Data-Driven State Estimation for Robotic Systems Using Koopman-Inspired Lifting

by

Zi Cong Guo

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Institute for Aerospace Studies
University of Toronto

Data-Driven State Estimation for Robotic Systems Using Koopman-Inspired Lifting

Zi Cong Guo
Doctor of Philosophy

Institute for Aerospace Studies
University of Toronto
2026

Abstract

State estimation methods rely heavily on process and measurement models, yet deriving such models analytically can be difficult in many robotics applications due to complex sensing modalities and nonlinear dynamics, often resulting in modeling errors. This thesis investigates Koopman-inspired lifting as a framework for incorporating learned models into probabilistic state estimation. By preserving useful inference structure, the resulting methods enable efficient system identification, inference, and uncertainty quantification. The proposed approach learns lifted representations in which nonlinear estimation problems admit approximately (bi)linear-Gaussian structure, enabling model learning through least-squares-like procedures while retaining many of the tools of classical estimation. We first develop the Koopman-Inspired Learned Observations Extended Kalman Filter (KILO-EKF), which incorporates learned lifted measurement models into classical filtering architectures. We then introduce the Koopman State Estimator (KoopSE), extending lifting to both process and measurement models for data-driven batch state estimation. Building on these ideas, we develop the Reduced Constrained Koopman Linearization (RCKL) framework for localization and simultaneous localization and mapping (SLAM), addressing the challenges that arise in large-scale lifted estimation. In studying uncertainty estimation for these constrained formulations, we uncover a broader connection to Gaussian inference on manifolds. This leads to a general framework for marginalizing and conditioning Gaussian distributions onto linearized manifolds and provides a principled probabilistic interpretation of uncertainty estimation in equality-constrained optimization. Collectively, these results demonstrate how learned lifted representations can be incorporated into increasingly complex estimation frameworks while preserving efficient inference and principled uncertainty quantification.

To my family and mentors, who inspired a lifelong curiosity.

Acknowledgements

I would like to express my deepest gratitude to Professors Timothy D. Barfoot and James R. Forbes for their mentorship throughout my graduate studies. I have been fortunate to learn from two advisors who share a genuine passion for research and a strong commitment to their students. Their high standards pushed me to become a better researcher, while their enthusiasm for robotics and scientific discovery made research an enjoyable pursuit. I am especially grateful for the thoughtful and detailed feedback they consistently provided, and for continually challenging me on the rigor and clarity of my work. I appreciate their willingness to pursue the deeper theoretical questions that emerged from practical robotics problems. Their curiosity and openness to exploring these directions shaped this thesis in ways I could not have anticipated at the outset.

I would like to thank Frederike Dümbgen and Sven Lilge for their support and collaboration throughout my graduate studies. I am grateful for the many discussions we shared and for the perspectives they brought to my research. I thank Connor Holmes, Vassili Korotkine, and many other colleagues and collaborators for their contributions to this work and for making graduate school an enjoyable experience. I am grateful for the friendships developed along the way.

I thank the University of Toronto Institute for Aerospace Studies (UTIAS) and the Robotics Institute for providing an inspiring research environment and opportunities to connect with a diverse community of students and researchers across disciplines. I feel fortunate to have spent my graduate studies in such a collaborative and intellectually stimulating community. I gratefully acknowledge the financial support provided by UTIAS and by the Natural Sciences and Engineering Research Council of Canada, which helped make this research possible.

Finally, I would like to thank my family for their unwavering support throughout my studies. I am especially grateful to my parents for fostering my curiosity and love of learning from an early age, and for the many opportunities they provided throughout my life. I would also like to thank Xuran Ma and Qiyang Li for their friendship and companionship throughout my graduate studies. Their presence and the many conversations we shared helped ensure that life remained about more than research alone.

Contents

List of Acronyms xii

Notation xiii

1 Introduction 1
- 1.1 Thesis Organization . . . 2

2 Related Work 4
- 2.1 Classical State Estimation . . . 4
  - 2.1.1 Filtering, Smoothing, and SLAM . . . 4
  - 2.1.2 Estimation on Manifolds . . . 5
- 2.2 Learning-Based State Estimation . . . 5
  - 2.2.1 General Learning-Based Modeling and Estimation . . . 5
  - 2.2.2 Koopman-Based Modeling, Control, and Estimation . . . 6

3 Background 8
- 3.1 State Estimation . . . 8
  - 3.1.1 Problem Formulation . . . 8
  - 3.1.2 Connection to Optimization . . . 9
- 3.2 Koopman Lifting for Dynamical Systems . . . 11
  - 3.2.1 Koopman Operator Theory . . . 11
  - 3.2.2 Learning Finite-Dimensional Koopman Models . . . 12

4 Koopman-Inspired Filtering 14
- 4.1 Summary of Contributions . . . 14
- 4.2 Motivation . . . 15
- 4.3 Related Work . . . 15
- 4.4 Preliminaries . . . 16
  - 4.4.1 Standard Discrete-Time EKF on $SE_2(3)$ . . . 16
- 4.5 The General KILO-EKF . . . 18
  - 4.5.1 Learning the Lifted Measurement Model . . . 19
  - 4.5.2 Inference . . . 20
- 4.6 KILO-EKF in $SE_2(3)$ . . . 20
  - 4.6.1 Lifting Functions . . . 20
  - 4.6.2 Measurement Jacobian . . . 22

4.7 Experiments . . . 22
4.7.1 Problem Setup . . . 22
4.7.2 EKF Baselines . . . 23
4.7.3 KILO-EKF Implementation . . . 24
4.7.4 Evaluation Method . . . 25
4.7.5 Results . . . 27
4.8 Conclusion . . . 29

5 Koopman-Inspired Smoothing **30**
5.1 Summary of Contributions . . . 30
5.2 Motivation . . . 31
5.3 Related Work . . . 32
5.4 Preliminaries . . . 33
5.4.1 Reproducing Kernel Hilbert Space (RKHS) Embeddings . . . 33
5.4.2 Lifting of Control-Affine Systems . . . 35
5.5 System Identification . . . 36
5.5.1 Lifted Matrix Form of Dataset . . . 36
5.5.2 Loss function . . . 37
5.6 Batch Linear State Estimation . . . 38
5.7 Recovering Estimates from Lifted Space . . . 38
5.8 Approximate Embeddings with Random Fourier Features . . . 40
5.8.1 Sketch that KoopSE is Kernelized . . . 40
5.8.2 Substituting with Random Fourier Features . . . 41
5.9 Experiments and Results . . . 42
5.9.1 Problem Setup . . . 42
5.9.2 Simulation Setup . . . 44
5.9.3 Experimental Setup . . . 45
5.10 Results and Discussion . . . 45
5.11 Conclusion . . . 46

6 Koopman-Inspired SLAM **48**
6.1 Summary of Contributions . . . 48
6.2 Motivation . . . 49
6.3 Related Work . . . 50
6.4 Lifting the Full System . . . 51
6.5 System Identification . . . 52
6.5.1 Lifted Matrix Form of Dataset . . . 52
6.5.2 Loss Function . . . 53
6.5.3 Data Augmentation . . . 54
6.6 Unconstrained Koopman Linearization (UKL) . . . 55
6.6.1 UKL Batch Estimation Problem Setup . . . 55
6.6.2 Solving UKL-SLAM . . . 55
6.6.3 Recovering Estimates from Lifted Space . . . 57
6.6.4 Other UKL Estimation Problems . . . 57

6.6.5 Limitations of Unconstrained Optimization in Lifted Spaces . . . 58
6.7 Constrained Koopman Linearization (CKL) . . . 61
6.7.1 CKL-SLAM Problem Setup . . . 61
6.7.2 Solving CKL-SLAM with a Sequential Quadratic Program . . . 62
6.7.3 Other CKL Estimation Problems and SQP Initializations . . . 63
6.8 Reduced Constrained Koopman Linearization (RCKL) . . . 63
6.8.1 Reducing the Koopman Process Model . . . 63
6.8.2 Theoretical Justification of RCKL . . . 65
6.9 Experiments and Results . . . 67
6.9.1 Simulation Setup . . . 67
6.9.2 Simulation Results Discussion . . . 69
6.9.3 Experiment 1: Indoor Navigation with Laser Rangefinder . . . 69
6.9.4 Experiment 2: Golf Cart with RFID Measurements . . . 71
6.9.5 Computation Time . . . 74
6.9.6 Hyperparameter Tuning . . . 76
6.10 Conclusion . . . 76

7 Gaussian Inference on Linearized Manifolds **79**
7.1 Summary of Contributions . . . 79
7.2 Motivation . . . 80
7.3 Related Work . . . 81
7.4 Marginalization and Conditioning Gaussians onto Axis-Aligned Linear Manifolds . . . 82
7.5 Marginalization and Conditioning Gaussians onto General Linear Manifolds . . . 83
7.5.1 General Manifolds . . . 83
7.5.2 Linear Manifolds . . . 84
7.5.3 Expressions for Gaussians . . . 84
7.5.4 Proof for Gaussian Inference on Linear Manifolds . . . 85
7.6 Marginalizing and Conditioning Gaussians onto Linearized Manifolds . . . 87
7.6.1 Approximation Method . . . 87
7.6.2 Theoretical Interpretation as Laplace's Approximation . . . 88
7.7 Applications . . . 89
7.7.1 Approximation of the Projected Normal Distribution . . . 89
7.7.2 Covariance Extraction in Koopman-Inspired Estimation . . . 90
7.7.3 Covariance Extraction in Constrained GTSAM . . . 93
7.8 Conclusion . . . 95

8 Conclusion **96**
8.1 Future Work . . . 97

A Chapter 4 Supplementary Materials **98**
A.1 Measurement Jacobian for $SE_2(3)$ . . . 98

B Chapter 5 Supplementary Materials **99**
B.1 Detailed Sketch that KoopSE is Kernelized . . . 99

C Chapter 6 Supplementary Materials **102**
C.1 Solving UKL-Loc with the RTS smoother 102
C.1.1 Gauss-Newton Setup for UKL-SLAM 103
C.1.2 Solving UKL-SLAM in Linear Time 104
C.1.3 Formulating the SQP for (R)CKL-SLAM 105
C.1.4 Solving (R)CKL-SLAM in Linear Time 106
C.1.5 Proof of (R)CKL-SLAM Covariance Extraction 108
C.1.6 Modifications for Other (R)CKL Estimation Problems 109
C.1.7 SQP Initialization 110
C.1.8 (R)CKL Estimation with Orientation 111

Bibliography **113**

List of Tables

4.1 Dataset splits for cross-validation of KILO-EKF against model-based baselines. . . . 24

5.1 RMSE and Mahalanobis distances of KoopSE and model-based smoother on simulated trajectories 44

6.1 Ablation study on RCKL's learned process and measurement model on Experiment 2 75

7.1 Expressions for marginalization and conditioning Gaussians onto *axis-aligned linear manifolds* 82

7.2 Expressions for marginalization and conditioning Gaussians onto *general linear manifolds* 84

List of Figures

1.1 Conceptual progression of the estimation frameworks developed in this thesis. 2

4.1 Conceptual progression for Chapter 4. 14
4.2 Overview of the proposed KILO-EKF 16
4.3 System overview of the quadrotor platform used to evaluate KILO-EKF 23
4.4 Trajectory estimates for the five cross-validation folds for KILO-EKF 24
4.5 Violin plots of errors across the five cross-validation folds for evaluating KILO-EKF 25
4.6 Error plots of KILO-EKF and model-based estimators 26
4.7 Two ablation studies for KILO-EKF 27
4.8 KILO-EKF inference time versus the number of SERFFs 28
4.9 KILO-EKF training time versus the percentage of training data 28

5.1 Conceptual progression from Chapter 4 to Chapter 5. 30
5.2 KoopSE concept flowchart and experimental setup 32
5.3 Error plots of KoopSE and model-based smoother for simulated trajectories 44
5.4 KoopSE's RMSE vs. number of RFFs and training data points 45
5.5 KoopSE and model-based smoother error plots for dataset experiment 46
5.6 Violin plots of errors across the six cross-validation folds for evaluating KoopSE . . . 47

6.1 Conceptual progression from Chapter 5 to Chapter 6. 48
6.2 Visualization of RCKL-Loc and RCKL-SLAM outputs for Experiment 2 50
6.3 Conceptual illustration of the effects of enforcing manifold consistency in lifted localization and SLAM. 59
6.4 Visualization of poor UKL-Loc and UKL-SLAM outputs, to justify the need for manifold constraints 60
6.5 UKL vs. CKL vs. RCKL in noiseless dead reckoning 66
6.6 Localization and SLAM results for Simulation 1 and Simulation 2 70
6.7 Setup for Experiment 1 71
6.8 Localization and SLAM errors for Experiment 1 to evaluate RCKL 72
6.9 Mahalanobis distances of localization and SLAM for Experiment 1 to evaluate RCKL 73
6.10 Visualization of SLAM output of RCKL-SLAM and MB-SLAM to compare their convergence properties 73
6.11 Error plots of RCKL and model-based estimators for Experiment 2 74
6.12 Error vs. runtime for RCKL and model-based estimators 75
6.13 RCKL hyperparameter sensitivity test on Experiment 1 77

7.1 Conceptual progression from Chapter 6 to Chapter 7. 79
7.2 Marginalizing and conditioning Gaussians onto linear constraint manifolds 81
7.3 Marginalization and conditioning onto axis-aligned manifolds 83
7.4 Visualization of conditioning a Gaussian onto a nonlinear manifold 88
7.5 Application: marginalizing a Gaussian onto a unit circle 90
7.6 Application: extracting RCKL-SLAM covariances via Gaussian conditioning 92
7.7 RCKL-SLAM error plots, with covariances from Gaussian conditioning 92
7.8 Application: extracting constrained GTSAM covariances via Gaussian conditioning . 94
7.9 Mahalanobis distances of constrained GTSAM covariance estimates at different noise levels 94

List of Acronyms

CKL	:	Constrained Koopman Linearization.
EDMD	:	extended dynamic mode decomposition.
EKF	:	extended Kalman filter.
GP	:	Gaussian process.
GTSAM	:	Georgia Tech Smoothing and Mapping.
IMU	:	inertial measurement unit.
KILO-EKF	:	Koopman-Inspired Learned Observations Extended Kalman Filter.
KKT	:	Karush–Kuhn–Tucker.
KoopSE	:	Koopman State Estimator.
LiDAR	:	light detection and ranging.
LTV	:	linear time-varying.
MAP	:	maximum a posteriori.
MPC	:	model predictive control.
NEES	:	normalized estimation error squared.
RCKL	:	Reduced Constrained Koopman Linearization.
RFF	:	random Fourier feature.
RFID	:	radio-frequency identification.
RKHS	:	reproducing kernel Hilbert space.
RMSE	:	root-mean-square error.
RTS	:	Rauch–Tung–Striebel.
SERFF	:	squared-exponential random Fourier feature.
SINDy	:	sparse identification of nonlinear dynamics.
SLAM	:	simultaneous localization and mapping.
SMW	:	Sherman–Morrison–Woodbury.
SQP	:	sequential quadratic program.
UKL	:	Unconstrained Koopman Linearization.
UWB	:	ultra-wideband.

Notation

Mathematical Notation

$\mathbb{R}^{M\times N}$	:	Real coordinate vector space of $M \times N$ matrices.
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	:	Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.
$\mathcal{N}^{-1}(\boldsymbol{\eta}, \boldsymbol{\Lambda})$	:	Gaussian distribution in information form, with information vector $\boldsymbol{\eta}$ and information matrix $\boldsymbol{\Lambda}$.
$\mathbb{E}[\cdot]$	:	Expectation of a random variable.
$SO(3)$	:	Special orthogonal group in three-dimensional space.
$SE(3)$	:	Special Euclidean group in three-dimensional space.
$SE_2(3)$	:	Matrix Lie group extending $SE(3)$ with velocity states.
$\mathfrak{so}(3)$	:	Lie algebra associated with $SO(3)$.
$(\cdot)^\wedge$	:	Hat operator mapping elements of $\mathbb{R}^n$ to elements of the Lie algebra.
$\exp(\cdot)$	:	Exponential map.
$\mathrm{Exp}(\cdot)$	:	Lie exponential map, defined as $\mathrm{Exp}(\cdot) = \exp((\cdot)^\wedge)$.
$\mathrm{vec}(\cdot)$	:	Column-wise vectorization of a matrix.
$\otimes$	:	Kronecker product.
$\odot$	:	Khatri–Rao product.
$(\mathbf{X})^{-\top}$	:	Inverse transpose, $(\mathbf{X}^{-1})^\top$.
$\mathrm{tr}(\mathbf{X})$	:	Trace of a square matrix $\mathbf{X}$.
$(\cdot)^\dagger$	:	Moore–Penrose pseudoinverse.
$\|\cdot\|$	:	Euclidean norm.
$\|\cdot\|_F$	:	Frobenius norm.
$\|\cdot\|_{\mathbf{W}}$	:	Weighted Frobenius norm, $\|\mathbf{X}\|_{\mathbf{W}} = \sqrt{\mathrm{tr}\left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)}$.
$\mathrm{diag}(\mathbf{A}_1, \ldots, \mathbf{A}_N)$	:	Block-diagonal matrix with blocks $\mathbf{A}_1, \ldots, \mathbf{A}_N$.
$\mathrm{span}(\mathbf{X})$	:	Span of the columns of $\mathbf{X}$.
$\mathrm{null}(\mathbf{X})$	:	Matrix whose columns form a basis for the nullspace of $\mathbf{X}$.
$\frac{\partial f}{\partial \mathbf{X}}\Big\|_{\mathbf{A}}$	:	Jacobian of $f : \mathbb{R}^n \to \mathbb{R}^m$ with respect to $\mathbf{X}$, evaluated at $\mathbf{X} = \mathbf{A}$, yielding a matrix in $\mathbb{R}^{m\times n}$.

Conventions

a	:	Symbols in this font are scalars, $a \in \mathbb{R}$.
$\mathbf{A}$	:	Symbols in this font are matrices, $\mathbf{A} \in \mathbb{R}^{M \times N}$.
$\mathbf{a}$	:	Symbols in this font are column vectors, $\mathbf{a} \in \mathbb{R}^{N}$.
$\boldsymbol{a}$	:	Symbols in this font are stacked or batch quantities.
$\mathbf{1}$	:	Identity matrix, with dimensions implied by context.
$\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$	:	Lifting function mapping $\boldsymbol{\xi}$ to a lifted representation.
$\boldsymbol{\xi}$	:	State variable in the original space.
$\mathbf{x}$	:	State variable in the lifted space.
$\boldsymbol{\nu}$	:	Control input in the original space.
$\mathbf{u}$	:	Control input in the lifted space.
$\boldsymbol{\gamma}$	:	Measurement in the original space.
$\mathbf{y}$	:	Measurement in the lifted space.
$\boldsymbol{\psi}$	:	Landmark position in the original space.
$\boldsymbol{\ell}$	:	Landmark position in the lifted space.
$\mathbf{Q}$	:	Process noise covariance.
$\mathbf{R}$	:	Measurement noise covariance.
$\boldsymbol{\lambda}$	:	Lagrange multiplier.
$\boldsymbol{\Sigma}$	:	Covariance matrix of a variable.
$\mathbf{P}$	:	Covariance matrix of a lifted variable.
$\check{(\cdot)}$	:	Prior quantity.
$\hat{(\cdot)}$	:	Posterior quantity.

Chapter 1

Introduction

Autonomous robots must make decisions using only partial and noisy observations of the world. Tasks such as localizing a drone from range measurements, estimating the pose of a mobile robot from cameras and LiDARs, or tracking the state of a legged robot during locomotion all fundamentally rely on *state estimation* [1]. Modern estimation methods combine probabilistic inference with process and measurement models to infer quantities that cannot be directly observed. Their success, however, depends heavily on the accuracy of the available models. In practice, deriving such models can be challenging. Measurement models may depend on complex sensing modalities and environmental interactions, while process models may involve nonlinear dynamics that are difficult to characterize analytically. Consequently, model mismatch often becomes a significant source of estimation error.

An alternative approach is to learn models directly from data rather than deriving them analytically. Recent advances in machine learning have enabled learned representations for dynamics and perception that can capture complex behaviors difficult to model from first principles [2, 3, 4, 5]. However, incorporating learned models into probabilistic estimation frameworks remains challenging. While learned representations can improve model expressiveness, they do not necessarily preserve the structure exploited by classical estimation methods, such as efficient optimization procedures or principled uncertainty representations. This tension between model flexibility and inference structure motivates the central theme of this thesis.

This thesis investigates Koopman-inspired lifting [6, 7] as a framework for incorporating learned models into probabilistic state estimation. Rather than learning state estimates directly, Koopman methods seek transformed representations in which nonlinear systems admit approximately linear structure. Such representations offer the possibility of combining the flexibility of learned models with the system identification and inference procedures used in classical estimation. In particular, they provide a path toward learning process and measurement models from data through structured regression procedures while retaining estimation frameworks such as filtering and nonlinear least-squares optimization. While Koopman methods have been studied extensively for system identification [8, 9], prediction [10], and control [11, 12, 13], extending these ideas to state estimation introduces several additional challenges. Estimation problems must account for noisy measurements and uncertainty, while many robotics systems involve states that evolve on nonlinear manifolds rather than Euclidean vector spaces. Moreover, large-scale problems such as simultaneous localization and mapping (SLAM) require reasoning over thousands of correlated variables while maintaining

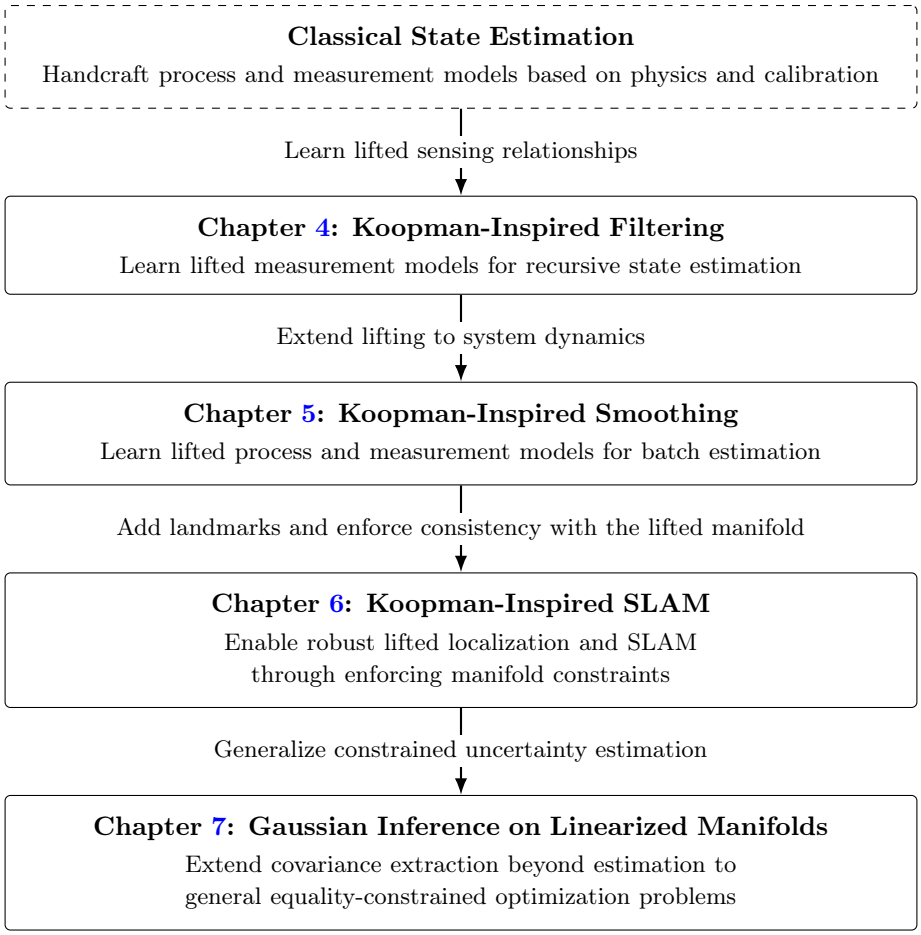


Figure 1.1: Conceptual progression of the estimation frameworks developed in this thesis.

computational tractability. Thus, although there is some prior work on Koopman-based state estimation [14, 15, 16, 17], its application to real-world robotics remains limited. The central question explored in this thesis is how learned lifted representations can be incorporated into probabilistic state estimation while preserving the structure required for efficient and principled inference.

We address this question through a progression of increasingly general problems. In Chapter 4, we begin with filtering, where lifting is applied to measurement models within an extended Kalman filter (EKF). We then extend lifting to both process and measurement models for batch estimation in Chapter 5, enabling both models to be learned directly from data. In Chapter 6, we further extend these ideas to SLAM, addressing the challenges associated with large-scale estimation in lifted spaces. Finally, the uncertainty estimation challenges encountered in lifted SLAM motivate a broader study of Gaussian inference on manifolds, culminating in a general framework for marginalization and conditioning on linearized manifolds in Chapter 7.

1.1 Thesis Organization

The remainder of this thesis is organized as follows. Note that the technical chapters (4, 5, 6, and 7) are not ordered chronologically according to the associated publications, but rather according to the logical progression of the ideas developed throughout the thesis. See Fig. 1.1 for a visual summary.

- **Chapter 2**: We review relevant work in classical state estimation, learning-based estimation, and Koopman-inspired modeling.

- **Chapter 3**: We provide the necessary theoretical background on state estimation and Koopman operator theory.

- **Chapter 4**: We introduce the Koopman-Inspired Learned Observations Extended Kalman Filter (KILO-EKF), which incorporates learned lifted measurement models into a classical EKF. This demonstrates how learned models can be integrated into classical estimation frameworks while preserving their underlying inference structure.

  Associated publication: Z. C. Guo, J. R. Forbes, and T. D. Barfoot, *KILO-EKF: Koopman-inspired learned observations extended Kalman filter*, Accepted to IEEE/RSJ International Conference on Intelligent Robots and Systems, 2026. arXiv: 2601.12463 [cs.RO].

- **Chapter 5**: We extend the lifting approach to both process and measurement models, enabling data-driven batch state estimation through the proposed Koopman State Estimator (KoopSE). This allows complex dynamics and sensing models to be learned directly from data within a unified estimation framework.

  Associated publication: Z. C. Guo, V. Korotkine, J. R. Forbes, and T. D. Barfoot, "Koopman linearization for data-driven batch state estimation of control-affine systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 866–873, 2022. DOI: 10.1109/LRA.2021.3133587.

- **Chapter 6**: We extend Koopman-inspired lifting to localization and SLAM problems by incorporating landmark variables into the learned models. This reveals new challenges associated with large-scale estimation in lifted spaces, which are addressed through the proposed Reduced Constrained Koopman Linearization (RCKL) framework.

  Associated publication: Z. C. Guo, F. Dümbgen, J. R. Forbes, and T. D. Barfoot, "Data-driven batch localization and SLAM using Koopman linearization," *IEEE Transactions on Robotics*, vol. 40, pp. 3964–3983, 2024. DOI: 10.1109/TRO.2024.3443674.

- **Chapter 7**: We develop a general framework that extends classical marginalization and conditioning operations to Gaussian distributions on linearized manifolds, enabling principled uncertainty extraction for equality-constrained estimators. This provides a probabilistic interpretation of uncertainty estimation in constrained optimization problems.

  Associated publication: Z. C. Guo, J. R. Forbes, and T. D. Barfoot, "Marginalizing and conditioning gaussians onto linear approximations of smooth manifolds with applications in robotics," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2606–2612, May 2025. DOI: 10.1109/ICRA55743.2025.11128000.

  Recognition: Recipient of the **Best Conference Paper Award** at ICRA 2025.

Chapter 2

Related Work

This chapter reviews the literature most relevant to the methods developed in this thesis. Rather than providing an exhaustive survey, the discussion focuses on themes central to the proposed approaches, including probabilistic state estimation, estimation on manifolds, learning-based representations, and Koopman-inspired modeling and inference. Particular emphasis is placed on methods that preserve structure useful for probabilistic inference while remaining scalable to larger robotics estimation problems.

2.1 Classical State Estimation

2.1.1 Filtering, Smoothing, and SLAM

Classical state estimation methods infer system states from noisy process and measurement data using probabilistic inference [1, 22]. Common robotics estimation formulations include filtering, smoothing, and SLAM. Filtering methods, including the Kalman filter [23] and its variants [24], estimate states recursively using only past measurements. Smoothing methods, such as the Rauch–Tung–Striebel (RTS) smoother [25], incorporate future measurements to jointly estimate a trajectory. In practice, smoothing can be performed either over a fixed window or over the entire trajectory through batch optimization. When the environment map is also unknown, the problem becomes SLAM, where the robot trajectory and landmark positions are estimated jointly [26, 27].

These and many other modern estimators rely on local linearization and Gaussian inference. For linear-Gaussian systems, inference often admits closed-form solutions. However, nonlinear robotics problems are typically solved through iterative local linearization methods such as Gauss–Newton or Levenberg–Marquardt optimization [28, 1]. Over time, unifying optimization-based frameworks such as factor graphs [29, 30] have emerged for localization and SLAM. Most methods considered in this thesis operate in discrete time, though continuous-time estimation methods have also received attention in robotics, particularly for applications involving asynchronous sensing or trajectory interpolation [31].[1]

The methods above primarily rely on Gaussian inference in locally linear spaces. However, many robotics estimation problems also involve geometric constraints or manifold structure, motivating

[1] Some continuous-time estimation problems can also be represented or approximated within discrete-time formulations. See [32] for examples.

extensions of classical inference methods to constrained and nonlinear spaces.

2.1.2 Estimation on Manifolds

Many estimation and optimization problems in robotics involve manifold structures. States such as orientations and poses in a three-dimensional environment lie on Lie groups [33], requiring estimation methods that account for their nonlinear geometry [1]. A range of EKF formulations have been developed directly on matrix Lie groups, representing uncertainty through concentrated Gaussian distributions and deriving prediction and update operations using Lie-group structure [34, 35, 36]. Related invariant and equivariant filtering approaches further exploit symmetry properties of the system to obtain structured error dynamics [37, 38]. Many such methods formulate inference locally in tangent spaces or minimal coordinates, allowing classical Gaussian techniques to be applied through local linearization. Gaussian distributions can likewise be approximated and fused through these local Euclidean representations [39]. Thus, for important manifold classes such as Lie groups, both mean estimation and uncertainty representations are relatively well developed.

In other scenarios, manifold structure arises from explicit, problem-specific constraints rather than from a standard geometric state space. For example, Chapter 7 considers estimation of a box trajectory constrained to remain in contact with a moving probe. Such problems are commonly addressed through constrained nonlinear optimization, often using sequential quadratic programming (SQP) [28], which repeatedly linearizes the objective and constraints and solves a local constrained approximation. Constrained factor-graph methods for localization and SLAM [40, 41] follow this general paradigm.

Although these approaches can produce valid mean estimates, covariance estimation and uncertainty propagation are less broadly developed for problem-specific constrained manifolds. Existing methods are often tailored either to particular constrained estimation problems [42, 43] or to specific manifold classes, such as circles [44], spheres [45], and Lie groups. More general formulations for Gaussian inference and covariance estimation on arbitrary smooth manifolds remain limited. This gap is particularly important in this thesis, where Koopman-inspired lifted representations introduce additional manifold structure requiring Gaussian inference beyond classical Lie-group settings.

2.2 Learning-Based State Estimation

2.2.1 General Learning-Based Modeling and Estimation

Learning-based state estimation methods use data-driven models to infer system states from control inputs and measurements. These approaches have become increasingly important in robotics, where accurate analytical models may be difficult to derive due to complex dynamics, sensing modalities, or environmental interactions. Compared to classical model-based estimators, learning-based methods can capture richer nonlinear behaviors directly from data while still being integrated into probabilistic estimation frameworks.

A wide range of learned representations have been proposed, including those based on neural networks, kernels [46], Gaussian processes [47], and the Koopman operator [6, 7]. Neural-network-based representations [2, 3, 4, 5] are often highly expressive and can learn complex sensing and dynamical relationships directly from data. However, incorporating such learned representations into

probabilistic estimation frameworks remains challenging, as the learned features are not generally constrained to preserve the structure exploited by classical estimation methods. This can complicate the use of classical inference algorithms and uncertainty representations.

On the other hand, kernel methods and Gaussian processes provide flexible nonparametric representations of nonlinear functions and probability distributions while preserving structure useful for probabilistic inference. In particular, lifted feature representations in kernel methods may admit simpler or approximately linear representations [48, 49]. These ideas have motivated the use of kernels [50, 51, 52] and Gaussian processes [53, 54] for data-driven modeling and state estimation. These methods often scale poorly with the amount of training data and require approximations for real-time deployment [55]. As a result, many of these methods have primarily been validated on relatively small-scale state-estimation problems and controlled experimental settings, with more limited adoption in large-scale real-time robotics and SLAM systems. In practice, many approaches also impose structural assumptions on the system dynamics or noise models to maintain tractability [56]. These challenges motivate the use of Koopman-based representations, which aim to preserve structure useful for inference while remaining scalable to larger robotics problems.

2.2.2 Koopman-Based Modeling, Control, and Estimation

Koopman-based methods represent nonlinear systems using approximately linear dynamics in lifted feature spaces [6, 7]. Rather than directly modeling nonlinear state evolution, these approaches seek feature representations in which the dynamics admit simpler linear or bilinear [13] structure. In robotics, this perspective has attracted significant attention in system identification [8, 9] and control [11, 12, 13] due to its ability to represent nonlinear systems using approximately linear dynamics compatible with classical linear systems techniques [7]. Their finite-dimensional lifted structure has enabled computationally efficient learning and inference for larger-scale robotics modeling and control [12, 57].

A central challenge in Koopman-based modeling is the selection of lifting functions, often referred to as *observables* in Koopman literature, used to represent the system in the lifted space. Existing approaches include hand-designed observables motivated by system structure [11, 58, 59, 60, 61], randomized feature representations [62, 63], and learned liftings based on neural networks [64, 65]. Finite-dimensional Koopman models are often constructed from candidate observables using methods such as Extended Dynamic Mode Decomposition (EDMD) [66, 67] or sparse identification techniques [68, 69]. While learned liftings are often more expressive, structured observables and randomized features remain attractive due to their simplicity and scalability, as well as compatibility with classical systems-theoretic techniques. The methods proposed in this thesis primarily leverage these structured lifted representations for state estimation.

Many of the practical methods discussed above learn lifted state-space models directly from data. However, these lifted representations are generally only approximately invariant under nonlinear dynamics, leading to degraded performance over long time horizons [70] even with reprojection approximations [71, 72]. Consequently, a major focus of the Koopman literature is the construction of Koopman-invariant subspaces and Koopman eigenfunctions from data [73, 74, 10]. These approaches are particularly attractive for prediction and control tasks, where preserving long-horizon dynamical consistency is important. This Koopman mode and eigenvalue perspective, however, is less naturally aligned with stochastic inference problems such as state estimation, where uncertainty

representations, measurement models, and probabilistic inference are often more central than exact long-horizon dynamical invariance [74, 10].

In general, Koopman-inspired methods for state estimation remain relatively limited. Existing work includes Koopman observer constructions for constrained estimation [14, 15], as well as Koopman Kalman filters based on spectral or kernel representations [16, 17]. These methods focus on constructing Koopman representations with the particular structure required by their estimators, such as observer forms, spectral coordinates, or kernel approximations. In practice, obtaining such representations and applying the resulting estimators at scale can be challenging, particularly for robotics problems with noisy measurements, model mismatches, and geometric state spaces such as Lie groups. Consequently, existing evaluations have largely remained in relatively low-dimensional or specialized settings.

More broadly, much of the Koopman literature focuses on prediction and control, whereas robotics estimation problems such as filtering, smoothing, and SLAM require probabilistic inference over coupled variables while maintaining meaningful uncertainty representations. The structural challenges introduced by lifted representations, including manifold consistency and uncertainty estimation, remain comparatively underexplored in Koopman-inspired probabilistic estimation.

This thesis aims to bridge this gap by developing Koopman-inspired methods for data-driven filtering, smoothing, and SLAM, validated on large-scale simulations and real-world datasets. Rather than requiring the explicit recovery of Koopman eigenfunctions or invariant subspaces, the proposed methods construct lifted representations that preserve structure useful for probabilistic inference and, when required, enforce consistency with the underlying lifted manifold.

Chapter 3

Background

This chapter reviews the mathematical and algorithmic background underlying the methods developed in this thesis. We first review probabilistic state estimation and its connection to optimization, highlighting how inference problems in robotics can be formulated as nonlinear least-squares estimation problems. This optimization perspective underlies the estimation frameworks developed throughout the thesis, including the filtering, smoothing, and SLAM formulations introduced in Chapters 4, 5, and 6. We also highlight the roles of marginalization and conditioning operations, which are fundamental to probabilistic inference and later form the basis for the developments in Chapter 7. We then review Koopman operator theory, which seeks lifted representations in which nonlinear systems admit approximately linear dynamics. This provides the foundation for the learned lifted process and measurement models developed in Chapters 4, 5, and 6.

3.1 State Estimation

In this section, we review probabilistic state estimation for dynamical systems and its connection to optimization. We introduce filtering and batch estimation formulations, along with the nonlinear least-squares perspective underlying many modern robotics estimation methods. We also discuss the roles of marginalization and conditioning operations in probabilistic inference. See [1] for a comprehensive treatment of state estimation in robotics.

3.1.1 Problem Formulation

Many robotics problems, including localization and mapping, can be viewed as estimating the hidden state of a dynamical system. Since the system state cannot typically be observed directly and measurements are corrupted by noise, state estimation is naturally formulated as a probabilistic inference problem. Suppose we aim to estimate the state $\boldsymbol{\xi}_k$ at timestep k, which may represent quantities such as robot pose, robot velocity, or map variables. Rather than estimating a deterministic $\boldsymbol{\xi}_k$, the objective is instead to infer a probability distribution, or belief, over $\boldsymbol{\xi}_k$ conditioned on noisy measurements and inputs.

State estimation problems are typically modeled with a process model and a measurement model:

$$\boldsymbol{\xi}_k = \mathbf{f}(\boldsymbol{\xi}_{k-1}, \boldsymbol{\nu}_k) + \boldsymbol{\epsilon}_k, \tag{3.1a}$$

$$\boldsymbol{\gamma}_k = \mathbf{g}(\boldsymbol{\xi}_k) + \boldsymbol{\eta}_k, \tag{3.1b}$$

where (3.1a) describes the system dynamics governing the evolution of the state from timestep $k-1$ to timestep k, and (3.1b) describes how the measurement depends on the state at timestep k. The measurements $\boldsymbol{\gamma}_k$ provide partial and noisy observations of the state, and the input $\boldsymbol{\nu}_k$ may represent control commands or other exogenous signals. The noise terms $\boldsymbol{\epsilon}_k$ and $\boldsymbol{\eta}_k$ capture uncertainty in the process and measurement models, respectively. Typically, we assume that the process and measurement noises are zero-mean Gaussians with covariances $\mathbf{Q}$ and $\mathbf{R}$, respectively:

$$\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \quad \boldsymbol{\eta}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}). \tag{3.2}$$

The state estimation objective may be to estimate either the current state or the full state trajectory. Recursive filtering methods propagate and update a belief over the current state $\boldsymbol{\xi}_k$ as new measurements and inputs arrive, yielding the posterior belief $p(\boldsymbol{\xi}_k \mid \boldsymbol{\gamma}_{1:k}, \boldsymbol{\nu}_{1:k})$. Conceptually, filtering repeatedly conditions on new measurements while marginalizing past states. Classical examples include the Kalman filter and its nonlinear variants [24], which are widely used in robotics due to their conceptual simplicity, computational efficiency, and suitability for real-time applications.

In batch estimation or smoothing problems, the objective is instead to estimate the full trajectory, $\boldsymbol{\xi}_{0:k}$, conditioned on all measurements, $\boldsymbol{\gamma}_{1:k}$, and inputs, $\boldsymbol{\nu}_{1:k}$. That is, we estimate the posterior distribution $p(\boldsymbol{\xi}_{0:k} \mid \boldsymbol{\gamma}_{1:k}, \boldsymbol{\nu}_{1:k})$. Unlike recursive filtering methods, batch formulations jointly infer states across multiple timesteps simultaneously, enabling future measurements to improve estimates of past states and allowing stronger exploitation of correlations across the trajectory. As a result, batch methods are often more accurate and flexible for large-scale robotics problems such as mapping and SLAM, where one frequently marginalizes or conditions subsets of variables to extract local state estimates and covariances from larger joint distributions. More generally, batch estimation encompasses formulations ranging from fixed-lag smoothing to full trajectory estimation. Although this thesis begins with a filtering formulation in Chapter 4, we primarily focus on batch estimation methods in Chapters 5, 6, and 7.

3.1.2 Connection to Optimization

We now review the connection between probabilistic inference and optimization in batch estimation problems, although many of the same ideas also apply to recursive filtering methods. See [1, §3.1] for more details. Using Bayes' rule, the posterior distribution factorizes as

$$p(\boldsymbol{\xi}_{0:k} \mid \boldsymbol{\gamma}_{1:k}, \boldsymbol{\nu}_{1:k}) \propto p(\boldsymbol{\xi}_{0:k} \mid \boldsymbol{\nu}_{1:k}) p(\boldsymbol{\gamma}_{1:k} \mid \boldsymbol{\xi}_{0:k}) \tag{3.3a}$$

$$= p(\boldsymbol{\xi}_0) \prod_{i=1}^{k} p(\boldsymbol{\xi}_i \mid \boldsymbol{\xi}_{i-1}, \boldsymbol{\nu}_i) \prod_{i=1}^{k} p(\boldsymbol{\gamma}_i \mid \boldsymbol{\xi}_i), \tag{3.3b}$$

where the last line follows from the Markov assumption together with conditional independence of the measurements. For general nonlinear systems, representing and computing the full posterior

distribution is often intractable. Instead, many state estimation methods seek a point estimate corresponding to the maximum a posteriori (MAP) solution. This estimate is obtained by maximizing the posterior, or equivalently minimizing the negative log posterior:

$$\hat{\boldsymbol{\xi}}_{0:k} = \operatorname*{argmax}_{\boldsymbol{\xi}_{0:k}} \; p(\boldsymbol{\xi}_{0:k} \mid \boldsymbol{\gamma}_{1:k}, \boldsymbol{\nu}_{1:k}) \tag{3.4a}$$

$$= \operatorname*{argmin}_{\boldsymbol{\xi}_{0:k}} \; -\log p(\boldsymbol{\xi}_{0:k} \mid \boldsymbol{\gamma}_{1:k}, \boldsymbol{\nu}_{1:k}) \tag{3.4b}$$

$$= \operatorname*{argmin}_{\boldsymbol{\xi}_{0:k}} \; -\left[\log p(\boldsymbol{\xi}_0) + \sum_{i=1}^{k} \log p(\boldsymbol{\xi}_i \mid \boldsymbol{\xi}_{i-1}, \boldsymbol{\nu}_i) + \sum_{i=1}^{k} \log p(\boldsymbol{\gamma}_i \mid \boldsymbol{\xi}_i)\right]. \tag{3.4c}$$

We assume that we have a Gaussian prior on the initial state: $\boldsymbol{\xi}_0 \sim \mathcal{N}(\check{\boldsymbol{\xi}}_0, \check{\boldsymbol{\Sigma}}_0)$. Under the assumption that the model noises are zero-mean Gaussians (3.2), this yields a weighted nonlinear least-squares problem of the form $\hat{\boldsymbol{\xi}}_{0:k} = \operatorname{argmin}_{\boldsymbol{\xi}_{0:k}} J(\boldsymbol{\xi}_{0:k})$, where the objective is given by

$$J(\boldsymbol{\xi}_{0:k}) = \frac{1}{2}\mathbf{e}_0^\top \boldsymbol{\Sigma}_0^{-1}\mathbf{e}_0 + \frac{1}{2}\sum_{i=1}^{k} \mathbf{e}_{\boldsymbol{\nu},i}^\top \mathbf{Q}_i^{-1}\mathbf{e}_{\boldsymbol{\nu},i} + \frac{1}{2}\sum_{i=1}^{k} \mathbf{e}_{\boldsymbol{\gamma},i}^\top \mathbf{R}_i^{-1}\mathbf{e}_{\boldsymbol{\gamma},i}, \tag{3.5}$$

where $\mathbf{e}_0$, $\mathbf{e}_{\boldsymbol{\nu},i}$, and $\mathbf{e}_{\boldsymbol{\gamma},i}$ are the initial, process, and measurement residuals, respectively, defined as

$$\mathbf{e}_0(\boldsymbol{\xi}_0) = \boldsymbol{\xi}_0 - \check{\boldsymbol{\xi}}_0, \qquad \mathbf{e}_{\boldsymbol{\nu},i}(\boldsymbol{\xi}_{i-1}, \boldsymbol{\xi}_i) = \boldsymbol{\xi}_i - \mathbf{f}(\boldsymbol{\xi}_{i-1}, \boldsymbol{\nu}_i), \qquad \mathbf{e}_{\boldsymbol{\gamma},i}(\boldsymbol{\xi}_i) = \boldsymbol{\gamma}_i - \mathbf{g}(\boldsymbol{\xi}_i). \tag{3.6}$$

We can then solve for the MAP estimate, $\hat{\boldsymbol{\xi}}_{0:k}$, using nonlinear least-squares optimization methods such as Gauss–Newton or Levenberg–Marquardt [1, 28].

In addition to the point estimate, we are often also interested in characterizing the uncertainty of the estimate. A common approach is to use *Laplace's approximation* [75] to obtain a local Gaussian approximation of the posterior distribution using the inverse Hessian of the negative log posterior evaluated at the MAP solution:

$$\hat{\boldsymbol{\Sigma}}_{0:k} \approx \left(\left.\frac{\partial^2 J(\boldsymbol{\xi}_{0:k})}{\partial \boldsymbol{\xi}_{0:k}^2}\right|_{\hat{\boldsymbol{\xi}}_{0:k}}\right)^{-1}. \tag{3.7}$$

We would then characterize the solution as $\boldsymbol{\xi}_{0:k} \sim \mathcal{N}(\hat{\boldsymbol{\xi}}_{0:k}, \hat{\boldsymbol{\Sigma}}_{0:k})$. Although the true posterior is generally non-Gaussian for nonlinear systems, Laplace's approximation is widely used in practice to obtain local Gaussian uncertainty estimates around the MAP solution. The resulting Gaussian approximation additionally supports efficient marginalization and conditioning operations over subsets of variables, which are central to many sparse inference and SLAM algorithms.

In general, computing the MAP estimate for nonlinear systems requires iterative optimization. However, the situation simplifies significantly for linear-Gaussian systems, where the process and measurement models take the form

$$\boldsymbol{\xi}_k = \mathbf{A}_{k-1}\boldsymbol{\xi}_{k-1} + \mathbf{B}_k\boldsymbol{\nu}_k + \boldsymbol{\epsilon}_k, \tag{3.8a}$$

$$\boldsymbol{\gamma}_k = \mathbf{D}_k\boldsymbol{\xi}_k + \boldsymbol{\eta}_k. \tag{3.8b}$$

In this case, the posterior distribution is exactly Gaussian, and the MAP estimate coincides with

the posterior mean. Moreover, the posterior mean and covariance can be computed efficiently using sparse linear algebra methods such as the Cholesky smoother and other batch solvers [1]. This observation motivates a central theme of this thesis: while nonlinear systems generally require iterative approximate inference, systems with linear-Gaussian structure admit efficient and scalable estimation algorithms. This motivates the use of Koopman operator theory, which seeks lifted representations in which nonlinear systems admit approximately linear structure.

3.2 Koopman Lifting for Dynamical Systems

In this section, we briefly review Koopman lifting [6] for dynamical systems and its use in learning finite-dimensional linear representations of nonlinear systems. We begin with the canonical definition of the Koopman operator before introducing practical finite-dimensional approximations and their associated challenges. For comprehensive treatments of the Koopman operator, see [67, 7].

3.2.1 Koopman Operator Theory

Koopman lifting has been driven largely by its historical use in modeling dynamical systems [11, 76]. The goal is to represent nonlinear dynamics as a linear evolution in a higher-dimensional space. Consider a noiseless discrete-time autonomous (i.e., no inputs) system,

$$\boldsymbol{\xi}_k = \mathbf{f}(\boldsymbol{\xi}_{k-1}). \tag{3.9}$$

The Koopman operator, $\boldsymbol{\mathcal{K}}$, is canonically defined as an operator acting on scalar-valued lifting functions, $p_{\boldsymbol{\xi}}(\cdot)$, according to

$$(\boldsymbol{\mathcal{K}} p_{\boldsymbol{\xi}})(\boldsymbol{\xi}) = p_{\boldsymbol{\xi}}(\mathbf{f}(\boldsymbol{\xi})). \tag{3.10}$$

These lifting functions are termed *observables* in the Koopman literature[1], and they can be thought of as features that extract relevant information from the original state. Importantly, the Koopman operator is linear on the space of observables, denoted $\mathcal{F}$. That is, for any observables $p_{\boldsymbol{\xi},1}, p_{\boldsymbol{\xi},2} \in \mathcal{F}$ and scalars $a, b \in \mathbb{R}$,

$$\boldsymbol{\mathcal{K}}\left(a p_{\boldsymbol{\xi},1} + b p_{\boldsymbol{\xi},2}\right) = a\boldsymbol{\mathcal{K}} p_{\boldsymbol{\xi},1} + b\boldsymbol{\mathcal{K}} p_{\boldsymbol{\xi},2}. \tag{3.11}$$

Thus, the Koopman framework shifts nonlinear state evolution into the space of observables, where the dynamics are governed by a linear operator.

Note that the Koopman operator is generally infinite-dimensional. As such, one typically constructs finite-dimensional approximations using a selected collection of observables stacked into a vector-valued lifting function (i.e., embedding),

$$\mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}) = \begin{bmatrix} p_{\boldsymbol{\xi},1}(\boldsymbol{\xi}) & \cdots & p_{\boldsymbol{\xi},n}(\boldsymbol{\xi}) \end{bmatrix}^\top. \tag{3.12}$$

We define the lifted state as the stacked observables, $\mathbf{x}_k = \mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k)$. Then, we have from (3.9) and

[1] Not to be confused with the concept of *observability* in control theory.

(3.10) that

$$\begin{bmatrix} p_{\boldsymbol{\xi},1}(\boldsymbol{\xi}_k) \\ \vdots \\ p_{\boldsymbol{\xi},n}(\boldsymbol{\xi}_k) \end{bmatrix} = \begin{bmatrix} (\mathcal{K}p_{\boldsymbol{\xi},1})(\boldsymbol{\xi}_{k-1}) \\ \vdots \\ (\mathcal{K}p_{\boldsymbol{\xi},n})(\boldsymbol{\xi}_{k-1}) \end{bmatrix} \implies \mathbf{x}_k = (\mathcal{K}\mathbf{p}_{\boldsymbol{\xi}})(\boldsymbol{\xi}_{k-1}). \tag{3.13}$$

For certain choices of observables in (3.12), the span of the observables may be closed under the Koopman operator. In that case, the observables span a *Koopman-invariant subspace*, and the Koopman operator admits a finite-dimensional matrix representation, $\mathbf{A}$. The lifted dynamics in (3.13) can then be written in the familiar linear form:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1}. \tag{3.14}$$

As a practical example from [77], suppose that a 2-dimensional system is governed by the nonlinear dynamics

$$\begin{bmatrix} \xi_{1,k} \\ \xi_{2,k} \end{bmatrix} = \mathbf{f}\left(\begin{bmatrix} \xi_{1,k-1} \\ \xi_{2,k-1} \end{bmatrix}\right) = \begin{bmatrix} \lambda\xi_{1,k-1} \\ \mu\xi_{2,k-1} + (\lambda^2 - \mu)\xi_{1,k-1}^2 \end{bmatrix}, \tag{3.15}$$

for some parameters λ and μ. In this case, the nonlinear system can be exactly transformed into a linear system with the lifting function $\mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k) = \begin{bmatrix} \xi_{1,k} & \xi_{2,k} & \xi_{1,k}^2 \end{bmatrix}^\top = \begin{bmatrix} x_{1,k} & x_{2,k} & x_{3,k} \end{bmatrix}^\top$, yielding

$$\begin{bmatrix} x_{1,k} \\ x_{2,k} \\ x_{3,k} \end{bmatrix} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \mu & \lambda^2 - \mu \\ 0 & 0 & \lambda^2 \end{bmatrix} \begin{bmatrix} x_{1,k-1} \\ x_{2,k-1} \\ x_{3,k-1} \end{bmatrix}. \tag{3.16}$$

3.2.2 Learning Finite-Dimensional Koopman Models

In practice, exact finite-dimensional Koopman invariant subspaces are rare, particularly for noisy real-world systems. As such, finite-dimensional Koopman representations are typically approximated from data using methods such as EDMD [66, 67]. Although the methods developed later in this thesis extend beyond the standard EDMD setting, they retain the same underlying perspective of learning lifted linear models from data.

EDMD learns a best-fit linear operator that approximates the lifted dynamics in (3.14) from trajectories of lifted observables. Suppose we are given a dataset of P state transitions, $\{\boldsymbol{\xi}_i^-, \boldsymbol{\xi}_i^+\}_{i=1}^P$, where $\boldsymbol{\xi}_i^-$ transitions to $\boldsymbol{\xi}_i^+$. We write the data in block-matrix form as

$$\boldsymbol{\Xi}_{(-)} = \begin{bmatrix} \boldsymbol{\xi}_1^- & \cdots & \boldsymbol{\xi}_P^- \end{bmatrix}, \quad \boldsymbol{\Xi}_{(+)} = \begin{bmatrix} \boldsymbol{\xi}_1^+ & \cdots & \boldsymbol{\xi}_P^+ \end{bmatrix}. \tag{3.17}$$

Applying the lifting function gives the lifted snapshots $\mathbf{x}_i^- = \mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_i^-)$, $\mathbf{x}_i^+ = \mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_i^+)$, which are stacked as

$$\mathbf{X}_{(-)} = \begin{bmatrix} \mathbf{x}_1^- & \cdots & \mathbf{x}_P^- \end{bmatrix}, \quad \mathbf{X}_{(+)} = \begin{bmatrix} \mathbf{x}_1^+ & \cdots & \mathbf{x}_P^+ \end{bmatrix}. \tag{3.18}$$

Under the lifted linear dynamics model, the dataset satisfies $\mathbf{X}_{(+)} \approx \mathbf{A}\mathbf{X}_{(-)}$. EDMD computes a

least-squares estimate of $\mathbf{A}$:

$$\mathbf{A}^\star = \underset{\mathbf{A}}{\operatorname{argmin}} \|\mathbf{X}_{(+)} - \mathbf{A}\mathbf{X}_{(-)}\|_F^2, \tag{3.19}$$

where $\|\cdot\|_F$ is the Frobenius norm. This admits the closed-form solution $\mathbf{A}^\star = \mathbf{X}_{(+)}\mathbf{X}_{(-)}^\dagger$, where $(\cdot)^\dagger$ represents the Moore–Penrose pseudoinverse. This can also be interpreted as maximum likelihood estimation under additive Gaussian residuals. If desired, regularization or Bayesian priors may also be incorporated to improve robustness and generalization.

Several challenges arise when applying Koopman lifting to real-world systems. Although finite-dimensional Koopman approximations can be effective [77], their quality depends strongly on the selected observables, which are often designed using domain knowledge or learned from data. Moreover, while Koopman lifting naturally extends to controlled systems [76], robotics problems additionally involve noisy processes and measurements. Existing Koopman identification methods such as EDMD are primarily formulated as deterministic least-squares regression problems, which do not explicitly model uncertainty or covariance structure. Incorporating stochastic process and measurement models into the lifted representation therefore remains challenging, and there is comparatively limited work on using Koopman methods for probabilistically consistent estimation in the presence of noise, inputs, and measurements. These challenges motivate the developments in this thesis, where we extend Koopman lifting to stochastic systems with noisy process and measurement models.

Chapter 4

Koopman-Inspired Filtering

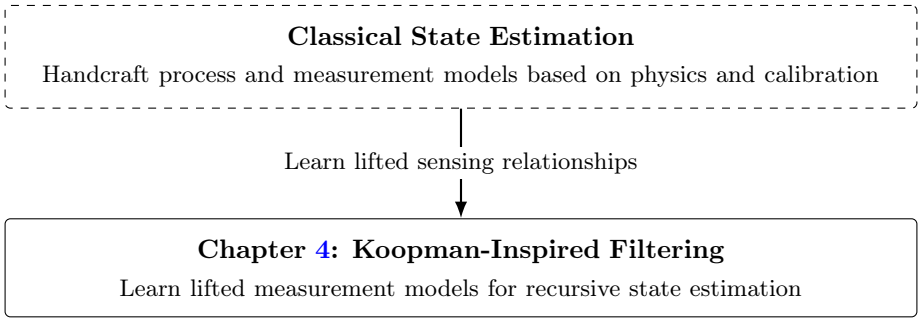


Figure 4.1: Conceptual progression for Chapter 4.

4.1 Summary of Contributions

This chapter introduces the Koopman-Inspired Learned Observations Extended Kalman Filter (KILO-EKF), a filtering framework that incorporates data-driven measurement models into the EKF. The method learns a lifted linear-Gaussian measurement model directly from data while preserving compatibility with classical filtering algorithms on Lie groups. This chapter

- introduces the KILO-EKF framework for learning measurement models within a filtering architecture,
- demonstrates that Koopman-inspired lifting can be integrated with Lie-group filtering while preserving the recursive structure of the EKF, and
- empirically demonstrates improved localization performance over classical EKF baselines on representative localization tasks.

Within the broader context of this thesis, this chapter represents the first step toward incorporating data-driven modeling into robotic state estimation while preserving the structure required for efficient filtering. Subsequent chapters extend this idea by learning both process and measurement models in a batch estimation framework, and by addressing structural issues that arise in large-scale problems such as SLAM.

Associated publication: Z. C. Guo, J. R. Forbes, and T. D. Barfoot, *KILO-EKF: Koopman-inspired learned observations extended Kalman filter*, Accepted to IEEE/RSJ International Conference on Intelligent Robots and Systems, 2026. arXiv: 2601.12463 [cs.RO].

4.2 Motivation

Recursive state estimation methods such as the EKF and its variants [24] are fundamental to many robotic systems requiring real-time operation. These methods rely on accurate process and measurement models to produce reliable state estimates and uncertainty representations [1, 22]. In practice, however, deriving precise analytical measurement models is often difficult due to sensing nonlinearities, calibration uncertainty, and environmental effects. For sensing modalities such as ultra-wideband (UWB) [78] and radio-frequency identification (RFID) [79], collecting training data may be significantly easier than constructing accurate analytical models and noise characteristics. As a result, model mismatch can substantially degrade estimation performance or even lead to filter divergence [22, 80].

These challenges motivate learning-based approaches that incorporate data-driven models into classical estimation frameworks. Among these, the Koopman operator theory [6] is attractive because it seeks to transform nonlinear relationships into simpler, approximately linear structure. Unlike highly expressive end-to-end neural-network-based estimators, Koopman-inspired representations can preserve structure useful for Gaussian inference while remaining computationally efficient for recursive filtering. Prior work has applied Koopman operator theory to system identification [76, 8] and control [11, 13, 58]. However, applications to probabilistic state estimation remain relatively limited, particularly for Lie-group states involving poses and orientations.

In this chapter, we introduce the KILO-EKF, which applies Koopman-inspired lifting to the measurement model of a classical EKF. Rather than learning the full system dynamics, KILO-EKF learns a lifted representation of the observation function while leaving the process model unchanged. This preserves compatibility with recursive filtering and geometric state representations such as Lie-group formulations used in inertial navigation. The resulting framework combines flexible data-driven measurement modeling with the structure and computational efficiency of classical filtering.

This chapter is organized as follows. Section 4.3 reviews related work on Koopman-based modeling and data-driven state estimation. Section 4.4 summarizes the standard EKF formulation on $\mathrm{SE}_2(3)$. Section 4.5 presents the general KILO-EKF framework, and Section 4.6 specializes the framework to $\mathrm{SE}_2(3)$. Section 4.7 experimentally validates KILO-EKF on a real-world dataset, and Section 4.8 concludes the chapter.

4.3 Related Work

Recent work in learning-based state estimation has explored integrating data-driven models into classical filtering frameworks. Neural-network-based methods, including differentiable filters [2], learned filtering frameworks [5], and end-to-end odometry systems [3, 4], are often highly expressive and can learn complex sensing relationships directly from data. However, these approaches typically require substantial training data and computational resources while sacrificing structure useful for probabilistic inference, making interpretability and integration with classical estimation frameworks

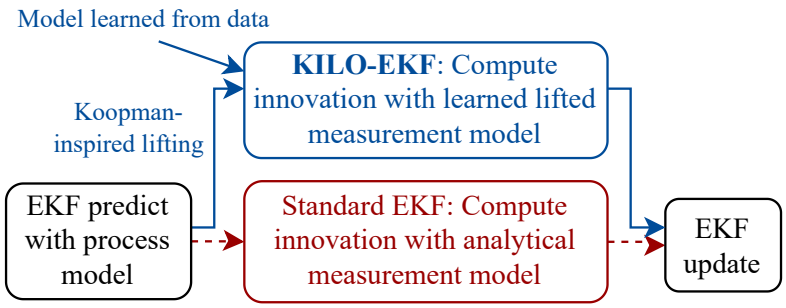


Figure 4.2: Overview of the proposed KILO-EKF: the standard EKF structure is retained, while the measurement model is replaced with a Koopman-inspired, data-driven representation.

more challenging.

Another line of work uses kernels and Gaussian processes to learn nonlinear system and measurement models for probabilistic inference [51, 52, 53]. Compared to neural-network-based methods, these approaches are typically less expressive but provide more structured probabilistic representations and can naturally model uncertainty. However, they incur cubic training complexity and require storing or approximating the training set at inference time, limiting real-time use without significant approximations [55].

Koopman-inspired methods instead seek lifted representations in which nonlinear dynamics or measurements admit simpler approximately linear structure. Existing Koopman-based estimation approaches have primarily focused on constructing lifted system representations with the particular structure required by a chosen estimator, including Koopman observer formulations for constrained estimation and lifted Kalman-filter models based on spectral or kernel representations [14, 15, 16, 17]. These approaches have largely considered vector-space states and simplified estimation settings. Although Koopman modeling has been explored on SO(3) [59] and SE(3) [60, 61], these works primarily consider system modeling and control rather than probabilistic filtering on Lie groups.

Motivated by these gaps, this chapter develops a Koopman-inspired learned measurement model within a classical EKF framework. The proposed approach preserves the structure and real-time performance of Lie-group filtering while enabling flexible data-driven measurement representations without requiring constrained lifted optimization.

4.4 Preliminaries

4.4.1 Standard Discrete-Time EKF on $\mathrm{SE}_2(3)$

In this section, we briefly summarize a standard formulation for discrete-time EKF on $\mathrm{SE}_2(3)$ and refer the reader to [81] for more details.

We begin with a general nonlinear system of the form,

$$\boldsymbol{\xi}_k = \mathbf{f}(\boldsymbol{\xi}_{k-1}, \boldsymbol{\nu}_k, \boldsymbol{\epsilon}_k), \tag{4.1a}$$

$$\boldsymbol{\gamma}_k = \mathbf{g}(\boldsymbol{\xi}_k, \boldsymbol{\eta}_k). \tag{4.1b}$$

where $\boldsymbol{\xi}_k \in \mathbb{R}^{N_\xi}$ is the state, $\boldsymbol{\nu}_k \in \mathbb{R}^{N_\nu}$ the control input, $\boldsymbol{\omega}_k \in \mathbb{R}^{N_\omega}$ the process noise, $\boldsymbol{\gamma}_k \in \mathbb{R}^{N_\gamma}$ the

measurement output, and $\boldsymbol{\eta}_k \in \mathbb{R}^{N_\eta}$ the measurement noise, all at timestep k.

Many robotic systems operating in three-dimensional space, such as aerial vehicles and legged platforms, are equipped with an inertial measurement unit (IMU). From the bias-corrected IMU readings, we construct the input $\boldsymbol{\nu}_k$, consisting of angular velocity $\boldsymbol{\omega}_k$ and specific force $\mathbf{a}_k$. These interoceptive measurements, together with additional exteroceptive sensors, are commonly fused using an EKF on the Lie group $\mathrm{SE}_2(3)$. Accordingly, we define the EKF state as $\boldsymbol{\xi}_k = \{\mathbf{T}_k,\ \mathbf{b}_k\} \in \mathrm{SE}_2(3) \times \mathbb{R}^6$, where

$$\mathbf{T}_k = \begin{bmatrix} \mathbf{C}_k & \mathbf{v}_k & \mathbf{t}_k \\ \mathbf{0} & 1 & 0 \\ \mathbf{0} & 0 & 1 \end{bmatrix} \in \mathrm{SE}_2(3), \quad \mathbf{b}_k = \begin{bmatrix} \mathbf{b}_k^\omega \\ \mathbf{b}_k^a \end{bmatrix} \in \mathbb{R}^6, \tag{4.2}$$

where $\mathbf{C}_k \in \mathrm{SO}(3)$ is the body-to-world rotation, $\mathbf{v}_k$ and $\mathbf{t}_k$ are the velocity and position in the world frame, and $\mathbf{b}_k$ is the IMU biases. We define the error state using a right perturbation on $\mathrm{SE}_2(3)$. Specifically, the true state $\boldsymbol{\xi}_k$ is related to a nominal state $\bar{\boldsymbol{\xi}}_k$ by

$$\mathbf{T}_k = \bar{\mathbf{T}}_k \mathrm{Exp}(\delta\boldsymbol{\zeta}_k), \quad \mathbf{b}_k = \bar{\mathbf{b}}_k + \delta\mathbf{b}_k, \tag{4.3}$$

where the 15-dimensional error state is

$$\delta\boldsymbol{\xi}_k = \begin{bmatrix} \delta\boldsymbol{\zeta}_k^\top & \delta\mathbf{b}_k^\top \end{bmatrix}^\top, \quad \delta\boldsymbol{\zeta}_k = \begin{bmatrix} \delta\boldsymbol{\theta}_k^\top & \delta\mathbf{v}_k^\top & \delta\mathbf{t}_k^\top \end{bmatrix}^\top. \tag{4.4}$$

Here, $\mathrm{Exp}(\cdot) = \exp((\cdot)^\wedge)$, where $(\cdot)^\wedge$ denotes the mapping from $\mathbb{R}^9$ to the Lie algebra $\mathfrak{se}_2(3)$ [33]. The nominal state is propagated using a deterministic process model:

$$\bar{\mathbf{T}}_k = \bar{\mathbf{T}}_{k-1} \mathrm{Exp}(\boldsymbol{\nu}_k \Delta t_k), \quad \bar{\mathbf{b}}_k = \bar{\mathbf{b}}_{k-1}. \tag{4.5}$$

The EKF assumes zero-mean Gaussian process and measurement noises: $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ and $\boldsymbol{\eta}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{\gamma,k})$. Process noise is modeled through the linearized error-state dynamics,

$$\delta\boldsymbol{\xi}_k = \mathbf{F}_{k-1}\, \delta\boldsymbol{\xi}_{k-1} + \mathbf{G}_k\, \boldsymbol{\epsilon}_k. \tag{4.6}$$

IMU noise enters through $\boldsymbol{\epsilon}_k$, and the biases are modeled as a random walk.

The EKF maintains a Gaussian belief over the error state with covariance $\boldsymbol{\Sigma}_k$. During the prediction step, the nominal state and covariance are propagated, where the covariance prediction is given by

$$\check{\boldsymbol{\Sigma}}_k = \mathbf{F}_{k-1}\, \hat{\boldsymbol{\Sigma}}_{k-1}\, \mathbf{F}_{k-1}^\top + \mathbf{G}_k\, \mathbf{Q}_k\, \mathbf{G}_k^\top, \tag{4.7}$$

yielding the prior $\{\check{\boldsymbol{\xi}}_k, \check{\boldsymbol{\Sigma}}_k\}$. Here, $\check{(\cdot)}$ and $\hat{(\cdot)}$ denote prior and posterior quantities, respectively.

For the measurement update, we receive an exteroceptive measurement $\boldsymbol{\gamma}_k$, related to the state by the nonlinear measurement model in (3.1b). The model, $\mathbf{g}(\cdot)$, is typically derived from sensor geometry, calibration parameters, and prior knowledge of the environment. Using the predicted state, $\check{\boldsymbol{\xi}}_k$, the EKF computes the innovation $\boldsymbol{r}_k = \boldsymbol{\gamma}_k - \mathbf{g}(\check{\boldsymbol{\xi}}_k, \mathbf{0})$, and linearizes the measurement model with respect to the error state to obtain the Jacobian, $\mathbf{H}_k = \left.\frac{\partial\delta\boldsymbol{\gamma}}{\partial\delta\boldsymbol{\xi}}\right|_{\delta\boldsymbol{\xi}=\mathbf{0}}$. See Appendix A.1

for details on the Jacobian derivation for $\mathrm{SE}_2(3)$. Then, the Kalman gain and the correction are, respectively,

$$\mathbf{K}_k = \check{\boldsymbol{\Sigma}}_k \mathbf{H}_k^\top (\mathbf{H}_k \check{\boldsymbol{\Sigma}}_k \mathbf{H}_k^\top + \mathbf{R}_{\gamma,k})^{-1}, \quad \delta\hat{\boldsymbol{\xi}}_k = \mathbf{K}_k \boldsymbol{r}_k. \tag{4.8}$$

Writing $\delta\hat{\boldsymbol{\xi}}_k = \begin{bmatrix} \delta\hat{\boldsymbol{\zeta}}_k^\top & \delta\hat{\mathbf{b}}_k^\top \end{bmatrix}^\top$, we update via

$$\hat{\mathbf{T}}_k = \check{\mathbf{T}}_k \mathrm{Exp}(\delta\hat{\boldsymbol{\zeta}}_k), \quad \hat{\mathbf{b}}_k = \check{\mathbf{b}}_k + \delta\hat{\mathbf{b}}_k, \tag{4.9}$$

followed by the standard EKF covariance update to obtain $\hat{\boldsymbol{\Sigma}}_k$.

Having established the standard EKF as our baseline, we next introduce the KILO-EKF, which adopts a Koopman-inspired approach for learning measurement models while retaining the classical prediction step.

4.5 The General KILO-EKF

In this section, we present KILO-EKF's general framework for mapping nonlinear measurement functions into a linear representation in a lifted feature space. The resulting formulation is agnostic to the underlying state space and filtering architecture. We specialize this approach to $\mathrm{SE}_2(3)$ in the next section by defining appropriate lifting functions for the states.

By lifting measurement model (4.1b), system (4.1) can be represented as

$$\boldsymbol{\xi}_k = \mathbf{f}(\boldsymbol{\xi}_{k-1}, \boldsymbol{\nu}_k, \boldsymbol{\epsilon}_k), \tag{4.10a}$$

$$\mathbf{y}_k = \mathbf{D}\mathbf{x}_k + \mathbf{n}_k, \tag{4.10b}$$

where for some nonlinear lifting functions $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$ and $\mathbf{p}_{\boldsymbol{\gamma}}(\cdot)$, the lifted quantities are $\mathbf{x}_k, \mathbf{n}_k \in \mathcal{X}$ and $\mathbf{y}_k \in \mathcal{Y}$, where

$$\mathbf{x}_k = \mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k), \quad \mathbf{y}_k = \mathbf{p}_{\boldsymbol{\gamma}}(\boldsymbol{\gamma}_k), \quad \mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}). \tag{4.11}$$

Throughout this chapter and the rest of the thesis, we use Greek letters to denote quantities in the original, unlifted space and Roman letters to denote quantities in the lifted space whenever possible. For example, $\boldsymbol{\xi}$ denotes the original state while $\mathbf{x}$ denotes its lifted representation.

Through this representation, we treated the lifted measurement model as linear Gaussian. While this is approximate, the aggregation of multiple error sources in high-dimensional feature spaces can often be reasonably approximated as Gaussian under the Central Limit Theorem [82]. Moreover, a linear-Gaussian formulation yields a tractable measurement update and is consistent with the assumptions underlying the EKF. This modeling choice also serves as the basis for the batch estimation frameworks developed in later chapters.

In general, lifting the measurements through $\mathbf{y}_k = \mathbf{p}_{\boldsymbol{\gamma}}(\boldsymbol{\gamma}_k)$ is optional, as sufficiently expressive state liftings can capture nonlinear measurement relationships. However, appropriate measurement liftings can simplify the learned model and improve numerical conditioning in practice. See Section 4.7.3 for an example. For now, we turn to learning the lifted model parameters from data.

4.5.1 Learning the Lifted Measurement Model

Our objective is to learn $\mathbf{D}$ and $\mathbf{R}$ in (4.10) and (4.11) from groundtruth data. Suppose we have a dataset consisting of P states and measurements: $\{\boldsymbol{\xi}_i, \boldsymbol{\gamma}_i\}_{i=1}^P$. We write the original data in block-matrix form:

$$\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\xi}_1 & \cdots & \boldsymbol{\xi}_P \end{bmatrix}, \quad \boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{\gamma}_1 & \cdots & \boldsymbol{\gamma}_P \end{bmatrix}. \tag{4.12}$$

The data translates to $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^P$ in the lifted space such that we have $\mathbf{y}_i = \mathbf{D}\mathbf{x}_i + \mathbf{n}_i$ for some unknown noise, $\mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. Then, we can write the lifted matrix form for the measurement model as $\mathbf{Y} = \mathbf{D}\mathbf{X} + \mathbf{N}$, where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_P \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & \cdots & \mathbf{y}_P \end{bmatrix}, \quad \mathbf{N} = \begin{bmatrix} \mathbf{n}_1 & \cdots & \mathbf{n}_P \end{bmatrix}. \tag{4.13}$$

We now formulate a loss function for learning the system matrices from data. Rather than EDMD-based formulations involving spectral decompositions [66], we adopt a Tikhonov-regularized approach that admits a closed-form solution. Specifically, we cast the learning problem as MAP estimation with conjugate priors[1] on $\mathbf{D}$ and $\mathbf{R}$. The resulting optimization problem is

$$\{\mathbf{D}^\star, \mathbf{R}^\star\} = \underset{\{\mathbf{D},\mathbf{R}\}}{\operatorname{argmin}} V(\mathbf{D}, \mathbf{R}), \tag{4.14}$$

where the loss function is $V = V_1 + V_2$ with

$$V_1 = \frac{1}{2} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_{\mathbf{R}^{-1}}^2 + \frac{1}{2} P \ln |\mathbf{R}|, \tag{4.15a}$$

$$V_2 = \frac{1}{2} P\tau_D \|\mathbf{D}\|_{\mathbf{R}^{-1}}^2 + \frac{1}{2} P\tau_R \operatorname{tr}(\mathbf{R}^{-1}). \tag{4.15b}$$

The norm is a weighted Frobenius matrix norm:

$$\|\mathbf{X}\|_{\mathbf{W}} = \sqrt{\operatorname{tr}(\mathbf{X}^T\mathbf{W}\mathbf{X})}. \tag{4.16}$$

Here, V_1 corresponds to the negative log-likelihood of the data under Gaussian noise, ignoring the normalizing constant. V_2 imposes priors on the model parameters: a Tikhonov prior on $\mathbf{D}$ to encourage a lower-complexity measurement model, and an (isotropic) inverse-Wishart (IW) prior on $\mathbf{R}$. IW priors regularize covariance estimation, helping avoid overfitting and degenerate estimates when training data are limited [83]. We tune the regularizing hyperparameters, τ_D and τ_R, from data using cross-validation.

We find the critical points by setting $\frac{\partial V}{\partial \mathbf{D}}$ and $\frac{\partial V}{\partial \mathbf{R}^{-1}}$ to zero, yielding

$$\mathbf{D} = (\mathbf{Y}\mathbf{X}^T)(\mathbf{X}\mathbf{X}^T + P\tau_D\mathbf{1})^{-1}, \tag{4.17a}$$

$$\mathbf{R} = \frac{1}{P}(\mathbf{Y} - \mathbf{D}\mathbf{X})(\mathbf{Y} - \mathbf{D}\mathbf{X})^T + \tau_D\mathbf{D}\mathbf{D}^T + \tau_R\mathbf{1}, \tag{4.17b}$$

where $\mathbf{1}$ represents the identity operator for the appropriate domains. This procedure is one shot

[1] A conjugate prior is a prior chosen such that, after combining it with the likelihood, the posterior remains in the same distribution family. Here, the priors on $\mathbf{D}$ and $\mathbf{R}$ preserve a closed-form, regularized regression-style MAP solution for the learned system matrices.

and linear in the amount of training data, P, for both computation and storage.

4.5.2 Inference

After learning the lifted measurement model parameters $\mathbf{D}$ and $\mathbf{R}$, we use them for state estimation within an EKF framework. In this setting, inference proceeds by computing the measurement Jacobian with respect to the original state. Using the lifted measurement model, this Jacobian can be written using the chain rule as

$$\mathbf{H}_k = \left.\frac{\partial \delta \mathbf{y}}{\partial \delta \boldsymbol{\xi}}\right|_{\delta\boldsymbol{\xi}=\mathbf{0}} = \mathbf{D} \left.\frac{\partial \delta \mathbf{p}_{\boldsymbol{\xi}}}{\partial \delta \boldsymbol{\xi}}\right|_{\delta\boldsymbol{\xi}=\mathbf{0}}. \tag{4.18}$$

This expression is straightforward to compute, as the lifting function $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$ is chosen a priori and is differentiable by construction. With this Jacobian, the EKF measurement update follows the standard formulation using the learned covariance $\mathbf{R}$ and leaving the prediction step unchanged[2]. With multiple sensors, we learn a separate lifted measurement model for each sensor, then apply the corresponding measurement updates sequentially, as in standard multi-sensor EKFs.

A convenient and widely used choice for the lifting function is to include the original state itself. Specifically, the state, $\boldsymbol{\xi}_k$, is concatenated with nonlinear features, $\tilde{\mathbf{p}}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k)$, as

$$\mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k) = \begin{bmatrix} \boldsymbol{\xi}_k^\top & \tilde{\mathbf{p}}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k)^\top \end{bmatrix}^\top. \tag{4.19}$$

This structure introduces only a modest increase in dimension, while enabling direct recovery of the original state.

In KILO-EKF, the goal is to directly estimate the original state $\boldsymbol{\xi}_k$. As we will see in later chapters, this inference procedure differs from lifted batch formulations, where the optimization variable is the full lifted state $\mathbf{x}_k$. In those settings, performance can be improved by enforcing consistency between the lifted and original representations through explicit constraints, discussed in detail in Section 6.6.5. In KILO-EKF, however, this consistency is automatically satisfied since the lifted representation is constructed directly from $\boldsymbol{\xi}_k$ only for the measurement. This avoids constrained optimization at inference time while preserving the standard EKF structure required for real-time operation.

4.6 KILO-EKF in $\mathrm{SE}_2(3)$

Having established the general framework of KILO-EKF, we now specialize to $\mathrm{SE}_2(3)$. We first define suitable lifting functions for the state, followed by the computation of the measurement Jacobian required for the EKF update.

4.6.1 Lifting Functions

In the following, we make explicit the construction of the lifting function $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$. From Section 4.4.1, our state is $\boldsymbol{\xi}_k = \{\mathbf{T}_k, \mathbf{b}_k\}$, as defined in (4.2). To apply Euclidean feature maps, we introduce $\mathbf{s}_k$,

[2]The proposed measurement-learning component is independent of the specific filtering formulation and can be integrated within alternative variants, such as the invariant EKF.

a vector representation of our EKF state via a mapping $\mathbf{q}(\cdot) : \mathrm{SE}_2(3) \times \mathbb{R}^6 \to \mathbb{R}^{15}$, where

$$\mathbf{s}_k = \mathbf{q}(\boldsymbol{\xi}_k) = \begin{bmatrix} \mathrm{vec}(\mathbf{C}_k)^\top & \mathbf{t}_k^\top & \mathbf{v}_k^\top & \mathbf{b}_{a,k}^\top & \mathbf{b}_{\omega,k}^\top \end{bmatrix}^\top, \tag{4.20}$$

where $\mathrm{vec}(\cdot)$ denotes column-wise vectorization. We then define the lifting function $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$ as a function of $\mathbf{s}_k$.

For lifting functions, we use random Fourier features (RFFs), specifically squared-exponential random Fourier features (SERFFs) [84]. These lifting functions map vector-space inputs to a finite-dimensional set of sinusoidal features. Given an input $\mathbf{s}$, the SERFF mapping is defined as

$$\mathbf{z}(\mathbf{s}) = \begin{bmatrix} \frac{1}{\sqrt{R}}\mathbf{z}_{\boldsymbol{\varpi}_1}(\mathbf{s})^\top & \cdots & \frac{1}{\sqrt{R}}\mathbf{z}_{\boldsymbol{\varpi}_R}(\mathbf{s})^\top \end{bmatrix}^\top, \quad \mathbf{z}_{\boldsymbol{\varpi}_i}(\mathbf{s}) = \begin{bmatrix} \sqrt{2}\cos(\boldsymbol{\varpi}_i^\top \mathbf{s}) & \sqrt{2}\sin(\boldsymbol{\varpi}_i^\top \mathbf{s}) \end{bmatrix}^\top, \tag{4.21}$$

where the frequency vectors are drawn independently as $\boldsymbol{\varpi}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{W}/k_\ell)$. Here, $\mathbf{W}$ is a typically diagonal weighting matrix that controls the relative importance of each input dimension, and k_ℓ is a length-scale parameter. Both hyperparameters can be selected a priori or tuned from data.

SERFFs are commonly used as lifting functions in Koopman-based methods due to their strong empirical performance [63]. In this chapter, their effectiveness is reflected in the performance of the learned models. In the next chapter, we develop a broader theoretical perspective on RFFs, including their connection to approximations of rich function classes. This perspective will show that selecting the number of RFFs provides a principled trade-off between approximation accuracy and computational cost. As we will see, SERFFs, and RFFs more generally, play important roles in Koopman-inspired batch estimation methods as well.

The proposed lifting framework can be applied directly without modification by defining $\mathbf{p}_{\boldsymbol{\xi}}(\mathbf{s}) = \mathbf{z}(\mathbf{s})$ from (4.21). In many practical scenarios, however, some prior knowledge of the system is available. In such cases, it is beneficial to incorporate this knowledge by constructing handcrafted features inspired by known measurement models. Although SERFFs alone provide a fully data-driven lifting, incorporating known structure can achieve comparable accuracy with substantially fewer random features and hence a lower-dimensional learned model. See Section 4.7 for an example. It is also often advantageous to restrict the lifting to a subset of the original state variables. For example, if the measurement is known to depend only on the pose, we modify (4.20) to retain only the corresponding components and discard the remaining state variables:

$$\mathbf{s}'_k = \begin{bmatrix} \mathrm{vec}(\mathbf{C}_k)^\top & \mathbf{t}_k^\top \end{bmatrix}^\top. \tag{4.22}$$

This reduces the dimensionality of the input to the lifting function and, consequently, the number of lifted features required. The resulting lifting function is then defined as

$$\mathbf{p}_{\boldsymbol{\xi}}(\mathbf{s}'_k) = \begin{bmatrix} \mathbf{s}'^\top_k & \mathbf{h}(\mathbf{s}'_k)^\top & \mathbf{z}(\mathbf{s}'_k)^\top \end{bmatrix}^\top, \tag{4.23}$$

where $\mathbf{h}(\cdot)$ is a handcrafted feature function. This design enables flexible incorporation of domain knowledge while preserving the generality of the KILO-EKF framework.

Algorithm 1: Koopman-Inspired Learned Observations EKF (KILO-EKF)

Training:

Input: Training data $\{\boldsymbol{\xi}_i, \boldsymbol{\gamma}_i\}_{i=1}^{P}$; lifting functions $\{\mathbf{p}_{\boldsymbol{\xi}}(\cdot), \mathbf{p}_{\boldsymbol{\gamma}}(\cdot)\}$ (e.g., SERFFs (4.21)); algorithm hyperparameters $\{\lambda_D, \lambda_R\}$.

1. Stack training data $\{\boldsymbol{\xi}_i, \boldsymbol{\gamma}_i\}_{i=1}^{P}$ into their block-matrix form, $\{\boldsymbol{\Xi}, \boldsymbol{\Gamma}\}$, with (4.12).
2. Embed $\{\boldsymbol{\Xi}, \boldsymbol{\Gamma}\}$ using the lifting functions $\{\mathbf{p}_{\boldsymbol{\xi}}(\cdot), \mathbf{p}_{\boldsymbol{\gamma}}(\cdot)\}$, yielding $\{\mathbf{X}, \mathbf{Y}\}$ defined in (4.13).
3. Solve for model matrices $\{\mathbf{C}, \mathbf{R}\}$ using (4.17).

Output: Measurement model matrices: $\{\mathbf{C}, \mathbf{R}\}$.

Inference:

Input: Previous state belief mean and covariance $\{\hat{\boldsymbol{\xi}}_{k-1}, \hat{\boldsymbol{\Sigma}}_{k-1}\}$; input $\boldsymbol{\nu}_k$; received measurement $\boldsymbol{\gamma}_k$.

1. *State prediction:* Apply the standard EKF prediction step with input $\boldsymbol{\nu}_k$ to obtain the predicted belief $\{\check{\boldsymbol{\xi}}_k, \check{\boldsymbol{\Sigma}}_k\}$. For $\mathrm{SE}_2(3)$, this procedure is summarized in Section 4.4.1.
2. *Lifted measurement prediction:* Construct the lifted state $\check{\mathbf{x}}_k = \mathbf{p}_{\boldsymbol{\xi}}(\check{\boldsymbol{\xi}}_k)$. Predict the lifted measurement as $\check{\mathbf{y}}_k = \mathbf{C}\check{\mathbf{x}}_k$.
3. *Linearization:* Embed the received measurement as $\mathbf{y}_k = \mathbf{p}_{\boldsymbol{\gamma}}(\boldsymbol{\gamma}_k)$, compute the innovation $\mathbf{e}_k = \mathbf{y}_k - \check{\mathbf{y}}_k$, and compute the measurement Jacobian $\mathbf{H}_k$ using (4.18).
4. *Correction:* Apply the standard EKF update using $\mathbf{e}_k$ and $\mathbf{H}_k$ to obtain the updated belief $\{\hat{\boldsymbol{\xi}}_k, \hat{\boldsymbol{\Sigma}}_k\}$.

Output: Updated state belief mean and covariance: $\{\hat{\boldsymbol{\xi}}_k, \hat{\boldsymbol{\Sigma}}_k\}$.

4.6.2 Measurement Jacobian

Using the lifted measurement model, the EKF measurement Jacobian follows directly from the chain rule as in (4.18). The geometric aspects of the $\mathrm{SE}_2(3)$ linearization are identical to those of a conventional EKF and are summarized in Appendix A.1. For the SERFF lifting functions defined in (4.21), the required derivatives are available in closed form:

$$\frac{\partial \mathbf{z}_{\boldsymbol{\varpi}_i}}{\partial \mathbf{s}} = \begin{bmatrix} -\sqrt{2}\sin(\boldsymbol{\varpi}_i^\top \mathbf{s}) \\ \sqrt{2}\cos(\boldsymbol{\varpi}_i^\top \mathbf{s}) \end{bmatrix} \boldsymbol{\varpi}_i^\top. \tag{4.24}$$

Then, we would follow the standard EKF update. See Algorithm 1 for a summary of the training and testing procedures of KILO-EKF.

4.7 Experiments

4.7.1 Problem Setup

We validate the proposed KILO-EKF on a real-world 3D localization dataset named MILUV [81]. A quadrotor operates in a $3\,\mathrm{m} \times 3\,\mathrm{m} \times 2\,\mathrm{m}$ indoor environment, with an onboard IMU providing linear

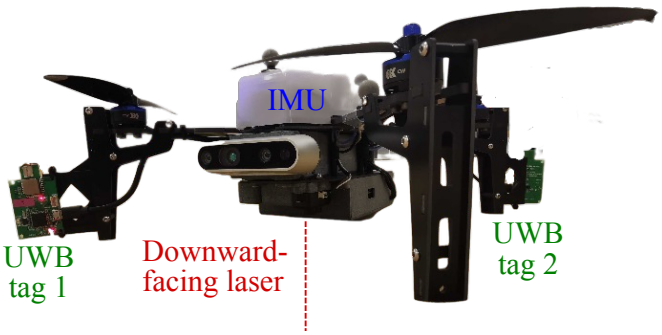


Figure 4.3: System overview of the quadrotor platform [81] used to evaluate KILO-EKF, which learns data-driven measurement models for UWB ranging and laser height sensing.

acceleration and angular velocity measurements. For exteroceptive sensing, the quadrotor carries two ultra-wideband (UWB) tags and receives range measurements from six fixed UWB anchors. In addition, a downward-facing laser provides measurements of the vertical height. See Fig. 4.3 for a visualization. The localization objective is to estimate the quadrotor's pose trajectory using IMU, UWB, and laser measurements. Groundtruth poses are obtained from a motion-capture system and are used for supervised learning and evaluation under a cross-validation protocol.

4.7.2 EKF Baselines

Our baseline estimators are standard $\mathrm{SE}_2(3)$ EKFs based on the open-source implementations released with the MILUV dataset [81]. For timestamp k, UWB tag i, and anchor j, the analytical UWB range measurement model is

$$\gamma_{i,j,k} = g_{i,j}(\boldsymbol{\xi}_k) = \|\mathbf{r}_{\ell,j} - \mathbf{t}_k - \mathbf{C}_k \mathbf{r}_{b,i}\| + \eta_{i,j,k}, \tag{4.25}$$

where $\mathbf{r}_{\ell,j}$ denotes the position of anchor j expressed in the world frame, $\mathbf{r}_{b,i}$ denotes the position of UWB tag i expressed in the vehicle frame, and $\eta_{i,j,k}$ is zero-mean measurement noise.

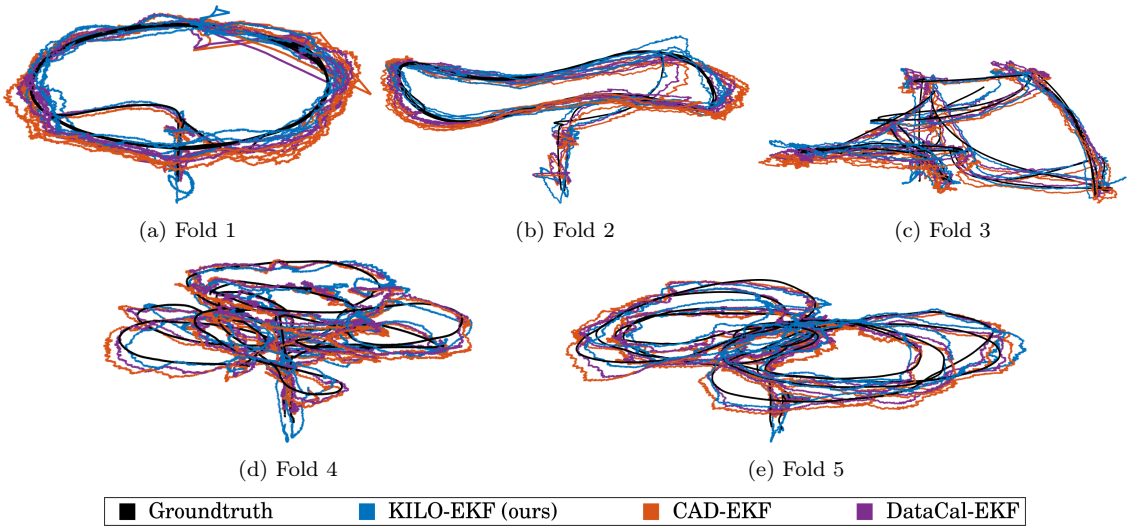
In addition to UWB range measurements, we have a downward-facing laser that measures the vertical height. The analytical laser measurement model is

$$\gamma_{L,k} = g_L(\boldsymbol{\xi}_k) = t_{z,k} + \eta_{L,k}, \tag{4.26}$$

where $t_{z,k}$ denotes the z-component of the position $\mathbf{t}_k$, and $\eta_{L,k}$ is zero-mean measurement noise.

To evaluate the proposed method across different calibration scenarios, we compare against three baseline EKF variants. These baselines differ in how the UWB calibration parameters, namely the anchor positions $\mathbf{r}_{\ell,j}$ and tag offsets $\mathbf{r}_{b,i}$ in the UWB measurement model (4.25), are obtained. Our baselines are designed to reflect common practical use cases, ranging from nominal computer-aided design (CAD)-based calibration to imperfect or data-driven calibration. The three baselines are:

- **CAD-EKF**: Anchor positions $\mathbf{r}_{\ell,j}$ and tag offsets $\mathbf{r}_{b,i}$ are taken directly from CAD measurements.
- **MisCAD-EKF**: Tag offsets are taken from CAD measurements and perturbed by a 1° rotation to simulate mild calibration errors.



(a) Fold 1 (b) Fold 2 (c) Fold 3

(d) Fold 4 (e) Fold 5

Figure 4.4: Trajectory estimates for the five cross-validation folds, illustrating the diversity of path geometries used in evaluation. Qualitatively, although not always visually pronounced, the proposed KILO-EKF generally tracks the groundtruth more closely than the two baselines.

Fold	Time [s]	Measurements	Name in MILUV [81]
1	146.1	14342	*default_1_circular2D_0*
2	135.2	13568	*default_1_circular3D_0*
3	205.2	20700	*default_1_random3_0*
4	195.1	19512	*default_1_remoteControlLowPace_0*
5	227.0	22534	*default_1_remoteControlLowPace_0_v2*

Table 4.1: Dataset splits for cross-validation of KILO-EKF against model-based baselines.

- **DataCal-EKF**: Anchor positions and tag offsets are initialized using CAD measurements, then subsequently refined by solving a nonlinear least-squares calibration problem with Levenberg-Marquardt optimization on training data.

For fairness, the calibration parameters in DataCal-EKF are optimized using the same training data as KILO-EKF.

4.7.3 KILO-EKF Implementation

The proposed KILO-EKF shares the same process model, IMU preprocessing, and state propagation as the baseline EKFs described in Section 4.7.2. This ensures a controlled comparison, as all methods differ only in their treatment of the measurement model. The key distinction of the KILO-EKF lies in replacing the analytical measurement models with lifted linear-Gaussian representations learned from data.

Inspired by the structure of the UWB range model (4.25), we restrict the measurement dependence to the pose variables by defining a reduced state as in (4.22). Based on this reduced state, we construct a set of handcrafted features,

$$\mathbf{h}(\mathbf{s}'_k) = \begin{bmatrix} 1 & \mathrm{vec}(\mathbf{C}_k)^\top & \mathbf{C}_k^\top \mathbf{t}_k & \mathbf{t}_k^\top \mathbf{t}_k \end{bmatrix}^\top, \tag{4.27}$$

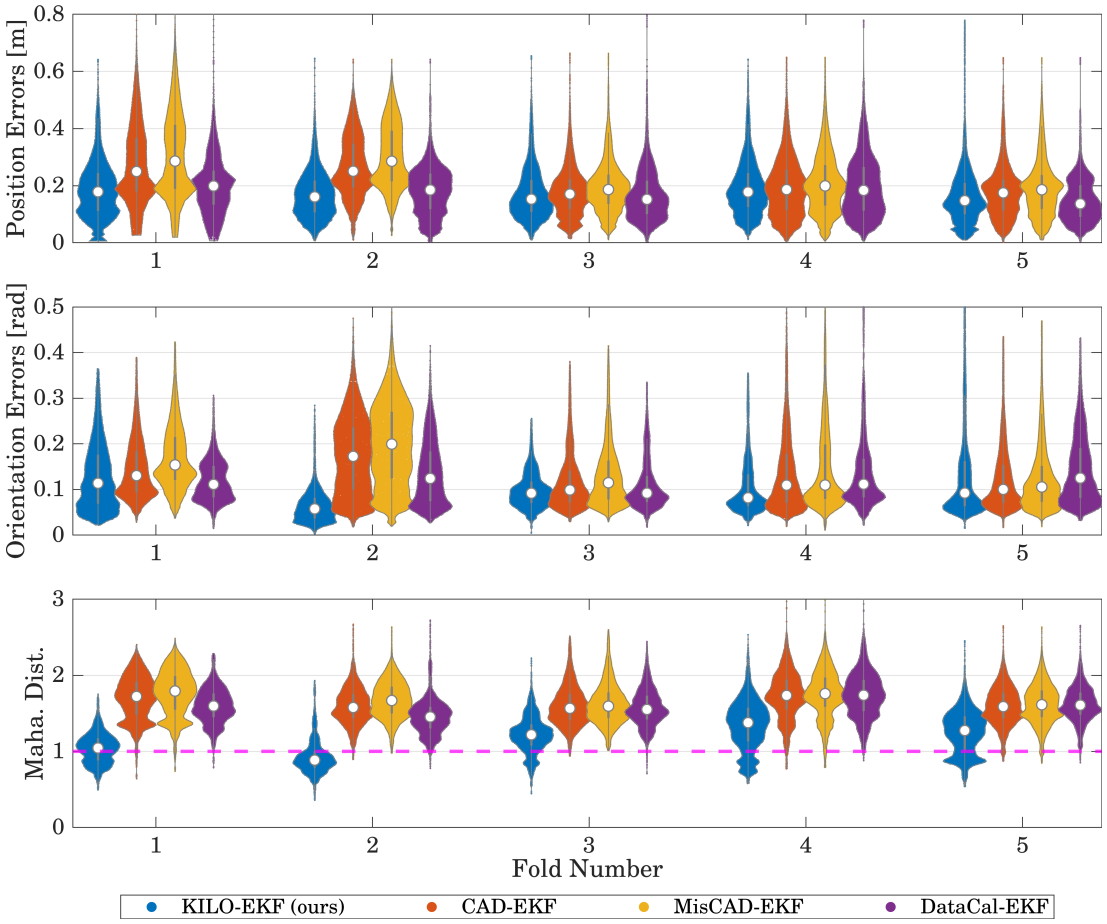


Figure 4.5: Violin plots of per-timestep position errors, orientation errors, and Mahalanobis distances across the five cross-validation folds. Each point corresponds to a single time step. KILO-EKF consistently achieves comparable or lower errors than the baseline methods across all folds, and exhibits Mahalanobis distances closer to 1, indicating more consistent covariance estimates.

which encode simple geometric relationships commonly used in range-based localization. These handcrafted features are augmented with SERFFs for modeling the residual nonlinearities, forming the full lifted state via (4.23).

In addition to lifting the state, we also lift measurements as defined in (4.11). Specifically, for UWB ranges, we use squared ranges as the lifted measurements: $y_{i,j,k} = p_\gamma(\gamma_{i,j,k}) = \gamma_{i,j,k}^2$. This avoids the square-root nonlinearity in (4.25), reducing the complexity of the learned model and the number of features required to capture the measurement relationship.

Because each UWB range measurement depends on a specific anchor position and tag offset, we learn separate lifted measurement models for each anchor-tag pair. This results in $6 \times 2 = 12$ UWB measurement models, along with an additional model for the laser height measurement.

4.7.4 Evaluation Method

We evaluate the proposed KILO-EKF against the three baseline EKF variants. Performance is assessed in terms of position and orientation accuracy and consistency. For quantitative comparison, we report the per-timestep root-mean-square errors (RMSEs) and normalized Mahalanobis

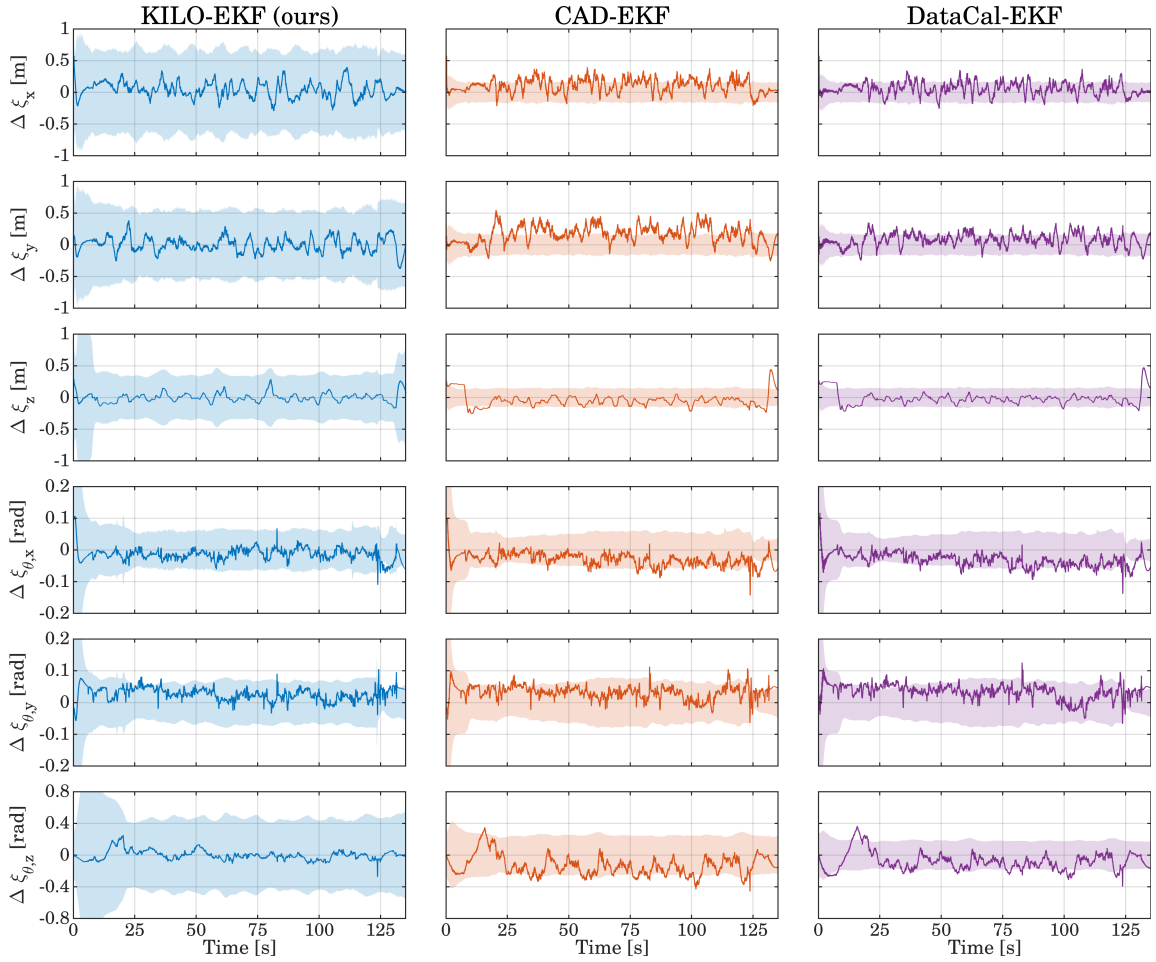


Figure 4.6: Per-axis position and orientation error time series for Fold 2. The shaded region indicates the 3σ covariance envelope. KILO-EKF exhibits lower errors and more consistent covariance estimates, while the baseline methods show larger errors that frequently exceed the predicted covariance bounds.

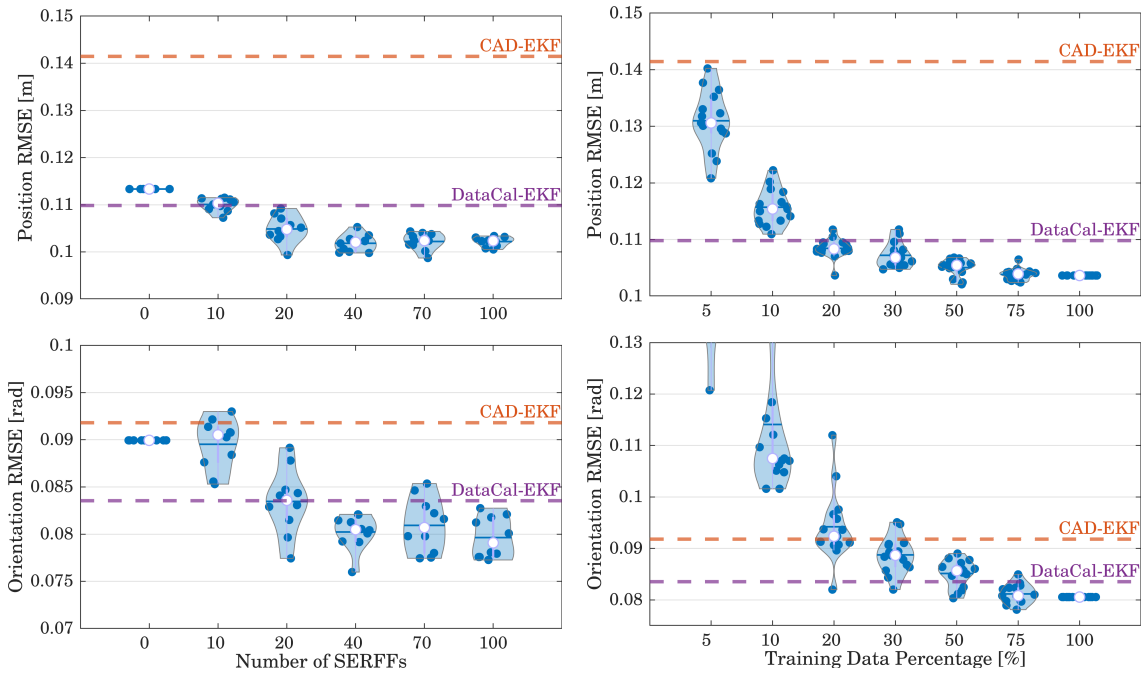
distances [85]. This distance is equivalent to the square root of the normalized estimation error squared (NEES) and is computed with

$$d_k = \sqrt{(\delta\boldsymbol{\xi}_k)^T \hat{\boldsymbol{\Sigma}}_k^{-1} (\delta\boldsymbol{\xi}_k)/N}, \tag{4.28}$$

where $\delta\boldsymbol{\xi}_k$ is the estimation error with respect to the groundtruth, $\hat{\boldsymbol{\Sigma}}_k$ is the estimated covariance, and N is the state dimension. Lower RMSE indicates higher accuracy in the mean estimates, while Mahalanobis distances close to 1 reflect statistically consistent covariance estimates.

Evaluation is performed using a five-fold cross-validation protocol, where one sequence is held out for testing and the remaining four are used for training. The evaluation folds are listed in Table 4.1, and visualizations of the trajectories are shown in Fig. 4.4. To increase the diversity of training data, we additionally include seven sequences exclusively for training[3]. These sequences are excluded from testing because their high speeds and frequent directional changes place them outside the evaluation regime considered in this work; both KILO-EKF and the baseline EKFs incur

[3]Training-only datasets from MILUV [81]: *cir_1_circular2D_0*, *cir_3_random3_0*, *default_3_random_0*, *default_3_random_0b*, *default_3_movingTriangle_0b*, *default_3_random2_0*, *default_3_zigzag_0*.



(a) Ablation study on SERFFs

(b) Ablation study on training data

Figure 4.7: Two ablation studies. Left: KILO-EKF RMSE versus the number of SERFFs, using the full training dataset. Right: KILO-EKF RMSE versus the percentage of training samples, with the number of SERFFs fixed at 100. Each dot denotes the average RMSE across all folds. In both studies, increasing the number of SERFFs or the amount of training data consistently reduces error. KILO-EKF outperforms both baseline methods when using 100 SERFFs or the full dataset.

very large estimation errors on these trajectories. Robustness to such challenging motions and other adverse sensing conditions, such as non-line-of-sight (NLOS) measurements, is left for future work.

4.7.5 Results

We compare the proposed KILO-EKF against the three baseline EKF variants using the evaluation metrics described above across the five folds. The distribution of errors and Mahalanobis distances is summarized in Fig. 4.5, and the error plots for Fold 2 are shown in Fig. 4.6. Overall, CAD-EKF and MisCAD-EKF exhibit larger errors and poorer consistency than the two data-calibrated approaches, highlighting the sensitivity of analytical measurement models to calibration accuracy. In particular, MisCAD-EKF demonstrates that even mild perturbations in the assumed tag or anchor geometry lead to noticeable performance degradation.

Comparing the two data-driven approaches, KILO-EKF achieves performance that is consistently comparable to, and in many cases better than, DataCal-EKF across all metrics. While both methods reduce the impact of miscalibration, DataCal-EKF remains constrained by the parametric structure of the analytical measurement model. In contrast, KILO-EKF learns a functional representation of the measurement model, allowing it to capture additional nonlinearities and systematic effects beyond geometric calibration alone. Depending on the fold and metric, this manifests as lower position error, improved orientation accuracy, or more consistent covariance estimates, while remaining competitive in the remaining dimensions.

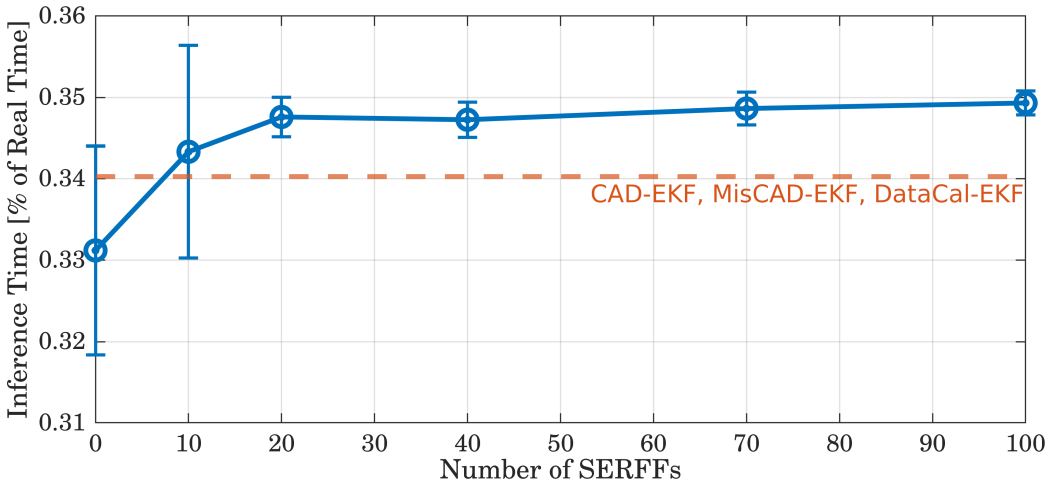


Figure 4.8: KILO-EKF inference time versus the number of SERFFs. Despite operating in a higher-dimensional space, KILO-EKF maintains real-time performance. Its inference time remains comparable to the three baselines, which all have similar inference times, even at 100 SERFFs.

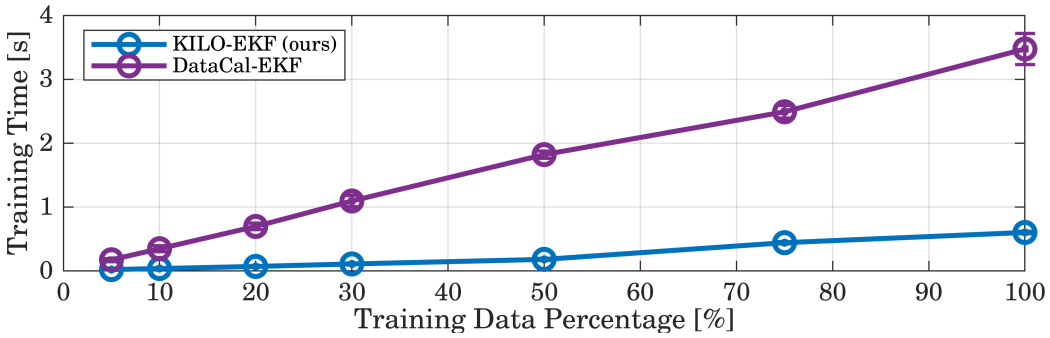


Figure 4.9: Training time versus the percentage of training data. KILO-EKF, which trains in a single pass, scales more favorably than DataCal-EKF, which relies on nonlinear least-squares calibration. As a result, KILO-EKF fits the full dataset (about 200,000 measurements from roughly 30 minutes of data) in under one second.

To further examine these trends, we conduct two ablation studies. First, Fig. 4.7a shows the impact of varying the number of SERFFs on the average RMSEs of trajectories. When no random features are used, the model relies solely on handcrafted structure and exhibits limited accuracy. As the number of SERFFs increases, the error consistently decreases, and at 100 features, KILO-EKF outperforms both baseline methods. This verifies that capturing additional nonlinearities beyond what is encoded by the analytical model and handcrafted features alone is essential for this problem. Second, Fig. 4.7b evaluates the effect of training data size. Increasing the amount of training data steadily improves performance, and with the full 30-minute dataset, KILO-EKF surpasses both baseline methods, highlighting the importance of data diversity for learning accurate models. In both cases, improvements exhibit diminishing returns, suggesting a graceful trade-off between model complexity, data volume, and accuracy.

Finally, we evaluate computational efficiency. The experiments are CPU-based implementations run on an Intel Core i7-9750H Processor. As shown in Fig. 4.8, KILO-EKF maintains real-time inference performance even with a large number of SERFFs, with runtimes comparable to the baseline EKFs. More notably, Fig. 4.9 demonstrates that KILO-EKF scales favorably in training, fitting the

full dataset in under one second due to its closed-form, least-squares-like formulation. This contrasts with DataCal-EKF, which relies on iterative nonlinear optimization and incurs significantly higher computational cost. Although DataCal-EKF converges reliably in our experiments, such nonlinear calibration procedures can also be susceptible to local minima in more complex settings. In contrast, KILO-EKF requires no initialization points and admits a one-shot solution for training.

4.8 Conclusion

We have presented KILO-EKF, a Koopman-inspired extension of the EKF in which the measurement model is learned from data using a linear representation in a lifted feature space. By retaining the standard process model and modifying only the measurement update, KILO-EKF naturally supports Lie-group state spaces such as $\mathrm{SE}_2(3)$. At the same time, it preserves the structure and efficiency of classical filtering while enabling flexible, data-driven modeling of sensor behavior. We have validated our approach on a real-world quadrotor dataset against multiple EKF baselines. The results demonstrate the effectiveness of learned measurement models and the efficiency of the lifted linear formulation, while maintaining real-time performance.

Possible extensions of this formulation include structured feature selection methods such as sparse identification of nonlinear dynamics (SINDy) [68, 77] and online adaptation of the learned measurement model. As well, the present KILO-EKF formulation targets low-dimensional ranging observations; extension to visual or dense LiDAR via feature-level representations is left for future work. Within the context of this thesis, however, the more immediate next step is to broaden the role of lifting in estimation itself. This chapter establishes the first central idea of the thesis: data-driven Koopman-inspired lifting can be integrated into classical Gaussian inference without sacrificing geometric structure. The next chapter builds on this idea by extending lifting beyond the measurement model and beyond recursive filtering, enabling data-driven batch state estimation where both process and measurement models are learned.

Chapter 5

Koopman-Inspired Smoothing

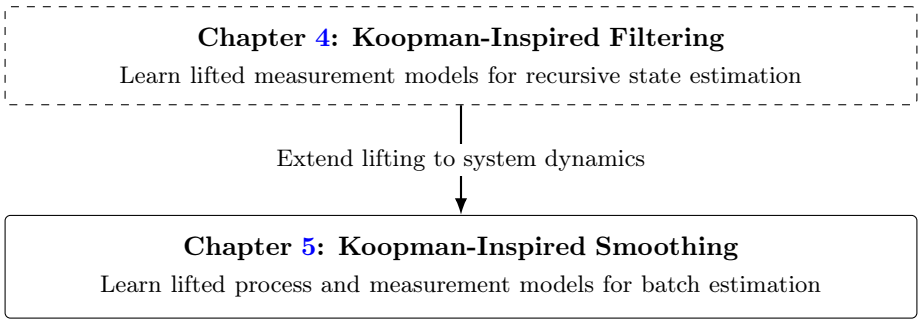


Figure 5.1: Conceptual progression from Chapter 4 to Chapter 5.

5.1 Summary of Contributions

This chapter extends the Koopman-inspired lifting framework of Chapter 4 from learning only measurement models to learning *both process and measurement models*, allowing nonlinear state estimation problems to be formulated using structured linear-Gaussian inference. It also provides a broader perspective on lifting, connecting Koopman-inspired representations to kernel-based embeddings and their finite-dimensional approximations. This chapter

- introduces a Koopman-inspired framework for learning lifted process and measurement models directly from data,
- shows how the resulting learned models can be incorporated into a lifted structured state estimation formulation with tractable mean and covariance recovery,
- empirically demonstrates the effectiveness of the approach on systems with complex dynamics and sensing relationships, and
- connects Koopman-inspired lifting to kernel embeddings, providing a theoretical basis for using RFF-based lifted representations to approximate rich function classes.

Within the broader context of this thesis, this chapter generalizes the use of data-driven lifting from sensing alone to full system-level estimation, providing the foundation for the localization and SLAM methods developed in the following chapter.

Associated publication: Z. C. Guo, V. Korotkine, J. R. Forbes, and T. D. Barfoot, "Koopman linearization for data-driven batch state estimation of control-affine systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 866–873, 2022. DOI: 10.1109/LRA.2021.3133587.

5.2 Motivation

In Chapter 4, we showed how data-driven lifting can be used to learn expressive measurement models while preserving compatibility with classical recursive estimation. However, in many robotic systems, estimation accuracy depends just as strongly on the process model, which is often difficult to derive accurately from first principles. Real-world dynamics may be nonlinear, partially unknown, or difficult to calibrate, and mismatch in the process model can significantly degrade estimation performance [22].

This challenge is particularly important in batch state estimation, where the entire trajectory is estimated jointly, with states coupled through the process model and all measurements incorporated simultaneously [1]. For linear-Gaussian systems, both system identification and inference can be performed efficiently within a common model class [1]. For nonlinear systems, however, both model learning and estimation typically require problem-specific parameterizations and iterative inference procedures [86], limiting scalability and generality.

Data-driven approaches offer a potential alternative. Neural network-based methods, such as differentiable filters [2], are highly expressive but often sacrifice structure, making it difficult to retain properties such as consistent uncertainty or efficient inference. Another class of approaches represents nonlinear systems in higher-dimensional spaces, either through Koopman-inspired lifting or kernel-based embeddings in a reproducing kernel Hilbert space (RKHS). However, existing Koopman-based approaches for modeling system dynamics typically rely on problem-specific basis functions and often assume linear lifted dynamics, whereas many robotic systems are control-affine and admit bilinear lifted representations [13]. Moreover, prior to this work, there were no data-driven methods that cast nonlinear state estimation into a clean batch framework with tractable mean and covariance recovery, limiting their applicability in robotics. Kernel-based approaches, on the other hand, are capable of representing a broad class of nonlinear functions. However, they suffer from poor scalability and are therefore less commonly used in practice.

In this chapter, we extend the Koopman-inspired lifting framework to learn both process and measurement models from data. The resulting Koopman State Estimator (KoopSE) approximately recasts nonlinear state estimation as linear-Gaussian inference in a lifted space, preserving much of the structure and efficiency of classical estimation while substantially expanding modeling flexibility. In particular, this formulation

- is applicable to control-affine systems with little prior knowledge of the process and measurement models,
- formulates the problem as a high-dimensional batch linear state estimation framework, which admits solutions for state means and covariances, and

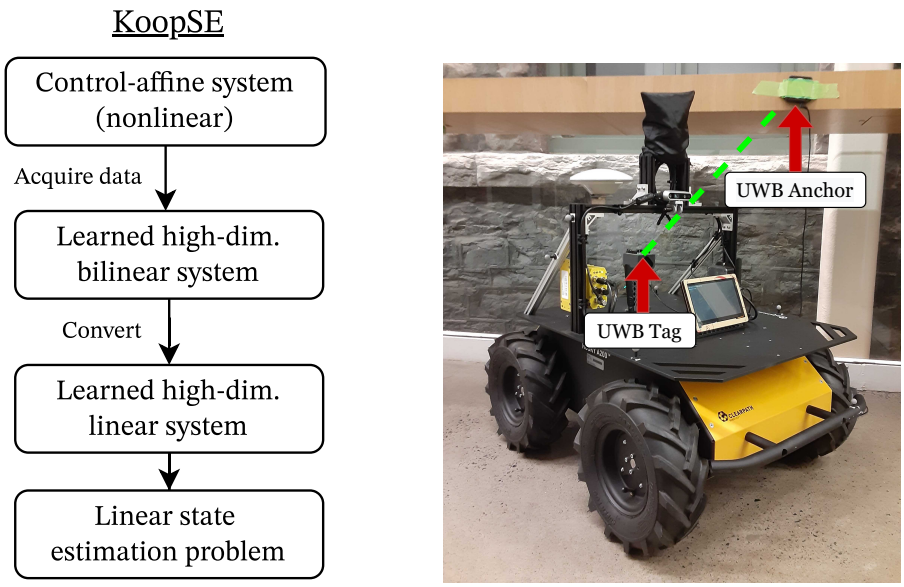


Figure 5.2: Left: KoopSE concept flowchart. KoopSE uses training data to learn a high-dimensional system, allowing inference to be done by solving a linear state estimation problem. Right: Experimental setup for validating KoopSE. A Husky robot drives around in an indoor environment while receiving range measurements from ultra-wideband (UWB) anchors and logging wheel odometry.

- has a training cost that scales linearly with the amount of data and an inference cost that is independent of the amount of training data.

During the derivation of KoopSE, we also establish a connection between Koopman-inspired lifting and kernel embeddings, providing a theoretical basis for the use of RFFs to approximate rich function classes. This connection yields a practical mechanism for controlling the trade-off between approximation accuracy and computational cost through the number of random features.

This chapter is structured as follows. After reviewing related work in Section 5.3, we summarize theories for kernel embeddings in Section 5.4. We derive KoopSE through Sections 5.5-5.7, then apply the RFF approximation in Section 5.8. We present experimental results in Sections 5.9 and conclude in Section 5.10.

5.3 Related Work

Methods involving kernels and RKHSs are becoming increasingly common in machine learning and robotics [46], enabling expressive nonparametric representations for nonlinear systems and probability distributions. Kernel mean embeddings allow probability distributions to be represented as elements of an RKHS, enabling probabilistic inference directly in lifted feature spaces [50]. These ideas led to the development of methods such as the kernel Bayes filter [51] and smoother [52]. Other kernelized approaches, including Gaussian process (GP) regression [47], have also been applied to data-driven state estimation [53]. Compared to kernel embedding methods, GP-based approaches are often more computationally efficient and naturally provide probabilistic uncertainty estimates [47]. However, many GP formulations assume additive Gaussian noise models, perform-

ing poorly compared to kernel embedding methods for systems with multimodal noise [56]. More broadly, kernel-based methods typically scale poorly with the number of training samples, often incurring cubic training complexity and requiring approximations or compression of the training set for practical inference [55, 87]. In addition, extending kernel embedding methods to systems with control inputs remains a challenge.

In contrast to kernel embedding methods, Koopman-inspired methods operate directly on finite-dimensional lifted representations [6, 7]. Finite-dimensional system matrices can often be learned efficiently using methods such as EDMD [66, 67], yielding inference procedures whose computational cost is independent of the amount of training data at test time. However, many existing approaches rely on observables tailored to specific systems or applications [59, 11]. One work has shown [13] that deterministic control-affine systems can be represented exactly as lifted bilinear systems under Koopman operators using generic feature families such as polynomial or Fourier observables. Although that work did not consider measurement models or probabilistic inference, this equivalence forms the basis of the KoopSE framework developed in this chapter.

RFFs [84] provide finite-dimensional approximations to kernel functions through randomized spectral representations. They have been effective in robotics applications including occupancy mapping [88] and dynamics learning [89]. Within the Koopman literature, however, RFFs are often used heuristically as lifting functions without explicitly leveraging their underlying connection to kernel embeddings [63]. This chapter develops that connection, combining the expressiveness of kernel-based representations with the computational efficiency of Koopman-inspired lifted inference.

5.4 Preliminaries

5.4.1 Reproducing Kernel Hilbert Space (RKHS) Embeddings

To provide a broader perspective on lifting, we review the fundamentals of kernel embeddings and RKHS. This viewpoint allows us to relate Koopman-inspired lifting to RKHS embeddings, which provide a general framework for representing complex probability distributions.

We begin with the general nonlinear system,

$$\boldsymbol{\xi}_k = \mathbf{f}(\boldsymbol{\xi}_{k-1}, \boldsymbol{\nu}_k, \boldsymbol{\omega}_k), \tag{5.1a}$$

$$\boldsymbol{\gamma}_k = \mathbf{g}(\boldsymbol{\xi}_k, \boldsymbol{\eta}_k), \tag{5.1b}$$

where $\boldsymbol{\xi}_k \in \mathbb{R}^{N_\xi}$ is the state, $\boldsymbol{\nu}_k \in \mathbb{R}^{N_\nu}$ the control input, $\boldsymbol{\omega}_k \in \mathbb{R}^{N_\omega}$ the process noise, $\boldsymbol{\gamma}_k \in \mathbb{R}^{N_\gamma}$ the measurement output, and $\boldsymbol{\eta}_k \in \mathbb{R}^{N_\eta}$ the measurement noise, all at timestep k. Analogous to Koopman lifting, we embed the states, inputs, and measurements into a higher-dimensional space,

$$\mathbf{x}_k = \mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k), \quad \mathbf{u}_k = \mathbf{p}_{\boldsymbol{\nu}}(\boldsymbol{\nu}_k), \quad \mathbf{y}_k = \mathbf{p}_{\boldsymbol{\gamma}}(\boldsymbol{\gamma}_k), \tag{5.2}$$

where $\mathbf{p}_{\boldsymbol{\xi}} : \mathbb{R}^{N_\xi} \to \mathcal{X}$, $\mathbf{p}_{\boldsymbol{\nu}} : \mathbb{R}^{N_\nu} \to \mathcal{U}$, and $\mathbf{p}_{\boldsymbol{\gamma}} : \mathbb{R}^{N_\gamma} \to \mathcal{Y}$ are (possibly infinite-dimensional) feature maps. In this section, we consider the case where $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{Y}$ are RKHSs [49, 90], with $\mathbf{p}_{\boldsymbol{\xi}}$, $\mathbf{p}_{\boldsymbol{\nu}}$, and $\mathbf{p}_{\boldsymbol{\gamma}}$ being feature maps associated with their respective kernels.

In contrast to RKHS embeddings, standard Koopman methods typically use explicit, finite-dimensional lifting functions selected based on empirical performance [8]. By using the RKHS

framework, however, we can embed entire distributions of random variables in the higher-dimensional space. The distribution is embedded by the *mean map* [91],

$$\boldsymbol{\mu}_k = \mathbb{E}[\mathbf{x}_k]. \tag{5.3}$$

So long as the kernel associated with $\mathcal{X}$ is characteristic [91], the mean map is injective, and thus uniquely represents probability distributions over $\boldsymbol{\xi}_k$. This is a key advantage of RKHS embeddings: they enable distributions, rather than just expectations, to be represented in feature space. While Koopman operators act linearly on observables, RKHS embeddings provide a complementary perspective in which distributions themselves can be embedded and manipulated using linear structure.

Given a finite set of samples of a random variable $\boldsymbol{\xi}$, the corresponding mean embedding can be approximated empirically as $\boldsymbol{\mu}_k \approx \sum_{i=0}^{M} m_{k,i}\mathbf{x}_i$, where $\mathbf{x}_i = \mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_i)$ and the weights $m_{k,i}$ depend on how the samples are obtained. This can be written compactly as

$$\boldsymbol{\mu}_k \approx \mathbf{X}\mathbf{m}_k, \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}_0 & \cdots & \mathbf{x}_M \end{bmatrix}, \quad \mathbf{m}_k = \begin{bmatrix} m_{k,0} & \cdots & m_{k,M} \end{bmatrix}^\top. \tag{5.4}$$

Thus, the mean embedding lies in the span of the embedded samples.

A key property of this representation is that expectations of functions can be computed directly from the weights. In particular, for any function $h(\cdot)$, we have

$$\mathbb{E}[h(\boldsymbol{\xi}_k)] \approx \sum_{i=0}^{M} m_{k,i}\, h(\boldsymbol{\xi}_i). \tag{5.5}$$

For estimation, this property allows moments of the original random variable to be recovered directly from computations in the feature space, without requiring an explicit inverse mapping from the RKHS back to the original space. To recover the first moment (mean), we set $h(\cdot)$ to the identity function:

$$\mathbb{E}[\boldsymbol{\xi}_k] \approx \sum_{i=0}^{M} m_{k,i}\boldsymbol{\xi}_i = \boldsymbol{\Xi}\,\mathbf{m}_k, \quad \boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\xi}_0 & \cdots & \boldsymbol{\xi}_M \end{bmatrix}. \tag{5.6}$$

To recover the second moment (covariance), we can define a *centered covariance map* [91] similarly to the mean map as

$$\boldsymbol{\Sigma}_{k\ell} = \mathbb{E}\left[(\mathbf{x}_k - \boldsymbol{\mu}_k)(\mathbf{x}_\ell - \boldsymbol{\mu}_\ell)^\top\right], \tag{5.7}$$

which can also represent any joint distribution over $(\boldsymbol{\xi}_k, \boldsymbol{\xi}_\ell)$ provided the kernel associated with $\mathcal{X}$ is characteristic. Given a finite set of samples from $(\boldsymbol{\xi}_k, \boldsymbol{\xi}_\ell)$, we can approximate the covariance map as $\boldsymbol{\Sigma}_{k\ell} \approx \mathbf{X}\mathbf{S}_{k\ell}\mathbf{X}^\top$, where $\mathbf{S}_{k\ell}$ is a weight matrix that depends on how the samples were drawn.

To summarize, although RKHS embeddings are generally infinite-dimensional, they provide a framework for representing and manipulating probability distributions in feature space, while allowing quantities of interest to be recovered without explicitly inverting the embedding. The main idea pursued in this chapter is to embed a control-affine system into such a feature space, where it admits a bilinear representation. In Sections 5.5 and 5.6, we will proceed under the assump-

tion that these embeddings can be accessed explicitly, even though $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$, $\mathbf{p}_{\boldsymbol{\nu}(\cdot)}$, and $\mathbf{p}_{\boldsymbol{\gamma}}(\cdot)$ may be infinite-dimensional. We will later show in Section 5.8 how RFFs allow us to approximate these infinite-dimensional embeddings in practice.

5.4.2 Lifting of Control-Affine Systems

In this section, we apply the RKHS embedding framework to control-affine systems, showing how they can be lifted into a feature space where the dynamics admit a bilinear representation. We begin with a control-affine system affected by process and measurement noise,

$$\boldsymbol{\xi}_k = \mathbf{f}_0(\boldsymbol{\xi}_{k-1}) + \sum_{i=1}^{N_\nu} \mathbf{f}_i(\boldsymbol{\xi}_{k-1})\nu_{k,i} + \boldsymbol{\omega}_k, \tag{5.8a}$$

$$\boldsymbol{\gamma}_k = \mathbf{g}(\boldsymbol{\xi}_k, \boldsymbol{\eta}_k), \tag{5.8b}$$

where $\boldsymbol{\xi}_k \in \mathbb{R}^{N_\xi}$, $\boldsymbol{\nu}_k = \begin{bmatrix} \nu_{k,1} & \cdots & \nu_{k,N_\nu} \end{bmatrix}^\top \in \mathbb{R}^{N_\nu}$, $\boldsymbol{\omega}_k \in \mathbb{R}^{N_\xi}$, $\boldsymbol{\gamma}_k \in \mathbb{R}^{N_\gamma}$, $\boldsymbol{\eta}_k \in \mathbb{R}^{N_\eta}$ represents the same quantities as in the general nonlinear system in (5.1), and $\mathbf{f}_i$ are the various components of the dynamic model. We look to lift this system into an RKHS. With a sufficiently rich feature map, $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$, a deterministic (i.e., $\boldsymbol{\omega}_k = \mathbf{0}$) control-affine motion model can be written exactly as a bilinear model in a lifted space [13],

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\boldsymbol{\nu}_k + \mathbf{H}\left(\boldsymbol{\nu}_k \otimes \mathbf{x}_{k-1}\right), \tag{5.9}$$

where $\otimes$ represents the tensor product, equivalent to the Kronecker product if $\mathcal{X}$ is finite-dimensional. In the deterministic case, a control-affine model involves products of nonlinear functions of the robot state and the input, but not products that are nonlinear in both. Such a model can be written exactly as a lifted bilinear model [13]. In our stochastic case, we have also extended the assumption made in Chapter 4 from the measurement model to the full system, modeling also the process model noise as approximately Gaussian. Although this is approximate, the aggregation of multiple noise sources in high-dimensional feature spaces often leads to distributions that are well-approximated as Gaussian under the Central Limit Theorem. This assumption provides a tractable framework for learning and inference in the lifted space. The original equivalence by [13] used the unlifted control input, $\boldsymbol{\nu}_k$, in the lifted space, but we will use its lifted counterpart, $\mathbf{u}_k$, since the equivalence still holds if $\mathbf{p}_{\boldsymbol{\nu}}(\cdot)$ is sufficiently rich.

For the measurement model, we assume that the lifted model is linear in the deterministic case,

$$\boldsymbol{\gamma}_k = \mathbf{g}(\boldsymbol{\xi}_k, \boldsymbol{\eta}_k = \mathbf{0}) \;\Rightarrow\; \mathbf{y}_k = \mathbf{D}\mathbf{x}_k, \tag{5.10}$$

and we make a similar additive-Gaussian assumption for the measurement noise. The resulting time-invariant stochastic bilinear system in the lifted space can be written as

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{H}\left(\mathbf{u}_k \otimes \mathbf{x}_{k-1}\right) + \mathbf{w}_k, \tag{5.11a}$$

$$\mathbf{y}_k = \mathbf{D}\mathbf{x}_k + \mathbf{n}_k, \tag{5.11b}$$

where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ are the process and measurement noises, respectively, and

$$\mathbf{w}_k \in \mathcal{X}, \quad \mathbf{n}_k \in \mathcal{Y}, \quad \mathbf{Q} \in \mathcal{X} \times \mathcal{X}, \quad \mathbf{R} \in \mathcal{Y} \times \mathcal{Y}, \tag{5.12a}$$

$$\mathbf{A} : \mathcal{X} \to \mathcal{X}, \quad \mathbf{B} : \mathcal{U} \to \mathcal{X}, \quad \mathbf{H} : \mathcal{U} \otimes \mathcal{X} \to \mathcal{X}, \quad \mathbf{D} : \mathcal{X} \to \mathcal{Y}. \tag{5.12b}$$

This lifted system with a bilinear motion model and a linear measurement model is significantly easier to work with than the general control-affine system in (5.8). However, since we assumed the system model is not given in either form (original or lifted), we first need a method of learning the lifted model from data.

5.5 System Identification

5.5.1 Lifted Matrix Form of Dataset

Our objective is to learn the lifted system matrices $\mathbf{A}, \mathbf{B}, \mathbf{H}, \mathbf{D}, \mathbf{Q}, \mathbf{R}$ in (5.11) from data. This training procedure is similar to that in Section 4.5.1 of the previous chapter, with the added complexity of also learning the process model. To this end, we assume a dataset of the control-affine system, including the groundtruth state transitions with their associated control inputs and measurements for P states: $\{\boldsymbol{\xi}_i^-, \boldsymbol{\xi}_i^+, \boldsymbol{\nu}_i, \boldsymbol{\gamma}_i\}_{i=1}^P$. Here, $\boldsymbol{\xi}_i^-$ transitions to $\boldsymbol{\xi}_i^+$ under input $\boldsymbol{\nu}_i$, and receives a measurement $\boldsymbol{\gamma}_i$ at $\boldsymbol{\xi}_i^+$. This format allows for data from one or multiple training trajectories to be used at once. If the dataset consists of a single trajectory of $P+1$ states and i represents the timestep, then we would set $\boldsymbol{\xi}_i^- = \boldsymbol{\xi}_{i-1}^+$. In any case, we start by writing the data in block-matrix form:

$$\boldsymbol{\Xi}_{(-)} = \begin{bmatrix} \boldsymbol{\xi}_1^- & \cdots & \boldsymbol{\xi}_P^- \end{bmatrix}, \quad \boldsymbol{\Xi}_{(+)} = \begin{bmatrix} \boldsymbol{\xi}_1^+ & \cdots & \boldsymbol{\xi}_P^+ \end{bmatrix}, \quad \boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{\gamma}_1 & \cdots & \boldsymbol{\gamma}_P \end{bmatrix}, \quad \boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{\nu}_1 & \cdots & \boldsymbol{\nu}_P \end{bmatrix}. \tag{5.13}$$

The data translates to $\{\mathbf{x}_i^-, \mathbf{x}_i^+, \mathbf{u}_i, \mathbf{y}_i\}_{i=1}^P$ in the lifted space such that

$$\mathbf{x}_i^+ = \mathbf{A}\mathbf{x}_i^- + \mathbf{B}\mathbf{u}_i + \mathbf{H}\left(\mathbf{u}_i \otimes \mathbf{x}_i^-\right) + \mathbf{w}_i, \tag{5.14a}$$

$$\mathbf{y}_i = \mathbf{D}\mathbf{x}_i^+ + \mathbf{n}_i, \tag{5.14b}$$

for some unknown noise, $\mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, $\mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. We rewrite the lifted versions of the data and the noises in block-matrix form:

$$\mathbf{X}_{(-)} = \begin{bmatrix} \mathbf{x}_1^- & \cdots & \mathbf{x}_P^- \end{bmatrix}, \quad \mathbf{X}_{(+)} = \begin{bmatrix} \mathbf{x}_1^+ & \cdots & \mathbf{x}_P^+ \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & \cdots & \mathbf{y}_P \end{bmatrix}, \tag{5.15a}$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_P \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_P \end{bmatrix}, \quad \mathbf{N} = \begin{bmatrix} \mathbf{n}_1 & \cdots & \mathbf{n}_P \end{bmatrix}. \tag{5.15b}$$

The lifted matrix form of the system for this dataset is

$$\mathbf{X}_{(+)} = \mathbf{A}\mathbf{X}_{(-)} + \mathbf{B}\mathbf{U} + \mathbf{H}\left(\mathbf{U} \odot \mathbf{X}_{(-)}\right) + \mathbf{W}, \tag{5.16a}$$

$$\mathbf{Y} = \mathbf{D}\mathbf{X}_{(+)} + \mathbf{N}, \tag{5.16b}$$

where $\odot$ denotes the Khatri-Rao (column-wise) tensor product.

5.5.2 Loss function

We aim to optimize for the system matrices from the data by designing and minimizing a loss function. Similar to the procedure in Section 4.5.1 in the previous chapter, the model learning problem is posed as

$$\{\mathbf{A}^\star, \mathbf{B}^\star, \mathbf{H}^\star, \mathbf{D}^\star, \mathbf{Q}^\star, \mathbf{R}^\star\} = \underset{\{\mathbf{A},\mathbf{B},\mathbf{H},\mathbf{D},\mathbf{Q},\mathbf{R}\}}{\operatorname{argmin}} V(\mathbf{A}, \mathbf{B}, \mathbf{H}, \mathbf{D}, \mathbf{Q}, \mathbf{R}), \tag{5.17}$$

where the loss function is $V = V_1 + V_2$ with

$$\begin{aligned} V_1 =& \frac{1}{2} \left\| \mathbf{X}_{(+)} - \mathbf{A}\mathbf{X}_{(-)} - \mathbf{B}\mathbf{U} - \mathbf{H}\left(\mathbf{U} \odot \mathbf{X}_{(-)}\right) \right\|_{\mathbf{Q}^{-1}}^2 \\ &+ \frac{1}{2} \left\| \mathbf{Y} - \mathbf{D}\mathbf{X}_{(+)} \right\|_{\mathbf{R}^{-1}}^2 - \frac{1}{2} P \ln \left| \mathbf{Q}^{-1} \right| - \frac{1}{2} P \ln \left| \mathbf{R}^{-1} \right|, \end{aligned} \tag{5.18a}$$

$$\begin{aligned} V_2 =& \frac{1}{2} P\tau_A \left\| \mathbf{A} \right\|_{\mathbf{Q}^{-1}}^2 + \frac{1}{2} P\tau_B \left\| \mathbf{B} \right\|_{\mathbf{Q}^{-1}}^2 + \frac{1}{2} P\tau_H \left\| \mathbf{H} \right\|_{\mathbf{Q}^{-1}}^2 \\ &+ \frac{1}{2} P\tau_D \left\| \mathbf{D} \right\|_{\mathbf{R}^{-1}}^2 + \frac{1}{2} P\tau_Q \operatorname{tr}(\mathbf{Q}^{-1}) + \frac{1}{2} P\tau_R \operatorname{tr}(\mathbf{R}^{-1}). \end{aligned} \tag{5.18b}$$

Similar to (4.15), $\|\cdot\|_{\mathbf{W}}$ is the weighted Frobenius norm (4.16), V_1 corresponds to the negative log-likelihood of the data under Gaussian noise, and V_2 corresponds to the negative log-priors on the model parameters. Here, the first four terms of V_2 encourage the description length of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{H}$, and $\mathbf{D}$ to be minimal while the last two are (isotropic) inverse-Wishart (IW) priors for the covariances $\mathbf{Q}$ and $\mathbf{R}$. The regularizing hyperparameters, $\tau_A, \tau_B, \tau_H, \tau_D, \tau_Q, \tau_R$, are tuned according to data.

We find the critical points by setting derivatives of V with respect to the model parameters ($\frac{\partial V}{\partial \mathbf{A}}$, $\frac{\partial V}{\partial \mathbf{B}}$, $\frac{\partial V}{\partial \mathbf{H}}$, $\frac{\partial V}{\partial \mathbf{D}}$, $\frac{\partial V}{\partial \mathbf{Q}^{-1}}$, and $\frac{\partial V}{\partial \mathbf{R}^{-1}}$) to zero. We define

$$\mathbf{V} = \mathbf{U} \odot \mathbf{X}_{(-)}, \quad \mathbf{J} = \mathbf{X}_{(+)} - \mathbf{A}\mathbf{X}_{(-)} - \mathbf{B}\mathbf{U} - \mathbf{H}\mathbf{V}. \tag{5.19}$$

This yields the following expressions:

$$\begin{bmatrix} \mathbf{X}_{(-)}\mathbf{X}_{(-)}^\top + P\tau_A\mathbf{1} & \mathbf{X}_{(-)}\mathbf{U}^\top & \mathbf{X}_{(-)}\mathbf{V}^\top \\ \mathbf{U}\mathbf{X}_{(-)}^\top & \mathbf{U}\mathbf{U}^\top + P\tau_B\mathbf{1} & \mathbf{U}\mathbf{V}^\top \\ \mathbf{V}\mathbf{X}_{(-)}^\top & \mathbf{V}\mathbf{U}^\top & \mathbf{V}\mathbf{V}^\top + P\tau_H\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{A}^\top \\ \mathbf{B}^\top \\ \mathbf{H}^\top \end{bmatrix} = \begin{bmatrix} \mathbf{X}_{(-)}\mathbf{X}_{(+)}^\top \\ \mathbf{U}\mathbf{X}_{(+)}^\top \\ \mathbf{V}\mathbf{X}_{(+)}^\top \end{bmatrix}, \tag{5.20a}$$

$$\mathbf{D} = (\mathbf{Y}\mathbf{X}_{(+)}^\top)(\mathbf{X}_{(+)}\mathbf{X}_{(+)}^\top + P\tau_D\mathbf{1})^{-1}, \tag{5.20b}$$

$$\mathbf{Q} = \frac{1}{P}\mathbf{J}\mathbf{J}^\top + \tau_A\mathbf{A}\mathbf{A}^\top + \tau_B\mathbf{B}\mathbf{B}^\top + \tau_H\mathbf{H}\mathbf{H}^\top + \tau_Q\mathbf{1}, \tag{5.20c}$$

$$\mathbf{R} = \frac{1}{P}(\mathbf{Y} - \mathbf{D}\mathbf{X}_{(+)})(\mathbf{Y} - \mathbf{D}\mathbf{X}_{(+)})^\top + \tau_D\mathbf{D}\mathbf{D}^\top + \tau_R\mathbf{1}. \tag{5.20d}$$

Thus, all matrices can be computed in closed form through a one-shot procedure: we first solve for $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{H}$ through solving a system of linear equations, then compute $\mathbf{D}$, and then use these results to compute $\mathbf{Q}$ and $\mathbf{R}$. This procedure is linear in the amount of training data, P, for both computation and storage.

5.6 Batch Linear State Estimation

Having learned the system matrices from training data, we now wish to solve for a sequence of test states, $\{\boldsymbol{\xi}_k\}_{k=0}^K$, given a series of inputs, $\{\boldsymbol{\nu}_k\}_{k=1}^K$, and measurements, $\{\boldsymbol{\gamma}_k\}_{k=0}^K$. We do this in the lifted space, where the quantities become $\{\mathbf{x}_k\}_{k=0}^K$, $\{\mathbf{u}_k\}_{k=1}^K$, and $\{\mathbf{y}_k\}_{k=0}^K$, respectively. The main insight is that since the inputs are completely determined at test time, we can manipulate the lifted time-invariant bilinear form of (5.11) into a lifted time-varying linear form. We observe that

$$\mathbf{u}_k \otimes \mathbf{x}_{k-1} = (\mathbf{u}_k \otimes \mathbf{1})\mathbf{x}_{k-1}, \tag{5.21}$$

where here $\mathbf{1} : \mathcal{X} \to \mathcal{X}$. The motion model for the test trajectory becomes, for $k = 1, \ldots, K$,

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{H}\left(\mathbf{u}_k \otimes \mathbf{1}\right)\mathbf{x}_{k-1} + \mathbf{w}_k. \tag{5.22}$$

As $\mathbf{u}_k$ is given at test time, we define a new time-varying system matrix $\mathbf{A}_{k-1}$ and input $\mathbf{v}_k$ as

$$\mathbf{A}_{k-1} = \mathbf{A} + \mathbf{H}(\mathbf{u}_k \otimes \mathbf{1}), \quad \mathbf{v}_k = \mathbf{B}\mathbf{u}_k. \tag{5.23}$$

With this, we have converted the bilinear system into a linear time-varying (LTV) system:

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k, \qquad k = 1, \ldots, K, \tag{5.24a}$$

$$\mathbf{y}_k = \mathbf{D}\mathbf{x}_k + \mathbf{n}_k, \qquad k = 0, \ldots, K, \tag{5.24b}$$

where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. This is the well-established batch state-estimation problem on linear-Gaussian systems [1]. The solution is in the form of

$$\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k), \quad k = 0, \ldots, K. \tag{5.25}$$

Here, $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{P}}_k$ are, respectively, the mean and covariance estimates for the test trajectory. There are many efficient methods for solving (5.24), including the RTS smoother and sparse Cholesky-based solvers [1]. In the present localization setting, the RTS smoother is convenient due to the chain-structured temporal dependencies. More general sparse solvers, such as Cholesky factorization, will become particularly useful in the next chapter when considering SLAM.

5.7 Recovering Estimates from Lifted Space

In lifted representations, a key challenge is recovering estimates in the original state space. The feature space may be implicit and potentially infinite-dimensional, making direct recovery infeasible. In our Koopman-inspired formulations developed in Chapter 4 and later in Chapter 6, this issue is addressed by explicitly including the original state within the lifted representation. While effective, we show that the structure of RKHS embeddings provides an alternative mechanism for recovering estimates without such explicit inclusion. In the next section, this recovery mechanism also plays a role in establishing the connection between kernel-based and Koopman-inspired formulations.

We have solved for the state estimates and covariances in the RKHS, and now wish to recover these quantities in the original space. By the Representer Theorem [92], the solution of the quadratic

batch estimation problem associated with the LTV system in (5.24) lies in the span of the training data used to form the system matrices. We can thus write the state and covariance outputs of each timestep as

$$\hat{\mathbf{x}}_k = \mathbf{X}_{(+)}\mathbf{x}_k^\bullet, \quad \hat{\mathbf{P}}_k = \mathbf{X}_{(+)}\mathbf{P}_k^\bullet\mathbf{X}_{(+)}^\top, \tag{5.26}$$

where $\mathbf{x}_k^\bullet \in \mathbb{R}^P$, $\mathbf{P}_k^\bullet \in \mathbb{R}^{P\times P}$ consist of the appropriate weights. Here, we use $\mathbf{X}_{(+)}$ as the spanning basis, though one could equivalently parameterize the solution using $\mathbf{X}_{(-)}$. We solve for the weights using the left pseudoinverse with a small hyperparameter τ_x to regularize the inversion of $\mathbf{X}_{(+)}^\top\mathbf{X}_{(+)}$:

$$\mathbf{x}_k^\bullet \approx (\mathbf{X}_{(+)}^\top\mathbf{X}_{(+)} + \tau_x\mathbf{1})^{-1}\mathbf{X}_{(+)}^\top\hat{\mathbf{x}}_k, \tag{5.27a}$$

$$\mathbf{P}_k^\bullet \approx (\mathbf{X}_{(+)}^\top\mathbf{X}_{(+)} + \tau_x\mathbf{1})^{-1}\mathbf{X}_{(+)}^\top\hat{\mathbf{P}}_k\mathbf{X}_{(+)}(\mathbf{X}_{(+)}^\top\mathbf{X}_{(+)} + \tau_x\mathbf{1})^{-1}. \tag{5.27b}$$

Now, by the properties of the mean map and the covariance map, these same weights can be used to construct the mean states and covariances in the original space through the weighted linear combination of training data,

$$\hat{\boldsymbol{\xi}}_k = \boldsymbol{\Xi}_{(+)}\mathbf{x}_k^\bullet, \quad \hat{\boldsymbol{\Sigma}}_k = \boldsymbol{\Xi}_{(+)}\mathbf{P}_k^\bullet\boldsymbol{\Xi}_{(+)}^\top, \tag{5.28}$$

where $\boldsymbol{\xi}_k \sim \mathcal{N}(\hat{\boldsymbol{\xi}}_k, \hat{\boldsymbol{\Sigma}}_k)$ is the state at timestep k. We can then solve for the state means and covariances with

$$\hat{\boldsymbol{\xi}}_k = \boldsymbol{\Xi}_{(+)}(\mathbf{X}_{(+)}^\top\mathbf{X}_{(+)} + \tau_x\mathbf{1})^{-1}\mathbf{X}_{(+)}^\top\hat{\mathbf{x}}_k, \tag{5.29a}$$

$$\hat{\boldsymbol{\Sigma}}_k = \boldsymbol{\Xi}_{(+)}(\mathbf{X}_{(+)}^\top\mathbf{X}_{(+)} + \tau_x\mathbf{1})^{-1}\mathbf{X}_{(+)}^\top\hat{\mathbf{P}}_k\mathbf{X}_{(+)}(\mathbf{X}_{(+)}^\top\mathbf{X}_{(+)} + \tau_x\mathbf{1})^{-1}\boldsymbol{\Xi}_{(+)}^\top. \tag{5.29b}$$

We define

$$\mathbf{O}_{\boldsymbol{\xi}} = \boldsymbol{\Xi}_{(+)}\mathbf{X}_{(+)}^\top(\mathbf{X}_{(+)}^\top\mathbf{X}_{(+)} + \tau_x\mathbf{1})^{-1}, \quad \mathbf{O}_{\boldsymbol{\xi}} : \mathcal{X} \to N_\xi. \tag{5.30}$$

While this expression is defined in the RKHS and may be infinite-dimensional, it admits a finite-dimensional approximation in the next section, enabling efficient precomputation during training. Using Sherman–Morrison–Woodbury (SMW) identities, (5.29) becomes

$$\hat{\boldsymbol{\xi}}_k = \mathbf{O}_{\boldsymbol{\xi}}\hat{\mathbf{x}}_k, \quad \hat{\boldsymbol{\Sigma}}_k = \mathbf{O}_{\boldsymbol{\xi}}\hat{\mathbf{P}}_k\mathbf{O}_{\boldsymbol{\xi}}^\top. \tag{5.31}$$

Note that (5.31) assumes that the original states are within a vector space. The states could instead contain angles or other quantities on a circular domain. In this case, $\hat{\boldsymbol{\xi}}_k$ should be a weighted average of circular quantities $\boldsymbol{\Xi}_{(+)}$ with weights $\mathbf{X}_{(+)}^\top(\mathbf{X}_{(+)}\mathbf{X}_{(+)}^\top + \tau_x\mathbf{1})^{-1}$ computed as usual, since the RKHS embeddings are within a vector space, and a similar weighted average is done for $\hat{\boldsymbol{\Sigma}}_k$. We still use (5.31) but slightly modify the computation of $\mathbf{O}_{\boldsymbol{\xi}}$. See Section 5.9 for an example.

5.8 Approximate Embeddings with Random Fourier Features

So far, we have been working in the RKHS space and using the feature maps directly. For many kernels, however, these RKHS feature maps are very high (potentially infinite) dimensional. As it is often unclear how to truncate the feature maps directly to get reasonable approximations, we look for another form of approximate embeddings for $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$, $\mathbf{p}_{\boldsymbol{\nu}}(\cdot)$, and $\mathbf{p}_{\boldsymbol{\gamma}}(\cdot)$ such that the solution computed through the algorithm approaches that from the true embeddings.

We notice that all of the RKHS quantities computed in our procedure appear only to be involved in the form of inner products, $\mathbf{X}^\top\mathbf{X}$, or outer products, $\mathbf{X}\mathbf{X}^\top$. Although the inner product is closely related to kernels, we look to avoid using only kernel evaluations for computation as this would yield a cost of at least $\mathcal{O}(P^3)$. Thus, we turn to RFFs [84] for deriving approximate embeddings. Suppose we have two quantities, $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_j$, in some state space, an operator, $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$, for lifting them into RKHS embeddings, $\mathbf{x}_i$ and $\mathbf{x}_j$, and the kernel function, κ, associated with this RKHS. We can use $\breve{\mathbf{x}}(\cdot)$, the RFF embedding corresponding to κ, to approximate the kernel evaluation as

$$\mathbf{x}_i^\top\mathbf{x}_j = \mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_i)^\top\mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_j) = \kappa(\boldsymbol{\xi}_i,\boldsymbol{\xi}_j) \approx \breve{\mathbf{x}}(\boldsymbol{\xi}_i)^\top\breve{\mathbf{x}}(\boldsymbol{\xi}_j). \tag{5.32}$$

We see that as long as any RKHS quantities within an expression are involved in the form of inner products within the same space (i.e., kernelized), we can swap the exact RKHS embeddings with their RFF embeddings for a valid approximation. We can freely manipulate RKHS quantities within kernelized expressions for efficient computation, and replacing the RKHS embeddings with RFF embeddings still yields a valid approximation.

5.8.1 Sketch that KoopSE is Kernelized

Since we use RFFs as approximate embeddings for $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$, $\mathbf{p}_{\boldsymbol{\nu}}(\cdot)$, and $\mathbf{p}_{\boldsymbol{\gamma}}(\cdot)$, our goal is to show that KoopSE depends on the RKHS quantities only through kernelized operations, even if these forms are not explicitly used for computation.

For training, the lifted training points are used to compute the bilinear system matrices in (5.11). Using SMW identities, it can be shown that the analytical solution of (5.20) depends only on kernelized quantities. Since $\mathbf{X}_{(-)}$ and $\mathbf{X}_{(+)}$ are both constructed from lifted training states and typically span similar RKHS subspaces, we drop the distinction between them and use $\mathbf{X}$ for notational simplicity in the remainder of this sketch. Then, the solutions can be written in the generic kernelized form

$$\mathbf{A} = \mathbf{X}\mathbf{W}_A\mathbf{X}^\top, \quad \mathbf{B} = \mathbf{X}\mathbf{W}_B\mathbf{U}^\top, \qquad \mathbf{H} = \mathbf{X}\mathbf{W}_H\mathbf{V}^\top, \tag{5.33a}$$

$$\mathbf{D} = \mathbf{Y}\mathbf{W}_D\mathbf{X}^\top, \quad \mathbf{Q} = \mathbf{X}\mathbf{W}_Q\mathbf{X}^\top + \tau_Q\mathbf{1}, \quad \mathbf{R} = \mathbf{Y}\mathbf{W}_R\mathbf{Y}^\top + \tau_R\mathbf{1}, \tag{5.33b}$$

where $\mathbf{W}_A$, $\mathbf{W}_B$, $\mathbf{W}_H$, $\mathbf{W}_D$, $\mathbf{W}_Q$, and $\mathbf{W}_R$ are kernelized matrices.

At test time, we assume that the initial condition, new control inputs, and new measurements can be written as linear combinations of those seen in training:

$$\breve{\mathbf{x}}_0 = \mathbf{X}\mathbf{w}_{\breve{x},0}, \tag{5.34a}$$

$$\mathbf{u}_k = \mathbf{U}\mathbf{w}_{u,k}, \quad k = 1,\dots,K, \tag{5.34b}$$

$$\mathbf{y}_k = \mathbf{Y}\mathbf{w}_{y,k}, \quad k = 0, \dots, K, \tag{5.34c}$$

for weights $\mathbf{w}_{\check{x},0}$, $\mathbf{w}_{u,k}$, and $\mathbf{w}_{y,k}$.

Then, from (5.23), the time-varying quantities have the form

$$\mathbf{A}_{k-1} = \mathbf{X}\mathbf{W}_{A,k}\mathbf{X}^\top, \qquad \mathbf{v}_k = \mathbf{X}\mathbf{w}_{v,k}, \qquad k = 0, \dots, K, \tag{5.35}$$

for some kernelized matrices $\mathbf{W}_{A,k}$ and $\mathbf{w}_{v,k}$.

Now, the LTV system (5.24) is solved exactly by the RTS smoother [1]. It can then be shown through induction on the forward and backward passes that the resulting state means and covariances have the forms

$$\hat{\mathbf{x}}_k = \mathbf{X}\mathbf{w}_{\hat{x},k}, \qquad \hat{\mathbf{P}}_k = \mathbf{X}\mathbf{W}_{\hat{P},k}\mathbf{X}^\top + c_k\mathbf{1}, \tag{5.36}$$

for $k = 0, \dots, K$, kernelized matrices $\mathbf{w}_{\hat{x},k}$, $\mathbf{W}_{\hat{P},k}$, and scalar c_k.

Substituting these expressions into (5.29) shows that the recovered means and covariances in the original space, $\hat{\boldsymbol{\xi}}_k$ and $\hat{\boldsymbol{\Sigma}}_k$, also depend only on kernelized quantities. Therefore, from input to output, KoopSE uses the RKHS quantities only through their kernelized forms.

For a more detailed sketch of the proof, see Appendix B.1.

5.8.2 Substituting with Random Fourier Features

As the algorithm is kernelized, we can approximate the kernel functions associated with embeddings $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$, $\mathbf{p}_{\boldsymbol{\nu}}(\cdot)$, and $\mathbf{p}_{\boldsymbol{\gamma}}(\cdot)$ by directly swapping them with the respective finite-dimensional RFF embeddings on the original quantities. When P is large, this yields a much more efficient procedure than using only kernel evaluations. We outline the procedure below, and we analyze the time and memory complexity of the algorithm.

Let $\breve{\mathbf{p}}_{\boldsymbol{\xi}}(\cdot) : \mathbb{R}^{N_\xi} \to \mathbb{R}^{R_x}$, $\breve{\mathbf{p}}_{\boldsymbol{\nu}}(\cdot) : \mathbb{R}^{N_\nu} \to \mathbb{R}^{R_u}$, and $\breve{\mathbf{p}}_{\boldsymbol{\gamma}}(\cdot) : \mathbb{R}^{N_\gamma} \to \mathbb{R}^{R_y}$ represent the RFF embedding for $\boldsymbol{\xi}$, $\boldsymbol{\nu}$, and $\boldsymbol{\gamma}$, with R_x, R_u, R_y being the respective ranks of the approximation. For training, we replace the RKHS embeddings with their RFF counterparts, denoted by $\breve{(\cdot)}$:

$$\mathbf{x}_i^- \leftarrow \breve{\mathbf{x}}_i^- = \breve{\mathbf{p}}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_i^-), \quad \mathbf{x}_i^+ \leftarrow \breve{\mathbf{x}}_i^+ = \breve{\mathbf{p}}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_i^+), \quad \mathbf{u}_i \leftarrow \breve{\mathbf{u}}_i = \breve{\mathbf{p}}_{\boldsymbol{\nu}}(\boldsymbol{\nu}_i), \quad \mathbf{y}_i \leftarrow \breve{\mathbf{y}}_i = \breve{\mathbf{p}}_{\boldsymbol{\gamma}}(\boldsymbol{\gamma}_i), \tag{5.37}$$

where $i = 1, \dots, P$. The embedded training data in block-matrix form have sizes

$$\mathbf{X}_{(-)} \in \mathbb{R}^{R_x \times P}, \quad \mathbf{X}_{(+)} \in \mathbb{R}^{R_x \times P}, \quad \mathbf{Y} \in \mathbb{R}^{R_y \times P}, \quad \mathbf{U} \in \mathbb{R}^{R_u \times P}, \tag{5.38}$$

which are used to compute the now finite-dimensional system matrices of sizes

$$\mathbf{A} \in \mathbb{R}^{R_x \times R_x}, \quad \mathbf{B} \in \mathbb{R}^{R_x \times R_u}, \quad \mathbf{H} \in \mathbb{R}^{R_x \times (R_x R_u)}, \tag{5.39a}$$

$$\mathbf{D} \in \mathbb{R}^{R_y \times R_x}, \quad \mathbf{Q} \in \mathbb{R}^{R_x \times R_x}, \quad \mathbf{R} \in \mathbb{R}^{R_y \times R_y}, \tag{5.39b}$$

through solving (5.20) with the embedded training data. Letting $R = \max(R_x, R_u, R_y)$, this procedure has a computational complexity of $\mathcal{O}(PR^3)$ and a memory complexity of $\mathcal{O}(PR^2)$, both scaling only linearly with the number of training points.

For notational simplicity, we drop the breve notation in what follows and treat all quantities as their finite-dimensional approximations. For testing, we similarly replace the initial condition, $\check{\boldsymbol{\xi}}_0$, incoming control inputs, $\boldsymbol{\nu}_k$, and incoming measurements, $\boldsymbol{\gamma}_k$, with their RFF embeddings, yielding $\check{\mathbf{x}}_0$, $\mathbf{u}_k$, and $\mathbf{y}_k$. We form $\mathbf{A}_{k-1} \in \mathbb{R}^{R_x \times R_x}$ and $\mathbf{v}_k \in \mathbb{R}^{R_x}$ using (5.23), then solve the LTV system in (5.24) using a linear batch state estimator, yielding the state mean, $\hat{\mathbf{x}}_k$, and covariance, $\hat{\mathbf{P}}_k$, for each timestep $k = 0, \ldots, K$. With an RTS smoother or a similarly efficient estimator, this procedure has computation complexity $\mathcal{O}(KR^3)$ and memory complexity $\mathcal{O}(KR^2)$. Note that if we require a filter solution for online state estimation, we can do only the forward pass, which is also kernelized. Finally, we convert the results back into state space using (5.31).

Due to kernelization, although the RFF-equivalents of the RKHS variables and model matrices likely look very different, the results for $\{\hat{\boldsymbol{\xi}}_k, \hat{\boldsymbol{\Sigma}}_k\}$ converge to the true kernelized results as the ranks of the approximations approach infinity. Note that $\mathbf{O}_{\boldsymbol{\xi}}$ can be precomputed during training. As such, the overall computation and memory complexity of testing are still $\mathcal{O}(KR^3)$ and $\mathcal{O}(KR^2)$, respectively, regardless of the amount of training data used. As mentioned before, the computation of $\mathbf{O}_{\boldsymbol{\xi}}$ would need to be slightly modified if there were circular quantities in the original states. See Algorithm 2 for a summary of the full algorithm.

In practice, we use SERFFs (4.21) as a general-purpose choice due to their strong empirical performance and ability to approximate a broad class of smooth functions. Other kernels and corresponding random feature constructions can be used to better capture problem-specific structure, such as periodicity or non-smooth behavior. In our experiments in the next section, we will use SERFFs for vector-space quantities and periodic kernels for angular components.

5.9 Experiments and Results

5.9.1 Problem Setup

KoopSE is evaluated on a control-affine system with a nonlinear measurement model in simulation, then on an experimental dataset with a similar setup. The problem is estimating the positions and headings of a wheeled robot driving in a 2D plane and receiving range measurements from five ultra-wideband (UWB) sensors. For timestep k, the state, input, and measurement are, respectively,

$$\boldsymbol{\xi}_k = \begin{bmatrix} \xi_{x,k} & \xi_{y,k} & \xi_{\theta,k} \end{bmatrix}^\top, \quad \boldsymbol{\nu}_k = \begin{bmatrix} u_k & \omega_k \end{bmatrix}^\top, \quad \boldsymbol{\gamma}_k = \begin{bmatrix} r_{k,1} & \cdots & r_{k,5} \end{bmatrix}^\top, \tag{5.40}$$

where $(\xi_{x,k}, \xi_{y,k})$ is the robot's position, $\xi_{\theta,k}$ is its orientation, u_k is its linear velocity, ω_k is its angular velocity, and $r_{k,j}$ is the range measurement from the robot to the jth UWB sensor. This uses the common state estimation practice of using interoceptive measurements as inputs in the process model [22]. This problem setup is identical to the 2D version of the setup in [1, S8.2].

For the state, we use a product of SERFFs for the robot's position and RFFs for the periodic kernel for its orientation. The formula for generating SERFFs is given in (4.21), and the formula for the periodic RFFs can be found in [93]. The combined RFFs for the state is thus the Kronecker product of the two RFF sets. For the input, we kept it as is since there were no improvements from lifting it to higher dimensions, thereby using a linear kernel. We used SERFFs for the measurement.

Since orientation is on a circular domain, we take special care in computing the weighted average

Algorithm 2: Koopman State Estimator (KoopSE)

Training:

Input: Training data $\{\boldsymbol{\xi}_i^-, \boldsymbol{\xi}_i^+, \boldsymbol{\nu}_i, \boldsymbol{\gamma}_i\}_{i=1}^P$; lifting functions $\{\mathbf{p}_{\boldsymbol{\xi}}(\cdot), \mathbf{p}_{\boldsymbol{\nu}}(\cdot), \mathbf{p}_{\boldsymbol{\gamma}}(\cdot)\}$; algorithm hyperparameters $\{\tau_A, \tau_B, \tau_H, \tau_D, \tau_Q, \tau_R, \tau_x\}$.

1. Stack training data into their block-matrix form with (5.13), yielding $\{\boldsymbol{\Xi}_{(-)}, \boldsymbol{\Xi}_{(+)}, \boldsymbol{N}, \boldsymbol{\Gamma}\}$.
2. Embed $\{\boldsymbol{\Xi}_{(-)}, \boldsymbol{\Xi}_{(+)}, \boldsymbol{N}, \boldsymbol{\Gamma}\}$ into their lifted form, yielding $\{\mathbf{X}_{(-)}, \mathbf{X}_{(+)}, \mathbf{U}, \mathbf{Y}\}$.
3. Solve for model matrices $\{\mathbf{A}, \mathbf{B}, \mathbf{H}, \mathbf{D}, \mathbf{Q}, \mathbf{R}\}$ using (5.20), and compute $\mathbf{O}_{\boldsymbol{\xi}}$ in (5.30).

Output: Process model matrices $\{\mathbf{A}, \mathbf{B}, \mathbf{H}, \mathbf{Q}\}$; measurement model matrices $\{\mathbf{C}, \mathbf{R}\}$; state recovery matrix $\{\mathbf{O}_{\boldsymbol{\xi}}\}$.

Testing:

Input: Initial condition $\check{\boldsymbol{\xi}}_0$; control inputs $\{\boldsymbol{\nu}_k\}_{k=1}^K$; measurements $\{\boldsymbol{\gamma}_k\}_{k=0}^K$.

1. Embed $\left\{\check{\boldsymbol{\xi}}_0, \{\boldsymbol{\nu}_k\}_{k=1}^K, \{\boldsymbol{\gamma}_k\}_{k=0}^K\right\}$ into their lifted form, yielding $\left\{\check{\mathbf{x}}_0, \{\mathbf{u}_k\}_{k=1}^K, \{\mathbf{y}_k\}_{k=0}^K\right\}$.
2. Compute time-varying quantities $\{\mathbf{A}_{k-1}, \mathbf{v}_k\}_{k=0}^K$ via (5.23).
3. Form LTV system in (5.24).
4. Solve for mean and covariance estimates, $\{\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k\}_{k=0}^K$, with an efficient batch state estimator (e.g., RTS smoother).
5. Convert $\{\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k\}_{k=0}^K$ into the original space with (5.31), yielding $\{\hat{\boldsymbol{\xi}}_k, \hat{\boldsymbol{\Sigma}}_k\}_{k=0}^K$.

Output: State means $\{\hat{\boldsymbol{\xi}}_k\}_{k=0}^K$; covariances $\{\hat{\boldsymbol{\Sigma}}_k\}_{k=0}^K$.

of states when recovering estimates via (5.31). We convert the orientations to their Cartesian form,

$$\mathbf{s}_i^+ = \mathbf{q}(\boldsymbol{\xi}_i^+) = \begin{bmatrix} \xi_{x,i}^+ & \xi_{y,i}^+ & \cos(\xi_{\theta,i}^+) & \sin(\xi_{\theta,i}^+) \end{bmatrix}^\top, \tag{5.41}$$

then use $\mathbf{S}_{(+)} = \begin{bmatrix} \mathbf{s}_1^+ & \cdots & \mathbf{s}_P^+ \end{bmatrix}$ instead of $\boldsymbol{\Xi}_{(+)}$ in the computation of $\mathbf{O}_{\boldsymbol{\xi}}$. When computing $\hat{\boldsymbol{\xi}}_k$ and $\hat{\boldsymbol{\Sigma}}_k$ at test time, we first compute their Cartesian-form equivalents: $\hat{\mathbf{s}}_k = \mathbf{O}_{\boldsymbol{\xi}}\hat{\mathbf{x}}_k$, $\hat{\boldsymbol{\Sigma}}_{s,k} = \mathbf{O}_{\boldsymbol{\xi}}\hat{\mathbf{P}}_k\mathbf{O}_{\boldsymbol{\xi}}^\top$, giving us the Gaussian distribution of $\xi_{x,k}$, $\xi_{y,k}$, $\cos\xi_{\theta,k}$, and $\sin\xi_{\theta,k}$. We then use the method of [44] to estimate the distribution of $\xi_{\theta,k}$ given the Gaussians of $\cos\xi_{\theta,k}$ and $\sin\xi_{\theta,k}$.

Similar to Section 4.7.4 in the previous chapter, we use per-timestep RMSE to evaluate accuracy and the normalized Mahalanobis distance (given in (4.28)) to evaluate consistency. An accurate estimator has an RMSE close to 0, and a consistent estimator has a Mahalanobis distance close to 1. We tune the RFF parameters and the regularizing hyperparameters for KoopSE accordingly for these objectives. We compare KoopSE against a model-based Lie-group RTS smoother, obtained by augmenting the Lie-group EKF [1, S8.2] with a backward smoothing pass. Its model covariances are tuned with the same objectives, including increasing the measurement covariances for the two noisy sensors. KoopSE is first verified in simulation, then validated on an experimental dataset of the same setup. We present our results for the two scenarios below.

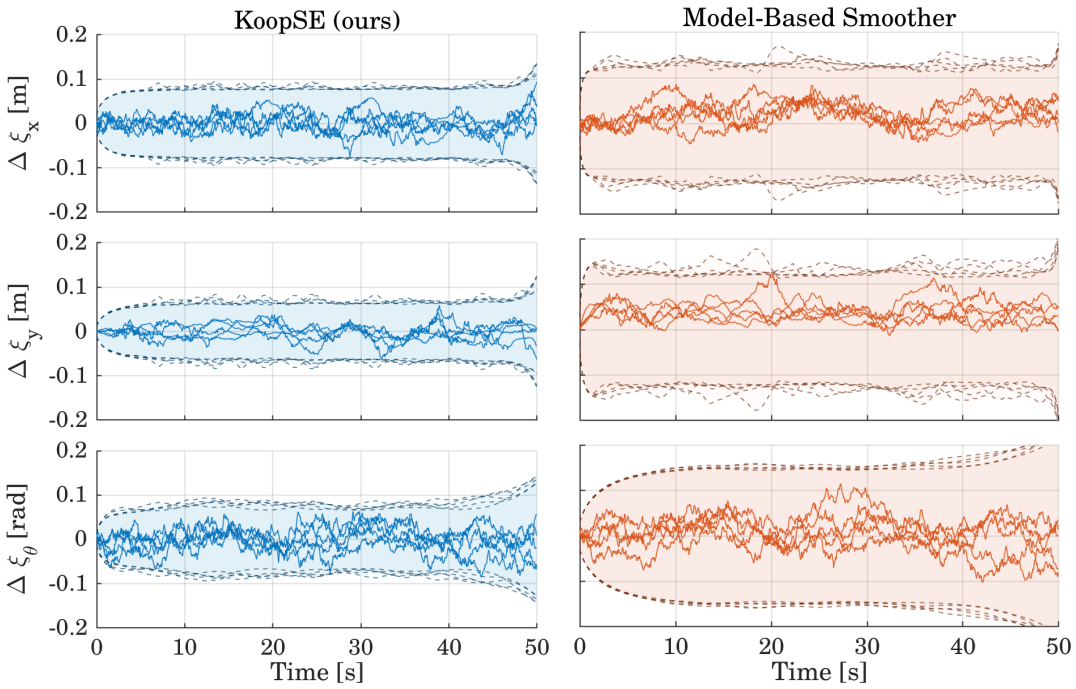


Figure 5.3: Per-axis errors of five test trajectories in simulation for KoopSE (left) and for the model-based extended RTS smoother (right). The solid lines represent the errors of the estimated trajectories, and the dashed envelopes represent the estimated 3σ bounds. Both errors are within the 3σ bounds, but the model-based smoother has larger errors than KoopSE, especially for y where we can see a small bias for the model-based smoother.

	KoopSE (ours)	Model-Based Smoother
Position RMSE [m]	**0.026**	0.053
Orientation RMSE [rad]	**0.026**	0.034
Position Maha. Distance	**0.915**	1.104
Orientation Maha. Distance	**0.777**	0.548

Table 5.1: Average RMSE and Mahalanobis distances of KoopSE and the model-based extended RTS smoother on 100 trajectories of 1000 timesteps in simulation. The Mahalanobis distances of both estimators are fairly close to 1, signifying that both algorithms are properly tuned under the ideal simulation environment. However, KoopSE has lower RMSE than the model-based smoother for both position and orientation.

5.9.2 Simulation Setup

For the simulation, we add a 20 cm unmodeled bias to two out of the five sensors, resulting in a UWB error profile similar to that observed experimentally in Section 5.9.3. Training data is generated by the robot roughly following randomly generated trajectories in an enclosed area. One hundred trajectories of 1000 timesteps are used for evaluation, and the combined results are presented in Table 5.1. The error plots for five trajectories are shown in Fig. 5.3. In Fig. 5.4, we present the results of testing KoopSE on the five trajectories using various numbers of RFFs and number of training data points, in comparison to the model-based smoother.

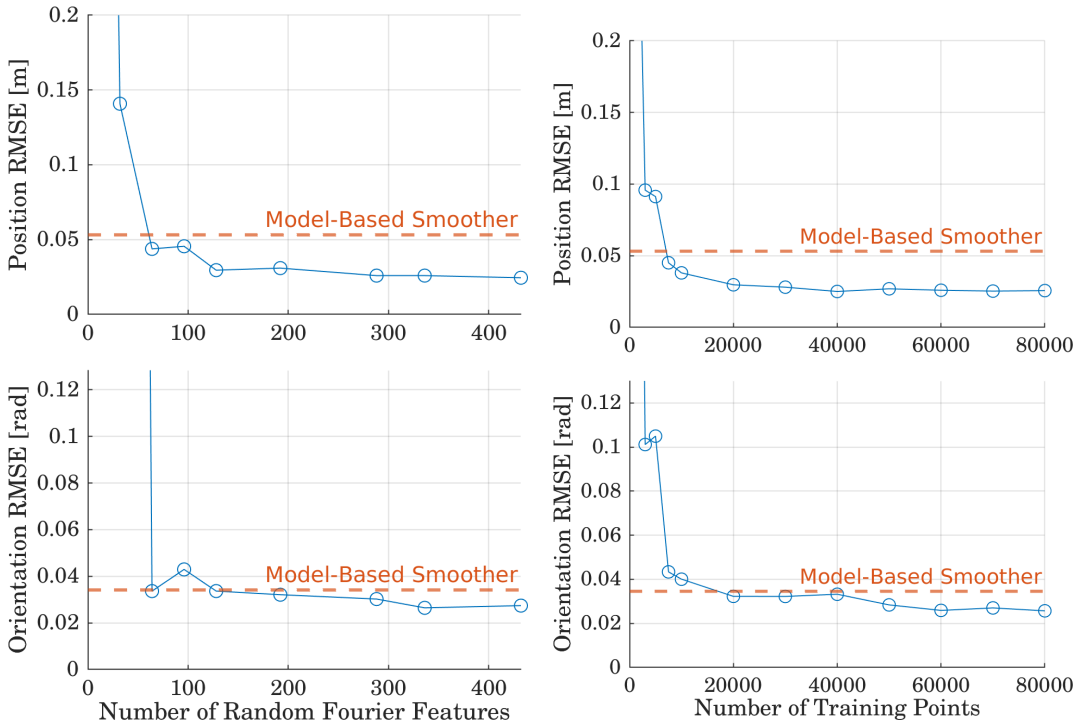


Figure 5.4: RMSE of KoopSE on test trajectories in simulation for various numbers of RFFs (left) and training data points (right). The dotted line represents the RMSE of the model-based smoother for both vertical axes (position and orientation). With only 128 RFFs and 20000 training points, the performance of KoopSE has already surpassed that of the model-based smoother, achieving much lower position RMSE and comparable orientation RMSE.

5.9.3 Experimental Setup

Experiments in a lab setting using a Clearpath Husky Unmanned Ground Vehicle (UGV) are used to validate the proposed approach. A 30-minute dataset of the UGV driving in an indoor environment is collected. Groundtruth positions and orientations are collected using an OptiTrack motion capture system. The wheel odometry consisting of forward velocity and yaw rate is calculated from wheel encoders. We used five UWB anchors transmitting range measurements, with two of the anchors obstructed with metal plates representing clutter in the environment, creating additional measurement bias on the order of 30 cm. A picture of the Husky and the anchors is shown in Fig. 5.2.

To validate the generality of KoopSE, we run a 6-fold cross-validation, training on 25 minutes of data and testing on a random 100-second section of the remainder. The error plots for each fold are shown in Fig. 5.5. The RMSE and Mahalanobis distances for the folds are shown in Fig. 5.6.

5.10 Results and Discussion

The results highlight the benefits of the data-driven approach. Table 5.1 and Fig. 5.3 show that in simulation, KoopSE has lower errors than the model-based extended RTS smoother, which is ignorant of the range biases, even though both methods are consistent. This shows that, despite having little knowledge of the robot's motion model, the UWB range measurement model, or the position of the

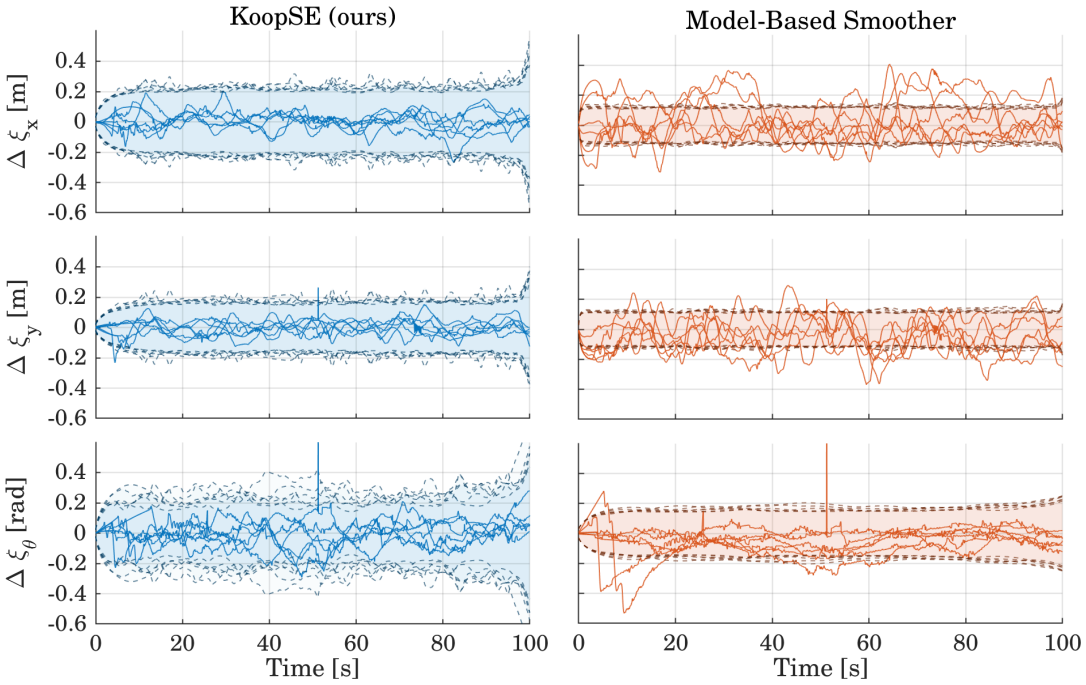


Figure 5.5: Error plots of the 6 folds of the UWB experimental dataset for KoopSE (left) and the model-based extended RTS smoother (right). The solid lines represent the errors of the estimated trajectories, and the dashed envelopes represent the estimated 3σ bounds. The errors of KoopSE are smaller and bounded by the 3σ bounds, while those of the model-based smoother are larger and often not bounded.

anchors, KoopSE outperforms the classical method when a small unmodeled measurement bias is introduced. In fact, when the bias is removed, we found that KoopSE performed just as well as the model-based smoother. In Fig. 5.4, the RMSE of KoopSE quickly decreases over the first 100 RFFs and the first 10000 training points. KoopSE surpassed the classical smoother with only 128 RFFs and 10000 training points (about 8 minutes), making it feasible for real-world applications.

The results on the experimental dataset validate our findings. As shown in Fig. 5.6, compared to the model-based smoother, KoopSE has achieved lower translational RMSE and similar orientation RMSE for all folds. Unlike for the simulation setting, tuning the covariance parameters of the model-based smoother to achieve the lowest RMSE results in overconfident position estimates. This is likely due to unmodeled effects in its motion or sensor models, which KoopSE overcame with its model-free approach.

5.11 Conclusion

Overall, KoopSE provides an efficient framework for data-driven state estimation of control-affine systems. By lifting the system into a higher-dimensional space, nonlinear dynamics can be approximated as bilinear-Gaussian, enabling state estimation using familiar linear tools. We demonstrated its effectiveness on a nonlinear UWB tracking problem, highlighting its applicability in scenarios with complex dynamics or unknown system models. One limitation of the current formulation is that despite not requiring prior knowledge of the underlying system model, training requires access

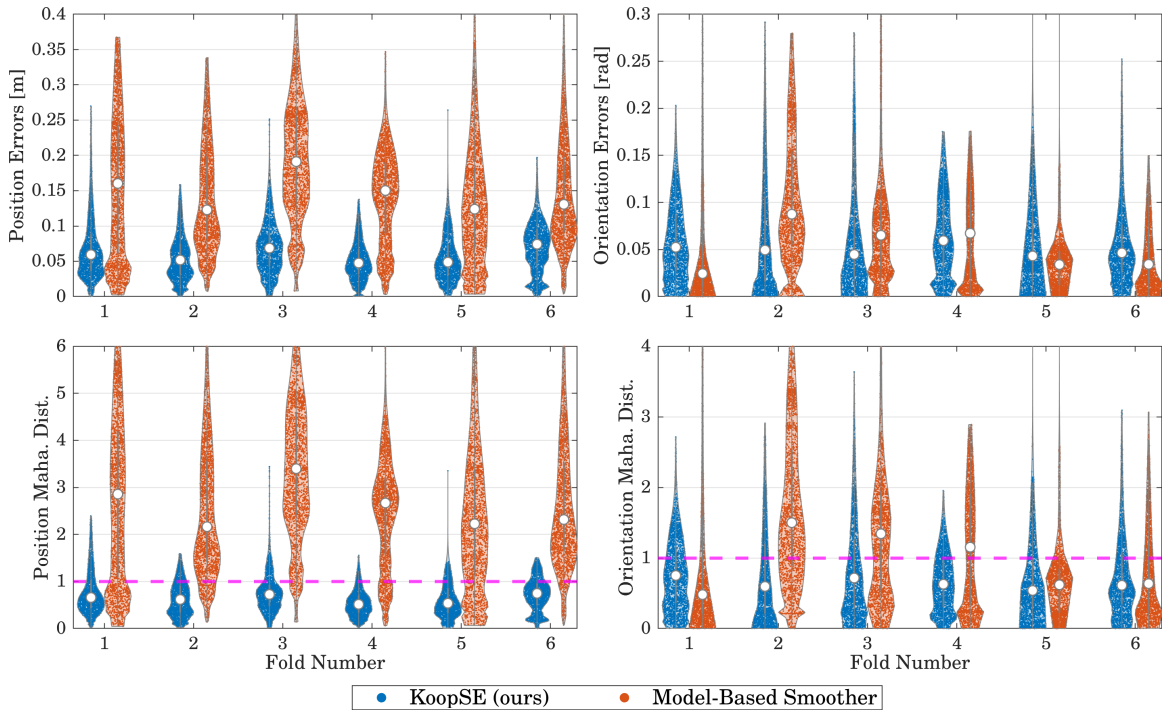


Figure 5.6: Errors and Mahalanobis distances for position and orientation for KoopSE and the model-based extended RTS smoother on the 6 folds of the robot dataset. For position errors, KoopSE has a lower RMSE and a more consistent Mahalanobis distance than the model-based smoother across all folds. For errors in orientation, which is less affected by the UWB range measurements, KoopSE performs just as well as the model-based smoother.

to groundtruth states. Future work could therefore investigate learning lifted system representations directly from measurements and control inputs.

Within the broader context of this thesis, this chapter extends Koopman-inspired lifting from learned measurement models to full system-level estimation. In contrast to the recursive filtering setting of Chapter 4, this formulation enables batch estimation by learning both process and measurement models within a unified lifted framework. More broadly, it shows that nonlinear state estimation problems can be approximately recast as structured linear-Gaussian inference in a lifted space, significantly expanding the role of data-driven lifting. This chapter also provides a theoretical interpretation of the lifting process, showing that Koopman-inspired representations can be viewed as approximations to kernel embeddings, with RFFs offering a tractable finite-dimensional realization.

A natural next step is to extend this framework to SLAM, where the robot trajectory and unknown landmarks must be estimated jointly. In such settings, however, accurate estimation requires more than expressive lifted models. When measurements are sparse or when multiple variables are estimated jointly in the lifted space, the inferred quantities can deviate from the underlying lifted manifold. In these settings, preserving manifold consistency becomes critical for accurate estimation. A conceptual preview of these challenges is provided in Fig. 6.3. This motivates the next chapter, where we extend Koopman-inspired lifted estimation to SLAM and develop methods for enforcing consistency constraints in lifted estimation frameworks.

Chapter 6

Koopman-Inspired SLAM

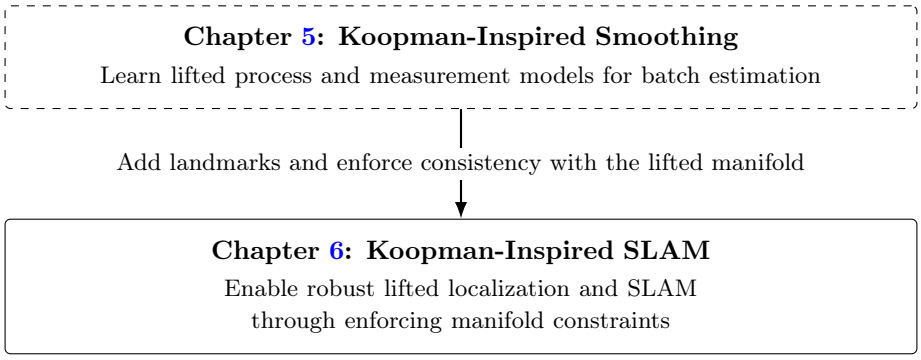


Figure 6.1: Conceptual progression from Chapter 5 to Chapter 6.

6.1 Summary of Contributions

This chapter extends the lifted estimation framework of Chapter 5 from trajectory estimation to full SLAM. In doing so, it reveals an important consideration that arises in these richer estimation settings: accurate inference requires the lifted variables to remain consistent with the underlying lifted manifold.

To address this, we develop the Reduced Constrained Koopman Linearization (RCKL) framework for data-driven localization and SLAM. This chapter

- extends Koopman-inspired lifted estimation to the joint inference of robot states and landmark locations,
- introduces a constrained estimation framework that enforces consistency with the lifted manifold during optimization, and
- demonstrates experimentally that these constraints enable accurate data-driven localization and SLAM in settings where unconstrained lifted estimation degrades.

Within the broader context of this thesis, this chapter shows that preserving consistency with the lifted manifold is essential for reliable Koopman-inspired estimation. By incorporating such

constraints, lifted methods become more robust in challenging estimation settings, enabling their extension to practical localization and mapping problems.

Associated publication: Z. C. Guo, F. Dümbgen, J. R. Forbes, and T. D. Barfoot, "Data-driven batch localization and SLAM using Koopman linearization," *IEEE Transactions on Robotics*, vol. 40, pp. 3964–3983, 2024. DOI: 10.1109/TRO.2024.3443674.

6.2 Motivation

Chapter 5 showed that Koopman-inspired lifting can be used for data-driven state estimation when the relevant environment structure is known. In many real-world settings, however, robots must operate in previously unseen environments, estimating their trajectory while simultaneously building a map of the surrounding landmarks. Extending lifted estimation to SLAM is therefore a natural and practically important next step.

The need for data-driven SLAM is especially relevant in systems where collecting data is easier than deriving accurate process or measurement models. For sensing modalities such as UWB [78], RFID [79], or contact-rich manipulation [94], the procedure for modeling and/or solving the estimation problem can be far more involved than the data-gathering process. A data-driven SLAM framework therefore offers an attractive alternative to conventional model-based estimation pipelines.

In this chapter, we develop the RCKL framework for data-driven localization and SLAM. The framework

- learns a high-dimensional bilinear system model from data without requiring prior models,
- is applicable to systems with control-affine dynamics,
- solves for the unknown states and landmark positions with an inference cost that scales linearly with the number of timesteps, and
- has a training cost that scales linearly with the amount of data and an inference cost that is independent of the amount of training data.

RCKL extends KoopSE by supporting localization with landmarks at new positions at test time, and by enabling SLAM when landmark locations are unknown. It further enforces manifold consistency through constrained optimization over the manifolds induced by the lifting functions, and solves the resulting problem efficiently by formulating an SQP at each iteration [28].

Given a robot with unknown nonlinear dynamics and measurements, the system model may first be learned in a controlled setting with ground truth data. The learned model can then be deployed in the field using RCKL-SLAM to estimate new landmarks, after which RCKL-Loc can be used for localization against the resulting map. See Fig. 6.2 for a preview of the outputs of RCKL-Loc and RCKL-SLAM on a real-world dataset.

This chapter is structured as follows. We review related work in Section 6.3 and discuss lifting the system in Section 6.4. We outline the model-learning procedure in Section 6.5. We then build up the design of RCKL through Sections 6.6-6.8, where we discuss the procedure and limitations of Unconstrained Koopman Linearization (UKL) in Section 6.6, Constrained Koopman Linearization (CKL) in Section 6.7, and finally RCKL in Section 6.8. We present experimental results in Section 6.9 and conclude in Section 6.10.

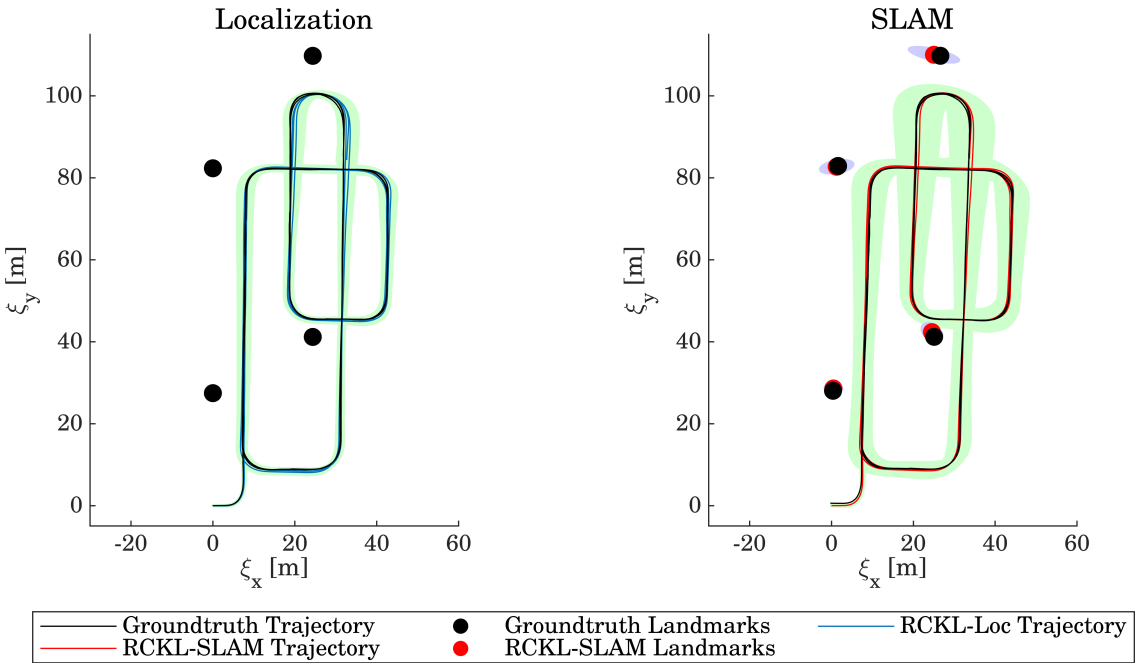


Figure 6.2: Visualization of the trajectory output of RCKL-Loc (left), and the trajectory and landmark output of RCKL-SLAM (right), on Experiment 2 described in Section 6.9.4, showing the estimators' mean states and mean landmark positions compared to the groundtruth. The green regions and the grey regions are, respectively, the 3σ covariances of the trajectory and of the landmarks. The estimators' trajectories and landmarks are close to the groundtruth and are within the estimated 3σ bounds, despite the algorithm having little prior knowledge of the system models.

6.3 Related Work

Koopman-inspired methods have been widely explored for system identification [8, 9], control [11, 7, 12], and more recently state estimation [14, 15, 16, 17]. Related lifted system-identification methods based on kernels and Koopman-inspired representations often yield well-conditioned linear-like learning problems in lifted spaces [95, 96, 71]. However, most existing estimation approaches have been validated only on relatively small simulation problems or short-horizon filtering tasks. Applications to large-scale batch estimation problems such as localization and SLAM remain comparatively limited, and incorporating learned lifted models into classical nonlinear estimation frameworks can be challenging when the lifted representations are noisy or high dimensional [1].

In the previous chapter, we introduced KoopSE as a Koopman-inspired framework for lifted batch state estimation using linear-Gaussian inference. However, KoopSE models all measurements jointly within a fixed training environment, requiring landmarks to remain at the same positions during deployment and preventing direct application to SLAM. Furthermore, extending lifted estimation to larger optimization problems such as SLAM introduces additional structural challenges that require constrained optimization in the lifted space. This chapter addresses these limitations by developing data-driven formulations for localization and SLAM that maintain consistency with the underlying lifted manifold during optimization.

6.4 Lifting the Full System

In this section, we generalize the Koopman lifting framework to our systems of interest, which include noisy process and measurement models. Unlike in the previous chapter, we explicitly include the landmark position in the measurement model. We focus on systems with process and measurement models of the form

$$\boldsymbol{\xi}_k = \mathbf{f}(\boldsymbol{\xi}_{k-1}, \boldsymbol{\nu}_k, \boldsymbol{\omega}_k) = \mathbf{f}_0(\boldsymbol{\xi}_{k-1}) + \sum_{i=1}^{N_\nu} \mathbf{f}_i(\boldsymbol{\xi}_{k-1})\nu_{k,i} + \boldsymbol{\omega}_k, \tag{6.1a}$$

$$\boldsymbol{\gamma}_{k,j} = \mathbf{g}(\boldsymbol{\xi}_k, \boldsymbol{\psi}_j, \boldsymbol{\eta}_{k,j}) \ = \sum_{i=1}^{N_{g,\xi}} \sum_{m=1}^{N_{g,\psi}} \mathbf{g}_{\xi,i}(\boldsymbol{\xi}_k)\pi_{\psi,m}(\boldsymbol{\psi}_j) + \boldsymbol{\eta}_{k,j}, \tag{6.1b}$$

where $\boldsymbol{\xi}_k \in \mathbb{R}^{N_\xi}$ is the robot state, $\boldsymbol{\nu}_k \in \mathbb{R}^{N_\nu}$ the control input, $\boldsymbol{\omega}_k \in \mathbb{R}^{N_\omega}$ the process noise, all at timestep k. $\boldsymbol{\psi}_j \in \mathbb{R}^{N_\psi}$ is the position of landmark j, $\boldsymbol{\gamma}_{k,j} \in \mathbb{R}^{N_\gamma}$ the measurement of $\boldsymbol{\psi}_j$ at timestep k, and $\boldsymbol{\eta}_{k,j} \in \mathbb{R}^{N_\eta}$ the measurement noise for $\boldsymbol{\gamma}_{k,j}$. $\mathbf{f}_i$ are the various components of the process model, and $\mathbf{g}_{\xi,i}, \pi_{\psi,m}$ are the various components of the measurement model. Similar to the previous chapter, we focus on systems with a control-affine process model (6.1a) since the model can be written exactly as a lifted bilinear model when the system is noiseless [13]. In the same spirit, we have also chosen a measurement model of the form (6.1b), where it is nonlinear in the state and the landmark position but not both, such that the model can be exactly written as a lifted bilinear model. Many common robot process and measurement models can be written in this form.

The problem of localization is to estimate the states, $\boldsymbol{\xi}_k$, given a series of inputs, $\boldsymbol{\nu}_k$, landmark positions, $\boldsymbol{\psi}_j$, and measurements, $\boldsymbol{\gamma}_{k,j}$. The problem of SLAM is to estimate the states while moving the landmark positions to the list of unknowns: estimate $\boldsymbol{\xi}_k$ and $\boldsymbol{\psi}_j$ given only $\boldsymbol{\nu}_k$ and $\boldsymbol{\gamma}_{k,j}$. Both problems are difficult when the models, $\mathbf{f}_i$, $\mathbf{g}_{\xi,i}$, $\pi_{\psi,m}$, are nonlinear, and especially so if the models are inaccurate or unknown. Rather than solving these problems directly, we first embed each of the states, inputs, landmarks, and measurements into high-dimensional spaces,

$$\mathbf{x}_k = \mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k), \quad \mathbf{u}_k = \mathbf{p}_{\boldsymbol{\nu}}(\boldsymbol{\nu}_k), \quad \boldsymbol{\ell}_j = \mathbf{p}_{\boldsymbol{\psi}}(\boldsymbol{\psi}_j), \quad \mathbf{y}_{k,j} = \mathbf{p}_{\boldsymbol{\gamma}}(\boldsymbol{\gamma}_{k,j}), \tag{6.2a}$$

where

$$\mathbf{p}_{\boldsymbol{\xi}} : \mathbb{R}^{N_\xi} \to \ \mathcal{X}, \quad \mathbf{p}_{\boldsymbol{\nu}} : \mathbb{R}^{N_\nu} \to \mathcal{U}, \quad \mathbf{p}_{\boldsymbol{\psi}} : \mathbb{R}^{N_\psi} \to \mathcal{L}, \quad \mathbf{p}_{\boldsymbol{\gamma}} : \mathbb{R}^{N_\gamma} \to \mathcal{Y}, \tag{6.3a}$$

are the embeddings associated with the manifolds $\mathcal{X}$, $\mathcal{U}$, $\mathcal{L}$, and $\mathcal{Y}$, respectively. While the previous chapter established a connection between lifting and kernel embeddings, the SLAM setting considered here introduces additional structure and, as shown later in this chapter, the need to enforce consistency constraints. These requirements are most naturally handled using explicit lifted variables, and we therefore adopt finite-dimensional embeddings in this chapter. In practice, the insights from the previous chapter still guide the choice of embeddings. In our experiments in Section 6.9, we still use RFFs as they provide a flexible and principled way to construct these representations, while retaining an implicit connection to kernel-based formulations. In any case, for the remainder

of the chapter, we assume that the manifolds are within finite-dimensional vector spaces:

$$\mathcal{X} \subseteq \mathbb{R}^{N_x}, \quad \mathcal{U} \subseteq \mathbb{R}^{N_u}, \quad \mathcal{L} \subseteq \mathbb{R}^{N_\ell}, \quad \mathcal{Y} \subseteq \mathbb{R}^{N_y}, \tag{6.4}$$

and let $N = \max(N_x, N_u, N_\ell, N_y)$.

For the process model, we follow the same lifting procedure as in the previous chapter, where the control-affine model becomes a bilinear-Gaussian model in the lifted space. That is, (6.1a) becomes

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{H}\left(\mathbf{u}_k \otimes \mathbf{x}_{k-1}\right) + \mathbf{w}_k. \tag{6.5}$$

Again, this lifting is exact in the deterministic case [13], whereas here we also have assumed that any noise becomes approximately Gaussian in the lifted space. In the same spirit, we now lift the measurement model in (6.1b) so that it also becomes bilinear in the deterministic case,

$$\boldsymbol{\gamma}_{k,j} = \mathbf{g}(\boldsymbol{\xi}_k, \boldsymbol{\psi}_j, \mathbf{0}) \;\Rightarrow\; \mathbf{y}_{k,j} = \mathbf{D}(\boldsymbol{\ell}_j \otimes \mathbf{x}_k), \tag{6.6}$$

and assume that the noise from a stochastic measurement model can also be modeled as additive-Gaussian noise. This results in the full lifted system,

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{H}\left(\mathbf{u}_k \otimes \mathbf{x}_{k-1}\right) + \mathbf{w}_k, \tag{6.7a}$$

$$\mathbf{y}_{k,j} = \mathbf{D}(\boldsymbol{\ell}_j \otimes \mathbf{x}_k) + \mathbf{n}_{k,j}, \tag{6.7b}$$

where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\mathbf{n}_{k,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ are the process and measurement noises, respectively, and

$$\mathbf{A} : \mathcal{X} \to \mathcal{X}, \quad \mathbf{B} : \mathcal{U} \to \mathcal{X}, \quad \mathbf{H} : \mathcal{U} \otimes \mathcal{X} \to \mathcal{X}, \quad \mathbf{D} : \mathcal{L} \otimes \mathcal{X} \to \mathcal{Y}, \tag{6.8a}$$

$$\mathbf{w}_k \in \mathcal{X}, \quad \mathbf{n}_{k,j} \in \mathcal{Y}, \quad \mathbf{Q} \in \mathcal{X} \times \mathcal{X}, \quad \mathbf{R} \in \mathcal{Y} \times \mathcal{Y}, \tag{6.8b}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are symmetric positive definite. Note that for a closer parallel to the process model, the measurement model would contain three terms, $\mathbf{y}_{k,j} = \mathbf{D}_1\mathbf{x}_k + \mathbf{D}_2\boldsymbol{\ell}_j + \mathbf{D}_3(\boldsymbol{\ell}_j \otimes \mathbf{x}_k) + \mathbf{n}_{k,j}$, and the subsequent derivations would still follow through. However, the extra terms had little effects during the experimental evaluation, and therefore the one-term version is presented for simplicity.

We will see in Section 6.6 that it is significantly easier to work with the lifted bilinear system in (6.7) than with the original nonlinear system in (6.1). To proceed, we first learn the lifted models from data.

6.5 System Identification

6.5.1 Lifted Matrix Form of Dataset

Our objective is to learn the lifted system matrices $\mathbf{A}, \mathbf{B}, \mathbf{H}, \mathbf{D}, \mathbf{Q}, \mathbf{R}$ in (6.7) from data. This learning procedure follows the approach in Section 6.5 of the previous chapter, with the addition of explicit landmark positions. We assume a dataset of the control-affine system in (6.1), including the groundtruth state transitions with their associated control inputs for P states: $\{\boldsymbol{\xi}_i^-, \boldsymbol{\xi}_i^+, \boldsymbol{\nu}_i\}_{i=1}^P$. Here, $\boldsymbol{\xi}_i^-$ transitions to $\boldsymbol{\xi}_i^+$ under input $\boldsymbol{\nu}_i$. We also assume a dataset of the associated landmark

measurements: $\{\{\boldsymbol{\gamma}_{i,j}, \boldsymbol{\psi}_{i,j}\}_{j=1}^{\beta_i}\}_{i=1}^{P}$. Here, β_i is the number of landmarks seen at timestep i, $\boldsymbol{\psi}_{i,j}$ is the position of the jth landmark at timestep i, and $\boldsymbol{\gamma}_{i,j}$ is the measurement received from landmark $\boldsymbol{\psi}_{i,j}$ at state $\boldsymbol{\xi}_i^+$. This format allows for data from multiple training trajectories to be used at once, and also allows us to combine datasets with different landmark positions. We write the data in block-matrix form:

$$\boldsymbol{\Xi}_{(-)} = \begin{bmatrix} \boldsymbol{\xi}_1^- & \cdots & \boldsymbol{\xi}_P^- \end{bmatrix}, \quad \boldsymbol{\Xi}_{(+)} = \begin{bmatrix} \boldsymbol{\xi}_1^+ & \cdots & \boldsymbol{\xi}_P^+ \end{bmatrix}, \quad \boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{\nu}_1 & \cdots & \boldsymbol{\nu}_P \end{bmatrix}, \tag{6.9a}$$

$$\boldsymbol{\Gamma} = \left[\begin{array}{ccc|c|ccc} \boldsymbol{\gamma}_{1,1} & \cdots & \boldsymbol{\gamma}_{1,\beta_1} & \cdots & \boldsymbol{\gamma}_{P,1} & \cdots & \boldsymbol{\gamma}_{P,\beta_P} \end{array}\right], \tag{6.9b}$$

$$\boldsymbol{\Psi} = \left[\begin{array}{ccc|c|ccc} \boldsymbol{\psi}_{1,1} & \cdots & \boldsymbol{\psi}_{1,\beta_1} & \cdots & \boldsymbol{\psi}_{P,1} & \cdots & \boldsymbol{\psi}_{P,\beta_P} \end{array}\right]. \tag{6.9c}$$

The data lifts to $\{\mathbf{x}_i^-, \mathbf{x}_i^+, \mathbf{u}_i\}_{i=1}^P$ and $\{\{\mathbf{y}_{i,j}, \boldsymbol{\ell}_{i,j}\}_{j=1}^{\beta_i}\}_{i=1}^P$ in the embedded space such that

$$\mathbf{x}_i^+ = \mathbf{A}\mathbf{x}_i^- + \mathbf{B}\mathbf{u}_i + \mathbf{H}\left(\mathbf{u}_i \otimes \mathbf{x}_i^-\right) + \mathbf{w}_i, \tag{6.10a}$$

$$\mathbf{y}_{i,j} = \mathbf{D}(\boldsymbol{\ell}_j \otimes \mathbf{x}_i^+) + \mathbf{n}_{i,j}, \tag{6.10b}$$

for some unknown noise, $\mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, $\mathbf{n}_{i,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. We rewrite the lifted versions of the data and the noises in block-matrix form:

$$\mathbf{X}_{(-)} = \begin{bmatrix} \mathbf{x}_1^- & \cdots & \mathbf{x}_P^- \end{bmatrix}, \quad \mathbf{X}_{(+)} = \begin{bmatrix} \mathbf{x}_1^+ & \cdots & \mathbf{x}_P^+ \end{bmatrix}, \tag{6.11a}$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_P \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_P \end{bmatrix}, \tag{6.11b}$$

$$\mathbf{Y} = \left[\begin{array}{ccc|c|ccc} \mathbf{y}_{1,1} & \cdots & \mathbf{y}_{1,\beta_1} & \cdots & \mathbf{y}_{P,1} & \cdots & \mathbf{y}_{P,\beta_P} \end{array}\right], \tag{6.11c}$$

$$\mathbf{L} = \left[\begin{array}{ccc|c|ccc} \boldsymbol{\ell}_{1,1} & \cdots & \boldsymbol{\ell}_{1,\beta_1} & \cdots & \boldsymbol{\ell}_{P,1} & \cdots & \boldsymbol{\ell}_{P,\beta_P} \end{array}\right], \tag{6.11d}$$

$$\mathbf{N} = \left[\begin{array}{ccc|c|ccc} \mathbf{n}_{1,1} & \cdots & \mathbf{n}_{1,\beta_1} & \cdots & \mathbf{n}_{P,1} & \cdots & \mathbf{n}_{P,\beta_P} \end{array}\right], \tag{6.11e}$$

$$\mathring{\mathbf{X}} = \left[\underbrace{\begin{array}{ccc} \mathbf{x}_1^+ & \cdots & \mathbf{x}_1^+ \end{array}}_{\beta_1} \middle| \cdots \middle| \underbrace{\begin{array}{ccc} \mathbf{x}_P^+ & \cdots & \mathbf{x}_P^+ \end{array}}_{\beta_P}\right], \tag{6.11f}$$

where we defined $\mathring{\mathbf{X}}$ as the states duplicated based on the number of landmarks seen at each timestep. We also define $S = \sum_{i=1}^P \beta_i$ as the total number of measurements. Then, $\mathbf{X}_{(-)}$, $\mathbf{X}_{(+)}$, $\mathbf{U}$, $\mathbf{W}$ are all P columns wide, while $\mathbf{Y}$, $\mathbf{L}$, $\mathbf{N}$, $\mathring{\mathbf{X}}$ are all S columns wide. With these definitions, the lifted matrix form of the system for (6.10) is

$$\mathbf{X}_{(+)} = \mathbf{A}\mathbf{X}_{(-)} + \mathbf{B}\mathbf{U} + \mathbf{H}\left(\mathbf{U} \odot \mathbf{X}_{(-)}\right) + \mathbf{W}, \tag{6.12a}$$

$$\mathbf{Y} = \mathbf{D}(\mathbf{L} \odot \mathring{\mathbf{X}}) + \mathbf{N}. \tag{6.12b}$$

6.5.2 Loss Function

Similar to Section 5.5.2 in the previous chapter, we optimize for the system matrices from the data by designing and minimizing a loss function, this time including the landmark positions. The model-learning problem is posed as

$$\{\mathbf{A}^\star, \mathbf{B}^\star, \mathbf{H}^\star, \mathbf{D}^\star, \mathbf{Q}^\star, \mathbf{R}^\star\} = \underset{\{\mathbf{A},\mathbf{B},\mathbf{H},\mathbf{D},\mathbf{Q},\mathbf{R}\}}{\operatorname{argmin}} V(\mathbf{A}, \mathbf{B}, \mathbf{H}, \mathbf{D}, \mathbf{Q}, \mathbf{R}), \tag{6.13}$$

where the loss function is $V = V_1 + V_2$ with

$$V_1 = \frac{1}{2}\left\|\mathbf{X}_{(+)} - \mathbf{A}\mathbf{X}_{(-)} - \mathbf{B}\mathbf{U} - \mathbf{H}(\mathbf{U} \odot \mathbf{X}_{(-)})\right\|^2_{\mathbf{Q}^{-1}} + \frac{1}{2}\left\|\mathbf{Y} - \mathbf{D}(\mathbf{L} \odot \check{\mathbf{X}})\right\|^2_{\mathbf{R}^{-1}} - \frac{1}{2}P\ln\left|\mathbf{Q}^{-1}\right| - \frac{1}{2}S\ln\left|\mathbf{R}^{-1}\right|, \tag{6.14a}$$

$$V_2 = \frac{1}{2}P\tau_A\left\|\mathbf{A}\right\|^2_{\mathbf{Q}^{-1}} + \frac{1}{2}P\tau_B\left\|\mathbf{B}\right\|^2_{\mathbf{Q}^{-1}} + \frac{1}{2}P\tau_H\left\|\mathbf{H}\right\|^2_{\mathbf{Q}^{-1}} + \frac{1}{2}S\tau_D\left\|\mathbf{D}\right\|^2_{\mathbf{R}^{-1}} + \frac{1}{2}P\tau_Q\operatorname{tr}(\mathbf{Q}^{-1}) + \frac{1}{2}S\tau_R\operatorname{tr}(\mathbf{R}^{-1}), \tag{6.14b}$$

Similar to (5.18), V_1 represents the negative log-likelihood of data, and V_2 contains prior terms. The regularizing hyperparameters, $\tau_A, \tau_B, \tau_H, \tau_D, \tau_Q, \tau_R$, can be tuned to maximize performance.

We find the critical points by setting the derivatives of V with respect to the model parameters ($\frac{\partial V}{\partial \mathbf{A}}$, $\frac{\partial V}{\partial \mathbf{B}}$, $\frac{\partial V}{\partial \mathbf{H}}$, $\frac{\partial V}{\partial \mathbf{D}}$, $\frac{\partial V}{\partial \mathbf{Q}^{-1}}$, and $\frac{\partial V}{\partial \mathbf{R}^{-1}}$) to zero. We define

$$\mathbf{V}_Q = \mathbf{U} \odot \mathbf{X}_{(-)}, \qquad \mathbf{J}_Q = \mathbf{X}_{(+)} - \mathbf{A}\mathbf{X}_{(-)} - \mathbf{B}\mathbf{U} - \mathbf{H}\mathbf{V}_Q, \tag{6.15a}$$

$$\mathbf{V}_R = \mathbf{L} \odot \check{\mathbf{X}}, \qquad \mathbf{J}_R = \mathbf{Y} - \mathbf{D}\mathbf{V}_R. \tag{6.15b}$$

This yields the following expressions that can be solved for $\mathbf{A}$, $\mathbf{B}$, $\mathbf{D}$, and $\mathbf{H}$:

$$\begin{bmatrix} \mathbf{X}_{(-)}\mathbf{X}_{(-)}^\top + P\tau_A\mathbf{1} & \mathbf{X}_{(-)}\mathbf{U}^\top & \mathbf{X}_{(-)}\mathbf{V}_Q^\top \\ \mathbf{U}\mathbf{X}_{(-)}^\top & \mathbf{U}\mathbf{U}^\top + P\tau_B\mathbf{1} & \mathbf{U}\mathbf{V}_Q^\top \\ \mathbf{V}_Q\mathbf{X}_{(-)}^\top & \mathbf{V}_Q\mathbf{U}^\top & \mathbf{V}_Q\mathbf{V}_Q^\top + P\tau_H\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{A}^\top \\ \mathbf{B}^\top \\ \mathbf{H}^\top \end{bmatrix} = \begin{bmatrix} \mathbf{X}_{(-)}\mathbf{X}_{(+)}^\top \\ \mathbf{U}\mathbf{X}_{(+)}^\top \\ \mathbf{V}_Q\mathbf{X}_{(+)}^\top \end{bmatrix}, \tag{6.16a}$$

$$\mathbf{D} = (\mathbf{Y}\mathbf{V}_R^\top)(\mathbf{V}_R\mathbf{V}_R^\top + S\tau_D\mathbf{1})^{-1}, \tag{6.16b}$$

after which we obtain

$$\mathbf{Q} = \frac{1}{P}\mathbf{J}_Q\mathbf{J}_Q^\top + \tau_A\mathbf{A}\mathbf{A}^\top + \tau_B\mathbf{B}\mathbf{B}^\top + \tau_H\mathbf{H}\mathbf{H}^\top + \tau_Q\mathbf{1}, \tag{6.16c}$$

$$\mathbf{R} = \frac{1}{S}\mathbf{J}_R\mathbf{J}_R^\top + \tau_D\mathbf{D}\mathbf{D}^\top + \tau_R\mathbf{1}. \tag{6.16d}$$

The time complexity of solving for $\mathbf{A}$, $\mathbf{B}$, $\mathbf{H}$, $\mathbf{Q}$ and $\mathbf{R}$ is $\mathcal{O}(N^3(P+S))$, which is linear in the number of training samples for fixed lifted dimension N.

6.5.3 Data Augmentation

If training data is scarce, system invariances can be exploited when known. The general system in (6.7) allows the model behavior to vary arbitrarily with the states and landmark locations, but this flexibility is often unnecessary. For example, a robot may exhibit similar motion across different regions of a room, and a range sensor may depend only on the relative configuration between the robot and landmarks. Although incorporating invariances directly into (6.7) is an interesting direction for future work, here we instead use data augmentation [97] to improve model accuracy. We generate additional samples by perturbing the data according to known invariances and include them in the dataset matrices in (6.9). See our experiments in Section 6.9 for details.

Data augmentation is particularly useful in the SLAM setting. While earlier chapters also involve bilinear interactions (e.g., between state and control input in the process model), in practice these are

easier to learn due to the limited variability of inputs and smoother system dynamics. In contrast, the SLAM measurement model depends on the joint configuration of the robot state and a potentially large number of landmarks, and is often more complex than the process model. This significantly increases the effective dimensionality of the learned mapping. As a result, substantially more data is required to achieve good generalization, making data augmentation especially beneficial for SLAM.

6.6 Unconstrained Koopman Linearization (UKL)

We now outline the procedure to set up and solve a batch estimation problem with UKL. This approach is in the same spirit as KoopSE in Chapter 5, which solves batch state estimation with fixed landmarks through unconstrained optimization. UKL generalizes KoopSE in that the formulation includes new landmark positions at test time, allowing for general localization (UKL-Loc) when the test landmarks are known but differ from those used during training, and also allowing for SLAM (UKL-SLAM) when the test landmarks are unknown. We then discuss the limitations of using the minimum-cost solution of UKL, and how it leads to the introduction of manifold constraints for CKL, presented in the next section. Although the constraints turn out to be crucial for performance, we start by outlining UKL as it serves as a basis for CKL.

6.6.1 UKL Batch Estimation Problem Setup

Having learned the system matrices from training data in Section 6.5, we now move to an environment with a new set of landmarks, $\{\boldsymbol{\psi}_j\}_{j=1}^{V}$, where we wish to solve for a sequence of states, $\{\boldsymbol{\xi}_k\}_{k=0}^{K}$, given a series of inputs, $\{\boldsymbol{\nu}_k\}_{k=1}^{K}$, and measurements, $\{\{\boldsymbol{\gamma}_{k,j}\}_{j=1}^{V}\}_{k=0}^{K}$. If the landmarks are all known, the problem is simply localization. If any landmarks are unknown, the problem is SLAM and we also wish to estimate the unknown landmarks. We do this in the lifted space, where the quantities become $\{\boldsymbol{\ell}_j\}_{j=1}^{V}$, $\{\mathbf{x}_k\}_{k=0}^{K}$, $\{\mathbf{u}_k\}_{k=1}^{K}$, and $\{\{\mathbf{y}_{k,j}\}_{j=1}^{V}\}_{k=0}^{K}$, respectively. Following a similar procedure as in Section 6.6 in the previous chapter, we simplify the bilinear process model of (6.7a) into a linear model using the inputs at test time. Observe that $\mathbf{u}_k \otimes \mathbf{x}_{k-1} = (\mathbf{u}_k \otimes \mathbf{1})\mathbf{x}_{k-1}$. Then, we define a new time-varying system matrix, $\mathbf{A}_{k-1} = \mathbf{A} + \mathbf{H}(\mathbf{u}_k \otimes \mathbf{1})$, and a new input, $\mathbf{v}_k = \mathbf{B}\mathbf{u}_k$. With this, we have a system in which the process model is linear-Gaussian:

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k, \qquad k = 1, \ldots, K, \tag{6.17a}$$

$$\mathbf{y}_{k,j} = \mathbf{D}(\boldsymbol{\ell}_j \otimes \mathbf{x}_k) + \mathbf{n}_{k,j}, \qquad k = 0, \ldots, K,\ j = 1, \ldots, V, \tag{6.17b}$$

where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\mathbf{n}_{k,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. As the input is always given at test time, this form applies to both localization and SLAM.

We now describe the process for solving UKL-SLAM, then briefly outline UKL-Loc as a special case of UKL-SLAM. A more efficient method for solving UKL-Loc using the RTS smoother [1] is described in Appendix C.1.

6.6.2 Solving UKL-SLAM

If some or all landmarks are unknown, we can solve for both the poses and the unknown landmarks by formulating and solving a batch linear SLAM problem. We first describe the case where all

landmark positions are unknown and the robot receives a measurement from every landmark at each timestep, and later describe how to modify the method to allow for variations in the problem. As is often done, we assume that $\mathbf{x}_0$ is known in order to render a unique solution to the SLAM problem. We define

$$\boldsymbol{x} = \begin{bmatrix} \mathbf{x}_1^\top & \cdots & \mathbf{x}_K^\top \end{bmatrix}^\top, \quad \boldsymbol{\ell} = \begin{bmatrix} \boldsymbol{\ell}_1^\top & \cdots & \boldsymbol{\ell}_V^\top \end{bmatrix}^\top, \quad \boldsymbol{v} = \begin{bmatrix} \mathbf{v}_1^\top & \cdots & \mathbf{v}_K^\top \end{bmatrix}^\top, \tag{6.18a}$$

$$\boldsymbol{y} = \begin{bmatrix} \mathbf{y}_1^\top & \cdots & \mathbf{y}_K^\top \end{bmatrix}^\top, \quad \mathbf{y}_k = \begin{bmatrix} \mathbf{y}_{k,1}^\top & \cdots & \mathbf{y}_{k,V}^\top \end{bmatrix}^\top, \tag{6.18b}$$

where the italic quantities $\boldsymbol{x}$, $\boldsymbol{\ell}$, $\boldsymbol{v}$, and $\boldsymbol{y}$ denote stacked batch quantities. This notation convention is used throughout the remainder of this chapter. The pose errors and measurement errors are, respectively,

$$\mathbf{e}_{\mathbf{v},k}(\mathbf{x}_k) = (\mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_k) - \mathbf{x}_k, \quad \mathbf{e}_{\mathbf{y},k,j}(\mathbf{x}_k, \boldsymbol{\ell}_j) = \mathbf{D}(\boldsymbol{\ell}_j \otimes \mathbf{x}_k) - \mathbf{y}_{k,j}. \tag{6.19}$$

We can formulate the cost function as

$$J(\boldsymbol{x}, \boldsymbol{\ell}) = \frac{1}{2}\sum_{k=1}^{K} \mathbf{e}_{\mathbf{v},k}(\mathbf{x}_k)^\top \mathbf{Q}^{-1} \mathbf{e}_{\mathbf{v},k}(\mathbf{x}_k) + \frac{1}{2}\sum_{k=1}^{K}\sum_{j=1}^{V} \mathbf{e}_{\mathbf{y},k,j}(\mathbf{x}_k, \boldsymbol{\ell}_j)^\top \mathbf{R}^{-1} \mathbf{e}_{\mathbf{y},k,j}(\mathbf{x}_k, \boldsymbol{\ell}_j), \tag{6.20}$$

or, in block-matrix form,

$$J(\boldsymbol{q}) = \frac{1}{2}\boldsymbol{e}(\boldsymbol{q})^\top \boldsymbol{W}^{-1} \boldsymbol{e}(\boldsymbol{q}), \tag{6.21}$$

where

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\ell} \end{bmatrix}, \quad \boldsymbol{W} = \begin{bmatrix} \boldsymbol{Q} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R} \end{bmatrix}, \tag{6.22a}$$

$$\boldsymbol{Q} = \operatorname{diag}(\mathbf{Q}, \ldots, \mathbf{Q}), \quad \boldsymbol{R} = \operatorname{diag}(\mathbf{R}, \ldots, \mathbf{R}), \tag{6.22b}$$

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{e}_{\boldsymbol{v}} \\ \boldsymbol{e}_{\boldsymbol{y}} \end{bmatrix}, \quad \boldsymbol{e}_{\boldsymbol{v}} = \begin{bmatrix} \mathbf{e}_{\mathbf{v},1} \\ \vdots \\ \mathbf{e}_{\mathbf{v},K} \end{bmatrix}, \quad \boldsymbol{e}_{\boldsymbol{y}} = \begin{bmatrix} \mathbf{e}_{\mathbf{y},1} \\ \vdots \\ \mathbf{e}_{\mathbf{y},K} \end{bmatrix}, \quad \mathbf{e}_{\mathbf{y},k} = \begin{bmatrix} \mathbf{e}_{\mathbf{y},k,1} \\ \vdots \\ \mathbf{e}_{\mathbf{y},k,V} \end{bmatrix}. \tag{6.22c}$$

Note that the sparsity pattern in $\boldsymbol{W}$ implicitly represents a co-visibility graph, since there are only entries corresponding to process priors among adjacent poses, or measurements from poses to visible landmarks. The UKL-SLAM optimization objective is

$$\min_{\boldsymbol{q}} \; J(\boldsymbol{q}). \tag{6.23}$$

Note that although $J(\boldsymbol{q})$ is quadratic with respect to $\boldsymbol{e}$, it is generally non-quadratic with respect to $\boldsymbol{q}$, and thus (6.23) cannot be solved in one shot. However, we can still use classical Gauss-Newton optimization to efficiently solve (6.23) iteratively by exploiting sparsity structures. See Appendix C.1.1 and Appendix C.1.2 for details on solving (6.23) with classical Gauss-Newton. The optimal update, $\delta\boldsymbol{q}$, can be computed in complexity $\mathcal{O}(N^3(V^3+V^2K))$ when we have more timesteps than landmarks, or $\mathcal{O}(N^3(K^3 + K^2V))$ when we have more landmarks than timesteps. We then

update the operating point with

$$\boldsymbol{x}_{\text{op}} \leftarrow \boldsymbol{x}_{\text{op}} + \alpha\delta\boldsymbol{x}, \quad \boldsymbol{\ell}_{\text{op}} \leftarrow \boldsymbol{\ell}_{\text{op}} + \alpha\delta\boldsymbol{\ell}, \tag{6.24}$$

for a step size α. An appropriate step size can be found using a backtracking line search [28]. Iterating until convergence yields $(\boldsymbol{x}^\star, \boldsymbol{\ell}^\star)$, or $\boldsymbol{q}^\star$ in batch form.

The distribution of the batch system's solution is $\boldsymbol{q} \sim \mathcal{N}(\hat{\boldsymbol{q}}, \hat{\boldsymbol{P}}_{\boldsymbol{q}})$, where $\hat{\boldsymbol{q}} = \begin{bmatrix} \hat{\boldsymbol{x}} \\ \hat{\boldsymbol{\ell}} \end{bmatrix}$ and $\hat{\boldsymbol{P}}_{\boldsymbol{q}} = \begin{bmatrix} \hat{\boldsymbol{P}}_{\boldsymbol{x}} & \hat{\boldsymbol{P}}_{\boldsymbol{x}\boldsymbol{\ell}} \\ \hat{\boldsymbol{P}}_{\boldsymbol{x}\boldsymbol{\ell}}^\top & \hat{\boldsymbol{P}}_{\boldsymbol{\ell}} \end{bmatrix}$. We have the batch mean from the Gauss-Newton solution: $\hat{\boldsymbol{q}} = \boldsymbol{q}^\star$. We can compute the individual distributions for $\mathbf{x}_k$ and $\boldsymbol{\ell}_j$ as follows. The means, $\hat{\mathbf{x}}_k$ and $\hat{\boldsymbol{\ell}}_j$, are simply the corresponding blocks from $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{\ell}}$. As a feature of classical Gauss-Newton SLAM, we can also compute the covariances, $\hat{\mathbf{P}}_{\mathbf{x},k}$ and $\hat{\mathbf{P}}_{\boldsymbol{\ell},j}$, without raising the computation complexity of SLAM. See Appendix C.1.2 for details on computing covariances. The final output is $\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_{\mathbf{x},k})$, $\boldsymbol{\ell}_j \sim \mathcal{N}(\hat{\boldsymbol{\ell}}_j, \hat{\mathbf{P}}_{\boldsymbol{\ell},j})$.

6.6.3 Recovering Estimates from Lifted Space

After solving for the lifted means and covariances of the states and landmarks, we now convert them back into the original space. We define the block structures of the means and covariances of states and landmarks in the original space for future reference:

$$\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{\xi}^\top & \boldsymbol{\psi}^\top \end{bmatrix}^\top, \quad \boldsymbol{\zeta} \sim \mathcal{N}(\hat{\boldsymbol{\zeta}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{\zeta}}), \quad \boldsymbol{\xi} \sim \mathcal{N}(\hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{\xi}}), \quad \boldsymbol{\psi} \sim \mathcal{N}(\hat{\boldsymbol{\psi}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{\psi}}), \tag{6.25a}$$

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\xi}_1^\top & \cdots & \boldsymbol{\xi}_K^\top \end{bmatrix}^\top, \ \boldsymbol{\psi} = \begin{bmatrix} \boldsymbol{\psi}_1^\top & \cdots & \boldsymbol{\psi}_V^\top \end{bmatrix}^\top, \ \boldsymbol{\xi}_k \sim \mathcal{N}(\hat{\boldsymbol{\xi}}_k, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{\xi},k}), \ \boldsymbol{\psi}_j \sim \mathcal{N}(\hat{\boldsymbol{\psi}}_j, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{\psi},j}). \tag{6.25b}$$

Thus, for localization, our goal is to recover the distribution of the robot trajectory, $\boldsymbol{\xi}$, while for SLAM, we seek the joint distribution of the trajectory and landmarks, $\boldsymbol{\zeta}$.

In general, recovering $\boldsymbol{\xi}_k$ from $\mathbf{x}_k$ involves a retraction step [72]. However, in Section 6.7, we will restrict the lifting functions to include the original variables as the top parts of the lifted states similar to (4.19) in Chapter 4. The recovery procedure for the states then becomes simply picking off the top part of $\hat{\mathbf{x}}_k$ to get $\hat{\boldsymbol{\xi}}_k$, and picking off the top-left block of $\hat{\mathbf{P}}_{\mathbf{x},k}$ to get $\hat{\boldsymbol{\Sigma}}_{\mathbf{x},k}$. The procedure for the landmarks is analogous.

6.6.4 Other UKL Estimation Problems

We can leverage the flexibility of classical Gauss-Newton SLAM to handle variations to UKL-SLAM, including localization and mapping as special cases. It is possible that $\mathbf{y}_{k,j}$ is missing, meaning the robot does not receive a measurement from $\boldsymbol{\ell}_j$ at timestep k. Or, it is possible that a pose, $\mathbf{x}_k$, or a landmark position, $\boldsymbol{\ell}_j$, is known. In both cases, we can modify (C.7) to incorporate missing or known variables without affecting complexity. See Appendix C.1.1 for more details. If all of the landmarks are known, SLAM reduces to localization (i.e., UKL-Loc). If all of the robot poses are known, SLAM reduces to mapping. In both of these cases, the cost in (6.21) becomes quadratic in the remaining unknowns, and using the sparse Cholesky solver (see Appendix C.1.2) yields the minimum-cost solution after just one iteration.

6.6.5 Limitations of Unconstrained Optimization in Lifted Spaces

This section discusses one of the core themes of this chapter: why UKL, and unconstrained optimization of lifted variables in general, may not be sufficient in some scenarios. In the previous chapter, we explored a similar unconstrained formulation for localization (KoopSE), where the lifted state is estimated without enforcing manifold constraints. As we have seen, KoopSE performs well when measurements are sufficiently informative. However, this behavior does not extend to scenarios with less informative measurements, nor to the SLAM setting. We show how the lack of manifold constraints can lead to very poor minimum-cost solutions in UKL, motivating the introduction of such constraints for CKL in the next section.

At first, UKL seems tempting. Solving localization and SLAM through unconstrained optimization is straightforward and the solution can be found in one shot. However, observe that there are no conditions that enforce the solution to be on the manifold defined by the lifting functions. In the case of UKL-Loc, we automatically enforce that $\mathbf{u}_k \in \mathcal{U}$, $\mathbf{y}_{k,j} \in \mathcal{Y}$, $\boldsymbol{\ell}_j \in \mathcal{L}$, as these quantities are given and are directly lifted at test time. However, there is no guarantee that the minimum-cost solution of (6.21) satisfies $\mathbf{x}_k^* \in \mathcal{X}$. Similarly, for UKL-SLAM, there is no guarantee that $\mathbf{x}_k^* \in \mathcal{X}$ and $\boldsymbol{\ell}_j^* \in \mathcal{L}$. In both cases, the minimum-cost solution could be very far from the lifting-function manifold. This can be problematic because the trained models are likely poor for governing the behavior of off-manifold states and landmarks. The states and landmarks within the training data are always on the manifold since they are being lifted from data (6.9) to their lifted versions in (6.11). If the minimum-cost solution for (6.21) happens to be far away from the manifold, the estimates may not be reflective of the training data. See Fig. 6.3 for a conceptual illustration of this manifold deviation issue.

For localization, our experiments suggest that the minimum-cost solution of UKL-Loc tends to remain sufficiently close to the manifold when measurements are frequent, as illustrated in the top row of Fig. 6.3. In the simulations in Section 6.9, we see that the UKL-Loc trajectory outputs are nearly as good as those of the model-based localizers. We hypothesize that the measurements are favoring estimates near the manifold, although the precise mechanism remains unclear. When measurements are sporadic, however, the intervals between measurement updates allow the learned process model to propagate the estimate far from the manifold. This is illustrated in the middle row of Fig. 6.3. Thus, although unconstrained Koopman localization can work in scenarios where the measurements are regular, such as the experiments in Section 5.9 in the previous chapter, it often struggles with sporadic measurements.

On the other hand, the minimum-cost solutions of UKL-SLAM tend to yield poor trajectory and landmark estimates, with both variable types deviating substantially from their respective manifolds. As illustrated in the bottom row of Fig. 6.3, pose–landmark correspondences appear to compound the deviation effect by coupling jointly optimized off-manifold poses and landmarks. This trend appears to be present regardless of the type or dimension of the lifting functions used in (6.2). Figure 6.4 visualizes UKL-Loc and UKL-SLAM outputs on a dataset with sporadic measurements.

The manifold deviation is related to a known challenge of finding Koopman-invariant subspaces for Koopman process models [67]. For a noiseless system, a Koopman-invariant subspace is such that any on-manifold state stays on the manifold after passing through the lifted process model. For our case, this means that given lifting functions $\mathbf{p}_{\boldsymbol{\xi}}$ and $\mathbf{p}_{\boldsymbol{\mu}}$, the original process model, $\boldsymbol{\xi}_k = \mathbf{f}(\boldsymbol{\xi}_{k-1}, \boldsymbol{\mu}_k)$,

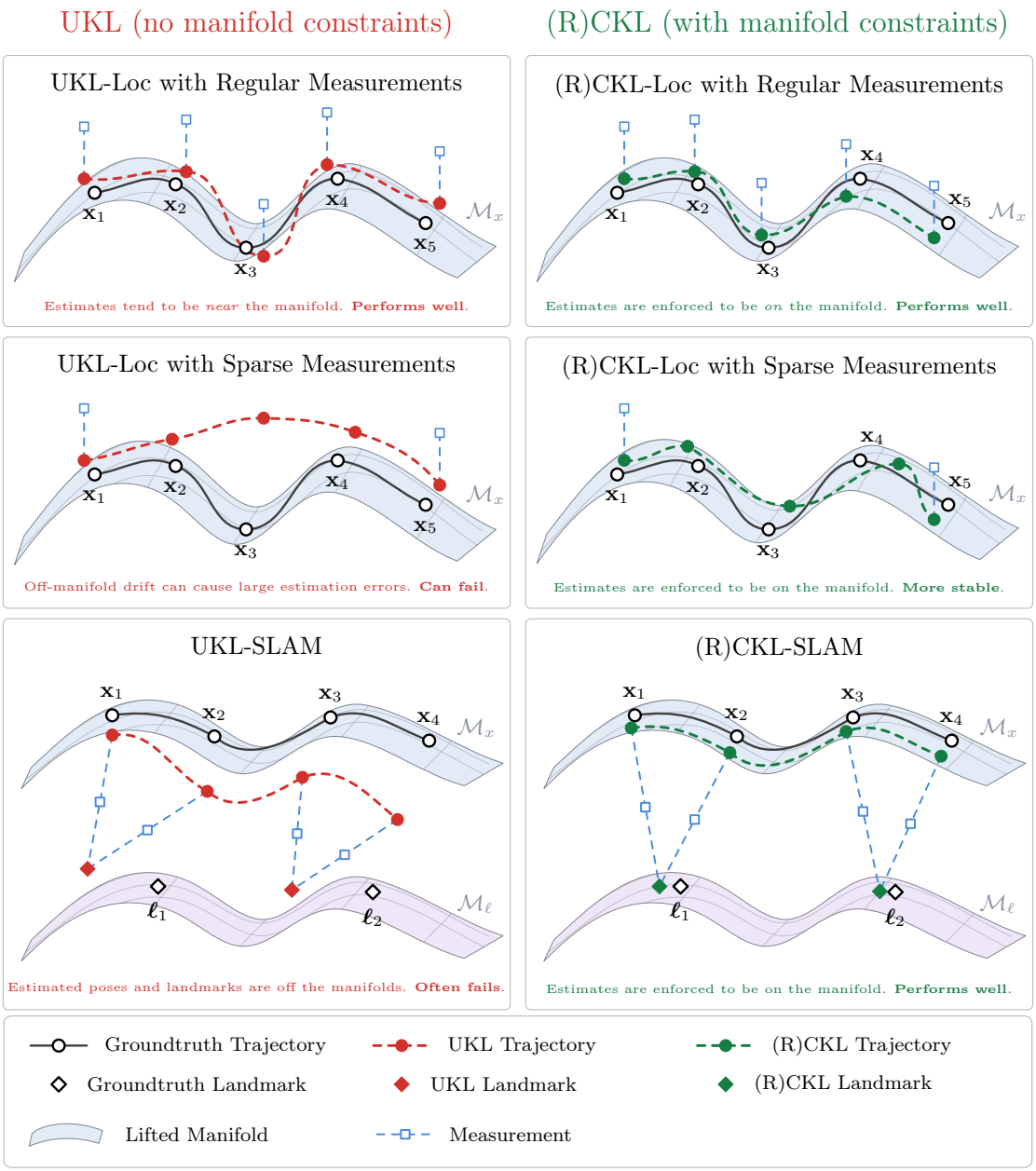


Figure 6.3: Conceptual illustration of the effects of enforcing manifold consistency in lifted localization and SLAM. The shaded surfaces represent the lifted manifolds, $\mathcal{M}_x$ and $\mathcal{M}_\ell$. Groundtruth and (R)CKL estimates lie on their corresponding manifolds, whereas UKL estimates are not constrained to do so. Manifold constraints are particularly beneficial for localization with sparse measurements and for SLAM, where poses and landmarks are jointly estimated. See Section 6.6.5 for further discussion.

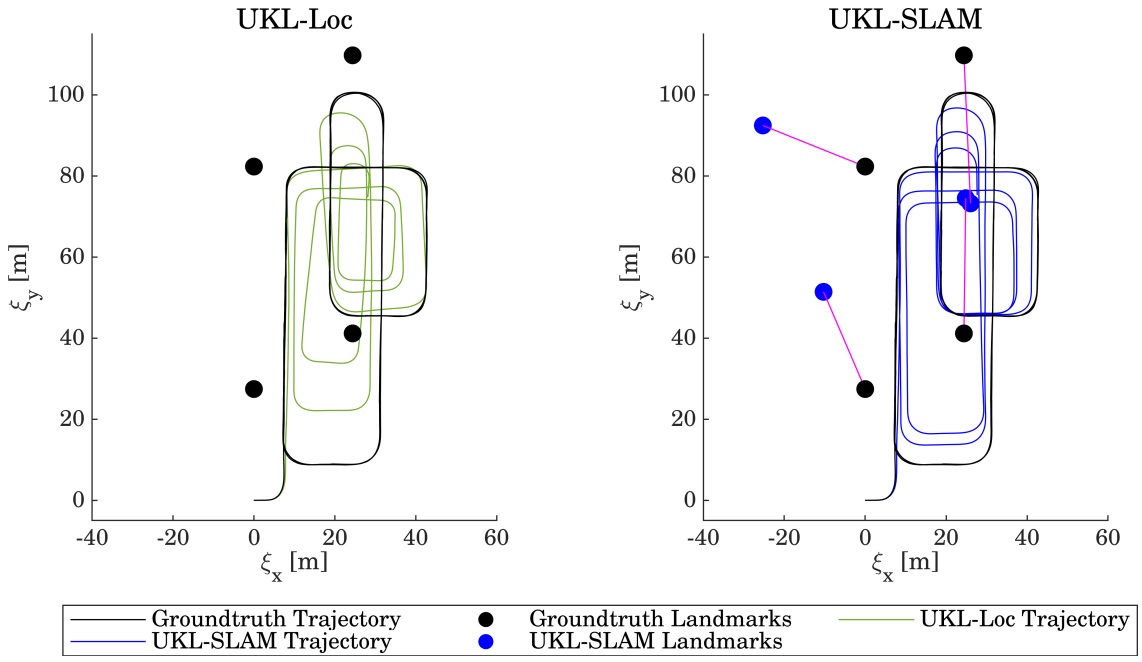


Figure 6.4: Visualization of the trajectory output of UKL-Loc (left), and the trajectory and landmark output of UKL-SLAM (right), on Experiment 2 described in Section 6.9.4, showing the estimators' mean states and mean landmark positions compared to the groundtruth. The pink lines show the landmark correspondences between the groundtruth and the output of UKL-SLAM. We observe that the trajectory estimate of UKL-Loc is poor, which we typically see when the measurements are sporadic. For UKL-SLAM, both the trajectory and the landmark estimates are poor, which we typically see regardless of the regularity of measurements.

can be written exactly as a lifted model, $\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{H}(\mathbf{u}_k \otimes \mathbf{x}_{k-1})$, such that

$$\mathbf{x}_k \in \mathcal{X} \quad \forall \mathbf{x}_{k-1} \in \mathcal{X},\ \forall \mathbf{u}_k \in \mathcal{U}. \tag{6.26}$$

Finding Koopman-invariant subspaces for real-world control problems is challenging and an active area of research [73, 74, 10, 98]. In the literature, the Koopman operator framework is commonly used for problems with short time horizons such as model-predictive control [12], where the manifold deviations are small and can be removed by a retraction step. For example, we could drop the solution back into the original space and then re-lift it at the beginning of a new time window [71, 72]. However, the solution quality deteriorates with longer time horizons [70]. In our case, ensuring invariance is further complicated with the addition of measurement models and unknown lifted landmarks. For batch optimization over long time horizons such as in SLAM, we find that reprojections cannot sufficiently improve the solution quality after converging to an off-manifold solution.

There is some work on converting nonlinear state-estimation problems into high-dimensional formulations with only linear constraints [14, 15]. However, these formulations assume noiseless systems and ideal, potentially infinite-dimensional Koopman operators, assumptions that cannot generally be satisfied in practice. They also do not address the learned lifted representations and long-horizon joint state–landmark estimation considered here. As a result, such approaches have yet to be demonstrated on real-world datasets.

In the next section, we solve the same estimation problem for the system (6.7), with the addition of constraints to enforce the solution to be on the manifold of the lifting functions.

6.7 Constrained Koopman Linearization (CKL)

In this section, we present the setup and the solution process for CKL. We present the setup of the optimization problem, the conversion to an SQP, and finally the algorithm for solving the SQP in linear time. We first focus on CKL-SLAM, where we reuse much of the development for UKL-SLAM. We then show the necessary modifications for other CKL estimation problems including dead reckoning, mapping, and localization (CKL-Loc), followed by a note on reasonable SQP initializations for these problems.

6.7.1 CKL-SLAM Problem Setup

For CKL-SLAM, the optimization objective is identical to (6.23), with the added constraint that the solution must lie on the manifold of the lifting functions. The objective is

$$\begin{aligned} \min_{\boldsymbol{q}} \quad & J(\boldsymbol{q}) \\ \text{s.t.} \quad & \mathbf{x}_k \in \mathcal{X}, \quad k = 1, \dots, K, \\ & \boldsymbol{\ell}_j \in \mathcal{L}, \quad j = 1, \dots, V, \end{aligned} \tag{6.27}$$

where $J(\boldsymbol{q})$ is defined in (6.21). To simplify the form of the constraints, we enforce that the estimated quantities' lifting functions, $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$ and $\mathbf{p}_{\boldsymbol{\ell}}(\cdot)$, consist of the original quantities stacked on top of the actual lifted features:

$$\mathbf{x}_k = \mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k) = \begin{bmatrix} \boldsymbol{\xi}_k^\top & \tilde{\mathbf{p}}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k)^\top \end{bmatrix}^\top = \begin{bmatrix} \boldsymbol{\xi}_k^\top & \tilde{\mathbf{x}}_k^\top \end{bmatrix}^\top, \tag{6.28a}$$

$$\boldsymbol{\ell}_j = \mathbf{p}_{\boldsymbol{\psi}}(\boldsymbol{\psi}_j) = \begin{bmatrix} \boldsymbol{\psi}_j^\top & \tilde{\mathbf{p}}_{\boldsymbol{\psi}}(\boldsymbol{\psi}_j)^\top \end{bmatrix}^\top = \begin{bmatrix} \boldsymbol{\psi}_j^\top & \tilde{\boldsymbol{\ell}}_j^\top \end{bmatrix}^\top, \tag{6.28b}$$

where

$$\tilde{\mathbf{p}}_{\boldsymbol{\xi}} : \mathbb{R}^{N_\xi} \to \mathbb{R}^{N_x - N_\xi}, \quad \tilde{\mathbf{p}}_{\boldsymbol{\psi}} : \mathbb{R}^{N_\psi} \to \mathbb{R}^{N_\ell - N_\psi}. \tag{6.29}$$

We can write the manifold constraints as equality constraints:

$$\mathbf{x}_k \in \mathcal{X} \;\Rightarrow\; \mathbf{h}_{\mathbf{x}}(\mathbf{x}_k) = \tilde{\mathbf{x}}_k - \tilde{\mathbf{p}}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k) = \mathbf{0}, \tag{6.30a}$$

$$\boldsymbol{\ell}_j \in \mathcal{L} \;\Rightarrow\; \mathbf{h}_{\boldsymbol{\ell}}(\boldsymbol{\ell}_j) = \tilde{\boldsymbol{\ell}}_j - \tilde{\mathbf{p}}_{\boldsymbol{\psi}}(\boldsymbol{\psi}_j) = \mathbf{0}. \tag{6.30b}$$

The optimization objective in (6.27) becomes

$$\begin{aligned} \min_{\boldsymbol{q}} \quad & J(\boldsymbol{q}) \\ \text{s.t.} \quad & \boldsymbol{h}(\boldsymbol{q}) = \mathbf{0}, \end{aligned} \tag{6.31}$$

where

$$\boldsymbol{h}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{x}) \\ \boldsymbol{h}_{\boldsymbol{\ell}}(\boldsymbol{\ell}) \end{bmatrix}, \quad \boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{x}) = \begin{bmatrix} \mathbf{h}_{\mathbf{x}}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}_{\mathbf{x}}(\mathbf{x}_K) \end{bmatrix}, \quad \boldsymbol{h}_{\boldsymbol{\ell}}(\boldsymbol{\ell}) = \begin{bmatrix} \mathbf{h}_{\boldsymbol{\ell}}(\boldsymbol{\ell}_1) \\ \vdots \\ \mathbf{h}_{\boldsymbol{\ell}}(\boldsymbol{\ell}_V) \end{bmatrix}. \tag{6.32}$$

This objective is identical to the UKL-SLAM objective (6.23) with the manifold constraints added. Note that $\boldsymbol{h}(\boldsymbol{q})$ is generally nonlinear, but it has an exploitable structure: each of the blocks in $\boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{x})$ can be nonlinear but affects only one robot state, and each of the blocks in $\boldsymbol{h}_{\boldsymbol{\ell}}(\boldsymbol{\ell})$ affects only one landmark. This blockwise structure is important for solving the optimization problem efficiently.

6.7.2 Solving CKL-SLAM with a Sequential Quadratic Program

We formulate an SQP [28] to solve (6.31). The Lagrangian is

$$L(\boldsymbol{q}, \boldsymbol{\lambda}) = J(\boldsymbol{q}) - \boldsymbol{\lambda}^\top \boldsymbol{h}(\boldsymbol{q}), \tag{6.33}$$

where $\boldsymbol{\lambda}$ is the Lagrange multiplier. Given the Lagrangian, we can solve for the SQP's optimal update, $(\delta\boldsymbol{q}, \delta\boldsymbol{\lambda})$. The solution process involves similar matrix structures as the unconstrained problem in (6.23). See Appendix C.1.3 for more details on the SQP formulation for CKL-SLAM. We then update the operating point of the primal variable and the multiplier as

$$\boldsymbol{q}_{\text{op}} \leftarrow \boldsymbol{q}_{\text{op}} + \alpha\delta\boldsymbol{q}, \quad \boldsymbol{\lambda}_{\text{op}} \leftarrow \boldsymbol{\lambda}_{\text{op}} + \alpha\delta\boldsymbol{\lambda}, \tag{6.34}$$

with an appropriate step size α. With (6.31) containing the same $J(\boldsymbol{q})$ as (6.23) and a blockwise structure on $\boldsymbol{h}(\boldsymbol{q})$, we can extend the sparsity exploitation process of UKL-SLAM to CKL-SLAM. With this, we can use the nullspace method [28] to efficiently solve the optimal update in the same time complexity as the unconstrained problem: $\mathcal{O}(N^3(V^3 + V^2K))$ when we have more timesteps than landmarks, or $\mathcal{O}(N^3(K^3 + K^2V))$ when we have more landmarks than timesteps. See Appendix C.1.4 for details.

To extract the covariances of the estimates, $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\xi},k}$ and $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\psi},j}$, we need to take into account the effect of the added constraints. We show in Appendix C.1.5 that at convergence, the batch covariance of the lifted variable is

$$\hat{\boldsymbol{P}}_{\boldsymbol{q}} = \boldsymbol{Z}(\boldsymbol{Z}^\top \boldsymbol{F} \boldsymbol{Z})^{-1} \boldsymbol{Z}^\top, \tag{6.35}$$

and the batch covariance of the original variable satisfies

$$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\zeta}}^{-1} = \boldsymbol{Z}^\top \boldsymbol{F} \boldsymbol{Z}, \tag{6.36}$$

where $\boldsymbol{F}$ is the Gauss-Newton approximation of the Hessian of the Lagrangian [43, §2], [99, §3.2], $\boldsymbol{Z} = \text{null}(\boldsymbol{S})$ is a matrix constructed by a basis of the nullspace of $\boldsymbol{S}$, and

$$\boldsymbol{S} = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\right|_{\boldsymbol{q}_{\text{op}}} \tag{6.37}$$

is the Jacobian of the constraints at the converged solution[1]. We can then use a similar procedure as for UKL-SLAM to extract the required covariances from $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\zeta}}^{-1}$. This can be done without raising the computational complexity of SLAM.

[1]Note that the definition of $\boldsymbol{S}$ here differs by a transpose from that used in the next chapter, where constraints are instead written in the form $\mathbf{S}^\top \mathbf{x} = \mathbf{c}$. The convention in this chapter is adopted to align with the standard SQP/KKT formulation, which is more natural for the derivations presented in Appendix C.1.4.

6.7.3 Other CKL Estimation Problems and SQP Initializations

CKL-SLAM can be easily modified for other estimation problems including dead reckoning, localization (CKL-Loc), and mapping. We would make similar adjustments to the cost as described in Section 6.6.4 for UKL-SLAM, and also analogous adjustments to the constraints. For all of these problems, the matrix sparsity structures are preserved, and we can still use the nullspace method to efficiently solve the SQP. See Appendix C.1.6 for details on these other problems.

In the presence of nonlinear constraints, dead reckoning, mapping, localization, and SLAM are all nonlinear optimization problems, and good initial points are often required to avoid convergence to local minima. For dead reckoning, mapping, and localization, we acquire an initial point by dropping the constraints (i.e., the UKL solution). For SLAM, we initialize the robot trajectory by dead reckoning, then initialize the landmarks by mapping from the dead-reckoned trajectory. See Appendix C.1.7 for details on the SQP initialization process.

6.8 Reduced Constrained Koopman Linearization (RCKL)

With the manifold constraints introduced, CKL-SLAM yields much better results than UKL-SLAM. However, as we will show, CKL is sensitive to poorly fit process models of the lifted features. As a result, CKL's outputs are often still less accurate than those of the model-based methods under experimental evaluation. We introduce RCKL, a framework that further improves upon CKL by removing the portion of the learned process model that tends to be poorly fit. We first present the derivation of RCKL, and then discuss our hypothesis for its improvement over CKL.

6.8.1 Reducing the Koopman Process Model

Starting with CKL, we analyze the Koopman process model in (6.7a) with the form of the lifting function, $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$, enforced in (6.28a). We break down $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{H}$ into two components:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{\boldsymbol{\xi}}^\top & \tilde{\mathbf{A}}^\top \end{bmatrix}^\top, \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_{\boldsymbol{\xi}}^\top & \tilde{\mathbf{B}}^\top \end{bmatrix}^\top, \quad \mathbf{H} = \begin{bmatrix} \mathbf{H}_{\boldsymbol{\xi}}^\top & \tilde{\mathbf{H}}^\top \end{bmatrix}^\top, \tag{6.38}$$

where

$$\mathbf{A}_{\boldsymbol{\xi}} \in \mathbb{R}^{N_\xi \times N_x}, \qquad \mathbf{B}_{\boldsymbol{\xi}} \in \mathbb{R}^{N_\xi \times N_u}, \qquad \mathbf{H}_{\boldsymbol{\xi}} \in \mathbb{R}^{N_\xi \times N_x N_u}, \tag{6.39a}$$

$$\tilde{\mathbf{A}} \in \mathbb{R}^{(N_x - N_\xi) \times N_x}, \qquad \tilde{\mathbf{B}} \in \mathbb{R}^{(N_x - N_\xi) \times N_u}, \qquad \tilde{\mathbf{H}} \in \mathbb{R}^{(N_x - N_\xi) \times N_x N_u}. \tag{6.39b}$$

Then, the process model (6.7a), along with the form of $\mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k)$ in (6.28a), yields

$$\boldsymbol{\xi}_k = \mathbf{A}_{\boldsymbol{\xi}} \mathbf{x}_{k-1} + \mathbf{B}_{\boldsymbol{\xi}} \mathbf{u}_k + \mathbf{H}_{\boldsymbol{\xi}} (\mathbf{u}_k \otimes \mathbf{x}_{k-1}) + \mathbf{w}_{\boldsymbol{\xi},k}, \tag{6.40a}$$

$$\tilde{\mathbf{x}}_k = \tilde{\mathbf{A}} \mathbf{x}_{k-1} + \tilde{\mathbf{B}} \mathbf{u}_k + \tilde{\mathbf{H}} (\mathbf{u}_k \otimes \mathbf{x}_{k-1}) + \tilde{\mathbf{w}}, \tag{6.40b}$$

where $\mathbf{w}_k = \begin{bmatrix} \mathbf{w}_{\boldsymbol{\xi},k} \\ \tilde{\mathbf{w}}_k \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{Q}_{\boldsymbol{\xi}} & \mathbf{Q}_{\boldsymbol{\xi}\mathbf{x}} \\ \mathbf{Q}_{\boldsymbol{\xi}\mathbf{x}}^\top & \mathbf{Q}_{\mathbf{x}} \end{bmatrix}\right)$.

If there are no constraints in place, both models are necessary to fully determine $\mathbf{x}_k$. With the manifold constraints introduced, however, (6.40a) alone is sufficient to determine $\mathbf{x}_k$, since the constraint $\mathbf{h}_{\mathbf{x}}(\mathbf{x}_k) = \mathbf{0}$ in (6.30a) enforces that $\tilde{\mathbf{x}}_k = \tilde{\mathbf{p}}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k)$. This means that when (6.40b) is a poor

model, we can use (6.40a) to only determine the original state variables, and let the lifting-function constraints determine the lifted features.

To clarify, the lifted features are still used in the process model of (6.40a) as part of the previous state, $\mathbf{x}_{k-1}$, but just not determined by the model for the current state. Note that reducing the model does not break the theoretical justifications mentioned in Section 6.4, including the fact that the conversion of a control-affine model to a lifted bilinear model is exact in the noiseless case.

To train the reduced process model, we train for $\mathbf{A}_{\boldsymbol{\xi}}, \mathbf{B}_{\boldsymbol{\xi}}, \mathbf{H}_{\boldsymbol{\xi}}, \mathbf{Q}_{\boldsymbol{\xi}}$. The procedure is similar to the one described in Section 6.5, except that the transitioned state is not lifted in the process model. That is, the model in (6.10) becomes

$$\boldsymbol{\xi}_i^+ = \mathbf{A}_{\boldsymbol{\xi}}\mathbf{x}_i^- + \mathbf{B}_{\boldsymbol{\xi}}\mathbf{u}_i + \mathbf{H}_{\boldsymbol{\xi}}\left(\mathbf{u}_i \otimes \mathbf{x}_i^-\right) + \mathbf{w}_i, \tag{6.41a}$$

$$\mathbf{y}_{k,j} = \mathbf{D}(\boldsymbol{\ell}_j \otimes \mathbf{x}_i^+) + \mathbf{n}_{i,j}, \tag{6.41b}$$

where $\mathbf{x}_i^+$ is replaced with $\boldsymbol{\xi}_i^+$ in the process model. After making similar modifications to the loss function in (6.14), the solutions for $\mathbf{A}_{\boldsymbol{\xi}}, \mathbf{B}_{\boldsymbol{\xi}}, \mathbf{H}_{\boldsymbol{\xi}}, \mathbf{Q}_{\boldsymbol{\xi}}$ can be found with

$$\mathbf{V}_Q = \mathbf{U} \odot \mathbf{X}_{(-)}, \quad \mathbf{J}_Q = \boldsymbol{\Xi}_{(+)} - \mathbf{A}_{\boldsymbol{\xi}}\mathbf{X}_{(-)} - \mathbf{B}_{\boldsymbol{\xi}}\mathbf{U} - \mathbf{H}_{\boldsymbol{\xi}}\mathbf{V}_Q, \tag{6.42a}$$

$$\begin{bmatrix} \mathbf{X}_{(-)}\mathbf{X}_{(-)}^\top + P\tau_A\mathbf{1} & \mathbf{X}_{(-)}\mathbf{U}^\top & \mathbf{X}_{(-)}\mathbf{V}_Q^\top \\ \mathbf{U}\mathbf{X}_{(-)}^\top & \mathbf{U}\mathbf{U}^\top + P\tau_B\mathbf{1} & \mathbf{U}\mathbf{V}_Q^\top \\ \mathbf{V}_Q\mathbf{X}_{(-)}^\top & \mathbf{V}_Q\mathbf{U}^\top & \mathbf{V}_Q\mathbf{V}_Q^\top + P\tau_H\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{\boldsymbol{\xi}}^\top \\ \mathbf{B}_{\boldsymbol{\xi}}^\top \\ \mathbf{H}_{\boldsymbol{\xi}}^\top \end{bmatrix} = \begin{bmatrix} \mathbf{X}_{(-)}\boldsymbol{\Xi}_{(+)}^\top \\ \mathbf{U}\boldsymbol{\Xi}_{(+)}^\top \\ \mathbf{V}_Q\boldsymbol{\Xi}_{(+)}^\top \end{bmatrix}, \tag{6.42b}$$

$$\mathbf{Q}_{\boldsymbol{\xi}} = \frac{1}{P}\mathbf{J}_Q\mathbf{J}_Q^\top + \tau_A\mathbf{A}_{\boldsymbol{\xi}}\mathbf{A}_{\boldsymbol{\xi}}^\top + \tau_B\mathbf{B}_{\boldsymbol{\xi}}\mathbf{B}_{\boldsymbol{\xi}}^\top + \tau_H\mathbf{H}_{\boldsymbol{\xi}}\mathbf{H}_{\boldsymbol{\xi}}^\top + \tau_Q\mathbf{1}. \tag{6.42c}$$

At test time, we modify the time-varying quantities defined in (5.23),

$$\mathbf{A}_{\boldsymbol{\xi},k-1} = \mathbf{A}_{\boldsymbol{\xi}} + \mathbf{H}_{\boldsymbol{\xi}}(\mathbf{u}_k \otimes \mathbf{1}), \quad \mathbf{v}_k = \mathbf{B}_{\boldsymbol{\xi}}\mathbf{u}_k, \tag{6.43}$$

and modify error functions in (6.19) to be

$$\mathbf{e}_{\mathbf{v},k}(\mathbf{x}_k) = (\mathbf{A}_{\boldsymbol{\xi},k-1}\mathbf{x}_{k-1} + \mathbf{v}_k) - \boldsymbol{\xi}_k, \quad \mathbf{e}_{\mathbf{y},k,j}(\mathbf{x}_k, \boldsymbol{\ell}_j) = \mathbf{D}(\boldsymbol{\ell}_j \otimes \mathbf{x}_k) - \mathbf{y}_{k,j}, \tag{6.44}$$

where the pose error is modified to weigh only the $\boldsymbol{\xi}_k$ component of $\mathbf{x}_k$, while the measurement error is still using the full set of features in $\mathbf{x}_k$. The cost function is modified from (6.20) to be

$$J(\boldsymbol{x}, \boldsymbol{\ell}) = \frac{1}{2}\sum_{k=1}^{K} \mathbf{e}_{\mathbf{v},k}(\mathbf{x}_k)^\top \mathbf{Q}_{\boldsymbol{\xi}}^{-1} \mathbf{e}_{\mathbf{v},k}(\mathbf{x}_k) + \frac{1}{2}\sum_{k=1}^{K}\sum_{j=1}^{V} \mathbf{e}_{\mathbf{y},k,j}(\mathbf{x}_k, \boldsymbol{\ell}_j)^\top \mathbf{R}^{-1} \mathbf{e}_{\mathbf{y},k,j}(\mathbf{x}_k, \boldsymbol{\ell}_j), \tag{6.45}$$

which can be written in the familiar block-matrix form of $J(\boldsymbol{q}) = \frac{1}{2}\boldsymbol{e}(\boldsymbol{q})^\top \boldsymbol{W}^{-1}\boldsymbol{e}(\boldsymbol{q})$ in (6.21). This means that the RCKL-SLAM objective has the same form as the CKL-SLAM objective in (6.32). To solve for $\boldsymbol{q}$, we can follow the same procedure as in Section 6.7.2. The SQP iterations have the same complexity, possibly with a smaller constant factor owing to the reduced process model. See Algorithm 3 for a summary of the training and testing procedures for RCKL-Loc and RCKL-SLAM.

Algorithm 3: Reduced Constrained Koopman-Linearized (RCKL) Localization and SLAM.

Training:

Input: Training data: process model data $\{\boldsymbol{\xi}_i^-, \boldsymbol{\xi}_i^+, \boldsymbol{\nu}_i\}_{i=1}^P$ and measurements with associated landmark positions $\{\{\boldsymbol{\gamma}_{i,j}, \boldsymbol{\psi}_{i,j}\}_{j=1}^{\beta_i}\}_{i=1}^P$; lifting functions $\{\mathbf{p}_{\boldsymbol{\xi}}(\cdot), \mathbf{p}_{\boldsymbol{\nu}}(\cdot), \mathbf{p}_{\boldsymbol{\psi}}(\cdot), \mathbf{p}_{\boldsymbol{\gamma}}(\cdot)\}$; hyperparameters $\{\tau_A, \tau_B, \tau_H, \tau_D, \tau_Q, \tau_R\}$.

1. Stack the training data into block-matrix form (6.9), yielding $\{\boldsymbol{\Xi}_{(-)}, \boldsymbol{\Xi}_{(+)}, \boldsymbol{\Upsilon}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda}\}$.
2. Lift the training data using the chosen embeddings (6.11), yielding $\{\mathbf{X}_{(-)}, \mathbf{X}_{(+)}, \mathbf{U}, \mathbf{Y}, \mathbf{L}, \check{\check{\mathbf{X}}}\}$.
3. Learn the reduced process model $\{\mathbf{A}_{\boldsymbol{\xi}}, \mathbf{B}_{\boldsymbol{\xi}}, \mathbf{H}_{\boldsymbol{\xi}}, \mathbf{Q}_{\boldsymbol{\xi}}\}$ and the measurement model $\{\mathbf{D}, \mathbf{R}\}$ by solving (6.42).

Output: Reduced process model $\{\mathbf{A}_{\boldsymbol{\xi}}, \mathbf{B}_{\boldsymbol{\xi}}, \mathbf{H}_{\boldsymbol{\xi}}, \mathbf{Q}_{\boldsymbol{\xi}}\}$; measurement model $\{\mathbf{D}, \mathbf{R}\}$.

Testing:

Input: Initial estimate or initialization procedure; control inputs $\{\boldsymbol{\nu}_k\}_{k=1}^K$; measurements $\{\{\boldsymbol{\gamma}_{k,j}\}_{j=1}^V\}_{k=0}^K$; landmark positions $\{\boldsymbol{\psi}_{k,j}\}_{j=1}^V$ which are either known values for localization or unknown variables for SLAM.

1. Lift the given inputs and measurements, yielding $\{\mathbf{u}_k\}_{k=1}^K$ and $\{\{\mathbf{y}_{k,j}\}_{j=1}^V\}_{k=0}^K$. If a landmark position, $\boldsymbol{\psi}_j$, is known, lift it to obtain $\boldsymbol{\ell}_j$.
2. Initialize an SQP with $\{\boldsymbol{q}^{(0)}, \boldsymbol{\lambda}^{(0)}\}$, where the primal variable, $\boldsymbol{q}^{(0)}$, is an initial estimate consisting of the lifted robot poses and any unknown lifted landmarks, using the procedure described in Section 6.7.3.
3. Repeat until $\boldsymbol{q}^{(i)}$ converges:
   (a) Form the reduced time-varying process quantities $\mathbf{A}_{\boldsymbol{\xi},k-1}$ and $\mathbf{v}_k$ using (6.43), and construct the residuals $\mathbf{e}_{\mathbf{v},k}(\mathbf{x}_k)$ and $\mathbf{e}_{\mathbf{y},k,j}(\mathbf{x}_k, \boldsymbol{\ell}_j)$ using (6.44).
   (b) Linearize the manifold constraints $\boldsymbol{h}(\boldsymbol{q}) = \mathbf{0}$ at $\boldsymbol{q}^{(i)}$, yielding the constraint Jacobian $\boldsymbol{S} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}|_{\boldsymbol{q}=\boldsymbol{q}^{(i)}}$.
   (c) Solve the SQP update using the nullspace method to obtain $\{\delta\boldsymbol{q}, \delta\boldsymbol{\lambda}\}$, where $\delta\boldsymbol{q} = \begin{bmatrix}\delta\boldsymbol{x}^\top & \delta\boldsymbol{\ell}^\top\end{bmatrix}^\top$ contains the lifted pose and landmark updates, and $\delta\boldsymbol{\lambda}$ is the Lagrange multiplier update. See Appendix C.1.3 and Appendix C.1.4 for details on solving for the update in linear time.
   (d) Update $\{\boldsymbol{q}^{(i)}, \boldsymbol{\lambda}^{(i)}\}$ using $\{\delta\boldsymbol{q}, \delta\boldsymbol{\lambda}\}$ via (6.34), yielding $\{\boldsymbol{q}^{(i+1)}, \boldsymbol{\lambda}^{(i+1)}\}$.
4. Extract $\{\hat{\boldsymbol{\xi}}_k\}_{k=0}^K$ and $\{\hat{\boldsymbol{\psi}}_j\}_{j=1}^V$, the original pose and landmark means, by picking off the appropriate entries from the converged lifted mean $\hat{\boldsymbol{q}} = \begin{bmatrix}\hat{\boldsymbol{x}}^\top & \hat{\boldsymbol{\ell}}^\top\end{bmatrix}^\top$.
5. Compute the required covariances using (6.36).

Output: State estimates $\{\hat{\boldsymbol{\xi}}_k\}_{k=0}^K$; landmark estimates $\{\hat{\boldsymbol{\psi}}_j\}_{j=1}^V$ when applicable; associated covariance estimates.

6.8.2 Theoretical Justification of RCKL

For the experimental scenarios presented in Section 6.9, RCKL's process model always performs better than CKL's model. See Fig. 6.5 for a visual example of noiseless dead reckoning with UKL,

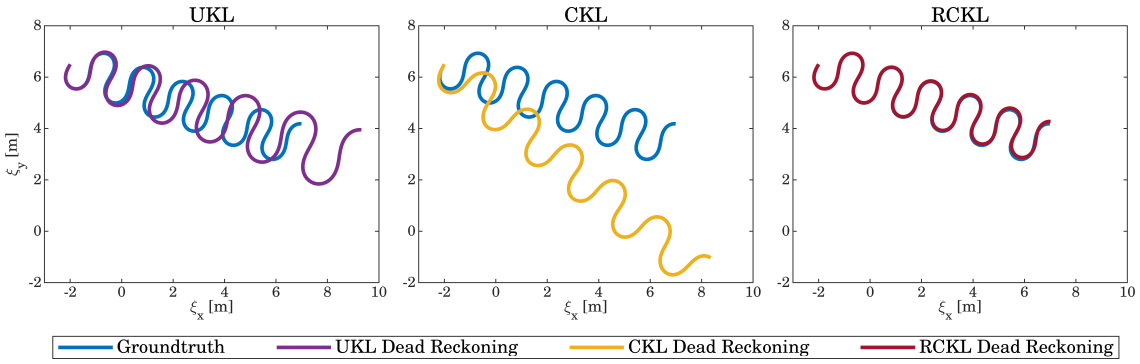


Figure 6.5: *UKL vs. CKL vs. RCKL in noiseless dead reckoning.* We train the three estimators on noiseless data corresponding to Simulation 1 described in Section 6.9.1, then compare their dead-reckoning outputs on a noiseless test trajectory. UKL's output is on top of groundtruth at first but eventually drifts off as the states deviate from the feature manifold. CKL's output remains on the feature manifold, but solution is worsened by the poor process models on the features. Meanwhile, RCKL stays on the manifold and its output corresponds to the groundtruth almost exactly.

CKL, and RCKL, where RCKL's output is much closer to the groundtruth trajectory. We now discuss our hypothesis for this improvement in performance.

The main difference between RCKL and CKL is that RCKL avoids fitting a model for the evolution of the lifted features. A poor process model of the features would negatively affect the whole system through the constraints. In our experiments, we see that when a model contains all of the lifting functions required to fit the system, adding extraneous lifting functions to CKL degrades its performance. Even when $\tilde{\mathbf{x}}_{k-1}$ is essential for determining $\boldsymbol{\xi}_k$ in (6.40a), its own evolution to $\tilde{\mathbf{x}}_k$ in (6.40b) may be poor, which would affect $\boldsymbol{\xi}_k$ through the constraint $\mathbf{h}_{\mathbf{x}}(\mathbf{x}_k) = \tilde{\mathbf{x}}_k - \tilde{\mathbf{p}}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k) = \mathbf{0}$.

This concept is related to how Koopman invariance (6.26) can be difficult to satisfy especially for the process models of the lifted features. See [67, p. 263] for an example of where the Koopman representation of even a simple system becomes intractable when using polynomial features. The higher-order polynomials added require even higher-order polynomials to fit their own process model, ad infinitum.

After selecting the features from a large pool of possible lifting functions, we find that the linear process models on the features tend to be poorly fit. RCKL, in contrast, fits a linear model on only the original variables and relies on the nonlinear manifold constraints for the evolution of the features. Assuming that we have enough training data to avoid overfitting, adding more lifting functions in RCKL would only add more expressiveness for fitting the model.

Note that RCKL's reduced model follows the same spirit as the original SINDy algorithm [68]. For a nonlinear system, $\boldsymbol{\xi}_k = \mathbf{f}(\boldsymbol{\xi}_{k-1})$, SINDy optimizes for a sparse $\mathbf{A}$ and a low-dimensional $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$ such that the model can be written as $\boldsymbol{\xi}_k = \mathbf{A}\mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_{k-1})$. That is, SINDy identifies the nonlinear functions in $\mathbf{p}_{\boldsymbol{\xi}}(\cdot)$ necessary to reconstruct the original system. This form is similar to our reduced process model in (6.41a) with $\mathbf{p}_{\boldsymbol{\xi}}$ as the lifting function. However, SINDy's nonlinear model is not originally compatible with Koopman methods (without constraints). Instead, many Koopman methods use a modified version of SINDy to reconstruct $\mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k) = \mathbf{A}\mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_{k-1})$ [77], which is essentially finding the full process model. With the manifold constraints introduced, the reduced model is once again possible for Koopman systems.

One potential drawback of RCKL is that for some systems, the process model of some features

could be simpler to construct than that of the original variables. In this case, a hybrid method could be used. As this is likely problem-specific, we leave the investigation of this idea for future work.

6.9 Experiments and Results

We evaluate the performance of the Koopman estimators for localization and SLAM in two simulation scenarios and on two real-world datasets. We investigate the performance of UKL, CKL, and RCKL in simulation, and focus on RCKL on the datasets, owing to its superior performance in simulation. We compare our results with those of a classical model-based nonlinear batch estimator optimized using Gauss-Newton [1, §8/9]. We term this estimator "MB" for model-based. To demonstrate the advantages of our data-driven approach, we also compare with a classical estimator where the model parameters are imperfect, and we term this estimator "MBI" for model-based imperfect. We will show that our Koopman estimators can outperform MBI by learning the model parameters. Both the Koopman estimators and the model-based estimators follow the initialization procedure outlined in Section 6.7.3: initializing with dead reckoning for localization, and initializing with mapping from dead reckoning for SLAM.

To evaluate accuracy, we compute the RMSE of an estimator's mean output. To evaluate consistency, we compute the normalized trajectory-level Mahalanobis distance of an estimator's mean and covariance outputs. This metric measures the consistency of the entire batch solution and is defined as

$$d_{\text{traj}} = \sqrt{(\delta\boldsymbol{\zeta})^{\top}\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\zeta}}^{-1}(\delta\boldsymbol{\zeta})/N}, \tag{6.46}$$

where $\delta\boldsymbol{\zeta}$ is the estimation error with respect to the groundtruth, $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\zeta}}^{-1}$ is the estimated information matrix, and N is the number of degrees of freedom in the system. In contrast to the step-wise Mahalanobis distance in (4.28) used in the previous two chapters, here $\delta\boldsymbol{\zeta}$, $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\zeta}}^{-1}$, and N correspond to the entire trajectory and map jointly. This distinction is particularly important in SLAM, where correlations between the trajectory and landmarks are significant, and the consistency of the full system is of interest. As mentioned in Section 6.7.2, we can exploit sparsity to efficiently compute $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\zeta}}^{-1}$ even for our large-scale batch problem.

For evaluating SLAM, we first perform an alignment step where the groundtruth is aligned to the output trajectory through a rigid transformation before computing the RMSEs and the Mahalanobis distances. This is because SLAM estimates a relative map of the trajectory and landmarks, and the global SLAM error within the frame of the first pose is usually of lesser interest.

In the experiments below, we once again use SERFFs [84] for their experimentally determined high performance [63] and, as seen in Chapter 5, for their connection to the squared-exponential kernel. See (4.21) for the formulas to generate SERFFs. The SERFF hyperparameters can be tuned based on data if necessary. Below, we use the notation $\mathbf{z}_{\mathbf{x}}(\mathbf{x})$ to denote the SERFFs for input $\mathbf{x}$.

6.9.1 Simulation Setup

For both simulation scenarios, the setup consists of a robot driving in a 2D plane, receiving measurements from landmarks scattered around the plane. We evaluate localization and SLAM on 100 test instances, each with a 1000-timestep trajectory and 10 landmarks. The measurements are scaled

by a constant factor of $\mu = 1.05$, which our Koopman estimators learn from data. We compare the performance of the Koopman estimators against MB, whose measurement model uses the correct value of $\mu = 1.05$, and MBI, whose measurement model uses $\mu = 1$. To clarify, MB and MBI are using the same raw data, but they are doing estimation with slightly different prior measurement models, thus affecting their solutions. See [1, §8/9] for how the prior measurement models are used for computing costs and Jacobians in classical Gauss-Newton estimation.

Simulation 1: Unicycle Model, Range Measurements

The inputs for the unicycle model [100] are the translational and rotation speeds of the robot, and the measurements are distances. For timestep k and landmark j, the robot state, input, landmark position, and measurement are, respectively,

$$\boldsymbol{\xi}_k = \begin{bmatrix} \xi_{x,k} & \xi_{y,k} & \xi_{\theta,k} \end{bmatrix}^\top, \quad \boldsymbol{\nu}_k = \begin{bmatrix} u_k & \omega_k \end{bmatrix}^\top, \quad \boldsymbol{\psi}_j = \begin{bmatrix} \psi_{x,j} & \psi_{y,j} \end{bmatrix}^\top, \quad \gamma_{k,j} = \mu r_{k,j}, \tag{6.47}$$

where $(\xi_{x,k}, \xi_{y,k})$ is the robot's global position, $\xi_{\theta,k}$ is the robot's global orientation, u_k is the robot's linear speed, ω_k is its angular speed, $(\psi_{x,j}, \psi_{y,j})$ is the landmark's global position, $r_{k,j}$ is the range measurement from the robot to the jth landmark, and μ is the constant scaling factor. Contrary to the assumption made in (6.28), the state is not within a vector space since it includes orientation as a circular quantity. As such, we convert it to an augmented state, $\boldsymbol{\xi}_k^{'}$, with the substitutions $\xi_{\cos\theta,k} = \cos\xi_{\theta,k}$ and $\xi_{\sin\theta,k} = \sin\xi_{\theta,k}$. For CKL and RCKL, we introduce an additional state constraint, $h_\xi(\boldsymbol{\xi}_k') = \xi_{\cos\theta,k}^2 + \xi_{\sin\theta,k}^2 - 1 = 0$. See Appendix C.1.8 for more details on handling the orientation. We used SERFFs for the state, the landmark position, the measurements, and input:

$$\boldsymbol{\xi}_k' = \begin{bmatrix} \xi_{x,k} \\ \xi_{y,k} \\ \xi_{\cos\theta,k} \\ \xi_{\sin\theta,k} \end{bmatrix}, \quad \mathbf{p}_{\boldsymbol{\xi}'}(\boldsymbol{\xi}_k^{'}) = \begin{bmatrix} \boldsymbol{\xi}_k^{'} \\ \mathbf{z}_{x,y}(\xi_{x,k}, \xi_{y,k}) \\ \mathbf{z}_{\cos\theta,\sin\theta}(\xi_{\cos\theta,k}, \xi_{\sin\theta,k}) \\ \mathbf{z}_{x,\cos\theta}(\xi_{x,k}, \xi_{\cos\theta,k}) \\ \mathbf{z}_{x,\sin\theta}(\xi_{x,k}, \xi_{\sin\theta,k}) \\ \mathbf{z}_{y,\cos\theta}(\xi_{y,k}, \xi_{\cos\theta,k}) \\ \mathbf{z}_{y,\sin\theta}(\xi_{y,k}, \xi_{\sin\theta,k}) \end{bmatrix}, \tag{6.48a}$$

$$\mathbf{p}_{\boldsymbol{\psi}}(\boldsymbol{\psi}_j) = \begin{bmatrix} \boldsymbol{\psi}_j \\ \mathbf{z}_{\boldsymbol{\psi}}(\boldsymbol{\psi}_j) \end{bmatrix}, \quad \mathbf{p}_\gamma(\gamma_{k,j}) = \begin{bmatrix} \gamma_{k,j} \\ \mathbf{z}_\gamma(\gamma_{k,j}) \end{bmatrix}, \quad \mathbf{p}_{\boldsymbol{\nu}}(\boldsymbol{\nu}_k) = \begin{bmatrix} \boldsymbol{\nu}_k \\ \mathbf{z}_{\boldsymbol{\nu}}(\boldsymbol{\nu}_k) \end{bmatrix}, \tag{6.48b}$$

where for the state, SERFFs are constructed on all possible pairs of variables in $\boldsymbol{\xi}_k'$ rather than constructing $\mathbf{z}_{\boldsymbol{\xi}'}(\xi_{x,k}, \xi_{y,k}, \xi_{\cos\theta,k}, \xi_{\sin\theta,k})$. We find this form to be sufficient for our environment, avoiding the burden of a much higher-dimensional lifted state and the need for substantially more training data to prevent overfitting.

Simulation 2: Bicycle Model, Range-Squared Measurements

The setup is the same as for Simulation 1, except that the robot has a bicycle process model [101] and the measurement is the squared range (to show the generalizability of our data-driven methods):

$$\boldsymbol{\nu}_k = \begin{bmatrix} u_k & \phi_k \end{bmatrix}^\top, \quad \gamma_{k,j} = \mu r_{k,j}^2, \tag{6.49}$$

where u_k is still the linear speed but ϕ_k is the steering angle of the bicycle's front axle. The lifting functions used are in the same form as in (6.48) except for the input, for which we account for the circular input of ϕ_k by using

$$\mathbf{p}_{\boldsymbol{\nu}}(\boldsymbol{\nu}_k) = \begin{bmatrix} u_k & \cos\phi_k & \sin\phi_k & \mathbf{z}_{u,\cos\phi,\sin\phi}(u_k, \cos\phi_k, \sin\phi_k)^\top \end{bmatrix}^\top. \tag{6.50}$$

6.9.2 Simulation Results Discussion

The results for Simulation 1 and Simulation 2 are shown in Fig. 6.6. We see that MB tends to have the lowest errors for localization and SLAM, whereas RCKL has similar or slightly higher errors. This outcome is expected since the models and the noise distributions are perfectly known for MB, whereas the Koopman estimators have learned the model through noisy data. However, RCKL has similar or better accuracy compared to MBI, demonstrating the advantages of the data-driven approach. In addition, while the model-based estimators, MB and MBI, encounter local minima in both localization and SLAM, RCKL appears to encounter local minima much less frequently. This suggests that another potential benefit of reformulating nonlinear estimation problems in a lifted space is improved convergence properties.

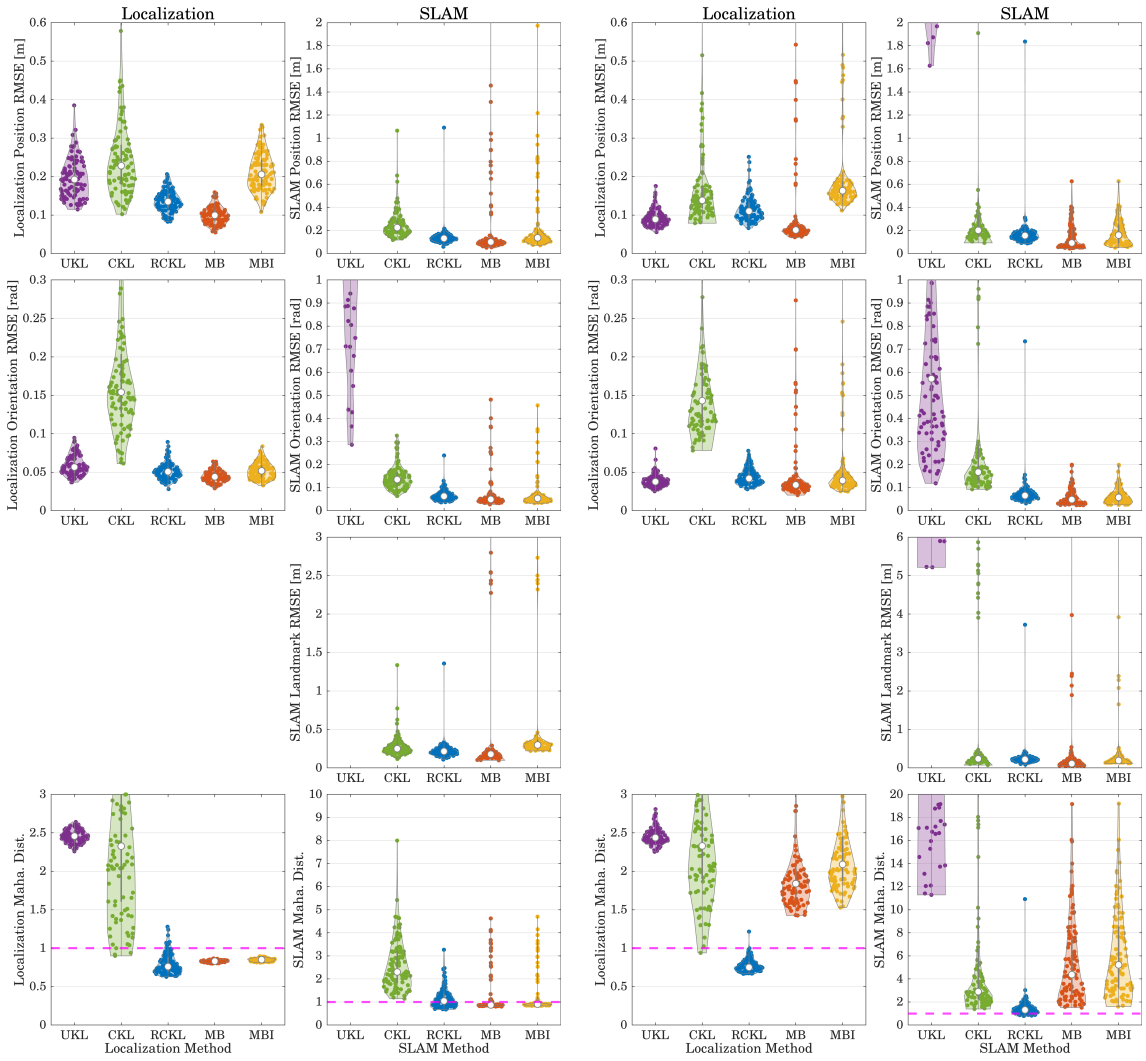
To further understand the observed convergence behavior, we verify the optimality of our solutions by running each optimization problem twice: the first time using the standard initialization procedure described in Section 6.7.3, and the second time initializing at groundtruth. Through this procedure, we can identify the presence of local minima when the first solution (standard initialization) has a higher cost than the second solution (groundtruth initialization). Global optimality is harder to verify, since computing globally optimal estimates is typically prohibitively expensive even for moderately sized problems. In our case, we assume that the groundtruth initialization converges near the global minimum for the considered noise levels. Under this assumption, a global minimum has likely been reached when both runs produce the same cost[2]. This procedure allows us to systematically assess the presence of local minima, supporting our observations regarding the improved convergence behavior of RCKL over MB and MBI.

For the other Koopman estimators, we see that UKL-Loc is viable in our environment where a measurement is received from every landmark at every timestep. However, UKL-SLAM is not at all viable, and only becomes viable when the constraints are added to yield CKL-SLAM. CKL still has higher errors than the model-based estimators, MB and MBI. It becomes comparable to MB only for RCKL, that is, after the reduced process model is employed.

6.9.3 Experiment 1: Indoor Navigation with Laser Rangefinder

We now move on to evaluation with datasets from real-world robots. For Experiment 1, we use the "Lost in the Woods" dataset in [32]. The setup consists of a wheeled robot driving around in a 2D indoor environment, with 17 tubes scattered throughout the space to act as landmarks (see Fig. 6.7). The robot has an odometer to measure its translational and rotational speed, and uses a laser rangefinder to measure its range to the cylindrical landmarks. The groundtruth positions of the robot are recorded using a Vicon motion capture system. All data are logged at 10 Hz. There are approximately 4 visible landmarks at any given time. The robot state, input, landmark position,

[2]For future work, one could incorporate a global optimality certificate [102] for either MB or RCKL if desired.



(a) Simulation 1 Results

(b) Simulation 2 Results

Figure 6.6: *Localization and SLAM results for Simulation 1 (left) and Simulation 2 (right).* RMSE and Mahalanobis distances for localization (left plots) and SLAM (right plots) for UKL, CKL, RCKL, MB (model-based with correct μ), and MBI (model-based with imperfect μ). In both simulations, RCKL has slightly higher RMSEs than MB but encounters local minima less frequently, and it has lower RMSEs than MBI. UKL-SLAM RMSEs are off the charts, and CKL-SLAM RMSEs are more reasonable than UKL-SLAM but still higher than RCKL-SLAM, demonstrating the necessity of the constraints and the reduced process model in RCKL. RCKL is also more consistent than MB and MBI.

and measurement are in the same form as in (6.47), where we adopt the common practice of using interoceptive measurements as inputs in the process model [22]. We use the lifting functions given in (6.48).

To empirically evaluate RCKL, we split the entire 20-minute dataset into training data and testing data. We use $5/6$ of the trajectories for training, where measurements derived from only 11 of the 17 landmarks are used to train the measurement model. We test on the remaining $1/6$ trajectories using measurements derived from only the six remaining landmarks. We assume that there are no factors such as slippage that would cause the robot to drive differently in one area of the room compared to another, and that the measurements are dependent only on the landmark

Figure 6.7: Setup for Experiment 1. A wheeled robot drives around 17 cylindrical landmarks (2 are not visible in this photo) in an indoor environment. It logs wheel odometry and measures the range to its surrounding landmarks using a laser rangefinder. The landmarks for testing are highlighted in red.

positions relative to the robot poses. Thus, we perform data augmentation by adding translational and rotational transformations of the original training data to the training set.

We do 6-fold cross-validation by repeating the training/testing procedure for different sections of the dataset's trajectories, and compare the results of RCKL to MB. To demonstrate the advantages of the model-free framework, we also compare these results with MBI, whose robot-to-rangefinder distance in the measurement model is offset by 10 cm with respect to the generated measurements. Again, MB and MBI are using the same raw measurements but different prior models. The errors for RCKL, MB, and MBI are shown in Fig. 6.8, and their respective Mahalanobis distances are shown in Fig. 6.9.

In Fig. 6.8, we see that for both localization and SLAM, RCKL has similar error levels as MB but outperforms MBI. The Mahalanobis distances in Fig. 6.9 show that RCKL is fairly consistent. Meanwhile, the model-based estimators are overconfident, and even more overconfident with the model offset. These results suggest that RCKL has learned a more accurate and more consistent model than MB and especially MBI. While RCKL appears to have not encountered any local minima in localization or SLAM, the model-based estimators encountered local minima in one of the six folds for localization and two of the six folds for SLAM (see Fig. 6.10). This behavior suggests that the lifted formulation of RCKL may provide improved convergence properties for these estimation problems. One possible explanation is that the lifted optimization landscape is locally easier to optimize than the original nonlinear formulation. We leave the investigation of this hypothesis for future work.

6.9.4 Experiment 2: Golf Cart with RFID Measurements

For this experiment, we train on Dataset A3 and test on 3000 steps of Dataset A1 of [103]. The setup for these two datasets consists of a cart driving on a golf course. The robot is equipped with a transponder with four radio-frequency (RF) antennae mounted on the four corners of the robot. As it moves, it measures ranges to RF tags placed on top of ten traffic cones that are scattered around the course. The range measurements are sporadic, occurring only at certain points on the trajectory.

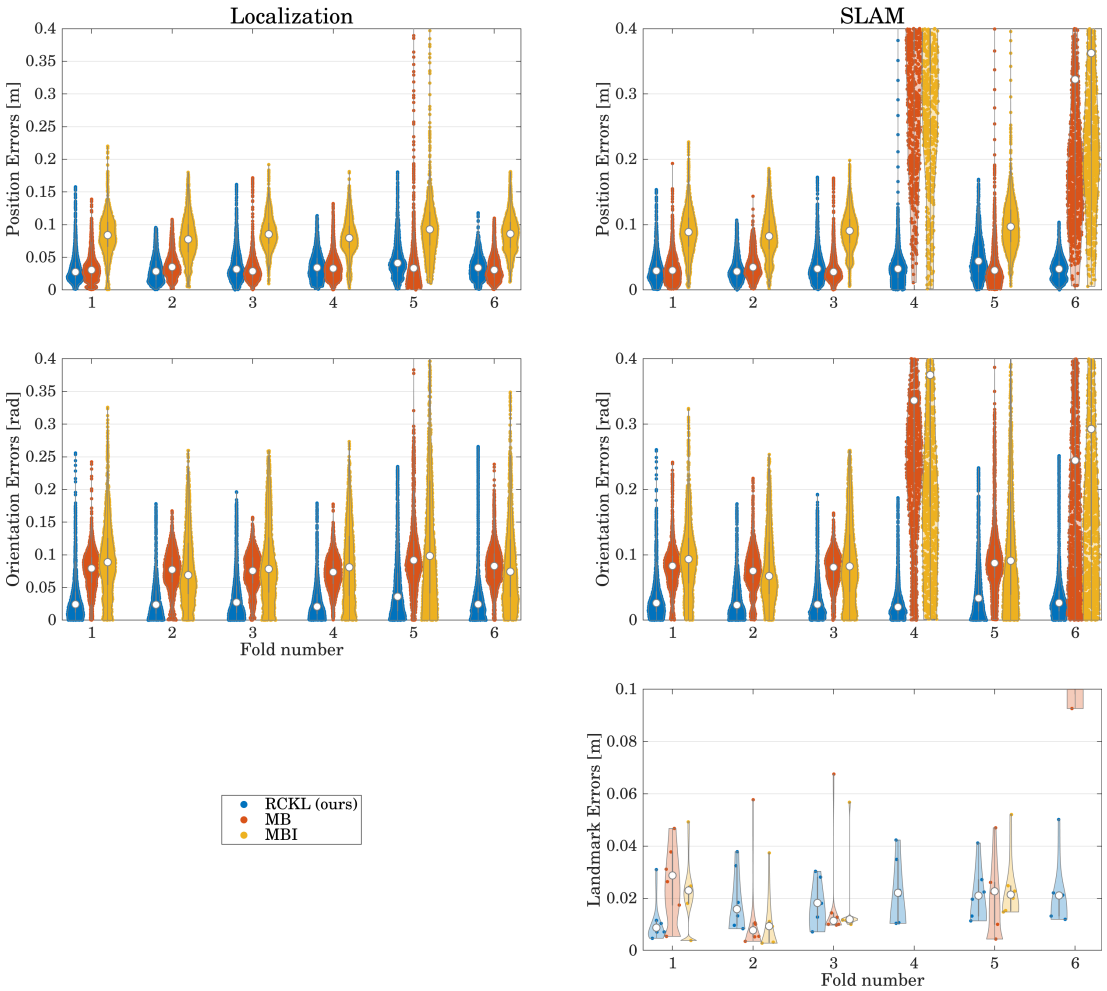


Figure 6.8: *Localization and SLAM errors for Experiment 1*: position and orientation errors of the mean trajectory and the mean landmark outputs of RCKL, MB, and MBI (whose robot-to-rangefinder distance is offset by 10 cm), on the six dataset folds. For localization, RCKL has similar position errors and generally lower orientation errors compared to MB. Against MBI, RCKL has lower position and orientation errors, demonstrating the advantage of the data-driven approach. These trends are also present for SLAM on folds 1, 2, 3, and 5. On folds 4 and 6, the model-based SLAM errors are much higher as a result of convergence to local minima (see Fig. 6.10 for a visualization).

The measurements for each transponder contain a rather large scaling factor, corresponding to μ in (6.47), of around 1.2. The groundtruth positions of the robot and of the cones are obtained using a Global Positioning System (GPS). The control input consists of linear and angular velocities obtained from a fiber-optic gyro and wheel encoders. With just dead reckoning, the test trajectory drifts in orientation owing to repeated turns in the same direction [103]. This drift will be corrected by localization or SLAM.

Similar to Experiment 1, the robot state, input, landmark position, and measurement are in the same form as in (6.47), and the same forms of lifting functions are used as in (6.48). The landmark locations in A1 and A3 are the same. As such, we train on A3 with measurements from only 5 landmarks, and test on A1 with measurements from the 4 remaining landmarks, ignoring the one landmark with no measurements. We treat each RFID tag as an independent measurement model, so there are three sets of $\{\mathbf{D}, \mathbf{R}\}$ to train, ignoring the one tag with only two measurements. We use

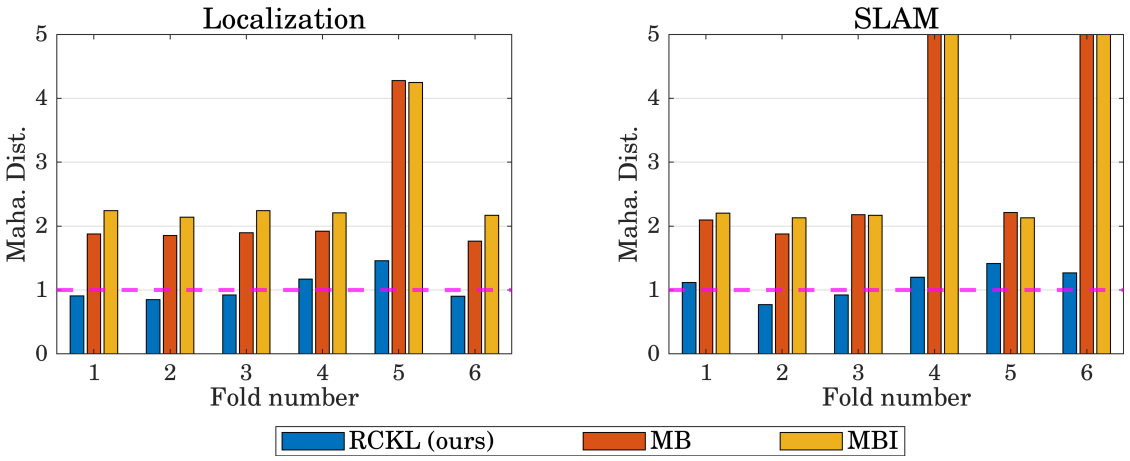


Figure 6.9: *Mahalanobis distances of localization and SLAM for Experiment 1*: Mahalanobis distances of the outputs of RCKL, MB, and MBI (whose robot-to-rangefinder distance is offset by 10 cm), on the six dataset folds. For localization, RCKL's Mahalanobis distances are close to 1, signifying that RCKL-Loc is consistent. MB's Mahalanobis distances are slightly higher than 1, signifying that its estimates are slightly overconfident. MBI is similarly, if not marginally more, overconfident than MB. For localization, MB and MBI are more overconfident in fold 5 than in the other folds as a result of converging to local minima. These trends are also present for SLAM on folds 1, 2, 3, and 5. On folds 4 and 6, the estimates of the model-based estimators, MB and MBI, are highly overconfident as a result of convergence to local minima.

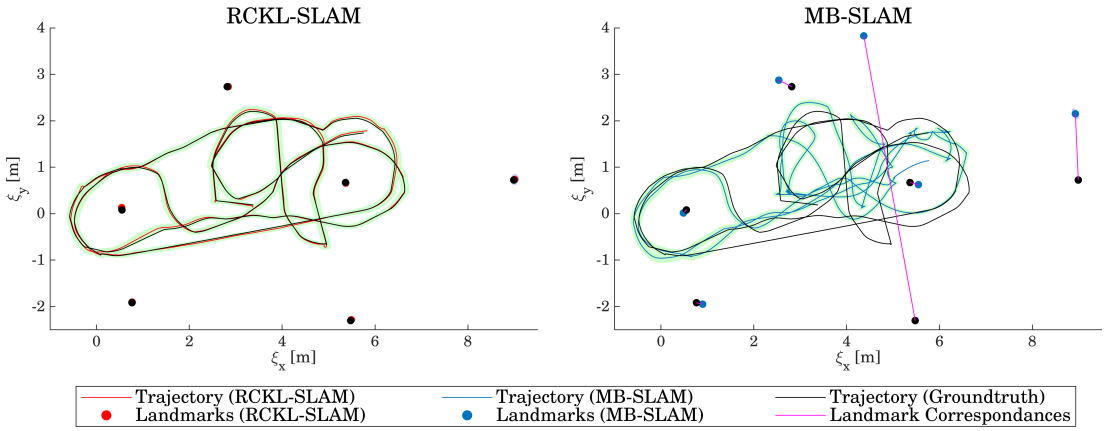


Figure 6.10: Visualization of SLAM output of Experiment 1 fold 6 for RCKL-SLAM and MB-SLAM, showing the estimators' mean states and mean landmark positions compared to the groundtruth. The green regions and the grey regions are, respectively, the 3σ covariances of the trajectory and of the landmarks, though the 3σ bounds of the landmarks are barely visible. The pink lines show the landmark correspondences between the groundtruth and the estimators' outputs. RCKL-SLAM's trajectories and landmarks are close to the groundtruth and are generally within the estimated 3σ bounds. On the other hand, MB-SLAM has converged to a local minimum with one landmark on the opposite side of the trajectories, and its estimate is far from being within 3σ bounds of the groundtruth.

a data augmentation procedure similar to that in Experiment 1, adding translational and rotational transformations on the original training data into the training set. We compare against MB, whose model contains the correct scaling factors, and against MBI, whose model uses a scaling factor of 1. The error plots for RCKL and the model-based estimators are shown in Fig. 6.11. A visualization of the outputs of RCKL-Loc and RCKL-SLAM can be seen in Fig. 6.2. As an ablation study, we also perform estimation with parts of RCKL's learned models swapped out with MB's prior models.

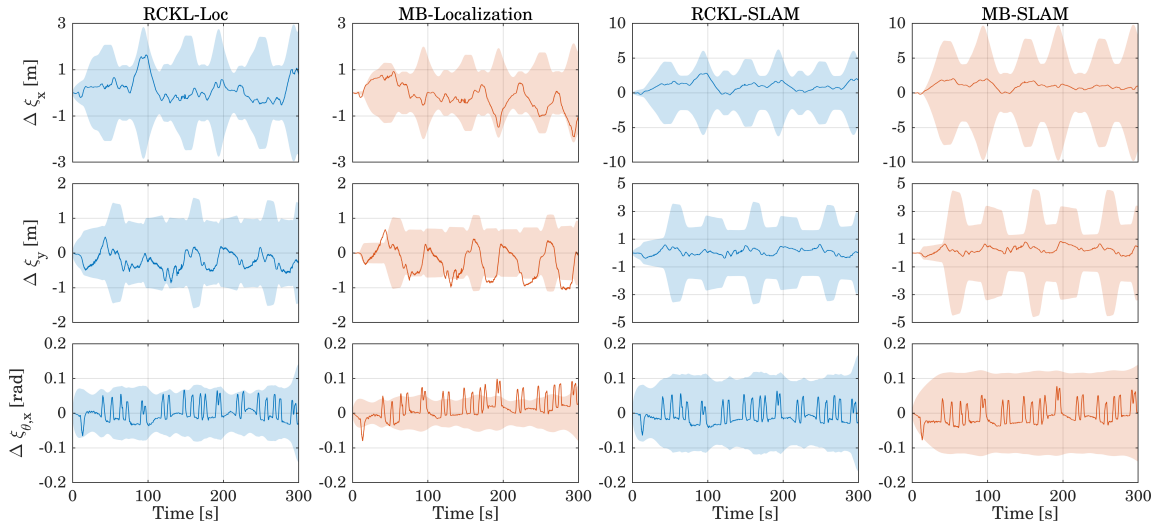


Figure 6.11: Error plots for Experiment 2, comparing RCKL-Loc with MB-Localization (left two plots) and comparing RCKL-SLAM with MB-SLAM (right two plots). The blue lines represent the errors of the estimated trajectories, and the red envelopes represent the estimated 3σ covariance bounds. For both localization and SLAM, RCKL's errors are similar to or smaller than those of MB. RCKL's errors are bounded by the 3σ bounds, while MB's errors sometimes exceed the 3σ bounds especially in localization. The shapes of the errors and the covariance envelopes of RCKL are similar to those of MB, suggesting that RCKL's outputs reflect the model reasonably well.

The RMSEs and Mahalanobis distances of the above estimators are shown in Table 6.1.

Fig. 6.11 qualitatively illustrates the effectiveness of RCKL, since the errors and covariances of RCKL appear similar to those of MB. Quantitatively, we see from Table 6.1 that RCKL has similar or better accuracy and consistency than MB in both localization and SLAM. This suggests that RCKL's learned models are better than MB's prior models. The ablation results in Table 6.1 also show that swapping to RCKL's measurement model reduces more error than swapping to RCKL's process model. This suggests that the main improvement comes from RCKL's measurement model, while the two process models are of similar quality. Finally, RCKL substantially outperforms MBI, demonstrating the advantage of the data-driven framework.

6.9.5 Computation Time

We briefly discuss the computation time of RCKL compared with that of the model-based estimators. Since RCKL and MB use substantially different cost functions, convergence should not be compared directly using cost tolerances. Thus, we instead view convergence as being sufficiently close to the groundtruth. In this sense, RCKL takes 1–3 times longer to converge than MB and MBI, depending on the environment setup. As an example, we compare CPU-based implementations of RCKL and MB on an Intel Core i7-9750H Processor. We show in Fig. 6.12 the estimator runtime for folds where both estimators converged near the groundtruth in Experiment 1. Each fold contains 200 seconds of data, consisting of 2000 timesteps (and inputs) and about 8000 range measurements. As seen in Fig. 6.12, both estimators converge near the groundtruth in less than 30 seconds. RCKL has the same Big-O inference complexity as the model-based estimators for both localization and SLAM, scaling linearly with the number of timesteps. The model-based estimators work with smaller matrices and thus have faster iterations than RCKL. However, they are not substantially faster overall because

Localization Method	Position RMSE [m]	Orientation RMSE [rad]	Maha. Dist.
RCKL Process, RCKL Measurement	0.592	**0.0246**	**0.834**
MB Process, MB Measurement	0.768	0.0339	6.167
RCKL Process, MB Measurement	0.790	0.0359	1.092
MB Process, RCKL Measurement	**0.582**	0.0253	6.161
MBI (imperfect model)	8.187	0.1249	6.422

SLAM Method	Position RMSE [m]	Orientation RMSE [rad]	Landmark RMSE [m]	Maha. Dist.
RCKL Process, RCKL Measurement	0.552	0.0293	**1.073**	0.865
MB Process, MB Measurement	**0.495**	**0.0280**	1.233	6.154
RCKL Process, MB Measurement	0.553	0.0289	1.298	**0.971**
MB Process, RCKL Measurement	1.781	0.0699	2.307	6.956
MBI (imperfect model)	1.428	0.0638	7.773	6.240

Table 6.1: *Ablation results of Experiment 2 for localization (top) and SLAM (bottom).* The first row corresponds to RCKL, the second row corresponds to MB, the third row corresponds to RCKL with MB's measurement model, and the fourth row corresponds to RCKL with MB's process model. The fifth row is MBI, the model-based estimator whose measurement model has imperfect scaling factors. For both localization and SLAM, RCKL has similar or lower RMSEs than MB, and a more consistent Mahalanobis distance. RCKL substantially outperforms MBI, demonstrating the advantage of the data-driven method.

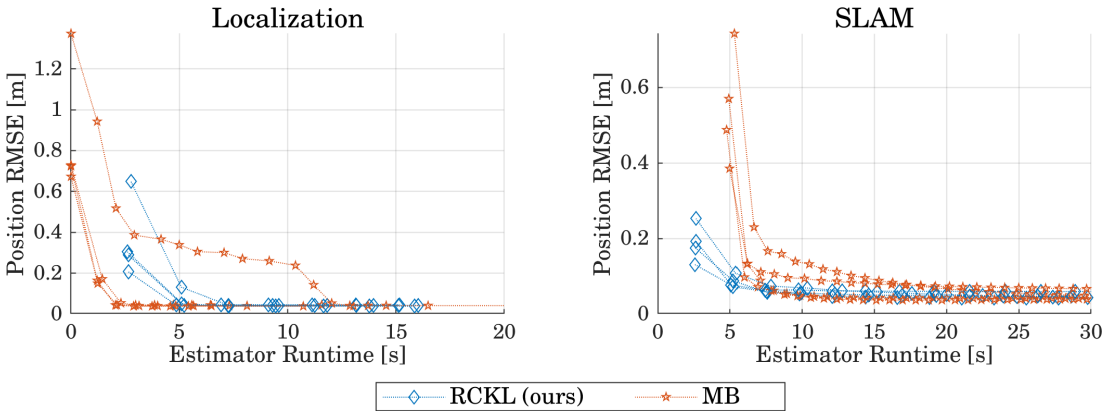


Figure 6.12: Position RMSE vs. estimator runtime for Experiment 1 for folds 1, 2, 3, 4, and 6 for localization, and folds 1, 2, 3, and 5 for SLAM, ignoring the folds where MB converges to a local minimum. For both RCKL and MB, markers are placed at every iteration, and the first markers are placed at the end of their initialization procedures. On all of these 200-second trajectories, both estimators converged to near the groundtruth within 30 seconds.

they usually require more iterations to converge near the groundtruth.

6.9.6 Hyperparameter Tuning

We discuss our procedure and recommendations for tuning the hyperparameters involved in RCKL. The main hyperparameters unique to this algorithm are the regularizers in (6.14b), namely τ_A, τ_B, τ_H, τ_D, τ_Q, τ_R. We used a different set of hyperparameter values for each of the experimental scenarios presented (Simulation 1, Simulation 2, Experiment 1, Experiment 2). We hand-tuned these values by manually adjusting them until a desired performance is reached on a validation dataset. Hyperparameters were never trained nor tuned on the test dataset. To achieve maximum performance, we recommend further refining the hyperparameters through a grid search or other hyperparameter optimization methods [104]. In our experiments, we saw that RCKL's performance is sensitive to only a few hyperparameter values. To show this, we performed a sensitivity analysis on the τ's used in Experiment 1. Fig. 6.13 shows the accuracy and consistency of RCKL-Loc and RCKL-SLAM as we individually vary each of the τ's within an order of magnitude above and below our hand-tuned values. We see that RCKL's accuracy and consistency do not change significantly across the tested ranges of τ_A, τ_B, τ_H, and τ_D. The covariance regularizers, τ_Q and τ_R, have larger effects on performance. Notably, the estimates are generally less consistent if τ_Q and τ_R are set either too low or too high, and generally less accurate if τ_Q and τ_R are set too high.

There are other settings that can be classified as hyperparameters, such as the lifting functions used and parameters within the SQP (e.g., μ in the line search acceptance criterion; see (C.15) in Appendix C.1.3). We recommend using standard methods from the literature to choose these parameters. For example, we chose SERFFs as our lifting functions, but more methodical Koopman identification methods [105, 69] can be used as well.

6.10 Conclusion

The results highlight the advantages of RCKL's data-driven approach to localization and SLAM. When the classical model-based estimator that relies on Gauss-Newton has access to perfect models in simulation, it has similar or slightly better accuracy than RCKL. However, RCKL has similar or slightly better accuracy than the model-based estimator with an imperfect model. On real-world datasets, RCKL is generally more accurate and more consistent than the classical model-based estimator, especially when compared against classical estimators with imperfect models. While the performance of the model-based estimators depends on the validity of their prior models, RCKL simply applies the high-dimensional model learned through data. In addition, the experiments suggest that RCKL may also be less prone to poor local minima.

RCKL provides a practical framework for data-driven localization and SLAM by converting nonlinear estimation into a constrained optimization problem over a lifted bilinear system. For localization, it generalizes KoopSE to environments with landmark configurations not seen during training. For SLAM, it enables the joint estimation of robot trajectories and previously unknown landmarks while preserving manifold consistency of the lifted variables. More generally, this chapter has shown that manifold-constrained optimization can substantially improve the reliability of Koopman-inspired estimation methods.

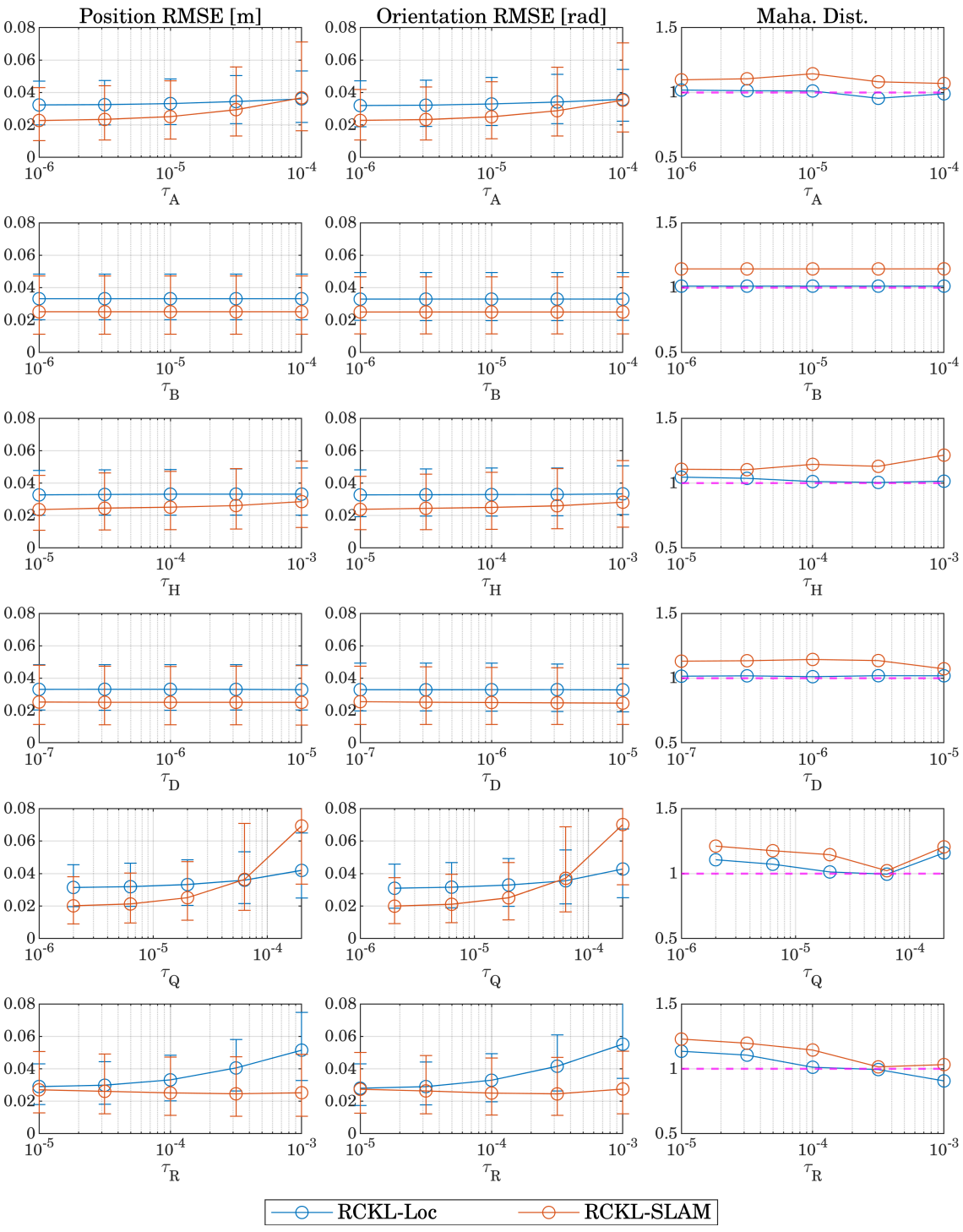


Figure 6.13: *RCKL hyperparameter sensitivity test*: Position RMSE (left column), orientation RMSE (middle column), and Mahalanobis distances (right column) of RCKL-Loc and RCKL-SLAM results on Experiment 1 (all 6 folds) vs. different hyperparameter values ($\tau_A, \tau_B, \tau_H, \tau_D, \tau_Q, \tau_R$). For all plots, the middle marker is placed at the hand-tuned hyperparameter value used to generate the results in Section 6.9.3. In each row, a hyperparameter is varied an order of magnitude above and below its hand-tuned value, while all other hyperparameters are set to their hand-tuned values. In the first two columns, we show the 50th, 25th, and 75th percentiles of the errors by representing them with the circle markers and the error bars' lower and upper bounds. In the third column, the markers represent the mean Mahalanobis distances.

Several avenues remain interesting directions for future work. It is worth investigating the origin of RCKL's improvement in convergence properties as a result of lifting estimation into a high-dimensional space. Another direction is to investigate updating the solution incrementally for real-time localization or SLAM, possibly by applying factor graph solvers [41, 106] or smoothing methods [40] in the lifted space. As well, we can broaden the model types that can be converted into lifted forms, making a further step towards data-driven state estimation for general robotics systems.

In developing RCKL-SLAM, it was necessary to extract covariance estimates from the resulting constrained optimization problem. Although such estimates were obtained through expressions tailored to the SQP formulation, their broader interpretation was not initially clear. This raises a more general question: how should Gaussian uncertainty be handled when variables are coupled through smooth constraints or manifolds? Perhaps surprisingly, examining this question reveals that these covariance expressions admit a clean probabilistic interpretation as conditioning operations onto linearized manifolds. This motivates the next chapter, where we develop a general framework for Gaussian inference on linearized manifolds.

Chapter 7

Gaussian Inference on Linearized Manifolds

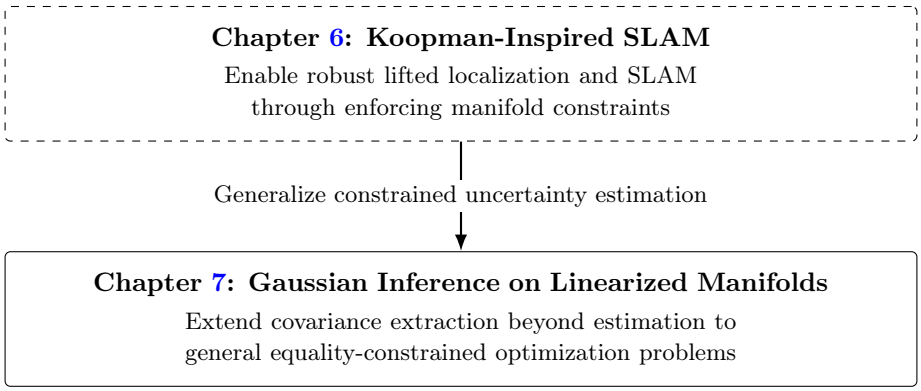


Figure 7.1: Conceptual progression from Chapter 6 to Chapter 7.

7.1 Summary of Contributions

This chapter develops a general framework for Gaussian inference on linearized manifolds, motivated by the constrained estimation problems encountered in Chapter 6. It provides a probabilistic interpretation of the covariance formulas that arise in such problems, and extends classical Gaussian inference operations to settings involving smooth equality constraints. In doing so, it supplies a principled uncertainty estimation framework for such constrained systems. Specifically, this chapter

- establishes the theoretical interpretation of covariance extraction for constrained estimators as Gaussian conditioning onto linearized manifolds,
- develops corresponding formulations for marginalization and conditioning onto smooth manifolds through local linearization, and
- demonstrates how these results enable principled uncertainty quantification in constrained estimation problems and smooth equality-constrained optimization frameworks more broadly.

Associated publication: Z. C. Guo, J. R. Forbes, and T. D. Barfoot, "Marginalizing and conditioning gaussians onto linear approximations of smooth manifolds with applications in robotics," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2606–2612, May 2025. DOI: 10.1109/ICRA55743.2025.11128000.

Recognition: Recipient of the **Best Conference Paper Award** at ICRA 2025.
Video summary of publication: https://youtu.be/O84uAMSZ-JI?si=JSjFVMeea87vwu5L.

7.2 Motivation

In robotics, there is an increasing need for probabilistic inference on smooth manifolds and other structured state spaces. One example arose in Chapter 6, where localization and SLAM were posed as constrained optimization problems in high-dimensional lifted spaces, and covariance extraction was required. Other examples include estimation frameworks based on Lie groups [34, 35, 36], other lifted-space representations [107], and explicit problem constraints [41]. For several important manifold classes, particularly Lie groups, established methods support both state estimation and uncertainty representation. However, covariance estimation becomes less straightforward when the manifold is induced by problem-specific constraints or learned representations. For example, although both unconstrained [30] and constrained [41] methods based on the Georgia Tech Smoothing and Mapping (GTSAM) library solve for the mean trajectory, only the unconstrained formulation readily provides covariance estimates [86]. Even when such covariance estimates can be obtained in practice, as in RCKL-SLAM in the previous chapter, their interpretation can be unclear: they may arise from optimizer-specific derivations rather than from a unified probabilistic framework.

This gap reflects the limited availability of general probabilistic inference tools for variables constrained to arbitrary smooth manifolds. In robotics, two prevalent inference operations are *marginalization* and *conditioning*. These operations are well established for Gaussian distributions in Euclidean settings [108] and are integral to the Bayesian-inference framework [22, 1]. However, their classical forms do not directly generalize to variables related through smooth constraints or manifold structures.

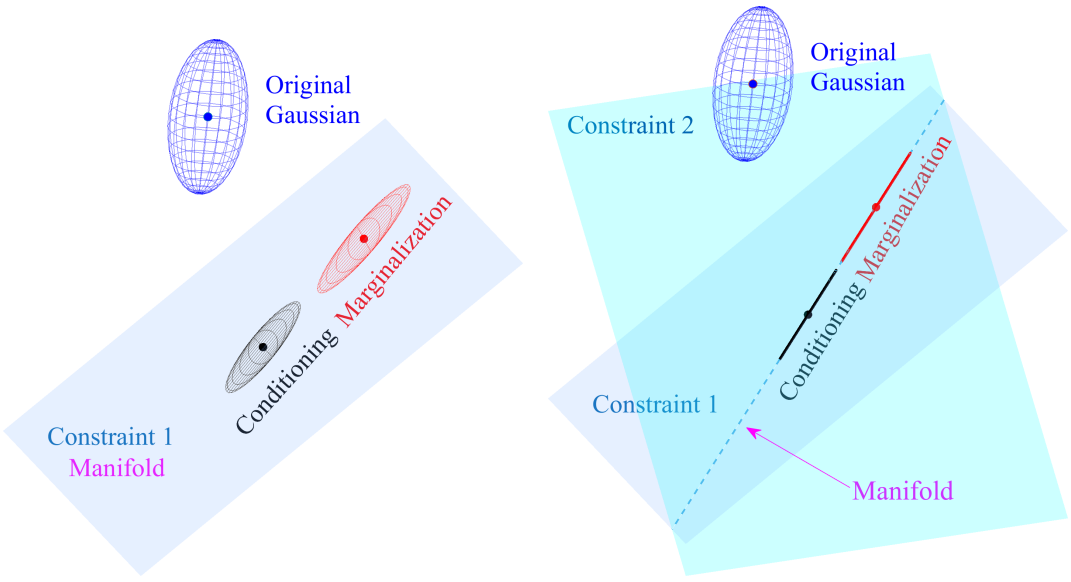
In this chapter, we address this gap by extending the concepts of marginalizing and conditioning to smooth manifolds. Our key technical results include

1. closed-form expressions for marginalizing and conditioning Gaussians onto general linear manifolds (see Fig. 7.2 for a preview),

2. approximations for marginalizing and conditioning Gaussians onto smooth nonlinear manifolds through local linearization, and

3. applications of our novel expressions, including approximation of the projected normal distribution [44], (a reinterpretation of) covariance extraction in RCKL-SLAM, and covariance extraction in constrained GTSAM [41].

This chapter is structured as follows. We review related work in Section 7.3, and discuss the established theory on marginalizing and conditioning Gaussians onto axis-aligned manifolds in Section 7.4. We present our expressions and derivations for these operations on general linear manifolds



(a) One Constraint (b) Two Constraints

Figure 7.2: *Marginalizing and conditioning Gaussians onto manifolds defined by linear constraints using Table 7.2.* Intuitively, marginalization corresponds to a projection onto the manifold, and conditioning corresponds to a normalized intersection with the manifold. Although the covariance of the original Gaussian is rank 3, it collapses to the rank of the manifold after marginalization or conditioning. In (a), the resulting Gaussians have rank-2 covariances, lying on the constraint plane. In (b), the resulting Gaussians have rank-1 covariances, lying on the intersection of the two planes.

in Section 7.5, and present our approximation method for nonlinear manifolds in Section 7.6. We present three applications of our theoretical contributions in Section 7.7, and conclude in Section 7.8.

7.3 Related Work

There is limited work on covariance estimation in a general constrained setting. One study [42] projects Gaussians onto linear constraint manifolds for sensor fusion, but the manifold is restricted to those passing through the origin, i.e., manifolds of the form $\mathbf{S}^\top \mathbf{x} = \mathbf{0}$. Another study [43] focuses on constrained weighted least squares and proves that the covariance is contained within the inverse Karush–Kuhn–Tucker (KKT) matrix. However, extracting this covariance remains challenging, especially for large systems. In addition, the probabilistic interpretations of both methods are not explicit; it is not immediately clear whether they are related to marginalization or conditioning. Although we do not pursue this connection in detail, the formulas developed later in this chapter suggest that both methods can be viewed as instances of conditioning Gaussian distributions onto constraint manifolds.

Meanwhile, there is interest in marginalizing and conditioning Gaussians onto specific manifolds, including two-dimensional circles [44] and n-dimensional spheres [45]. In robotics, Lie groups [33] are often a main focus [1, 34, 35, 36, 37, 38, 39]. To our knowledge, however, no methods have been presented for arbitrary smooth manifolds. This chapter begins to fill this gap by developing a general framework for marginalizing and conditioning Gaussian distributions onto smooth manifolds through local linearization.

Marginalization and Conditioning onto Axis-Aligned Manifolds

$$p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}^{-1}(\boldsymbol{\eta}, \boldsymbol{\Lambda}),$$
$$\boldsymbol{\eta} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \quad \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1},$$
Manifold: $\mathbf{x}_\beta = \boldsymbol{\beta}$,
$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_\alpha \\ \mathbf{x}_\beta \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_\alpha \\ \boldsymbol{\mu}_\beta \end{bmatrix}, \quad \boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\eta}_\alpha \\ \boldsymbol{\eta}_\beta \end{bmatrix},$$
$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\alpha\alpha} & \boldsymbol{\Sigma}_{\alpha\beta} \\ \boldsymbol{\Sigma}_{\alpha\beta}^\top & \boldsymbol{\Sigma}_{\beta\beta} \end{bmatrix}, \quad \boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\Lambda}_{\alpha\alpha} & \boldsymbol{\Lambda}_{\alpha\beta} \\ \boldsymbol{\Lambda}_{\alpha\beta}^\top & \boldsymbol{\Lambda}_{\beta\beta} \end{bmatrix}.$$

Marginalization

$$\boldsymbol{\mu}_{\text{marg}} = \begin{bmatrix} \boldsymbol{\mu}_{\text{marg},\alpha} \\ \boldsymbol{\beta} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{\text{marg}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\text{marg},\alpha\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{7.0.1}$$

Covariance Form

$$\boldsymbol{\mu}_{\text{marg},\alpha} = \boldsymbol{\mu}_\alpha, \tag{7.0.2}$$
$$\boldsymbol{\Sigma}_{\text{marg},\alpha\alpha} = \boldsymbol{\Sigma}_{\alpha\alpha}. \tag{7.0.3}$$

Information Form

$$\boldsymbol{\eta}_{\text{marg},\alpha} = \boldsymbol{\Sigma}_{\text{marg},\alpha\alpha}^{-1}\boldsymbol{\mu}_{\text{marg},\alpha} = \boldsymbol{\eta}_\alpha - \boldsymbol{\Lambda}_{\alpha\beta}\boldsymbol{\Lambda}_{\beta\beta}^{-1}\boldsymbol{\eta}_\beta, \tag{7.0.4}$$
$$\boldsymbol{\Lambda}_{\text{marg},\alpha\alpha} = \boldsymbol{\Sigma}_{\text{marg},\alpha\alpha}^{-1} = \boldsymbol{\Lambda}_{\alpha\alpha} - \boldsymbol{\Lambda}_{\alpha\beta}\boldsymbol{\Lambda}_{\beta\beta}^{-1}\boldsymbol{\Lambda}_{\alpha\beta}^\top. \tag{7.0.5}$$

Conditioning

$$\boldsymbol{\mu}_{\text{cond}} = \begin{bmatrix} \boldsymbol{\mu}_{\text{cond},\alpha} \\ \boldsymbol{\beta} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{\text{cond}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\text{cond},\alpha\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{7.0.6}$$

Covariance Form

$$\boldsymbol{\mu}_{\text{cond},\alpha} = \boldsymbol{\mu}_\alpha + \boldsymbol{\Sigma}_{\alpha\beta}\boldsymbol{\Sigma}_{\beta\beta}^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}_\beta), \tag{7.0.7}$$
$$\boldsymbol{\Sigma}_{\text{cond},\alpha\alpha} = \boldsymbol{\Sigma}_{\alpha\alpha} + \boldsymbol{\Sigma}_{\alpha\beta}\boldsymbol{\Sigma}_{\beta\beta}^{-1}\boldsymbol{\Sigma}_{\alpha\beta}^\top. \tag{7.0.8}$$

Information Form

$$\boldsymbol{\eta}_{\text{cond},\alpha} = \boldsymbol{\Sigma}_{\text{cond},\alpha\alpha}^{-1}\boldsymbol{\mu}_{\text{cond},\alpha} = \boldsymbol{\eta}_\alpha - \boldsymbol{\Lambda}_{\alpha\beta}\boldsymbol{\beta}, \tag{7.0.9}$$
$$\boldsymbol{\Lambda}_{\text{cond},\alpha\alpha} = \boldsymbol{\Sigma}_{\text{cond},\alpha\alpha}^{-1} = \boldsymbol{\Lambda}_{\alpha\alpha}. \tag{7.0.10}$$

Table 7.1: Expressions for marginalization and conditioning Gaussians onto *axis-aligned linear manifolds*, reproduced from [108].

7.4 Marginalization and Conditioning Gaussians onto Axis-Aligned Linear Manifolds

Prior to presenting our results for general linear manifolds, we first review marginalization and conditioning onto axis-aligned manifolds. This section will be referenced while deriving our general results in Section 7.5, where the rationale for the term 'axis-aligned manifold' becomes apparent.

Suppose we have a random variable, $\mathbf{x} \in \mathbb{R}^n$, associated with a probability distribution, $p(\mathbf{x})$. Suppose $\mathbf{x}$ can be broken down as $\mathbf{x} = \begin{bmatrix} \mathbf{x}_\alpha^\top & \mathbf{x}_\beta^\top \end{bmatrix}^\top$. The marginal distribution of $\mathbf{x}$ over $\mathbf{x}_\beta$ is denoted as $p_{\text{marg:}\beta}(\mathbf{x})$. Intuitively, marginalization summarizes $\mathbf{x}$ by summing up the distribution along $\mathbf{x}_\beta$. This can be seen as a projection of $p(\mathbf{x})$ onto the subspace $\mathbf{x}_\alpha$. The conditional distribution of $\mathbf{x}$ over $\mathbf{x}_\beta = \boldsymbol{\beta}$ is denoted as $p_{\text{cond:}\mathbf{x}_\beta=\boldsymbol{\beta}}(\mathbf{x})$. Conditioning corresponds to finding the distribution of $\mathbf{x}$ given that $\mathbf{x}_\beta = \boldsymbol{\beta}$ is known. It can be visualized as taking the slice of $p(\mathbf{x})$ where $\mathbf{x}_\beta = \boldsymbol{\beta}$, and then renormalizing the distribution along the slice. For more information, see [109].

Finding marginal and conditional distributions on axis-aligned manifolds can be difficult for a general $p(\mathbf{x})$. However, in the common case that $p(\mathbf{x})$ is Gaussian, the results can be written in closed form. It is known that marginalizing or conditioning Gaussians onto axis-aligned manifolds results in Gaussians. Gaussians are typically characterized either in covariance form as $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\Sigma}$ is the covariance, or in information form as $p(\mathbf{x}) \sim \mathcal{N}^{-1}(\boldsymbol{\eta}, \boldsymbol{\Lambda})$, where $\boldsymbol{\eta}$ is the information vector and $\boldsymbol{\Lambda}$ is the information matrix. Given these representations, the closed-form

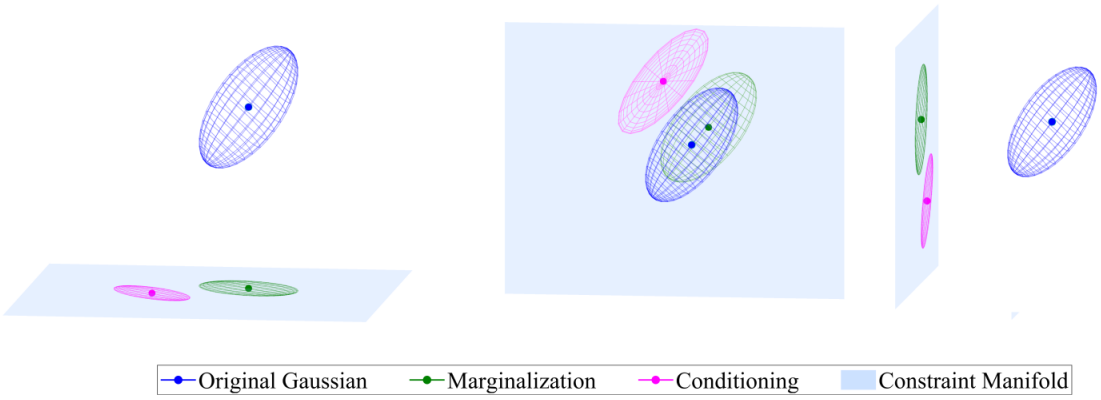


Figure 7.3: Visualization of marginalization and conditioning of a Gaussian onto axis-aligned manifolds of the form $\mathbf{x}_\beta = \boldsymbol{\beta}$. The left, middle, and right planes illustrate manifolds aligned with the xy-, yz-, and xz-planes, respectively, which correspond to fixing different coordinate subsets of $\mathbf{x}$. Marginalization (green) can be interpreted as projecting the Gaussian onto the manifold, while conditioning (magenta) corresponds to taking a slice of the Gaussian along the manifold and renormalizing the resulting distribution.

expressions for Gaussians under marginalization and conditioning are presented in Table 7.1, which is rewritten from [108] using our notation. For convenience, results are expressed in covariance form and in information form. See Fig. 7.3 for a visualization of these results.

After marginalizing or conditioning onto $\mathbf{x}_\beta = \boldsymbol{\beta}$, the resulting covariances are rank-deficient. In (7.0.1) and (7.0.6), $\text{rank}(\boldsymbol{\Sigma}_{\text{marg}}) = \text{rank}(\boldsymbol{\Sigma}_{\text{cond}}) = n_\alpha$. Both inference operations effectively set $\mathbf{x}_\beta$ to a fixed value, and the $\mathbf{x}_\beta$ component is no longer associated with a distribution. Despite this, the Gaussians retain their original dimensionalities: $\dim(\boldsymbol{\Sigma}_{\text{marg}}) = \dim(\boldsymbol{\Sigma}_{\text{cond}}) = n$. This situation is analogous to projecting a vector onto a subspace; the orthogonal component becomes 0, but the projected vector's dimensionality is not reduced. The conventional expressions for marginalization and conditioning [108] are written only for the $\mathbf{x}_\alpha$ subspace and have implicitly stripped off the padding associated with the $\mathbf{x}_\beta$ subspace. In (7.0.1) and (7.0.6), we leave in the padding. This turns out to be essential for deriving our results for general linear manifolds.[1]

For many applications, the inference results in Table 7.1 are sufficient. As a classical example, the Kalman filter expressions can be derived [1, §3] using Table 7.1. However, as we will see in Section 7.7, there is a growing demand for marginalizing and conditioning Gaussians onto *general manifolds*.

7.5 Marginalization and Conditioning Gaussians onto General Linear Manifolds

7.5.1 General Manifolds

Before confining to linear manifolds, we first clarify our definitions for marginalizing and conditioning onto smooth, potentially nonlinear manifolds, in preparation for Section 7.6. Suppose we have a random variable, $\mathbf{x} \in \mathbb{R}^n$, an associated probability distribution, $p(\mathbf{x})$, and a manifold, $\mathcal{M} : \mathbf{f}(\mathbf{x}) = \mathbf{0}$,

[1] Note that the *information form* results in Table 7.1 are still defined over only the $\mathbf{x}_\alpha$ subspace, since $\boldsymbol{\Sigma}^{-1}_{\text{marg}}$ and $\boldsymbol{\Sigma}^{-1}_{\text{cond}}$ do not exist.

Marginalization and Conditioning onto Linear Manifolds	
$p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \mathbf{x} \in \mathbb{R}^n,$	
Manifold: $\mathbf{S}^\top \mathbf{x} = \mathbf{c}, \quad \mathbf{S} \in \mathbb{R}^{n \times m},$	
$\mathbf{Z} = \text{null}(\mathbf{S}^\top) \in \mathbb{R}^{n \times (n-m)}, \quad \boldsymbol{\Pi} = \mathbf{Z}(\mathbf{Z}^\top \mathbf{Z})^{-1}\mathbf{Z}^\top,$	
$\mathbf{x}_0 = \mathbf{S}(\mathbf{S}^\top \mathbf{S})^{-1}\mathbf{c}.$	
Marginalization	**Conditioning**
$\boldsymbol{\mu}_{\text{marg}} = \boldsymbol{\Pi}\boldsymbol{\mu} + \mathbf{x}_0, \quad (7.1.1)$	$\boldsymbol{\mu}_{\text{cond}} = \mathbf{x}_0 + \boldsymbol{\Sigma}_{\text{cond}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{x}_0), \quad (7.1.3)$
$\boldsymbol{\Sigma}_{\text{marg}} = \boldsymbol{\Pi}\boldsymbol{\Sigma}\boldsymbol{\Pi}^\top. \quad (7.1.2)$	$\boldsymbol{\Sigma}_{\text{cond}} = \mathbf{Z}(\mathbf{Z}^\top \boldsymbol{\Sigma}^{-1}\mathbf{Z})^{-1}\mathbf{Z}^\top. \quad (7.1.4)$

Table 7.2: Expressions for marginalization and conditioning Gaussians onto *general linear manifolds*

where $\mathbf{f}(\cdot)$ is differentiable. Suppose the manifold is m-dimensional. That is, $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^{n-m}$.

For marginalization, suppose $\mathcal{M}$ has a projection method, $\mathbf{g}_{\mathcal{M}} : \mathbb{R}^n \to \mathbb{R}^n$, where for any $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{g}_{\mathcal{M}}(\mathbf{x}) = \tilde{\mathbf{x}} \in \mathcal{M}$, meaning that $\mathbf{f}(\tilde{\mathbf{x}}) = \mathbf{0}$. A common projection method is the closest on-manifold point measured via the Euclidean distance: $\mathbf{g}_{\mathcal{M}}(\mathbf{x}) = \text{argmin}_{\mathbf{p} \in \mathcal{M}} ||\mathbf{p} - \mathbf{x}||^2$. Then, $p_{\text{marg}:\mathcal{M}}(\mathbf{x})$ is the result of projecting $p(\mathbf{x})$ onto $\mathcal{M}$ with $\mathbf{g}_{\mathcal{M}}$. For conditioning, we simply condition $p(\mathbf{x})$ over the points on the manifold: $p_{\text{cond}:\mathcal{M}}(\mathbf{x}) = p(\mathbf{x} \mid \mathbf{f}(\mathbf{x}) = \mathbf{0})$. Instead of slicing and renormalizing $p(\mathbf{x})$ over fixed values, we renormalize over the slice defined by the manifold. Note that for both marginalization and conditioning, the density is zero at all points off the manifold.

7.5.2 Linear Manifolds

Marginal and conditional distributions are difficult to compute for a general $\mathbf{f}(\mathbf{x})$. However, linear manifolds greatly simplify the situation. Suppose we have $\mathbf{f}(\mathbf{x}) = \mathbf{S}^\top \mathbf{x} - \mathbf{c} = \mathbf{0}$, where $\mathbf{S} \in \mathbb{R}^{n \times m}$, $m < n$, $\mathbf{c} \in \mathbb{R}^m$. We assume that $\text{rank}(\mathbf{S}) = m$, meaning that $\mathbf{S}$ has full column rank. Then, the marginal distribution of $p(\mathbf{x})$ onto $\mathcal{M}$ is

$$p_{\text{marg}:\mathcal{M}}(\mathbf{x}) = \begin{cases} \int_{\mathbf{v} \in \text{span } \mathbf{S}} p(\mathbf{x} + \mathbf{v}) \mathrm{d}\mathbf{v}, & \mathbf{S}^\top \mathbf{x} = \mathbf{c}, \\ 0, & \text{otherwise}, \end{cases} \tag{7.1}$$

where $\mathrm{d}\mathbf{v}$ denotes integration over the subspace span $\mathbf{S}$, whose vectors are orthogonal to the manifold. For conditioning, the distribution is similar to the case for general manifolds: $p_{\text{cond}:\mathcal{M}}(\mathbf{x}) = p(\mathbf{x} \mid \mathbf{S}^\top \mathbf{x} = \mathbf{c})$.

7.5.3 Expressions for Gaussians

Now, suppose that $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. We present one of our main contributions in Table 7.2: the closed-form expressions for marginalizing and conditioning a Gaussian onto a general linear manifold. Here, $\text{null}(\mathbf{S}^\top)$ denotes a matrix whose columns form a basis for the nullspace of $\mathbf{S}^\top$.

Note that the formulation for Table 7.1, as described in the previous Section 7.4, can be written in the formulation for Table 7.2 with $\mathbf{S}^\top = \begin{bmatrix} \mathbf{0} & \mathbf{1} \end{bmatrix}$ and $\mathbf{c} = \boldsymbol{\beta}$. That is, *Table 7.1 is a special case of Table 7.2 where the linear manifold is axis-aligned.* One can verify that the expressions for axis-aligned manifolds can be derived given our general expressions. Similar to the axis-aligned case described in Section 7.4, the Gaussian obtained after inference onto a linear manifold is degenerate. For marginalization, $\boldsymbol{\mu}_{\text{marg}} \in \mathbb{R}^n$, $\boldsymbol{\Sigma}_{\text{marg}} \in \mathbb{R}^{n \times n}$, but $\text{rank}(\boldsymbol{\Sigma}_{\text{marg}}) = n - m$, inheriting the rank of

the manifold. See Fig. 7.2 for a visualization.

Geometrically, the vector $\mathbf{x}_0$ used in $\boldsymbol{\mu}_{\text{marg}}$ and $\boldsymbol{\mu}_{\text{cond}}$ is the orthogonal projection of the origin onto the affine manifold. We can see this by noting that the expression for $\mathbf{x}_0$ corresponds to the minimum-norm point satisfying the linear constraints, i.e.,

$$\mathbf{x}_0 = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{x}\|^2 \quad \text{s.t.} \quad \mathbf{S}^\top \mathbf{x} = \mathbf{c}. \tag{7.2}$$

This interpretation also explains why $\mathbf{x}_0$ appears in $\boldsymbol{\mu}_{\text{marg}}$ and $\boldsymbol{\mu}_{\text{cond}}$: both means must lie on the affine manifold, and are therefore composed of a feasible offset point, $\mathbf{x}_0$, together with directions spanning the nullspace of $\mathbf{S}^\top$.

Unlike in Table 7.1, there are no clear information-form representations for marginal and conditional distributions on general linear manifolds; the resulting covariances are rank-deficient and thus not invertible. If desired, one can presumably write the information-form expressions within a subspace where the Gaussian is nondegenerate. However, the choice of subspace would then depend on the application. In any case, the covariance-form results in Table 7.2 turn out to be sufficient for our use cases described in Section 7.7.

7.5.4 Proof for Gaussian Inference on Linear Manifolds

We now prove the correctness of the expressions in Table 7.2 using coordinate transformations. We use the 'x-frame' to denote the original frame, and the 'y-frame' to denote the new frame after our coordinate transformation. We proceed in three main steps: 1) transform the Gaussian into a frame where the linear manifold is axis-aligned; 2) apply Table 7.1 formulas to marginalize or condition the Gaussian in this new frame; and 3) transform the resulting Gaussian back into the original frame.

The validity of this procedure follows from the fact that conditioning commutes with frame transformations. That is, conditioning in the x-frame is equivalent to first converting the distribution to the y-frame, then conditioning in the y-frame, and then converting the conditioned distribution back into the x-frame. The same reasoning applies to marginalization, with the added requirement that $\mathbf{v} \in \text{span } \mathbf{S} \;\Rightarrow\; \mathbf{H}^{-1}\mathbf{v} \in \text{span } \mathbf{H}^\top \mathbf{S}$, where $\mathbf{H}$ is the frame transformation. In other words, vectors orthogonal to the manifold in the x-frame are still orthogonal to the manifold in the y-frame. The transformation $\mathbf{H}$ used below satisfies this requirement.

Transform the Gaussian into an Axis-Aligned Frame

Let the transformation from the x-frame to the y-frame be $\mathbf{y} = \mathbf{H}^{-1}\mathbf{x}$, where $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{H} \in \mathbb{R}^{n \times n}$. We break down $\mathbf{y}$ as

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_\alpha^\top & \mathbf{y}_\beta^\top \end{bmatrix}^\top, \quad \mathbf{y}_\alpha \in \mathbb{R}^{n-m}, \quad \mathbf{y}_\beta \in \mathbb{R}^m. \tag{7.3}$$

Let the transformation matrix be $\mathbf{H} = \begin{bmatrix} \mathbf{Z} & \mathbf{S} \end{bmatrix}$. Note that since $\mathbf{S}$ has full column rank and $\mathbf{Z} = \text{null}(\mathbf{S}^\top)$, $\mathbf{H}$ is guaranteed to be full rank. Then, we can see that $\mathbf{H}^{-1} = \begin{bmatrix} (\mathbf{Z}^\top \mathbf{Z})^{-1}\mathbf{Z}^\top \\ (\mathbf{S}^\top \mathbf{S})^{-1}\mathbf{S}^\top \end{bmatrix}$, using the fact that $\mathbf{S}^\top \mathbf{Z} = \mathbf{0}$.

Let the original Gaussian in the x-frame be $p_x(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) = \mathcal{N}^{-1}(\boldsymbol{\eta}_x, \boldsymbol{\Lambda}_x)$, where the

$(\cdot)_x$ subscript clarifies the parameters' frame. In the y-frame, the Gaussian becomes $p_y(\mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y) = \mathcal{N}^{-1}(\boldsymbol{\eta}_y, \boldsymbol{\Lambda}_y)$, where

$$\boldsymbol{\mu}_y = \mathbf{H}^{-1}\boldsymbol{\mu}_x, \qquad \boldsymbol{\Sigma}_y = \mathbf{H}^{-1}\boldsymbol{\Sigma}_x\mathbf{H}^{-\top}, \tag{7.4a}$$

$$\boldsymbol{\eta}_y = \mathbf{H}^{\top}\boldsymbol{\eta}_x, \qquad \boldsymbol{\Lambda}_y = \mathbf{H}^{\top}\boldsymbol{\Lambda}_x\mathbf{H}. \tag{7.4b}$$

The linear manifold, $\mathbf{S}^{\top}\mathbf{x} = \mathbf{c}$, in the y-frame becomes

$$\mathbf{S}^{\top}\mathbf{H}\mathbf{y} = \mathbf{c} \;\Rightarrow\; \begin{bmatrix}\mathbf{0} & \mathbf{S}^{\top}\mathbf{S}\end{bmatrix}\begin{bmatrix}\mathbf{y}_\alpha \\ \mathbf{y}_\beta\end{bmatrix} = \mathbf{c} \;\Rightarrow\; \mathbf{y}_\beta = (\mathbf{S}^{\top}\mathbf{S})^{-1}\mathbf{c}. \tag{7.5}$$

Thus, in this y-frame, the manifold is axis-aligned. We are now in position to apply the formulas in Table 7.1 to marginalize or condition in the y-frame.

Marginalize or Condition in the Axis-Aligned Frame

For brevity, we omit denoting the manifold by using $p_{\text{marg},x}(\mathbf{x})$ for $p_{\text{marg}:\mathcal{M},x}(\mathbf{x})$ and $p_{\text{cond},x}(\mathbf{x})$ for $p_{\text{cond}:\mathcal{M},x}(\mathbf{x})$ in the x-frame, and similarly in the y-frame. Using (7.0.1), the marginalized Gaussian in the y-frame is $p_{\text{marg},y}(\mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}_{\text{marg},y}, \boldsymbol{\Sigma}_{\text{marg},y})$, where

$$\boldsymbol{\mu}_{\text{marg},y} = \begin{bmatrix}\boldsymbol{\mu}_{\text{marg},y,\alpha} \\ (\mathbf{S}^{\top}\mathbf{S})^{-1}\mathbf{c}\end{bmatrix}, \quad \boldsymbol{\Sigma}_{\text{marg},y} = \begin{bmatrix}\boldsymbol{\Sigma}_{\text{marg},y,\alpha\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \tag{7.6}$$

where, from applying (7.0.2), (7.0.3), and (7.4a), we have

$$\boldsymbol{\mu}_{\text{marg},y,\alpha} = \boldsymbol{\mu}_{y,\alpha} = (\mathbf{Z}^{\top}\mathbf{Z})^{-1}\mathbf{Z}^{\top}\boldsymbol{\mu}_x, \quad \boldsymbol{\Sigma}_{\text{marg},y,\alpha\alpha} = \boldsymbol{\Sigma}_{y,\alpha\alpha} = (\mathbf{Z}^{\top}\mathbf{Z})^{-1}\mathbf{Z}^{\top}\boldsymbol{\Sigma}_x\mathbf{Z}(\mathbf{Z}^{\top}\mathbf{Z})^{-1}. \tag{7.7}$$

We now move on to conditioning, whose proof is simpler by working with $p_y(\mathbf{y})$ in information form. From (7.4b), we have

$$\boldsymbol{\eta}_y = \mathbf{H}^{\top}\boldsymbol{\eta}_x = \begin{bmatrix}\mathbf{Z}^{\top}\boldsymbol{\Sigma}_x^{-1}\boldsymbol{\mu}_x \\ \mathbf{S}^{\top}\boldsymbol{\Sigma}_x^{-1}\boldsymbol{\mu}_x\end{bmatrix} = \begin{bmatrix}\boldsymbol{\eta}_{y,\alpha} \\ \boldsymbol{\eta}_{y,\beta}\end{bmatrix}, \quad \boldsymbol{\Lambda}_y = \begin{bmatrix}\mathbf{Z}^{\top}\boldsymbol{\Sigma}_x^{-1}\mathbf{Z} & \mathbf{Z}^{\top}\boldsymbol{\Sigma}_x^{-1}\mathbf{S} \\ \mathbf{S}^{\top}\boldsymbol{\Sigma}_x^{-1}\mathbf{Z} & \mathbf{S}^{\top}\boldsymbol{\Sigma}_x^{-1}\mathbf{S}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Lambda}_{y,\alpha\alpha} & \boldsymbol{\Lambda}_{y,\alpha\beta} \\ \boldsymbol{\Lambda}_{y,\alpha\beta}^{\top} & \boldsymbol{\Lambda}_{y,\beta\beta}\end{bmatrix}. \tag{7.8}$$

We now apply (7.0.9) and (7.0.10) in Table 7.1 to obtain

$$\boldsymbol{\eta}_{\text{cond},y,\alpha} = \mathbf{Z}^{\top}\boldsymbol{\Sigma}_x^{-1}(\boldsymbol{\mu}_x - \mathbf{S}(\mathbf{S}^{\top}\mathbf{S})^{-1}\mathbf{c}), \quad \boldsymbol{\Lambda}_{\text{cond},y,\alpha\alpha} = \mathbf{Z}^{\top}\boldsymbol{\Sigma}_x^{-1}\mathbf{Z}. \tag{7.9}$$

Then, using (7.0.6), we get $p_{\text{cond},y}(\mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}_{\text{cond},y}, \boldsymbol{\Sigma}_{\text{cond},y})$ for the conditioned Gaussian in the y-frame, where

$$\boldsymbol{\mu}_{\text{cond},y} = \begin{bmatrix}\boldsymbol{\mu}_{\text{cond},y,\alpha} \\ (\mathbf{S}^{\top}\mathbf{S})^{-1}\mathbf{c}\end{bmatrix}, \quad \boldsymbol{\Sigma}_{\text{cond},y} = \begin{bmatrix}\boldsymbol{\Sigma}_{\text{cond},y,\alpha\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \tag{7.10}$$

where $\boldsymbol{\mu}_{\text{cond},y,\alpha} = \boldsymbol{\Sigma}_{\text{cond},y,\alpha\alpha}\boldsymbol{\eta}_{\text{cond},y,\alpha}$ and $\boldsymbol{\Sigma}_{\text{cond},y,\alpha\alpha} = \boldsymbol{\Lambda}_{\text{cond},y,\alpha\alpha}^{-1}$.

Transform the Gaussian Back into the Original Frame

To obtain $p_{\text{marg},x}(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_{\text{marg},x}, \boldsymbol{\Sigma}_{\text{marg},x})$, we simply apply the inverse transformation, $\mathbf{x} = \mathbf{H}\mathbf{y}$, to $p_{\text{marg},y}(\mathbf{y})$. This yields $\boldsymbol{\mu}_{\text{marg},x} = \mathbf{H}\boldsymbol{\mu}_{\text{marg},y}$ and $\boldsymbol{\Sigma}_{\text{marg},x} = \mathbf{H}\boldsymbol{\Sigma}_{\text{marg},y}\mathbf{H}^\top$ from inverting the transformations in (7.4a). After simplifying, the marginalized mean becomes (7.1.1), and the marginalized covariance becomes (7.1.2). The procedure is similar for proving (7.1.3) and (7.1.4) for conditioning. We have proven the results in Table 7.2, as desired.

7.6 Marginalizing and Conditioning Gaussians onto Linearized Manifolds

Although we have generalized marginalization and conditioning to linear manifolds, many scenarios involve nonlinear manifolds. Section 7.7 discusses several examples, including RCKL-SLAM from the previous chapter, where the manifold is defined by nonlinear lifting functions, and constrained GTSAM, where the manifold is often nonlinear from the problem setup. Unlike the case with linear manifolds, the distributions obtained by marginalizing and conditioning Gaussians onto nonlinear manifolds are no longer Gaussian. Nevertheless, we will show in Section 7.7 that Gaussian approximations of these distributions are often sufficient in many applications. Below, we describe our approximation method, followed by a theoretical justification through interpreting our method as Laplace's approximation in constrained settings.

7.6.1 Approximation Method

We now describe our approximation method for marginalizing and conditioning Gaussians onto nonlinear manifolds through local linearization. Let the original Gaussian be $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and the manifold be $\mathcal{M} : \mathbf{f}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{f}(\cdot)$ is nonlinear but differentiable. We approximate $p_{\text{marg}:\mathcal{M}}(\mathbf{x})$ and $p_{\text{cond}:\mathcal{M}}(\mathbf{x})$ by operating on a tangent plane of $\mathcal{M}$. We first find a linearization point by projecting $\boldsymbol{\mu}$ onto the manifold: $\tilde{\boldsymbol{\mu}} = \mathbf{g}_{\mathcal{M}}(\boldsymbol{\mu})$. Note that in many optimization settings, $\boldsymbol{\mu}$ corresponds to the constrained solution and thus is already on the manifold. In this case, projections should not shift the mean. That is, if $\mathbf{f}(\boldsymbol{\mu}) = \mathbf{0}$, then $\tilde{\boldsymbol{\mu}} = \mathbf{g}_{\mathcal{M}}(\boldsymbol{\mu}) = \boldsymbol{\mu}$.

We then construct a tangent plane at $\tilde{\boldsymbol{\mu}}$, given as

$$T_{\tilde{\boldsymbol{\mu}}}\mathcal{M} : \mathbf{S}^\top \mathbf{x} = \mathbf{c}, \quad \text{where } \mathbf{S} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}^\top}\right|_{\mathbf{x}=\tilde{\boldsymbol{\mu}}}, \ \mathbf{c} = \mathbf{S}^\top \tilde{\boldsymbol{\mu}}. \tag{7.11}$$

Since $T_{\tilde{\boldsymbol{\mu}}}\mathcal{M}$ is linear, we can then apply our Table 7.2 formulas to marginalize or condition $p(\mathbf{x})$ onto $T_{\tilde{\boldsymbol{\mu}}}\mathcal{M}$. Since $\tilde{\boldsymbol{\mu}}$ lies on the tangent plane by construction, neither marginalization nor conditioning shifts the mean under the linear formulas in Table 7.2. Thus, after projecting the mean onto the manifold, only the covariance requires updating. The resulting distribution is still over $T_{\tilde{\boldsymbol{\mu}}}\mathcal{M}$ rather than $\mathcal{M}$. Thus, we construct a local chart around $\tilde{\boldsymbol{\mu}}$ based on a retraction method [110], and then map the distribution support from $T_{\tilde{\boldsymbol{\mu}}}\mathcal{M}$ to $\mathcal{M}$. See Fig. 7.4 for an example.

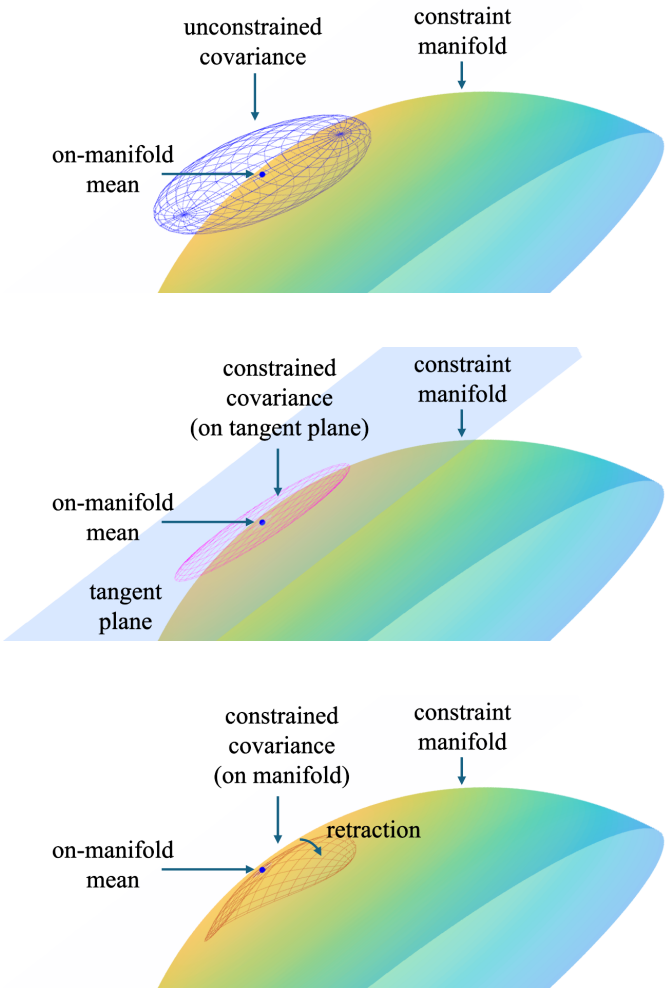


Figure 7.4: Visualization of conditioning a Gaussian onto a nonlinear manifold in an optimization context. *Top*: in many optimization settings, the constrained solution is approximated as a Gaussian with its mean already on the manifold. *Middle*: the Gaussian is conditioned onto the tangent plane linearized at the mean, yielding a constrained Gaussian approximation on the plane. *Bottom*: the conditioned Gaussian is retracted back onto the nonlinear manifold. In optimization settings, this conditioning procedure corresponds to finding Laplace's approximation of the constrained likelihood.

7.6.2 Theoretical Interpretation as Laplace's Approximation

We show that this approximation method has an interesting theoretical interpretation—conditioning the Gaussian solutions of optimization problems onto linearized constraints can be interpreted as Laplace's approximation. To see why, first note that many unconstrained optimization problems aim to find the MAP point of a likelihood function, $p(\mathbf{x})$. The setup is typically $\mathbf{x}^* = \text{argmin}_{\mathbf{x}}[-\log p(\mathbf{x})]$. Optimizers find the optimum point, $\mathbf{x}^*$, along with a local covariance, $\boldsymbol{\Sigma} = -\frac{\partial^2 \log p(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^\top}|_{\mathbf{x}^*}$. This is really approximating $p(\mathbf{x})$ with a Gaussian, $q(\mathbf{x}) \sim \mathcal{N}(\mathbf{x}^*, \boldsymbol{\Sigma})$, which corresponds to Laplace's approximation [75] of the (unconstrained) likelihood. Meanwhile, constrained optimization problems

are often set up as

$$\mathbf{x}^*_{\text{cond}} = \underset{\mathbf{x}\in\mathcal{M}}{\text{argmin}}[-\log p(\mathbf{x})] = \underset{\mathbf{x}}{\text{argmin}}[-\log p_{\text{cond}}(\mathbf{x})], \tag{7.12}$$

where $p_{\text{cond}}(\mathbf{x}) = p(\mathbf{x} \mid \mathbf{f}(\mathbf{x}) = \mathbf{0})$ is the likelihood conditioned on the constraints. On-manifold optimizers find the constrained optimum point, $\mathbf{x}^*_{\text{cond}}$, and its local unconstrained covariance, $\boldsymbol{\Sigma}' = -\frac{\partial^2 \log p(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^\top}|_{\mathbf{x}^*_{\text{cond}}}$. When we condition $\boldsymbol{\Sigma}'$ onto the linearized constraints, we are finding

$$\boldsymbol{\Sigma}_{\text{cond}} = -\frac{\partial^2 \log p(\mathbf{x} \mid \mathbf{f}(\mathbf{x}) = \mathbf{0})}{\partial \mathbf{x} \partial \mathbf{x}^\top}\bigg|_{\mathbf{x}^*_{\text{cond}}} = -\frac{\partial^2 \log p_{\text{cond}}(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^\top}\bigg|_{\mathbf{x}^*_{\text{cond}}}. \tag{7.13}$$

This shows that we are approximating $p_{\text{cond}}(\mathbf{x})$ with a Gaussian, $q_{\text{cond}}(\mathbf{x}) \sim \mathcal{N}(\mathbf{x}^*_{\text{cond}}, \boldsymbol{\Sigma}_{\text{cond}})$, corresponding to Laplace's approximation of the conditioned likelihood. Thus, many unconstrained optimization problems already admit a probabilistic interpretation as Laplace's approximation, and our method extends the same interpretation to constrained settings.

7.7 Applications

Having described our procedures for marginalizing and conditioning Gaussians onto linear and nonlinear manifolds, we now present three robotics applications: 1) using marginalization to approximate the projected normal distribution used for modeling angular uncertainty; 2) using conditioning to reinterpret the covariance extraction procedure of RCKL-SLAM from the previous chapter; and 3) using conditioning to extract covariances in constrained GTSAM.

7.7.1 Approximation of the Projected Normal Distribution

In this section, we illustrate marginalizing a Gaussian onto a typical nonlinear manifold, and we compare our approximation with the true marginal. In robotics, we are often interested in modeling angular uncertainty [111, 112, 113]. Angles and other directional data are modeled with directional distributions, including von Mises distributions [114] and projected normal distributions [115]. The general projected normal distribution [44] is the result of projecting (i.e., marginalizing) a Gaussian onto a unit circle. Given a Gaussian, $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the analytical expression of the marginal density function, $p(\theta \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$, is presented in [44].

We compare our linear approximation with the analytical marginal in Fig. 7.5 for three Gaussians with increasing noise. Following our procedure in Section 7.6, we first set a linearization point by projecting $\boldsymbol{\mu}$ onto the circle based on the minimum Euclidean distance. We then marginalize the Gaussian onto the tangent plane. To evaluate against the analytical marginal [44], we retract this marginal onto the circular manifold [116]. We then use the Kullback-Leibler (KL) divergence [117] between the retracted marginal and the analytical marginal to evaluate our approximation quality.

In Fig. 7.5, our approximation is close to the analytical marginal, especially for Gaussians with lower covariances. We see this visually and through inspecting the KL divergence values. This is expected since our approximation is exact for linear manifolds. Increasing either the curvature of the manifold or the noise level of the Gaussian increases the problem nonlinearity. However, for slightly nonlinear cases, our approximation is sufficient.

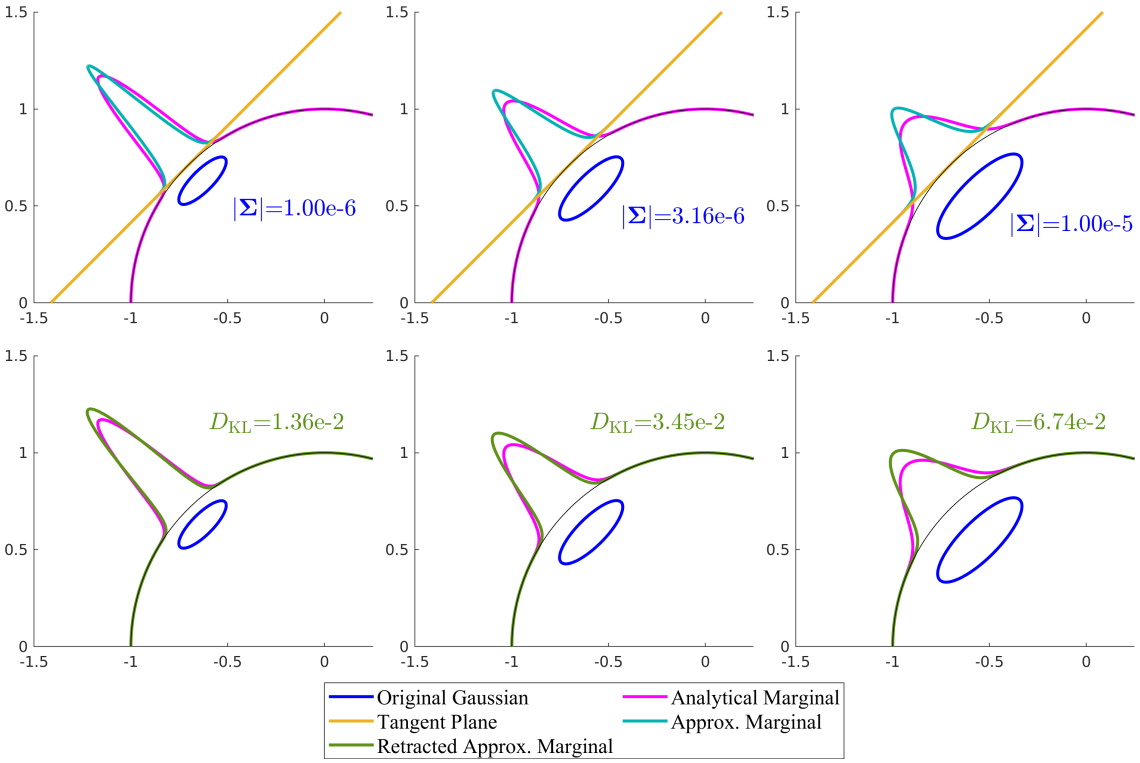


Figure 7.5: *Approximating the projected normal distribution [44].* The analytical marginal (pink) is the approximation target. The top figures show Gaussians (dark blue) marginalized (light blue) onto the tangent plane (orange), and the bottom figures show the retractions (green) of these tangent marginals onto the unit circle. Each column corresponds to a different Gaussian covariance, as indicated by the different $|\boldsymbol{\Sigma}|$ values. The retracted marginals are generally close to the analytical marginals. D_{KL}, the KL divergence between the analytical marginal and the retracted marginal, is especially small when $|\boldsymbol{\Sigma}|$ is low.

In this case, the analytical expression of the marginal is known, allowing us to directly verify the correctness and behavior of our approximation. However, the primary use of our method is in settings where analytical inference expressions are unavailable. We now consider two such applications.

7.7.2 Covariance Extraction in Koopman-Inspired Estimation

We begin with a familiar problem: covariance extraction in RCKL-SLAM from Chapter 6. We reinterpret these covariance expressions through the lens of Gaussian conditioning, showing that these expressions, originally derived in the context of SQP, can be understood as conditioning operations and thus as instances of Laplace's approximation. We also present an additional RCKL-SLAM experiment to validate the consistency of the resulting covariance estimates. Together, these results show that although constraints complicate covariance extraction in optimization problems, our approximation is theoretically justified and can be empirically effective.

Reinterpreting RCKL-SLAM Covariance Formula as Conditioning

As a reminder, the RCKL framework lifts the original system into a high-dimensional space by augmenting the state with nonlinear features. The mean and covariance estimates of robot poses and landmark positions are obtained through optimization, with the optimal solution approximated

as a high-dimensional Gaussian. As discussed in Section 6.7, feature constraints are often necessary when the model subspace is not Koopman-invariant. We solve for the constrained mean estimates using an SQP solver, then solve for the constrained covariance estimates via the expression (6.35). This expression was derived specifically for constrained weighted least squares, where the covariance appears as a block of the inverse KKT matrix [43] (see Appendix C.1.5 for details).

We now make the key observation: the covariance expression (6.35) from Chapter 6 is equivalent to the conditioning expression (7.1.4) derived in this chapter. This equivalence is natural and admits a simple interpretation. RCKL-SLAM's covariance formula corresponds to conditioning the unconstrained covariance, obtained from a Gaussian approximation of the distribution at the constrained mean, onto the linearized constraints. As mentioned in Section 7.6, this corresponds to Laplace's approximation. More generally, this interpretation extends beyond constrained weighted least squares to any setting in which the unconstrained solution is approximated as a Gaussian.

From this example, we also see that covariance computation can remain efficient when $\mathbf{\Sigma}^{-1}$ and $\mathbf{Z}$ in (7.1.4) exhibit exploitable structure. In RCKL-SLAM, $\mathbf{\Sigma}^{-1}$ for the long trajectory in Fig. 7.6 has dimensions on the order of $50{,}000 \times 50{,}000$, making direct inversion impractical. By leveraging its block-tridiagonal structure along with the structure of $\mathbf{Z}$, we obtain a sparse solver that scales linearly with the number of timesteps. See Appendix C.1.4 for details. As a result, the covariance for the 200-second trajectory in Fig. 7.6 is computed in about 10 seconds.

RCKL-SLAM Experiment

We verify the validity of RCKL-SLAM's covariance estimation on the experimental dataset "Lost in the Woods" from [32]. This is the same setup as Experiment 1 (Section 6.9.3) in the previous chapter, but we use different trajectory segments and landmark selections to better evaluate the quality of the resulting covariance estimates. As a reminder, the wheeled robot operates in an indoor 2D environment containing 17 cylindrical landmarks. The robot measures translational and rotational velocity using a wheel odometer and yaw-rate gyroscope, and measures landmark ranges using a laser rangefinder. Groundtruth trajectories are recorded using a Vicon motion-capture system.

As in the previous chapter, the 20-minute dataset is split into training and testing since RCKL-SLAM is data-driven. Models are trained using $5/6$ of the trajectories and measurements from 14 landmarks, with data augmentation applied through translational and rotational transformations [97]. Testing uses the remaining $1/6$ of the trajectories and measurements from the remaining 3 landmarks.

Unlike the previous chapter, the test landmarks selected here are clustered in the right half of the environment. The test trajectory is a 200-second segment containing two traversals into the left half of the environment, where the robot is farther from the landmarks and measurements are sparser. This setup allows us to evaluate the covariance estimates under varying landmark densities and measurement availability. To evaluate consistency, we use d_{traj}, the normalized trajectory-level Mahalanobis distance, defined in (6.46).

The covariance estimates of RCKL-SLAM are fairly consistent. Quantitatively, $d_{\text{traj}} = 1.50$, indicating that the estimator is slightly overconfident. For comparison, we use the model-based baseline MB-SLAM from the previous chapter, which corresponds to classical nonlinear batch SLAM optimized using Gauss-Newton [1, §8, 9]. For this trajectory, MB-SLAM yields $d_{\text{traj}} = 1.85$, indicating that it is more overconfident, and therefore less consistent, than RCKL-SLAM. Qualitatively, the

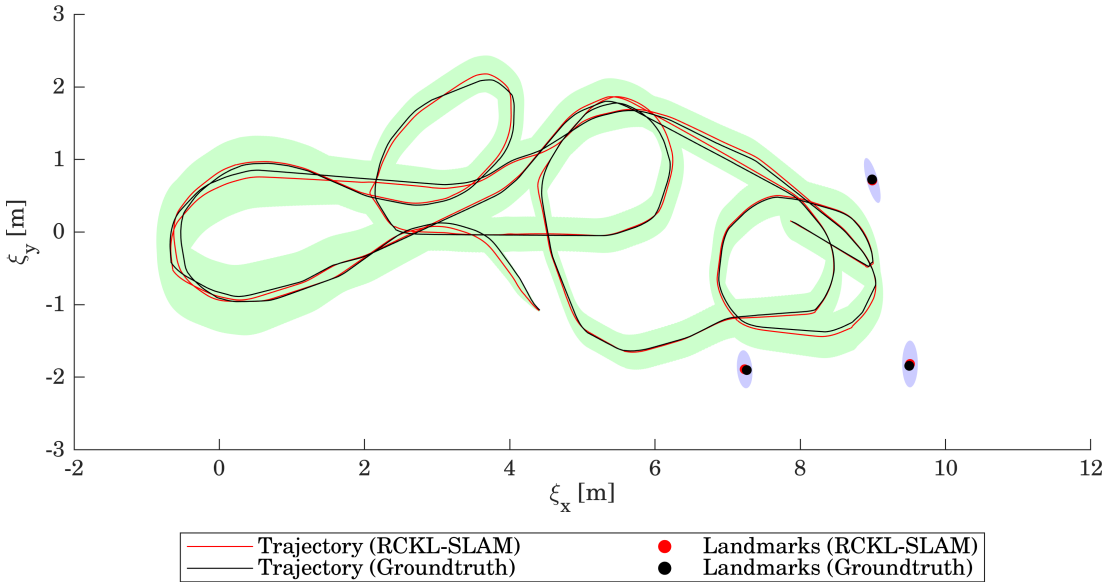


Figure 7.6: Visualization of the RCKL-SLAM output, showing its mean states and mean landmark positions compared to the groundtruth. The green regions and the grey regions are, respectively, the 3σ covariances of the trajectory and of the landmarks. The trajectory and landmark estimates are generally within the estimated 3σ bounds, suggesting that the covariances are consistent.

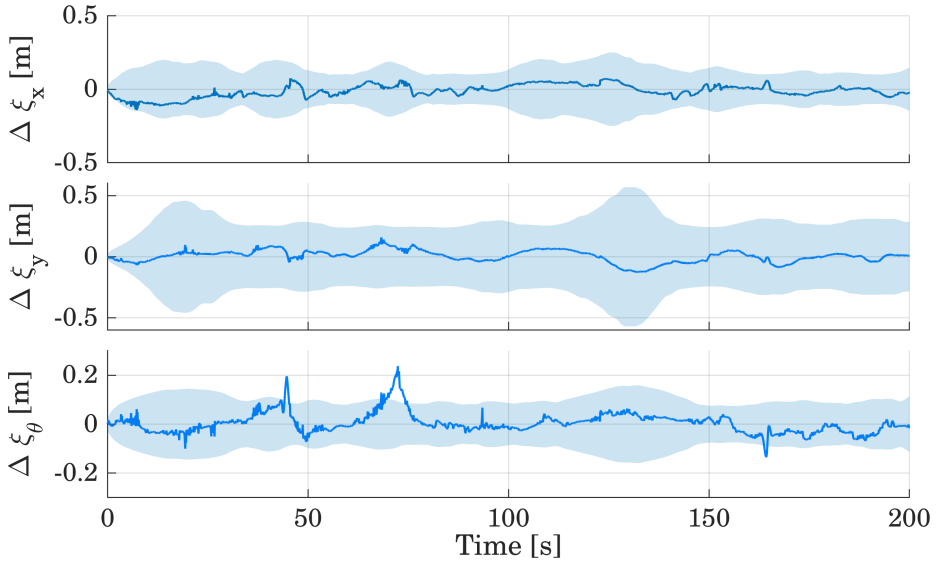


Figure 7.7: Error plots for RCKL-SLAM. The solid lines represent the errors of the estimated trajectories, and the shaded regions represent the estimated 3σ bounds. The errors are generally within the 3σ bounds, suggesting that the covariance estimates are consistent.

RCKL-SLAM estimates are generally within the 3σ bounds in both Fig. 7.6 and Fig. 7.7. Notably, the trajectory covariances are larger when the robot is farther away from the landmarks. In Fig. 7.6, the trajectory on the left half has larger covariances than that on the right half. In Fig. 7.7, the two trips to the left half at around 20 seconds and 130 seconds led to larger covariances in y. This behavior supports the consistency of our covariance estimates, since when the landmarks are farther away, measurements are sparser and distances are less reliable for triangulating robot positions.

7.7.3 Covariance Extraction in Constrained GTSAM

Our final application considers covariance extraction for constrained GTSAM algorithms, representing a more practical robotics setting than the previous two applications. Unlike in the previous two examples, here our conditioning framework enables covariance extraction in a setting where it was previously unavailable. We first introduce the constrained GTSAM formulation, then present an experiment evaluating the consistency of the resulting covariance estimates.

Constrained GTSAM Formulation

On-manifold optimizers are prevalent in robotics but often make covariance extraction difficult. This is the case for constrained GTSAM algorithms including Incremental Constrained Smoothing (ICS) [40] and Incremental Constrained Optimization (InCOpt) [41]. The original unconstrained GTSAM [30] performs incremental smoothing and mapping by updating the affected factors in the factor graph as new information arrives, effectively updating the batch information matrix. As a result, the unconstrained covariance can be computed as a byproduct of solving for the mean [86]. Constrained GTSAM, however, no longer directly provides covariance estimates due to the added hard constraints. Using the framework developed in this chapter, we can restore this capability by conditioning the unconstrained covariance onto the linearized constraints. Similar to Section 7.7.2, we apply (7.1.4), where $\mathbf{\Sigma}^{-1}$ is the unconstrained batch information matrix and $\mathbf{Z}$ spans the nullspace of $\mathbf{S}$, the linearized constraints.

This procedure can also be implemented efficiently. Both $\mathbf{\Sigma}^{-1}$ and $\mathbf{S}$ are already maintained while solving for the state mean, allowing $\mathbf{\Sigma}_{\text{cond}}$ to be computed largely as a byproduct. Moreover, constrained localization and SLAM problems often involve structured constraints, such as constraints that do not couple distant timesteps. In these settings, the SLAM information matrix has an arrowhead structure [1], while $\mathbf{S}$ and $\mathbf{Z}$ are block diagonal. Consequently, sparse solvers similar to those used in RCKL-SLAM can be used to efficiently compute $\mathbf{\Sigma}_{\text{cond}}$ in (7.1.4).

Constrained GTSAM Experiment

We evaluate our covariance estimates of InCOpt using the "2D Planar Pushing" scenario described in [41]. A circular probe pushes a box across a plane. The box is constrained to be in permanent contact with the probe. Given noisy box odometry, noisy contact measurements, and a perfectly known trajectory of the probe, InCOpt estimates the mean trajectory of the box. Then, we use (7.1.4) to extract the trajectory covariance. See Fig. 7.8 for a visualization.

To evaluate consistency, we use d_{traj} from (6.46). Note that d_{traj} cannot be computed in $SE(2)$ space since $\mathbf{\Sigma}_{\text{cond}}$ is rank-deficient. Instead, we compute d_{traj} in $T_{\tilde{\boldsymbol{\mu}}}\mathcal{M}$, where $\mathbf{\Sigma}_{\text{cond}}^{-1}$ exists. We also need $\delta\boldsymbol{\zeta}$, the error between the groundtruth and the mean, in $T_{\tilde{\boldsymbol{\mu}}}\mathcal{M}$. We first compute the error in $\mathcal{M}$, then map this error from $\mathcal{M}$ to $T_{\tilde{\boldsymbol{\mu}}}\mathcal{M}$ through an inverse retraction method. In this case, $\mathcal{M}$ can be written as a kinematic chain [118] consisting of a revolute joint (α) followed by a prismatic joint (d), assuming that the box does not slide past a corner. We parametrize poses on $\mathcal{M}$ with $\mathbf{x} = \mathbf{h}(\alpha, d)$. Then, $T_{\tilde{\boldsymbol{\mu}}}\mathcal{M}$ is parametrized with $\frac{\partial \mathbf{h}}{\partial \alpha}|_{\tilde{\alpha},\tilde{d}}$ and $\frac{\partial \mathbf{h}}{\partial d}|_{\tilde{\alpha},\tilde{d}}$. To find $\delta\boldsymbol{\zeta}$, we first write the error on $\mathcal{M}$ in terms of $(\Delta\alpha, \Delta d)$, then map this error onto $T_{\tilde{\boldsymbol{\mu}}}\mathcal{M}$. With this, we can compute d_{traj}.

Qualitatively, we see in Fig. 7.8 that the estimated covariance aligns with the geometry of the contact constraint. The uncertainty is concentrated primarily along the tangential direction of the

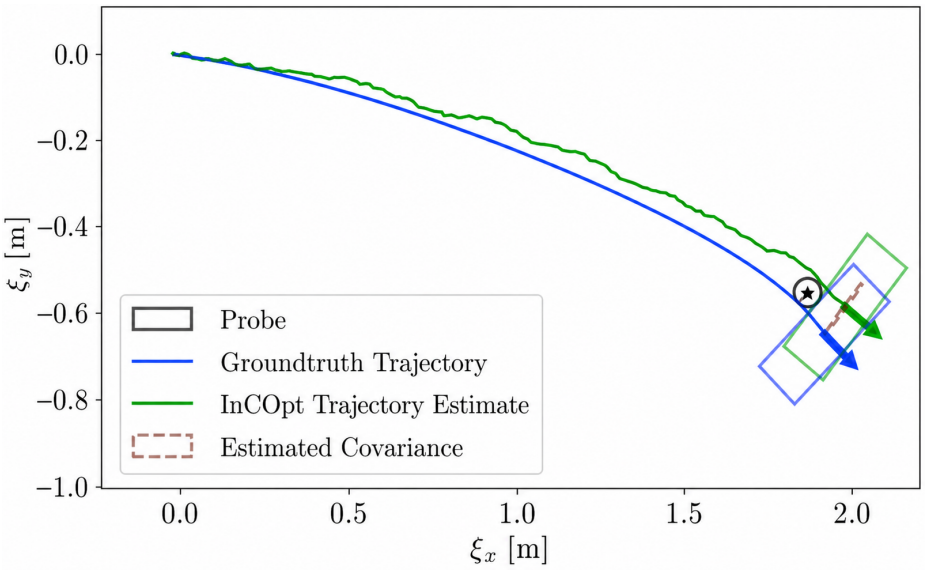


Figure 7.8: Visualization of the InCOpt (constrained GTSAM) output. The arrows represent the box's end poses. Throughout the trajectory, the box is constrained to remain in contact with the circular probe. The estimated covariance is visualized using its 3σ contour. Due to the contact constraint, the covariance ellipse is highly eccentric, appearing nearly one-dimensional.

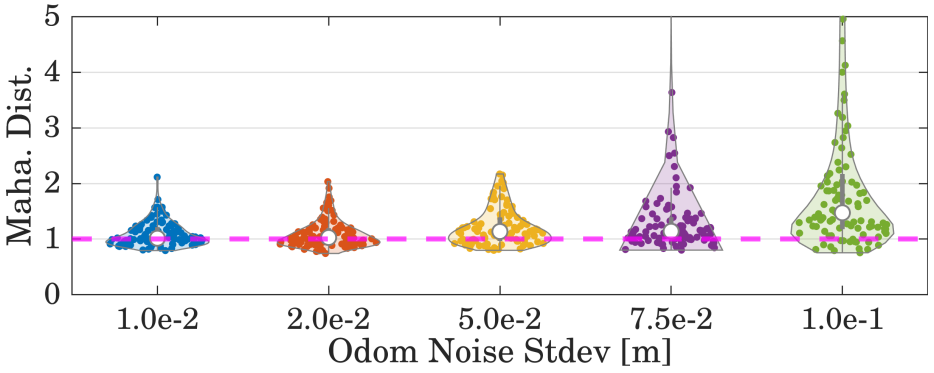


Figure 7.9: Mahalanobis distances (d_{traj}) of 100 trials of InCOpt at different odometry noise levels. At lower noises levels, $d_{\text{traj}} \approx 1$, signifying that the covariance estimates are consistent. As the noise level increases, $d_{\text{traj}} > 1$ more often, signifying that the covariance estimates are generally more overconfident.

contact surface, resulting in a highly eccentric covariance ellipse. This behavior is consistent with the intuition that the contact constraint removes most of the positional uncertainty orthogonal to the probe surface. Interestingly, the positional covariance does not collapse completely in this direction. Although the box position is constrained by contact, uncertainty in the box orientation still induces small positional variations normal to the surface. When visualized through the marginal position covariance in the Cartesian plane, these variations produce a small but nonzero covariance in the orthogonal direction. Thus, while the full constrained covariance remains rank deficient in the ambient state space, its Cartesian position marginal retains a small but nonzero variance in the normal direction rather than collapsing to a line. For a visualization of this scenario in action, see the video at https://youtu.be/O84uAMSZ-JI?si=KIAL3R--sj1NLYko&t=205.

For quantitative evaluation, we repeat the experiment for different levels of odometry noise, with 100 simulated trajectories at each noise level, and measure d_{traj}. In Fig. 7.9, we see that

the covariance estimates are fairly consistent at lower noise levels, but become overconfident at higher noise levels. This trend is expected; larger noise is more amplified by the nonlinearity of the estimation problem, and thus decreases the quality of our linear approximation. Nevertheless, our approximation is sufficiently consistent at lower noise levels.

7.8 Conclusion

This chapter extends the application of marginalization and conditioning from axis-aligned manifolds to general smooth manifolds. It provides closed-form expressions for marginalizing and conditioning Gaussians onto general linear manifolds, and proposes a tangent-plane approximation for nonlinear manifolds. We have presented three applications of our theoretical contributions: approximating the projected normal distribution, covariance extraction in Koopman-inspired SLAM, and covariance extraction in constrained GTSAM.

For future work, more accurate techniques could be developed for marginalizing and conditioning onto nonlinear manifolds, possibly with sigma points [119, 120]. Covariance extraction in applications with constrained settings has received limited attention to date, and perhaps this work could stimulate further exploration.

More broadly, this chapter places the covariance estimation methods used in Chapter 6 within a more general probabilistic framework. What initially arose as a practical question of uncertainty estimation for constrained lifted SLAM led to a broader interpretation as Gaussian marginalization and conditioning onto linearized manifolds. The resulting framework extends beyond the specific estimation problems considered in this thesis and may prove useful in other applications involving equality-constrained optimization and inference.

Chapter 8

Conclusion

In this thesis, we investigated how learned models can be incorporated into probabilistic state estimation while preserving the structure that enables efficient and principled inference. Classical estimation methods rely heavily on process and measurement models, yet deriving such models analytically can be difficult in many robotics applications due to complex sensing modalities, nonlinear dynamics, and modeling uncertainties. While data-driven approaches offer an attractive alternative, learned models are often difficult to integrate into existing estimation frameworks because they may not preserve the structure exploited by classical inference algorithms.

To address this challenge, this thesis explored Koopman-inspired lifting as a framework for combining learned representations with probabilistic state estimation. By learning lifted representations that preserve useful inference structure, the proposed methods retain many of the mechanisms underlying classical estimation, including efficient system identification and inference as well as principled uncertainty quantification. We first demonstrated that learned lifted measurement models can be incorporated into classical filtering architectures while preserving their underlying inference structure. We then extended lifting to both process and measurement models, enabling data-driven batch state estimation with learned dynamics and observations. Building on these ideas, we developed lifted localization and SLAM frameworks, uncovering and addressing new challenges associated with large-scale estimation in lifted spaces. These challenges raised fundamental questions regarding uncertainty estimation in constrained systems, motivating a general framework for conditioning and marginalizing Gaussian distributions onto linearized manifolds. The resulting theory provides a probabilistic interpretation of uncertainty estimation in equality-constrained optimization problems and extends classical Gaussian inference to constrained estimation settings.

More broadly, the results of this thesis suggest that learned models and probabilistic estimation need not be viewed as competing paradigms. By designing learned representations that preserve useful inference structure, it is possible to combine the flexibility of data-driven modeling with the strengths of classical estimation, including computational efficiency, interpretability, and principled uncertainty quantification. We hope that the ideas developed in this thesis help bridge the gap between data-driven modeling and probabilistic state estimation, and encourage further exploration of structured learning approaches for robotics.

8.1 Future Work

Throughout this thesis, lifted representations are constructed using handcrafted feature classes and large collections of random features. An important direction for future work is the development of more principled methods for selecting or learning informative lifted features. Sparse model discovery or feature selection methods such as SINDy [68] may help identify compact lifted representations that improve efficiency and estimation performance. Such methods may also improve interpretability by identifying the features most relevant for estimation.

Another interesting direction is understanding how lifting affects the optimization landscape of estimation problems. The lifted localization and SLAM experiments of Chapter 6 suggest that lifted formulations may be less susceptible to poor local minima than their nonlinear counterparts. This raises a broader question regarding the role of lifting in nonlinear estimation. A deeper theoretical understanding of when and why this occurs, and whether lifting can provide stronger guarantees regarding global optimality, remains an open problem. One possible connection is to certifiable estimation [121], where related lifting-based relaxations can sometimes provide guarantees of global optimality. It would be worthwhile to examine whether Koopman-inspired lifting can admit similar guarantees.

A further open question is how much lifted structure is required to obtain the benefits of Koopman-inspired estimation. Chapter 6 explores one extreme through the RCKL framework, which propagates only the original state variables rather than the full lifted state. While this improves robustness and computational efficiency, it suggests a tradeoff between retaining lifted dynamics and maintaining estimation stability. Future work could investigate hybrid formulations that retain selected lifted variables within the process model, potentially combining the modeling flexibility of full lifted dynamics with the robustness of reduced lifted representations.

Finally, Chapter 7 focuses on equality-constrained estimation problems. Extending the proposed framework to inequality constraints remains an open question. Such constraints arise naturally in many robotics applications and would significantly expand the scope of the proposed inference framework. More broadly, probabilistic inference for constrained systems remains relatively unexplored and warrants further study. Together, these directions point toward estimation frameworks that are not only accurate, but also sufficiently reliable in the presence of real-world noise.

Appendix A

Chapter 4 Supplementary Materials

A.1 Measurement Jacobian for $\mathrm{SE}_2(3)$

This section is referenced in Section 4.4.1. We derive the measurement Jacobian used in the EKF measurement update. Dropping the timestep index k, consider a measurement model of the form $\boldsymbol{\gamma} = \mathbf{g}(\boldsymbol{\xi}, \boldsymbol{\eta})$ defined in (3.1b), with the state quantities defined in (4.2) and the error-state perturbation defined in (4.4). Following the standard right-multiplicative EKF formulation on Lie groups, we linearize the measurement model about a nominal state $\bar{\boldsymbol{\xi}} = (\bar{\mathbf{T}}, \bar{\mathbf{b}})$ and zero measurement noise:

$$\boldsymbol{\gamma} = \bar{\boldsymbol{\gamma}} + \delta\boldsymbol{\gamma}, \quad \delta\boldsymbol{\gamma} = \mathbf{g}(\boldsymbol{\xi}, \mathbf{0}) - \mathbf{g}(\bar{\boldsymbol{\xi}}, \mathbf{0}) \approx \mathbf{H}(\bar{\boldsymbol{\xi}})\, \delta\boldsymbol{\xi}, \tag{A.1}$$

where the measurement Jacobian is defined as

$$\mathbf{H}(\bar{\boldsymbol{\xi}}) = \left.\frac{\partial \delta\boldsymbol{\gamma}}{\partial \delta\boldsymbol{\xi}}\right|_{\delta\boldsymbol{\xi}=\mathbf{0}} = \left.\frac{\partial}{\partial \delta\boldsymbol{\xi}}(\mathbf{g}(\bar{\boldsymbol{\xi}} \oplus \delta\boldsymbol{\xi}, \mathbf{0}) - \mathbf{g}(\bar{\boldsymbol{\xi}}, \mathbf{0}))\right|_{\delta\boldsymbol{\xi}=\mathbf{0}}. \tag{A.2}$$

Dropping the $\bar{(\cdot)}$, the Jacobian can be written in block form as

$$\mathbf{H}(\boldsymbol{\xi}) = \begin{bmatrix} \mathbf{H}_\theta & \mathbf{H}_v & \mathbf{H}_r & \mathbf{H}_b \end{bmatrix} \in \mathbb{R}^{m \times 15}, \tag{A.3}$$

with

$$\mathbf{H}_\theta = \frac{\partial \mathbf{g}}{\partial \mathrm{vec}(\mathbf{C})} \begin{bmatrix} \mathrm{vec}(\mathbf{C}\mathbf{S}_x) & \mathrm{vec}(\mathbf{C}\mathbf{S}_y) & \mathrm{vec}(\mathbf{C}\mathbf{S}_z) \end{bmatrix} \in \mathbb{R}^{m \times 3}, \tag{A.4}$$

$$\mathbf{H}_v = \frac{\partial \mathbf{g}}{\partial \mathbf{v}} \in \mathbb{R}^{m \times 3}, \quad \mathbf{H}_r = \frac{\partial \mathbf{g}}{\partial \mathbf{t}}\, \mathbf{C} \in \mathbb{R}^{m \times 3}, \quad \mathbf{H}_b = \frac{\partial \mathbf{g}}{\partial \mathbf{b}} \in \mathbb{R}^{m \times 6}. \tag{A.5}$$

Here, $\mathbf{S}_x$, $\mathbf{S}_y$, and $\mathbf{S}_z$ denote the standard skew-symmetric basis matrices of $\mathfrak{so}(3)$ [33].

In the common case where the measurement depends only on the pose variables $(\mathbf{C}, \mathbf{t})$, the Jacobian reduces to

$$\mathbf{H}(\boldsymbol{\xi}) = \begin{bmatrix} \mathbf{H}_\theta & \mathbf{H}_r \end{bmatrix} \mathbf{E}, \quad \mathbf{E} = \begin{bmatrix} \mathbf{1}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3 & \mathbf{0}_3 \end{bmatrix}, \tag{A.6}$$

where $\mathbf{E}$ selects the orientation and position components of the error state.

Appendix B

Chapter 5 Supplementary Materials

B.1 Detailed Sketch that KoopSE is Kernelized

This section is referenced in Section 5.8.1. Our goal is to show that KoopSE uses the RKHS quantities only in their kernelized forms. For training, the lifted training points are used to compute the bilinear system matrices in (5.11). Using SMW identities, we can write the analytical solution of (5.20) for $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{H}$, as well as an alternative expression for $\mathbf{D}$:

$$\mathbf{A} = \frac{1}{\tau_A}\mathbf{X}_{(+)}\mathbf{L}\mathbf{X}_{(-)}^\top, \quad \mathbf{B} = \frac{1}{\tau_B}\mathbf{X}_{(+)}\mathbf{L}\mathbf{U}^\top, \tag{B.1a}$$

$$\mathbf{H} = \frac{1}{\tau_H}\mathbf{X}_{(+)}\mathbf{L}\mathbf{V}^\top, \quad \mathbf{D} = \mathbf{Y}\left(\mathbf{X}_{(+)}^\top\mathbf{X}_{(+)} + \tau_D\mathbf{1}\right)^{-1}\mathbf{X}_{(+)}^\top, \tag{B.1b}$$

where $\mathbf{L} = \left(\frac{1}{\tau_A}\mathbf{X}_{(-)}^\top\mathbf{X}_{(-)} + \frac{1}{\tau_B}\mathbf{U}^\top\mathbf{U} + \frac{1}{\tau_H}\mathbf{V}^\top\mathbf{V} + \mathbf{1}\right)^{-1}$. This solution is not used in practice because it involves inverting $P \times P$ matrices, where P is the amount of training data. Since $\mathbf{X}_{(-)}$ and $\mathbf{X}_{(+)}$ are both constructed from lifted training states and typically span similar RKHS subspaces, we drop the distinction between them and use $\mathbf{X}$ for notational simplicity in the remainder of this proof. Then, the solutions can be written in the generic kernelized form:

$$\mathbf{A} = \mathbf{X}\mathbf{W}_A\mathbf{X}^\top, \qquad \mathbf{B} = \mathbf{X}\mathbf{W}_B\mathbf{U}^\top, \tag{B.2a}$$

$$\mathbf{H} = \mathbf{X}\mathbf{W}_H\mathbf{V}^\top, \qquad \mathbf{D} = \mathbf{Y}\mathbf{W}_D\mathbf{X}^\top, \tag{B.2b}$$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q\mathbf{X}^\top + \tau_Q\mathbf{1}, \quad \mathbf{R} = \mathbf{Y}\mathbf{W}_R\mathbf{Y}^\top + \tau_R\mathbf{1}, \tag{B.2c}$$

where $\mathbf{W}_A, \mathbf{W}_B, \mathbf{W}_H, \mathbf{W}_D, \mathbf{W}_Q, \mathbf{W}_R$ are some kernelized matrices.

At test time, we assume that the initial condition, new control inputs, and new measurements can be written as linear combinations of those seen in training:

$$\check{\mathbf{x}}_0 = \mathbf{X}\mathbf{w}_{\check{x},0}, \tag{B.3a}$$

$$\mathbf{u}_k = \mathbf{U}\mathbf{w}_{u,k}, \quad k = 1, \ldots, K, \tag{B.3b}$$

$$\mathbf{y}_k = \mathbf{Y}\mathbf{w}_{y,k}, \quad k = 0, \ldots, K \tag{B.3c}$$

for weights $\mathbf{w}_{\check{x},0}$, $\mathbf{w}_{u,k}$, and $\mathbf{w}_{y,k}$. Looking at the definition of $\mathbf{A}_{k-1}$ in (5.23), we notice that

$$\begin{aligned}
\mathbf{H}(\mathbf{u}_k \otimes \mathbf{1}) &= \mathbf{X}\mathbf{W}_H(\mathbf{U} \odot \mathbf{X})^\top(\mathbf{u}_k \otimes \mathbf{1}) && \text{(B.4a)}\\
&= \mathbf{X}\mathbf{W}_H \begin{bmatrix} (\mathbf{u}_1^\top \otimes \mathbf{x}_1^\top)(\mathbf{u}_k \otimes \mathbf{1}) \\ \vdots \\ (\mathbf{u}_P^\top \otimes \mathbf{x}_P^\top)(\mathbf{u}_k \otimes \mathbf{1}) \end{bmatrix} && \text{(B.4b)}\\
&= \mathbf{X}\mathbf{W}_H \begin{bmatrix} ((\mathbf{u}_1^\top \mathbf{u}_k) \otimes \mathbf{1})\mathbf{x}_1^\top \\ \vdots \\ ((\mathbf{u}_P^\top \mathbf{u}_k) \otimes \mathbf{1})\mathbf{x}_P^\top \end{bmatrix} && \text{(B.4c)}\\
&= \mathbf{X}\mathbf{W}_H \begin{bmatrix} (\mathbf{u}_1^\top \mathbf{u}_k) \otimes \mathbf{1} & & \\ & \ddots & \\ & & (\mathbf{u}_P^\top \mathbf{u}_k) \otimes \mathbf{1} \end{bmatrix} \mathbf{X}^\top && \text{(B.4d)}\\
&= \mathbf{X}\mathbf{W}_{H,k}\mathbf{X}^\top, && \text{(B.4e)}
\end{aligned}$$

where $\mathbf{W}_{H,k}$ is the product of $\mathbf{W}_H$ and the kernelized block diagonal matrix in (B.4d). Then, for $k = 0, \ldots, K$,

$$\begin{aligned}
\mathbf{A}_{k-1} &= \mathbf{A} + \mathbf{H}(\mathbf{u}_k \otimes \mathbf{1}) && \text{(B.5a)}\\
&= \mathbf{X}\mathbf{W}_A\mathbf{X}^\top + \mathbf{X}\mathbf{W}_{H,k}\mathbf{X}^\top && \text{(B.5b)}\\
&= \mathbf{X}\mathbf{W}_{A,k}\mathbf{X}^\top, && \text{(B.5c)}
\end{aligned}$$

$$\begin{aligned}
\mathbf{v}_k &= \mathbf{B}\mathbf{u}_k && \text{(B.6a)}\\
&= (\mathbf{X}\mathbf{W}_B\mathbf{U}^\top)(\mathbf{U}\mathbf{w}_{u,k}) && \text{(B.6b)}\\
&= \mathbf{X}\mathbf{w}_{v,k}, && \text{(B.6c)}
\end{aligned}$$

where

$$\mathbf{W}_{A,k} = \mathbf{W}_A + \mathbf{W}_{H,k}, \quad \mathbf{w}_{v,k} = \mathbf{W}_B(\mathbf{U}^\top\mathbf{U})\mathbf{w}_{u,k} \tag{B.7}$$

are clearly kernelized. Now, the solution to (5.24) is exactly solved by the RTS smoother [1]. We can then look at the structure of the estimated state means and covariances of the forward pass, then of backward pass, through induction. We outline the setup below. Let $\{\check{\mathbf{x}}_{k,f}, \check{\mathbf{P}}_{k,f}\}_{k=0}^K$, $\{\hat{\mathbf{x}}_{k,f}, \hat{\mathbf{P}}_{k,f}\}_{k=0}^K$, and $\{\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_k\}_{k=0}^K$ denote the means and covariances of the forward pass prior (i.e., dead reckoning), forward pass posterior (i.e., Kalman filter), and the backward pass (i.e., smoother solutions), respectively. For the forward pass, the initial condition is

$$\check{\mathbf{x}}_0 = \mathbf{X}\mathbf{w}_{\check{x},0}, \quad \check{\mathbf{P}}_0 = \mathbf{Q} = \mathbf{X}\mathbf{W}_Q\mathbf{X}^\top + \tau_Q\mathbf{1}. \tag{B.8}$$

Suppose at timestep k, the priors have the form

$$\check{\mathbf{x}}_k = \mathbf{X}\mathbf{w}_{\check{x},k,f}, \quad \check{\mathbf{P}}_k = \mathbf{X}\mathbf{W}_{\check{P},k,f}\mathbf{X}^\top + \check{c}_{k,f}\mathbf{1}, \tag{B.9}$$

for some kernelized matrices $\mathbf{w}_{\check{x},k,f}$ and $\mathbf{W}_{\check{P},k,f}\mathbf{X}^\top$ and scalar $\check{c}_{k,f}$. Then, going through the forward pass and using the derived kernelized structures for $\mathbf{A}_{k-1}$, $\mathbf{D}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{v}_k$, and $\mathbf{y}_k$, it is straightforward to show that

$$\hat{\mathbf{x}}_k = \mathbf{X}\mathbf{w}_{\hat{x},k,f}, \quad \hat{\mathbf{P}}_k = \mathbf{X}\mathbf{W}_{\hat{P},k,f}\mathbf{X}^\top + \check{c}_{k,f}\mathbf{1}, \tag{B.10}$$

and then that

$$\check{\mathbf{x}}_{k+1} = \mathbf{X}\mathbf{w}_{\check{x},k+1,f}, \quad \check{\mathbf{P}}_{k+1} = \mathbf{X}\mathbf{W}_{\check{P},k+1,f}\mathbf{X}^\top + \check{c}_{k+1,f}\mathbf{1}, \tag{B.11}$$

with the same matrix structures. The proof setup is similar for the backward pass. Through these induction processes, we can show that the mean and covariance outputs from the forward pass prior, forward pass posterior, and the backward pass all have the structure, for $k = 0, \ldots, K$,

$$\hat{\mathbf{x}}_k = \mathbf{X}\mathbf{w}_{\hat{x},k}, \quad \hat{\mathbf{P}}_k = \mathbf{X}\mathbf{W}_{\hat{P},k}\mathbf{X}^\top + c_k\mathbf{1}, \tag{B.12}$$

with kernelized matrices $\mathbf{w}_{\hat{x},k}$ and $\mathbf{W}_{\hat{P},k}$ and scalar c_k. We substitute these expressions for $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{P}}_k$ into (5.29), yielding

$$\hat{\boldsymbol{\xi}}_k = \boldsymbol{\Xi}(\mathbf{X}^\top\mathbf{X} + \tau_x\mathbf{1})^{-1}\mathbf{X}^\top(\mathbf{X}\mathbf{w}_{x,k}), \tag{B.13a}$$

$$\hat{\boldsymbol{\Sigma}}_k = \boldsymbol{\Xi}(\mathbf{X}^\top\mathbf{X} + \tau_x\mathbf{1})^{-1}\mathbf{X}^\top(\mathbf{X}\mathbf{W}_{P,k}\mathbf{X}^\top + c_k\mathbf{1})\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \tau_x\mathbf{1})^{-1}\boldsymbol{\Xi}^\top. \tag{B.13b}$$

We can see that the results for $\hat{\boldsymbol{\xi}}_k$ and $\hat{\boldsymbol{\Sigma}}_k$, the states and covariances in the original space, are indeed kernelized. Therefore, from input to output, the algorithm only uses RKHS quantities in their kernelized forms.

Appendix C

Chapter 6 Supplementary Materials

C.1 Solving UKL-Loc with the RTS smoother

This section is referenced in Section 6.6.1. Here, we outline the process of applying the RTS smoother to the system in (6.17) to solve UKL-Loc. This may be a more efficient solution process than the sparse Cholesky solver. Note that the two methods are algebraically equivalent [1, §3].

With the landmarks given, we convert the measurement model to be linear-Gaussian using the same trick as in the process model. Observe that

$$\boldsymbol{\ell}_j \otimes \mathbf{x}_k = (\boldsymbol{\ell}_j \otimes \mathbf{1})\mathbf{x}_k = (\mathbf{1} \otimes \mathbf{x}_k)\boldsymbol{\ell}_j. \tag{C.1}$$

With this, we can rewrite the measurement equation at each timestep k by stacking the measurements, $\mathbf{y}_k$, and defining matrices, $\mathbf{D}'$ and $\mathbf{R}'$ as

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{y}_{k,1}^\top & \cdots & \mathbf{y}_{k,V}^\top \end{bmatrix}^\top, \quad \mathbf{D}' = \begin{bmatrix} (\mathbf{D}(\boldsymbol{\ell}_1 \otimes \mathbf{1}))^\top & \cdots & (\mathbf{D}(\boldsymbol{\ell}_V \otimes \mathbf{1}))^\top \end{bmatrix}^\top, \quad \mathbf{R}' = \operatorname{diag}(\mathbf{R}, \ldots, \mathbf{R}), \tag{C.2}$$

where $\mathbf{y}_k$ is filled with known values of the landmarks and $\mathbf{R}'$ is $\mathbf{R}$ stacked diagonally V times. The system in (6.17) can then be written as

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k, \qquad k = 1, \ldots, K, \tag{C.3a}$$

$$\mathbf{y}_k = \mathbf{D}'\mathbf{x}_k + \mathbf{n}_k, \qquad k = 0, \ldots, K, \tag{C.3b}$$

where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}')$. If any measurement is missing, we can make $\mathbf{D}'$ and $\mathbf{R}'$ be time-varying and omit the corresponding entries when constructing $\mathbf{y}_k$, $\mathbf{D}'_k$, and $\mathbf{R}'_k$. In any case, we have converted the bilinear time-invariant system in (6.7) into a linear time-varying (LTV) system. We can then apply the standard RTS smoother [1] to (C.3) to solve for the states in $\mathcal{O}(N^3KV)$ time. The output is $\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_{\mathbf{x},k})$, where $\hat{\mathbf{x}}_k$ is the state mean and $\hat{\mathbf{P}}_{\mathbf{x},k}$ is the covariance at timestep k, for all $k = 0, \ldots, K$. In contrast to KoopSE in Chapter 5, by using a measurement model that is individually applicable to each landmark, we have generalized localization to environments with new landmark positions.

C.1.1 Gauss-Newton Setup for UKL-SLAM

This section is referenced in Section 6.6.2. We use Gauss-Newton optimization to solve (6.23) for $\boldsymbol{x}$ and $\boldsymbol{\ell}$. Given an operating point, $\boldsymbol{q}_{\text{op}}$, the Gauss-Newton approximation of (6.21) yields

$$J(\boldsymbol{q}) = J(\boldsymbol{q}_{\text{op}} + \delta\boldsymbol{q}) \approx J(\boldsymbol{q}_{\text{op}}) - \boldsymbol{g}^\top \delta\boldsymbol{q} + \frac{1}{2}\delta\boldsymbol{q}^\top \boldsymbol{F}\delta\boldsymbol{q}, \tag{C.4}$$

where

$$\boldsymbol{F} = \boldsymbol{H}^\top \boldsymbol{W}^{-1}\boldsymbol{H}, \quad \boldsymbol{g} = \boldsymbol{H}^\top \boldsymbol{W}^{-1}\boldsymbol{e}(\boldsymbol{q}_{\text{op}}), \tag{C.5a}$$

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{A}^{-1} & \boldsymbol{0} \\ \boldsymbol{G}_{\boldsymbol{x}} & \boldsymbol{G}_{\boldsymbol{\ell}} \end{bmatrix}, \quad \boldsymbol{A}^{-1} = \begin{bmatrix} -\mathbf{A}_0 & \mathbf{1} & & \\ & \ddots & \ddots & \\ & & -\mathbf{A}_{K-1} & \mathbf{1} \end{bmatrix}, \tag{C.5b}$$

$$\boldsymbol{G}_{\boldsymbol{x}} = \operatorname{diag}(\mathbf{G}_{\mathbf{x},1}, \ldots, \mathbf{G}_{\mathbf{x},K}), \quad \boldsymbol{G}_{\boldsymbol{\ell}} = \begin{bmatrix} \mathbf{G}_{\boldsymbol{\ell},1}^\top & \cdots & \mathbf{G}_{\boldsymbol{\ell},K}^\top \end{bmatrix}^\top, \tag{C.5c}$$

$$\mathbf{G}_{\mathbf{x},k} = \begin{bmatrix} \mathbf{G}_{\mathbf{x},k,1}^\top & \cdots & \mathbf{G}_{\mathbf{x},k,V}^\top \end{bmatrix}^\top, \quad \mathbf{G}_{\boldsymbol{\ell},k} = \operatorname{diag}(\mathbf{G}_{\boldsymbol{\ell},k,1}, \ldots, \mathbf{G}_{\boldsymbol{\ell},k,V}), \tag{C.5d}$$

where, using (C.1), the measurement Jacobians are

$$\mathbf{G}_{\mathbf{x},k,j} = \left.\frac{\partial \mathbf{y}_{k,j}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op},k}, \boldsymbol{\ell}_{\text{op},j}} = \mathbf{D}(\boldsymbol{\ell}_{\text{op},j} \otimes \mathbf{1}), \qquad \mathbf{G}_{\boldsymbol{\ell},k,j} = \left.\frac{\partial \mathbf{y}_{k,j}}{\partial \boldsymbol{\ell}}\right|_{\mathbf{x}_{\text{op},k}, \boldsymbol{\ell}_{\text{op},j}} = \mathbf{D}(\mathbf{1} \otimes \mathbf{x}_{\text{op},k}). \tag{C.6}$$

Note that $\boldsymbol{F}$ is at least positive semidefinite when $\mathbf{Q}$ and $\mathbf{R}$ are positive definite, and $\boldsymbol{F}$ is positive definite when the lifted system is observable [1]. Observability of the lifted system depends on the test data and the Jacobians in $\boldsymbol{H}$ [122]. We found that in simulation and experimental testing, when UKL has a good initialization, $\boldsymbol{F}$ is positive definite and successful convergence of the Gauss-Newton algorithm is realized. See Section 6.7.3 for an initialization procedure. This empirically suggests that when the original system is observable, the lifted system is also observable once given good priors in (6.14b), reasonable lifting functions, and sufficient training data. An interesting avenue for future work is explicitly enforcing observability when learning $\mathbf{A}$, $\mathbf{B}$, $\mathbf{H}$, and $\mathbf{D}$.

The optimal update, $\delta\boldsymbol{q}$, satisfies

$$\boldsymbol{F}\delta\boldsymbol{q} = \boldsymbol{g}, \tag{C.7}$$

where we can solve for $\delta\boldsymbol{q}$ by following the standard approach for classical batch SLAM [1]. As we will see in Appendix C.1.2, the complexity of solving (C.7) is $\mathcal{O}(N^3(V^3 + V^2K))$ when we have more timesteps than landmarks, or $\mathcal{O}(N^3(K^3 + K^2V))$ when we have more landmarks than timesteps.

As mentioned in Section 6.6.4, we can easily accommodate missing or known variables. If $\mathbf{y}_{k,j}$ is missing, we simply remove $\mathbf{e}_{\mathbf{y},k,j}$ from the cost and delete the corresponding blocks in $\boldsymbol{R}^{-1}$, $\mathbf{G}_{\mathbf{x},k}$, and $\mathbf{G}_{\boldsymbol{\ell},k}$. If $\mathbf{x}_k$ or $\boldsymbol{\ell}_j$ is already known, we modify (C.7) to incorporate the known variable by setting $\mathbf{x}_k$ or $\boldsymbol{\ell}_j$ to its value, removing it from the list of variables in $\boldsymbol{q}$, and moving its corresponding equation in $\boldsymbol{F}\delta\boldsymbol{q}$ to the right-hand side as part of $\boldsymbol{g}$.

The CKL-SLAM problem (6.31) contains the same objective function as UKL-SLAM (6.23). As we will see in Appendices C.1.3 and C.1.4, CKL-SLAM's solution process uses the same Gauss-Newton approximation as described above. The case is similar for RCKL-SLAM, where owing to its

slightly modified cost function (6.45), the only required changes are to replace $\mathbf{Q}^{-1} = \begin{bmatrix} \mathbf{Q}_{\boldsymbol{\xi}}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ in (6.22b) and $\mathbf{A}_{k-1} = \begin{bmatrix} \mathbf{A}_{\boldsymbol{\xi},k-1} \\ \mathbf{0} \end{bmatrix}$ in (C.5b).

C.1.2 Solving UKL-SLAM in Linear Time

This section is referenced in Section 6.6.2. Although the linear system in (C.7) is very large, it can be efficiently solved with Cholesky factorization and by exploiting sparsity patterns. $\boldsymbol{F}$ has the structure

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{A}^{-T}\boldsymbol{Q}^{-1}\boldsymbol{A}^{-1} + \boldsymbol{G}_{\boldsymbol{x}}^{\top}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{x}} & \boldsymbol{G}_{\boldsymbol{x}}^{\top}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\ell}} \\ \boldsymbol{G}_{\boldsymbol{\ell}}^{\top}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{x}} & \boldsymbol{G}_{\boldsymbol{\ell}}^{\top}\boldsymbol{R}^{-1}\boldsymbol{G}_{\boldsymbol{\ell}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{\boldsymbol{x}} & \boldsymbol{F}_{\boldsymbol{x}\boldsymbol{\ell}} \\ \boldsymbol{F}_{\boldsymbol{x}\boldsymbol{\ell}}^{\top} & \boldsymbol{F}_{\boldsymbol{\ell}} \end{bmatrix}, \tag{C.8}$$

which exhibits the usual SLAM arrowhead pattern where $\boldsymbol{F}_{\boldsymbol{x}}$ is block-tridiagonal and $\boldsymbol{F}_{\boldsymbol{\ell}}$ is block-diagonal [1]. As discussed in Section 6.6.2, $\boldsymbol{F}$ is usually positive definite. Then, when we have more robot poses than landmarks, we can use the lower-Cholesky decomposition on $\boldsymbol{F}$,

$$\boldsymbol{F} = \underbrace{\begin{bmatrix} \boldsymbol{L}_{\boldsymbol{x}} & \mathbf{0} \\ \boldsymbol{L}_{\boldsymbol{x}\boldsymbol{\ell}} & \boldsymbol{L}_{\boldsymbol{\ell}} \end{bmatrix}}_{\boldsymbol{L}} \underbrace{\begin{bmatrix} \boldsymbol{L}_{\boldsymbol{x}}^{\top} & \boldsymbol{L}_{\boldsymbol{x}\boldsymbol{\ell}}^{\top} \\ \mathbf{0} & \boldsymbol{L}_{\boldsymbol{\ell}}^{\top} \end{bmatrix}}_{\boldsymbol{L}^{\top}} = \begin{bmatrix} \boldsymbol{L}_{\boldsymbol{x}}\boldsymbol{L}_{\boldsymbol{x}}^{\top} & \boldsymbol{L}_{\boldsymbol{x}}\boldsymbol{L}_{\boldsymbol{x}\boldsymbol{\ell}}^{\top} \\ \boldsymbol{L}_{\boldsymbol{x}\boldsymbol{\ell}}\boldsymbol{L}_{\boldsymbol{x}}^{\top} & \boldsymbol{L}_{\boldsymbol{x}\boldsymbol{\ell}}\boldsymbol{L}_{\boldsymbol{x}\boldsymbol{\ell}}^{\top} + \boldsymbol{L}_{\boldsymbol{\ell}}\boldsymbol{L}_{\boldsymbol{\ell}}^{\top} \end{bmatrix}, \tag{C.9}$$

where we can use the fact that $\boldsymbol{F}_{11}$ is block-tridiagonal to efficiently compute the nonzero blocks in $\boldsymbol{L}_{\boldsymbol{x}\boldsymbol{x}}$ (see [1, §3]), and then compute $\boldsymbol{L}_{\boldsymbol{x}\boldsymbol{\ell}}$ and $\boldsymbol{L}_{\boldsymbol{\ell}\boldsymbol{\ell}}$. For the right-hand side of (C.7), $\boldsymbol{g}$ has the structure

$$\boldsymbol{g} = \boldsymbol{H}^{\top}\boldsymbol{W}^{-1}\boldsymbol{e}_{\mathrm{op}} = \begin{bmatrix} \boldsymbol{A}^{-T}\boldsymbol{Q}^{-1}\boldsymbol{e}_{\boldsymbol{v},\mathrm{op}} + \boldsymbol{G}_{\boldsymbol{x}}^{\top}\boldsymbol{R}^{-1}\boldsymbol{e}_{\boldsymbol{y},\mathrm{op}} \\ \boldsymbol{G}_{\boldsymbol{\ell}}^{\top}\boldsymbol{R}^{-1}\boldsymbol{e}_{\boldsymbol{y},\mathrm{op}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_{\boldsymbol{x}} \\ \boldsymbol{g}_{\boldsymbol{\ell}} \end{bmatrix}, \tag{C.10}$$

which we can efficiently compute using the appropriate blocks. The solution process for (C.7) consists of first solving $\boldsymbol{L}\boldsymbol{p} = \boldsymbol{g}$ for a placeholder variable $\boldsymbol{p}$ using forward substitution, then solving $\boldsymbol{L}^{\top}\hat{\boldsymbol{q}} = \boldsymbol{p}$ for $\hat{\boldsymbol{q}}$ using backward substitution. Thus, (C.7) can be solved without ever constructing the large sparse system, instead working only with the required blocks. The exact solution for $\hat{\boldsymbol{q}}$ can be computed in $\mathcal{O}(N^3(V^3+V^2K))$ time. On the other hand, if we have more landmarks than poses, we can use the upper-Cholesky decomposition with a similar procedure, solving in $\mathcal{O}(N^3(K^3+K^2V))$ time. See [1, §9] for an example.

After iterating until convergence, we can also efficiently compute the covariances of the system. To find $\hat{\mathbf{P}}_{\mathbf{x},k}$ and $\hat{\mathbf{P}}_{\boldsymbol{\ell},j}$, we compute the diagonal blocks of $\boldsymbol{F}^{-1}$. Since $\hat{\boldsymbol{P}}_{\boldsymbol{q}} = \boldsymbol{F} = \boldsymbol{L}\boldsymbol{L}^{\top}$ [1], we can use [123] to compute only the blocks of $\boldsymbol{F}^{-1}$ corresponding to the nonzero blocks of $\boldsymbol{L}$. This can be done without raising the computational complexity of SLAM given the sparsity pattern of $\boldsymbol{L}$ present in SLAM problems [124]. The final output is $\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{P}}_{\mathbf{x},k})$, $\boldsymbol{\ell}_j \sim \mathcal{N}(\hat{\boldsymbol{\ell}}_j, \hat{\mathbf{P}}_{\boldsymbol{\ell},j})$.

C.1.3 Formulating the SQP for (R)CKL-SLAM

This section is referenced in Section 6.7.2. We formulate an SQP [28] to solve (6.31). The Lagrangian is $L(\boldsymbol{q}, \boldsymbol{\lambda}) = J(\boldsymbol{q}) - \boldsymbol{\lambda}^\top \boldsymbol{h}(\boldsymbol{q})$, where

$$\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_{\boldsymbol{x}}^\top & \boldsymbol{\lambda}_{\boldsymbol{\ell}}^\top \end{bmatrix}^\top, \quad \boldsymbol{\lambda}_{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{\lambda}_{\mathbf{x},1}^\top & \cdots & \boldsymbol{\lambda}_{\mathbf{x},K}^\top \end{bmatrix}^\top, \quad \boldsymbol{\lambda}_{\boldsymbol{\ell}} = \begin{bmatrix} \boldsymbol{\lambda}_{\boldsymbol{\ell},1}^\top & \cdots & \boldsymbol{\lambda}_{\boldsymbol{\ell},V}^\top \end{bmatrix}^\top \tag{C.11}$$

are the Lagrange multipliers. Using the same matrix structures as (C.5), we can write the Jacobian of the Lagrangian at an operating point, $\boldsymbol{q}_{\text{op}}$ and $\boldsymbol{\lambda}_{\text{op}}$, as $\frac{\partial L}{\partial \boldsymbol{q}^\top}|_{\boldsymbol{q}_{\text{op}}} = -\boldsymbol{g} - \boldsymbol{S}^\top \boldsymbol{\lambda}_{\text{op}}$, where $\boldsymbol{g}$ is defined in (C.5), and $\boldsymbol{S}$ is the Jacobian of the constraints, defined as

$$\boldsymbol{S} = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\right|_{\boldsymbol{q}_{\text{op}}} = \operatorname{diag}(\boldsymbol{S}_{\boldsymbol{x}}, \boldsymbol{S}_{\boldsymbol{\ell}}), \quad \boldsymbol{S}_{\boldsymbol{x}} = \operatorname{diag}(\mathbf{S}_{\mathbf{x},1}, \ldots, \mathbf{S}_{\mathbf{x},K}), \quad \boldsymbol{S}_{\boldsymbol{\ell}} = \operatorname{diag}(\mathbf{S}_{\boldsymbol{\ell},1}, \ldots, \mathbf{S}_{\boldsymbol{\ell},V}), \tag{C.12a}$$

$$\mathbf{S}_{\mathbf{x},k} = \left.\frac{\partial \mathbf{h}_{\mathbf{x}}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op},k}}, \quad \mathbf{S}_{\boldsymbol{\ell},j} = \left.\frac{\partial \mathbf{h}_{\boldsymbol{\ell}}(\boldsymbol{\ell})}{\partial \boldsymbol{\ell}}\right|_{\boldsymbol{\ell}_{\text{op},j}}. \tag{C.12b}$$

Note that $\boldsymbol{S}$ is block-diagonal owing to the blockwise structure of the constraints.

We use a generalized Gauss-Newton approximation of the Hessian of the Lagrangian [43, §2], [99, §3.2], which simply corresponds to the Gauss-Newton approximation of the objective function:

$$\left.\frac{\partial^2 L}{\partial \boldsymbol{q} \partial \boldsymbol{q}^\top}\right|_{\boldsymbol{q}_{\text{op}}} \approx \left(\left.\frac{\partial J}{\partial \boldsymbol{q}^\top}\right|_{\boldsymbol{q}_{\text{op}}}\right)\left(\left.\frac{\partial J}{\partial \boldsymbol{q}}\right|_{\boldsymbol{q}_{\text{op}}}\right) = \boldsymbol{F}, \tag{C.13}$$

where $\boldsymbol{F}$ is defined in (C.5). In contrast to the full Hessian of the Lagrangian, which may be indefinite, the Gauss-Newton approximation is guaranteed to be at least positive semidefinite. The condition for a valid descent direction from the SQP is that the reduced Hessian, or the Hessian projected onto the tangent space of the constraints, is positive definite [28, p. 452]. With our Gauss-Newton approximation, the reduced Hessian is at least positive semidefinite. It is also empirically full rank in our experiments, thus meeting the SQP's requirement. See Appendix C.1.4 for more discussion on this condition.

We now follow the procedure in [28] to set up the SQP system of equations at each iteration as

$$\begin{bmatrix} \boldsymbol{F} & \boldsymbol{S}^\top \\ \boldsymbol{S} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta \boldsymbol{q} \\ -\delta \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{g} + \boldsymbol{S}^\top \boldsymbol{\lambda}_{\text{op}} \\ -\boldsymbol{h} \end{bmatrix}, \tag{C.14}$$

then update the operating point of the primal variable and the multiplier as $\boldsymbol{q}_{\text{op}} \leftarrow \boldsymbol{q}_{\text{op}} + \alpha \delta \boldsymbol{q}$ and $\boldsymbol{\lambda}_{\text{op}} \leftarrow \boldsymbol{\lambda}_{\text{op}} + \alpha \delta \boldsymbol{\lambda}$, with an appropriate step size α. We find α using a backtracking line search with an acceptance criterion [28],

$$\phi_1(\boldsymbol{q}_{\text{op}} + \alpha \delta \boldsymbol{q}, \mu) \leqslant \phi_1(\boldsymbol{q}_{\text{op}}, \mu) + \eta \alpha \left.\frac{\partial J}{\partial \boldsymbol{q}^\top}\right|_{\boldsymbol{q}_{\text{op}}} \delta \boldsymbol{q}, \tag{C.15}$$

where $\eta \in (0, 1)$, $\mu > 0$, and

$$\phi_1(\boldsymbol{q}, \mu) = J(\boldsymbol{q}) + \mu ||\boldsymbol{h}(\boldsymbol{q})||_1 \tag{C.16}$$

is the $\mathcal{L}_1$ merit function.

We can now solve for $(\delta\boldsymbol{q}, \delta\boldsymbol{\lambda})$ and iterate to convergence, yielding $(\boldsymbol{q}^*, \boldsymbol{\lambda}^*)$. We then recover the mean estimates in the original space, $\hat{\boldsymbol{\zeta}}$, by picking off the top blocks of the lifted estimates in $\boldsymbol{q}^*$. Although the linear system in (C.14) is quite large, as we will see in Appendix C.1.4, we can solve it efficiently by exploiting the SLAM arrowhead structure in $\boldsymbol{F}$ and the block-diagonal structure in $\boldsymbol{S}$.

C.1.4 Solving (R)CKL-SLAM in Linear Time

This section is referenced in Section 6.7.2. We efficiently solve the large linear system in (C.14) for $\delta\boldsymbol{q}$ and $\delta\boldsymbol{\lambda}$, and also describe the requirements on the SQP Hessian for valid descent directions. We define $\boldsymbol{t} = \delta\boldsymbol{q}$ and $\boldsymbol{\nu} = \delta\boldsymbol{\lambda}$ for brevity.

We solve the SQP with the nullspace method [28]. This method is recommended for systems with few degrees of freedom in the free variables. For our system, this number simply corresponds to the degrees of freedom of the original system in (6.1), since the rest of the variables are lifted from, and thus constrained by, the original variables. As we will see, (C.14) can be solved in the same time complexity as the unconstrained problem: $\mathcal{O}(N^3(V^3 + V^2K))$ when we have more timesteps than landmarks, or $\mathcal{O}(N^3(K^3 + K^2V))$ when we have more landmarks than timesteps.

We break down $\boldsymbol{t}$ and $\boldsymbol{v}$ as

$$\boldsymbol{t} = \begin{bmatrix} \boldsymbol{t}_{\boldsymbol{x}}^\top & \boldsymbol{t}_{\boldsymbol{\ell}}^\top \end{bmatrix}^\top, \quad \boldsymbol{t}_{\boldsymbol{x}} = \begin{bmatrix} \mathbf{t}_{\mathbf{x},1}^\top & \cdots & \mathbf{t}_{\mathbf{x},K}^\top \end{bmatrix}^\top, \quad \boldsymbol{t}_{\boldsymbol{\ell}} = \begin{bmatrix} \mathbf{t}_{\boldsymbol{\ell},1}^\top & \cdots & \mathbf{t}_{\boldsymbol{\ell},V}^\top \end{bmatrix}^\top, \tag{C.17a}$$

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_{\boldsymbol{x}}^\top & \boldsymbol{v}_{\boldsymbol{\ell}}^\top \end{bmatrix}^\top, \quad \boldsymbol{v}_{\boldsymbol{x}} = \begin{bmatrix} \mathbf{v}_{\mathbf{x},1}^\top & \cdots & \mathbf{v}_{\mathbf{x},K}^\top \end{bmatrix}^\top, \quad \boldsymbol{v}_{\boldsymbol{\ell}} = \begin{bmatrix} \mathbf{v}_{\boldsymbol{\ell},1}^\top & \cdots & \mathbf{v}_{\boldsymbol{\ell},V}^\top \end{bmatrix}^\top. \tag{C.17b}$$

Let $\boldsymbol{Z} = \text{null}(\boldsymbol{S})$ be a matrix whose columns form a basis for the nullspace of $\boldsymbol{S}$, where $\boldsymbol{S}$ is defined in (C.12). Then, $\text{span}(\boldsymbol{Z})$ represents the tangent space of the linearized constraints. Let $\boldsymbol{T}$ be a matrix that completes the basis for $\boldsymbol{Z}$, meaning that the square matrix $\begin{bmatrix} \boldsymbol{Z} | \boldsymbol{T} \end{bmatrix}$ is full rank. Note that one choice for $\boldsymbol{T}$ that contains the desired block-diagonal sparsity structure is simply $\boldsymbol{T} = \boldsymbol{S}^\top$. However, other choices of a block-diagonal $\boldsymbol{T}$ are just as efficient, and fixing $\boldsymbol{T} = \boldsymbol{S}^\top$ in particular does not simplify any future computations. Rather, this restriction could limit a user from freely constructing $\boldsymbol{T}$ to resolve any numerical issues that may arise. Thus, in this section, we kept $\boldsymbol{T}$ as a separate entity from $\boldsymbol{S}^\top$, in the same spirit as how the nullspace method is presented in [28].

Since $\boldsymbol{S}$ is block-diagonal, we can choose a block-diagonal construction of $\boldsymbol{Z}$ and $\boldsymbol{T}$ as

$$\boldsymbol{Z} = \text{diag}(\boldsymbol{Z}_{\boldsymbol{x}}, \boldsymbol{Z}_{\boldsymbol{\ell}}), \quad \boldsymbol{Z}_{\boldsymbol{x}} = \text{diag}(\mathbf{Z}_{\mathbf{x},1}, \ldots, \mathbf{Z}_{\mathbf{x},K}), \quad \boldsymbol{Z}_{\boldsymbol{\ell}} = \text{diag}(\mathbf{Z}_{\boldsymbol{\ell},1}, \ldots, \mathbf{Z}_{\boldsymbol{\ell},V}), \tag{C.18a}$$

$$\boldsymbol{T} = \text{diag}(\boldsymbol{T}_{\boldsymbol{x}}, \boldsymbol{T}_{\boldsymbol{\ell}}), \quad \boldsymbol{T}_{\boldsymbol{x}} = \text{diag}(\mathbf{T}_{\mathbf{x},1}, \ldots, \mathbf{T}_{\mathbf{x},K}), \quad \boldsymbol{T}_{\boldsymbol{\ell}} = \text{diag}(\mathbf{T}_{\boldsymbol{\ell},1}, \ldots, \mathbf{T}_{\boldsymbol{\ell},V}), \tag{C.18b}$$

where

$$\mathbf{Z}_{\mathbf{x},k} = \text{null}(\mathbf{S}_{\mathbf{x},k}), \quad \mathbf{Z}_{\boldsymbol{\ell},j} = \text{null}(\mathbf{S}_{\boldsymbol{\ell},j}), \quad \mathbf{T}_{\mathbf{x},k} = \text{null}(\mathbf{Z}_{\mathbf{x},k}^\top), \quad \mathbf{T}_{\boldsymbol{\ell},j} = \text{null}(\mathbf{Z}_{\boldsymbol{\ell},j}^\top). \tag{C.19}$$

We decompose $\boldsymbol{t}$ into two components, $\boldsymbol{t}_{\boldsymbol{Z}}$ and $\boldsymbol{t}_{\boldsymbol{T}}$:

$$\boldsymbol{t} = \boldsymbol{t}_{\boldsymbol{Z}} + \boldsymbol{t}_{\boldsymbol{T}}, \quad \boldsymbol{t}_{\boldsymbol{Z}} = \boldsymbol{Z}\boldsymbol{c}_{\boldsymbol{Z}}, \quad \boldsymbol{t}_{\boldsymbol{T}} = \boldsymbol{T}\boldsymbol{c}_{\boldsymbol{T}}. \tag{C.20}$$

Here, $\boldsymbol{t}_{\boldsymbol{Z}}$ represents the update direction tangent to the constraints, and $\boldsymbol{t}_{\boldsymbol{T}}$ represents the update direction orthogonal to the constraints. $\boldsymbol{c}_{\boldsymbol{Z}}$ and $\boldsymbol{c}_{\boldsymbol{T}}$ are, respectively, the coordinates of $\boldsymbol{t}_{\boldsymbol{Z}}$ and $\boldsymbol{t}_{\boldsymbol{T}}$ in

the basis of $\boldsymbol{Z}$ and $\boldsymbol{T}$. We write $\boldsymbol{c}_{\boldsymbol{Z}}$ and $\boldsymbol{c}_{\boldsymbol{T}}$ as

$$\boldsymbol{c}_{\boldsymbol{Z}} = \begin{bmatrix} \boldsymbol{c}_{\boldsymbol{Z},\boldsymbol{x}}^\top & \boldsymbol{c}_{\boldsymbol{Z},\boldsymbol{\ell}}^\top \end{bmatrix}^\top, \quad \boldsymbol{c}_{\boldsymbol{Z},\boldsymbol{x}} = \begin{bmatrix} \mathbf{c}_{\mathbf{Z},\mathbf{x},1}^\top & \cdots & \mathbf{c}_{\mathbf{Z},\mathbf{x},K}^\top \end{bmatrix}^\top, \quad \boldsymbol{c}_{\boldsymbol{Z},\boldsymbol{\ell}} = \begin{bmatrix} \mathbf{c}_{\mathbf{Z},\boldsymbol{\ell},1}^\top & \cdots & \mathbf{c}_{\mathbf{Z},\boldsymbol{\ell},V}^\top \end{bmatrix}^\top, \tag{C.21}$$

$$\boldsymbol{c}_{\boldsymbol{T}} = \begin{bmatrix} \boldsymbol{c}_{\boldsymbol{T},\boldsymbol{x}}^\top & \boldsymbol{c}_{\boldsymbol{T},\boldsymbol{\ell}}^\top \end{bmatrix}^\top, \quad \boldsymbol{c}_{\boldsymbol{T},\boldsymbol{x}} = \begin{bmatrix} \mathbf{c}_{\mathbf{T},\mathbf{x},1}^\top & \cdots & \mathbf{c}_{\mathbf{T},\mathbf{x},K}^\top \end{bmatrix}^\top, \quad \boldsymbol{c}_{\boldsymbol{T},\boldsymbol{\ell}} = \begin{bmatrix} \mathbf{c}_{\mathbf{T},\boldsymbol{\ell},1}^\top & \cdots & \mathbf{c}_{\mathbf{T},\boldsymbol{\ell},V}^\top \end{bmatrix}^\top. \tag{C.22}$$

To make the coordinates $\mathbf{c}_{\mathbf{Z},\mathbf{x},k}$ and $\mathbf{c}_{\mathbf{Z},\boldsymbol{\ell},j}$ more interpretable, we can enforce that the choice of $\mathbf{Z}_{\mathbf{x},k}$ and $\mathbf{Z}_{\boldsymbol{\ell},j}$ has the identity matrix of the appropriate size on top:

$$\mathbf{Z}_{\mathbf{x},k} = \begin{bmatrix} \mathbf{1}_{N^\xi} \\ * \end{bmatrix}, \quad \mathbf{Z}_{\boldsymbol{\ell},j} = \begin{bmatrix} \mathbf{1}_{N^\psi} \\ * \end{bmatrix}, \tag{C.23}$$

where the $*$ block consists of any entries such that (C.19) is satisfied. Then, each element in $\mathbf{c}_{\mathbf{Z},\mathbf{x},k}$ and $\mathbf{c}_{\mathbf{Z},\boldsymbol{\ell},j}$ represents, respectively, the update in each dimension of $\boldsymbol{\xi}_k$ and $\boldsymbol{\psi}_j$ tangent to the constraints. This structure also simplifies the extraction of covariances later.

To solve for $\boldsymbol{t}$, we first solve for $\boldsymbol{t}_{\boldsymbol{T}}$, then solve for $\boldsymbol{t}_{\boldsymbol{Z}}$. From the second equation of (C.14), we have

$$\boldsymbol{S}\boldsymbol{t} = -\boldsymbol{h} \;\Rightarrow\; \boldsymbol{S}(\boldsymbol{Z}\boldsymbol{c}_{\boldsymbol{Z}} + \boldsymbol{T}\boldsymbol{c}_{\boldsymbol{T}}) = -\boldsymbol{h} \;\Rightarrow\; (\boldsymbol{S}\boldsymbol{T})\boldsymbol{c}_{\boldsymbol{T}} = -\boldsymbol{h}. \tag{C.24}$$

We assume that $\boldsymbol{S}$ has full row rank, or else the SQP is not feasible. Note that $\boldsymbol{S}\boldsymbol{T}$ is nonsingular. To see why, notice that $\begin{bmatrix} \boldsymbol{Z} | \boldsymbol{T} \end{bmatrix}$ has full rank, so the product $\boldsymbol{S}\begin{bmatrix} \boldsymbol{Z} | \boldsymbol{T} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} | \boldsymbol{S}\boldsymbol{T} \end{bmatrix}$ has full row rank, and thus $\boldsymbol{S}\boldsymbol{T}$ is a square nonsingular matrix. Therefore, we can uniquely solve for $\boldsymbol{c}_{\boldsymbol{T}}$ from (C.24). With our choice of $\boldsymbol{T}$ in (C.18), $\boldsymbol{S}\boldsymbol{T}$ also becomes block-diagonal with the same structure as in (C.18). We can thus solve for $\boldsymbol{c}_{\boldsymbol{T}}$ in $\mathcal{O}(N^3(K+V))$ time with

$$(\mathbf{S}_{\mathbf{x},k}\mathbf{T}_{\mathbf{x},k})\mathbf{c}_{\mathbf{T},\mathbf{x},k} = -\mathbf{h}_{\mathbf{x},k}, \quad k = 1,\ldots,K, \tag{C.25a}$$

$$(\mathbf{S}_{\boldsymbol{\ell},j}\mathbf{T}_{\boldsymbol{\ell},j})\mathbf{c}_{\boldsymbol{T},\boldsymbol{\ell},j} = -\mathbf{h}_{\boldsymbol{\ell},j}, \quad j = 1,\ldots,V. \tag{C.25b}$$

To find $\boldsymbol{t}_{\boldsymbol{Z}}$, we premultiply the first equation of (C.14) by $\boldsymbol{Z}^\top$ to elimininate any $\boldsymbol{Z}^\top\boldsymbol{S}^\top$ terms:

$$\boldsymbol{Z}^\top(\boldsymbol{F}\boldsymbol{t} - \boldsymbol{S}^\top\boldsymbol{v}) = \boldsymbol{Z}^\top(\boldsymbol{g} + \boldsymbol{S}^\top\boldsymbol{\lambda}), \tag{C.26a}$$

$$\Rightarrow \boldsymbol{Z}^\top\boldsymbol{F}(\boldsymbol{Z}\boldsymbol{c}_{\boldsymbol{Z}} + \boldsymbol{T}\boldsymbol{c}_{\boldsymbol{T}}) = \boldsymbol{Z}^\top\boldsymbol{g}, \tag{C.26b}$$

$$\Rightarrow (\boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{Z})\boldsymbol{c}_{\boldsymbol{Z}} = \boldsymbol{Z}^\top\boldsymbol{g} - \boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{T}\boldsymbol{c}_{\boldsymbol{T}}, \tag{C.26c}$$

$$\Rightarrow \boldsymbol{F}_{\boldsymbol{Z}}\boldsymbol{c}_{\boldsymbol{Z}} = \boldsymbol{g}_{\boldsymbol{Z}}, \tag{C.26d}$$

where $\boldsymbol{F}_{\boldsymbol{Z}} = \boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{Z}$ and $\boldsymbol{g}_{\boldsymbol{Z}} = \boldsymbol{Z}^\top\boldsymbol{g} - \boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{T}\boldsymbol{c}_{\boldsymbol{T}}$. Here, $\boldsymbol{F}_{\boldsymbol{Z}}$ represents the reduced Hessian [28], with its size just being the degrees of freedom of the original unlifted system.

The SQP solution is guaranteed to be a direction of descent if $\boldsymbol{F}_{\boldsymbol{Z}}$ is positive definite [28, p. 452]. Since $\boldsymbol{F}$ is at least positive semidefinite, $\boldsymbol{F}_{\boldsymbol{Z}}$ is at least positive semidefinite. We then only require that $\boldsymbol{F}_{\boldsymbol{Z}}$ is full rank. This can be interpreted as observability of the lifted constrained system, where the solution of the states and landmarks is unique given the process prior, measurements, and constraints. Unlike in UKL, the constraints contribute to the observability of the system. That is, $\boldsymbol{F}$ may not be full rank and is only positive semidefinite (i.e., the unconstrained system is unobservable), but $\boldsymbol{F}_{\boldsymbol{Z}}$ is positive definite (i.e., the constrained system is observable). This happens when the unobservable

space in $\boldsymbol{F}$ is projected out by $\boldsymbol{Z}$. We see this commonly in RCKL-SLAM, where the evolution of the lifted features is 'moved' from the process model to the nonlinear constraints, and the system is only observable with the constraints in place.

Similar to the case for UKL, the observability of (R)CKL depends on the priors, lifting functions, training data, and the observability of the original system. In our experiments, the rank requirement always holds empirically when the SQP is initialized according to Section 6.7.3.

Using $\boldsymbol{F}$'s breakdown in (C.8), $\boldsymbol{F}_{\boldsymbol{Z}}$ can be written as

$$\boldsymbol{F}_{\boldsymbol{Z}} = \begin{bmatrix} \boldsymbol{Z}_{\boldsymbol{x}}^\top \boldsymbol{F}_{\boldsymbol{x}\boldsymbol{x}} \boldsymbol{Z}_{\boldsymbol{x}} & \boldsymbol{Z}_{\boldsymbol{x}}^\top \boldsymbol{F}_{\boldsymbol{x}\boldsymbol{\ell}} \boldsymbol{Z}_{\boldsymbol{\ell}} \\ \boldsymbol{Z}_{\boldsymbol{\ell}}^\top \boldsymbol{F}_{\boldsymbol{x}\boldsymbol{\ell}}^\top \boldsymbol{Z}_{\boldsymbol{x}} & \boldsymbol{Z}_{\boldsymbol{\ell}}^\top \boldsymbol{F}_{\boldsymbol{\ell}\boldsymbol{\ell}} \boldsymbol{Z}_{\boldsymbol{\ell}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{\boldsymbol{Z},\boldsymbol{x}\boldsymbol{x}} & \boldsymbol{F}_{\boldsymbol{Z},\boldsymbol{x}\boldsymbol{\ell}} \\ \boldsymbol{F}_{\boldsymbol{Z},\boldsymbol{x}\boldsymbol{\ell}}^\top & \boldsymbol{F}_{\boldsymbol{Z},\boldsymbol{\ell}\boldsymbol{\ell}} \end{bmatrix}, \tag{C.27}$$

where, owing to the block-diagonal structure of $\boldsymbol{Z}$, the SLAM arrowhead sparsity structure is preserved. This is to say, $\boldsymbol{F}_{\boldsymbol{Z},\boldsymbol{x}\boldsymbol{x}}$ is block-tridiagonal and $\boldsymbol{F}_{\boldsymbol{Z},\boldsymbol{\ell}\boldsymbol{\ell}}$ is block-diagonal. The blocks in $\boldsymbol{F}_{\boldsymbol{Z}}$ can be computed in parallel in $\mathcal{O}(N^3KV)$ time. The computation of $\boldsymbol{g}_{\boldsymbol{Z}}$ is similarly efficient, since $\boldsymbol{Z}^\top$ is block-diagonal and $\boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{Z}$ also preserves the sparsity pattern of $\boldsymbol{F}$. Then, we can use a Cholesky decomposition on $\boldsymbol{F}_{\boldsymbol{Z}}$, similar to the one done on $\boldsymbol{F}$ in (C.9), to efficiently solve for $\boldsymbol{c}_{\boldsymbol{Z}}$ in (C.26d). Again, the complexity is $\mathcal{O}(N^3(V^3+V^2K))$ when the lower Cholesky decomposition is used when there are more timesteps than landmarks, but the upper Cholesky decomposition can be used for the other case. Finally, we use $\boldsymbol{c}_{\boldsymbol{Z}}$ and $\boldsymbol{c}_{\boldsymbol{T}}$ to form $\boldsymbol{t}_{\boldsymbol{Z}}$ and $\boldsymbol{t}_{\boldsymbol{T}}$, and then form the full update variable, $\boldsymbol{t}$, with (C.20), all in linear time.

To solve for the Lagrange multiplier update, $\boldsymbol{v}$, we premultiply the first equation of (C.14) by $\boldsymbol{T}^\top$:

$$\boldsymbol{T}^\top(\boldsymbol{F}\boldsymbol{t} - \boldsymbol{S}^\top\boldsymbol{v}) = \boldsymbol{T}^\top(\boldsymbol{g} + \boldsymbol{S}^\top\boldsymbol{\lambda}), \tag{C.28a}$$

$$\Rightarrow (\boldsymbol{S}\boldsymbol{T})^\top\boldsymbol{v} = \boldsymbol{T}^\top(\boldsymbol{F}\boldsymbol{t} - \boldsymbol{g} - \boldsymbol{S}^\top\boldsymbol{\lambda}), \tag{C.28b}$$

where we can solve for $\boldsymbol{v}$ in linear time since the square matrix $\boldsymbol{S}\boldsymbol{T}$ is full rank and block-diagonal. With this, we have efficiently solved for $\boldsymbol{t}$ and $\boldsymbol{v}$ in time $\mathcal{O}(N^3(V^3+V^2K))$.

C.1.5 Proof of (R)CKL-SLAM Covariance Extraction

This section is referenced in Section 6.7.2 and Section 7.7.2. We show that when $\mathbf{Z}_{\mathbf{x},k}$ and $\mathbf{Z}_{\boldsymbol{\ell},j}$ have the structures shown in (C.23), then the batch covariance of the primal (lifted) variables satisfies

$$\hat{\boldsymbol{P}}_{\boldsymbol{q}} = \boldsymbol{Z}\boldsymbol{F}_{\boldsymbol{Z}}^{-1}\boldsymbol{Z}^\top = \boldsymbol{Z}(\boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{Z})^{-1}\boldsymbol{Z}^\top, \tag{C.29}$$

and the covariance of the original variables satisfies

$$\hat{\boldsymbol{\Sigma}}_{\zeta} = \boldsymbol{F}_{\boldsymbol{Z}}^{-1} = (\boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{Z})^{-1}, \tag{C.30}$$

where $\boldsymbol{F}_{\boldsymbol{Z}}$ is constructed at the solution $(\boldsymbol{q}^\star, \boldsymbol{\lambda}^\star)$. Since $\boldsymbol{F}_{\boldsymbol{Z}}$ has the SLAM arrowhead sparsity structure, we use a similar procedure as for UKL-SLAM to extract the required covariances by finding the corresponding nonzero blocks of $\boldsymbol{F}_{\boldsymbol{Z}}$.

We now show (C.30). After convergence, we construct the left-hand side of the SQP in (C.14)

once more, yielding $\boldsymbol{M} = \begin{bmatrix} \boldsymbol{F} & \boldsymbol{S}^\top \\ \boldsymbol{S} & \boldsymbol{0} \end{bmatrix}$. $\boldsymbol{M}$ is termed the Karush-Kuhn-Tucker (KKT) matrix [125]. Before inverting $\boldsymbol{M}$, note that $\boldsymbol{F}$ may not be invertible. As discussed in Appendix C.1.4, this occurs when only the constrained system is observable, which is most commonly seen in RCKL-SLAM. However, as long as $\boldsymbol{F}_{\boldsymbol{Z}}$ is invertible and $\boldsymbol{S}$ has full row rank, then $\boldsymbol{M}$ is invertible. This is because the SQP solution, $\delta\boldsymbol{q}$ and $\delta\boldsymbol{\lambda}$ in (C.14), is unique, as seen in Appendix C.1.4 by construction. We now write the inverse of the KKT matrix as $\boldsymbol{M}^{-1} = \begin{bmatrix} \boldsymbol{P}_{\mathbf{F}} & \boldsymbol{P}_{\mathbf{S}}^\top \\ \boldsymbol{P}_{\mathbf{S}} & \boldsymbol{P}_{\mathbf{Z}} \end{bmatrix}$, where $\boldsymbol{P}_{\mathbf{F}}$, $\boldsymbol{P}_{\mathbf{S}}$, and $\boldsymbol{P}_{\mathbf{Z}}$ have the same sizes as $\boldsymbol{F}$, $\boldsymbol{S}$, and the zero block in $\boldsymbol{M}$, respectively. Then, it is known that $\hat{\boldsymbol{P}}_{\boldsymbol{q}}$, the covariance of the primal variable, satisfies $\hat{\boldsymbol{P}}_{\boldsymbol{q}} = \boldsymbol{P}_{\mathbf{F}}$ [43].

We cannot use the Schur complement on $\boldsymbol{M}$ to solve for $\boldsymbol{P}_{\mathbf{F}}$ since $\boldsymbol{F}$ is not necessarily invertible. Instead, notice that when constraints are satisfied at the minimum-cost solution, any uncertainty resulting from noisy information would lie only within the constraint manifold's tangent space [43]. That is, we can write $\boldsymbol{q}^\star = \boldsymbol{Z}\boldsymbol{c}^\star$, where $\boldsymbol{c}^\star$ is the coordinate of $\boldsymbol{q}^\star$ in the basis of $\boldsymbol{Z}$. The solution distribution in the basis of $\boldsymbol{Z}$ satisfies $\boldsymbol{c} \sim \mathcal{N}(\hat{\boldsymbol{c}}, \hat{\boldsymbol{P}}_{\boldsymbol{c}})$, where $\hat{\boldsymbol{c}} = \boldsymbol{c}^\star$ and $\hat{\boldsymbol{P}}_{\mathbf{q}} = \boldsymbol{Z}\hat{\boldsymbol{P}}_{\boldsymbol{c}}\boldsymbol{Z}^\top$, affirming that the covariance of $\boldsymbol{q}$ is within the space of $\boldsymbol{Z}$.

We write out the expression $\boldsymbol{M}\boldsymbol{M}^{-1} = \boldsymbol{1}$, yielding

$$\begin{bmatrix} \boldsymbol{F} & \boldsymbol{S}^\top \\ \boldsymbol{S} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_{\mathbf{F}} & \boldsymbol{P}_{\mathbf{S}}^\top \\ \boldsymbol{P}_{\mathbf{S}} & \boldsymbol{P}_{\mathbf{Z}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}. \tag{C.31}$$

We premultiply first expression of (C.31) by $\boldsymbol{Z}^\top$:

$$\boldsymbol{F}\boldsymbol{P}_{\mathbf{F}} + \boldsymbol{S}^\top\boldsymbol{P}_{\mathbf{S}} = \boldsymbol{1}, \tag{C.32a}$$

$$\Rightarrow \boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{Z}\hat{\boldsymbol{P}}_{\boldsymbol{c}}\boldsymbol{Z}^\top + \boldsymbol{Z}^\top\boldsymbol{S}^\top\boldsymbol{P}_{\mathbf{S}} = \boldsymbol{Z}^\top, \tag{C.32b}$$

$$\Rightarrow (\boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{Z})\hat{\boldsymbol{P}}_{\boldsymbol{c}}\boldsymbol{Z}^\top = \boldsymbol{Z}^\top, \tag{C.32c}$$

$$\Rightarrow \hat{\boldsymbol{P}}_{\boldsymbol{c}} = (\boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{Z})^{-1}, \tag{C.32d}$$

where the last derivation uses the fact that $\boldsymbol{Z}$ has full column rank. Thus, we have shown that the covariance of the primal variable, corresponding to the lifted state, is

$$\hat{\boldsymbol{P}}_{\boldsymbol{q}} = \boldsymbol{Z}\hat{\boldsymbol{P}}_{\boldsymbol{c}}\boldsymbol{Z}^\top = \boldsymbol{Z}(\boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{Z})^{-1}\boldsymbol{Z}^\top. \tag{C.33}$$

We can proceed to get $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\zeta}}$, the covariance of the original state variables, by using (C.33) to prove (C.30). Given the structure of the bases in (C.23), however, we can prove (C.30) in an easier way: Owing to (C.23), the dimensions in $\boldsymbol{c}$ exactly represent the dimensions of the original variables, so the covariance of $\boldsymbol{\zeta}$ is equivalent to the covariance of $\boldsymbol{c}$. Thus, $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\zeta}} = \hat{\boldsymbol{P}}_{\boldsymbol{c}} = (\boldsymbol{Z}^\top\boldsymbol{F}\boldsymbol{Z})^{-1}$.

C.1.6 Modifications for Other (R)CKL Estimation Problems

This section is referenced in Section 6.7.3 and in Appendix C.1.7. We discuss the modifications required to solve (R)CKL estimation problems other than SLAM: namely, localization, dead reckoning, and mapping. One may solve these problems to acquire good initial points for the SQPs of harder estimation problems including CKL-Loc and CKL-SLAM. The initialization procedure for these CKL estimation problems is discussed in the next section.

- *Localization*: We use measurements from known landmarks to optimize for the robot states. We modify the optimization objective in (6.31) to treat $\boldsymbol{\ell}$ as a constant, removing $\boldsymbol{\ell}$ from the objective function and from the constraints:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2}\boldsymbol{e}(\boldsymbol{x})^{\top}\boldsymbol{W}^{-1}\boldsymbol{e}(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{x}) = \boldsymbol{0}, \end{aligned} \tag{C.34}$$

  where $\boldsymbol{W}$ and $\boldsymbol{e}$ are, respectively, still as described by (6.22b) and (6.22c), but $\boldsymbol{e}$ is only a function of $\boldsymbol{x}$. $\boldsymbol{h}_{\boldsymbol{x}}$ has the same block structure as described in (6.32). Then, we follow the same procedure as (R)CKL-SLAM for formulating and solving SQP, with only $\boldsymbol{x}$ as an unknown. The quantities $\boldsymbol{\lambda}, \boldsymbol{S}, \boldsymbol{Z}, \boldsymbol{T}, \boldsymbol{t}, \boldsymbol{v}$ would have, respectively, the same block structures as $\boldsymbol{\lambda}_{\boldsymbol{x}}, \boldsymbol{S}_{\boldsymbol{x}}, \boldsymbol{Z}_{\boldsymbol{x}}, \boldsymbol{T}_{\boldsymbol{x}}, \boldsymbol{t}_{\boldsymbol{x}}, \boldsymbol{v}_{\boldsymbol{x}}$ in (R)CKL-SLAM. We use the unconstrained Hessian $\boldsymbol{F}_{\boldsymbol{x}\boldsymbol{x}}$ in place of $\boldsymbol{F}$ in (C.8). We can use the same nullspace method to solve for $\boldsymbol{t}$ in linear time. Note that unlike in UKL-Loc where the cost is quadratic and the solution is one shot, we can no longer avoid an iterative process in (R)CKL-Loc owing to the nonlinear constraints.

- *Dead Reckoning*: There are no measurements, and we optimize for the robot states from only the inputs and the process model. The objective is

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2}\boldsymbol{e}_{\boldsymbol{v}}(\boldsymbol{x})^{\top}\boldsymbol{Q}^{-1}\boldsymbol{e}_{\boldsymbol{v}}(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{x}) = \boldsymbol{0}, \end{aligned} \tag{C.35}$$

  which is the same objective as for localization with the measurement components removed. The Hessian of the unconstrained system, $\boldsymbol{F}$, would simply be $\boldsymbol{A}^{-T}\boldsymbol{Q}^{-1}\boldsymbol{A}^{-1}$, and the rest of the procedure for setting up and solving the SQP is similar to constrained localization as described above. Note that while unconstrained systems can dead reckon by simply recursively applying the process model, constrained dead reckoning requires an iterative process to ensure that the solution is on the manifold.

- *Mapping*: The problem is the inverse of localization: given a trajectory with known robot states, map the landmarks from the measurements. The optimization objective is

$$\begin{aligned} \min_{\boldsymbol{\ell}} \quad & \frac{1}{2}\boldsymbol{e}_{\boldsymbol{y}}(\boldsymbol{\ell})^{\top}\boldsymbol{R}^{-1}\boldsymbol{e}_{\boldsymbol{y}}(\boldsymbol{\ell}) \\ \text{s.t.} \quad & \boldsymbol{h}_{\boldsymbol{\ell}}(\boldsymbol{\ell}) = \boldsymbol{0}, \end{aligned} \tag{C.36}$$

  which is modified from (6.31) such that $\boldsymbol{x}$ is known and $\boldsymbol{\ell}$ is unknown. The quantities $\boldsymbol{\lambda}, \boldsymbol{S}, \boldsymbol{Z}, \boldsymbol{T}, \boldsymbol{t}, \boldsymbol{v}$ would have, respectively, the same block structures as $\boldsymbol{\lambda}_{\boldsymbol{\ell}}, \boldsymbol{S}_{\boldsymbol{\ell}}, \boldsymbol{Z}_{\boldsymbol{\ell}}, \boldsymbol{T}_{\boldsymbol{\ell}}, \boldsymbol{t}_{\boldsymbol{\ell}}, \boldsymbol{v}_{\boldsymbol{\ell}}$ in (R)CKL-SLAM. The unconstrained Hessian for mapping, $\boldsymbol{F}$, would correspond to $\boldsymbol{F}_{\boldsymbol{\ell}\boldsymbol{\ell}}$ in (C.8). We can use the same nullspace method to solve for $\boldsymbol{t}$ in linear time.

C.1.7 SQP Initialization

This section is referenced in Section 6.7.3. We discuss the initializations of the SQPs for (R)CKL-based mapping, dead reckoning, localization, and SLAM. Detecting and overcoming local minima in estimation problems including localization and SLAM are open research problems in general [126,

127]. In our case, the optimization problem setups in (6.31), (C.34), (C.35), and (C.36) are all nonlinear owing to the presence of nonlinear constraints, and a reasonable initial guess is often required to prevent the convergence to a local minimum. When they do occur, the SQP has usually converged to a place where the constraints are not satisfied, which we can detect by observing the value of $\mu||\boldsymbol{h}(\boldsymbol{q})||_1$ in (C.16).

The solution procedure with initialization is as follows:

- *Dead reckoning*: We solve (C.35) for $\boldsymbol{x}$ by first dropping the constraints and solving the unconstrained problem, then passing the unconstrained solution as the initial guess for the SQP. Note that the UKL solution of dead reckoning simply corresponds to recursively applying the learned process model. To ensure that no local minima are encountered, the problem is not solved over the whole trajectory, but rather over windows of size N, $\{\mathbf{x}_k, \mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+N-1}\}$, where we know $\mathbf{x}_k$ and solve for $\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+N-1}$. We start at a known initial state $\mathbf{x}_0$ and dead reckon through the windows recursively until the end of the trajectory. A window size of two often guarantees that local minima are not encountered, though a larger window size can usually be used for a computational speedup without affecting results, depending on how far the state drifts from the manifold within N timesteps.

- *Mapping*: To initialize $\boldsymbol{\ell}$ to solve (C.36), we drop the constraints and use the UKL mapping solution. Similar to UKL-Loc, the unconstrained mapping problem is quadratic and can be solved in one shot. Mirroring the validity of UKL-Loc described in Section 6.6.5, unconstrained mapping works well when there are enough measurements received per landmark to keep the solution close to the manifold.

- *Localization*: To initialize $\boldsymbol{x}$ to solve (C.34), we simply initialize $\boldsymbol{x}$ from constrained dead reckoning.

- *SLAM*: To initialize $(\boldsymbol{x}, \boldsymbol{\ell})$ to solve (6.31), we first initialize $\boldsymbol{x}$ by doing constrained dead reckoning, then initialize $\boldsymbol{\ell}$ by doing mapping from the dead-reckoned trajectory.

C.1.8 (R)CKL Estimation with Orientation

This section is referenced in Section 6.9.1. During the lifting process, we have assumed in (6.1) that the state of the robot, $\boldsymbol{\xi}$, lies within a vector space. While this is true for many robots, there are exceptions where the assumption does not hold. A common example of a non-vector space state variable is a robot's orientation, which is a circular quantity. In this section, we present the required modifications for a 2D pose, i.e., position and orientation. This section acts as an example of variable transformation and shows the ease of including additional state constraints within (R)CRKL.

Suppose the state of a 2D robot is $\boldsymbol{\xi}_k = \begin{bmatrix}\xi_{x,k} & \xi_{y,k} & \xi_{\theta,k}\end{bmatrix}^\top$, where $(\xi_{x,k}, \xi_{y,k})$ is its position and $\xi_{\theta,k}$ is its orientation. Since $\xi_{\theta,k}$ is circular, simply lifting with $\mathbf{x}_k = \mathbf{p}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_k)$ in (6.28a) would not work: $\xi_{\theta,k}$ would be part of the lifted state but we cannot enforce that $\xi_{\theta,k} = \xi_{\theta,k} + 2m\pi$ for $m \in \mathbb{Z}$. Instead, we do a variable transformation on $\boldsymbol{\xi}_k$ as $\boldsymbol{\xi}_k' = \begin{bmatrix}\xi_{x,k} & \xi_{y,k} & \xi_{\cos\theta,k} & \xi_{\sin\theta,k}\end{bmatrix}^\top$, where the conversion from $\boldsymbol{\xi}_k$ to $\boldsymbol{\xi}_k'$ is $\xi_{\cos\theta,k} = \cos\xi_{\theta,k}$, $\xi_{\sin\theta,k} = \sin\xi_{\theta,k}$. We introduce the constraint $h_\xi(\boldsymbol{\xi}_k') = \xi_{\cos\theta,k}^2 + \xi_{\sin\theta,k}^2 - 1 = 0$. We then select a lifting function $\tilde{\mathbf{p}}_{\boldsymbol{\xi}'}(\boldsymbol{\xi}_k')$, and then write the lifted state as $\mathbf{x}_k = \begin{bmatrix}\boldsymbol{\xi}_k' \\ \tilde{\mathbf{p}}_{\boldsymbol{\xi}'}(\boldsymbol{\xi}_k')\end{bmatrix} = \begin{bmatrix}\boldsymbol{\xi}_k' \\ \tilde{\mathbf{x}}_k\end{bmatrix}$, where the manifold constraint is now the lifting function constraint

in addition to the orientation constraint on $\boldsymbol{\xi}_k^{'}$:

$$\mathbf{x}_k \in \mathcal{X} \;\Rightarrow\; \mathbf{h}_{\mathbf{x}}(\mathbf{x}_k) = \begin{bmatrix} \tilde{\mathbf{x}}_k - \tilde{\mathbf{p}}_{\boldsymbol{\xi}'}(\boldsymbol{\xi}_k^{'}) \\ \xi_{\cos\theta,k}^2 + \xi_{\sin\theta,k}^2 - 1 \end{bmatrix} = \mathbf{0}. \tag{C.37}$$

Note that the additional constraint still acts on each state individually at each timestep, meaning that $\boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{x})$ is still block-diagonal. Thus, the rest of the procedure for (R)CKL estimation follows as usual. Afterwards, we recover $\hat{\xi}_{\theta,k} = \text{atan2}(\hat{\xi}_{\sin\theta,k}, \hat{\xi}_{\cos\theta,k})$.

With the additional constraint, the covariance extraction needs to be slightly modified. $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\xi},k}$, the covariance of $\boldsymbol{\xi}_k$, is not the same as $\hat{\mathbf{P}}_{\mathbf{c},k}$, the covariance of $\mathbf{x}_k$ in the coordinates of the constraint tangent space defined by $\mathbf{Z}_{\mathbf{x},k}$. $\boldsymbol{\xi}_k'$ has 4 dimensions but only 3 degrees of freedom. As such, we enforce a different structure on $\mathbf{Z}_{\mathbf{x},k}$ than in (C.23):

$$\mathbf{Z}_{\mathbf{x},k} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & Z_{||,\cos\theta,k} \\ 0 & 0 & Z_{||,\sin\theta,k} \\ * & * & * \\ \vdots & \vdots & \vdots \end{bmatrix}, \tag{C.38}$$

where the $*$ entries are any entries so that $\mathbf{Z}_{\mathbf{x},k} = \text{null}(\mathbf{S}_{\mathbf{x},k})$ is satisfied. Then, given $\hat{\mathbf{P}}_{\mathbf{c},k}$ in the form

$$\hat{\mathbf{P}}_{\mathbf{c},k} = \begin{bmatrix} \hat{\sigma}^2_{\mathbf{c},x,k} & \hat{\sigma}^2_{\mathbf{c},xy,k} & \hat{\sigma}^2_{\mathbf{c},x\theta,k} \\ \hat{\sigma}^2_{\mathbf{c},xy,k} & \hat{\sigma}^2_{\mathbf{c},y,k} & \hat{\sigma}^2_{\mathbf{c},y\theta,k} \\ \hat{\sigma}^2_{\mathbf{c},x\theta,k} & \hat{\sigma}^2_{\mathbf{c},y\theta,k} & \hat{\sigma}^2_{\mathbf{c},\theta,k} \end{bmatrix}, \tag{C.39}$$

the position covariance is still the upper-left 2×2 block,

$$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\xi},xy,k} = \begin{bmatrix} \hat{\sigma}^2_{\mathbf{c},x,k} & \hat{\sigma}^2_{\mathbf{c},xy,k} \\ \hat{\sigma}^2_{\mathbf{c},xy,k} & \hat{\sigma}^2_{\mathbf{c},y,k} \end{bmatrix}, \tag{C.40}$$

and the variance of $\xi_{\cos\theta,k}$ and $\xi_{\sin\theta,k}$ can be found with

$$\hat{\Sigma}_{\boldsymbol{\xi},\cos\theta,k} = S^2_{||,\cos\theta,k}\hat{\sigma}^2_{\mathbf{c},\theta,k}, \quad \hat{\Sigma}_{\boldsymbol{\xi},\sin\theta,k} = S^2_{||,\sin\theta,k}\hat{\sigma}^2_{\mathbf{c},\theta,k}, \tag{C.41}$$

either of which can be converted to the orientation variance, $\hat{\Sigma}_{\boldsymbol{\xi},k,\theta}$, through a linear transformation of a Gaussian at the mean, $\hat{\xi}_{\theta,k}$. We see in our experiments that when $\hat{\Sigma}_{\boldsymbol{\xi},\cos\theta,k}$ and $\hat{\Sigma}_{\boldsymbol{\xi},\sin\theta,k}$ are small, using either variance yields the same value for $\hat{\Sigma}_{\boldsymbol{\xi},\theta,k}$.

Bibliography

[1] T. D. Barfoot, *State Estimation for Robotics*, 1st ed. Cambridge, U.K.: Cambridge University Press, 2017. DOI: 10.1017/9781316671528.

[2] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable particle filters: End-to-end learning with algorithmic priors," in *Proceedings of Robotics: Science and Systems*, Jun. 2018. DOI: 10.15607/RSS.2018.XIV.001.

[3] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017. DOI: 10.1109/ICRA.2017.7989236.

[4] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "VINet: Visual-inertial odometry as a sequence-to-sequence learning problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017. DOI: 10.1609/aaai.v31i1.11215.

[5] G. Revach et al., "KalmanNet: Neural network aided Kalman filtering for partially known dynamics," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1532–1547, 2022. DOI: 10.1109/TSP.2022.3158588.

[6] B. O. Koopman, "Hamiltonian systems and transformation in Hilbert space," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 17, no. 5, pp. 315–318, 1931. DOI: 10.1073/pnas.17.5.315.

[7] A. Mauroy, I. Mezić, and Y. Susuki, Eds., *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications* (Lecture Notes in Control and Information Sciences). Cham: Springer, 2020, vol. 484, ISBN: 978-3-030-35713-9. DOI: 10.1007/978-3-030-35713-9.

[8] A. Mauroy and J. Goncalves, "Koopman-based lifting techniques for nonlinear systems identification," *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2550–2565, 2020. DOI: 10.1109/TAC.2019.2941433.

[9] D. Bruder, C. D. Remy, and R. Vasudevan, "Nonlinear system identification of soft robot dynamics using Koopman operator theory," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019, pp. 6244–6250. DOI: 10.1109/ICRA.2019.8793766.

[10] G. Mamakoukas, I. Abraham, and T. D. Murphey, "Learning stable models for prediction and control," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2255–2275, 2023. DOI: 10.1109/TRO.2022.3228130.

[11] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using Koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019. DOI: 10.1109/TRO.2019.2923880.

[12] M. Korda and I. Mezic, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, Nov. 2016. DOI: 10.1016/j.automatica.2018.03.046.

[13] D. Bruder, X. Fu, and R. Vasudevan, "Advantages of bilinear Koopman realizations for the modeling and control of systems with unknown dynamics," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4369–4376, 2021. DOI: 10.1109/LRA.2021.3068117.

[14] A. Surana, M. O. Williams, M. Morari, and A. Banaszuk, "Koopman operator framework for constrained state estimation," in *Proceedings of the 56th IEEE Conference on Decision and Control*, 2017, pp. 94–101. DOI: 10.1109/CDC.2017.8263649.

[15] A. Surana, "Koopman framework for nonlinear estimation," in *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications*, A. Mauroy, I. Mezić, and Y. Susuki, Eds. Cham: Springer, 2020, pp. 59–79, ISBN: 978-3-030-35713-9. DOI: 10.1007/978-3-030-35713-9_3.

[16] M. Netto and L. Mili, "A robust data-driven Koopman Kalman filter for power systems dynamic state estimation," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 7228–7237, 2018. DOI: 10.1109/TPWRS.2018.2846744.

[17] D. Olguín, A. Osses, and H. Ramírez, *Koopman Kalman filter (KKF): An asymptotically optimal nonlinear filtering algorithm with error bounds and its application to parameter estimation*, 2025. arXiv: 2511.04252 [math.DS].

[18] Z. C. Guo, J. R. Forbes, and T. D. Barfoot, *KILO-EKF: Koopman-inspired learned observations extended Kalman filter*, Accepted to IEEE/RSJ International Conference on Intelligent Robots and Systems, 2026. arXiv: 2601.12463 [cs.RO].

[19] Z. C. Guo, V. Korotkine, J. R. Forbes, and T. D. Barfoot, "Koopman linearization for data-driven batch state estimation of control-affine systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 866–873, 2022. DOI: 10.1109/LRA.2021.3133587.

[20] Z. C. Guo, F. Dümbgen, J. R. Forbes, and T. D. Barfoot, "Data-driven batch localization and SLAM using Koopman linearization," *IEEE Transactions on Robotics*, vol. 40, pp. 3964–3983, 2024. DOI: 10.1109/TRO.2024.3443674.

[21] Z. C. Guo, J. R. Forbes, and T. D. Barfoot, "Marginalizing and conditioning gaussians onto linear approximations of smooth manifolds with applications in robotics," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2606–2612, May 2025. DOI: 10.1109/ICRA55743.2025.11128000.

[22] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: The MIT Press, 2005, ISBN: 978-0-262-20162-9.

[23] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960. DOI: 10.1115/1.3662552.

[24] D. Simon, *Optimal State Estimation: Kalman, H∞, and Nonlinear Approaches*. Hoboken, NJ: Wiley-Interscience, 2006, ISBN: 978-0-471-70858-2. DOI: 10.1002/0470045345.

[25] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, 1965. DOI: 10.2514/3.3166.

[26] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006. DOI: 10.1109/MRA.2006.1638022.

[27] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006. DOI: 10.1109/MRA.2006.1678144.

[28] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006, ISBN: 978-0-387-30303-1. DOI: 10.1007/978-0-387-40065-5.

[29] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends in Robotics*, vol. 6, no. 1–2, pp. 1–139, 2017. DOI: 10.1561/2300000043.

[30] M. Kaess et al., "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 3281–3288. DOI: 10.1109/ICRA.2011.5979641.

[31] S. Anderson and T. D. Barfoot, "Full STEAM ahead: Exactly sparse Gaussian process regression for batch continuous-time trajectory estimation on SE(3)," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015. DOI: 10.1109/IROS.2015.7353368.

[32] C. Holmes, S. Lilge, Z. C. Guo, F. Dellaert, and T. D. Barfoot, "Smoothing out the edges: Continuous-time estimation with Gaussian process motion priors on factor graphs," *Accepted to Foundations and Trends in Robotics*, 2026. arXiv: 2605.09073 [cs.RO].

[33] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," 2021. arXiv: 1812.01537 [cs.RO].

[34] G. Bourmaud, R. Mégret, A. Giremus, and Y. Berthoumieu, "Discrete extended Kalman filter on Lie groups," in *Proceedings of the 21st European Signal Processing Conference*, Sep. 2013, pp. 1–5.

[35] G. Bourmaud, R. Mégret, M. Arnaudon, and A. Giremus, "Continuous-discrete extended Kalman filter on matrix Lie groups using concentrated Gaussian distributions," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 209–228, 2015. DOI: 10.1007/s10851-014-0517-0.

[36] Y. Song et al., "A right invariant extended Kalman filter for object based SLAM," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1316–1323, 2022. DOI: 10.1109/LRA.2021.3139370.

[37] S. Bonnable, P. Martin, and E. Salaün, "Invariant extended Kalman filter: Theory and application to a velocity-aided attitude estimation problem," in *Proceedings of the 48th IEEE Conference on Decision and Control held jointly with the 28th Chinese Control Conference*, 2009, pp. 1297–1304. DOI: 10.1109/CDC.2009.5400372.

[38] P. van Goor, T. Hamel, and R. Mahony, "Equivariant filter (EqF)," *IEEE Transactions on Automatic Control*, vol. 68, no. 6, pp. 3501–3512, 2023. DOI: 10.1109/TAC.2022.3194094.

[39] Y. Ge, P. van Goor, and R. Mahony, "A geometric perspective on fusing Gaussian distributions on Lie groups," *IEEE Control Systems Letters*, vol. 8, pp. 844–849, 2024. DOI: 10.1109/LCSYS.2024.3405485.

[40] P. Sodhi, S. Choudhury, J. G. Mangelson, and M. Kaess, "ICS: Incremental constrained smoothing for state estimation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020, pp. 279–285. DOI: 10.1109/ICRA40945.2020.9196649.

[41] M. Qadri, P. Sodhi, J. G. Mangelson, F. Dellaert, and M. Kaess, "InCOpt: Incremental constrained optimization using the Bayes tree," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 6381–6388. DOI: 10.1109/IROS47612.2022.9982178.

[42] M. Abu Bakr and S. Lee, "A framework of covariance projection on constraint manifold for data fusion," *Sensors*, vol. 18, no. 5, 2018. DOI: 10.3390/s18051610.

[43] H. G. Bock, E. Kostina, and O. Kostyukova, "Covariance matrices for parameter estimates of constrained parameter estimation problems," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 2, pp. 626–642, 2007. DOI: 10.1137/040617893.

[44] F. Wang and A. E. Gelfand, "Directional data analysis under the general projected normal distribution," *Statistical Methodology*, vol. 10, no. 1, pp. 113–127, 2013. DOI: 10.1016/j.stamet.2012.07.005.

[45] D. Hernandez-Stumpfhauser, F. Breidt, and M. Woerd, "The general projected normal distribution of arbitrary dimension: Modeling and Bayesian inference," *Bayesian Analysis*, vol. 12, Jan. 2016. DOI: 10.1214/15-BA989.

[46] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, Jun. 2008. DOI: 10.1214/009053607000000677.

[47] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006, ISBN: 978-0-262-18253-9. DOI: 10.7551/mitpress/3206.001.0001.

[48] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002, ISBN: 978-0-262-19475-4. DOI: 10.7551/mitpress/4175.001.0001.

[49] A. Smola, A. Gretton, L. Song, and B. Schölkopf, "A Hilbert space embedding for distributions," in *Algorithmic Learning Theory*, M. Hutter, R. A. Servedio, and E. Takimoto, Eds., ser. Lecture Notes in Computer Science, vol. 4754, Berlin, Heidelberg: Springer, 2007, pp. 13–31, ISBN: 978-3-540-75225-7. DOI: 10.1007/978-3-540-75225-7_5.

[50] L. Song, J. Huang, A. Smola, and K. Fukumizu, "Hilbert space embeddings of conditional distributions with applications to dynamical systems," in *Proceedings of the 26th International Conference on Machine Learning*, ser. ICML '09, Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 961–968, ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553497.

[51] K. Fukumizu, L. Song, and A. Gretton, "Kernel Bayes' rule: Bayesian inference with positive definite kernels," *Journal of Machine Learning Research*, vol. 14, no. 118, pp. 3753–3783, 2013. [Online]. Available: https://jmlr.org/papers/v14/fukumizu13a.html.

[52] Y. Nishiyama, A. Afsharinejad, S. Naruse, B. Boots, and L. Song, "The nonparametric kernel bayes smoother," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 51, PMLR, 2016, pp. 547–555. [Online]. Available: https://proceedings.mlr.press/v51/nishiyama16.html.

[53] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3471–3476. DOI: 10.1109/IROS.2008.4651188.

[54] B. D. Ferris, D. Fox, and N. D. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007, pp. 2480–2485.

[55] G. Gebhardt, A. Kupcsik, and G. Neumann, "The kernel Kalman rule: efficient nonparametric inference by recursive least-squares and subspace projections," *Machine Learning*, vol. 108, Jun. 2019. DOI: 10.1007/s10994-019-05816-z.

[56] L. McCalman, S. O'Callaghan, and F. Ramos, "Multi-modal estimation with kernel embeddings for learning motion models," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013, pp. 2845–2852. DOI: 10.1109/ICRA.2013.6630971.

[57] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-driven control of soft robots using Koopman operator theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 948–961, 2021. DOI: 10.1109/TRO.2020.3038693.

[58] S. Hagane, L. Jamone, and G. Venture, "Linearizing robotic manipulator's dynamics using Koopman operator and applying generalized predictive control," in *Proceedings of the IEEE/SICE International Symposium on System Integration*, 2023, pp. 1–8. DOI: 10.1109/SII55687.2023.10039188.

[59] T. Chen and J. Shan, "Koopman-operator-based attitude dynamics and control on SO(3)," *Journal of Guidance, Control, and Dynamics*, vol. 43, Sep. 2020. DOI: 10.2514/1.G005006.

[60] V. Zinage and E. Bakolas, "Koopman operator based modeling for quadrotor control on SE(3)," *IEEE Control Systems Letters*, vol. 6, pp. 752–757, 2022. DOI: 10.1109/LCSYS.2021.3085963.

[61] S. S. Narayanan, D. Tellez-Castro, S. Sutavani, and U. Vaidya, "SE(3) Koopman-MPC: Data-driven learning and control of quadrotor UAVs," *IFAC-PapersOnLine*, vol. 56, no. 3, pp. 607–612, 2023. DOI: 10.1016/j.ifacol.2023.12.091.

[62] T. Salam, A. K. Li, and M. A. Hsieh, "Online estimation of the Koopman operator using Fourier features," in *Proceedings of the 5th Annual Learning for Dynamics and Control Conference*, N. Matni, M. Morari, and G. J. Pappas, Eds., ser. Proceedings of Machine Learning Research, vol. 211, PMLR, Jun. 2023, pp. 1271–1283. [Online]. Available: https://proceedings.mlr.press/v211/salam23a.html.

[63] A. M. DeGennaro and N. M. Urban, "Scalable extended dynamic mode decomposition using random kernel approximation," *SIAM Journal on Scientific Computing*, vol. 41, no. 3, A1482–A1499, 2019. DOI: 10.1137/17M115414X.

[64] N. Takeishi, Y. Kawahara, and T. Yairi, "Learning Koopman invariant subspaces for dynamic mode decomposition," in *Advances in Neural Information Processing Systems*, 2017.

[65] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, no. 1, 2018, Art. no. 4950. DOI: 10.1038/s41467-018-07210-0.

[66] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Philadelphia, PA, USA: SIAM, 2016, ISBN: 978-1-61197-449-2. DOI: 10.1137/1.9781611974508.

[67] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge, U.K.: Cambridge University Press, 2019. DOI: 10.1017/9781108380690.

[68] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of nonlinear dynamics with control (SINDYc)," in *Proceedings of the 10th IFAC Symposium on Nonlinear Control Systems*, vol. 49, 2016, pp. 710–715. DOI: 10.1016/j.ifacol.2016.10.249.

[69] B. M. de Silva et al., "PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data," *Journal of Open Source Software*, vol. 5, no. 49, p. 2104, 2020. DOI: 10.21105/joss.02104.

[70] E. L. Jenson, M. Bando, K. Sato, and D. J. Scheeres, "Robust nonlinear optimal control using Koopman operator theory," in *Proceedings of the 2022 AAS/AIAA Astrodynamics Specialist Conference*, Paper AAS 22-647, Aug. 2022.

[71] A. Mauroy and J. Goncalves, "Linear identification of nonlinear systems: A lifting technique based on the Koopman operator," in *Proceedings of the 55th IEEE Conference on Decision and Control*, 2016, pp. 6500–6505. DOI: 10.1109/CDC.2016.7799269.

[72] P. van Goor, R. Mahony, M. Schaller, and K. Worthmann, "Reprojection methods for Koopman-based modelling and prediction," in *Proceedings of the 62nd IEEE Conference on Decision and Control*, 2023, pp. 315–321. DOI: 10.1109/CDC49753.2023.10383796.

[73] N. Takeishi, Y. Kawahara, and T. Yairi, "Learning Koopman invariant subspaces for dynamic mode decomposition," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 1130–1140.

[74] M. Haseli and J. Cortés, "Learning Koopman eigenfunctions and invariant subspaces from data: Symmetric subspace decomposition," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3442–3457, 2022. DOI: 10.1109/TAC.2021.3105318.

[75] W. D. Penny, J. Mattout, and N. J. Trujillo-Barreto, "Bayesian model selection and averaging," in *Statistical Parametric Mapping: The Analysis of Functional Brain Images*, K. J. Friston, J. T. Ashburner, S. J. Kiebel, T. E. Nichols, and W. D. Penny, Eds., 1st ed., Academic Press, 2007, pp. 454–467, ISBN: 978-0-12-372560-8. DOI: 10.1016/B978-012372560-8/50035-8.

[76] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing Koopman theory to allow for inputs and control," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018. DOI: 10.1137/16M1062296.

[77] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PLOS ONE*, vol. 11, no. 2, Feb. 2016, Art. no. e0150171. DOI: 10.1371/journal.pone.0150171.

[78] W. Zhao, J. Panerati, and A. P. Schoellig, "Learning-based bias correction for time difference of arrival ultra-wideband localization of resource-constrained mobile robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3639–3646, 2021. DOI: 10.1109/LRA.2021.3064199.

[79] Y. Tao, L. Wu, J. Sidén, and G. Wang, "Monte Carlo-based indoor RFID positioning with dual-antenna joint rectification," *Electronics*, vol. 10, no. 13, 2021. DOI: 10.3390/electronics10131548.

[80] A. Alarifi et al., "Ultra wideband indoor positioning technologies: Analysis and recent advances," *Sensors*, vol. 16, no. 5, 2016. DOI: 10.3390/s16050707.

[81] M. A. Shalaby et al., "MILUV: A multi-UAV indoor localization dataset with UWB and vision," *The International Journal of Robotics Research*, 2026. DOI: 10.1177/02783649251405898.

[82] R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science* (Cambridge Series in Statistical and Probabilistic Mathematics), 1st ed. Cambridge University Press, 2018, vol. 47, ISBN: 978-1-108-41519-4. DOI: 10.1017/9781108231596.

[83] J. N. Wong, D. J. Yoon, A. P. Schoellig, and T. D. Barfoot, "Variational inference with parameter learning applied to vehicle trajectory estimation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5283–5290, 2020. DOI: 10.1109/LRA.2020.3007381.

[84] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems*, vol. 20, 2007, pp. 1177–1184.

[85] V. Kotu and B. Deshpande, "Anomaly detection," in *Data Science*, V. Kotu and B. Deshpande, Eds., 2nd ed., Morgan Kaufmann, 2019, pp. 447–465, ISBN: 978-0-12-814761-0. DOI: 10.1016/B978-0-12-814761-0.00013-7.

[86] F. Dellaert and GTSAM Contributors, *borglab/gtsam*, version 4.2a8, May 2022. DOI: 10.5281/zenodo.5794541. [Online]. Available: https://github.com/borglab/gtsam.

[87] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*, vol. 18, 2006, pp. 1257–1264.

[88] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016. DOI: 10.1177/0278364916684382.

[89] A. Gijsberts and G. Metta, "Incremental learning of robot dynamics using random features," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 951–956, May 2011. DOI: 10.1109/ICRA.2011.5980191.

[90] J. H. Manton and P.-O. Amblard, "A primer on reproducing kernel Hilbert spaces," *Foundations and Trends in Signal Processing*, vol. 8, no. 1–2, pp. 1–126, 2015. DOI: 10.1561/2000000050.

[91] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf, "Kernel mean embedding of distributions: A review and beyond," *Foundations and Trends in Machine Learning*, vol. 10, no. 1–2, pp. 1–141, 2017. DOI: 10.1561/2200000060.

[92] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Computational Learning Theory*, D. Helmbold and B. Williamson, Eds., ser. Lecture Notes in Computer Science, vol. 2111, Berlin, Heidelberg: Springer, 2001, pp. 416–426, ISBN: 978-3-540-44581-4. DOI: 10.1007/3-540-44581-1_27.

[93] A. Tompkins and F. Ramos, "Fourier feature approximations for periodic kernels in time-series modelling," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018. DOI: 10.1609/aaai.v32i1.11696.

[94] C. Pezzato, R. Ferrari, and C. H. Corbato, "A novel adaptive controller for robot manipulators based on active inference," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2973–2980, 2020. DOI: 10.1109/LRA.2020.2974451.

[95] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50, no. 3, pp. 657–682, 2014. DOI: 10.1016/j.automatica.2014.01.001.

[96] C.-A. Cheng, H.-P. Huang, H.-K. Hsu, W.-Z. Lai, and C.-C. Cheng, "Learning the inverse dynamics of robotic manipulators in structured reproducing kernel Hilbert space," *IEEE Transactions on Cybernetics*, vol. 46, no. 7, pp. 1691–1703, 2016. DOI: 10.1109/TCYB.2015.2454334.

[97] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning). Cambridge, MA, USA: MIT Press, 2016, ISBN: 978-0-262-03561-3.

[98] S. Pan, N. Arnold-Medabalimi, and K. Duraisamy, "Sparsity-promoting algorithms for the discovery of informative Koopman-invariant subspaces," *Journal of Fluid Mechanics*, vol. 917, A18, 2021. DOI: 10.1017/jfm.2021.271.

[99] F. Messerer, K. Baumgärtner, and M. Diehl, "Survey of sequential convex programming and generalized Gauss–Newton methods," *ESAIM: Proceedings and Surveys*, vol. 71, pp. 64–88, 2021. DOI: 10.1051/proc/202171107.

[100] D. P. Tsakiris, K. Kapellos, C. Samson, P. Rives, and J.-J. Borrelly, "Experiments in real-time vision-based point stabilization of a nonholonomic mobile manipulator," in *Experimental Robotics V*, A. Casals and A. T. de Almeida, Eds., ser. Lecture Notes in Control and Information Sciences, vol. 232, Berlin, Heidelberg: Springer, 1998, pp. 570–581. DOI: 10.1007/BFb0112993.

[101] P. Polack, F. Altché, B. d'Andréa-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 812–818. DOI: 10.1109/IVS.2017.7995816.

[102] F. Dümbgen, C. Holmes, and T. D. Barfoot, "Safe and smooth: Certified continuous-time range-only localization," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1117–1124, 2023. DOI: 10.1109/LRA.2022.3233232.

[103] J. Djugash, B. Hamner, and S. Roth, "Navigating with ranging radios: Five data sets with ground truth," *Journal of Field Robotics*, vol. 26, pp. 689–695, Sep. 2009. DOI: 10.1002/rob.20311.

[104] B. Bischl et al., "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges," *WIREs Data Mining and Knowledge Discovery*, vol. 13, Jan. 2023. DOI: 10.1002/widm.1484.

[105] S. Dahdah and J. R. Forbes, *decargroup/pykoop*, version v1.2.4, 2022. DOI: 10.5281/zenodo.5576490. [Online]. Available: https://github.com/decargroup/pykoop.

[106] B. Bazzana, T. Guadagnino, and G. Grisetti, "Handling constrained optimization in factor graphs for autonomous navigation," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 432–439, 2023. DOI: 10.1109/LRA.2022.3228175.

[107] S. Teng et al., "GMKF: Generalized moment Kalman filter for polynomial systems with arbitrary noise," 2024. arXiv: 2403.04712 [cs.RO].

[108] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1100–1114, 2006. DOI: 10.1109/TRO.2006.886264.

[109] J. Pitman, *Probability* (Springer Texts in Statistics). Springer, 1993, ISBN: 978-0-387-97974-8. DOI: 10.1007/978-1-4612-4374-8.

[110] N. Boumal, *An Introduction to Optimization on Smooth Manifolds*. Cambridge University Press, 2023. DOI: 10.1017/9781009166164. [Online]. Available: https://www.nicolasboumal.net/book.

[111] R. Senanayake and F. Ramos, "Directional grid maps: Modeling multimodal angular uncertainty in dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3241–3248. DOI: 10.1109/IROS.2018.8594041.

[112] R. Senanayake, M. Toyungyernsub, M. Wang, M. J. Kochenderfer, and M. Schwager, "Directional primitives for uncertainty-aware motion estimation in urban environments," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, Rhodes: IEEE Press, 2020, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294288.

[113] K. Li, F. Pfaff, and U. D. Hanebeck, "Circular discrete reapproximation," in *Proceedings of the 25th International Conference on Information Fusion*, 2022, pp. 1–8. DOI: 10.23919/FUSION49751.2022.9841269.

[114] K. V. Mardia and P. E. Jupp, *Directional Statistics* (Wiley Series in Probability and Statistics). Wiley, 2000, ISBN: 978-0-471-95333-3. DOI: 10.1002/9780470316979.

[115] G. S. Watson, *Statistics on Spheres* (University of Arkansas Lecture Notes in the Mathematical Sciences). Wiley, 1983, vol. 6, ISBN: 978-0-471-88866-6.

[116] A. Müller and M. Bischoff, "A consistent finite element formulation of the geometrically nonlinear Reissner-Mindlin shell model," *Archives of Computational Methods in Engineering*, vol. 29, no. 5, pp. 3387–3434, Aug. 2022. DOI: 10.1007/s11831-021-09702-7.

[117] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). Springer, 2006, ISBN: 978-0-387-31073-2. DOI: 10.1007/978-0-387-45528-0.

[118] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 4th ed. Pearson, 2018, ISBN: 978-0-13-348979-8.

[119] H. M. T. Menegaz, J. Y. Ishihara, G. A. Borges, and A. N. Vargas, "A systematization of the unscented Kalman filter theory," *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2583–2598, 2015. DOI: 10.1109/TAC.2015.2404511.

[120] J. K. Uhlmann, "Dynamic map building and localization: New theoretical foundations," Ph.D. dissertation, University of Oxford, 1995.

[121] F. Dümbgen, C. Holmes, B. Agro, and T. Barfoot, "Toward globally optimal state estimation using automatically tightened semidefinite relaxations," *IEEE Transactions on Robotics*, vol. 40, pp. 4338–4358, 2024. DOI: 10.1109/TRO.2024.3454570.

[122] Z. Wang and G. Dissanayake, "Observability analysis of SLAM using fisher information matrix," in *Proceedings of the IEEE International Conference on Robotics, Automation and Computer Vision*, 2008, pp. 1242–1247. DOI: 10.1109/ICARCV.2008.4795699.

[123] K. Takahashi, J. Fagan, and M. S. Chen, "Formation of a sparse bus impedance matrix and its application to short circuit study," in *Proceedings of the 8th Power Industry Computer Application Conference*, 1973, pp. 63–69.

[124] T. D. Barfoot, J. R. Forbes, and D. J. Yoon, "Exactly sparse Gaussian variational inference with application to derivative-free batch nonlinear state estimation," *The International Journal of Robotics Research*, vol. 39, pp. 1473–1502, Jul. 2020. DOI: 10.1177/0278364920937608.

[125] S. P. Boyd and L. Vandenberghe, *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge, UK: Cambridge University Press, 2018, ISBN: 978-1-316-51896-0.

[126] F. H. Kong, J. Zhao, L. Zhao, and S. Huang, "Analysis of minima for geodesic and chordal cost for a minimal 2-D pose-graph SLAM problem," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 323–330, 2020. DOI: 10.1109/LRA.2019.2958492.

[127] C. Holmes and T. D. Barfoot, "An efficient global optimality certificate for landmark-based SLAM," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1539–1546, 2023. DOI: 10.1109/LRA.2023.3238173.