessing steps is important for interpreting the resource experiments. The energy normalization of Eq. (3.5) is applied to the full CSI tensor before the mask of Eq. (3.6) is applied, so the masked input retains only a fraction of the reference energy E_0 . A resource action therefore reduces the resolution of the observation and the received energy at the same time, which corresponds to a receiver that stops using part of the array and part of the band rather than one that redistributes the same power over fewer resources. The two effects are not separated in this thesis, and the reported resource dependence is interpreted as the combined effect of both.

The same representation supports two evaluated model families. The convolutional estimator in Section 3.3.2 processes $\bar{\mathbf{X}}_i$ as a two-channel tensor with local antenna-subcarrier adjacency. The FCNN baseline in Section 3.3.3 flattens it into a vector of length $2MN$, preserving all coefficients and their fixed ordering but using no convolutional weight sharing or explicit local adjacency bias. This provides a structural reference for localization. Because model sizes and training budgets differ, the comparison does not isolate the benefit of preserving two-dimensional adjacency.

Sequence and patch-token views of the CSI tensor were also implemented for recurrent and Transformer models, respectively, but neither is evaluated in this thesis. No localization or closed-loop results are claimed for them. A predicted variance head is not a prerequisite for the reported control setup, which supplies a fixed measurement covariance separately.

After these preprocessing steps, each training sample consists of a fixed-size tensor $\bar{\mathbf{X}}_i$ and the relative position label $\mathbf{y}_i$ defined in Section 3.2. The next subsection describes the neural estimator that maps this preprocessed CSI representation to a position prediction.

3.3.2 Neural Position Estimator

The neural position estimator maps the preprocessed CSI tensor to a localization output. Its input is the masked and normalized CSI representation $\bar{\mathbf{X}}_i \in \mathbb{R}^{2 \times M \times N}$ defined in Section 3.3.1. For a mini-batch of B samples, the network input has shape

$$\bar{\mathbf{X}} \in \mathbb{R}^{B \times 2 \times M \times N}, \quad (3.10)$$

where the second dimension corresponds to the real and imaginary parts of the CSI. The output consists of a two-dimensional position estimate and, when uncertainty estimation is enabled, a two-dimensional variance estimate.

The main estimator used in this thesis is based on a residual convolutional network. A ResNet-34 backbone is adapted to the CSI localization problem by replacing the first convolutional layer so that it accepts two input channels instead of the three color channels used in image classification. The final classification layer is removed, and the backbone is used as a feature extractor over the antenna-subcarrier plane. Conceptually, the backbone computes a feature vector

$$\mathbf{r}_i = g_\theta(\bar{\mathbf{X}}_i), \quad \mathbf{r}_i \in \mathbb{R}^d, \quad (3.11)$$

where $g_\theta(\cdot)$ denotes the adapted residual network and d is the dimension of the extracted feature representation.

This convolutional design is suitable for CSI-based localization because the CSI tensor has a meaningful two-dimensional structure. Local patterns across neighboring antenna elements describe spatial signatures of the arriving paths, while local patterns across neighboring subcarriers describe frequency-dependent changes caused by multipath delays. Residual blocks allow the model to combine such local patterns into deeper hierarchical features without requiring the input tensor to be flattened at the beginning of the network.

Two output heads are attached to the shared feature vector. Each applies dropout with probability 0.2 before a linear layer during training. The following equations describe inference, when dropout is inactive. The localization head predicts the relative UE position,

$$\hat{\mathbf{y}}_i = W_\mu \mathbf{r}_i + \mathbf{b}_\mu, \quad \hat{\mathbf{y}}_i \in \mathbb{R}^2, \quad (3.12)$$

where the two components correspond to the horizontal position in the base-station-centered coordinate frame. The uncertainty head predicts a log-variance vector,

$$\hat{\mathbf{l}}_i = \max(W_\sigma \mathbf{r}_i + \mathbf{b}_\sigma, -2), \quad \hat{\mathbf{l}}_i \in \mathbb{R}^2. \quad (3.13)$$

Here the maximum is applied element-wise, matching the lower clamp used by the implementation. The variance estimate is obtained by exponentiating this effective log variance,

$$\hat{\sigma}_i^2 = \exp(\hat{\mathbf{l}}_i), \quad \hat{\sigma}_i^2 \in \mathbb{R}_+^2. \quad (3.14)$$

This parameterization ensures positive predicted variance and implements a candidate measurement-uncertainty interface. Similar closed-loop ISAC formulations use learned localization observations and uncertainty estimates as inputs to downstream decision-making modules [1]. Here, both heads are implemented and trained, but the learned covariance fails the reported offline consistency checks. The reported feedback experiments therefore use the position head with a separately specified fixed covariance; they do not test learned uncertainty in control.

For a batch input, the model therefore produces

$$\hat{\mathbf{Y}} \in \mathbb{R}^{B \times 2}, \quad \hat{\mathbf{V}} \in \mathbb{R}^{B \times 2}, \quad (3.15)$$

where $\hat{\mathbf{Y}}$ contains the predicted relative positions and $\hat{\mathbf{V}}$ contains the predicted coordinate-wise variances. Figure 3.2 summarizes the offline data-generation and training workflow; the backbone and output heads are specified by Eqs. (3.11)–(3.14). The following subsection explains how these outputs are used in the training objective.

3.3.3 Structural Baseline Estimator

The convolutional design of Section 3.3.2 is motivated by the two-dimensional structure of the CSI tensor. A fully connected neural network (FCNN) provides a localization baseline using the same preprocessed CSI coefficients without convolutional

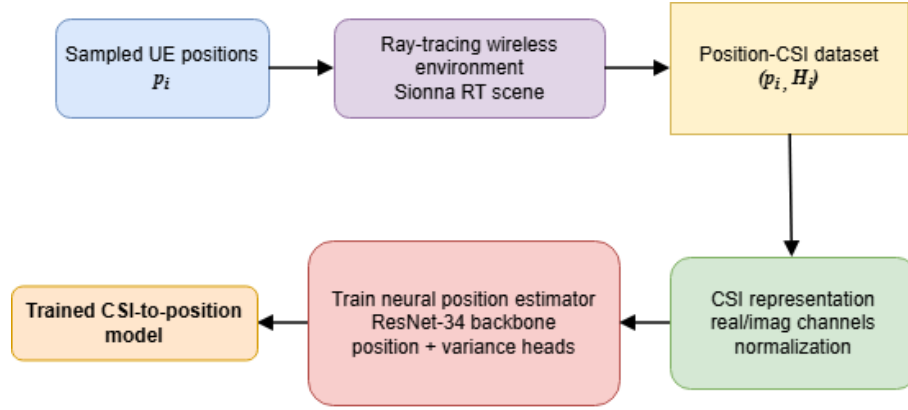


Figure 3.2: Offline data-generation and training workflow. Sampled UE positions and ray-traced CSI form a supervised position–CSI dataset. Real and imaginary channel components are normalized and supplied to the ResNet-based position and variance training stages, yielding the trained CSI-to-position model.

weight sharing or an explicit local antenna–subcarrier adjacency bias. The baseline flattens the preprocessed tensor into a single vector,

$$\mathbf{x}_i = \text{vec}(\bar{\mathbf{X}}_i) \in \mathbb{R}^{2MN}, \quad (3.16)$$

which for $M = 32$ and $N = 400$ has dimension 25 600, and maps it to a position estimate with a fully connected network,

$$\hat{\mathbf{y}}_i^{\text{fc}} = h_\phi(\mathbf{x}_i), \quad \hat{\mathbf{y}}_i^{\text{fc}} \in \mathbb{R}^2, \quad (3.17)$$

where h_ϕ consists of three hidden layers of width 1024, 1024 and 256 with ReLU activations, followed by a linear output layer. Flattening preserves the coefficients and their fixed ordering; it changes the architectural treatment of the input rather than removing information from the CSI.

The FCNN has 27 527 938 parameters, compared with 21 283 588 for the residual backbone and its two output heads. These counts describe the evaluated implementations; parameter count alone does not establish equal effective capacity or an otherwise controlled architectural comparison.

Both models use the same 16 000/2000/2000 training, validation, and test split, sample-wise CSI normalization, and uniform distribution over the eight resource actions in Eq. (3.7). The FCNN minimizes position MSE using Adam with learning rate 10^{-4} , weight decay 10^{-4} , and batch size 32, matching the optimizer settings of the ResNet localization stage. The FCNN checkpoint is selected by minimum logged validation localization error; the maximum training budget is 150 epochs with early-stopping patience 25. Training stops after 147 epochs, with the selected checkpoint at zero-based epoch 121. The ResNet localization stage runs for 500 epochs; its subsequent variance stage freezes the trainable parameters of the backbone and position head. The optimization budgets are therefore unequal. Each architecture is represented by one training run, so the comparison does not quantify variation across training seeds.

Evaluation uses the same ordered test indices and resource-mask sequence, generated with seed 42 and without shuffling, for both models. The resulting errors are paired

by test position and resource action. The FCNN predicts position only and is evaluated only for localization; it has no predicted covariance or NEES result and is not used in the reported closed-loop experiments. Its position output could in principle be combined with a separately specified measurement covariance, but that additional control experiment is outside the present comparison. Table 4.4 reports the localization results.

3.3.4 Training Objective

The neural estimator is trained using the supervised position-channel pairs generated in Section 3.2. The objective is to make the predicted position close to the relative UE position label while also learning an uncertainty estimate that reflects the reliability of the localization output. This follows the uncertainty-aware localization idea used in closed-loop ISAC systems, where the localization model provides both a position observation and an associated covariance for downstream state inference [1]. Let $\mathbf{y}_i \in \mathbb{R}^2$ be the relative position label for sample i , and let $\hat{\mathbf{y}}_i$ and $\hat{\boldsymbol{\sigma}}_i^2$ be the position and variance outputs of the neural estimator. If the position labels are normalized by a scale vector $\mathbf{s}_p$, the normalized label is

$$\tilde{\mathbf{y}}_i = \mathbf{y}_i \oslash \mathbf{s}_p, \quad (3.18)$$

where $\oslash$ denotes element-wise division. In the configuration used for the reported experiments $\mathbf{s}_p = [1, 1]^T$, so the normalized coordinates coincide with the position labels in metres; the scale is kept in the formulation because it is what makes the conversion of the predicted variance in Eq. (3.27) explicit for any other choice of label scaling. The prediction error in normalized coordinates is then

$$\mathbf{e}_i = \tilde{\mathbf{y}}_i - \hat{\mathbf{y}}_i. \quad (3.19)$$

The first part of the objective is a position regression loss. For a mini-batch $\mathcal{B}$ with B samples, the mean squared error (MSE) loss is

$$\mathcal{L}_{\text{pos}} = \frac{1}{2B} \sum_{i \in \mathcal{B}} \|\tilde{\mathbf{y}}_i - \hat{\mathbf{y}}_i\|_2^2. \quad (3.20)$$

The factor $1/(2B)$ averages over both samples and the two coordinate components, matching the implemented MSE. This differs from the reported two-dimensional localization RMSE, which takes the square root of the sample mean squared Euclidean error without dividing by two (Eq. (4.3)).

The second part of the objective is used when the model is trained to output localization uncertainty. The variance head provides the lower-clamped log-variance vector $\hat{\mathbf{l}}_i$ defined in Eq. (3.13), which is converted into a positive variance estimate by $\hat{\boldsymbol{\sigma}}_i^2 = \exp(\hat{\mathbf{l}}_i)$. Assuming independent Gaussian errors in the two coordinate dimensions, the negative log-likelihood loss is

$$\mathcal{L}_{\text{var}} = \frac{1}{2B} \sum_{i \in \mathcal{B}} \sum_{d \in \{x, y\}} \left(\hat{l}_{i,d} + \frac{e_{i,d}^2}{\hat{\sigma}_{i,d}^2} \right), \quad (3.21)$$

where $e_{i,d}$ is the normalized position error in coordinate d . This loss encourages the model to assign small variance to accurate predictions and larger variance to samples where the CSI leads to larger localization errors. The logarithmic term prevents the model from increasing the variance without penalty.

The training can therefore be interpreted as learning both the conditional mean and the coordinate-wise uncertainty of the position given the CSI input,

$$p(\mathbf{y}_i | \bar{\mathbf{X}}_i) \approx \mathcal{N}(\hat{\mathbf{y}}_i, \text{diag}(\hat{\boldsymbol{\sigma}}_i^2)). \quad (3.22)$$

This distribution defines the covariance evaluated in the offline uncertainty diagnostics. It could provide a measurement covariance for Kalman filtering if sufficiently calibrated; the reported closed-loop experiments instead use fixed covariances.

The two loss terms are not optimized simultaneously. The estimator is trained in two sequential stages, and this separation is a design decision rather than an implementation detail.

In the first stage, the backbone g_θ and position-head parameters are trained for 500 epochs with $\mathcal{L}_{\text{pos}}$, while the variance-head parameters are frozen. The checkpoint minimizing the logged validation localization error is retained.

In the second stage, that checkpoint initializes 100 epochs of training with $\mathcal{L}_{\text{var}}$. Only the variance-head parameters are optimized, and the checkpoint minimizing logged validation NLL is retained. The implementation checks that the backbone and position-head parameters remain frozen. This does not freeze all model state: batch-normalization running statistics differ between the saved stage-1 and stage-2 checkpoints, so unchanged position predictions cannot be inferred from the parameter checks. Appendix A specifies the checkpoint identities and validation aggregation.

The purpose of this ordering is to learn the position mapping first and then fit uncertainty without jointly optimizing the mean-prediction parameters. For fixed variance, increasing the squared position residual increases the Gaussian NLL; worsening the position estimate is therefore not, by itself, a way to reduce the stated loss. The variance head must use features learned for localization, which may limit its ability to represent uncertainty. This is a possible explanation for the observed calibration limitations, not a mechanism established by the present experiments.

During training and validation, the predicted normalized position and variance are converted back to physical units using the position scale $\mathbf{s}_p$: the predicted mean is multiplied by $\mathbf{s}_p$ and the predicted variance by $\mathbf{s}_p^2$, which also defines the candidate learned covariance in Eq. (3.27). That candidate is evaluated offline and is not the covariance used in the reported Kalman-filter experiments. The localization error is then measured as the Euclidean distance between the predicted and ground-truth positions,

$$d_i = \|\mathbf{y}_i - \hat{\mathbf{y}}_i^{\text{phys}}\|_2, \quad (3.23)$$

where $\hat{\mathbf{y}}_i^{\text{phys}}$ denotes the prediction expressed in meters. This metric is not the training objective itself, but it provides an interpretable measure of localization accuracy and is later used in the experimental evaluation.

The training objective is supervised CSI-based position regression followed by variance-head training. It implements task-relevant information extraction motivated by L-2, without reconstructing the full CSI. No communication encoder-decoder is trained,

and neither transport cost nor LQR tracking cost is included in the learning objective.

3.4 Closed-loop Control Method

This section describes how the learned localization model is used inside a closed-loop control system. The purpose of this part of the method is to connect CSI-based state estimation to control: the wireless channel is used to infer the current position of the agent, and the resulting estimate is used to compute a control action. This makes the localization model part of a feedback loop rather than only a standalone predictor.

The closed-loop system is built around a discrete-time state representation. The agent state contains its horizontal position and velocity,

$$\mathbf{x}_k = [x_k, y_k, v_{x,k}, v_{y,k}]^T, \quad (3.24)$$

where k denotes the control time step. At each step, the ray-tracing environment provides a channel observation for the current agent position. The trained neural estimator predicts position and coordinate-wise variance. In the reported closed-loop experiments, only its position output supplies the filter measurement; a fixed measurement covariance is selected separately for each rollout. The variance output is assessed in offline uncertainty diagnostics.

The first component of the closed-loop method is the Kalman filter. Its role is to combine the system dynamics with the noisy position measurements produced by the wireless localization model. The filter predicts the next state from the previous state and control action, and then corrects this prediction using the new position observation. This is important because the neural localization output may contain errors, while the physical motion model provides temporal consistency across time steps.

The second component is the linear quadratic regulator (LQR). The LQR computes the control action from the difference between the estimated state and a reference state. In this thesis, the control input represents acceleration in the horizontal plane. The controller therefore converts the filtered state estimate into an action that moves the agent toward the desired trajectory while balancing tracking accuracy and control effort.

The third component is the feedback loop connecting sensing, estimation, and control. After the LQR action is applied, the agent state evolves according to the motion model. The new position changes the wireless channel, which produces a new CSI observation in the next step. This closes the loop: CSI is used for localization, localization is used for state estimation, state estimation is used for control, and control changes the future channel observation.

The following subsections describe these components in order. Section 3.4.1 introduces the Kalman filter and its input-output relation. Section 3.4.2 describes the LQR control law and the control objective. Section 3.4.3 explains how the full sensing-estimation-control loop is formed.

3.4.1 Kalman Filter

The Kalman filter is used to estimate the full dynamic state of the agent from noisy position measurements. The neural localization model described in Section 3.3.2 provides a two-dimensional position observation, but the controller requires a state that also includes velocity. The Kalman filter bridges this gap by combining a motion model with the wireless localization output. The measurement covariance sets how strongly the filter weights those observations against its motion-model prediction. The model design permits a learned covariance, while the reported experiments use a fixed covariance throughout each rollout.

The state vector is defined in Eq. (3.24). A constant-velocity motion model is used, where the control input is the two-dimensional acceleration

$$\mathbf{u}_k = [a_{x,k}, a_{y,k}]^T. \quad (3.25)$$

The discrete-time state transition is the constant-velocity model of Eq. (2.12), with the matrices $\mathbf{F}$ and $\mathbf{B}$ given in Eq. (2.13) for sampling time Δt . This model assumes that the acceleration command affects both the position and velocity of the agent between two consecutive time steps.

The measurement is the position estimate produced by the wireless localization model,

$$\mathbf{z}_k = [\hat{x}_k, \hat{y}_k]^T. \quad (3.26)$$

Since only position is observed, the selection matrix $\mathbf{H}$ in Eq. (2.14) extracts the two position components. The neural estimator predicts coordinate-wise variances $\hat{\sigma}_k^2$ in the normalized label coordinates. Converting them to physical units defines a candidate learned measurement covariance,

$$\mathbf{R}_k^{\text{learned}} = \text{diag}(\hat{\sigma}_k^2 \odot \mathbf{s}_p^2), \quad (3.27)$$

where $\odot$ denotes element-wise multiplication. The mean is multiplied by $\mathbf{s}_p$ and translated from the base-station-centered frame to absolute coordinates to form $\mathbf{z}_k$; translation does not change the covariance. The squared scale factor is required for covariance diagnostics in physical units. In the reported configuration $\mathbf{s}_p = [1, 1]^T$. The learned covariance is evaluated offline and is not supplied to the reported filter runs. Those runs instead use $\mathbf{R}_k = r\mathbf{I}_2$, with $r \in \{0.01, 0.1, 1, 10, 100\} \text{ m}^2$, held constant within each rollout. This sweep tests sensitivity to an assumed measurement covariance; it does not adapt measurement confidence using the variance head.

At each control step, the filter applies the standard prediction, gain, and update equations in Section 2.5. The CSI-derived observation $\mathbf{z}_k$ corrects the motion-model prediction with weighting determined by the selected $\mathbf{R}_k$. Larger fixed r values reduce the measurement weight relative to the same predicted state covariance. This is a tuning choice in the reported sweep, rather than a response to the network’s confidence in an individual CSI snapshot.

In the closed-loop pipeline, this procedure is applied once per control step. The previous control action is used in the prediction step, the CSI-based localization output is used as the measurement, and the filtered state estimate is passed to the LQR controller. The Kalman filter is therefore the interface between the learned wireless sensing module and the model-based control module.

3.4.2 LQR Control

The LQR controller is used to compute the control action from the filtered state estimate produced by the Kalman filter. Its role in the closed-loop pipeline is to move the agent toward a reference trajectory while avoiding unnecessarily large control inputs. This is suitable for the setting considered in this thesis because the agent dynamics are represented by a linear discrete-time model, and the desired behavior can be expressed as a trade-off between state-tracking error and control effort.

Let $\hat{\mathbf{x}}_{k|k}$ be the filtered state estimate at time step k , and let $\mathbf{x}_k^{\text{ref}}$ be the reference state. The tracking error is defined as

$$\mathbf{e}_k = \hat{\mathbf{x}}_{k|k} - \mathbf{x}_k^{\text{ref}}. \quad (3.28)$$

The LQR objective penalizes this state error and the control effort over time. For a finite horizon, the cost can be written as

$$J = \sum_{k=0}^{T-1} \left(\mathbf{e}_k^T \mathbf{Q}_{\text{lqr}} \mathbf{e}_k + \mathbf{u}_k^T \mathbf{R}_{\text{lqr}} \mathbf{u}_k \right), \quad (3.29)$$

where $\mathbf{Q}_{\text{lqr}}$ weights the importance of tracking the reference state and $\mathbf{R}_{\text{lqr}}$ weights the control effort. In this thesis, the state cost mainly penalizes position error, while the control cost penalizes large acceleration commands. This makes the controller prefer actions that reduce localization-based tracking error without producing overly aggressive motion.

The optimal linear feedback controller has the form

$$\mathbf{u}_k = -\mathbf{K}_k \mathbf{e}_k, \quad (3.30)$$

where $\mathbf{K}_k$ is the LQR gain. For the discrete-time dynamics in Eq. (2.12) over a horizon of T steps, the gain follows from the discrete Riccati recursion, which is solved *backwards* in time from the terminal cost,

$$\mathbf{P}_t = \mathbf{F}^T \mathbf{P}_{t+1} \mathbf{F} - \mathbf{F}^T \mathbf{P}_{t+1} \mathbf{B} \left(\mathbf{R}_{\text{lqr}} + \mathbf{B}^T \mathbf{P}_{t+1} \mathbf{B} \right)^{-1} \mathbf{B}^T \mathbf{P}_{t+1} \mathbf{F} + \mathbf{Q}_{\text{lqr}}, \quad \mathbf{P}_T = \mathbf{Q}_{\text{lqr}}, \quad (3.31)$$

where the terminal condition $\mathbf{P}_T$ is the terminal state cost, taken equal to $\mathbf{Q}_{\text{lqr}}$ in this thesis. The corresponding feedback gain at time k is

$$\mathbf{K}_k = \left(\mathbf{R}_{\text{lqr}} + \mathbf{B}^T \mathbf{P}_{k+1} \mathbf{B} \right)^{-1} \mathbf{B}^T \mathbf{P}_{k+1} \mathbf{F}. \quad (3.32)$$

Since $\mathbf{F}$, $\mathbf{B}$, $\mathbf{Q}_{\text{lqr}}$ and $\mathbf{R}_{\text{lqr}}$ are constant, $\mathbf{P}_{k+1}$ depends only on the number of remaining steps $T - k$, so the gain can equivalently be indexed by the steps-to-go. The finite-horizon implementation uses this form: it propagates the recursion once from the terminal condition and caches the gain for every horizon length, which gives the same gains as recomputing the recursion at each step but with linear instead of quadratic cost in the number of steps. The implementation supports a stationary infinite-horizon gain and a finite-horizon gain indexed by the remaining trajectory length. The reported fixed-covariance sweep uses the stationary gain with `horizon=None`; finite-horizon control is an available implementation option, rather than the configuration of that sweep.

After the LQR action is computed, the acceleration command is bounded to remain within a feasible range,

$$\mathbf{u}_k = \text{clip}(-\mathbf{K}_k \mathbf{e}_k, \mathbf{u}_{\min}, \mathbf{u}_{\max}). \quad (3.33)$$

This bound prevents unrealistically large acceleration commands and makes the simulated control behavior more consistent with a physical agent.

The LQR cost also provides a control-oriented performance measure for the closed-loop system. A low localization error alone does not fully describe closed-loop behavior, because an estimate that appears accurate at one time step can still lead to unstable or inefficient control if it produces poor state tracking over time. The LQR cost evaluates the combined effect of state-estimation quality and control effort, making it a suitable metric for analyzing the connection between CSI-based localization and downstream control.

In the full method, the LQR does not operate directly on the raw CSI or on the unfiltered neural network output. Instead, it receives the state estimate from the Kalman filter. This design separates the roles of the modules: the neural network extracts position information from CSI, the Kalman filter produces a temporally consistent state estimate, and the LQR converts that estimate into a control action. The next subsection explains how these modules are connected into a complete feedback loop.

3.4.3 Feedback Loop

The feedback loop combines the channel generator, neural localization model, Kalman filter, and LQR controller into one closed-loop procedure. The key idea is that the agent motion changes the wireless channel, and the changed channel produces the next localization measurement. Therefore, sensing and control are coupled through the physical state of the agent.

At control step k , the true agent state is denoted by $\mathbf{x}_k$, and its horizontal position is

$$\mathbf{p}_k = [x_k, y_k]^T. \quad (3.34)$$

This position is used as the receiver location in the ray-tracing environment. Sionna RT then computes the propagation paths between the configured base station and the current agent position, and converts these paths into a frequency-domain CSI observation,

$$\mathbf{H}_k = \Phi_{\text{RT}}(\mathbf{p}_k), \quad (3.35)$$

where $\Phi_{\text{RT}}(\cdot)$ denotes the ray-tracing-based channel generation process. This expression emphasizes the role of the simulator in the closed loop: after the agent moves, the channel is regenerated from the updated position instead of being sampled independently from a static dataset.

The generated CSI is converted to the normalized real-valued tensor $\bar{\mathbf{X}}_k$ described in Section 3.3.1. The model design produces a position prediction and a coordinate-wise variance prediction,

$$(\hat{\mathbf{y}}_k, \hat{\boldsymbol{\sigma}}_k^2) = f_{\theta}(\bar{\mathbf{X}}_k), \quad (3.36)$$

where both outputs use the normalized label coordinates. The position prediction is scaled and translated into the absolute position observation $\mathbf{z}_k$. Although Eq. (3.27) defines a learned covariance from the variance output, the reported loop discards that output and supplies the independently chosen fixed $\mathbf{R}_k = r\mathbf{I}_2$. The filter combines $\mathbf{z}_k$ with its motion-model prediction to obtain $\hat{\mathbf{x}}_{k|k}$.

The LQR controller then computes the acceleration command from the tracking error between the filtered estimate and the reference state,

$$\mathbf{u}_k = -\mathbf{K}_k(\hat{\mathbf{x}}_{k|k} - \mathbf{x}_k^{\text{ref}}). \quad (3.37)$$

After applying this command, the agent evolves according to the discrete-time dynamics,

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k. \quad (3.38)$$

The updated position $\mathbf{p}_{k+1}$ is then used to generate the next CSI observation through Sionna RT. This creates the closed-loop sequence

$$\mathbf{x}_k \rightarrow \mathbf{H}_k \rightarrow \mathbf{z}_k \rightarrow \hat{\mathbf{x}}_{k|k} \rightarrow \mathbf{u}_k \rightarrow \mathbf{x}_{k+1}. \quad (3.39)$$

This loop captures the interaction between wireless sensing and control. Localization errors affect the filter update and can propagate into control actions. The selected covariance determines the measurement weight, but the reported fixed-covariance filter does not adjust that weight using the network’s predicted uncertainty. The resulting state estimate changes the LQR action, and the action changes the future trajectory of the agent. Since the future trajectory determines the next ray-tracing-based CSI observation, localization errors can influence the later sensing conditions and control performance.

The loop is executed over a reference trajectory. At each step, a channel observation is generated for the current position, the localization output is converted into a Kalman-filter measurement, the LQR computes the next control action, and the motion model updates the agent state. The history of true states, estimated states, control actions, measurement covariances, and control costs is stored for later evaluation. This makes it possible to analyze not only the standalone localization accuracy of the neural estimator, but also how the CSI-based estimator affects the closed-loop behavior of the controlled agent.

4

Experimental Evaluation

This chapter evaluates the proposed CSI-based localization and closed-loop control method. The experiments are designed to assess not only the standalone accuracy of the neural position estimator, but also how its output can be used as a measurement source for tracking and control.

4.1 Evaluation Goals

The evaluation is organized around the four research questions in Section 1.1.5. Each question is linked to the experiments and measures used to assess it:

- **RQ1 – Localization accuracy.** Sections 4.5.1 and 4.5.2 report the Euclidean error distribution and its spatial variation on the random held-out split. Mean error, RMSE, percentiles, and the error CDF characterize both typical accuracy and the high-error tail. The spatial map describes within-scene variation; it does not establish generalization to a new region or environment.
- **RQ2 – CSI availability.** Section 4.5.4 evaluates the same held-out samples under eight fixed joint antenna–subcarrier configurations using the same trained model. This assesses how localization accuracy changes with retained CSI coefficients under the adopted normalization and masking procedure. Because the two resource dimensions change together, the experiment does not separate antenna, subcarrier, and retained-energy effects.
- **RQ3 – Uncertainty quality.** Section 4.5.3 compares predicted covariance with observed errors using predicted and empirical variance quantities, spatial correlations, and NEES diagnostics. Section 4.5.4 additionally examines whether the uncertainty output follows resource-dependent error changes. Poor calibration is treated as an empirical finding, rather than evidence of optimization instability across training runs. These diagnostics assess the learned covariance separately from its potential downstream use.
- **RQ4 – Closed-loop sensitivity.** Section 4.5.5 evaluates tracking RMSE and accumulated LQR cost across fixed diagonal measurement covariances in the simulated feedback loop. The trajectory plot illustrates the reference, true, and estimated motion, while the cost figure shows only accumulated true-state LQR cost; the corresponding table additionally reports estimated-state cost. The main comparison uses fixed covariance because the learned covariance fails the reported consistency check. It assesses sensitivity in the evaluated scenario rather than a generally optimal confidence setting.

The following sections specify the datasets, metrics, implementation, and results

supporting these questions. Table 4.4 compares the residual estimator with an FCNN localization baseline under the same test-input protocol, while documenting unequal training budgets and the absence of repeated training runs. Oracle feedback, unfiltered-measurement control, and model-only prediction are not evaluated, so the closed-loop results do not establish a performance advantage from CSI updates or Kalman filtering over those alternatives. Broader comparisons and robustness tests remain necessary to extend the evidence beyond the present setting.

4.2 Datasets

The experiments in this thesis use a ray-tracing-based dataset generated with the procedure described in the Method chapter. The purpose of the dataset is to provide paired samples between UE positions and the corresponding channel frequency responses observed by the base stations. Each sample therefore contains a physical UE position, the complex CSI generated from that position, and the relative position label used for supervised learning.

The channels are generated with Sionna RT in its built-in street-canyon scene. The simulated wireless scene contains two base stations located at $[-32, 12, 35]^T$ m and $[33, -12, 35]^T$ m. Both base stations use a uniform linear antenna configuration with 32 antenna elements, corresponding to one row and 32 columns in the configured planar array, with vertical polarization and a TR 38.901 antenna pattern. The retained generator revisions pass a spacing argument of 0.05 to `PlanarArray`. In the inspected Sionna RT 1.2.1 API this means 0.05λ , approximately 5 mm at 3 GHz. The original CSI-generation environment is not fully recorded, so historical physical half-wavelength spacing is not established (Appendix A). The receiver is a single-antenna UE. The carrier frequency is 3 GHz and the system bandwidth is 100 MHz. The frequency response is sampled over 400 subcarriers, so the complex channel for one base station has dimension 32×400 before conversion to a real-valued neural network input. The UE height is fixed at 1.5 m, and the localization task is restricted to the horizontal plane.

The main dataset consists of 20,000 randomly sampled UE positions. For each sample, the horizontal coordinates are drawn from the rectangular region $x \in [-80, 80]$ m and $y \in [-8, 8]$ m, while the height remains fixed. A Sionna RT path solver is then used to compute the propagation paths between the transmitters and the UE, including line-of-sight, reflection, refraction, diffraction, and diffuse components according to the simulation configuration. These paths are converted into channel frequency responses for both base stations. Although channels from both base stations are stored, the localization experiments in this thesis use the CSI from the first base station as the main input, because the method is evaluated primarily in a single-base-station localization setting.

The labels are defined as the two-dimensional UE position relative to the first base station. This is consistent with the method definition in Eq. (3.4) and ensures that all supervised targets are expressed in the same coordinate frame as the single-BS estimator. The complex channel is represented by its real and imaginary components, producing a real-valued tensor of size $2 \times 32 \times 400$ for each BS1 sample. Before being passed to the neural network, the CSI tensor is normalized as described in

Table 4.1: Dataset configuration used in the experimental evaluation.

Configuration item	Value
Number of random samples	20,000
Training samples	16,000
Validation samples	2,000
Held-out test samples	2,000
UE sampling region	$x \in [-80, 80]$ m, $y \in [-8, 8]$ m
UE height	1.5 m
Carrier frequency	3 GHz
Bandwidth	100 MHz
Number of subcarriers	400
BS antenna elements	32
CSI tensor for one BS	$2 \times 32 \times 400$
Position label	Relative horizontal position to BS1

Section 3.3.1, so that the model focuses on spatial-frequency structure rather than absolute channel power alone.

The 20,000 random samples are randomly permuted with a fixed seed and divided into three disjoint subsets according to an 80/10/10 split: 16,000 training samples, 2,000 validation samples and 2,000 held-out test samples. The validation subset is used only to monitor convergence and to select the best checkpoint, whereas the test subset is used exclusively for the final evaluation and is never seen during training or model selection. The three index sets are stored in a split file and their disjointness is asserted programmatically at the beginning of every evaluation run, so all test metrics reported in this chapter, including the localization-error CDF, are computed on the same 2000 unseen samples. Spatial maps are derived from these held-out predictions. The evaluators aggregate samples by position and can interpolate and smooth the resulting statistics onto a display grid; they do not generate additional ray-traced samples for these maps. The display grid is therefore a visualization of the sampled evidence, not a denser independent test set. Appendix A distinguishes raw spatial statistics from display settings.

For the antenna and subcarrier resource study, the raw dataset is not regenerated. Instead, resource-limited inputs are produced during data loading by masking parts of the full CSI tensor. The available subcarrier counts are 50, 100, 150, 200, 250, 300, 350, and 400, and the corresponding antenna counts are 4, 8, 12, 16, 20, 24, 28, and 32. The active antennas and subcarriers are selected around the center of the array and frequency grid, while the remaining entries are set to zero. This design allows the same underlying channel samples and position labels to be used when comparing different sensing-resource configurations.

The trajectory and closed-loop control experiments use the trained neural estimator together with newly generated channel observations along the simulated UE motion. At each time step, the current UE position is inserted into the same Sionna RT scene, a channel response is generated, and the trained model produces a position measurement for the Kalman filter and LQR controller. The closed-loop run

uses the same named scene and nominal radio settings as the inspected dataset generator, with its current software environment and array spacing explicitly archived. The original dataset has no equivalent generation manifest, so exact equivalence of historical and current channel generation is not established. The loop tests the saved estimator sequentially rather than only on independent random samples.

4.3 Evaluation Measures

The evaluation uses complementary measures because the thesis studies both standalone localization and closed-loop control. The localization metrics quantify how accurately the neural estimator predicts the UE position from CSI. The uncertainty metrics evaluate whether the predicted variance is meaningful as a measure of confidence. The control metrics then measure whether the resulting position estimates are useful when they are inserted into a Kalman-filtered LQR feedback loop.

For a test sample i , let $\mathbf{p}_i \in \mathbb{R}^2$ denote the ground-truth horizontal UE position label and let $\hat{\mathbf{p}}_i \in \mathbb{R}^2$ denote the predicted position mean produced by the neural estimator after converting the normalized network output back to metres. The two-dimensional localization error is defined as

$$e_i = \|\hat{\mathbf{p}}_i - \mathbf{p}_i\|_2, \quad (4.1)$$

where e_i is measured in metres. This scalar error is the main per-sample measure used in the offline localization experiments, because it directly describes the distance between the estimated and true UE positions in the horizontal plane.

Aggregate localization accuracy is reported using the mean absolute error (MAE), the root mean squared error (RMSE), and selected percentiles of the empirical error distribution. For N evaluation samples, these are computed as

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N e_i, \quad (4.2)$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N e_i^2}. \quad (4.3)$$

The MAE gives the average positioning error, while the RMSE gives larger weight to samples with high error. The median, 90th percentile, and 95th percentile are also reported to describe the spread and tail of the error distribution. In addition, the empirical cumulative distribution function (CDF) of e_i is plotted. The CDF is useful because it shows what fraction of test samples fall below any chosen error threshold, rather than reducing the full test set to a single number.

The spatial error maps use the same per-sample error in Eq. (4.1), but aggregate it according to the physical UE position. The sampled area is divided into spatial bins, and the mean localization error is computed for each bin with a sufficient number of evaluation samples. This produces a heat map that indicates whether the estimator performs uniformly across the environment or whether errors are concentrated in specific regions. For the uncertainty analysis, analogous spatial maps are generated for the predicted total variance and for NEES-related quantities, which makes

it possible to compare where the model is inaccurate with where it claims to be uncertain.

The variance head predicts a diagonal position covariance for each sample. Let

$$\mathbf{\Sigma}_i = \text{diag}(\sigma_{x,i}^2, \sigma_{y,i}^2) \quad (4.4)$$

be the predicted covariance in physical units. The total predicted variance is defined as $\sigma_{x,i}^2 + \sigma_{y,i}^2$ and is compared with the squared localization error e_i^2 . The evaluation reports the mean total variance, the empirical mean squared error energy, and the ratio between these two quantities. It also reports the Pearson and Spearman correlations between cell-averaged squared error and cell-averaged predicted variance. These correlation measures are used to test whether regions with larger predicted uncertainty also tend to be regions with larger empirical localization error.

Calibration of the predicted covariance is evaluated using the normalized estimation error squared (NEES). For each sample, the NEES is computed as

$$\text{NEES}_i = (\hat{\mathbf{p}}_i - \mathbf{p}_i)^T \mathbf{\Sigma}_i^{-1} (\hat{\mathbf{p}}_i - \mathbf{p}_i). \quad (4.5)$$

With the diagonal covariance used in this thesis, this is equivalent to $e_{x,i}^2/\sigma_{x,i}^2 + e_{y,i}^2/\sigma_{y,i}^2$, where $e_{x,i}$ and $e_{y,i}$ are the coordinate-wise localization errors. For a well-calibrated two-dimensional Gaussian error model, NEES follows a chi-square distribution with two degrees of freedom and has expected value 2. The average NEES (ANEES) over N samples is therefore used as a compact calibration measure,

$$\text{ANEES} = \frac{1}{N} \sum_{i=1}^N \text{NEES}_i. \quad (4.6)$$

Values far below 2 indicate that the predicted covariance is too large relative to the observed errors, while values far above 2 indicate overconfident uncertainty estimates. In the present thesis, this metric is mainly used as a diagnostic for the learned variance head rather than as the primary success criterion, since the main closed-loop experiments use fixed measurement variances.

For training and model selection, the validation loss is recorded together with the validation localization error. During location-oriented training, the loss corresponds to the mean squared error between the normalized predicted position and the normalized target. During variance-oriented training, the objective is the Gaussian negative log-likelihood induced by the predicted variance. These losses are useful for monitoring optimization, but they are not by themselves the main experimental measures, because the thesis conclusions depend on physical localization error, uncertainty calibration, and closed-loop tracking behavior.

The closed-loop experiments evaluate a four-dimensional state over discrete time step k ,

$$\mathbf{x}_k = [x_k, y_k, v_{x,k}, v_{y,k}]^T. \quad (4.7)$$

Let $\mathbf{x}_k^{\text{ref}}$ be the reference state and let $\mathbf{x}_k^{\text{true}}$ be the simulated true state. The tracking error is measured on the horizontal position components,

$$e_k^{\text{track}} = \left\| \mathbf{x}_{k,1:2}^{\text{true}} - \mathbf{x}_{k,1:2}^{\text{ref}} \right\|_2, \quad (4.8)$$

and the trajectory-level tracking RMSE is

$$\text{RMSE}_{\text{track}} = \sqrt{\frac{1}{T} \sum_{k=1}^T (e_k^{\text{track}})^2}. \quad (4.9)$$

This measure reflects the actual control performance, because it compares the true controlled trajectory with the desired reference trajectory rather than comparing only the filter estimate with the reference.

The LQR cost is also recorded to quantify the trade-off between tracking accuracy and control effort. For a state $\mathbf{x}_k$, reference state $\mathbf{x}_k^{\text{ref}}$, and control input $\mathbf{u}_k$, the instantaneous cost is

$$c_k = (\mathbf{x}_k - \mathbf{x}_k^{\text{ref}})^T \mathbf{Q}_{\text{LQR}} (\mathbf{x}_k - \mathbf{x}_k^{\text{ref}}) + \mathbf{u}_k^T \mathbf{R}_{\text{LQR}} \mathbf{u}_k. \quad (4.10)$$

Two versions of this cost are reported in Table 4.7. Figure 4.9 plots only the accumulated true-state LQR cost. The estimated cost evaluates tracking using the Kalman filter state estimate supplied to the controller. The true cost uses the simulated true state and therefore reflects the realized tracking performance. Comparing these two costs helps reveal the effect of localization and filtering errors on the closed-loop system: a controller may appear successful in the estimated state while still producing a larger cost when evaluated against the true trajectory.

4.4 Implementation Details

Detailed configurations and provenance qualifications are provided in Appendix A. Code, required artifacts, and reproduction commands are listed in Appendix B.

The implementation consists of three connected parts: ray-tracing-based channel generation, supervised neural network training, and closed-loop tracking and control evaluation. The channel generation is implemented in Python using Sionna RT, while the learning pipeline is implemented with PyTorch, Torchvision, and PyTorch Lightning. This separation makes it possible to generate an offline dataset for reproducible supervised training and then reuse the trained estimator in online trajectory simulations.

For the neural localization model, the primary architecture is the ResNet-based estimator introduced in Section 3.3.2. The implementation uses a Torchvision ResNet34 backbone whose first convolutional layer is modified to accept the two-channel CSI representation consisting of the real and imaginary channel components. The original classification layer is replaced by an identity mapping, and two lightweight output heads are attached to the extracted feature vector. The first head predicts the two-dimensional position mean, while the second head predicts the two-dimensional log-variance used for uncertainty estimation. The input batch therefore has shape $B \times 2 \times 32 \times 400$, and the two model outputs each have shape $B \times 2$.

Training is performed with PyTorch Lightning using shuffled training mini-batches and unshuffled validation and test loaders. The reported ResNet run has two stages: 500 epochs of position training, followed by 100 epochs of variance-head training initialized from the selected stage-1 checkpoint. Both stages use Adam with learning

Table 4.2: Main settings for the reported ResNet training and control implementation.

Item	Setting
Frameworks	Python, PyTorch, PyTorch Lightning, Torchvision, Sionna RT
Primary model	Modified ResNet34 backbone with location and variance heads
Input tensor shape	$B \times 2 \times 32 \times 400$
Output tensor shape	Position mean $B \times 2$ and log-variance $B \times 2$
Optimizer	Adam
Learning rate	10^{-4}
Weight decay	10^{-4}
Batch size	32
Training epochs	500 position stage; 100 variance stage
Precision and device	16-bit mixed precision on one CUDA GPU
Control sampling time	$\Delta t = 0.1$ s
LQR state weight	$\mathbf{Q}^{\text{LQR}} = \text{diag}(1, 1, 0, 0)$
LQR control weight	$\mathbf{R}^{\text{LQR}} = \text{diag}(0.1, 0.1)$
Logging	TensorBoard
Checkpoint selection	Minimum logged validation localization error (stage 1); minimum logged validation NLL (stage 2)

rate 10^{-4} , weight decay 10^{-4} , and batch size 32, on one CUDA GPU with 16-bit mixed precision. Checkpoint selection minimizes the logged validation localization error in stage 1 and logged validation NLL in stage 2. The offline results use the selected stage-2 checkpoint, identified in Appendix A. TensorBoard records losses, localization error, and predicted variance.

During training, each CSI sample is normalized before being passed to the network. In addition, the data loader applies the antenna and subcarrier masking procedure described in Section 3.3.1. A resource configuration is selected from the predefined set of antenna and subcarrier counts, and the unavailable CSI entries are set to zero. This implementation detail is important because it allows the model to encounter different resource budgets during training and enables later experiments on how localization performance changes when fewer antennas or subcarriers are available. The variance head is implemented by predicting a log-variance vector. The predicted log-variance is lower-bounded at -2 before exponentiation, which ensures that the covariance used in uncertainty-related calculations remains positive. The implementation supports both a location-oriented training mode based on mean squared error and a variance-oriented mode based on a Gaussian negative log-likelihood objective. In the reported evaluation, the learned variance is mainly analyzed as an auxiliary uncertainty output, while the main localization results are based on the predicted position mean.

The closed-loop sweep uses the canyon stage-2 checkpoint identified in Appendix A,

with gradient computation disabled. Its source snapshot, checkpoint hash, command, software inventory, and saved arrays are archived as documented in that appendix. The state contains horizontal position and velocity, with sampling interval 0.1 s. LQR weights are $\mathbf{Q}^{\text{LQR}} = \text{diag}(1, 1, 0, 0)$ and $\mathbf{R}^{\text{LQR}} = 0.1\mathbf{I}_2$; the stationary infinite-horizon gain is used and acceleration is clipped to $[-5, 5]$ per axis. Each step generates CSI at the true UE position, obtains a neural position measurement, updates the Kalman filter, and computes the applied control.

The five fixed-covariance settings share the checkpoint, full 32×400 CSI input, initial conditions, reference, controller weights, and process-noise setting. NumPy and PyTorch seeds are reset to 0 before each rollout, and saved state-transition residuals verify paired process noise within numerical tolerance. The realized paths and resulting CSI can differ because covariance changes the feedback. Per-step true and estimated states, measurements, controls, covariance, reference, and both costs are retained for metric verification and figure generation. Appendix B gives the explicit command.

4.5 Experiments

This section presents the experiments used to evaluate the proposed CSI-based localization and closed-loop control pipeline. The experiments follow the goals stated at the beginning of this chapter. First, the offline localization accuracy of the trained neural estimator is measured on a held-out test split. Second, the spatial distribution of the localization error is analyzed to determine whether the estimator behaves uniformly across the simulated area. Third, the uncertainty output of the variance head is compared with the observed localization errors. Fourth, the sensing resources allocated to one measurement are varied over the eight antenna and subcarrier configurations defined in Section 3.3.1, in order to quantify how localization accuracy and uncertainty calibration depend on the resource budget. Finally, the trained estimator is inserted into the Kalman-filtered LQR loop, and the assumed measurement variance is varied to study how localization confidence affects closed-loop tracking.

4.5.1 Localization Accuracy

The first experiment evaluates the standalone CSI-based position estimator. The trained ResNet-based model is applied to a held-out test split that was not used during training or validation. For each test sample, the model receives the normalized two-channel CSI tensor from BS1 and predicts the two-dimensional UE position relative to BS1. The prediction is then converted back to metres and compared with the ground-truth position label using the Euclidean localization error defined in Eq. (4.1).

The results in Table 4.3 show that the CSI representation carries enough spatial information for sub-metre localization on average, but also that the error distribution is strongly right-skewed. The median error is 0.673 m while the mean error is 0.857 m, so half of the test positions are localized more accurately than the average value suggests. The RMSE of 1.204 m is about 1.4 times the MAE, because squaring weights the large errors more strongly; together with the gap between the mean

Table 4.3: Localization accuracy of the trained CSI-based position estimator on the held-out test split.

Metric	Value
Number of test samples	2000
MAE	0.857 m
RMSE	1.204 m
Median error	0.673 m
90th percentile	1.559 m
95th percentile	2.184 m

Table 4.4: Localization comparison on the same 2000 held-out test positions and resource masks. All errors are in metres. The models use unequal training budgets and one training run each.

Metric	ResNet-34	FCNN
Mean Euclidean error	0.857	2.033
RMSE	1.204	2.732
Median error	0.673	1.513
90th percentile	1.559	4.174
95th percentile	2.184	5.486

and the median, this confirms the skew. The 90th and 95th percentiles quantify that tail directly: one test sample in ten is misplaced by more than 1.559 m, and one in twenty by more than 2.184 m. These aggregate values are obtained with the resource mask active, so they describe the estimator averaged over the eight resource configurations of Eq. (3.7); the per-configuration results are reported separately in Section 4.5.4. The estimator is therefore accurate for the bulk of the sampled area, but it still produces occasional gross errors, and the spatial origin of those errors is examined in the next experiment.

Fig. 4.1 complements the scalar values in Table 4.3 by showing the full empirical error distribution. Unlike selected summary statistics, the CDF exposes the tail behavior of the localization error. Here the CDF shows that 73.3% of the test samples are localized within 1 m and 94.2% within 2 m, while the distribution extends to a maximum of 20.7 m. Fewer than 0.5% of the samples exceed 5 m, so the high-error tail is sparse, but it is heavy enough to dominate the RMSE and to matter for closed-loop operation, where a single large outlier can perturb the filtered state.

Comparison with the fully connected baseline. The FCNN in Section 3.3.3 uses the same 2000 test positions, normalization, and paired resource-mask sequence as the ResNet. Table 4.4 compares their mean Euclidean error, RMSE, median, and upper percentiles. The FCNN is evaluated for localization only.

The ResNet has a lower mean error and lower reported percentiles in this evaluation. Fig. 4.2 shows both empirical error distributions. The FCNN nevertheless has a smaller error on 19.95% of the paired test samples, so the aggregate difference does not imply that the ResNet is better at every test position and resource action.

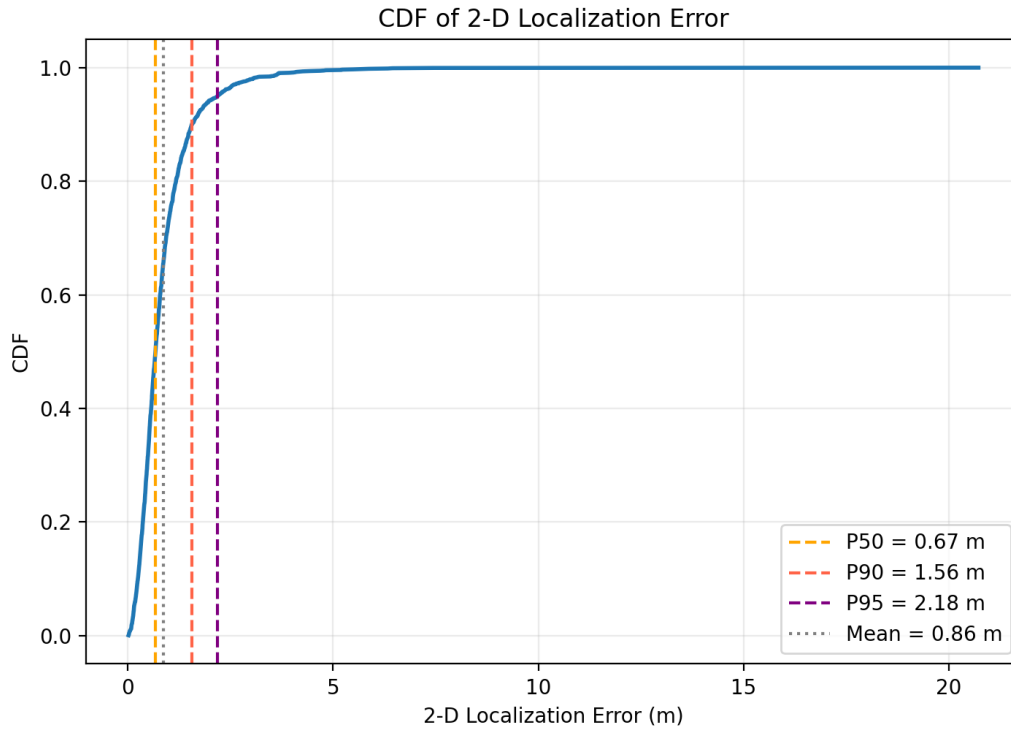


Figure 4.1: Empirical CDF of the two-dimensional localization error on the held-out test split. The curve shows the fraction of test samples whose localization error is below a given threshold.

The comparison supports lower observed localization error for the reported ResNet checkpoint under this within-scene evaluation protocol. It does not isolate the effect of preserving two-dimensional adjacency: the architectures, parameter counts, and optimization budgets differ, with 500 ResNet localization epochs and 147 FCNN epochs. One run per model also leaves seed variability unmeasured. Matched-budget training and repeated runs would be needed for a stronger attribution of the performance difference to architectural design.

4.5.2 Spatial Error Analysis

The second experiment examines where the localization errors occur in the physical scene. The same test-set predictions used in the localization accuracy experiment are grouped according to the ground-truth UE position. The mean localization error is then visualized over the sampled area. This experiment is included because aggregate metrics do not show whether the model performs equally well throughout the environment. In a ray-tracing-based channel, the difficulty of localization can vary with position because the multipath structure, angular separation, and received channel signatures depend on the local geometry.

The spatial error map in Fig. 4.3 shows how localization accuracy varies relative to BS1 and across the sampled UE region. Regions with low error indicate positions where the CSI pattern gives the neural network a distinctive and stable localization cue. Regions with higher error indicate positions where different UE locations can

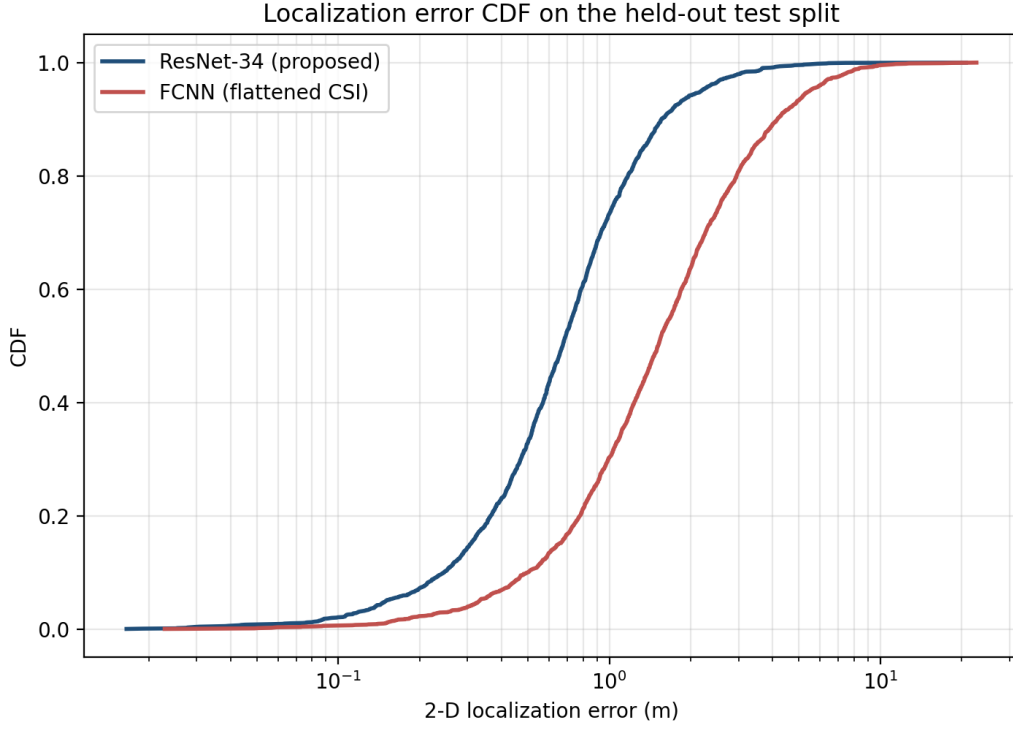


Figure 4.2: Empirical localization-error CDFs for the ResNet and FCNN on the same held-out positions and resource masks. The horizontal axis is logarithmic.

produce more similar channel observations or where the learned model generalizes less well. This analysis is therefore useful for identifying whether the localization pipeline fails randomly or whether the errors are connected to the propagation geometry.

A useful observation from this experiment is that spatially resolved evaluation gives a more practical view of the estimator than a single global RMSE. In the closed-loop setting, persistent errors in a specific part of the trajectory can affect the Kalman filter and controller more strongly than isolated errors at random positions. The spatial heat map therefore serves as a bridge between the offline localization experiment and the trajectory-level control experiment: it shows where the neural measurement source is expected to be more or less reliable.

4.5.3 Uncertainty Analysis

The third experiment evaluates the variance output of the neural estimator. For each test sample, the model predicts both a position mean and a diagonal covariance. The predicted total variance is compared with the empirical squared localization error, and the NEES metric in Eq. (4.5) is used to assess calibration. The purpose of this experiment is not only to measure whether the variance head gives numerically accurate covariances, but also to determine whether it contains useful information about where the estimator is likely to be wrong.

The diagnostics in Table 4.5 show two weaknesses of the variance head. First, the average predicted variance is about 2.5 times the empirical squared error energy, and

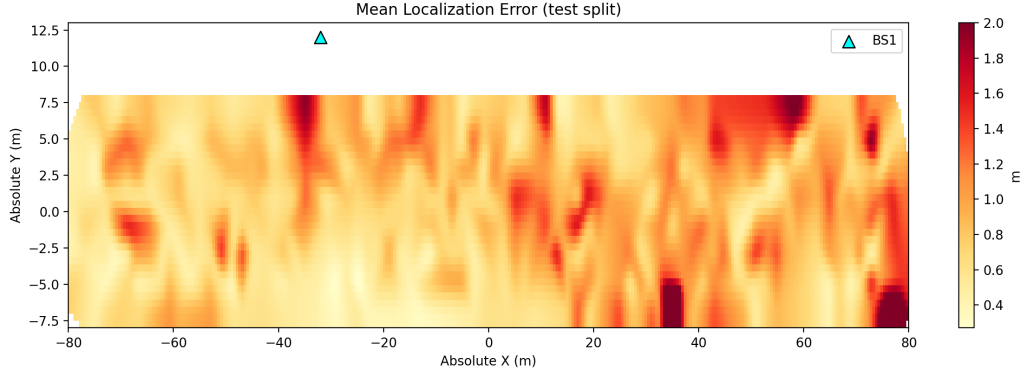


Figure 4.3: Spatial map of mean localization error, constructed from spatially grouped held-out test predictions. The regular display grid interpolates the sampled cell statistics.

Table 4.5: Uncertainty diagnostics for the variance head on the held-out test split.

Metric	Value
Mean total predicted variance $\mathbb{E}[\sigma_x^2 + \sigma_y^2]$	3.654 m ²
Empirical mean squared error energy $\mathbb{E}[e_x^2 + e_y^2]$	1.449 m ²
Empirical-to-predicted variance ratio	0.396
Pearson correlation between cell MSE and variance	0.070
Spearman correlation between cell MSE and variance	0.173
ANEES	1.351
95% ANEES consistency interval ($n = 2000$)	[1.913, 2.089]
Calibrated ANEES target	2.000

the observed ANEES of 1.351 lies below the 95% consistency interval [1.913, 2.089] obtained for $n = 2000$ samples. The predicted covariance is therefore conservative: the model is less confident than the observed errors would justify under a calibrated Gaussian error model. Second, the cell-wise correlations between predicted variance and empirical squared error are weak, 0.070 for Pearson and 0.173 for Spearman, so the variance output does not follow the spatial distribution of the error closely either. Taken together, these two observations indicate that the variance head has learned little beyond a roughly constant uncertainty level, and Section 4.5.4 shows that this level is also insensitive to the sensing-resource configuration.

Fig. 4.4 and Fig. 4.5 provide the visual interpretation of the uncertainty results. The variance heat map and the localization-error heat map in Fig. 4.3 reveal whether predicted uncertainty follows the spatial error pattern. If the two maps have similar spatial structure, the variance output can still be useful as a relative confidence indicator, even if the absolute scale is not calibrated. The NEES plot then reveals the calibration issue: because the ANEES is below the theoretical consistency interval, the covariance values are too large relative to the observed prediction errors. For this reason, the learned uncertainty is treated as an auxiliary diagnostic in the remainder

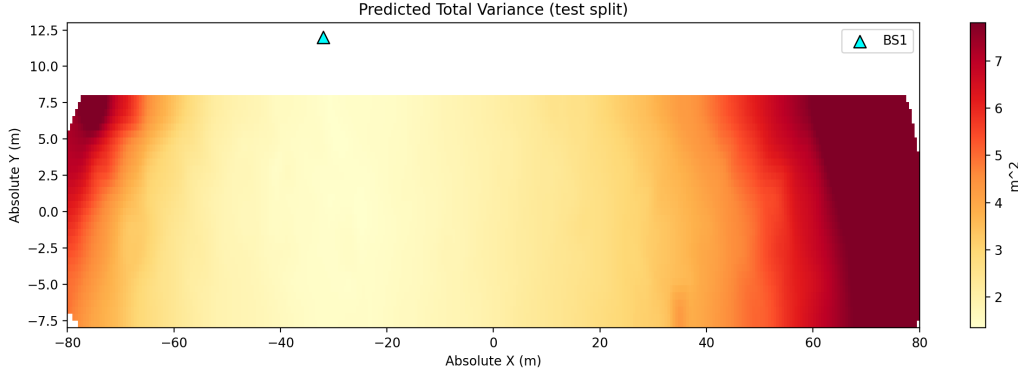


Figure 4.4: Spatial map of predicted total variance, $v_x + v_y$, constructed from the held-out test predictions. The colors visualize spatially aggregated uncertainty estimates.

of the thesis, while the closed-loop experiments use controlled fixed measurement variances.

4.5.4 Sensitivity to Sensing-Resource Allocation

This experiment evaluates how the localization performance depends on the amount of sensing resources allocated to one measurement. This is the experiment that supports the resource-configuration claim stated in the introduction, and it uses the masking mechanism defined in Section 3.3.1. During training, the resource action is drawn uniformly for every sample, so the aggregate results reported in the previous sections describe the estimator averaged over the eight configurations rather than at any single one. Here the random draw is replaced by a fixed action so that each configuration can be measured separately.

The experimental setup is deliberately minimal, so that the resource budget is the only quantity that changes. The same trained checkpoint used in the preceding experiments is evaluated without any retraining or fine-tuning; only forward passes are performed. For each action $a \in \{1, \dots, 8\}$ of Eq. (3.7), all 2000 samples of the held-out test split are masked with that same action, passed through the identical forward path used in the previous experiments, and scored with the metrics of Eqs. (4.1), (4.2) and (4.5). Because the sample indices, the normalization, the model weights and the metric definitions are held fixed across the eight runs, any difference between rows is attributable to the resource configuration alone. The results are summarized in Table 4.6 and visualized in Fig. 4.6 and Fig. 4.7.

Localization error decreases as the smallest masks are enlarged, then reaches a broad plateau. The 4×50 configuration gives mean error 1.878 m and P90 3.455 m. The lowest observed mean error is 0.626 m at 24×300 , whereas the lowest P90 is 1.085 m at 20×250 ; P90 is 1.088 m at 24×300 . The full input gives mean error 0.681 m and P90 1.208 m, so using every coefficient does not improve either metric in this evaluation. This non-monotonic behavior is not explained by the sweep alone. Shared training across masks and changes in retained input energy are possible contributors, but no ablation or repeated training runs isolate their effects.

This accuracy plateau is the main finding of the resource experiment. The $24 \times$

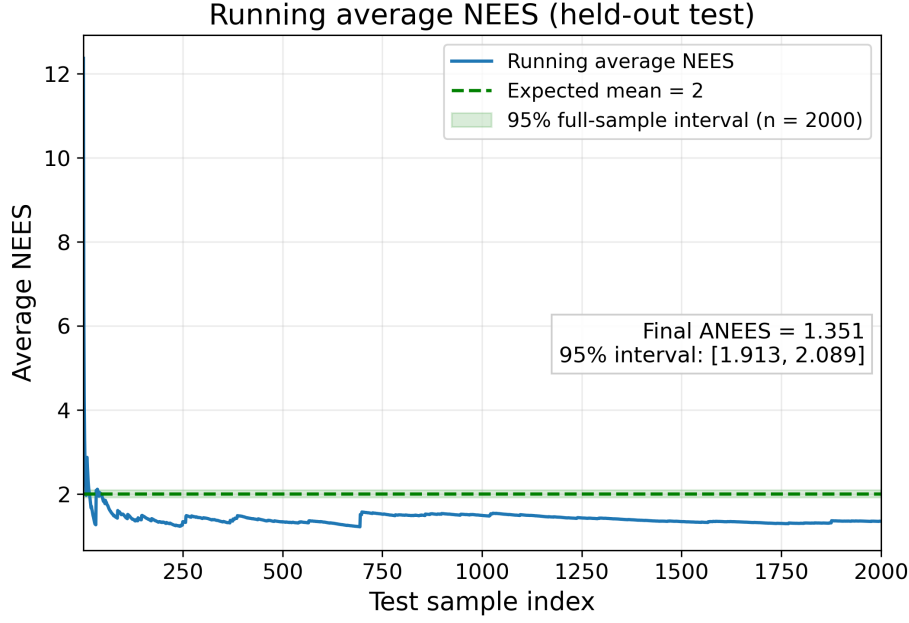


Figure 4.5: Running average NEES for the learned covariance over the 2000 held-out test samples. The target is 2; the shaded 95% consistency interval [1.913, 2.089] applies to the full sample count. The final ANEES is 1.351.

300 mask retains 56.25% of the full CSI coefficients and gives the lowest observed mean localization error, 0.626 m, under the adopted normalization and masking procedure. The result is consistent with the L-2 motivation that position information can be extracted without reconstructing all CSI coefficients. It measures input selection for the localization model, not compression of a transmitted message. The experiment does not establish that this mask preserves closed-loop tracking, define a generally optimal resource allocation, or quantify savings in bitrate, latency, energy, or inference computation.

The uncertainty results show weak adaptation to the resource budget at the aggregate level. The square root of the mean predicted total variance changes by less

Table 4.6: Localization accuracy and uncertainty calibration for the eight sensing-resource configurations, evaluated on the same held-out test split of 2000 samples with a single trained model.

K_a^a	K_a^f	ρ_a	MAE [m]	RMSE [m]	P90 [m]	Predicted std [m]	ANEES
4	50	0.016	1.878	3.106	3.455	1.942	7.69
8	100	0.063	1.022	1.250	1.826	1.931	1.54
12	150	0.141	0.764	0.937	1.353	1.919	0.84
16	200	0.250	0.666	0.825	1.165	1.912	0.67
20	250	0.391	0.630	0.759	1.085	1.901	0.59
24	300	0.563	0.626	0.744	1.088	1.895	0.56
28	350	0.766	0.641	0.756	1.122	1.892	0.55
32	400	1.000	0.681	0.799	1.208	1.894	0.58

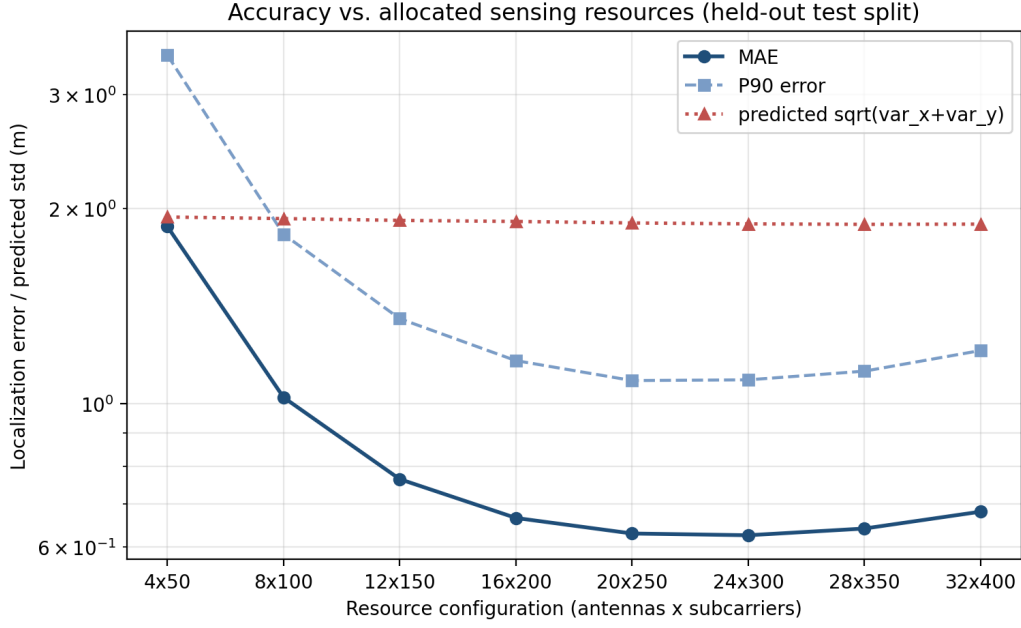


Figure 4.6: Mean and 90th-percentile localization error across the eight joint antenna–subcarrier configurations. The square root of mean predicted total variance is shown for comparison.

than 3%, ranging from 1.892 m to 1.942 m, while mean localization error changes by about a factor of three. At 4×50 , ANEES is 7.69, indicating that the covariance is too small relative to the squared position residuals on average. At the largest budgets, ANEES is about 0.55–0.58, indicating conservative covariance on average. These values do not translate directly into a single measurement-accuracy radius. The 8×100 configuration is closest to the target of 2, but its ANEES of 1.536 is still below the nominal 95% consistency interval $[1.913, 2.089]$; none of the eight configurations falls inside that interval.

The training protocol is one possible contributor to this mismatch, but the experiment does not establish its cause. Although no separate action identifier is supplied, the zero-filled mask makes the resource configuration implicitly observable in the input. Randomizing masks therefore does not force the network to predict only a variance averaged over budgets. Explicit action conditioning or calibration fitted separately for each configuration are candidate improvements to test, not validated requirements or remedies. Since Kalman filtering uses covariance to weight measurements, the observed inconsistency motivates validation across resource settings before using the learned covariance in an adaptive control loop. Antenna count, frequency span, and retained input energy change together in this sweep, so their individual effects remain unresolved.

4.5.5 Closed-loop Control and Measurement-Variance Sensitivity

The final experiment evaluates covariance sensitivity in the CSI-based Kalman-filter and LQR loop. It uses the same canyon stage-2 checkpoint as the offline evaluation,

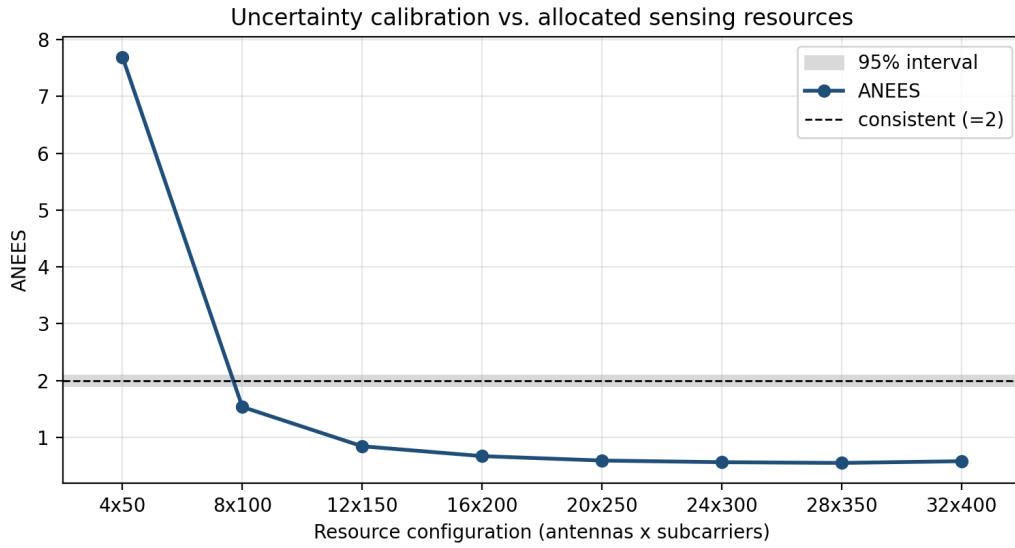


Figure 4.7: ANEES across the eight joint antenna–subcarrier configurations. The smallest mask is overconfident on average, whereas the larger masks are conservative. None of the configurations lies inside the nominal 95% consistency interval.

Table 4.7: Closed-loop performance using the canyon stage-2 checkpoint and stationary LQR gain. Each setting uses one 160 s rollout with seed 0 and paired process noise.

r [m ²]	True LQR cost	Estimated LQR cost	Tracking RMSE [m]
0.01	873.68	159.27	0.715
0.1	825.44	79.72	0.709
1	750.27	32.87	0.683
10	820.81	23.51	0.716
100	1800.51	25.29	1.061

full 32×400 CSI, and a straight reference $(-80 + t, 0)$ m at 1 m/s for 160 s. Each variance has one rollout with seed 0 and 1600 recorded samples. The stationary LQR gain, initial state, process-noise setting, and paired noise realization are held fixed. The run configuration and saved arrays are documented in Appendix A. Because the learned covariance fails the consistency check, the filter uses $R = rI_2$ for $r \in \{0.01, 0.1, 1, 10, 100\}$ m². Changing R affects the realized path and the CSI encountered, so the sweep tests feedback sensitivity rather than filtering identical observations. Small r increases measurement weight; large r increases model-prediction weight.

The lowest observed tracking RMSE is 0.683 m at $r = 1$ m², which also gives the lowest true-state cost, 750.27 (Table 4.7). This intermediate setting performs better than the smallest and largest tested variances in this rollout. The result is consistent with balancing measurement correction against model prediction, but does not identify the mechanism or establish a generally optimal covariance. Figure 4.8 shows this same sweep rollout rather than a separate experiment.

The estimated-state and true-state costs lead to different assessments. From $r = 1$ to

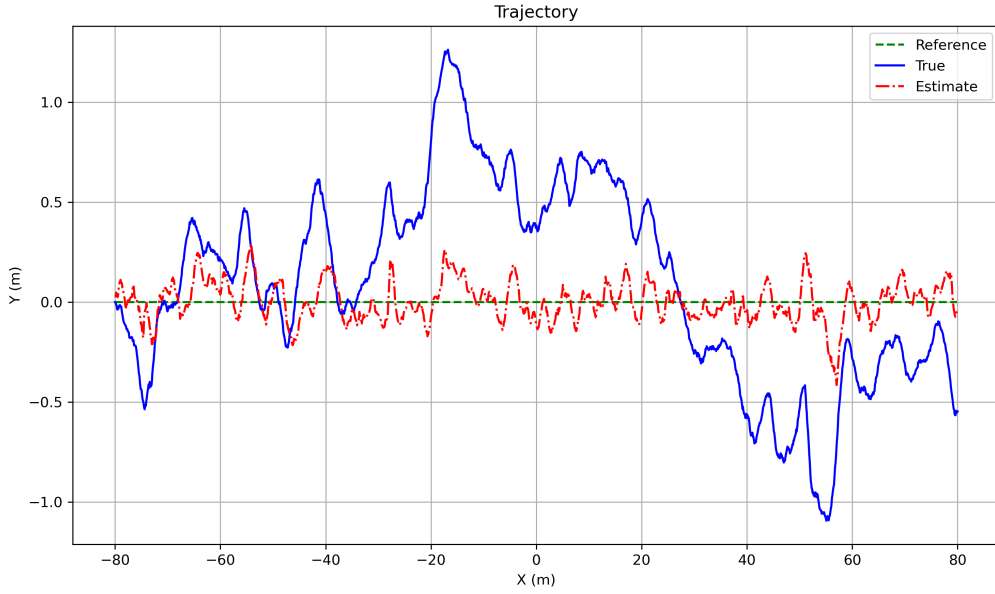


Figure 4.8: Reference, simulated true, and estimated trajectories for $r = 1 \text{ m}^2$, the lowest observed tracking RMSE in the single-seed sweep in Table 4.7.

$r = 10$, estimated cost falls while true cost and tracking RMSE rise (Table 4.7). The controller’s internal state can therefore appear closer to the reference even as realized tracking worsens. Figures 4.9 and 4.10 assess the realized outcome directly. These results establish covariance sensitivity in one configured simulation. They do not establish gains from CSI updates or filtering, or proximity to ideal feedback, because oracle, unfiltered-measurement, and model-only controls are absent. Multiple seeds and trajectory geometries are also needed to assess robustness.

4.6 Societal, Ethical, and Sustainability Considerations

The practical value of CSI-based state estimation depends on its effects on people, controlled systems, and the environment. The following assessment distinguishes potential applications from the evidence obtained in this simulation study.

Societal value. CSI-based localization could support tracking of mobile equipment and robots using wireless infrastructure. The present experiments demonstrate position estimation and feedback tracking within one simulated scene, providing a basis for investigating such applications. They do not establish improvements in public safety, productivity, or access to services. Moreover, the spatially uneven errors in Fig. 4.3 suggest that service quality could differ across operating areas. A deployment would need to characterize those differences and provide alternative sensing where performance is inadequate.

Privacy, consent, and dual use. The dataset contains synthetic UE positions and simulated channels; the reported experiments do not collect personal trajectories. In a real system, however, CSI-derived positions and motion histories could reveal

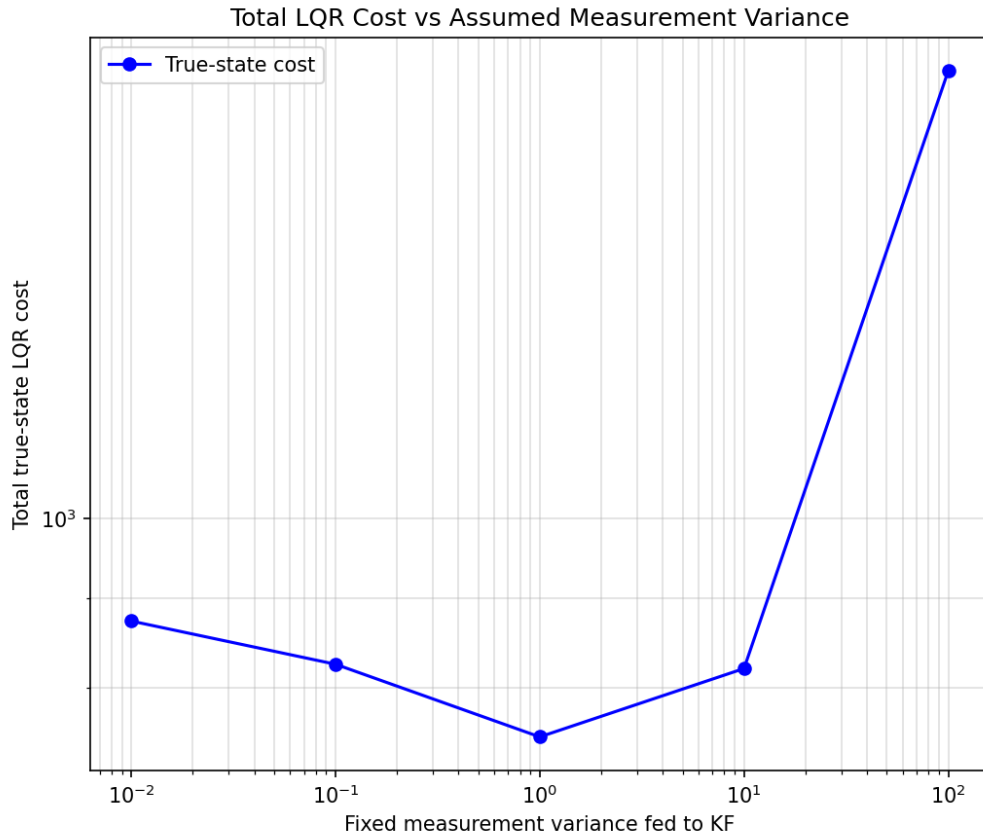


Figure 4.9: Accumulated true-state LQR cost across the five fixed measurement variances, using the simulated true state and applied controls. Only true-state cost is plotted; estimated-state cost is reported in Table 4.7.

where people or their devices are located. Using radio observations instead of camera images does not remove this privacy concern. The NIST AI Risk Management Framework identifies privacy, individual control, and accountability as considerations throughout system development and use [40]. For this application, responsible deployment should define authorized sensing purposes, inform affected users, obtain appropriate consent or authorization, restrict access, and limit retention of CSI and trajectories. The same localization capability could assist legitimate operations or enable unauthorized tracking. Publishing synthetic data and documenting limitations supports scrutiny, while future release of real measurements would require a separate privacy assessment.

Safety and accountability. The localization-error CDF includes outliers reaching 20.7 m, and the learned covariance fails the reported consistency check (Fig. 4.1 and Table 4.5). Such errors could distort a filtered state and the resulting control input. The fixed-covariance sweep also shows that lower estimated control cost need not mean lower true cost (Table 4.7). Consequently, favorable average localization or tracking error is insufficient evidence of safe operation. Before use near people or valuable equipment, the system would need validated anomaly handling, bounds on its operating conditions, and a fallback response when measurements become

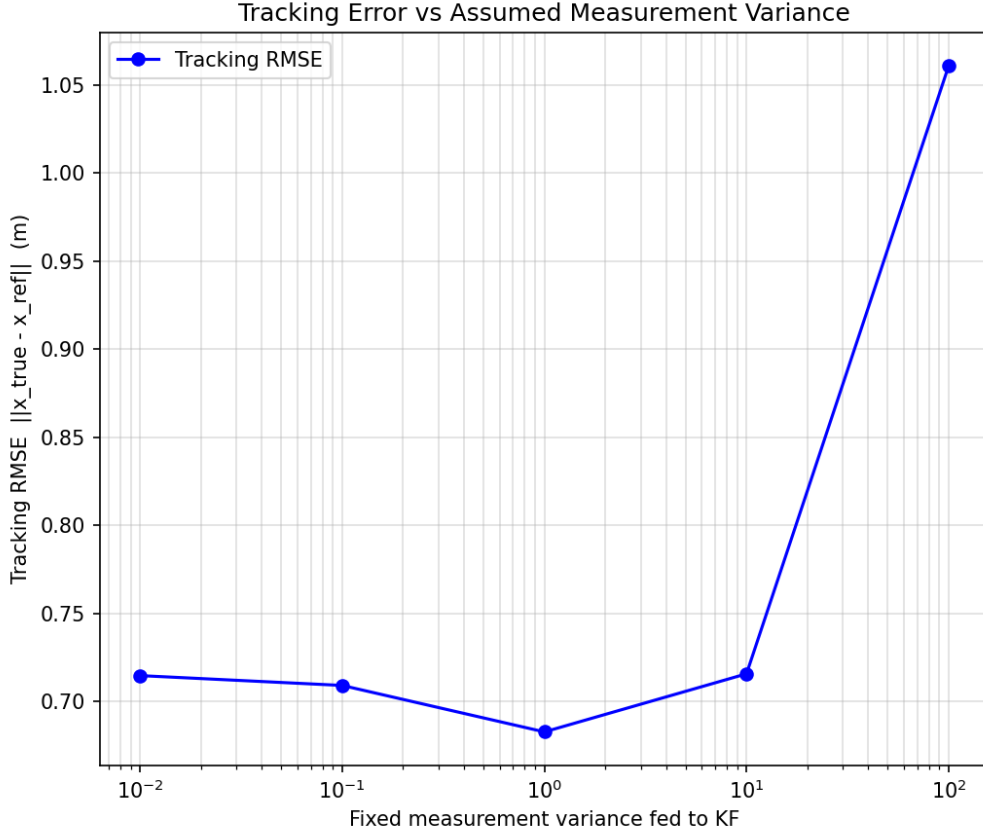


Figure 4.10: True-position tracking RMSE across the five fixed measurement variances. Each point is one seed-0 rollout of the same reference geometry; no repeated-run confidence interval is estimated. The minimum is limited to the tested values.

unreliable. Developers and operators should document responsibility for monitoring and responding to failures, consistent with the framework’s emphasis on safety evaluation and accountability [40]. These safeguards have not been validated here, and the fixed-variance results do not constitute a safety guarantee.

Ecological impact and sustainability. Ray tracing, neural-network training, and repeated inference incur computational costs. Energy and carbon accounting requires information about runtime, hardware, power consumption, and the electricity supply [41]. This thesis does not report measured energy consumption or a lifecycle assessment, including hardware production and disposal. The 24×300 configuration retains 56.25% of the CSI coefficients in the full 32×400 input (Table 4.6); energy savings remain unmeasured. Masked inputs retain the full tensor shape, so fewer nonzero coefficients do not by themselves reduce the network workload. A sustainability comparison would need to account for sensing and radio hardware, computation, and any actual transmission costs at comparable task performance. Reusing generated data and trained models may avoid redundant computation, but no net environmental benefit is established by the present experiments.

Deployment boundaries. The single scene, random within-scene split, and limited closed-loop conditions leave uncertainty about performance under environmen-

tal change and real hardware impairments. Before deployment, independent measurements and broader trajectory tests would be needed, together with privacy and safety assessments and energy accounting. The supported outcome is a simulation case study of CSI-based sensing and control; deployment readiness, privacy protection, and sustainability require further evidence.

5

Conclusion and Future Work

5.1 Conclusion

This thesis examined CSI-based position estimation and its use in simulated feedback control. The results show that extracting useful position information from CSI and assessing its reliability are distinct challenges. The following answers connect the four research questions to the experimental findings and their evidence boundaries.

RQ1: accurate on average, uneven in space. CSI supports sub-metre average localization in the evaluated scene, but the error distribution and spatial map show that reliability varies across positions. A small proportion of large errors remains important for feedback use, because average accuracy does not describe the measurements encountered along a particular trajectory. The ResNet’s lower aggregate error than the FCNN supports its use in this evaluation (Table 4.4). Unequal training budgets and single runs, however, prevent attributing that difference solely to architectural design.

RQ2: additional CSI has diminishing returns. Localization improves substantially as the smallest joint mask grows, then changes little across the larger configurations (Table 4.6). This suggests that many additional coefficients provide limited benefit to the trained estimator in this scene. Resource selection should therefore be judged by the estimation performance it preserves. The observed plateau does not establish an optimal allocation: antenna count, frequency span, and retained energy vary together, and the small differences among larger masks have not been tested across repeated runs. The experiment also provides no measurement of communication or energy savings.

RQ3: localization accuracy does not ensure trustworthy confidence. The learned covariance fails the consistency check and poorly reflects measurement quality: its output is weakly related to spatial error and changes little when resource restrictions substantially degrade localization. Moreover, conservative uncertainty in aggregate conceals overconfidence under the smallest mask (Tables 4.5 and 4.6). The main weakness is therefore both the scale of confidence and its responsiveness to measurement quality. A global rescaling may correct average confidence, but cannot by itself establish useful uncertainty across operating conditions. Covariance needs validation at the resource-configuration level before it can serve as a reliable filtering input; a resulting tracking benefit has not yet been demonstrated.

RQ4: measurement confidence changes realized tracking. The fixed-covariance sweep favors an intermediate confidence setting, consistent with balancing neural measurements against motion-model prediction. More revealingly,

estimated-state cost can fall while true tracking becomes worse (Table 4.7). A controller can therefore appear successful according to its internal state while the simulated system departs from the reference. True-state cost, shown in Fig. 4.9, is needed to assess that outcome. The archived sweep links this finding to the offline checkpoint and an explicit controller configuration (Appendix A). It establishes sensitivity for one seed and one reference geometry. Missing controller controls prevent attributing gains to CSI updates or filtering, and robustness across noise realizations and trajectories remains untested.

Together, these findings show why localization accuracy, confidence quality, and realized control performance require separate validation. The evidence is limited to one simulated scene, a random within-scene split, limited training runs, and one reference trajectory. Ideal interfaces connect the modules, so the study supports conclusions about CSI-based sensing and estimation in simulation; it does not establish semantic-message transmission, communication efficiency, or deployment robustness. The contribution is an implemented sensing-and-control pipeline that identifies confidence quality as an unresolved requirement for connecting learned localization to reliable feedback use.

5.2 Limitations

These technical limitations also constrain claims about safe deployment and environmental benefit; Section 4.6 discusses the associated privacy risks, safety responsibilities, and unmeasured energy and lifecycle costs.

The main limitation of the current work is that the L-2 estimator does not yet provide a well-trained and well-calibrated variance head. The predicted variance is only weakly correlated with the empirical error field, and its absolute scale is not reliable enough for direct closed-loop use. The ANEES diagnostics show conservative covariance in aggregate, while the smallest resource mask produces overconfidence. The reported closed-loop experiments therefore use manually selected fixed measurement variances and do not evaluate the learned variance output in control. This limits the strength of the conclusion that can be drawn about uncertainty-aware CSI-based state estimation. The thesis demonstrates that measurement uncertainty is important, but it does not yet demonstrate a fully successful learned uncertainty interface between the CSI estimator and the Kalman filter. The two-stage training procedure in Section 3.3.4 is a possible contributor, but no ablation establishes the cause of the observed calibration failure. Freezing backbone and position-head parameters during the second stage limits which parameters the variance loss can adapt; it does not alone guarantee unchanged predictions because training-mode buffers can still change. If the frozen features do not contain enough information to explain heteroscedastic localization errors, the variance output can only compensate by adjusting its final layers. In addition, minimizing negative log-likelihood does not by itself guarantee calibration under distribution shift, finite data, or model mismatch. As a result, the present variance head is an initial diagnostic component rather than a finished probabilistic estimator.

The second limitation is that L-2 is used as a functional motivation for state estimation. The neural model is trained to predict position and variance from CSI, while

the Kalman filter and controller are applied afterwards. The learning objective is not optimized end-to-end through LQR tracking cost, and it does not include the cost or impairments of transmitting a representation. This modular structure supports analysis of estimation and covariance sensitivity, but it does not establish a control-optimal L-3 representation or a jointly optimized semantic transceiver.

The third limitation concerns the absent communication transport model. Antenna-subcarrier masking selects CSI inputs, and the neural estimator extracts position information from those inputs. Neither operation demonstrates compression of a transmitted message. The implementation has no learned encoder-decoder pair operating across a modeled channel, and no explicit source or channel coding, quantization, packet loss, transport latency, or bitrate constraint. The physical channel model supplies sensing features, while downstream modules exchange data through ideal interfaces. Thus, the resource study does not quantify communication overhead or energy savings.

The fourth limitation is the narrowness of the simulated setting, in three related respects. The channel data come from a controlled ray-tracing environment with fixed base-station geometry, fixed carrier and bandwidth settings, a fixed receiver height and a single street-canyon layout, so real impairments such as hardware imperfections, synchronization and calibration errors, moving scatterers and imperfect scene knowledge are absent. The dynamics are equally simplified: the controller assumes a constant-velocity model with acceleration inputs, while the channel generation models neither Doppler nor the temporal correlation induced by motion, so the temporal structure exploited by the Kalman filter comes from the control model rather than from a physically time-correlated channel. Finally, although the scene contains two base stations, the pipeline uses the CSI of a single serving station, leaving cooperative sensing and multi-view fusion, which could reduce ambiguity in non-line-of-sight regions, outside the present scope. Together these choices make the results evidence of feasibility in the simulated scenario rather than a prediction of real-world performance.

Finally, the evaluation uses a limited set of baselines and operating conditions. The FCNN comparison supplies an offline localization reference, but its training budget differs from that of the ResNet, and each model is represented by a single training run. Classical geometry-based positioning, nearest-neighbor fingerprinting, a training-set mean predictor, and CSI reconstruction baselines are not evaluated. The closed-loop study compares fixed covariance choices within one pipeline; it has no oracle state-feedback, unfiltered-measurement, or model-only control case. Consequently, it does not attribute tracking gains to CSI updates or filtering, establish proximity to ideal sensing, or provide a comprehensive ranking against other localization or control methods. These missing comparisons are separate from the observed covariance-calibration failure.

5.3 Future Work

Four directions follow from the results of this thesis. The first two answer the questions that the experiments left open: how a localization network can be made to estimate its own uncertainty properly, and how the modular pipeline studied here

could be pushed toward a genuinely control-oriented, L-3 formulation.

The most immediate direction is to make the variance head produce a covariance that a filter can use without manual tuning. The experiments identify two distinct failures, and they call for different remedies. The first is one of scale: the predicted total variance exceeds mean squared localization error by roughly a factor of two and a half in aggregate, and a post-hoc calibration step fitted on the validation split could absorb much of this, for instance by learning a single scalar or a per-coordinate factor that minimizes the negative log-likelihood, or by temperature-style scaling of the predicted log-variance [36]. The second concern is resource responsiveness: the predicted variance barely responds to the sensing-resource configuration, even though the true error varies by a factor of three across the eight configurations of Section 4.5.4. This points to conditioning the variance head explicitly on the resource action, either by appending the action index to the shared feature vector or by fitting a separate covariance scale per configuration, as an additional way to represent the difference between resource configurations; its benefit would need to be tested.

A third possibility is that the frozen backbone simply does not carry the information needed to explain the residual error. In that case the two-stage procedure of Section 3.3.4 would have to be relaxed, for example by fine-tuning the last residual blocks jointly with the variance head under a heteroscedastic likelihood [34], or replaced altogether by an ensemble or Monte Carlo dropout estimate of the predictive variance [33, 35]. Whichever route is taken, the evaluation protocol should change with it: calibration should be reported per resource configuration and not only in aggregate, because the single ANEES value of this thesis hides a swing from strongly over-confident at the smallest budget to conservative at the largest, and metrics such as NIS and reliability diagrams should be used alongside ANEES so that the filter is checked in operation and not only offline.

A second direction is to move from modular L-2 estimation toward joint L-2/L-3 optimization, that is, to optimize the learned representation against the control objective rather than against localization error. Making this concrete requires a differentiable downstream chain. The Kalman filter update is already a sequence of differentiable linear-algebra operations, and the LQR gain is a fixed function of the system matrices, so the accumulated tracking cost of a simulated trajectory can in principle be back-propagated into the estimator. Training the network to minimize that cost instead of position error would let it decide for itself which channel features matter, and would test a hypothesis that the modular results here can only hint at: that a representation with slightly worse standalone localization accuracy may still yield a lower closed-loop cost, if its errors are smoother in time or its covariance is better calibrated.

Two difficulties should be expected on that path. The first is credit assignment over long trajectories, since the control cost incurred at one step depends on estimates made many steps earlier; training on short rollouts before extending the horizon is the natural mitigation. The second is that the resource action is discrete. Choosing how many antennas and subcarriers to spend at each step is not a differentiable operation, so an L-3 formulation that also allocates resources would need a policy-learning or bandit treatment, with the accumulated LQR cost as the reward and the

resource fraction as the price paid for it. That formulation would turn the trade-off measured statically in Section 4.5.4 into a decision the system makes online, as a possible extension of the present input-selection study. An explicit transport model would additionally be required to interpret that allocation in terms of communication cost.

A third direction is to introduce an explicit communication transport model. Future experiments can include explicit bandwidth, quantization, latency, packet loss, and channel-noise constraints so that semantic performance can be measured against communication-resource consumption. This would turn the current resource-selection study into a fuller semantic communication problem, where the system must decide what information to transmit, at what resolution, and with what reliability in order to support closed-loop control.

A fourth direction is to broaden the propagation and mobility scenarios. The pipeline can be tested with more base-station layouts, different carrier frequencies and bandwidths, dynamic scatterers, Doppler effects, and real or measurement-calibrated channel data. Multi-base-station sensing and fusion can also be incorporated, since closed-loop systems are likely to benefit from redundant spatial observations when the target moves through non-line-of-sight or weak-coverage regions. This thesis provides a simulation study of CSI-based position estimation and covariance-sensitive feedback tracking, motivated by L-2 semantic communication. Within the evaluated scene, the estimator achieves the reported localization accuracy, the learned covariance remains poorly calibrated, and fixed measurement covariance affects realized tracking and true control cost. These findings concern the implemented sensing and estimation pipeline. Benefits from semantic message transmission, robustness across environments, and gains over unreported control alternatives remain outside the demonstrated evidence.

Bibliography

- [1] G. Pan, Z. Tian, M. Bennis, and H. Wymeersch, “Active Inference-Enabled Agentic Closed-Loop ISAC with Long-Horizon Planning,” arXiv:2604.19599, 2026.
- [2] E. Calvanese Strinati and S. Barbarossa, “6G networks: Beyond Shannon towards semantic and goal-oriented communications,” *Computer Networks*, vol. 190, Art. no. 107930, 2021.
- [3] X. Luo, H.-H. Chen, and Q. Guo, “Semantic communications: Overview, open issues, and future research directions,” *IEEE Wireless Communications*, vol. 29, no. 1, pp. 210–219, 2022.
- [4] W. Yang, H. Du, Z. Q. Liew, W. Y. B. Lim, Z. Xiong, D. Niyato, X. Chi, X. Shen, and C. Miao, “Semantic communications for future Internet: Fundamentals, applications, and challenges,” *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 213–250, 2023.
- [5] D. Gündüz, Z. Qin, I. E. Aguerri, H. S. Dhillon, Z. Yang, A. Yener, K. K. Wong, and C.-B. Chae, “Beyond transmitting bits: Context, semantics, and task-oriented communications,” *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 1, pp. 5–41, 2023.
- [6] J. Shao, Y. Mao, and J. Zhang, “Task-oriented communication for multidevice cooperative edge inference,” *IEEE Transactions on Wireless Communications*, vol. 22, no. 1, pp. 73–87, 2023.
- [7] E. Bourtsoulatze, D. Burth Kurka, and D. Gündüz, “Deep joint source-channel coding for wireless image transmission,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 3, pp. 567–579, 2019.
- [8] C.-K. Wen, W.-T. Shih, and S. Jin, “Deep learning for massive MIMO CSI feedback,” *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 748–751, 2018.
- [9] F. Liu, Y. Cui, C. Masouros, J. Xu, T. X. Han, Y. C. Eldar, and S. Buzzi, “Integrated sensing and communications: Toward dual-functional wireless networks for 6G and beyond,” *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 6, pp. 1728–1767, 2022.
- [10] J. A. Zhang, F. Liu, C. Masouros, R. W. Heath Jr., Z. Feng, L. Zheng, and A. Petropulu, “An overview of signal processing techniques for joint communication and radar sensing,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 6, pp. 1295–1315, 2021.
- [11] C.-H. Hsieh, J.-Y. Chen, and B.-H. Nien, “Deep learning-based indoor localization using received signal strength and channel state information,” *IEEE Access*, vol. 7, pp. 33256–33267, 2019.

- [12] A. Alkhateeb, “DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications,” in *Proc. Information Theory and Applications Workshop (ITA)*, San Diego, CA, USA, Feb. 2019, pp. 1–8.
- [13] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, “Wireless network design for control systems: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 978–1013, 2018.
- [14] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, “Deep learning enabled semantic communication systems,” *IEEE Transactions on Signal Processing*, vol. 69, pp. 2663–2675, 2021.
- [15] Y. S. Cho, J. Kim, W. Y. Yang, and C. G. Kang, *MIMO-OFDM Wireless Communications with MATLAB*. Singapore: John Wiley & Sons (Asia), 2010.
- [16] T. L. Marzetta, “Noncooperative cellular wireless with unlimited numbers of base station antennas,” *IEEE Transactions on Wireless Communications*, vol. 9, no. 11, pp. 3590–3600, 2010.
- [17] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, “Massive MIMO for next generation wireless systems,” *IEEE Communications Magazine*, vol. 52, no. 2, pp. 186–195, 2014.
- [18] 3GPP, “Study on channel model for frequencies from 0.5 to 100 GHz,” 3rd Generation Partnership Project (3GPP), Technical Report TR 38.901, version 19.1.0, Oct. 2025.
- [19] J. Hoydis, F. Aït Aoudia, S. Cammerer, M. Nimier-David, N. Binder, G. Marcus, and A. Keller, “Sionna RT: Differentiable ray tracing for radio propagation modeling,” in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 317–321.
- [20] F. Aït Aoudia, J. Hoydis, M. Nimier-David, B. Nicolet, S. Cammerer, and A. Keller, “Sionna RT: Technical Report,” arXiv:2504.21719, 2025.
- [21] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero III, R. L. Moses, and N. S. Correal, “Locating the nodes: Cooperative localization in wireless sensor networks,” *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54–69, 2005.
- [22] H. Wymeersch, G. Seco-Granados, G. Destino, D. Dardari, and F. Tufvesson, “5G mmWave positioning for vehicular networks,” *IEEE Wireless Communications*, vol. 24, no. 6, pp. 80–86, 2017.
- [23] J. Xiao, K. Wu, Y. Yi, and L. M. Ni, “FIFS: Fine-grained indoor fingerprinting system,” in *Proc. 21st International Conference on Computer Communications and Networks (ICCCN)*, Munich, Germany, 2012, pp. 1–7.
- [24] X. Wang, L. Gao, S. Mao, and S. Pandey, “DeepFi: Deep learning for indoor fingerprinting using channel state information,” in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, USA, 2015, pp. 1666–1671.
- [25] X. Wang, L. Gao, S. Mao, and S. Pandey, “CSI-based fingerprinting for indoor localization: A deep learning approach,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, 2017.
- [26] X. Wang, L. Gao, and S. Mao, “CSI phase fingerprinting for indoor localization with a deep learning approach,” *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1113–1123, 2016.

- [27] T. Li, H. Wang, Y. Shao, and Q. Niu, “Channel state information-based multi-level fingerprinting for indoor localization with deep learning,” *International Journal of Distributed Sensor Networks*, vol. 14, no. 10, Art. no. 1550147718806719, 2018.
- [28] C. Studer, S. Medjkouh, E. Gonultas, T. Goldstein, and O. Tirkkonen, “Channel charting: Locating users within the radio environment using channel state information,” *IEEE Access*, vol. 6, pp. 47682–47698, 2018.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.
- [30] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.
- [32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *Proc. International Conference on Learning Representations (ICLR)*, 2021.
- [33] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” in *Proc. 33rd International Conference on Machine Learning (ICML)*, New York, NY, USA, 2016, pp. 1050–1059.
- [34] A. Kendall and Y. Gal, “What uncertainties do we need in Bayesian deep learning for computer vision?” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5574–5584.
- [35] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 6402–6413.
- [36] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in *Proc. 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, 2017, pp. 1321–1330.
- [37] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
- [38] S. Tatikonda and S. K. Mitter, “Control under communication constraints,” *IEEE Transactions on Automatic Control*, vol. 49, no. 7, pp. 1056–1068, 2004.
- [39] G. N. Nair, F. Fagnani, S. Zampieri, and R. J. Evans, “Feedback control under data rate constraints: An overview,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 108–137, 2007.
- [40] E. Tabassi, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, National Institute of Standards and Technology, NIST AI 100-1, Jan. 2023. doi: 10.6028/NIST.AI.100-1.
- [41] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, “Towards the systematic reporting of the energy and carbon footprints of machine learning,” *Journal of Machine Learning Research*, vol. 21, no. 248, pp. 1–43, 2020. [Online]. Available: <https://jmlr.org/papers/v21/20-312.html>.

A

Detailed Experimental Configuration

This appendix records the configuration needed to interpret and repeat the experiments in Chapter 4. Saved logs support the offline training and checkpoint identities. Retained generator revisions and a direct check of the saved position array recover part of the dataset provenance, while the original CSI-generation environment remains unverified. The closed-loop sweep has an explicit source snapshot, checkpoint hash, command, software inventory, and saved trajectories.

A.1 Channel Data and Preprocessing

The offline dataset is `datasets/channel_dataset100M400SC32A_canyon_random20000.npz`. It contains 20000 UE positions and complex CSI arrays for two base stations, stored under `ue_position`, `channel_bs1`, and `channel_bs2`. Each BS channel has shape $20000 \times 32 \times 400$ and complex64 storage. Only BS1 is used by the reported estimators. UE positions are sampled uniformly over $x \in [-80, 80]$ m and $y \in [-8, 8]$ m, at height 1.5 m. All 20000 saved positions are reproduced exactly by alternating x and y draws from NumPy `RandomState(42)` with these bounds and the fixed height. This recovers a deterministic reproduction sequence for positions, rather than proving the original invocation. Exact CSI reproduction still requires the saved channel arrays because the original software environment is not fully recorded.

The retained generator revisions `621c2f8` and `575fbd9` both compute the numerical value 0.05 and pass it directly to `PlanarArray`. In the inspected Sionna RT 1.2.1 implementation, spacing is expressed in wavelength units, so this argument corresponds to 0.05λ (approximately 5 mm at 3 GHz). The saved CSI arrays contain no generation manifest linking them to a particular source revision and software environment; historical physical half-wavelength spacing is therefore not established. The new closed-loop run explicitly records the 0.05λ configuration. No pilot-based CSI estimation error, Doppler, or communication transport impairment is added in the inspected pipeline.

For each sample, the real and imaginary components of BS1 CSI are stacked into a float32 tensor $H_{\text{RI}} \in \mathbb{R}^{2 \times 32 \times 400}$. The implemented normalization is

$$\widetilde{H}_{\text{RI}} = \sqrt{\frac{64}{\sum_{c,m,n} |H_{\text{RI},c,m,n}|^2}} H_{\text{RI}}. \quad (\text{A.1})$$

Table A.1: Settings in the inspected channel generator. These values document the code configuration; a historical generator-version manifest is unavailable.

Setting	Value
Scene	Sionna RT <code>simple_street_canyon</code> ; scene-provided materials
BS1 / BS2 position	$(-32, 12, 35) / (33, -12, 35)$ m
BS1 / BS2 orientation	$(-\pi/2, 0, 0) / (\pi/2, 0, 0)$ radians
Carrier / nominal bandwidth	3 GHz / 100 MHz
Subcarriers	400 centered frequencies; spacing 250 kHz
BS / UE array	1×32 , <code>tr38901</code> , V polarization / 1×1 , isotropic, V polarization
Spacing argument	0.05 passed to <code>PlanarArray</code>
Transmit power parameter	30 dBm per BS
Path-solver limits	Depth 5; 100000 samples and maximum paths per source; seed 43
Enabled path mechanisms	LOS, specular and diffuse reflection, refraction, diffraction; lit-region diffraction enabled; edge diffraction disabled
Array / CFR options	Solver <code>synthetic_array=False</code> ; CFR delay normalization and amplitude normalization disabled; reverse direction enabled

The numerator follows the first two tensor dimensions in the code, not the total number of coefficients. There is no explicit zero-energy guard. Normalization precedes masking, and the retained coefficients are not renormalized afterwards. Position targets are the absolute UE coordinates minus the BS1 horizontal coordinates $(-32, 12)$ m.

The eight paired mask configurations are $(k_{\text{at}}, k_{\text{sc}})$ pairs $(4, 50)$, $(8, 100)$, $(12, 150)$, $(16, 200)$, $(20, 250)$, $(24, 300)$, $(28, 350)$, and $(32, 400)$. Each mask retains centered contiguous antenna and subcarrier indices and zeros all other entries; the tensor remains $2 \times 32 \times 400$. Training and validation draw a uniform action at each sample access. The test evaluator uses a sequential NumPy `default_rng(42)` action sequence, and the FCNN evaluator uses the same order and sequence. The resource sweep instead evaluates every action on all 2000 test positions. Antenna count, frequency span, and retained energy therefore change together; the experiment does not isolate these effects or measure energy consumption.

A.2 Split, Training, and Checkpoint Selection

The split file `datasets/split_801010_seed42.npz` stores ordered `train`, `val`, and `test` index arrays. It is formed by a NumPy `default_rng` generator with seed 42 and a permutation of 20000 indices, followed by 16000, 2000, and 2000 indices respectively. This is a random within-scene split, not a spatially blocked split. The test evaluator checks that its indices are disjoint from training and validation.

The reported ResNet run uses `train_split_two_stage.py`, with seed 42 for

Python, NumPy, and PyTorch, batch size 32, Adam learning rate 10^{-4} , and weight decay 10^{-4} . Stage 1 trains the backbone and position head for 500 epochs using coordinate-averaged squared error; the variance-head parameters are frozen. The checkpoint minimizing the logged validation localization error, `val/ave_loc_err`, is retained. The saved run records its best stage-1 validation error at epoch 478 (zero-based). Stage 2 starts from that checkpoint, freezes backbone and position-head parameters, and trains the variance head for 100 epochs using diagonal Gaussian negative log-likelihood. It selects the minimum logged validation NLL, `val/ave_loss`. Both monitored metrics average batch means without weighting the final smaller batch by its sample count; the FCNN uses the same aggregation for validation localization error. With 2000 validation samples and batch size 32, the final batch contains 16 samples. Parameter-freezing checks do not establish invariant training-mode buffers: the saved stage-1 and stage-2 checkpoints differ in their batch-normalization running means and variances. The reported offline results therefore refer to the selected stage-2 model, without assuming its position predictions equal those of stage 1.

The network uses an unpretrained torchvision ResNet-34, a two-channel initial convolution, a 512-dimensional pooled feature vector, and two dropout-linear heads (dropout 0.2), each with two outputs. The position-scale buffer is (1,1) in both the inspected implementation and the reported stage-2 checkpoint; physical outputs must use the buffers stored in the selected checkpoint. Predicted log variance is lower-clamped at -2 and exponentiated. The active likelihood is $\frac{1}{2} \sum_{j=1}^2 (\log v_j + e_j^2/v_j)$, averaged over samples. Stored minimum/maximum variance buffers are not an additional active upper clamp in this forward path.

The reported offline checkpoint is

`model/MaMIMO_ResNet32_two_stage_canyon_v2/stage2_best.ckpt.`

The saved stage-2 checkpoint records selection at epoch 53 (zero-based). Its evaluation provenance is recorded in `results/two_stage_canyon_v2_test/diagnostic_summary.txt` and the accompanying training log. The FCNN checkpoint is `model/FCNN_baseline_canyon/fcnn_best.ckpt`. Its flattened 25600 inputs feed ReLU hidden layers of widths 1024, 1024, and 256, followed by two position outputs. It has 27527938 parameters and no variance head. Its configured training budget is 150 epochs, with validation-error early stopping (patience 25); the reported run stopped after 147 epochs and its saved best checkpoint records epoch 121 (zero-based). Adam, learning rate, weight decay, batch size, split, and seed match the settings above. Optimization budgets and architectures nevertheless differ.

Data loaders use zero workers, training shuffle enabled, validation shuffle disabled, and no dropped final batch. The training script selects one GPU and 16-bit mixed precision when CUDA is available, otherwise CPU and 32-bit precision. The saved ResNet log identifies an NVIDIA GeForce RTX 4060 Laptop GPU. Seed setting alone does not guarantee bitwise repeatability, and deterministic execution was not established for the historical training runs.

A.3 Evaluation and Plotting

Offline evaluation uses batch size 256, inference mode, and the ordered test indices. Mean Euclidean error is $N^{-1} \sum_i \|e_i\|_2$; localization RMSE is $\sqrt{N^{-1} \sum_i \|e_i\|_2^2}$. These are distinct from coordinate-averaged training MSE. Predicted total variance is $v_x + v_y$. Offline NEES is $e_x^2/v_x + e_y^2/v_y$, with numerical clipping of denominators at 10^{-12} in the resource evaluator. The two-dimensional target ANEES is 2. The reported 95% interval [1.913, 2.089] is a chi-square consistency interval for $N = 2000$, under the Gaussian and independence assumptions, rather than a multi-seed confidence interval for model accuracy.

Spatial correlations are computed from raw 2 m cells containing at least three test samples. Interpolation, smoothing, and color limits affect only the visualization, not the sample-level metrics or raw-cell correlations. The current evaluator defaults to a 0.5 m display grid, Gaussian smoothing with standard deviation one grid cell, a linear color mapping, a 2 m error color limit, and a variance color limit at the 90th percentile of finite display-grid values. These are current reproduction settings; the complete plotting arguments for the original figure files were not retained. The error map in Fig. 4.3 corresponds to `results/heatmap_template_p90/resnet_canyon/error_heatmap.png`, and the variance map in Fig. 4.4 to `results/two_stage_canyon_v2_test/variance_heatmap.png`. Both maps visualize saved test predictions, without additional ray-traced grid samples.

A.4 Closed-loop Configuration and Provenance

The state is (x, y, v_x, v_y) , with acceleration input (u_x, u_y) and sampling interval $\Delta t = 0.1$ s. The dynamics use the constant-velocity transition matrix and input matrix with position entries $\Delta t^2/2$ and velocity entries Δt . Measurements observe position only. The reference is $(-80 + t, 0, 1, 0)$ for 160 s (1600 samples), and both the simulated true state and initial estimate start at $(-80, 0, 1, 0)$. Initial filter covariance is $0.1I_4$. Independent Gaussian process noise has standard deviation 0.01 per state component per step; the filter uses $Q = 10^{-4}I_4$.

LQR weights are $Q_{\text{LQR}} = \text{diag}(1, 1, 0, 0)$ and $R_{\text{LQR}} = 0.1I_2$. The sweep uses the stationary infinite-horizon gain (`horizon=None`), computed by the implemented Riccati iteration with a maximum of 300 iterations and a stopping tolerance of 10^{-9} . The applied acceleration is clipped to $[-5, 5]$ per axis. CSI is regenerated at the simulated true position and supplied with all 32×400 coefficients after normalization. BS1-relative predictions are translated back to absolute coordinates. The learned variance is ignored; measurement covariance is $R = rI_2$ with $r \in \{0.01, 0.1, 1, 10, 100\}$ m². State and costs are recorded before the next dynamics update. True cost sums $(x_k - x_k^{\text{ref}})^T Q_{\text{LQR}} (x_k - x_k^{\text{ref}}) + u_k^T R_{\text{LQR}} u_k$, using the applied control; tracking RMSE uses true horizontal position error.

The reported sweep is generated by `rerun_closedloop_provenance.py` with the same canyon stage-2 checkpoint used for the offline evaluation. It uses one trial per variance and seed 0, resetting NumPy and PyTorch seeds before each rollout. For the 1599 stored transitions, reconstructed process-noise residuals agree across

variance values within the float32 reconstruction tolerance (5×10^{-5}). The reference geometry remains one straight trajectory; this is not a multiple-seed or multiple-trajectory robustness study.

The run archive is `results/closed_loop_canyon_v2_manifest_20261009/`. Its `manifest.json` records the full command, source and checkpoint SHA-256 hashes, package versions, scene and controller settings. `metrics.csv` contains true and estimated cost, tracking and estimation RMSE, and trial identities. Per-variance NPZ files retain true and estimated states, neural measurements, reference, applied controls, measurement variance, and both instantaneous costs. Figures 4.8–4.10 use these same saved arrays. The cost figure shows true-state cost only. Historical results without complete run provenance are superseded by this explicitly configured run.

B

Code and Data Reproduction Guide

This appendix explains how to repeat the offline evaluations and run the current simulation. Re-evaluating saved checkpoints, retraining a model, and regenerating a dataset are different levels of reproduction. The first preserves the evaluated model and test set; the latter two can produce different results. Appendix A identifies the configuration and remaining provenance limitations.

B.1 Repository, Environment, and Required Artifacts

The configured Git remote is <https://github.com/MlrRoranGe/Semantic-Communication-Encoder>. The inspected working tree contains local modifications and untracked experiment scripts; the base commit `575fbd9` alone does not represent the complete thesis experiments. A clean clone of that commit is consequently not a verified reproduction package. Dataset, checkpoint, and result directories are excluded from normal Git tracking. A complete release must include the experiment scripts listed below, a fixed source revision, an environment export, and a separate archive of the required data and checkpoints with checksums. No public archive identifier or verified download link for those artifacts is available in the inspected records.

The working environment is named `sionna_env`. An inventory taken on 8 October 2026 gives Python 3.12.12, NumPy 2.4.3, PyTorch 2.11.0+cu128, torchvision 0.26.0+cu128, PyTorch Lightning 2.6.1, Sionna RT 1.2.1, Mitsuba 3.7.1, Dr.Jit 1.2.0, SciPy 1.17.0, and Matplotlib 3.10.8. Both inspected best checkpoints record PyTorch Lightning 2.6.1. The remaining inventory describes the current environment, not a verified historical version list for every reported run. The training log supplies the GPU model listed in Appendix A; historical driver and CUDA runtime details are not fully archived.

Before distributing a reproduction snapshot, export the environment and source status from the repository root:

```
conda env export -n sionna_env > environment.yml
conda run -n sionna_env python -m pip freeze > requirements.txt
git rev-parse HEAD
git status --short
```

```
git diff > working-tree.patch
```

The patch does not include untracked files, which must be added explicitly to the release. Install from the exported environment, with a platform-compatible CUDA/PyTorch build if necessary, rather than treating the inventory above as proof that all platforms reproduce the historical results. Retain the export and run logs with each output directory.

Required offline artifacts are the canyon dataset, the saved seed-42 split, the canyon stage-2 ResNet checkpoint, and the FCNN checkpoint named in Appendix A. Retain the stage-1 checkpoint and training logs for training provenance. The files `diagnostic_arrays.npz` and `resource_sweep.csv` support inspection of predictions and metrics without running ray tracing. In PowerShell, `Get-FileHash <path> -Algorithm SHA256` can record artifact identities; hashes must be computed from the distributed files rather than supplied as guessed identifiers.

B.2 Dataset and Split Preparation

Place the saved dataset and split in `datasets/`, preserving their filenames. Check the three dataset keys and channel dimensions specified in Appendix A. Use the saved split for reproducing the reported evaluation. If it is unavailable, the following Python snippet regenerates the index rule for the same ordered 20000-sample dataset; it cannot reconstruct missing CSI or the original position samples:

```
import numpy as np
from pathlib import Path
p = Path("datasets/split_801010_seed42.npz")
assert not p.exists(), "Preserve the existing split"
i = np.random.default_rng(42).permutation(20000)
np.savez(p, train=i[:16000], val=i[16000:18000],
         test=i[18000:])
```

For a new dataset, `gen_data.py` calls the channel generator and saves the three arrays. Its output path is currently an absolute path for an older local checkout and must be changed before running it. To reproduce the saved position sequence, initialize NumPy's legacy generator with seed 42 and draw x and y alternately for each sample. A different ordering or `default_rng` generator does not reproduce this sequence. Record the full generation command and environment before producing new CSI. Confirm the scene, spacing argument, and solver options before generation. A newly generated dataset must receive a new filename and provenance record; it cannot be assumed to reproduce the historical dataset bit for bit. Generation is unnecessary when repeating offline evaluation with the saved arrays.

B.3 Offline Evaluation and Figure Generation

The following PowerShell commands run from the repository root in an activated `sionna_env`. Explicit paths avoid obsolete script defaults. New output directories preserve historical results:

Table B.1: Mapping between thesis evidence and reproduction entry points.

Evidence	Script and saved reference
Localization, spatial error, uncertainty	<code>eval_test_split.py</code> ; <code>results/two_stage_canyon_v2_test/</code>
FCNN comparison and paired CDF	<code>eval_fcnn_baseline.py</code> ; <code>results/fcnn_baseline_test/</code>
Eight resource configurations	<code>eval_resource_sweep.py</code> ; <code>results/resource_sweep_two_stage_canyon_v2/</code>
Closed-loop trajectory illustration	<code>rerun_closedloop_provenance.py</code> ; saved sweep arrays at the lowest observed tracking RMSE
Fixed-covariance sweep and true-cost plot	<code>rerun_closedloop_provenance.py</code> ; <code>results/closed_loop_canyon_v2_manifest_20261009/</code>

```

conda activate sionna_env
$data = "datasets/channel_dataset100M400SC32A_" + '
        "canyon_random20000.npz"
$split = "datasets/split_801010_seed42.npz"
$model = "model/MaMIMO_ResNet32_two_stage_canyon_v2"
$ckpt = "$model/stage2_best.ckpt"
$fcnn = "model/FCNN_baseline_canyon/fcnn_best.ckpt"

python eval_test_split.py --dataset $data --split_file $split '
    --ckpt $ckpt --seed 42 --batch_size 256 '
    --out_dir results/reproduce_resnet

python eval_fcnn_baseline.py --dataset $data '
    --split_file $split --ckpt $fcnn --seed 42 '
    --resnet_arrays results/reproduce_resnet/diagnostic_arrays.npz '
    --out_dir results/reproduce_fcnn

python eval_resource_sweep.py --dataset $data '
    --split_file $split --ckpt $ckpt --batch_size 256 '
    --out_dir results/reproduce_resources

```

Do not use `-no_resource_mask` for the mixed-mask localization result. The first evaluator produces localization and uncertainty summaries, diagnostic arrays, a CDF, spatial maps, and NEES plots. The second produces the FCNN summary and paired CDF when the ResNet arrays are supplied. The resource evaluator produces per-action arrays, a CSV, a LaTeX table, and error/ANEES plots. Compare numerical summaries before copying figures to Overleaf. Agreement of scalar metrics does not guarantee identical rendering if plotting settings or software versions differ.

The reported reference summaries contain ResNet mean error 0.857413 m and RMSE 1.203577 m, FCNN mean error 2.033135 m and RMSE 2.731836 m, and resource-action (24, 300) mean error 0.626025 m. These values provide numerical checks

for saved-checkpoint evaluation, not guarantees for retraining. More recent output directories ending in `rerun` or containing `plot_refinement` must be identified separately before replacing thesis figures or tables.

B.4 Retraining and Control Simulations

Retraining is not required to evaluate the saved models. To train new checkpoints with the documented budgets, reuse the data and split variables above:

```
python train_split_two_stage.py --dataset $data '
--split_file $split --seed 42 --batch_size 32 '
--stage1_epochs 500 --stage2_epochs 100 '
--lr_stage1 1e-4 --lr_stage2 1e-4 '
--model_dir model/reproduce_resnet
```

```
python train_fcnn_baseline.py --dataset $data '
--split_file $split --seed 42 --batch_size 32 '
--epochs 150 --patience 25 --lr 1e-4 --weight_decay 1e-4 '
--model_dir model/reproduce_fcnn
```

Evaluate those new checkpoint paths explicitly and report them as new runs. Do not assume that seed 42 recreates the saved weights exactly. Archive standard output, selected-checkpoint identity, validation selection criteria, and full command arguments.

The reported fixed-covariance sweep is repeated with the following explicit command, using the data/model variables above and a new output directory:

```
python rerun_closedloop_provenance.py --model-path $ckpt '
--out-dir results/reproduce_control_manifest '
--variances 0.01 0.1 1 10 100 --trials 1 --seed 0 '
--total-time 160 --velocity-ref 1 --process-noise-std 0.01 '
--horizon none
```

The runner requires a new output directory and archives the source, checkpoint identity, software inventory, metrics, and per-step arrays. Preserve stdout as a run log. After completion, validate the archive and generate figures from its saved arrays:

```
$run = "results/reproduce_control_manifest"
$fig = "results/reproduce_control_figures"
python agent_run.py --run-dir $run --variance 1 --out_dir $fig
python eval_variance_sweep.py --run-dir $run '
--out-dir $fig --csv "$fig/eval_var_sweep.csv"
```

These plotting modes read saved arrays without further simulation. The $r = 1 \text{ m}^2$ trajectory is the reported sweep's best tested setting, not a separate experiment. Without `-run-dir`, the older entry points have different defaults: `agent_run.py` uses remaining-horizon control, and `eval_variance_sweep.py` uses an older checkpoint. Use the explicit configuration above. More trials vary process noise on the same reference geometry. Oracle, unfiltered-measurement, and model-only controls are not reported.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se



CHALMERS
UNIVERSITY OF TECHNOLOGY