Adapting to Context in Robot State Estimation

Humphrey Hu

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
humhu@cmu.edu

Thesis Committee:
George Kantor, Chair
Sebastian Scherer
Katharina Muelling
Ingmar Posner, University of Oxford

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

CMU-RI-TR-18-14

April 25, 2018

Abstract

The promised future filled with robots sensing and acting intelligently in the world is near fruition, thanks in part to continuous progress in robotic perception and state estimation. However, a number of challenges remain before state estimation systems and the robots that rely on them can be considered truly reliable. In particular, we must consider what happens when highly complex hardware and software systems designed and validated in laboratory environments enter the unbounded variety of reality. Will these systems fail innocuously or catastrophically? If so, how can we avoid or eliminate these failures to achieve reliable, robust behavior?

The premise of this thesis is that engineering constraints and human finitude result in fallible systems that cannot compensate for all possible factors and situations. We refer to the collection of uncompensated factors as the *context* of a system, and propose that variations in context can explain why it is difficult to make state estimation reliable at scale. Vexingly, since context is, by nature, unknowable and unmodeled, we cannot rely on prediction and foresight to compensate for it.

Instead, this thesis proposes that state estimation systems can adapt their behavior *after* deployment to the operating site to correct for unknown contextual effects. An example of this is the widespread and common practice of "parameter tuning", typically performed by a human expert to specialize a system to each deployment. To generalize this and other mechanisms of adaptation, we first develop a general theory of context in estimation and establish a statistical definition for estimation performance. We then develop a practical method for evaluating performance on-site without supervision, enabling estimation systems to observe the effects of context during operation. Finally, we explore automatic parameter tuning and experience-driven failure prediction as two methods of general adaptation. We demonstrate and validate this work on state estimation systems using offline data from an instrumented automobile as well as online an indoor ground robot.

Acknowledgements

First and foremost, I would like to thank all of my friends and family for supporting me throughout my studies. To say that this process has been challenging would be an understatement, and it's unlikely I would have made it this far without the closest people in my life always hearing my complaints, helping take my mind of work, and forcibly feeding me copious amounts of ice cream.

I would like to thank all of the staff at the Robotics Institute for making this work possible through their efforts in building and maintaining the fantastic robots I've loved and hated, often simultaneously. In particular, I'd like to thank Srinivasan Vijayarangan and Trevor Decker for helping me with C++ incantations, Chuck Whittaker for allowing me to use way too much of the high bay, Jim Picard for all of his help in the machine shop, David Kohanbash for saving me from networking nightmares and battery fires, and Alex Long for making sure the wheels never fell off the robots.

Thanks also to my fellow students for all of our mind-stretching discussions and other tomfoolery. I owe Vishnu Desaraju for convincing me that Gaussian processes are useful, Abhijeet Tallavajhula for introducing mathematical rigor into my work, Kumar Shaurya Shankar for making me a believer in vision, Leo Keselman for encouraging me to give deep learning a second chance, and Karthik Lakshmanan and Arun Venkatraman for dozens of hours of brainstorming and proofreading.

I've also had the great pleasure of learning from and working with many brilliant and exceptional faculty, both at Carnegie Mellon and around the world. I'd specifically like to thank my thesis committee members, Sebastian Scherer, Katharina Muelling, and Ingmar Posner, as well as Howie Choset, David Wettergreen, and Reid Simmons for their advice throughout the years.

And last but not least, I would like thank my adviser George Kantor for putting up with me for all these years. Though he is not a famous mathematician like Georg Cantor, he is a renowned beer afficionado, published rock star, recognized black belt, and all-around fantastic mentor to all of his students. They say you can't beat entropy, but somehow George has managed to guide my spastic energies into something useful with his limitless patience, wisdom, and perspective.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Continual advances in perception have finally propelled robots to the cusp of practicality. Leaving the safe confines of the laboratory, however, raises a number of fundamental questions, particularly with regards to the generality and reliability of perception systems: How will a system that works well in test conditions fare when deployed at scale into the real world? If the system fails, will it be catastrophic or minor? Can we understand the cause of failure enough to prevent it in the future?

To answer these questions we must first understand that perception does not occur in a vacuum; everything around and within a perception system affects its behavior. When designing a perception system, we typically think of systematic, environmental, and behavioral factors, such as the available computational resources, local scene textures, and robot velocity. Much progress in perception has come by the way of developing better techniques to estimate, model, and compensate for these factors. Thus, it is not surprising for perception to comprise the bulk of complexity in a modern robotic system.

This approach of explicit compensation works well when the critical factors are known and their quantity and variety can be controlled. An extreme example of this is in manufacturing, where careful engineering allows some robotic systems to operate with almost no perceptive capabilities at all. However, with more unstructured deployments, such as in residential homes, it becomes impractical to predict and compensate for every possible factor. Instead, engineering finitude forces us to choose between incorporating a factor as part of the system's *belief state* where it can be explicitly compensated for, or being agnostic to a factor as part of the system's operating *context*.

Ideally, context should have a negligible effect on the behavior of the system. However, it may happen that relatively important factors are relegated to context, either by necessity or a lack of awareness. In these cases, differences in validation and deployment contexts may result in poor perception performance so that a system which works perfectly in a test laboratory may still fail catastrophically upon deployment. This sets us up for the perception engineer's equivalent of Pascal's Wager, where instead of betting about the existence of

God, we gamble on the importance of context.

As described, the context wager is a losing gamble in which every deployment is a roll of the context failure dice. This rather negative outlook is because we do not consider remedial actions that can be taken after deployment. In fact, the ubiquitous mechanism of parameter tuning is intended exactly for dealing with context, and is largely performed on-site.

From an engineering perspective, designing systems with behavior that is set by various parameters is a natural way to enable reuse and specialization. In our view of perception, parameter tuning can also be understood as a mechanism for rapid adaptation to context. Instead of entirely redesigning a system for each deployment, an engineer can simply tune parameters until the desired behavior is achieved. The key to parameter tuning is that it is performed on-site, where the engineer can iterate between changing the parameters and observing the system behavior, instead of relying on pre-deployment predictions. This allows parameter tuning to compensate for a wide variety of unpredictable context effects.

Parameter tuning's efficacy speaks to the flexibility of easily-changeable parametric perception behavior, but also to the power of adaptation driven by empiricism. We do not claim that our field has reached the limits of modeling and forecasting, but rather propose that adaptation is underutilized. More generally, this thesis seeks to develop and explore mechanisms, such as parameter tuning, that enable perception systems to adapt to their deployment contexts. In particular, we study adaptation for state estimation systems, a subset of general perception systems. Formally, the thesis of this work is:

> *State estimation systems can mitigate the effects of context on performance by adapting their behavior after deployment.*

Perception and state estimation are both expansive topics, and while our work is developed to be as general as possible, we restrict our validation demonstrations and experiments to a few tasks and systems for practicality. Specifically, we consider the task of state estimation on ground vehicles operating in somewhat structured 2D environments, but propose that this work can generalize to more complex systems with a few implementation extensions. More details about our experimental systems can be found in Chapter 2

1.1 Challenges Addressed and Overall Approach

There are a number of technical challenges in compensating for context through adaptation, certainly beyond the scope of a single work. This thesis focuses on the following core challenges to establish a basis for future work on this subject.

Unavailability of feedback and supervision

Feedback and supervision quantifying performance is crucial for meaningful adaptation. A number of techniques exist for measuring performance in lab-

oratory settings, but rarely are these practical for use on-site. For instance, it is unlikely that motion capture systems or human labelers will be available to provide ground truth in a real deployment.

To address this challenge, we first develop a formal treatment in Chapter 3 of perception performance by borrowing concepts from reinforcement learning and estimation theory. In Chapter 4, we then propose a statistical technique utilizing the outputs of Bayesian estimators, already commonplace in perception, for quantifying perception performance without ground truth by using empirical trials.

Limited data and computational resources

As part of a robotic deployment, we desire that adaptation take as little time as possible, as it is simply impractical to require days of data collection for adaptation before the system can be used. Similarly, an adaptation scheme will have to operate largely onboard, or at least on-site. This restricts the scale of computational resources we can use, as most robots are not mobile datacenters.

To respect these constraints, we rely on models learned from data collected on-site to intelligently drive adaptation. Primarily, we use models as a way to generalize experiences such that we can seek out trends resulting in good performance and avoid correlations with bad performance. In the case of tuning parameters, discussed in Chapter 5, we use a model to predict performance for different parameter values. This allows us to focus time searching promising areas of the parameter space. For the failure prediction approach in Chapter 6, we use a model to generalize past failure instances and detect future failures.

Complex and stochastic behavior

As discussed in the previous challenge, we desire a reasonably good model that can predict the effect of different actions on system performance. However, modeling perception system behavior is challenging due to the large number of relevant factors and tunable parameters, compounded by stochasticity from noisy sensors and random algorithms. As an example, Dequaire et al. [2016] showed that it is possible to predict visual odometry performance with respect to a range of outdoor lighting conditions. Extending that same model to include the effects of various system parameters in an indoors setting, however, is quite challenging.

We observe that many factors can be considered constant when adapting on-site. Specifically, we can imagine limiting contextual variation and stochasticity to focus on the effect of parameters, or fix the parameters to build a model for context and stochasticity. For instance, the approach for efficiently tuning parameters in Chapter 5 relies on a model that predicts performance for different parameter values using an evaluation method that limits contextual variation and stochasticity. Similarly, the approach for predicting future performance from sensor data in Chapter 6 fits a model for a particular set of parameter values.

1.2 Contributions

In summary, the primary contributions of this work are:

1. A general theoretical framework for thinking about context in perception and its effect on performance
2. A practical and general technique for evaluating perception performance without ground truth
3. An application of stochastic black-box optimization algorithms to efficient automated parameter tuning
4. Local-learning approaches for predicting stochastic perception performance using on-site experiences
5. Extensive validation on multiple real world systems

Chapter 2

Experimental Systems

Throughout this thesis we demonstrate and validate our approaches by applying them to state estimation systems operating on two different experimental platforms. We describe our core state estimation system and each of the platforms in detail below.

2.1 Test Platforms

In order to test the efficacy of our approaches in compensating for context, we must experiment with systems that actually interface with the real world. We use two such experimental platforms throughout this work, one a physical robot and the other a dataset collected from an instrumented vehicle. It is particularly illustrative to run online experiments in a wide variety of environments on the physical robot, but also quite time consuming. As such, we perform most of our comparative studies on the dataset system and perform validation and demonstration on the physical robot.

2.1.1 Intelligent Mobile Robot (IMR)

Our hardware system is a custom-built indoor ground robot used for industrial automation projects with an omnidirectional "mecanum" wheel drivetrain, and is shown in Fig. 2.1. The robot is able to move at up to 1.0 m/s and can operate for approximately 1 h continuously. Being sized to fit through a standard doorway, the IMR is a relatively mobile platform, so we are able to run experiments in a wide variety of environments around the Carnegie Mellon campus. However, these experiments are time-intensive, so we use this platform primarily for demonstrative and validation experiments.

The IMR has a variety of sensors to support reliable odometry and localization in different environments. This work focuses primarily on the odometry systems, using a downward-facing monocular camera, an upward-facing stereo camera, and two side-facing planar laser rangefinders. The downward-facing

camera is an IDS UI-3140CP USB 3.0 camera capturing 400×400 resolution frames at 200 frames per second. We employ high-intensity lighting under the robot with low camera exposure times to minimize the effect of motion blur. The stereo camera is a StereoLabs ZED camera mounted on the front of the robot facing directly upward. The camera outputs synchronized VGA (640×480) resolution image pairs at 120 Hz. The laser rangefinders are two Hokuyo URG-04LX-UG01 planar rangefinders primarily for collision avoidance, but which are also used for laser odometry. The rangefinders are located on opposite corners of the robot and each scans 240° at 10 Hz with a maximum range of 5.6m. Example data from these sensors can be seen in Figure 2.2. We also have an Invensense MPU-6050 IMU onboard providing 3-axis acceleration and angular velocity data at 50 Hz.

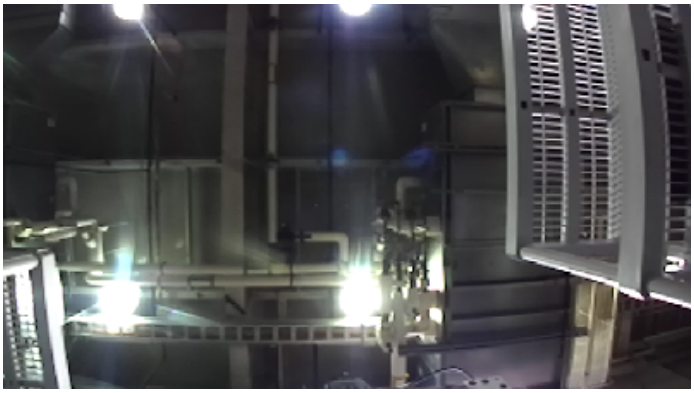
Computation is spread throughout the robot subsystems. The robot base has one Intel Core i5-4250U 2-core computer interfacing with the motion controllers and running control and planning software, and another dedicated to processing the visual and laser odometry sensors. The stereo camera is processed by a larger Intel Core i7 4-core computer. All onboard computers are connected over ethernet to an onboard WiFi router.

In a portion of our experiments we use a four camera Vicon Bonita motion capture setup for ground truth[1]. The system provides pose data at 100 Hz, which we differentiate to produce ground truth body velocities at 10 Hz. A dedicated laptop computer runs the motion capture processing software and broadcasts the pose data across WiFi to the robot where it is timestamped upon reception. This introduces a small amount of latency in the ground truth velocity signal that we do not compensate for, as it is negligible on the scale of the robot dynamics.

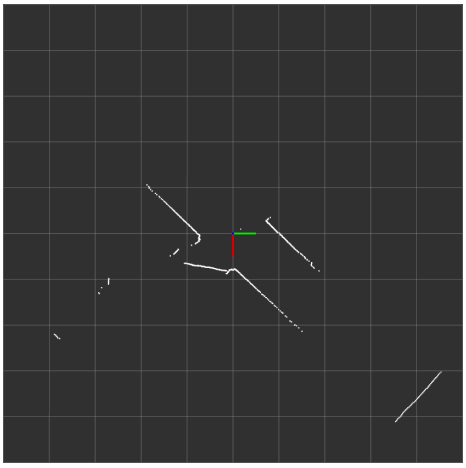
[1] `http://www.vicon.com`

Figure 2.1: The Intelligent Mobile Robot (IMR) platform

(a) Example image from downward camera



(b) Example image from upward camera



(c) Example laser scan

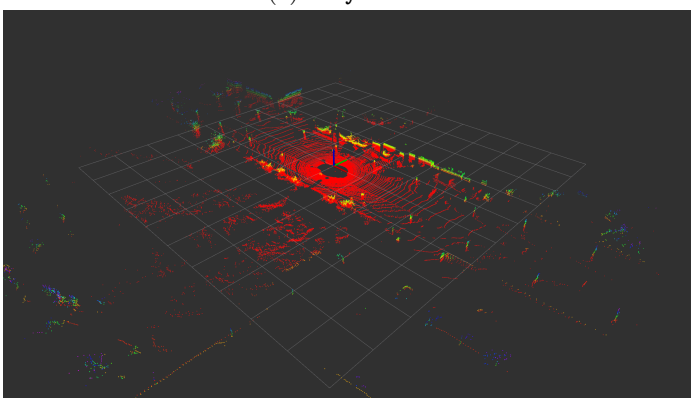
Figure 2.2: Example data from the IMR odometry sensors.

2.1.2 KITTI Vision Benchmark Dataset

Our second platform is the ubiquitous KITTI dataset, named for the Karlsruhe Institute of Technology (KIT) and Toyota Technological Institute (TTI) Geiger et al. [2013]. This dataset consists of sensor data sequences recorded from a car driving around the streets of Karlsruhe, Germany, and is commonly used as a benchmark of vision and laser-based odometry algorithms. We playback data from the dataset in real-time to simulate state estimation on the vehicle, since this highlights computational constraints that are not apparent in a completely offline setting. We run this system on a desktop computer with an 8-core Intel Core i7 processor and 16 GB of RAM. Examples of the dataset are shown in Figure 2.3.

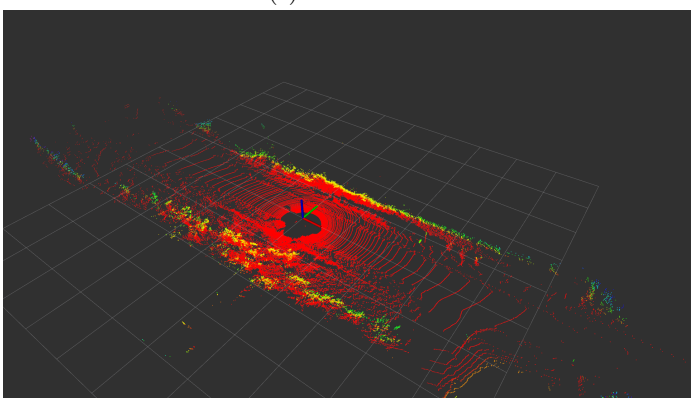
We use the forward-facing stereo cameras, the laser rangefinder, and the IMU data from the dataset. The stereo cameras each output undistorted grayscale 1242×375 resolution images over a wide horizontal field of view at 10 Hz. The laser is a Velodyne HDL-64E spinning laser rangefinder that outputs approximately 1.3 million points per second. The laser output is preprocessed into motion-corrected $130,000$-point scans at 10 Hz, synchronized to the camera frames. The IMU is part of an onboard OXTS RT 3003 localization system and provides $3D$ linear acceleration and angular velocity measurements, also synchronized and at 10 Hz. Finally, highly accurate ground truth velocities are available from the OXTS system, which incorporates the IMU with GPS and GLONASS satellites with RTK corrections.

(a) `city` frame



(b) `city` scan

(c) `road` frame



(d) `road` scan

Figure 2.3: Frames and scans from the two odometry dataset contexts used in our KITTI test system.

2.2 Software Systems

Our experiments all use a common state estimation system that estimates the velocity of the robot from sensor data. The software system is designed such that removing or adding different processing subsystems is straightforward, allowing us to test our approaches on each of the subsystems separately. We describe below the core state estimation system as well as each of the subsystems. All of our software is open sourced and can be found at `https://github.com/Humhu`.

2.2.1 Core State Estimation System

The core of our state estimation system is an adaptive Kalman filter that fuses velocity observations from each subsystem into a single estimate of the robot body velocity. We use a buffer architecture to support out-of-order observations, where a fixed lag of observations are buffered and reprocessed on a copy of the filter for each time step.

The filter estimates the observation covariance for each observation source by using a sliding time window over previous observation prediction errors. Our implementation differs from the standard AKF by weighting recent observations more to allow faster adaptation, and also by using a prior model that slowly blends to and from a fixed prior covariance when there are no observations in the window. We use fixed covariances for the process transition as well as the IMU observations, as we find this helps condition the other covariance estimators.

2.2.2 Sparse Planar Monocular Visual Odometry (SPM-VO)

SPM-VO is a point-based visual odometry system intended to be used with the IMR's floor-facing camera. This system performs Lucas-Kanade tracking on a regularly-spaced grid of points to find correspondences in the previous frame. If the correspondence image similarity error is lower than a threshold, the system then estimates a rigid $2D$ transformation with RANSAC between the images. Since the camera is at a fixed height off of the ground, the translation scale is constant and can be solved for during calibration. An illustration of the tracking can be seen in Figure 2.4.

We use the OpenCV library's implementation of Lucas-Kanade with pyramids and rigid transformation estimation[2]. The majority of the tunable parameters for this system relate to the Lucas-Kanade tracker, with a few other parameters specifying the point grid density and the RANSAC behavior.

[2] `http://opencv.org/`

Table 2.1: SPM-VO tunable parameters

Parameter	Type	Values
Point grid dimension	`int`	$\in [5, 30]$
LK min solver improvement	`float`	$\in [10^{-6}, 1.0]$
LK search window	`int`	$\in [10, 40]$
LK pyramid level	`int`	$\in [0, 5]$
LK max solution error threshold	`float`	$\in [0, 7.5]$
RANSAC max error	`float`	$\in [0.0, 0.05]$

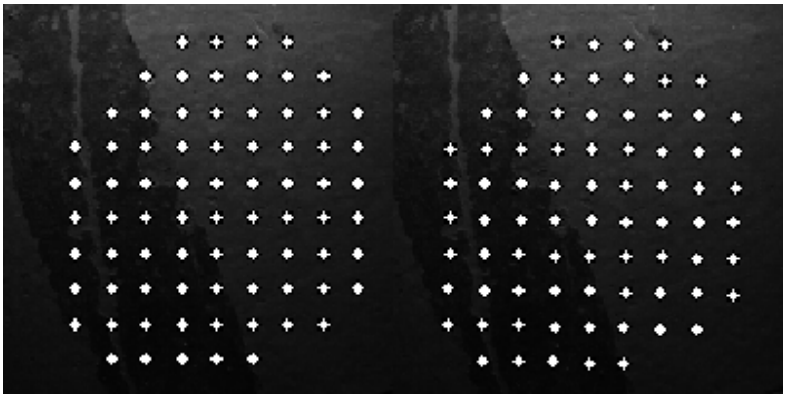


Figure 2.4: Example illustrating SPM-VO operation. Points in the keyframe (left) are tracked to the current frame (right) to estimate camera motion.

2.2.3 Dense Planar Monocular Visual Odometry (DPM-VO)

DPM-VO is the dense counterpart to SPM-VO. This system estimates camera motion by finding a rigid transformation that minimizes the per-pixel differences between a current and keyframe image. Since the computational complexity of this alignment depends heavily on the image size, we first linearly downsample the image. The alignment is then initialized using the displacement predicted by the velocity filter, and run until convergence or termination. If the alignment error surpasses a threshold, the result is discarded, the keyframe reset to the previous frame (if available), and alignment is re-attempted. Alternatively, if the keyframe is determined to be farther than a threshold distance away, it is reset. An illustration of this process is shown in Figure 2.5.

Like SPM-VO, we use the OpenCV library for image downsampling and ECC-based alignment. The majority of tunable parameters for this system deal with the ECC image alignment algorithm, with the remainder controlling the camera capture, image downsampling, and keyframing behavior.

Table 2.2: DPM-VO tunable parameters

Parameter	Type	Values
Camera gain	`int`	$\in [0, 100]$
Camera exposure time (ms)	`float`	$\in [0, 3]$
Image downsample scale	`float`	$\in [0.25, 1.0]$
ECC pyramid depth	`int`	$\in [0, 2]$
ECC max iterations	`int`	$\in [10, 100]$
ECC $\log_{10}$ objective tolerance	`float`	$\in [-4, -2]$
ECC $\log_{10}$ min correlation	`float`	$\in [-3, -1.875]$
Max keyframe movement	`float`	$\in [0.05, 0.25]$

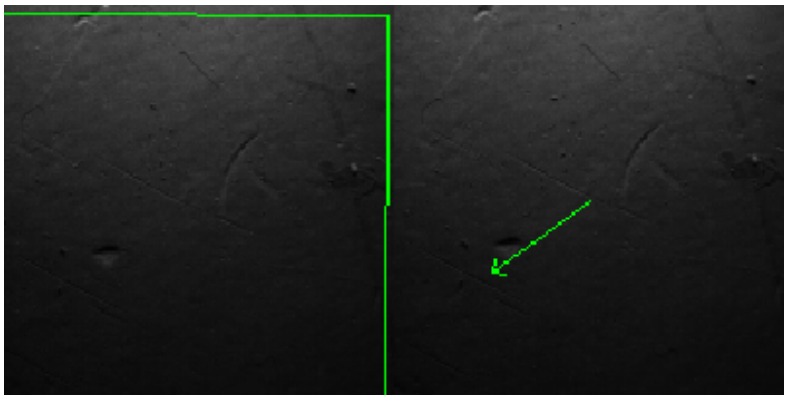


Figure 2.5: Example illustrating DPM-VO operation. The current frame (right) is aligned to the keyframe (left) with the pose indicated by the green box, resulting in estimated velocity shown by the arrow.

2.2.4 ICP Laser Odometry (ICP-LO)

ICP-LO aligns laser scans against keyframe scans to estimate the motion of the laser scanner. Similarly to DPM-VO, the alignment process complexity scales dramatically with the number of points in the scan, so we use an approximate voxel filter to produce a spatially-even downsampling of scans. The scan is checked for symmetries and degeneracies, e.g. parallel lines and circles. If no degeneracies exist, we then use the iterative closest point (ICP) algorithm to align the current scan against the keyframe scan with an outer RANSAC loop to compensate for outliers. This alignment process is initialized using the displacement predicted by the velocity filter. If the final alignment error is too high or the proportion of inliers is too low, the keyframe is reset to the previous scan (if available) and alignment is re-attempted. The keyframe is also reset automatically once its age exceeds a specified time limit. An illustration of this process is shown in Figure 2.6.

We use the Point Cloud Library (PCL)[3] implementation of ICP, which features a large number of numerical parameters. The remaining tunable parameters for this system concern the voxel filter and error thresholding.

Table 2.3: ICP-LO tunable parameters

Parameter	**Type**	**Values**
$\log_{10}$ voxel filter width	float	$\in [-2, 0]$
ICP max iterations	`int`	$\in [10, 100]$
ICP max corresp. distance (m)	`int`	$\in [0, 1]$
ICP max solution error	`float`	$\in [0.01, 1.0]$
ICP $\log_{10}$ objective tolerance	`float`	$\in [-6, -3]$
ICP min inlier ratio	`float`	$\in [0.5, 0.95]$
RANSAC iterations	`int`	$\in [0, 100]$
RANSAC inlier distance (m)	`float`	$\in [0.1, 1.0]$

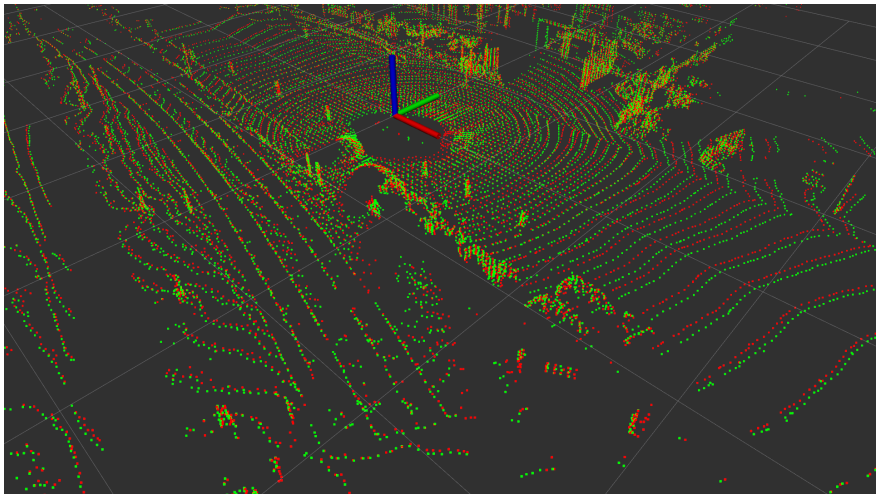


Figure 2.6: Example illustrating ICP-LO operation on the KITTI dataset. The current frame (red) is aligned to the keyframe (green) with the laser scanner pose indicated by coordinate axes.

2.2.5 Sparse Stereo Visual Odometry (Fovis)

The Fovis visual odometry library Huang et al. [2017] finds point correspondences between image pairs and then tracks these points across consecutive frames to estimate the 3D motion of the camera. An illustration of the algorithm is shown in Figure 2.7.

[3] `http://pointclouds.org/`

We use the open source core library[4] and ROS interface[5], modified slightly to support parameter reconfiguration and data replay in our test system. When operating on the IMR, we can additionally tune the camera capture parameters.

Table 2.4: SS-VO tunable parameters

Parameter	**Type**	**Values**
Feature window size	int	$\in [3, 21]$
Feature search window	`int`	$\in [5, 50]$
Max pyramid level	`int`	$\in [0, 5]$
Target pixels per feature	`int`	$\in [150, 350]$
FAST threshold	`int`	$\in [5, 70]$
Bucket size	`int`	$\in [20, 100]$
Max keypoints per bucket	`int`	$\in [10, 40]$
Inlier reprojection threshold	`float`	$\in [0.25, 3.0]$
Clique inlier threshold	`float`	$\in [0.01, 0.2]$
Min feature threshold	`int`	$\in [10, 30]$
Mean reprojection threshold	`float`	$\in [5.0, 15.0]$
Max epipolar distance	`float`	$\in [0.5, 5.0]$
Max refinement distance	`float`	$\in [0.5, 2.0]$
Max disparity	`int`	$\in [60, 200]$

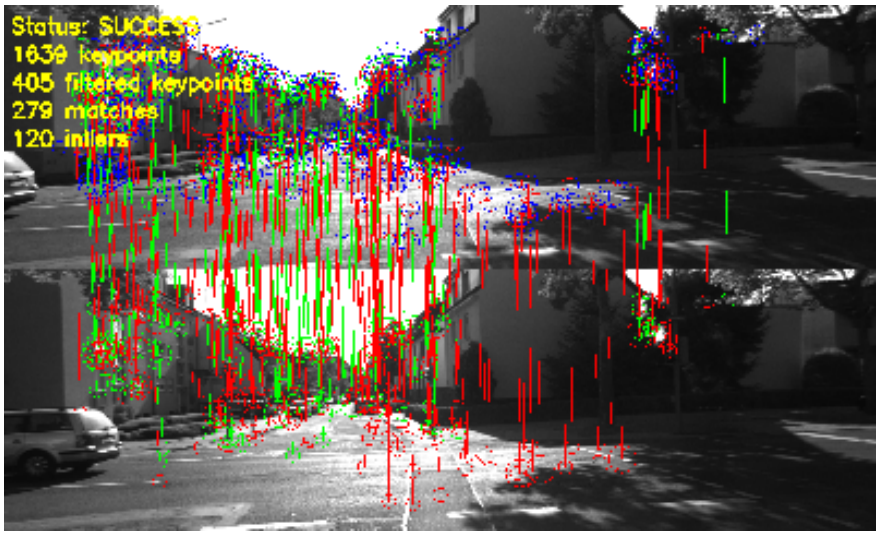


Figure 2.7: Example illustrating SS-VO operation on the KITTI dataset. Keypoints (circles) are matched between the left (top) and right (bottom) images, with inliers shown by green lines and outliers shown by red lines.

[4] `http://fovis.github.io/`
[5] `http://wiki.ros.org/fovis_ros`

2.2.6 Dense Stereo Visual Odometry (DS-VO)

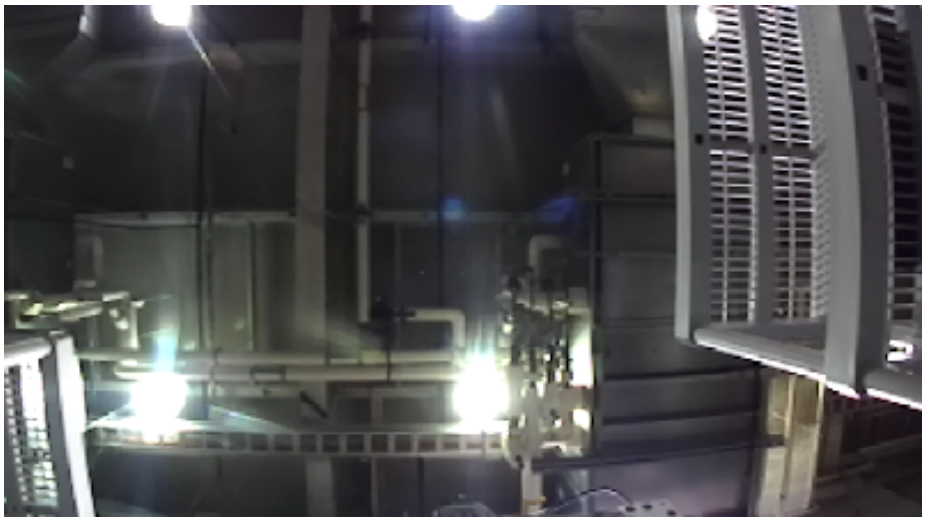
Our dense analogue to the Fovis stereo system combines the BitPlanes Visual Odometry (BPVO) library Alismail et al. [2016] with a block-matching disparity estimation algorithm to provide dense stereo odometry. Image pairs are processed into a disparity (inverse depth) image by finding small image blocks with similar appearance. The disparity image and corresponding original grayscale image are then aligned to a keyframe scene to estimate the 3D motion of the camera. We use the displacement predicted by the velocity filter to initialize this alignment process.

We modify the open-source BPVO library[6] to support parameter reconfiguration, and wrap it in our own software for velocity prediction and ROS interfacing. We use the ROS `stereo_image_proc` package for disparity estimation[7], which already supports parameter reconfiguration. The block-matching and BPVO algorithms contribute slightly under half of the overall tunable parameters each, with a few parameters dealing with the camera capture and final error thresholding.

Table 2.5: DS-VO tunable parameters

Parameter	**Type**	**Values**
Camera gain	int	$\in [0, 8]$
Correlation window size	`int`	$\in [5, 25]$
Disparity search range	`int`	$\in [64, 128]$
Min uniqueness ratio	`float`	$\in [10.0, 30.0]$
Texture threshold	`int`	$\in [600, 1000]$
Speckle filter size	`int`	$\in [100, 400]$
Speckle range	`int`	$\in [0, 31]$
Optimization $\log_{10}$ tolerance	`float`	$\in [-8, -3]$
Max solution error	`float`	$\in [0.02, 0.25]$
Min saliency	`false`	$\in [0.05, 0.5]$

[6] `https://github.com/halismai/bpvo`
[7] `http://wiki.ros.org/stereo_image_proc`



(a) Raw image from left camera

(b) Disparity image generated from image pair. Red is largest disparity (closer) and purple is smallest (farther).

Figure 2.8: Disparity image generated from Figure 2.2b used as input for BPVO algorithm.

Chapter 3

Formalizing Perception Context and Performance

To have a meaningful discussion about perception performance and context, we must first establish a reasonable theoretical framework. In this chapter we formally define terms and concepts relating to perception and context that we use throughout this work.

3.1 A Probabilistic Model of Perception

We begin by formalizing the perception task itself, which we define as processing observations in order to estimate an unknown *latent* of interest. The nature of the latent changes between applications. For instance, in state estimation the latent is typically the position or velocity of a robot, while in classification the latent is a discrete object class. Let x denote the latent variable belonging to a latent space $\mathbf{X}$, and let $\hat{x}$ denote estimates of the latent, also belonging to the latent space $\mathbf{X}$.

To estimate the latent, the perceptual system collects data referred to as *observations*. Let z denote an observation belonging to an observation space $\mathbf{Z}$. For generality, we understand that an observation may be a single piece of data, such as an image captured by a camera, a sequence of data, such as a waveform from an IMU, or data in another more abstract form.

3.1.1 The Standard Model of Perception

In the standard latent model of perception, shown in Figure 3.1, the latent generates an observation which is then processed to produce an *estimate* of the latent $\hat{x} \in \mathbf{X}$. Since the estimation process may be stochastic if the algorithms used involve randomness, e.g. random initializations, stochastic gradient descent, etc., we explicitly represent the estimate in the graphical model.

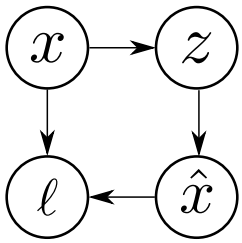


Figure 3.1: The standard latent model of perception. The latent x generates observation z, which in turn generate an estimate $\hat{x}$ which is used with the latent x to compute loss ℓ.

We quantify the quality of a latent estimate with an application-specific *loss* function $\ell(\cdot, \cdot) : \mathbf{X} \times \mathbf{X} \mapsto \mathbf{R}$, which compares the estimated and true latent values and returns the error as a scalar. For example, the sum squared error (SSE), mean squared error (MSE), and root mean squared error (RMS) are all commonly used losses in state estimation, whereas the indicator, hinge, and logistic losses are often used for classification.

Since we use a loss-based formulation (as opposed to reward-based), our overall goal in perception is to generate estimates with low loss. Accordingly, much work in perception focuses on the fidelity of the estimate $\hat{x}$ derived from observations z. Instead, we are focused on better understanding what factors affect the generation of both the observations and estimates themselves, and correspondingly their downstream effects on the performance.

3.1.2 Tunable Parameters

We now consider *tunable parameters*, variables within the control of the perception system that affect the system behavior. For a system with M tunable parameters, we denote the parameters as $c_1, \ldots, c_M$ where each parameter c_i belongs to a corresponding parameter space $\mathbf{C}_i$. We refer to a set of parameters as a *configuration* $\theta = \{c_1, \ldots, c_M\}$ belonging to the *configuration space* $\theta = \prod_{i=1}^{M} \mathbf{C}_i$. Accordingly, *reconfiguration* is the act of changing the tunable parameters.

Tunable parameters are typically discrete, continuous, or categorical in nature. For example, discrete parameters may include integer-valued parameters, such as image search windows, while floating-point parameters, such as convergence criteria, can be thought of as continuous. Finally, toggle parameters that enable or disable functionality, or selections from a group, such as selecting one out of many error functions, are categorical. In this work we only consider continuous parameters and discrete parameters approximated by continuous values.

The configuration of a perception system can affect both the generation of observations and of the latent estimate. For instance, the exposure setting of a camera affects the images captured and the convergence threshold for a tracking algorithm affects the displacement estimated from images. Thus we

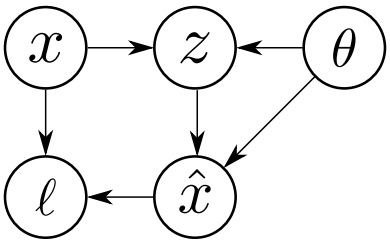


Figure 3.2: The model with effects from tunable parameters added. The parameters θ affect the generation of observations z with the latent x, and the generation of estimates $\hat{x}$ with the observations z.

can naturally introduce the effects of tunable parameters into the perception model as a dependency between the current configuration θ and the observations z and latent estimate $\hat{x}$. This is shown in Figure 3.2.

3.1.3 Context

Next we consider the effects of context. Formally, we define *context* as all variables, aside from the latent and tunable parameters, that affect the generation of observations. In other words, the latent, configuration, and context jointly generate the observations, as shown in Figure 3.3. As previously mentioned, variables that are explicitly compensated for are considered as part of the latent, whereas the unmodeled remainder form the context. Our definition of context is closely related to *nuisance variables* in statistics, quantities that are statistically important but otherwise undesired for estimation. For convenience, let us represent context abstractly as ϕ in some undefined context space $\mathbf{\Phi}$.

Explicitly representing context is challenging as it may consist of many unpredictable or unobservable factors. It may be possible to determine a context representation by finding correlations with observable quantities which explain observation variations, but this is outside the scope of this work. Rather, our goal is to understand how context can affect perception, and then develop techniques to compensate for unknown context in certain reasonable situations. In particular, we consider operating with a stationary distribution of contexts, which we refer to as a *deployment*, denoted by $p(\phi)$. The context distribution is typically unknown and alters the system behavior from a laboratory setting, forcing us to adapt on-site.

3.2 Defining Performance

3.2.1 For Independent Executions

As previously mentioned, we quantify the quality of a single latent estimate with a loss function. However, perception is stochastic, with varying latents and observations, so we imagine that a measure of performance should take

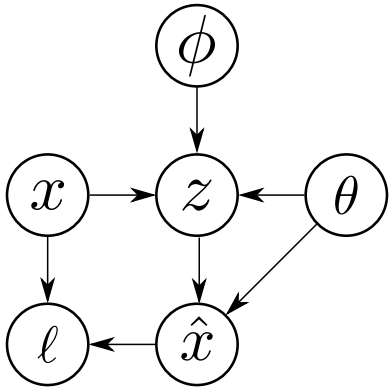


Figure 3.3: The model with effects from tunable parameters and context. The context ϕ affect the generation of observations z with the latent x and parameters θ.

into account the possible loss for various situations. This intuition can be succinctly captured by defining *performance* ρ as the expected loss over latents and estimates for a particular configuration θ:

$$\rho = E_{x,\hat{x}|\theta}\left[\ell(x, \hat{x})\right] \tag{3.1}$$

Note that variations in observations and context do not appear explicitly, as they are marginalized out. This is directly analogous to the Bayesian estimation theory concept of risk, which describes the expected loss over estimator inputs and outputs. With this definition, the performance can be affected by changing the configuration to affect the distribution of observations and latent estimates.

3.2.2 For Stateful Systems and Processes

Note that Equation 3.1 relies on independent perception instances. This is rarely applicable to robotic systems, which perceive the world as part of a physical process, and thus have highly correlated inputs and outputs. To compensate for this correlation, we can borrow from the reinforcement learning literature and define the performance ρ_t at a discrete time t as the expected discounted sum of losses from t onward:

$$\rho_t = \frac{E_{x,\hat{x}|\theta}\left[\sum_{i=0}^{T-1} \gamma^i \ell(x_{t+i}, \hat{x}_{t+i})\right]}{\sum_{i=0}^{T-1} \gamma^i} \tag{3.2}$$

where T is a finite horizon length and $\gamma \in [0, 1]$ is the *discount factor*. Intuitively, a larger discount factor corresponds to estimates more greatly affecting future losses. This captures the behavior of systems with high sensing rates, high overlap between consecutive data, and slow physical evolutions, all of which exhibit stronger correlations in time. In contrast, systems operating offline on

unordered data with no correlations, such as offline classifiers or detectors, can still be modeled with $\gamma = 0$, which recovers Equation 3.1.

3.3 Case Studies

To illustrate these concepts, we analyze a few of the test systems described in Chapter 2, as well as a typical object classification system.

3.3.1 Floor-facing Visual Odometry System

We consider first the floor-facing visual odometry system described in Section 2.2.3. This system registers consecutive images of the floor in order to maximize raw appearance correlations and estimate the 2D motion of the camera. The latent for this system consists simply of the camera motion, as nothing else is estimated or modeled, and the observations are simply the image pairs.

Given the minimal nature of this system's latent, there are a number of potentially important factors that we can consider contextual. For instance, the system assumes that the floor is planar, but this may not always be true. Further, this system relies on capturing enough appearance texture to perform registration. If the floor surface is more reflective or absorbant than expected, the image may not have enough texture. The dense registration algorithm is also relatively computationally-intensive. If additional software is deployed on the computer hosting this system's software components, the perception performance may be degraded.

We can imagine compensating for these contextual factors in a number of ways. Motion estimates from slightly non-planar floor patches can be rejected with an appropriately selected error threshold, or identified and avoided altogether. The camera exposure and gain can be adjusted to compensate for lighting properties, and the camera images can be downsampled when the computational resource availabilities change.

Finally, given the camera's framerate, field of view, and height off the ground, we know that consecutive images have a large amount of overlap between them. Thus we expect that observations, and thus estimates and losses, will be highly correlated in time. Further, previously estimated camera motions are used to initialize the image registration process, resulting in even more correlation between losses. As such, performance for this system should be computed with a discount factor γ near 1.0 and a long horizon T to reflect our belief that the current behavior of the system has a strong effect on future losses.

3.3.2 ICP Laser Odometry

Next we discuss the laser rangefinder-based odometry system described in Section 2.2.4. This system estimates the robot motion by registering 2D laser scans of the local environment against previous scans. Unlike the previously discussed image alignment algorithm, this point cloud alignment algorithm performs an

outlier rejection step, which can be thought of as estimating whether objects in the laser scan belong to static or moving objects. Thus, the latent for this system consists not only of the laser rangefinder motion, but also variables describing whether or not local objects are moving, even though an estimate loss, e.g. SSE, would involve only the motion. Like before, the observations are consecutive laser scans.

Though the laser odometry system's latent is slightly more expansive than the floor-facing odometry system's, there are still a number of potentially important contextual factors. In particular, since the laser rangefinders are planar, the system relies on the rangefinders staying parallel with the plane that the robot moves in. Tilting resulting from uneven ground, or the addition of a suspension on the drivetrain may violate this planarity assumption and degrade performance. The system's outlier rejection also relies on the majority of the scans belonging to static objects. Operating in a highly dynamic environment, for instance a busy corridor, will likely violate this assumption as well.

There are a few tunable parameters that may be able to help adapt to these context factors. Specifically, we can set the keyframing thresholds to more frequently update the registration keyframes such that only very recent scans will be used in registration. This may mitigate effects from tilting. We can also compensate for highly dynamic environments by lowering the minimum required inlier ratio such that the system will not reject the correct motion estimate.

The laser rangefinders we use on the IMR operate at a relatively low rate of 10 Hz, but with a moderate range of approximately 5.6 m that ensures a fair amount of overlap between consecutive scans. This system also uses previous motion estimates to initialize registration, like the floor-facing odometry system. As such, this system likely exhibits fairly strong correlations in time as well.

3.3.3 Fiducial-Based Localization System

We depart the world of odometry and consider the IMR's fiducial-based localization system, which can be seen in Figure 2.1. This system uses 20 side-facing cameras to detect illuminated AprilTag Olson [2011] fiducials in the environment. Since the pose of all tags are determined during a pre-deployment mapping step, detected tags can be used to estimate the pose of the robot. The latent for this system is thus simply the pose of the robot, and the observations are the images captured by the cameras.

This system has typical visual contextual factors, such as local lighting conditions, which can be compensated for by tuning the camera exposures. The detection algorithm also has various outlier rejection parameters that can be used to make the localization system more aggressive if odometry information is unavailable, or more conservative when there is high-quality odometry. This consideration can be thought of as another contextual factor.

Unlike the previously discussed odometry systems, the estimates produced by this system are nearly independent from each other. In some cases false detections may continue in multiple consecutive images due to a common source, but this is relatively rare.

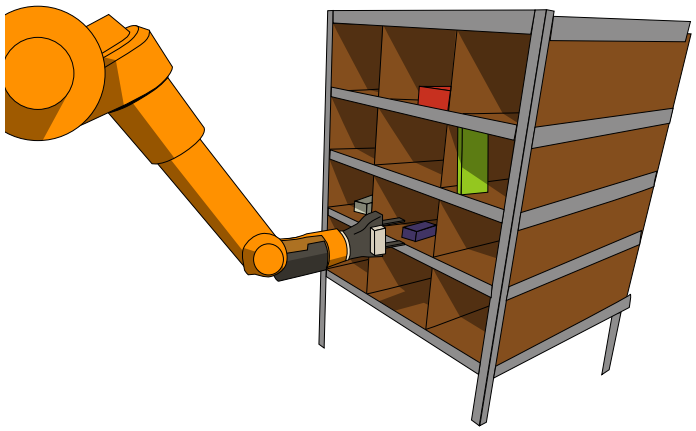


Figure 3.4: An illustrative robot system for bin picking with a wrist-mounted depth camera.

.

3.3.4 Bin Picking System

We now consider an imaginary perception system designed to detect and localize an object in a bin, similar to the Amazon Picking Challenge robots discussed by Eppner et al. [2016] and shown in Figure 3.4. The system consists of a depth camera on a robotic arm that captures depth and RGB images of the item in the bin. The depth image is used to segment out the object from the bin, and the object appearance is used with a feature-based classifier to determine the object type. We assume that the types of possible objects are finite and known beforehand.

The latent of interest for this system is the type of object in the bin, which is a categorical variable. For instance, the object may be a toothbrush, a jar of peanuts, or a mug. The observations are the depth and RGB image captured for a particular bin and item instance. A reasonable loss function for this classification task is the hinge loss. Since this system will likely operate once per bin and item, we do not expect strong correlations between observations, and thus would compute performance over each estimate separately, i.e. with $\gamma = 0$.

A number of factors may form the context for this system. For instance, Eppner et al. [2016] note anecdotally that "the lighting conditions were particularly difficult due to very bright spot lights directly above the competition area: objects in the front of each bin appeared to be effectively white, while objects in the back appeared as nearly black". This unexpected lighting variation resulted in many teams' perceptual systems failing, and is an example of context resulting in degraded performance. To compensate for the lighting context, we may want to tune the camera exposure as well as the position and orientation of the camera relative to the bin when capturing images. We may also wish to consider algorithmic parameters, such as thresholds, for segmenting out the object from the bin and matching object appearances.

Chapter 4

Evaluating Perceptual Performance for Adaptation

To adapt the perception behavior, we must first quantify the performance of the system. Our ideal evaluation scheme would have the following properties:

- **Practical to use at scale:** This means that the method cannot rely on excessive amounts of computation, data, or other resources that might not be available on-site.
- **Compatible with our adaptation mechanisms:** In this work, we explore parameter tuning and failure prediction as adaptation mechanisms. Thus, our evaluation approach should be consistent across different parameter configurations and be structured in a way that we can reasonably define failure.
- **Generalizable to a wide variety of systems:** By this we mean applying the approach to a new system should not require undue engineering effort.

In our survey of prior work on evaluating performance, we find that none of the existing methods satisfy all of these requirements. As such, we develop a new approach for evaluation, building off a statistical understanding of performance, and validate it empirically.

4.1 Prior Work on Measuring Performance

We survey and evaluate here prior work in quantifying the performance of perception systems. These techniques can be classified into three major categories, which we discuss below.

4.1.1 Approaches using Ground Truth

The most commonly accepted method for evaluating performance is to compare system behavior against a trusted source, hence the term "ground truth". As such, adapting using ground-truth-based methods effectively entails altering the system behavior to match reference behavior as closely as possible.

Sources of ground truth vary by application, but can largely be split into human, instrumentation, or by-design. Manual ground truth relies on humans to produce the expected result of an algorithm. Manual labeling is used by Jammalamadaka et al. [2012] to identify human poses in images, by Rogers et al. [2014] to align planar laser scans, and by Civera et al. [2010] to choose image feature correspondences. Narayanan and Likhachev [2016] use a dataset with annotated object poses to study localization performance.

Instrumentation-based ground truth uses precise measurements for supervision, and is typically used in estimation applications. Motion capture systems as used by Rwekmper et al. [2012] and Shen et al. [2016] can provide excellent pose data, but are generally restricted to limited indoor spaces. Other sources have also been used, such as automated survey equipment by Krsi et al. [2015], augmented GPS/INS systems by Geiger et al. [2013], and more recently, detailed urban maps by Jo et al. [2015].

Perhaps the least common, but most practical source of ground truth is that which exists automatically, typically achieved through use of simulation, data augmentation, or experiment design. Handa et al. [2012] use a photorealistic simulation to study a visual odometry algorithm. Montesano et al. [2005] have used laser scans collected at the same location but artificially perturbed to test a scan-matching algorithm. Another common approach in state estimation is to test loop-closure accuracy by traversing a closed loop and measuring the integrated displacement residualClipp et al. [2010].

Ground-truth-based approaches reliance on external information makes them impractical to use on-site at scale, as both human supervision and instrumentation are typically expensive to obtain. It may be possible to rely on carefully-designed experiments with built-in ground-truth, but such an approach may be difficult to generalize to a variety of systems. We note, however, that advances in system modeling may make simulation-based approaches like the one used by Handa et al. [2012] practical for our application in the future.

4.1.2 Approaches using Heuristics

Another common method for evaluating performance is to use *heuristic* quantities that differentiate between nominal and abnormal system performance. As such, using a heuristic to adapt can be thought of as altering system behavior to match indications of nominal behavior.

Typically heuristics are designed to rely only on data available at runtime, in contrast to ground-truth methods that use external information. For instance, a common heuristic in sparse visual odometry is the number of feature correspondences between frames, which is accepted as an indicator of tracking

quality. This heuristic is used by Churchill et al. [2015] and Dequaire et al. [2016] where the authors report the number of correspondences as a substitute for localization performance. Other works by Brunner et al. [2011] and Brunner and Peynot [2014] study visual odometry performance in adverse conditions, focusing on correlations between image quantities, e.g. brightness, sharpness, blur, with the number of feature correspondences. Zhu and Milanfar [2010] use a gradient-based coherence heuristic to quantify image noisiness without ground truth.

Statistical metrics are sometimes used as heuristics in the filtering and modeling communities. For instance, work by Snderhauf and Protzel [2012] reports the chi-squared test as a measure of solution quality or the optimization objective itself, which Grisetti et al. [2012] showed is related to the joint observation likelihood. Similarly, Abbeel et al. [2005] used observation likelihood is used as an optimization objective to select Kalman filter parameters and improve state estimation. Dunik et al. [2012] report the normalized estimate entropy for their unscented Kalman filtering approach, which reflects the state tracking and observation prediction quality.

We note that heuristics are also used in perception to provide supervision to an learning method by using domain information or assumptions. For instance, the self-supervised river segmentation approach detailed by Achar et al. [2011] assumes that the river lays below the horizon to generate training data during operation. This is similar to the work of Hawke et al. [2016] where scale and ground-plane information are used to find hard-negative pedestrian detection examples. Works in grasping by Pinto and Gupta [2016] and Levine et al. [2016] are also relevant, as they use a gripper force sensor to determine whether a grasp is successful or not. However, the heuristics in these works are not evaluating the performance of an existing system so much as operating as small estimation systems unto themselves.

Heuristic approaches are attractive with respect to ease of implementation and deployment, but can be challenging to design, especially for a variety of systems. Statistical heuristics generalize more easily, but do not handle changing quantities of observations well, as we show later in our experiments.

4.1.3 Introspective Approaches

The last class of evaluation methods we discuss here are part of a recent trend in understanding perception *introspectively*, reasoning about the certainty or quality of a estimate as it relates to the input. Though the nature of different introspective approaches are different, we can think of adapting with introspective feedback as changing system behavior to maximize certainty or minimize the chance of failure.

Though a number of methods may declare themselves as "introspective", their philosophies can vary considerably. One approach to introspection is explicitly reasoning about the space of hypotheses to explain observed data, and correspondingly translating this into estimate uncertainty. An example of such an approach is the work of Grimmett et al. [2016], which considers classifica-

tion output uncertainty over distributions of models on training data. Related are studies on filtering optimism, a condition wherein a filter becomes overly confident in its estimate [Bailey et al., 2006].

The other major family of introspection approaches seek to predict what data inputs or conditions result in system failure. Much of this work has been done on vision systems, for instance Zhang et al. [2014] work on predicting segmentation or horizon detection failures, or Daftry et al. [2016] work on navigation estimation failures. Other work by Churchill et al. [2015] and Dequaire et al. [2016] have shown prediction of the heuristic performance of a vision-based navigation system, and by Gurău et al. [2017] on a classification system. All of these approaches rely on applying machine learning techniques to large amounts of examples.

Introspective approaches are promising for adaptation as, like heuristics, they do not rely on external information. However, existing methods are not mature enough to be generally applied to a variety of systems. In addition, it may be challenging to extend learning-based methods over changing parameters due to the increase in data required.

4.2 Monte Carlo Performance Evaluation

Since we are unable to identify a satisfactory existing technique for evaluating performance, we set out now to develop such a technique.

Recall that in Chapter 3.2 we defined in Equation 3.1 the performance of a perception system, expressed as an expectation of a perception loss. We repeat it here for convenience:

$$\rho = E_{x,\hat{x}|\theta}\left[\ell(x,\hat{x})\right] \tag{3.1}$$

Since the distribution $p(x,\hat{x}|\theta)$ is not usually known, we cannot simply compute the expectation. However, it is possible to collect samples from the distribution by executing the perception system. As such, we describe a Monte Carlo evaluation method and provide extensions to reduce variance.

4.2.1 Review of Monte Carlo Approximations

We briefly review Monte Carlo methods for approximating expectations of a random quantity using samples. The basic approximation for a function $f(\cdot)$ of random variable x over distribution $p(x)$ is given by:

$$E_x\left[f(x)\right] \approx Q_N \equiv \frac{1}{N}\sum_{i=1}^{N} f(x_i),\ x_i \sim p(x) \tag{4.1}$$

In words, we can approximate the function simply by drawing samples, computing the function, and averaging the results. Being an average of independent

random quantities, the variance of this estimator is given by:

$$Var\left[Q_N\right] = Var\left[\frac{1}{N}\sum_{i=1}^{N} f(x_i)\right] \tag{4.2}$$

$$= \frac{1}{N^2} Var\left[\sum_{i=}^{N} f(x_i)\right] \tag{4.3}$$

$$= \frac{1}{N} Var\left[f(x)\right] \tag{4.4}$$

As such, we see that the variance in the Monte Carlo estimate Q_N decreases rapidly with an increasing number of samples N. In practice, since the variance of $f(x)$ is not known (or else we would likely not be using Monte Carlo), we can estimate it empirically:

$$Var\left[f(x)x\right] \approx V_N = \frac{1}{N-1}\sum_{i=1}^{N}(f(x_i) - Q_N) \tag{4.5}$$

$$Var\left[Q_N\right] \approx \frac{V_N}{N} \tag{4.6}$$

In practice, the variance of the Monte Carlo estimate can be quite large. Fortunately a number of variance reduction approaches exist. The most common of these approaches is *importance sampling*, in which samples are drawn from a proposal distribution $q(x)$ instead of the target distribution $p(x)$ and the samples are weighted to compensate:

$$Q_N^q \equiv \frac{1}{N}\sum_{i=1}^{N}\frac{p(x_i)}{q(x_i)} f(x_i) \tag{4.7}$$

Importance sampling and its variants are quite effective in practice, but cannot be used when the probability density $p(x)$ is not known, as is the case here. Thus, we must find other methods of reducing evaluation variance.

4.2.2 Application to Performance Evaluation

Returning to our performance estimation task, we can use a Monte Carlo approximation to the expectation in Equation 3.2 by sampling from the distribution $p(x, \hat{x}|\theta)$:

$$\rho = E_{x,\hat{x}|\theta}\left[\ell(x, \hat{x})\right] \tag{4.8}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\ell(x_i, \hat{x}_i) \tag{4.9}$$

$$x_i, \hat{x}_i \sim p(x, \hat{x}|\theta)$$

This corresponds to executing the system multiple times with the latent x_i revealed by ground truth. Recall that the effects of observation and context variations are marginalized out in our definition of performance, and as such, should

be sampled over rather than held constant. Most ground-truth-based methods rely on this approximation in one way or another, while heuristic methods use a similar approximation to estimated expected heuristics. In either case, having more samples reduces the approximation variance.

4.2.3 Reducing Variance with a Capture-Processing Decomposition

Previously we noted that the expected variance in a Monte Carlo estimate decreases inversely with the number of samples used. In practice it may be difficult to execute the physical system a large number of times. However, not all of the system stochasticity may be from data collection, but from algorithm randomness instead. Thus, if we can decompose the expectation in Equation 4.9 into a component that requires physical executions and one which does not, we may be able to use more samples. Intuitively, physical executions are required when data from the real world must be captured, but not when already captured data must be processed. As such, we refer to such a decomposition as a capture-processing decomposition, as it splits the system into a data capture half which involves physical executions, and a data processing half which does not.

Representing this decomposition analytically is straightforward since we consider the collected data z separately from the generated estimate $\hat{x}$. Applied to Equation 4.9 we have:

$$\begin{aligned}
\rho &= E_{x,\hat{x}|\theta}\left[\ell(x,\hat{x})\right] \\
&= E_{z|\theta}\left[E_{x,\hat{x}|z,\theta}\left[\ell(x,\hat{x})\right]\right] \\
&= E_{z|\theta}\left[E_{x|z,\theta}\left[E_{\hat{x}|z,\theta}\left[\ell(x,\hat{x})\right]\right]\right] && (4.10)\\
&= E_{x,z,|\theta}\left[E_{\hat{x}|z,\theta}\left[\ell(x,\hat{x})\right]\right] && (4.11)\\
&\approx \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\rho(x_i,\hat{x}_{i,j}) && (4.12)
\end{aligned}$$

$$\begin{aligned}
x_i, z_i &\sim p(x_i, z_i|\theta) \\
\hat{x}_{i,j} &\sim p(\hat{x}_{i,j}|z_i,\theta)
\end{aligned}$$

where we have used the conditional independence relation $\hat{x} \perp\!\!\!\perp x|z,\theta$ to simplify $p(x,\hat{x}|z,\theta) = p(x|z,\theta)\,p(\hat{x}|z,\theta)$. Since in practice it is much easier to sample from the estimate distribution $p(\hat{x}|z,\theta)$ than $p(x,z|\theta)$, we can have $M \gg N$. This reduces the variance from stochasticity in the estimate.

4.3 Approximate Posterior Evaluation

The methods described in Section 4.2 meet two of our three stated goals for an evaluation method, but their reliance on ground truth conflicts with the remaining goal of practicality at scale. To overcome this, we propose a technique

to remove the need for ground-truth while retaining much of the Monte Carlo methodology.

4.3.1 The Approximate Posterior Estimate

In Section 4.2.3 we decomposed the performance expectation into three nested expectations in Equation 4.10, and then consolidated them into capture and processing components. We consider now the alternative ordering:

$$\rho = E_{z|\theta}\left[E_{\hat{x}|z,\theta}\left[E_{x|z,\theta}\left[\ell(x,\hat{x})\right]\right]\right] \tag{4.13}$$

The innermost expectation is over $p(x|z,\theta)$, which is known in Bayesian estimation literature as the *posterior distribution*, and is tracked or produced as an artifact by Bayesian estimators. Thus, given the posterior, we can evaluate the inner expectation analytically, removing the need for ground truth, while still relying on samples for the outer expectation over observations and estimates. This can be understood as Rao-Blackwellization, which has been applied to particle filtering wherein some variables can be analytically updated while others use sampling Doucet et al. [2000]. Fully expanded, our estimator is given by:

$$\rho \approx \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} E_{x|z_i,\theta}\left[\ell(x,\hat{x}_{i,j})\right] \tag{4.14}$$

$$z_i \sim p(z_i|\theta)$$
$$\hat{x}_{i,j} \sim p(\hat{x}_{i,j}|z_i,\theta)$$

Since it relies on the posterior distribution, we refer to this as the *approximate posterior estimate* of performance, abbreviated as APE. Even if a Bayesian estimator is not used, it is straightforward to derive the posterior for a single observation from a forward observation model $p(z|x,\theta)$ using Bayes' Rule:

$$p(x|z,\theta) = \frac{p(z|x,\theta)p(x|\theta)}{p(z|\theta)} \tag{4.15}$$

$$\propto p(z|x,\theta)p(x) \tag{4.16}$$

where $p(x|\theta) = p(x)$ from conditional independence.

Intuitively, the APE equates estimate quality to estimate certainty. In other words, instead of looking at whether a system behaves similarly to a ground-truth reference, the APE judges a system based on how confident its estimates are, and thus relies on having accurate statistical models.

4.3.2 Application to Squared Error Losses

We derive the APE of the common sum squared error (SSE) and its related losses, the mean squared error (MSE) and root mean squared error (RMS),

given below:

$$\ell_{SSE}(x, \hat{x}) = (x - \hat{x})^T (x - \hat{x}) \tag{4.17}$$

$$\ell_{MSE}(x, \hat{x}) = \frac{(x - \hat{x})^T (x - \hat{x})}{\dim (x - \hat{x})} = \frac{\ell_{SSE}(x, \hat{x})}{\dim (x - \hat{x})} \tag{4.18}$$

$$\ell_{RMS}(x, \hat{x}) = \sqrt{\frac{(x - \hat{x})^T (x - \hat{x})}{\dim (x - \hat{x})}} = \sqrt{\ell_{MSE}(x, \hat{x})} \tag{4.19}$$

where $\dim y$ denotes the dimensionality of vector y. Since these three losses are effectively equivalent, we perform our analysis with the SSE for simplicity. Substituting the expression from Equation 4.17 into Equation 4.14, we get:

$$\rho \approx \frac{1}{N} \sum_{i=1}^{N} E_{x|z_i,c} \left[(x - \hat{x})^T (x - \hat{x}) \right] \tag{4.20}$$

$$= \frac{1}{N} \sum_{i=1}^{N} E_{x|z_i,\theta} \left[\operatorname{tr} \left((x - \hat{x})(x - \hat{x})^T \right) \right] \tag{4.21}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \operatorname{tr} \left(E_{x|z_i,\theta} \left[(x - \hat{x})(x - \hat{x})^T \right] \right) \tag{4.22}$$

where $\operatorname{tr} A$ is the trace of matrix A, equal to the sum along the diagonal of A. If we assume that the estimate is the mean of the posterior $\hat{x} = E_{x|z,c}[x]$, then Equation 4.22 further simplifies to:

$$\rho \approx \frac{1}{N} \sum_{i=1}^{N} \operatorname{tr} \left(E_{x|z_i,\theta} \left[(x - E_{x|z_i,\theta}[x])(x - E_{x|z_i,\theta}[x])^T \right] \right) \tag{4.23}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \operatorname{tr} \left(\operatorname*{cov}_{x|z_i,\theta} (x) \right) \tag{4.24}$$

which is simply the trace of the covariance of the posterior, averaged over multiple trials. This makes computing the APE for SSE loss on posteriors with closed-form expressions for their covariance straightforward. If the estimate is not selected as the mean, or $\hat{x} \neq E_{x|z,\theta}[x]$, as may occur for multi-modal posteriors, then we can instead evaluate the expectation in Equation 4.22 with another Monte Carlo approximation.

Extending these results for the MSE loss is trivial, as it is simply a scaled version of the SSE. Extending to the RMS loss, however, is quite involved, as it requires the $\frac{1}{2}$-th moment of the posterior. Further, since the transformation from MSE to RMS is non-linear, a configuration that minimizes MSE may not be the minimizer for the equivalent RMS. However, RMS, MSE, and SSE are used almost interchangably throughout the literature, and as such, we use the SSE in our demonstrations for simplicity.

4.3.3 Tracking Posteriors with Adaptive Kalman Filtering

As discussed in Section 4.3.1, the APE relies on having an accurate posterior distribution $p(x|z,\theta)$. Much of the work to-date in perception has focused on generating accurate mean predictions as opposed to accurate distributions, and as such, there are relatively few methods for calibrating statistical models. A regression-type approach was previously explored by Hu and Kantor [2015] and Haarnoja et al. [2016]. More classically, the EM approach proposed by Ghahramani and Roweis [1998] could also be used to learn system models from data. We detail here the Adaptive Kalman Filter, a flexible and powerful non-parametric technique.

In a standard Kalman filter the estimate covariance does not involve the observation themselves, evolving purely as a function of the system model, *i.e.*, the transition and observation models. Intuitively, we expect the error or noise magnitude to vary in time due to changing contexts or configurations. We can capture this intuition with the adaptive Kalman filter: Let $x_t^{(-)}$ and $x_t^{(+)}$ denote the estimate mean before and after performing an update, respectively, with a similar notation for the estimate covariance $P_t^{(-)}$ and $P_t^{(+)}$. Further define the transition function Jacobian as F_t and the observation function Jacobian as H_t, with the observation at time t written as y_t. Using the approach detailed in Mohamed and Schwarz [1999], the online estimates of the transition covariance Q_t and observation covariance R_t are:

$$Q_{t+1} = \frac{1}{W_Q} \sum_{\tau=t-W_Q+1}^{t} \Delta x_\tau \Delta x_\tau^T \tag{4.25}$$

$$R_{t+1} = \frac{1}{W_R} \sum_{\tau=t-W_R+1}^{t} \nu_\tau \nu_\tau^T + H_t P_t^+ H_t^T \tag{4.26}$$

where W_Q and W_R are sliding window lengths, $\Delta x_t = x_t^{(+)} - x_t^{(-)})$ is the *state correction*, and $\nu_t = y_t - \hat{y}_t^{(+)}$ is the post-update measurement prediction error, often referred to as the *residual*. Intuitively, Eqs. 4.25 and 4.26 adjust the covariances to match the observation prediction errors during execution: When the state evolves predictably, the state corrections will be small, resulting in a small Q. Similarly, when the observations are well-predicted, the residuals will be small, resulting in a small R.

We note that the AKF estimates rely on an assumption of uncorrelated transition and observation noise to be meaningful. When the noise is systematic or heavily correlated in time, *e.g.*, calibration errors or software bugs, the estimated covariances may not be accurate, resulting in poor introspection.

4.4 Experimental Validation

We validate our proposed methods for performance evaluation with experiments on our demonstrative state estimation application. Our primary goal is to investigate the fidelity of the APE for use in adaptation to context, and additionally are interested in what situations the APE does and does not perform well in, as well as the importance of using multiple samples.

Our overall approach is to directly compare evaluation approaches by executing our test systems many times over a variety of configurations and environments. Given this population of evaluations, we can quantify the overall quality of an evaluation approach compared to a standard ground-truth-based approach. Further, with multiple executions on the same configuration and environment, we can also test the effect of using multiple samples for evaluation. It is difficult to gain a deeper understanding of the APE with this purely empirical approach, however, so we also use executions of a simulated system where we can control sensing phenomena.

Below we describe our experimental procedure, present results from each of our tests, and conclude with a discussion and analysis of the results overall.

4.4.1 Evaluation Methods Tested

We test three canonical evaluation approaches in addition to our proposed APE method, described below:

Sum of Squares Error (SSE)

The conventional baseline that uses ground truth to compute Eq. 4.17 for the body velocity error. Low SSE corresponds to better performance. We compute the SSE for two-dimensional body velocities $(\dot{x}, \dot{y}, \omega)$ and do not weight the components in the SSE summation.

Sum Observation Log-likelihood (SOL)

The SOL is a statistical heuristic computed as the sum of log-likelihoods for all observations received in the trajectory. This is equivalent to computing the joint probability of all received observations assuming they are independent. We expect that high SOL corresponds to better performance.

Average Observation Log-likelihood (AOL)

One weakness of the SOL is that it can vary dramatically depending on trajectory length. To compensate for this, we can normalize the SOL by the number of observations received and compute the AOL, which is equivalent to the average observation probability. We expect that high AOL corresponds to better performance.

Approximate Posterior Error (APE)

Our proposed approach, computed as described in Eq. 4.23 as the trace of the state estimate covariance. Like the SSE, the APE is computed over a trajectory with numerical integration. Low APE corresponds to better performance. To correspond to the unweighted SSE on two-dimensional velocities, we compute the APE with the covariance over the two-dimensional body velocities only.

4.4.2 Test Metrics

Given a population of evaluation comparisons, we can quantify the quality of an evaluation method against ground truth performance. We do not have ground truth performance, per se, but can estimate it using multiple evaluation samples as in Equation 4.12. Once ground truth is computed, we then report the following two test metrics for each evaluation method.

It is tempting to assert that a good evaluation approach should be highly correlated with ground truth in the traditional sense, but this is a much stricter requirement than what is needed for adaptation. Instead, we propose that a good evaluation approach need only rank configurations similarly to ground-truth. This property can be captured with a rank correlation, such as Spearman's ρ or Kendall's τ, which reflect how consistently two quantities rank items together.

In our experiments, we are interested in the rank correlation between each evaluation method and the ground truth rankings, as well as the variance in the rank correlation resulting from evaluation variance. In previous work, Newson [2002] suggested that the Kendall-τ has better confidence intervals than Spearman's ρ. Gilpin [1993] also favors Kendall-τ, for its faster convergence. Nonetheless, we report both on a limited set of our results for comparison, and detail both measures below, as they are not commonly used in the robotics community.

Kendall-τ

The Kendall-τ coefficient can be understood as the normalized number of bubble-sort insertions required to transform one ordering into the other, or alternatively as related to the probability of a ranking being in agreement. As such, making multiple small ordering mistakes produces higher τ values than a single severe ordering mistake. In these results, we use the τ-b variant implementation in SciPy[1] which normalizes to the number of data and accounts for ties.

Consider a set of joint observations $\{(x_i, y_i)\}$. In our setting, x_i is the trial mean SSE and y_i is a performance evaluation. A pair of observations (x_i, y_i) and (x_j, y_j) with $i \neq j$ is *concordant* if the ordering of observations match, *i.e.*, $x_i > x_j$ and $y_i > y_j$, and *discordant* if the ordering of observations is reversed,

[1] https://www.scipy.org/

i.e., $x_i > x_j$ and $y_i < y_j$. The τ-b coefficient is computed as:

$$\tau = \frac{n_C - n_D}{\sqrt{(n_C + n_D + T_x)(n_C + n_D + T_y)}} \tag{4.27}$$

where n_C is the number of concordant pairs, n_D is the number of condordant pairs, T_x is the number of ties in the x_i observations, and T_y is the number of ties in the y_i observations. A $\tau = -1$ corresponds to exactly opposite rankings and a $\tau = 1$ corresponds to exactly identical rankings. Thus, in our analysis, approaches that produce τ values near 1 can be understood as accurately replicating the ground-truth ordering across all trials.

The Kendall's τ coefficient allows us to measure the accuracy of a single ordering. Accordingly, we can measure the effect of stochasticity in the executions and the number of executions N per trial on ordering stability. Ideally we would collect a large number of independent datasets, but this is rather impractical, so we instead bootstrap to generate synthetic datasets by sampling executions from each trial.

Spearman-ρ

The Spearman-ρ coefficient is the application of "standard" correlation to the ranks generated by two sets of values. In other words, it is the direct measure of how well two assigned rankings correlate with each other. Like the Kendall-τ, this allows us to measure the ordinal correlation instead of linear correlation between two quantities.

Consider again a set of joint observations $\{(x_i, y_i)\}$. Let k_i be an integer rank for x_i within the population, and l_i be an integer rank for y_i within the population. The Spearman-ρ coefficient is computed as:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{4.28}$$

where $d_i = k_i - l_i$ is the difference in ranks for element i and n is the number of observations. Like with the Kendall-τ, a value of 1.0 means an exactly identical ranking, while -1.0 means an exactly reversed ranking.

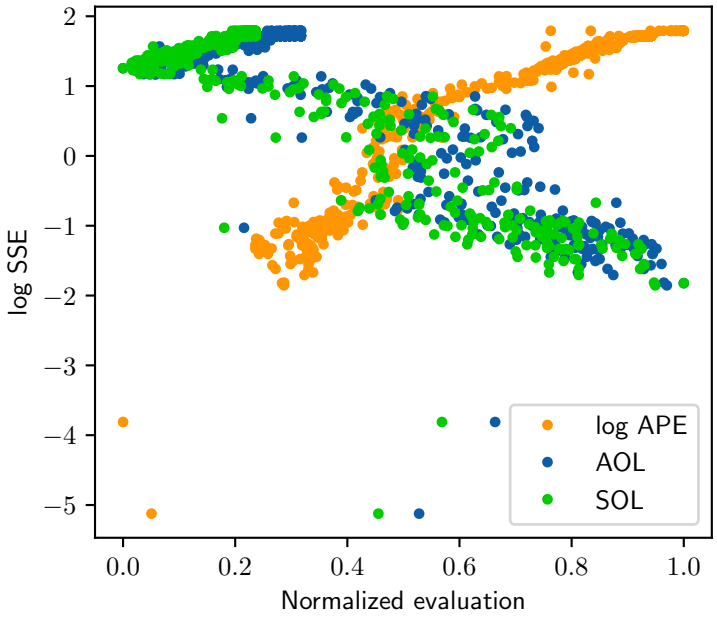
4.4.3 KITTI Experiments

We test the ICP-based laser odometry (ICP-LO) and sparse stereo (SS-VO) odometry systems in two KITTI environments, for a total of four different test conditions. We test 100 uniformly-randomly selected configurations 10 times each in each of the test conditions by executing the system on each recorded trajectory.

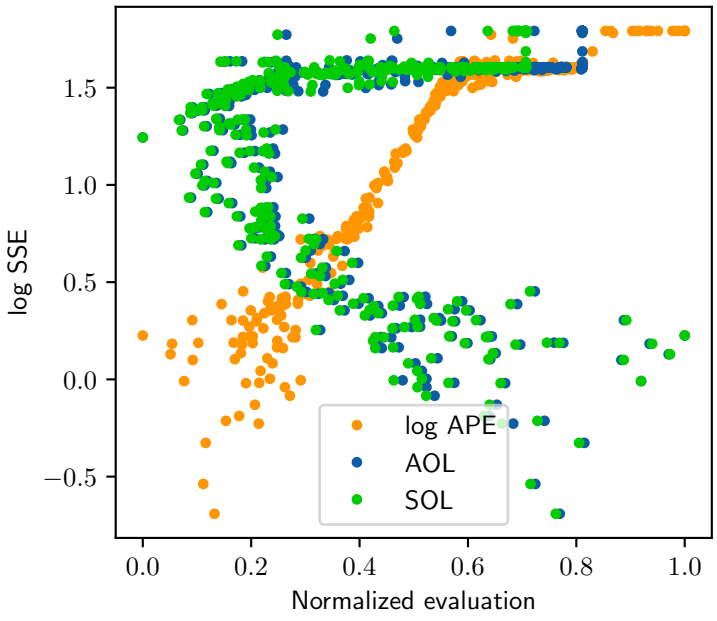
We begin each system execution by resetting the perception system state, setting the configuration, and then initializing the velocity filter mean to the true velocity with a high covariance. This is needed for initialization-dependent methods, such as ICP-LO, since some dataset trials begin with the vehicle at

high speeds ($\approx$15 m/s). This initialization also helps the AKF covariance estimators converge more quickly on startup.

Visualizations of the relation between the ground truth SSE and various evaluated methods, which we dub "parity", are given in Figure 4.1. Correlation metrics computed from 1000 bootstrap instances for each of the environments and systems are shown in Figure 4.2. We report both the Kendall-τ and Spearman-ρ correlations for the aggregated environments and systems in Figures 4.2a and 4.2b, respectively, to show the similarity of the two metrics, and report only the Kendall-τ for the individual conditions. Note that since the AOL and SOL are negatively correlated with the ground-truth SSE, we report the absolute value of the correlation in our results.

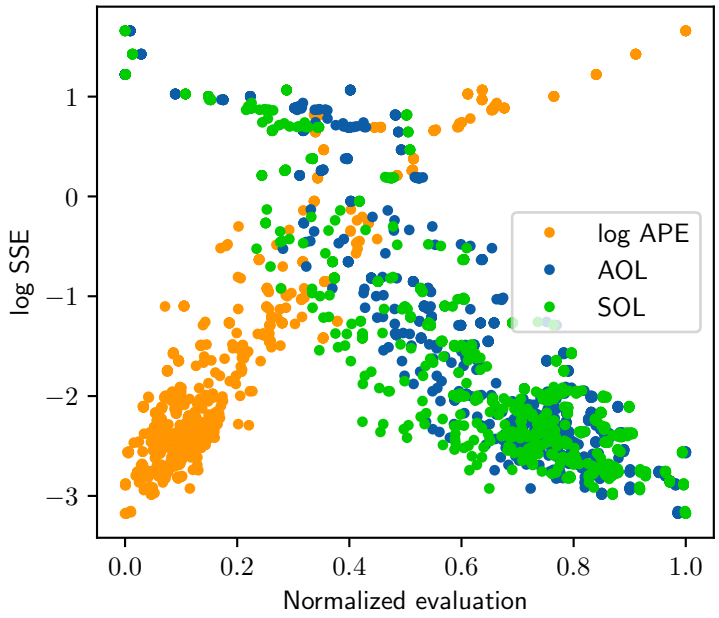


(a) ICP-LO city

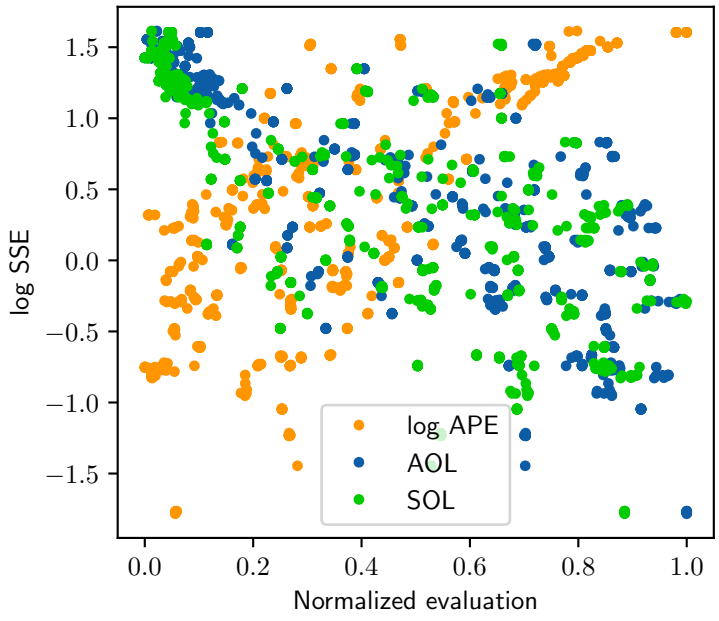


(b) ICP-LO road

Figure 4.1: Parity plots of the ground-truth-free methods against SSE on the KITTI system. The ground-truth-free evaluations are normalized to $[0, 1.0]$ on the x-axis for comparison. Each point represents the mean of 10 repeated samples.

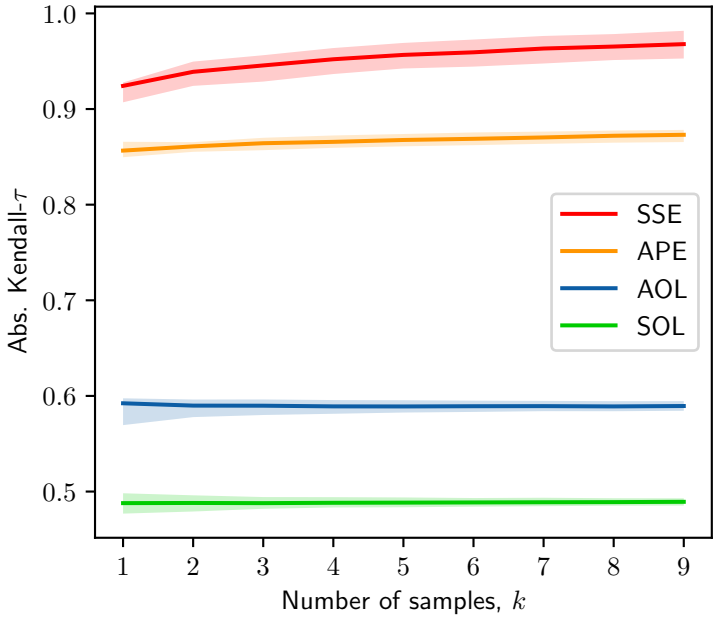


(c) SS-VO city

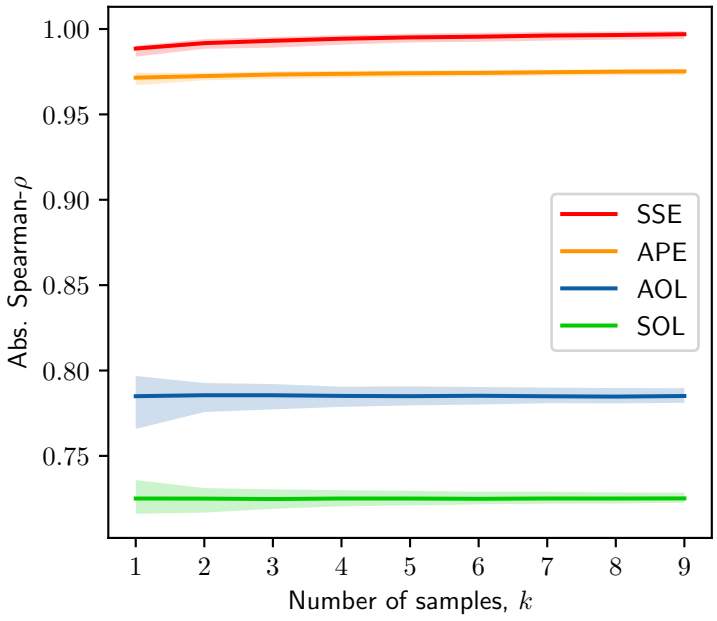


(d) SS-VO road

Figure 4.1: Parity plots of the ground-truth-free methods against SSE on the KITTI system (continued).

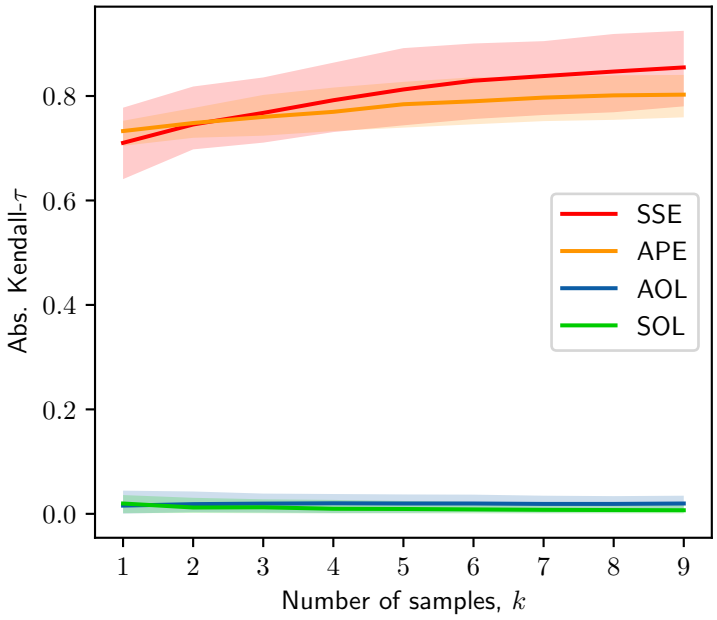


(a) Overall KITTI Kendall-τ

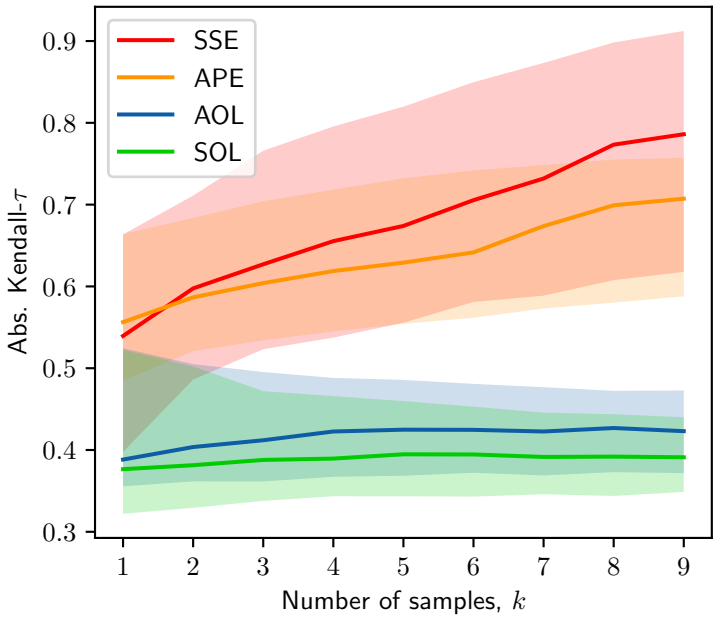


(b) Overall KITTI Spearman-ρ

Figure 4.2: Correlation measures for each evaluation method using varying number of samples on the KITTI system. The line indicates the median correlation over 1000 bootstrap instances, and the shaded region indicates the 5th and 95th percentiles.

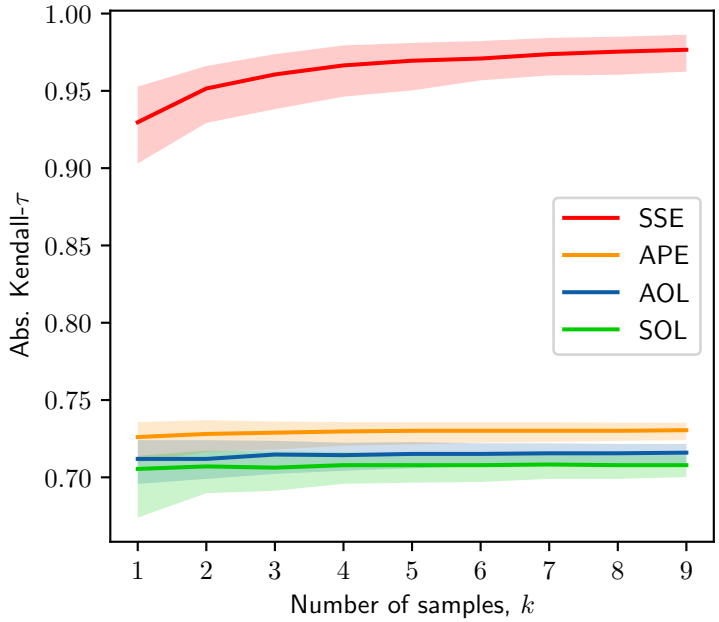


(c) ICP-LO city Kendall-τ

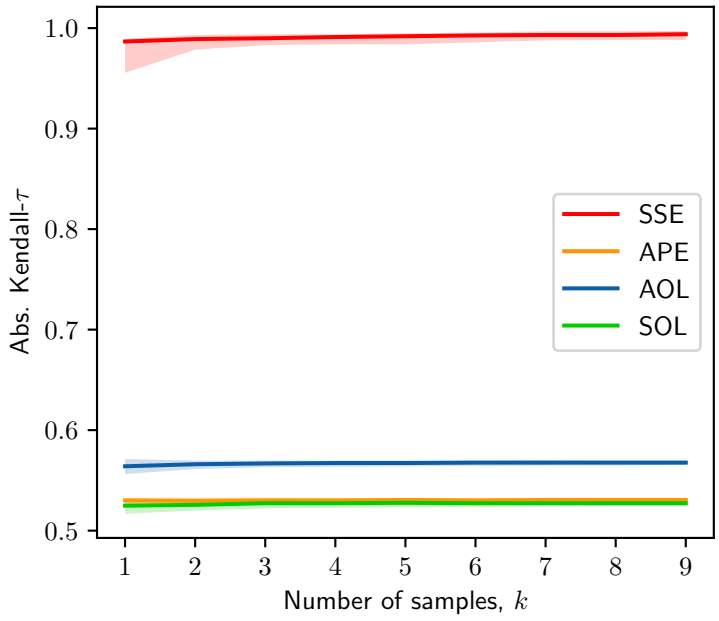


(d) ICP-LO road Kendall-τ

Figure 4.2: Correlation measures for each evaluation method using varying number of samples on the KITTI system (continued).



(e) SS-VO city Kendall-τ



(f) SS-VO road Kendall-τ

Figure 4.2: Correlation measures for each evaluation method using varying number of samples on the KITTI system (continued).

4.4.4 IMR Hardware Experiments

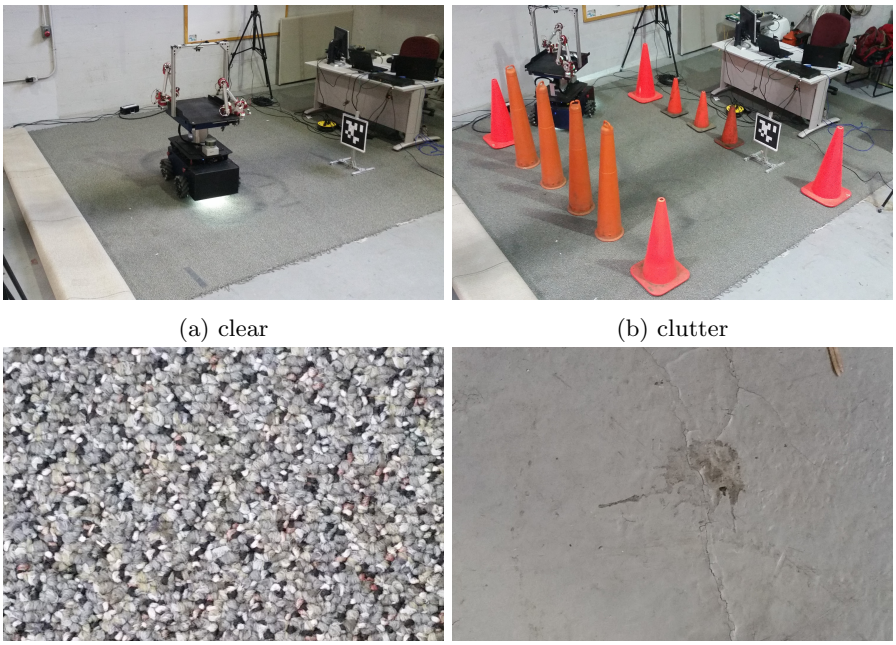
We test the ICP-based laser odometry (ICP-LO) and sparse planar monocular (SPM-VO) odometry systems in two environments each, for a total of four different test conditions. The conditions are shown in Fig. 4.3 and detailed below.

- **clear:** LO in a cleared indoor area
- **clutter:** LO in an area cluttered with traffic cones
- **carpet:** VO on low-pile speckled carpeting
- **concrete:** VO on medium-gloss painted concrete floor

Like in the KITTI experiments, we executed the system 10 times for 100 uniformly-randomly sampled configurations in each test condition. Each system execution is performed with the following procedure:

1. The perception parameters are set to their test values. Trial data recording begins.
2. The robot remains stationary for one second to initialize the AKF.
3. The robot drives forward at 0.5 m/s for 1.0 m.
4. The robot turns in place at 1.0 rad/s for a half-rotation (π radians).

This motion tests the odometry system performance for both translational and rotational motion and is illustrated in Fig. 4.4. The robot resets its pose every two executions using a side-facing camera and a fiducial in the test area to avoid drifting over time.



(a) clear (b) clutter

(c) carpet (d) concrete

Figure 4.3: The test conditions for the APE validation experiments on the IMR.

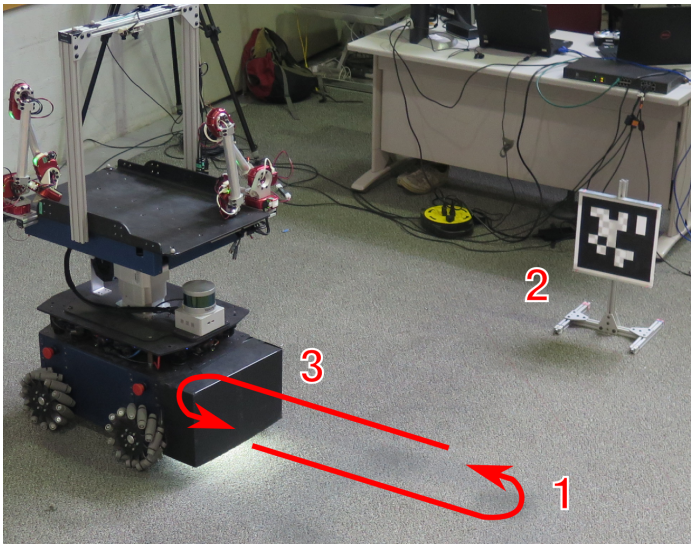
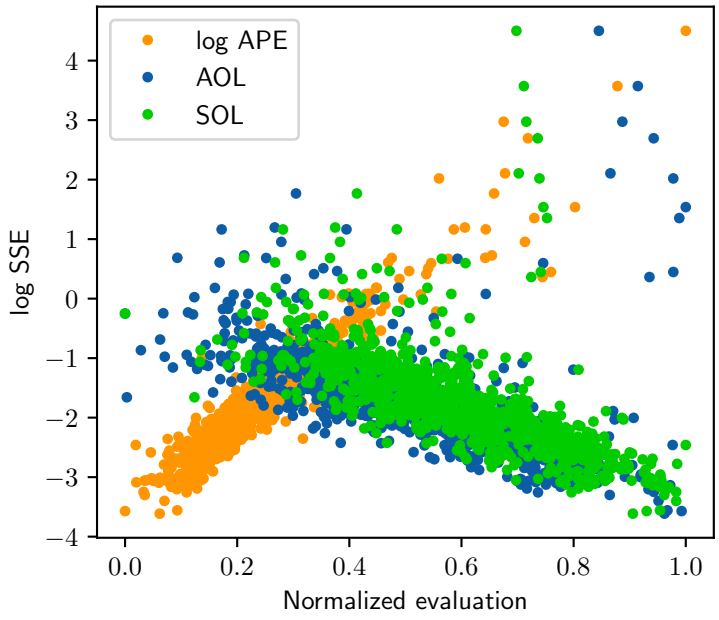
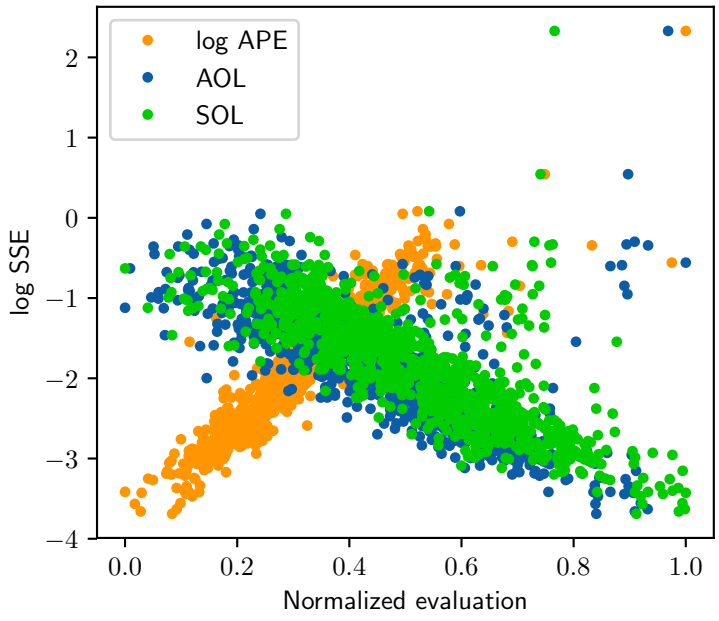


Figure 4.4: An illustration of the system execution procedure. 1.) The robot drives an open-loop trajectory consisting of 0.5 meters straight forward, followed by a 180° turn. 2.) The robot resets its pose using a side-facing camera to observe a fiducial. 3.) The robot can then execute its next open-loop trajectory.

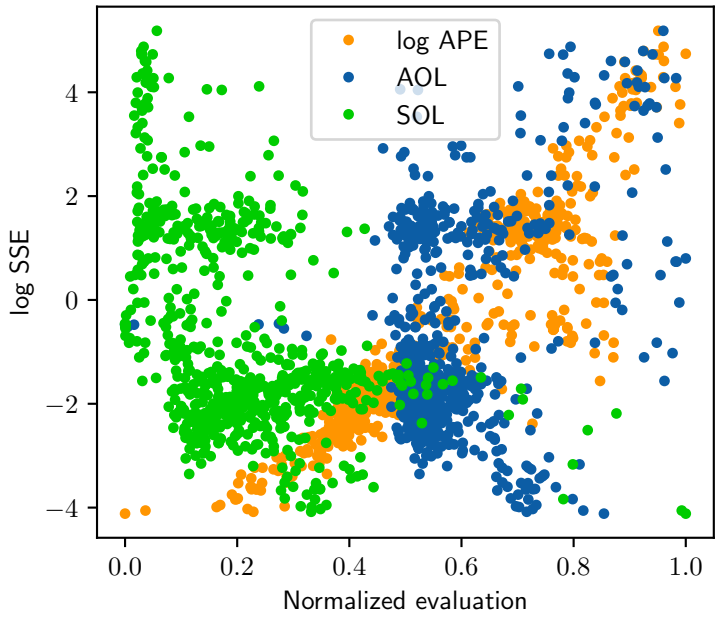


(a) ICP-LO clear

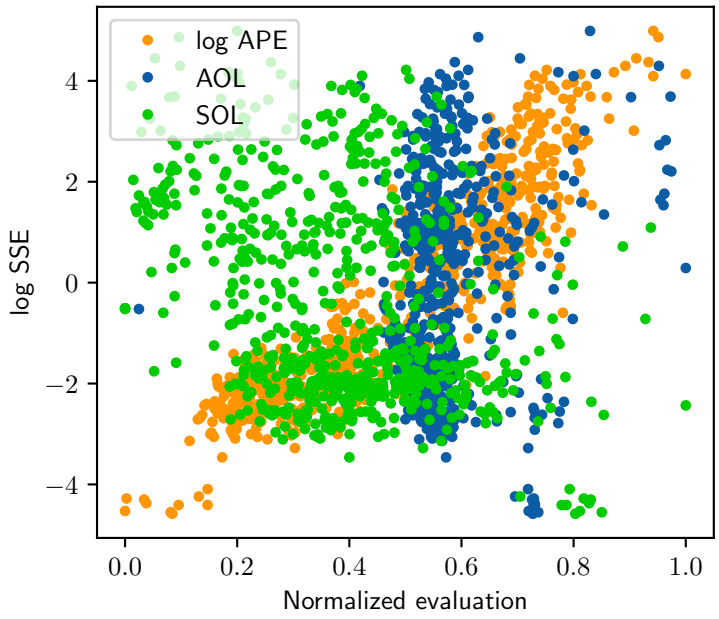


(b) ICP-LO clutter

Figure 4.5: Parity plots of the ground-truth-free methods against SSE on the IMR system. The ground-truth-free evaluations are normalized to $[0, 1.0]$ on the x-axis for comparison. Each point represents the mean of 10 repeated samples.

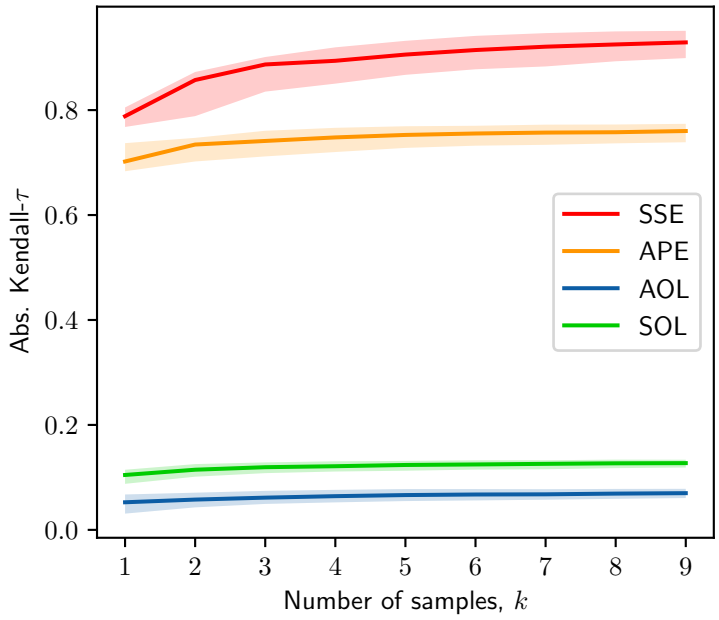


(c) SPM-VO carpet

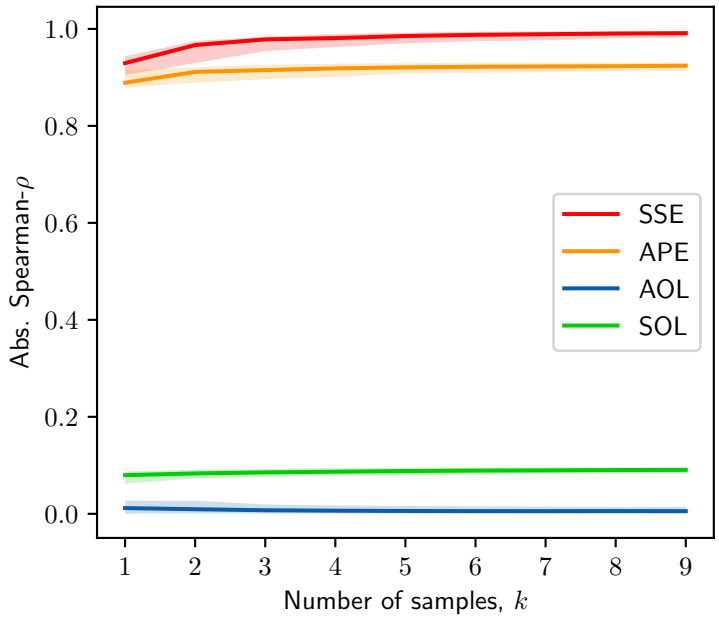


(d) SPM-VO concrete

Figure 4.5: Parity plots of the ground-truth-free methods against SSE on the IMR system (continued).

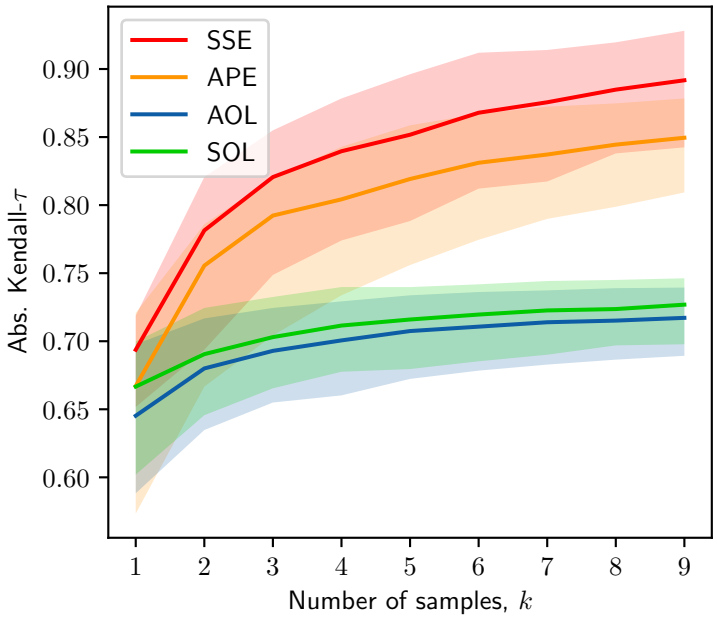


(a) Overall IMR Kendall-τ

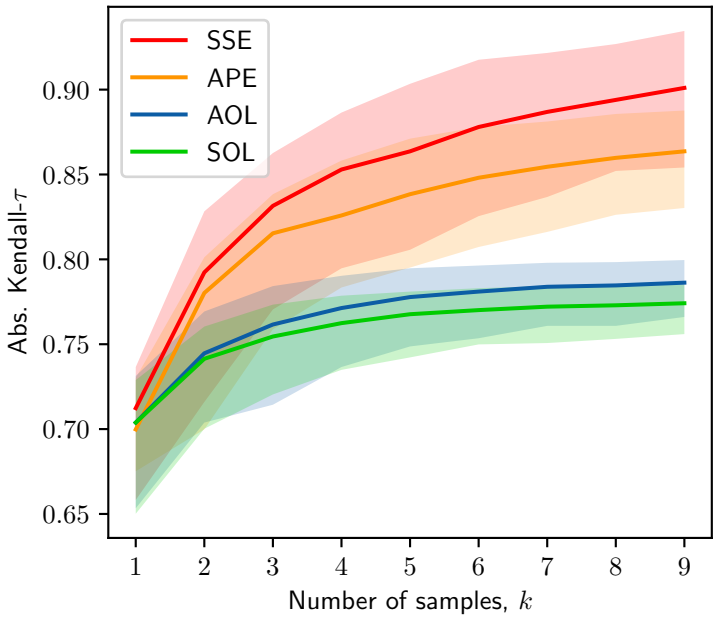


(b) Overall IMR Spearman-ρ

Figure 4.6: Correlation measures for each evaluation method using varying number of samples on the IMR system. The line indicates the median correlation over 1000 bootstrap instances, and the shaded region indicates the 5th and 95th percentiles.

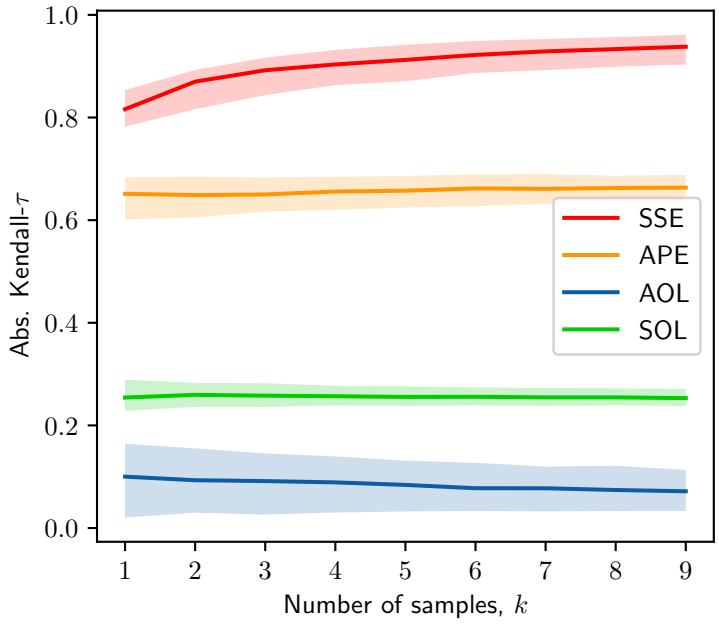


(c) ICP-LO clear Kendall-τ

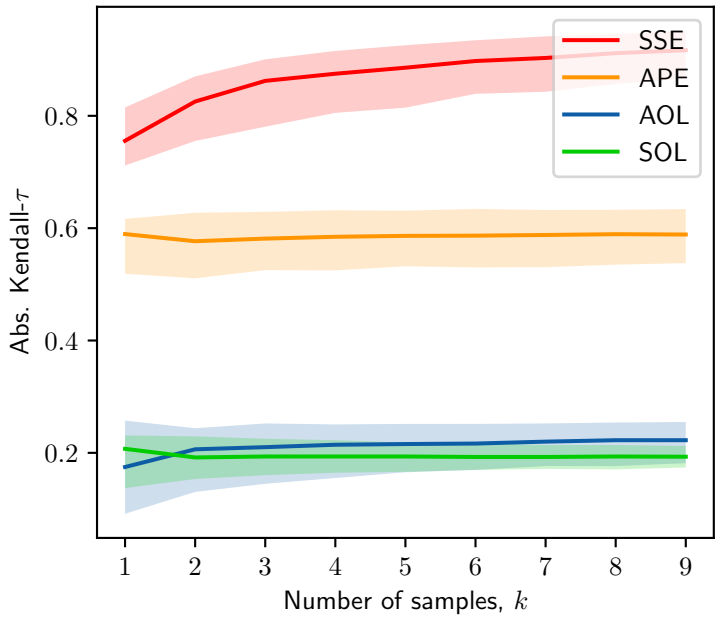


(d) ICP-LO clutter Kendall-τ

Figure 4.6: Correlation measures for each evaluation method using varying number of samples on the IMR system (continued).



(e) SPM-VO carpet Kendall-τ



(f) SPM-VO concrete Kendall-τ

Figure 4.6: Correlation measures for each evaluation method using varying number of samples on the IMR system (continued).

4.4.5 IMR Simulated Experiments

We supplement our IMR hardware experiments with experiments in simulation designed to test the robustness of different evaluation approaches to different sensing phenomena. The simulation itself is simple and emulates the IMR as a point mass with second-order dynamics operating in a 2D environment. The simulated robot receives observations of its body velocity corrupted with zero-mean Gaussian noise, and runs the same core state estimation system as the physical test systems.

We also run the same experimental procedure as on the IMR hardware system, but with three test conditions corresponding to different sensing phenomena which manifest as varying observation rates and noise magnitude. We expect that both suboptimal configurations and higher robot speeds affect sensing performance and quantify this intuition with a "hardness factor" κ:

$$\kappa = \|v\|_2 \cdot \|\theta\|_2 \tag{4.29}$$

where v is the robot body speed and we have arbitrarily set the optimal configuration $\theta^* = 0$. This heuristic reflects that deviating from the optimal parameters increases the noise effects of higher speeds, and vice-versa. The hardness factor is then used in the following three sensing test conditions:

Dependent Noise (DN)

This model captures perception difficulty as sensor noise covariance R that increases exponentially with hardness:

$$R = R_0 + \tilde{R}\,[1 - \exp(k_R \cdot \kappa)] \tag{4.30}$$

where $k_R > 0$ so that R achieves its minimum value R_0 at $\kappa = 0$ and increases with increasing hardness. The sensor rate f is fixed at 200 Hz, and we use constants $R_0 = 1E-6I$, $\tilde{R} = 0.5I$, $k_R = 0.75$.

Dependent Noise and Rate (DNR)

This model builds upon the DN model by additionally having the sensor rate exponentially decrease with hardness:

$$f = f_0 + \tilde{f}\exp(k_f \cdot \kappa) \tag{4.31}$$

where $k_f < 0$ so that f achieves its maximum value $f_0 + \tilde{f}$ at $\kappa = 0$ and decreases with increasing hardness to a minimum of f_0. We use the same constants as the Dn model with additionally $f_0 = 20$, $\tilde{f} = 180$, and $k_f = -1.0$, corresponding to a maximum rate of 200Hz.

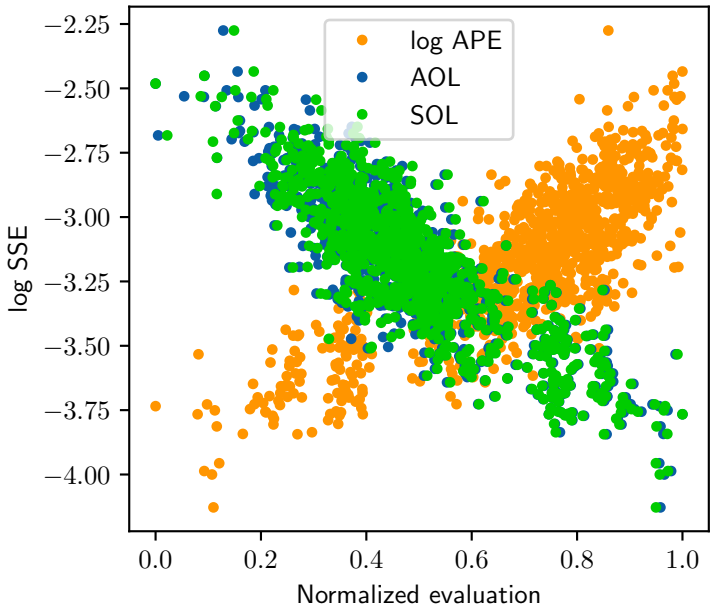
Dependent Noise and Rate with Cutout (DNR+C)

This model modifies the DNR model by modeling a phenomena where the sensor will fail or "cut out" above a certain hardness κ_{max} and not produce any

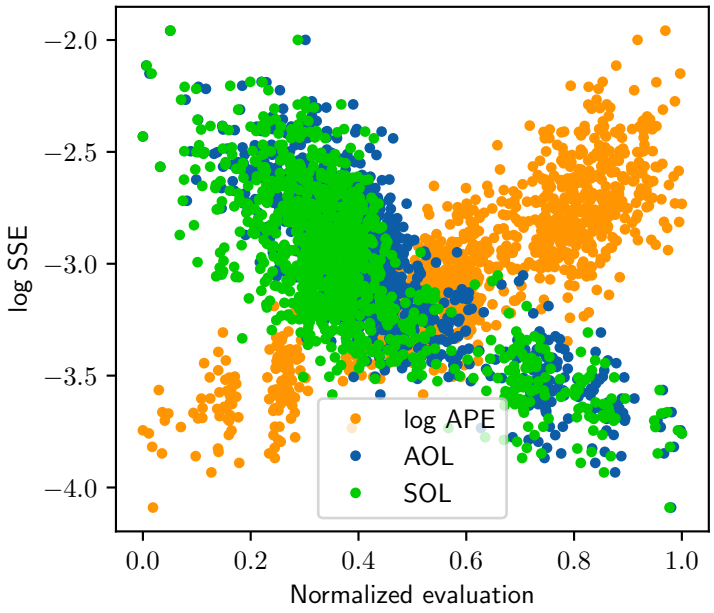
observations:

$$f = \begin{cases} 0, & \kappa > \kappa_{max} \\ f_0 + \tilde{f} \exp(k_f \cdot \kappa), & o/w \end{cases} \tag{4.32}$$

We use the same constants as the DNR model with additionally $\kappa_{max} = 0.8$.

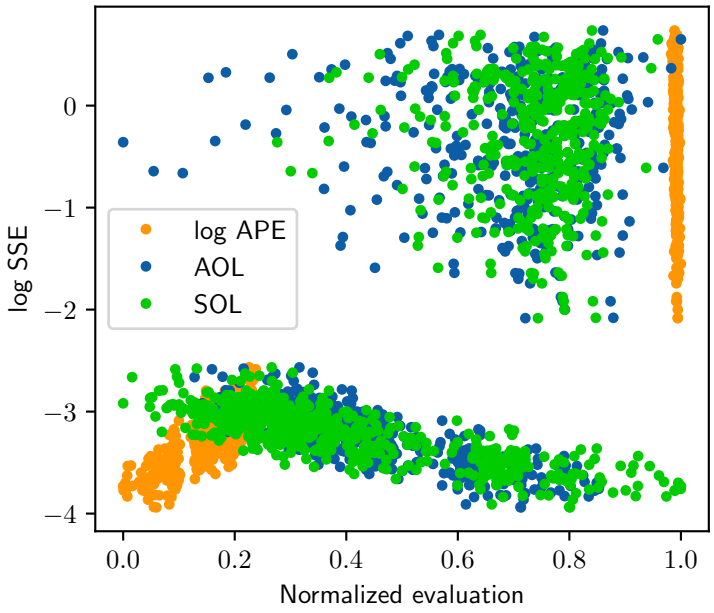


(a) Simulated DN



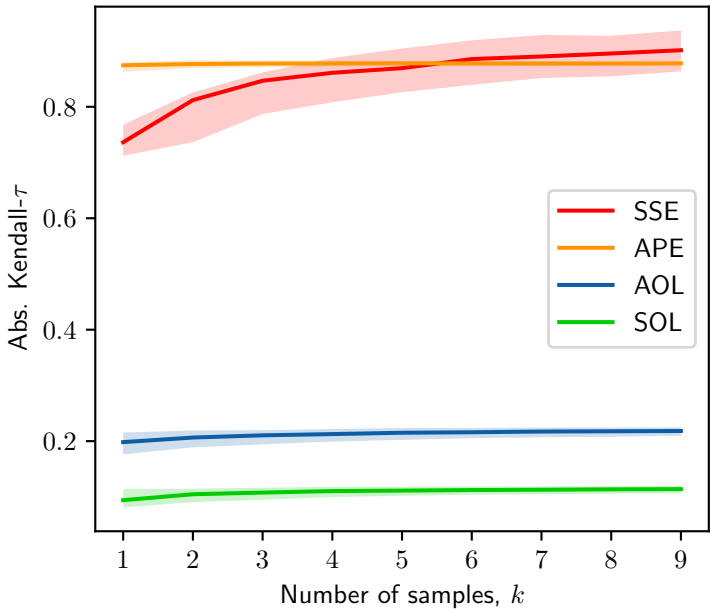
(b) Simulated DNR

Figure 4.7: Parity plots of the ground-truth-free methods against SSE on the simulated system. The ground-truth-free evaluations are normalized to $[0, 1.0]$ on the x-axis for comparison. Each point represents the mean of 10 repeated samples.

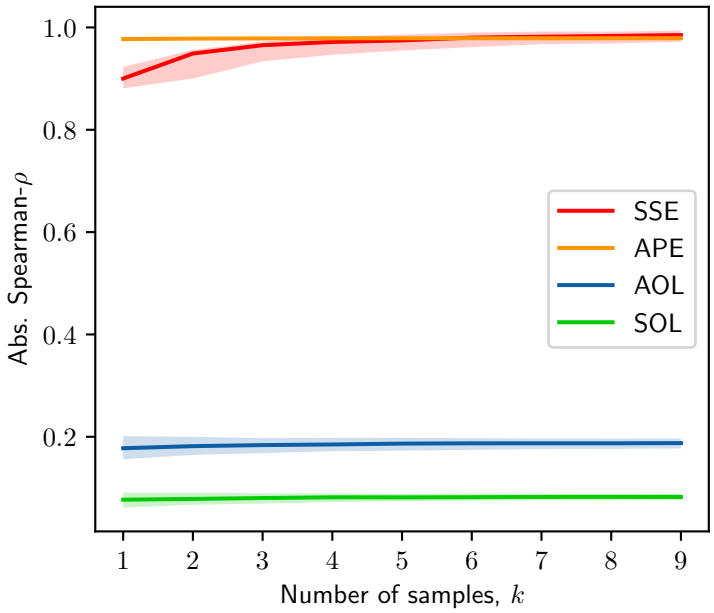


(c) DNR+C

Figure 4.7: Parity plots of the ground-truth-free methods against SSE on the simulated system (continued).

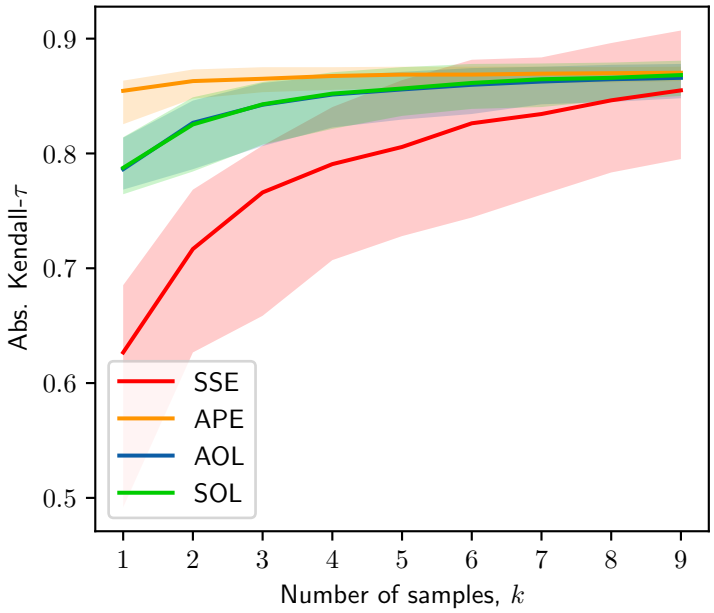


(a) Overall IMR Kendall-τ

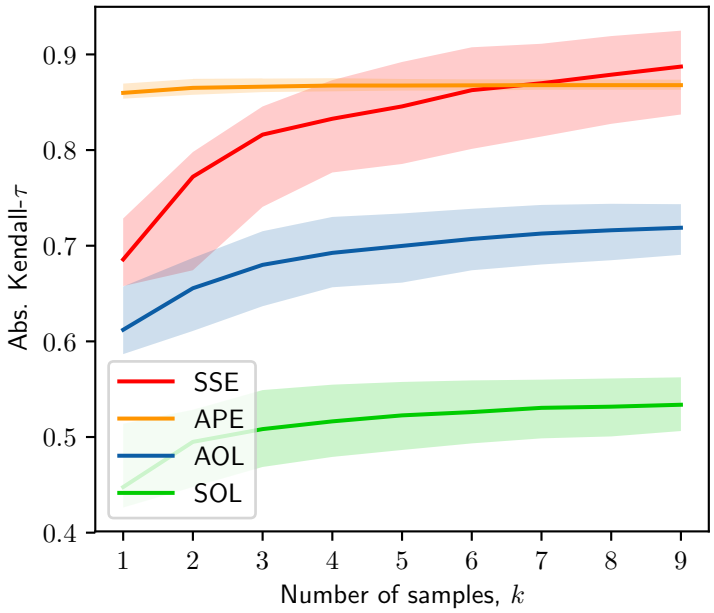


(b) Overall IMR Spearman-ρ

Figure 4.8: Correlation measures for each evaluation method using varying number of samples on the simulated system. The line indicates the median correlation over 1000 bootstrap instances, and the shaded region indicates the 5th and 95th percentiles.

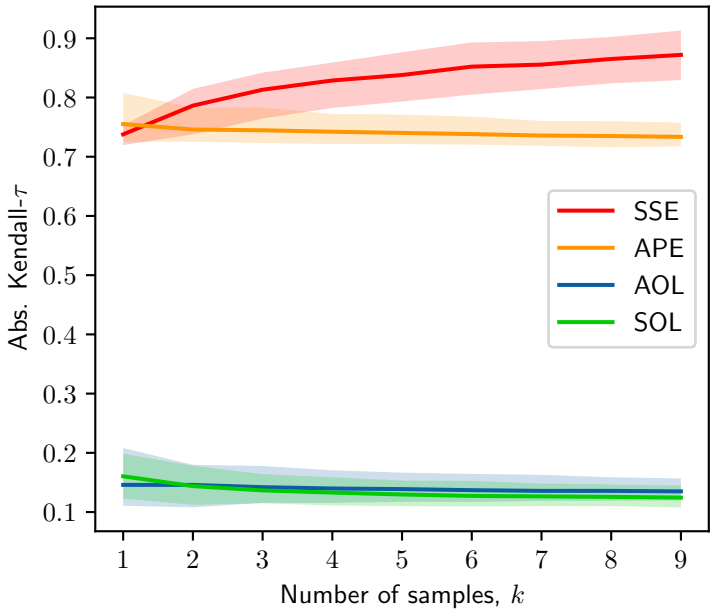


(a) Simulated DN Kendall-τ



(b) Simulated DNR Kendall-τ

Figure 4.9: Correlation measures for each evaluation method using varying number of samples on the simulated system (continued).



(c) Simulated DNR+C Kendall-τ

Figure 4.9: Correlation measures for each evaluation method using varying number of samples on the simulated system (continued).

4.4.6 Discussion

Kendall-τ versus Spearman-ρ

First, we consider the differences between the Kendall-τ and Spearman-ρ correlation metrics, both of which have been reported for the overall rankings of KITTI in Figures 4.2a and 4.2b, for the IMR in Figures 4.6a and 4.6b, and for the simulation in Figures 4.8a and 4.8b. We observe that Spearman-ρ generally reports larger correlations and lower variances than Kendall-τ, the trends are otherwise very similar. Given the relatively large bootstrapped populations we compute correlations over, it is possible that the concerns from Gilpin [1993] about convergence rate are not relevant here. As such, we continue our discussion using the Kendall-τ correlations.

Analysis of Simulated Results

We begin with analysis of the simulated results, as this will help build an intuition with which we can better understand the physical experiment results. First we consider the relations between metrics shown in the parity plots for each of the simulated test conditions. We see in the DN test condition results in Figure 4.7a a clearly linear, albeit noisy relation between the different normalized metrics and the ground truth SSE. In particular, we observe a strong

positive correlation between SSE and APE, and a strong negative correlation between SSE and AOL, as well as with SOL. This can be explained as the higher the log-likelihood of observations, the lower loss we expect to receive. Overall this suggests that all three ground-truth-free metrics perform well when only the observation noise magnitude changes.

When we add variable sensing rates for the DNR test condition, however, we see in Figure 4.7b that the AOL and SOL are no longer quite as linearly correlated with the SSE. In contrast, the APE appears unaffected and still exhibits a strongly linear relation. Adding in sensor cutout for the DNR+C condition, shown in Figure 4.7c, we see that the correlation between AOL and SOL with SSE entirely disappears: The portion of evaluations which do not experience cutout still correlate relatively well with SSE, but the remainder with cutout are distributed with no clear relation whatsoever. A similar separation exists for the APE evaluations, but preserves the overall relation with SSE. These two results together suggest that the observation log-likelihood-based methods cannot compensate with dramatically different numbers of observations across evaluations, whereas the APE can.

This qualitative conclusion is supported by our quantified correlation results as well. In Figure 4.9a we see that all methods perform relatively well, scoring τ values around 0.8 for 5 samples. It is interesting to note that SSE performs the worst in this condition, both in mean performance and variance. We suggest that the SSE results suffer from high variance due to *serendipity*, wherein an estimated trajectory may generate low loss by random chance, or "luck", even if the perception system did not receive high-quality observations. As such, multiple samples are needed to capture this effect from variance, whereas the quality of observations received, used by the ground-truth-free methods, is apparent from a single trial.

In the DNR test condition, however, AOL and SOL, in particular, do not perform as well. This can be seen in Figure 4.9b, where AOL and SOL achieve less than 0.7 and 0.5, respectively. As previously discussed, we propose that this drop in performance is due to a change in the number of observations received. SOL is affected the most because it explicitly associates the quantity of observations with the quality. Thus, it is difficult to differentiate between a large number of poor observations from a lower number of good observations. The AOL compensates for this by normalizing by the number of observations received, and thus does not suffer as great a performance drop. The APE still performs very well in this condition, exhibiting considerably lower variance and better sample efficiency than SSE.

Finally, in the DNR+C test condition, shown in Figure 4.9c, both the AOL and SOL methods completely fail, with correlations of ≈ 0.15, indicating nearly no correlation. In contrast, the APE still performs quite well, scoring above 0.7 with slightly less variance than SSE. We previously proposed that the AOL is able to partially compensate for changing numbers of observations by normalizing, and thus capturing the average quality of observations. With the presence of cutout, however, there are now different numbers of observations within parts of the trajectories, as well as across tests.

Analysis of KITTI Results

Next we consider the results from our experiments on the KITTI system, again starting by qualitatively inspecting the parity plots. Here we can see some similarities to simulated test conditions. For instance the ICP-LO road condition in Figure 4.1b resembles the DNR+C condition, suggesting that cutout occurs, whereas the SS-VO city parity in Figure 4.1c resembles more closely the DNR condition. None of the KITTI test conditions exhbit the ideal linear correlations seen in the DN simulated condition. This aligns with our expectations, as all of the KITTI test systems use error thresholding to reject poor estimates and thus exhibit changing observation rates. The computational intensity of the ICP-LO system results in very long processing times per scan for certain configurations, which results in cutout-like behavior on the high-speed road datasets since the velocity-based initialization is not as accurate as in the lower speed city datasets. Similar, we see that the SS-VO road condition is quite challenging to evaluate, as evident from the noisy relations seen in Figure 4.1d. Though the SS-VO system is less computationally intensive than the ICP-LO system, it is more affected by outliers from other moving vehicles due to its smaller field of view. Thus, in the high speed environment of the road, the system can end up tracking the wrong motions, resulting in highly biased estimates that are difficult to detect with the AKF.

Looking at the Kendall-τ metrics for the various metrics, we see that the APE significantly outperforms the AOL and SOL in ranking different configurations both within a test condition and across conditions. In Figure 4.2a we report a nearly constant Kendall-τ of ≈ 0.85 for the APE. In contrast, the AOL and SOL achieve only 0.6 and 0.5 correlations, respectively. This overall pattern is similar to the correlation results for the DNR simulated condition, suggesting that changing observation rates may be the most significant differential across conditions.

Within conditions, the trends are more nuanced. APE performs on-par with SSE in the ICP-LO conditions, seen in Figures 4.2c and 4.2d, whereas AOL and SOL both perform very poorly. This pattern resembles the DNR+C simulated condition correlations, supporting our previous hypothesis of ICP-LO cutout. On the SS-VO conditions, however, all three ground-truth-free metrics perform similarly, as can be seen in Figures 4.2e and 4.2f. This resembles the simulated DN condition, but with all correlations shifted downward, which can be explained by the AKF not fully capturing the noise dynamics. Part of this may be due to biases that the AKF cannot detect.

Analysis of IMR Results

Finally we consider the results from the IMR hardware experiments, beginning again with a qualitative assessment of the parity plots. Overall we see that all methods exhibit almost entirely linear relations with SSE for the ICP-LO test conditions, shown in Figures 4.5a and 4.5b, much like the simulated DN condition. However, there are a few outliers where very high SSE trials also

have very high AOL and SOL. This means that the overall trend of lower SSE correlating with higher AOL and SOL may not be reliable for parameter tuning. In the SPM-VO conditions, however, there is almost no discernable trend for the AOL and SOL, as can be seen in Figures 4.5c and 4.5d, while the APE still exhibits a strong linear relationship. This suggests that the SPM-VO system may exhibit extreme phenomena, such as dropout, which is expected given the nature of the system.

These qualitative trends are mirrored in the correlation metrics as well. We see that the APE performs well when comparing evaluations across test conditions, achieving Kendall-τ correlations of ≈ 0.7 in Figure 4.6a, while the AOL and SOL methods achieve ≈ 0.1. Within conditions, the AOL and SOL perform much better, in particular achieving ≈ 0.7 on the ICP-LO test conditions shown in Figure 4.6c and 4.6d, but are still outperformed by the APE, whose performance versus number of samples closely matches that of the SSE. The qualitative failure of AOL and SOL on the SPM-VO conditions is quantified as low correlations in Figures 4.6e and 4.6f of ≈ 0.1 to 0.3, whereas the APE again performs well at ≈ 0.7.

Chapter 5

Reconfiguring Parameters

In this section we discuss how perception systems can perform parameter tuning on-site without external supervision as a method of adapting to contextual changes. Specifically, we cast parameter tuning as a numerical optimization task with the system performance as the objective and the parameters as the input. Further, by using the techniques developed in Chapter 4, we avoid the need for ground truth, allowing parameter tuning to be performed in wide variety of applications.

We first survey existing approaches for tuning parametric behaviors of robots and algorithms in various fields, and then formalize our parameter tuning task as numeric optimization. We also discuss the applicability of various popular optimization techniques, and present experimental results demonstrating the practicality and importance of fully automated parameter tuning on the KITTI and IMR systems.

5.1 Related Works on Parameter Tuning

There exists a vast body of literature that falls under the term "parameter tuning," though the nature of a "parameter" in these works and the applications themselves vary considerably. In our setting, we consider parameters as a relatively low-dimensional specification of behavior for a complex system. We cover here relevant works which are similar in concept or application to our work on perception system tuning.

5.1.1 Robot Perception

The idea of tuning perception parameters to match the task at hand can be traced back to active perception, started by Bajcsy [1988] and Aloimonos et al.. These works, particularly in active vision, focused on geometric factors, such as viewpoint selection, but also considered data capture parameters, such as camera exposure and zoom.

Following these concepts was active localization, proposed by Burgard et al. [1997], who demonstrated that altering a mobile robot's motion and use of multiple sonar rangefinders could improve localization performance. Fairfield and Wettergreen [2008] used this technique for localizing an underwater robot by moving to maximize information gathered by sonar sensors against a map. More recently, Hausman et al. [2016] considered switching between different modalities of a

Controlling perception systems has also been considered by Ondruska et al. [2015] for balancing between power usage and localization accuracy. By modeling localization uncertainty, the authors demonstrate a policy for disabling the robot's visual odometry system when it is not needed, allowing them to save energy.

5.1.2 Robot Control

The bulk of modern parameter tuning work can be found in the setting of control and learning for dynamical systems. In particular, there is a large body of work for tuning parametric gaits to achieve robust and fast motion, explored in the context of bipeds by Calandra et al. [2014] Wang et al. [2009a] and Endo et al. [2008], quadrapeds by Chernova and Veloso [2004], Kohl and Stone [2004], and Röfer [2005], and for snakes by Tesch et al. [2011]. The dynamics of a particular gait depends heavily on the local environment, making it challenging to model and predict performance. As such, these works rely on empirical trials using the robot itself to measure performance and search the parameter space. Many optimization approaches have been explored, as Chernova and Veloso [2004] and Röfer [2005] used evolutionary algorithms, while Kohl and Stone [2004] used numerical gradients and Tesch et al. [2011] used Bayesian optimization.

Also related are works on tuning parameters of classical control algorithms. Marco et al. [2016] demonstrate learning an inverted mass balancing controller by optimizing LQR weights, which in turn produce a closed-loop controller. Berkenkamp et al. [2016] similarly optimize a quadrotor's flight performance by applying a conservative "safe" Bayesian optimization approach to tune controller gains.

5.1.3 Optimizers and Solvers

Algorithm parameter tuning has been studied outside of robotics for a while now, primarily in the computational sciences. Tuning has been demonstrated for constraint satisfaction solvers by Hutter et al. [2011], for formal verification algorithms by [Hutter et al., 2007], evolutionary algorithms by Smit and Eiben [2009], and image denoising by Zhu and Milanfar [2010]. Bourki et al. [2010] demonstrated tuning a Go-learning algorithm over a discretized configuration space, and show that a uniform search approach can take advantage of parallelization, and in some instances, outperform a model-based search. This is similar to many of the aforementioned works on gait tuning, such as that by Kohl and Stone [2004], where multiple robots were used to run tests in parallel.

We assume in our setting that we have only one robot, but could extend our approach using the techniques described by Snoek et al. [2012] to parallelize the Bayesian optimization if there were multiple robots available.

In addition to parallelization, Snoek et al. [2012] and Loshchilov and Hutter [2016] consider evaluation costs that may arise when tuning hyperparameters of machine learning algorithms, *i.e.*, training a small versus large neural network. Kandasamy et al. [2016] consider a related setting where the search algorithm can evaluate at varying fidelities, or as described in their machine learning application, train on varying amounts of data. In this work we use only a single empirical test to evaluate the performance of the perception system, though our approach could be extended in the future to incorporate multiple tests of varying length and thoroughness.

5.2 General Reconfiguration as Optimization

We begin with a high-level comparison of optimization approaches. Optimization in general is the task of finding inputs to a function that produce an extreme value. With this definition, parameter tuning can be viewed as optimizing parameter values to result in maximum performance. However, optimization is a broad task applicable to a huge variety of fields.

5.2.1 Characterizing Reconfiguration

To utilize existing work on optimization, we must first classify the parameter tuning task in optimization terms. Below we consider aspects of parameter tuning relevant for selecting an appropriate optimizer and relate them to the previously discussed prior works.

Characteristics of the Input Space

The first distinction we consider is the nature of the variables we seek to optimize. Parameters are quite varied in nature, sometimes involving mixes of continuous, integer, and categorical variables. *Numerical optimization* refers to optimizing continuous variables, *integer programming* for optimizing integers, and *categorical optimization* for optimizing categorical variables. Were we to try and optimize all of these variable types, we would need to delve into *mixed-variable optimization*.

Most of the prior work discussed previously falls into the numerical optimization category. For instance, Kohl and Stone [2004] tuned gaits parameterized by continuous values. A notable exception is the work of Hutter et al. [2011] on tuning with categorical and numerical parameters. It can be argued that optimization of group elements, such as viewpoints and poses, demand special treatment, but these problems can often be transformed into a numerical optimization. An example of this can be found in the work of Grisetti et al. [2010] for the case of graph-based SLAM.

Due to scope and complexity, we restrict ourselves to consider only continuous variables. In practice, many integer-valued parameters can be well-represented as continuous variables due to their high resolution. Further, the bulk of parameters in the systems we consider are numerical in nature. For completeness, we will discuss possible extensions to our work for the mixed-variable setting as well, but do not show any experiments.

We also note that many perceptual systems have anywhere from two or three to dozens of tunable parameters. As such, we would like an approach that can scale to roughly 20 or so parameters. This puts us in the middle range of typical optimization dimensionality: neither very low-dimensional nor high-dimensional.

Knowledge of the Objective Function

Next we consider the objective function we are seeking to optimize. In parameter tuning, we desire to optimize the expected performance of the system. As discussed in Chapter 4, this is typically evaluated by executing the system and observing the performance on-site. Thus, we can query the objective function to determine the performance of a particular configuration, but do not have an analytical form of the objective function. The lack of knowledge of the objective function's form means it is difficult to use analytical models to find good parameters without interacting with the system itself. As such, a more practical approach is to sequentially test configurations on the system itself to provide feedback for the optimizer, a regime referred to as *sequential black-box optimization*.

The black-box formulation is common in robotics, as the complexity of interactions generally precludes analytical modeling. For instance, the work by Chernova and Veloso [2004] and many other authors on optimizing quadruped gaits involves many complex dynamical interactions. Similarly, there is very little theory for predicting the effects of machine learning hyperparameters, the task of interest to Snoek et al. [2012]. The effectiveness of these approaches comes from their use of the system itself, as opposed to relying on inaccurate analytical models.

Characteristics of the Objective Function

Perception systems interact with the world, and thus, can exhibit a significant amount of randomness in their behavior. This is discussed in Chapter 4, and though we discuss methods for reducing variance in performance estimates, it is impossible to remove entirely. Thus, the performance of a particular configuration may vary between repeated queries, meaning that the objective function is noisy. This is referred to as a *stochastic optimization* problem.

Stochasticity as a property often comes with a black box formulation as a necessity. In other words, the cost of being agnostic about the objective function is inexplicable variance. For instance, executions of the same quadruped gait may result in slightly different motions due to variations in initial conditions,

unmodeled effects, etc. As such, Kohl and Stone [2004] used the average of multiple evaluations of a gait to compensate for stochasticity. An alternative approach is that used by Tesch et al. [2011], which is to use an optimization formulation that explicitly models stochasticity. Appropriately enough, the work by Zhu and Milanfar [2010] on denoising images uses a noiseless optimization approach, as the denoising algorithm itself is deterministic.

We also note that the objective function itself may not be very smooth. Parameters controlling discontinuous behavior, such as outlier rejection thresholds, may dramatically affect the performance for small perturbations. This also means that the behavior of the performance may vary dramatically across the entire configuration space, and thus searching in a small local area may not be a reliable way to find good performance. Further, we do not desire to rely on a good initialization for a local search. Together, these factors mean that we desire a *global optimization* over the entire configuration space.

Discontinuity in the objective is quite common in robotics applications and locomotion in particular, as contact forces are inherently discontinuous. As such, changing the gait of a quadruped ever so slightly may dramatically change its behavior. Röfer [2005] use a gradient-free optimization approach to compensate for these discontinuities, whereas Wang et al. [2009a] use an optimization approach that smooths out objective discontinuities by introducing stochasticity in the optimization process itself. Other works explicitly rely on objective continuity and smoothness, such as in the case of Berkenkamp et al. [2016] to safely search the input space.

Optimization Quality Criteria

Finally, we also consider the desired outcomes and constraints for our optimization task. First, we desire that self-tuning be done in a relatively short time. Since evaluating the objective function corresponds to executing the physical robot, methods that require a large number of evaluations will require a large amount of time to tune. We can translate this desire as an upper-bound on the number of evaluations allowable. This is often referred to as *budgeted optimization* or *sample-efficient optimization*.

Sample efficiency is particularly important in robotics and machine learning, as executing a robotic or learning system many times can be quite resource-intensive. For instance, the work by Tesch et al. [2011] uses an optimization approach specifically designed to be sample efficient, as running trials on the snake robot requires considerable time. In contrast, Bourki et al. [2010] demonstrate the effectiveness of a highly parallelizable but otherwise extremely sample-inefficient optimization approach, as their Go-solving algorithm is easily executed on a large number of servers.

We also do not require that the output of the optimization be the globally-optimal parameters. Putting aside the fact that this is infeasible in our setting, we intuitively only require that the optimization produce a sufficiently performant configuration. This is often referred to as *near-optimality* or *approximate optimization*.

The importance and practicality of searching globally varies across applications. For instance, the quadruped gait tuning work by Kohl and Stone [2004] relies on improving an existing gait, whereas that of Tesch et al. [2011] on snake gaits does not use any initial information. This is justified by Kohl *et. al.* as due to poor initializations resulting in extremely slow learning, whereas the learning process gets stuck in local minima when starting from finely hand-tuned initializations. In contrast, the gait parameterization for the snake is relatively low-dimensional, such that most gaits result in useful motion. In both of these cases, global search is important to avoid the plethora of poor local minima, but once a good gait is found, it is practical to rely on local refinement. In contrast, Bourki et al. [2010] note that the hyperparameters of their solver vary dramatically depending on the application, and as such, they cannot rely purely on local refinement.

5.2.2 Black Box Optimization Approaches

Here we review common numerical optimization algorithms and discuss their applicability to parameter tuning, focusing on sample efficiency and robustness to stochasticity. For consistency with existing literature, we will discuss algorithms for a maximization task, though our task is actually a minimization.

Nelder-Mead Simplex (NMS)

NMS is a well-known algorithm with good empirical performance, but few theoretical guarantees. At a high level, NMS maintains a set of points in the input space that define a simplex in which the optima is believed to exist. It uses a set of heuristic updates to shift, shrink, and expand this simplex by evaluating the objective at new points. However, NMS does not compensate for stochasticity and has been shown by Han and Neumann [2006] to be sample inefficient in higher dimensions, making it a poor candidate for parameter tuning.

Covariance Matrix Adaptation (CMA-ES)

Introduced by Hansen and Ostermeier [2001], CMA-ES can be thought of a standard evolutionary algorithm where the population of individuals is parameterized by a multivariate Gaussian. At each iteration, individuals (inputs) are sampled from the population distribution and evaluated. The population is then updated according to the individual fitnesses (objective) to favor high-fitness individuals. CMA-ES is known to be robust to noisy evaluations and be relatively efficient in high dimensions with potential for parallelization due to its generation-based iterations. It is also unaffected by affine transformations of the input space when the population covariance is allowed to be dense, making it overall quite effective as a generic black box optimizer. CMA-ES has seen success in generic numerical optimization as well as machine learning for hyperparameter tuning Loshchilov and Hutter [2016] and in dynamic legged walking community for optimizing control algorithms Wang et al. [2009a].

Numerical Gradients (NG)

We use the term "numerical gradients" to refer to a large family of algorithms which estimate the gradient of the objective with respect to the inputs by sampling the objective function around a linearization point. In some cases the sample deviations are regularly spaced, as in Kohl and Stone [2004], whereas in others they are normally distributed Mania et al. [2018]. Once estimated, the gradient is used to increment the linearization point towards a local optima, and the process is repeated until convergence.

Previous applications of NG methods to reinforcement learning for physical systems by Kohl and Stone [2004] and simulated systems by Mania et al. [2018] have shown them to be robust to stochasticity in relatively high dimensions. However, these methods are still dependent on initializations, and thus must be restarted many times to assure good performance, making their sample efficiency too low to be practical for tuning physical systems.

Uniform Random Search (UR)

UR is often studied as an exercise, but in fact exhibits good theoretical properties. Random sampling can be generated in a number of ways to reduce this variance. Random search is easy to parallelize, making it ideal for tasks such as hyperparameter tuning, where it can be surprisingly effective Bergstra and Bengio [2012], Hutter et al. [2011]. However, it is unlikely that we will have opportunities to parallelize tuning on physical systems, as this would require multiple robots. Nonetheless, UR provides a reliable baseline for determining the difficulty of a particular optimization task.

Bayesian Optimization (BO)

Bayesian Optimization (BO) is a black box optimization technique that incorporates objective function queries into a model of the objective function, which is then used to select new promising queries. These approaches can be highly sample-efficient if the model and search strategy are well-matched to the task. As such, BO and its variants have seen success in applications where querying the objective function is expensive, *i.e.* running an experiment on a robot, and has been demonstrated for quadrotor tuning by Berkenkamp et al. [2016] and for snake gait learning by Tesch et al. [2011].

BO approaches can be classified largely by their query strategy and choice of model. Popular query strategies include Expected Improvement, which was used by Tesch et al. [2011], SafeOpt Berkenkamp et al. [2016], and Upper Confidence Bounds Srinivas et al. [2010]. Most of these approaches use a Gaussian Process (GP) to model the relation between inputs and objective, as GPs can naturally account for stochasticity and can report both mean predictions as well as uncertainty. Other approaches, such as work by Hutter et al. [2011], uses a Random Forest to incorporate both numerical and categorical parameters.

In our work we use the Gaussian Process Upper Confidence Bound (GP-UCB) algorithm, first proposed by Srinivas et al. [2010]. GP-UCB is simple and

easy to implement, and has been proven to have good cumulative regret bounds if the true objective function is appropriately representable (has low reproducing kernel Hilbert space norm). Recently, GP-UCB has also been shown to have good simple regret bounds for the commonly used squared exponential (SE) and Matérn kernels by Scarlett et al. [2017].

The GP-UCB algorithm models the objective function with a Gaussian Process model that integrates all previous queries. At each iteration t, the algorithm queries the input x_t with the highest upper confidence bound $\varphi_t(x)$ which is given (with high probability) by:

$$\varphi_{t+1}(x) = \mu_t(x) + \sqrt{\beta_t}\sigma_t(x) \tag{5.1}$$

$$x_t = \arg\max_{x \in \mathbf{X}} \varphi_t(x) \tag{5.2}$$

where $\mu_t(x)$ and $\sigma_t(x)$ are the mean and standard deviation predicted at input x by the GP model, and β_t is a scheduled exploration rate.

Multi-Fidelity Bayesian Optimization (MFBO)

MFBO is an extension of BO which considers optimizing an objective using biased lower-fidelity objectives that are cheaper to evaluate. In our setting, this corresponds to running a faster evaluation using a smaller benchmark, which gives an evaluation biased from the full benchmark in an unknown way. MFBO provides a principled way of using a hierarchy of such lower-fidelity objectives to quickly explore the input space and gradually expend more effort refining promising areas.

Conceptually, MFBO is very similar to BO, though instead of a single objective function $f(x)$ we now consider a series of objective functions $f^{(m)}(x)$, $m = 0, \ldots, M$, where each function has a level of fidelity m with $f^{(M)}(x)$ at the highest fidelity equal to the objective function we wish to optimize. Intuitively, we would like these lower fidelity functions to represent objective queries that use fewer resources, but which also return less information. The example used by Kandasamy et al. [2016] in the original presentation of MFBO is a machine learning hyperparameter tuning task: We would like to find parameters that maximize validaiton peformance as quickly as possible, but running learning on the large dataset is time-intensive. Instead, we can learn on a subset of the data, but this will return a slightly incorrect validation performance. Importantly, these errors will be biased due to the data selection, and as such, we cannot simply represent lower fidelities as higher variance observations.

Using this intuition, Kandasamy et al. [2016] define the lower fidelity functions as being within some bias $\zeta^{(m)}$ of the true objective function, with higher fidelities having less bias:

$$|f^{(m)}(x) - f^{(M)}(x)| \leq \zeta^{(m)} \ \forall x \in \mathbf{X} \tag{5.3}$$

$$\zeta^{(1)} > \zeta^{(2)}, \ldots, \zeta^{(M-1)} \tag{5.4}$$

where $\mathbf{X}$ is the optimization domain. This representation can be adapted into the UCB algorithm discussed for standard BO, as each bias can be thought of

as an additional upper bound. Modifying Equation 5.1, the bound at iteration t for each fidelity becomes:

$$\varphi_{t+1}^{(m)}(x) = \mu_t^{(m)}(x) + \sqrt{\beta_t}\sigma_t^{(m)}(x) + \zeta^{(m)} \tag{5.5}$$

where $\mu_t^{(m)}$ and $\sigma_t^{(m)}$ are the mean and variance predicted by a GP at each fidelity m. Accordingly, the lowest of these upper bounds is the tightest upper bound, which serves as the UCB acquisition function used to pick the next sample x_t:

$$\varphi_t(x) = \min_{m=1,\ldots,M} \varphi^{(m)}(x) \tag{5.6}$$

$$x_t = \arg\max_{x \in \mathcal{X}} \varphi_t(x) \tag{5.7}$$

Maximizing the acquisition function tells us where to next sample the objective function, but it does not tell us which fidelity to use. The original presentation of MFBO uses the lowest fidelity for which the uncertainty is greater than a threshold $\gamma^{(m)}$:

$$m_t = \arg\min_{m=1,\ldots,M} m | \beta_t^{1/2}\sigma_t^{(m)}(x_t) \geq \gamma^{(m)} \tag{5.8}$$

The original presentation of MFBO presents a heuristic algorithm for learning both the biases $\zeta^{(m)}$ and thresholds $\gamma^{(m)}$ online.

5.3 Experimental Validation

We explore various approaches for automatic parameter tuning with experiments on our KITTI and IMR state estimation systems. Our primary goal is to determine which optimization algorithms work well in practice for this task. In addition, we validate that the conclusions of Chapter 4 about the effectiveness of APE evaluations for parameter tuning by comparing it against tuning with ground truth. Finally, we seek to demonstrate that parameter reconfiguration on-site can mitigate performance degradation from contextual effects.

We compare various optimization algorithms using the KITTI system, as it is relatively easier to run a large number of experiments compared to the IMR. Conversely, we validate the APE for tuning and demonstrate context adaptation on the IMR system, as it exhibits more significant contextual variations than the KITTI dataset. We discuss our procedure in depth below, followed by results from each of our experiments and a discussion of the results in conclusion.

5.3.1 Optimization Algorithms Tested

We list here the algorithms we consider and describe the particular implementations used in our experiments. Like in Chapter 4, we use the log SSE loss on two-dimensional body velocities $(\dot{x}, \dot{y}, \omega)$ and its corresponding log APE to quantify the system performance.

Uniform Random (UR)

We uniformly randomly sample configurations using the random number generator from the Python `numpy` package with a fixed seed of 0. We simulate the effect of repeated UR trials by randomly reordering the samples.

Covariance Matrix Adaptation (CMA-ES)

We use the implementation of CMA-ES from the Python `cma` package[1]. In our experiments we fix the random seed for each trial to be $9000 + i$ for trial i, e.g. trial 1 will use seed 9001, etc. We configure the algorithm to use 10 samples in each iteration, but otherwise use all default parameters.

Bayesian Optimization (BO)

Our BO implementation uses the `GPy` package[2] implementation of Gaussian Process (GP) models. We use a GP with an Matern kernel ($\nu = 1.5$) with automatic relevance detection (ARD). Each optimization is initialized with 10 uniformly randomly sampled configurations, with model hyperparameters fitted after the initial samples and then every 5 samples afterwards. We use the exploration rate schedule form given in Srinivas et al. [2010], but as a function of the elapsed optimization runtime t as $\beta = \alpha d \log(\gamma t)$, where d is the configuration space dimensionality. In our experiments we used $\alpha = 0.5$ and $\gamma = 0.2$ with t measured in seconds, for relatively aggressive exploration behavior.

5.3.2 Test Metrics

We use the following test metrics to quantify the efficacy of various optimizers and evaluation approaches for parameter tuning, and to quantify the benefits of on-site tuning for parameter adaptation.

Best Seen Performance

An ideal optimization and evaluation approach will return a final solution that results in good performance. However, given the variability of different convergence criteria that determine when an optimization is "done", it is also illustrative to consider the behavior of an optimizer as it explores the configuration space. Specifically, we are interested in seeing what is the performance of the configuration the optimizer believes is the best seen so far, as this allows us to infer final solution performance over varying convergence criteria. When optimizing with APE evaluations, the best seen configuration is the previously tested configuration with the best APE. As such, the best-seen performance will sometimes increase over time. We also compute the variance of the best-seen performance over repeated optimization trials. This gives us insight into how reliable an optimization approach is.

[1] `https://github.com/CMA-ES/pycma`
[2] `https://github.com/SheffieldML/GPy`

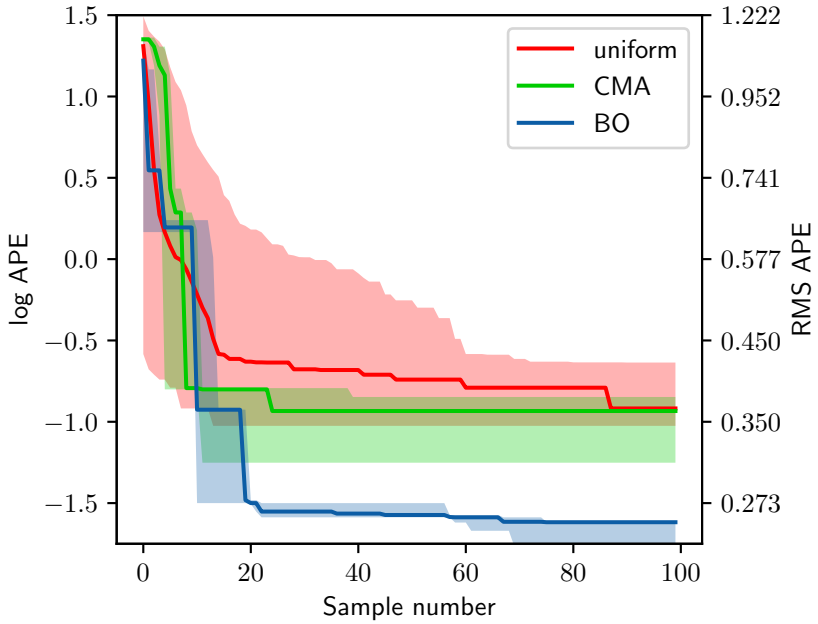
Context Suboptimality

We propose to quantify the benefit of on-site adaptation by measuring the change in performance when tuning in one context and operating in another. In other words, this suboptimality captures the loss in performance incurred by ignoring changes in context as the robot transitions between environments.

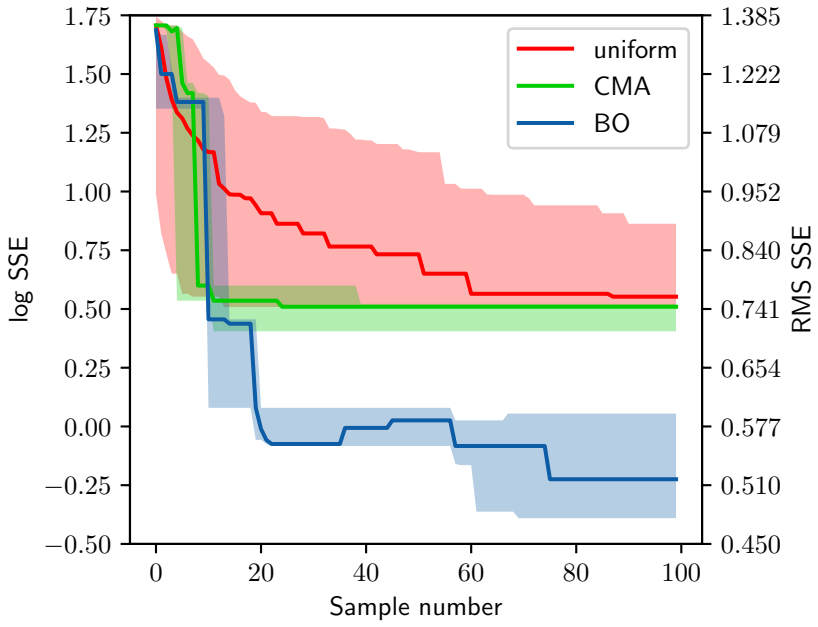
5.3.3 KITTI Experiments

We ran experiments on the `city` and `road` contexts in the KITTI dataset, using approximately 3 minutes of trajectories for each context. We ran each of the optimization algorithms on these two contexts using APE evaluations for both the ICP-LO and SS-VO systems, repeated three times each, for a total of 48 optimizations. For MFBO we created a mid-fidelity evaluation using just one 30 second trajectory in each context, and a low-fidelity evaluation using the first 10 seconds of the mid-fidelity trajectory. All algorithms were given a time budget of 5 hours, which is approximately 100 highest fidelity evaluations.

Traces of the best-seen performance for the tested optimizers are shown in Figures 5.1. We also plot the range of parameter values whose performance surpasses a threshold in Figure 5.2 as a way to visualize the importance and fineness of each parameter.

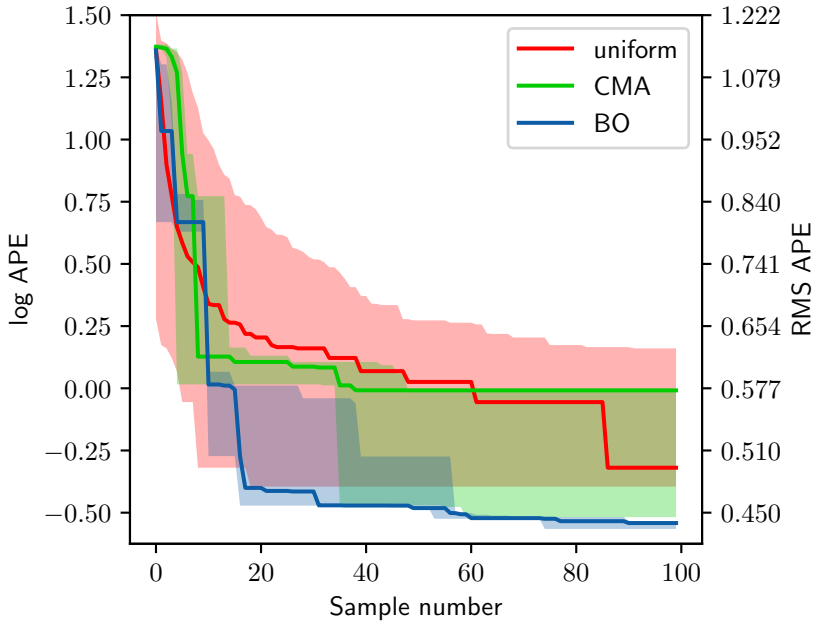


(a) ICP-LO city APE

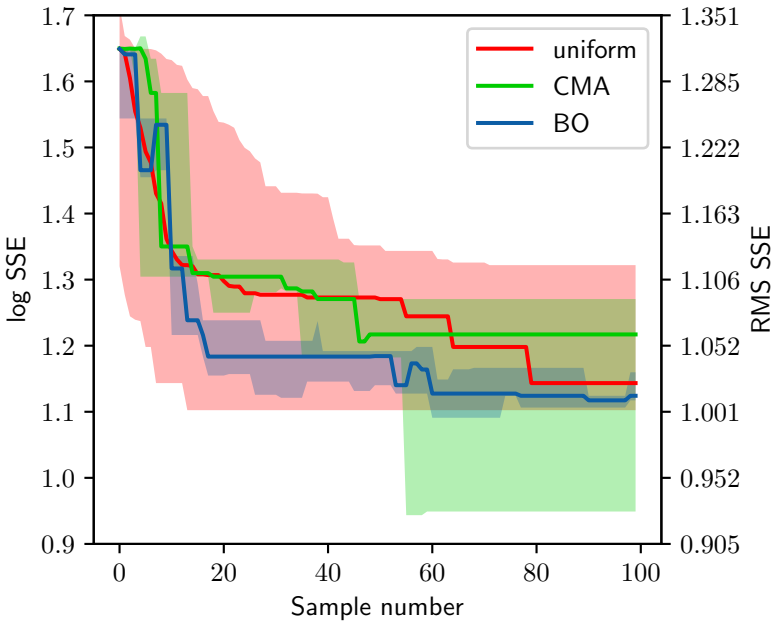


(b) ICP-LO city SSE

Figure 5.1: Best-seen performances on KITTI tuning with uniform, CMA, and BO approaches for approximately 6.5 hours of evaluation per trial. Solid lines denote medians, and filled regions denote extreme ranges for CMA and BO over 3 trials, and the 5th and 95th percentile performances over 1000 bootstrap samplings for the uniform results.

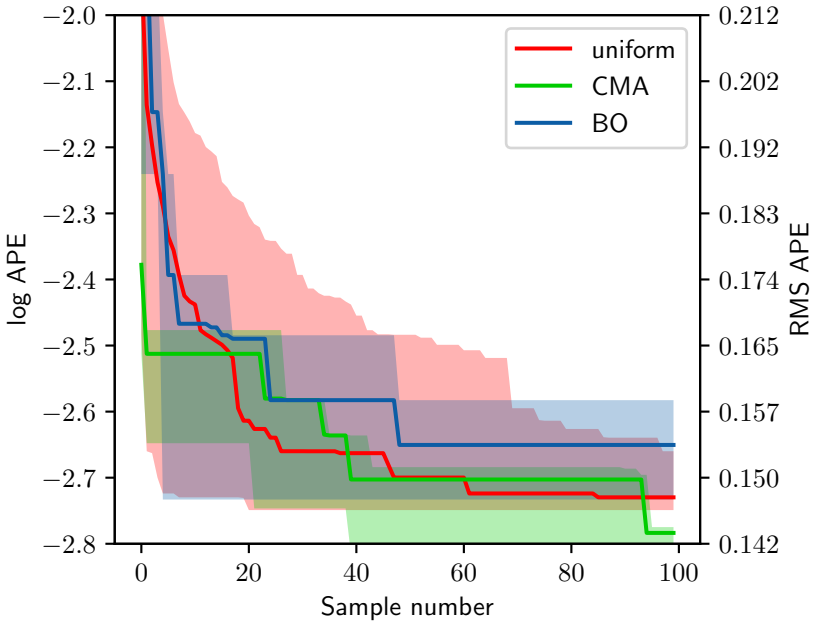


(c) ICP-LO road APE

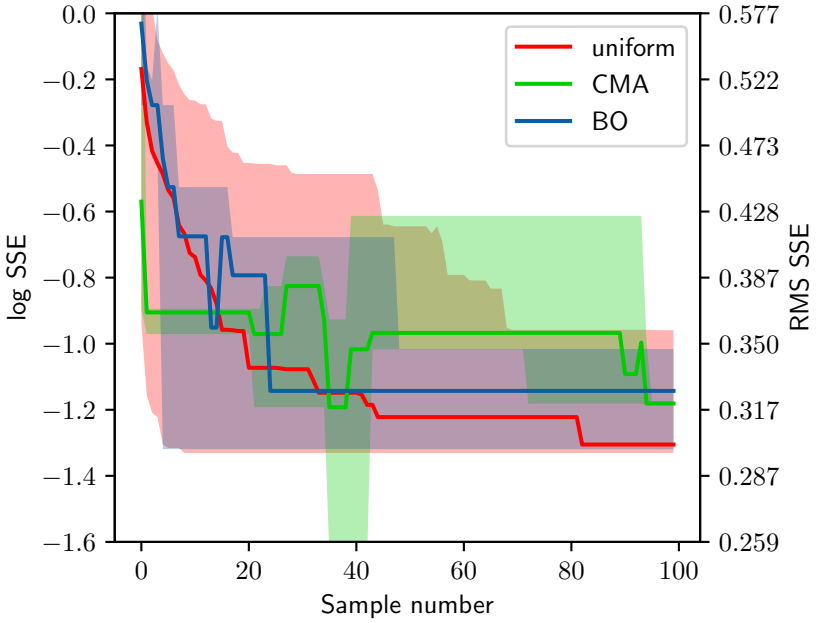


(d) ICP-LO road SSE

Figure 5.1: Best-seen performances for tuning on KITTI with uniform, CMA, and Bayesian optimization approaches (continued).

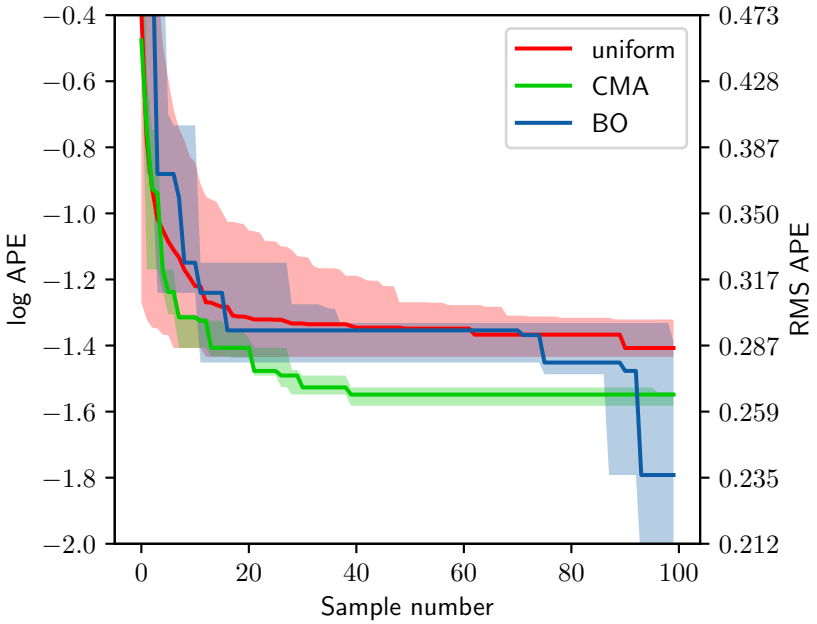


(e) SS-VO city APE

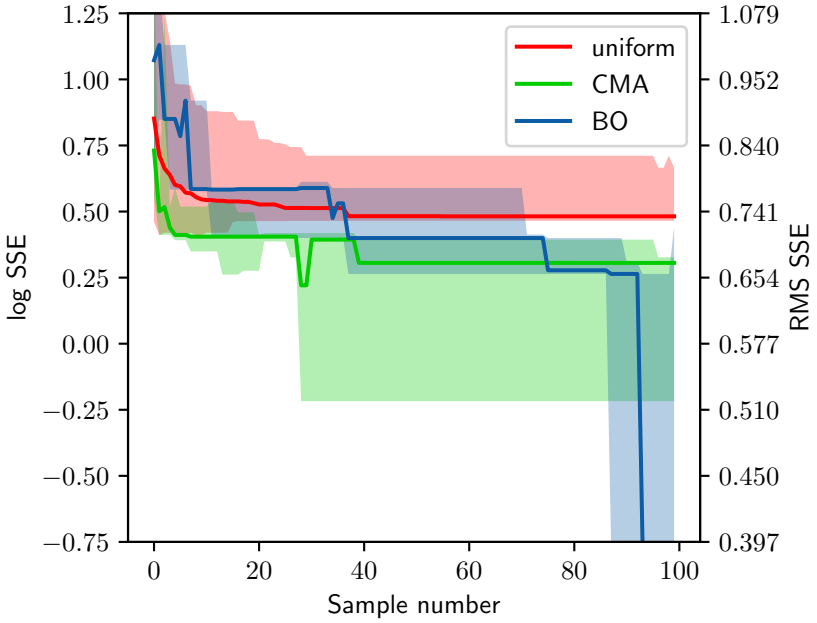


(f) SS-VO city SSE

Figure 5.1: Best-seen performances for tuning on KITTI with uniform, CMA, and Bayesian optimization approaches (continued).

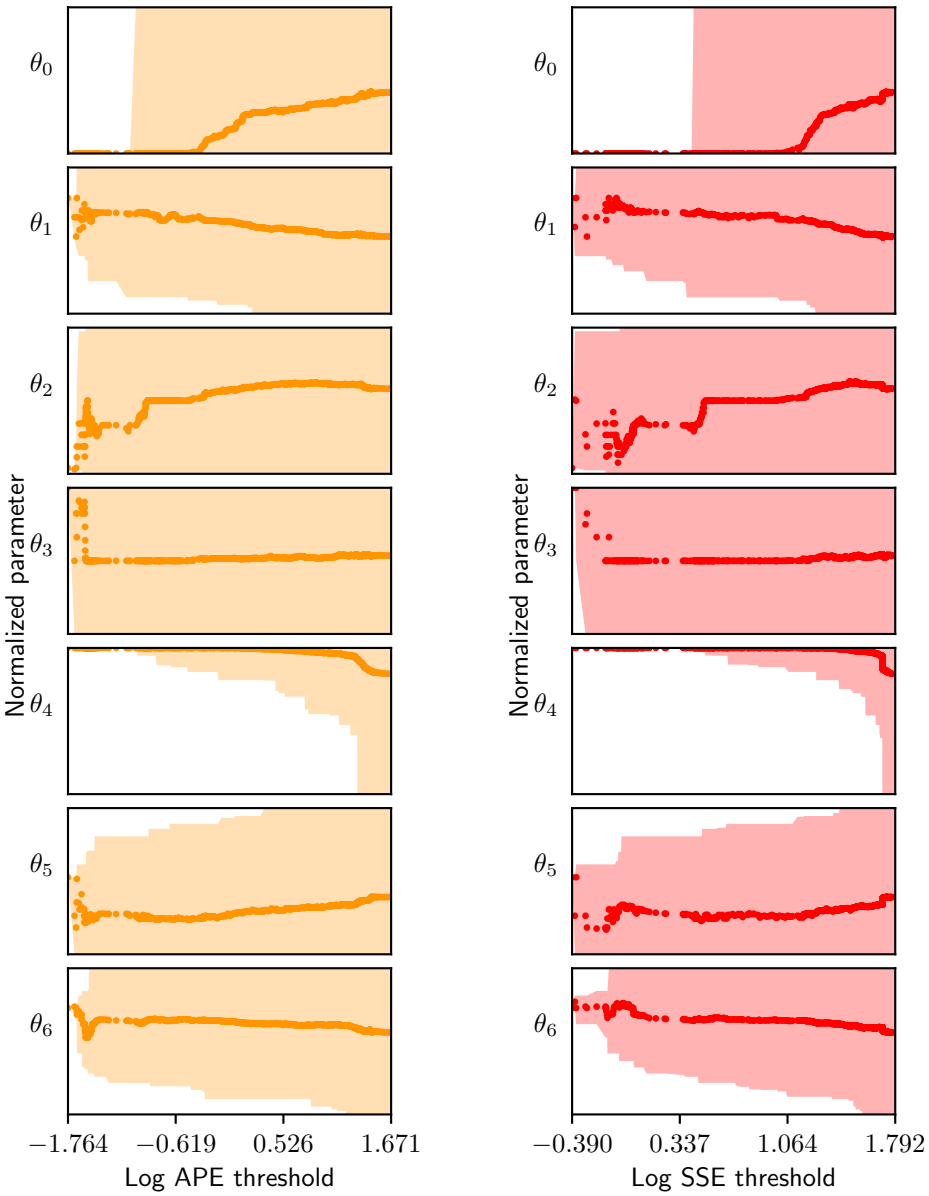


(g) SS-VO road APE



(h) SS-VO road SSE

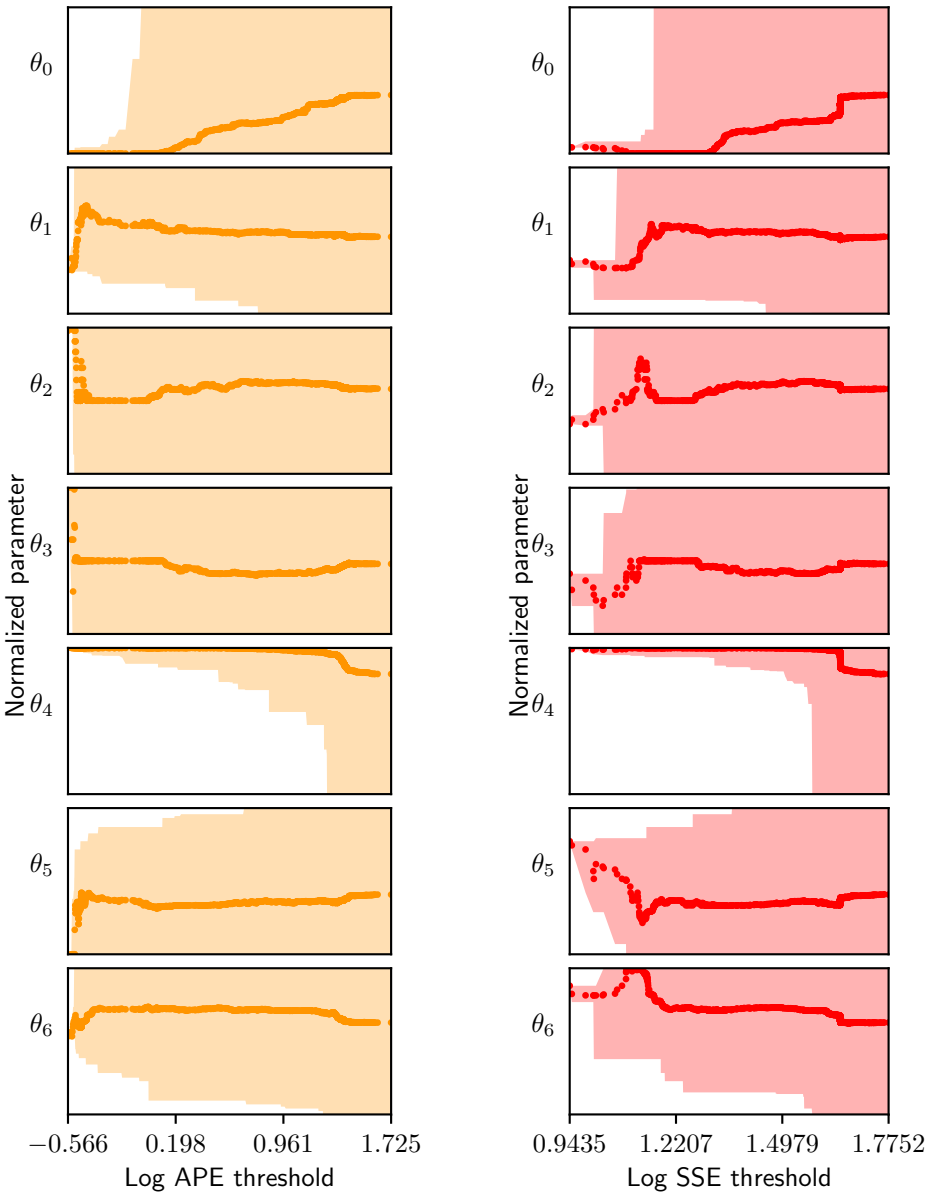
Figure 5.1: Best-seen performances for tuning on KITTI with uniform, CMA, and Bayesian optimization approaches (continued).



(a) ICP-LO city APE

(b) ICP-LO city SSE

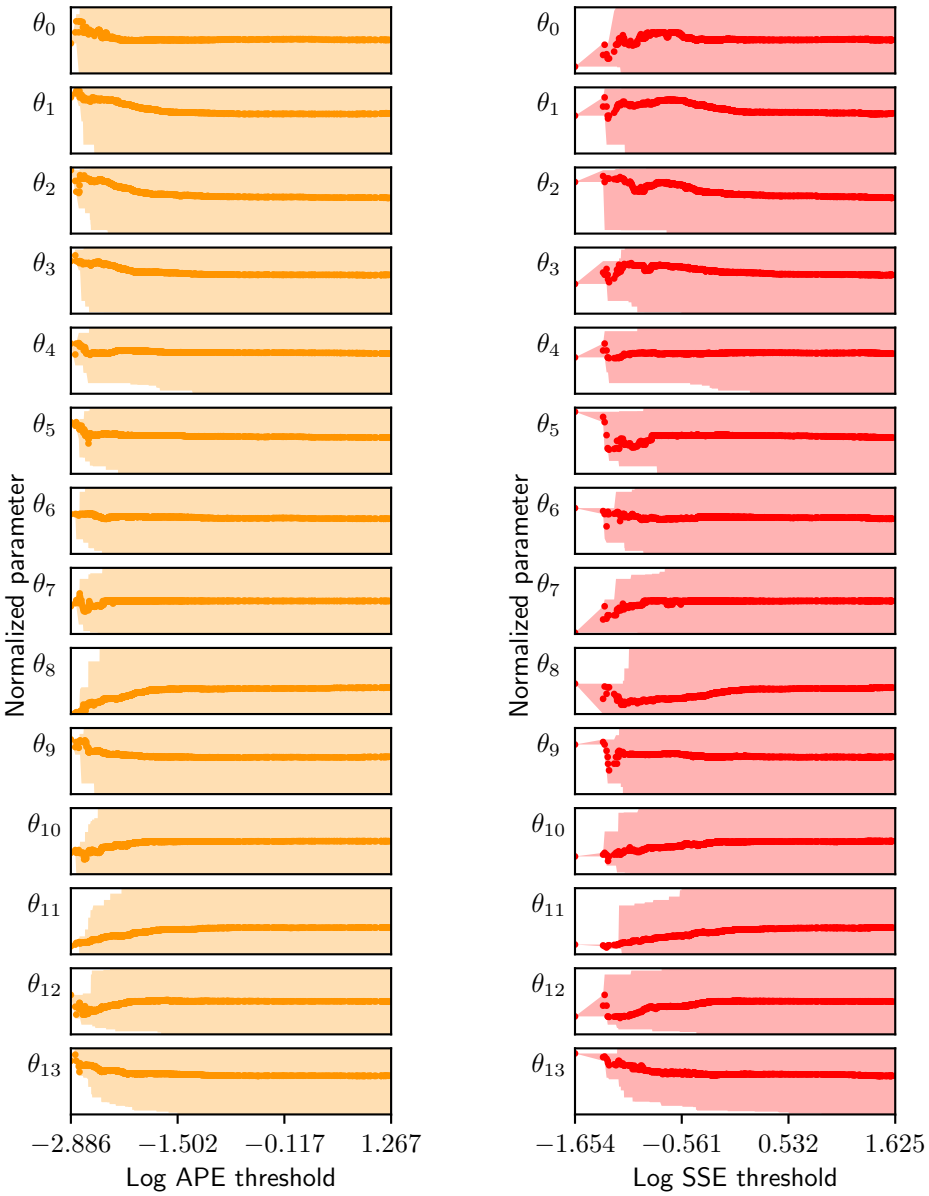
Figure 5.2: Ranges of parameters versus performance thresholds as seen across all optimization trials on KITTI. Dots represent the median of the set of parameters performing at least as well as the x-coordinate value, and the shaded area shows the extreme ranges of the parameters. Note that the upper (rightmost) threshold range is reduced from the full dataset to show detail.



(c) ICP-LO road APE

(d) ICP-LO road SSE

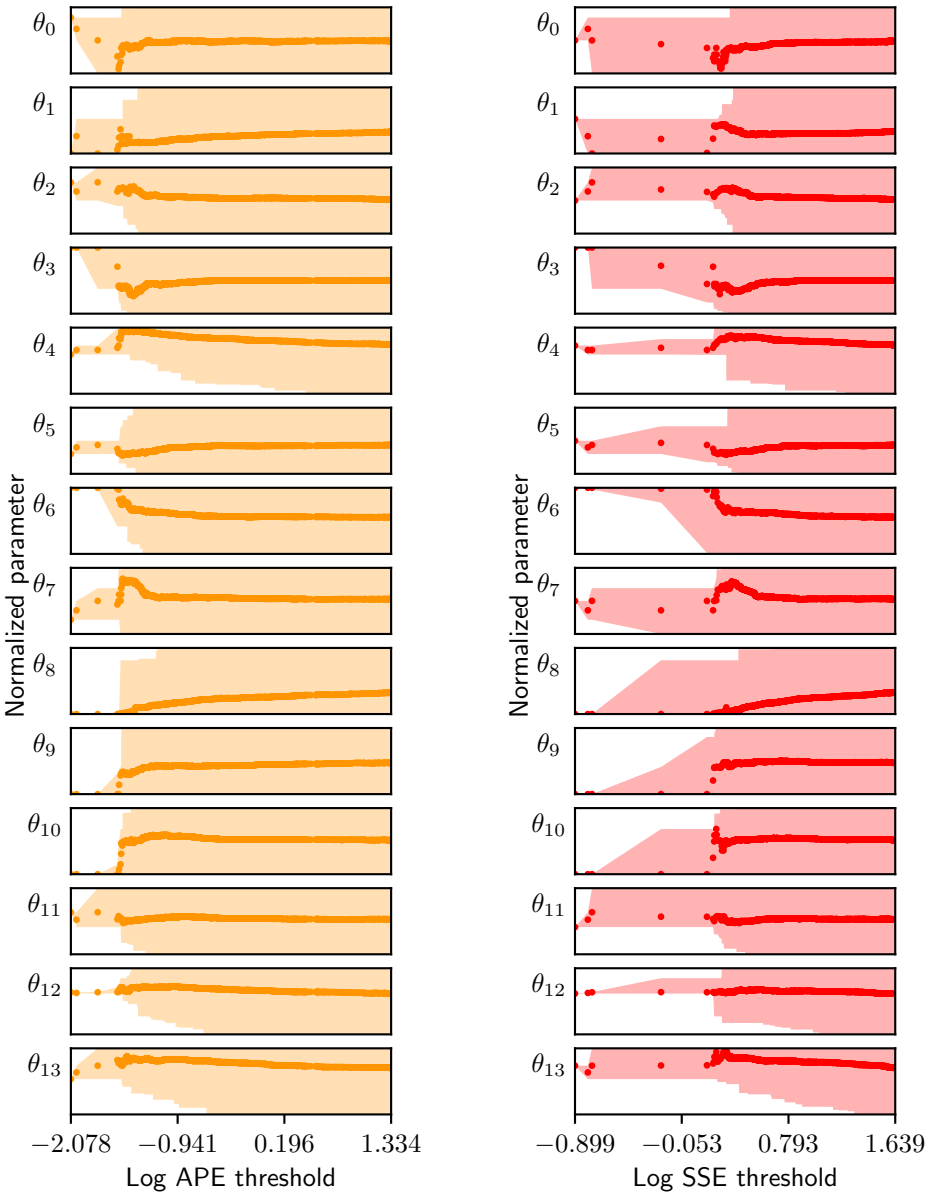
Figure 5.2: Ranges of parameters versus performance seen across all optimization trials on KITTI (continued).



(e) SS-VO city APE

(f) SS-VO city SSE

Figure 5.2: Ranges of parameters versus performance seen across all optimization trials on KITTI (continued).



(g) SS-VO road APE

(h) SS-VO road SSE

Figure 5.2: Ranges of parameters versus performance seen across all optimization trials on KITTI (continued).

5.3.4 IMR Experiments

Informed by the results from the KITTI experiments, we focus on using the UR and BO algorithms for tuning the parameters of the ICP-LO and DPM-VO systems in three different contexts, shown in Fig. 5.3.

- **lab** environment has medium-gloss painted concrete floors, desks, and test equipment tripods
- **tunnel** environment has bare concrete floors, and pipes and sandbags along the walls
- **office** environment has glossy linoleum tile floors and multiple chairs and tables

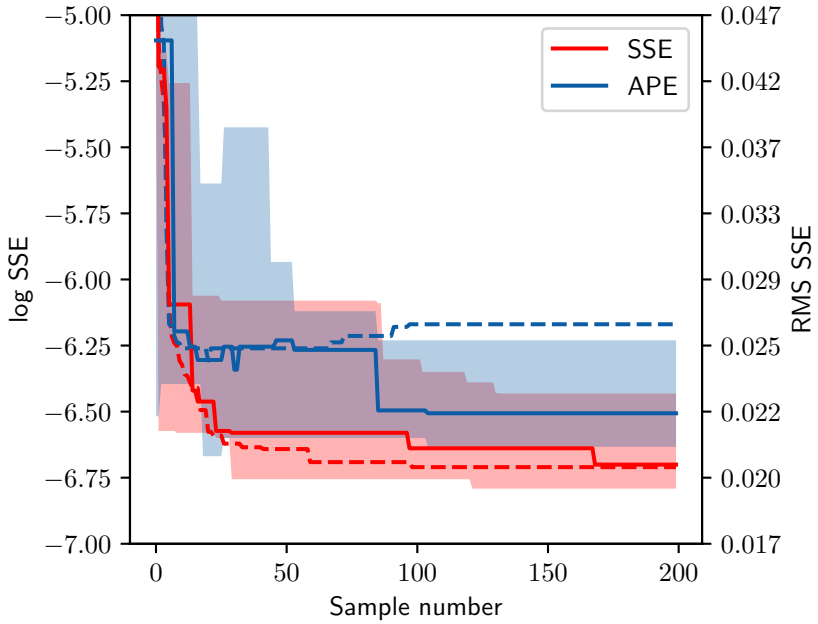
During the experiments we mount our ground truth system cameras along the upper walls and ceiling in the tunnel and office environments so as to not alter the local geometry and affect the laser odometry. The camera tripods serve as part of the local environment in the lab environment.

We use the same system execution procedure in our previous experiments described in Section 4.4.4, but with a trajectory consisting of a forward and backward motion, followed by left and right point turns. This trajectory succinctly tests both the linear and angular tracking performance of the odometry systems and takes on average eight seconds per execution.

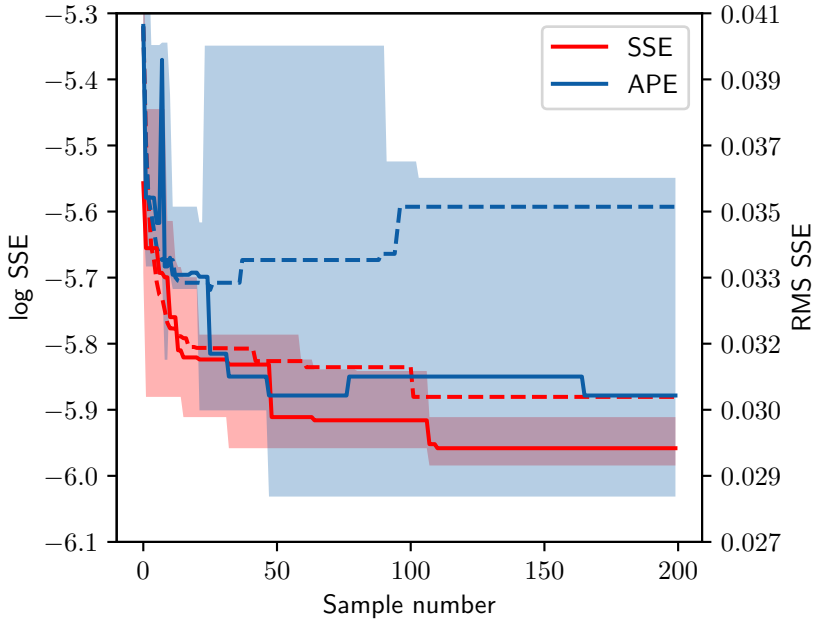
Tuning

We tuned each system 5 times in each context with APE and ground-truth evaluations for a total of 60 BO trials. Each trial lasted 30 minutes, or approximately 200 evaluations. UR was run once in each context for each system with the samples permuted for variance computations. In total the tuning experiments consist of $13,200$ evaluations, or 33 hours of robot runtime.

Best-seen performance traces for BO and UR in each context are shown in Figures 5.4. We again show the distribution of the ten best performing configurations across all trials in each context in Figure 5.5.

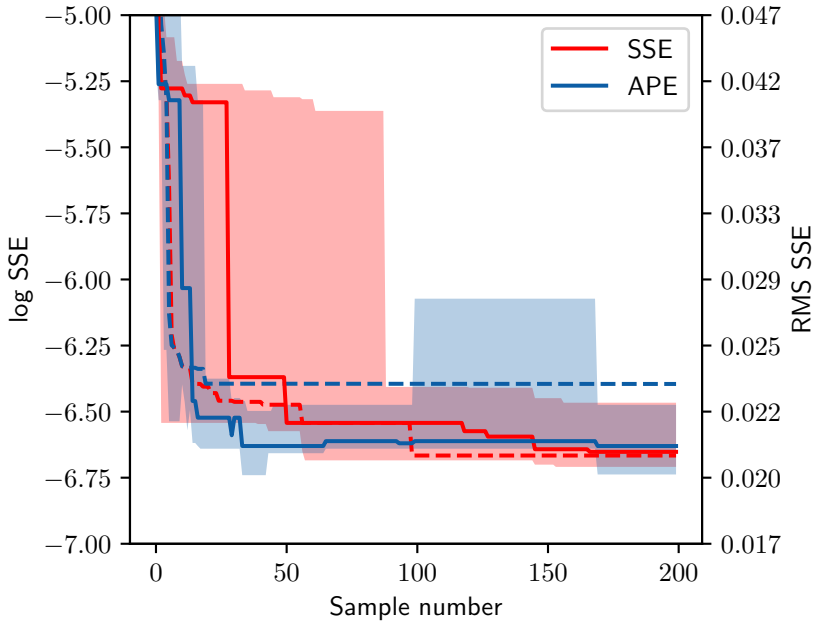


(a) DPM-VO nshhb

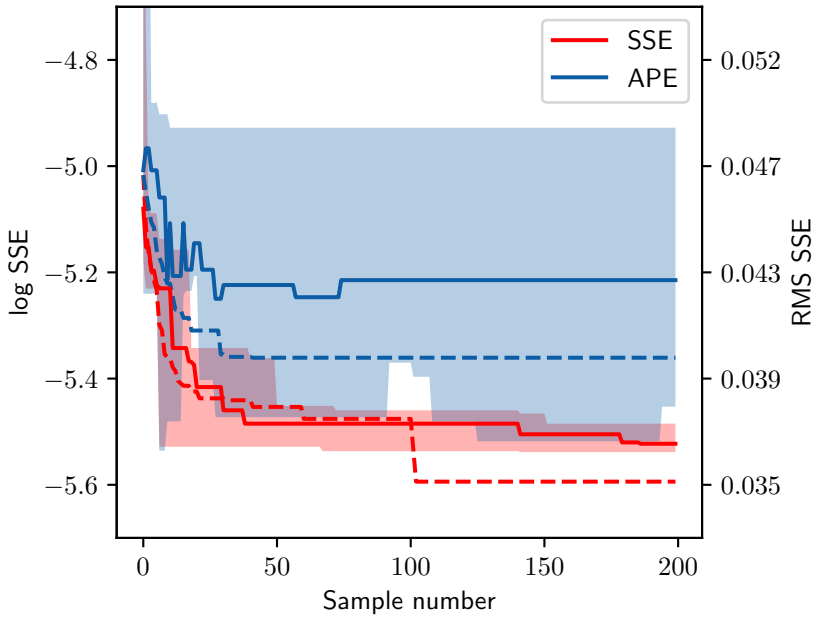


(b) ICP-LO nshhb

Figure 5.4: Best-seen performances on IMR tuning with BO and uniform approaches with approximately 30 minutes of evaluation per trial. Solid lines denote medians and filled regions denote extreme ranges for the 5 BO trials. Dotted lines denote the median over 1000 bootstrap samplings of the uniform evaluations.

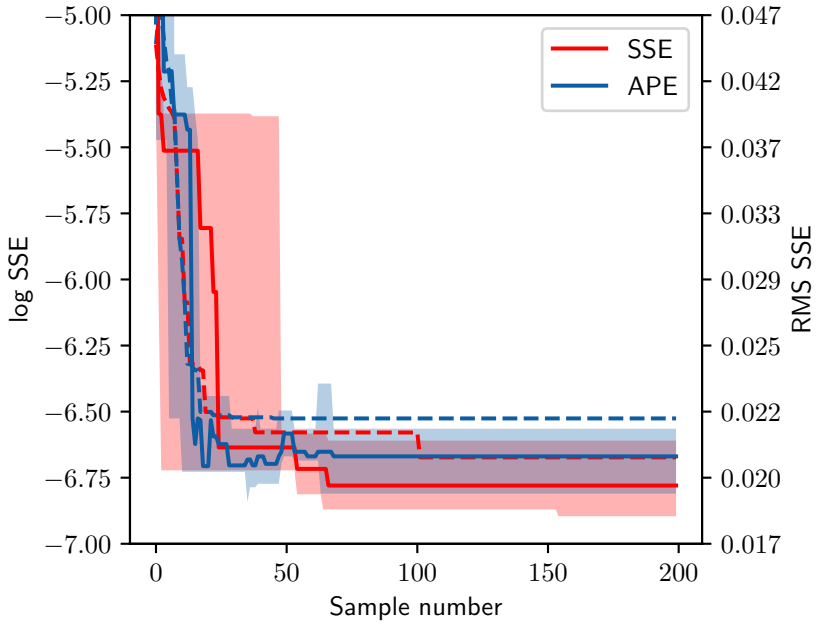


(c) DPM-VO tunnel

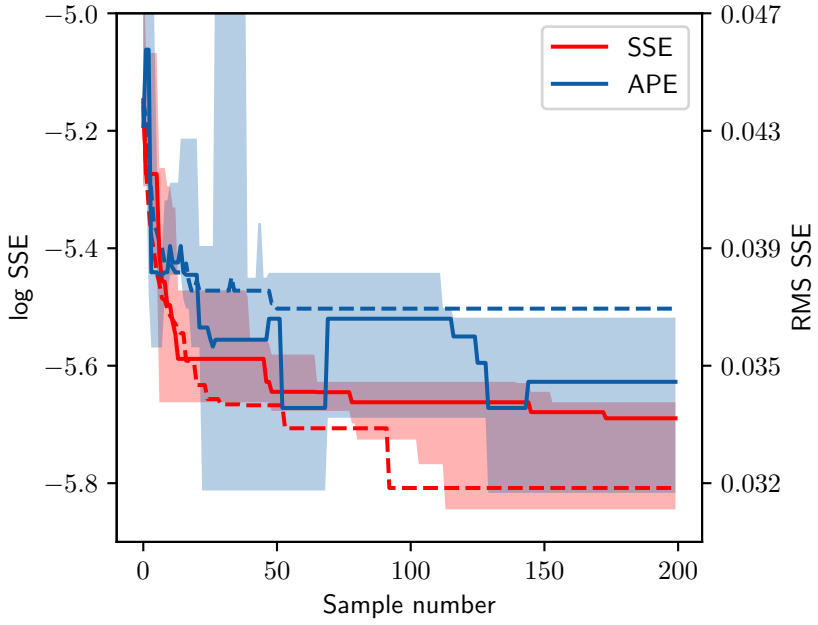


(d) ICP-LO tunnel

Figure 5.4: Best-seen performances on IMR tuning with uniform and BO approaches (continued).

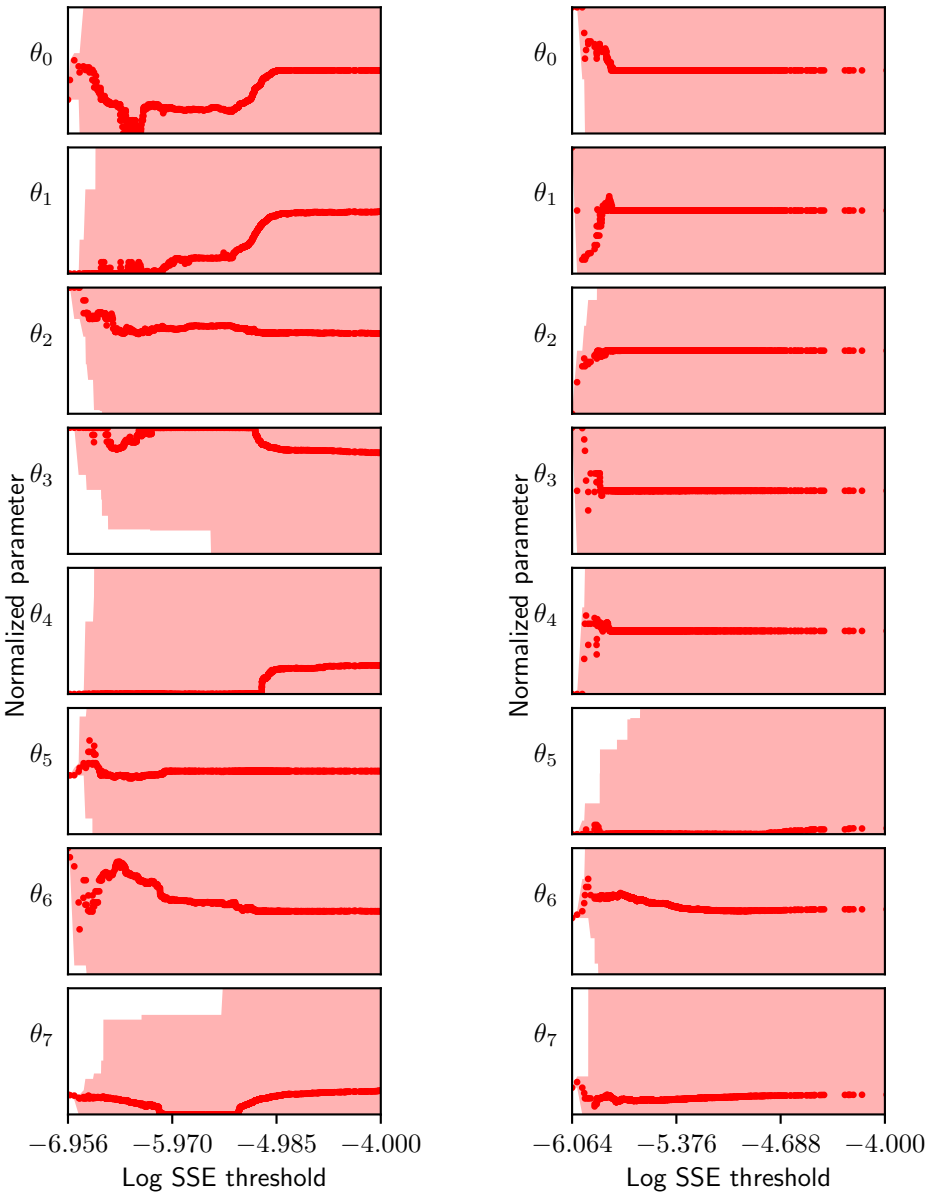


(e) DPM-VO office



(f) ICP-LO office

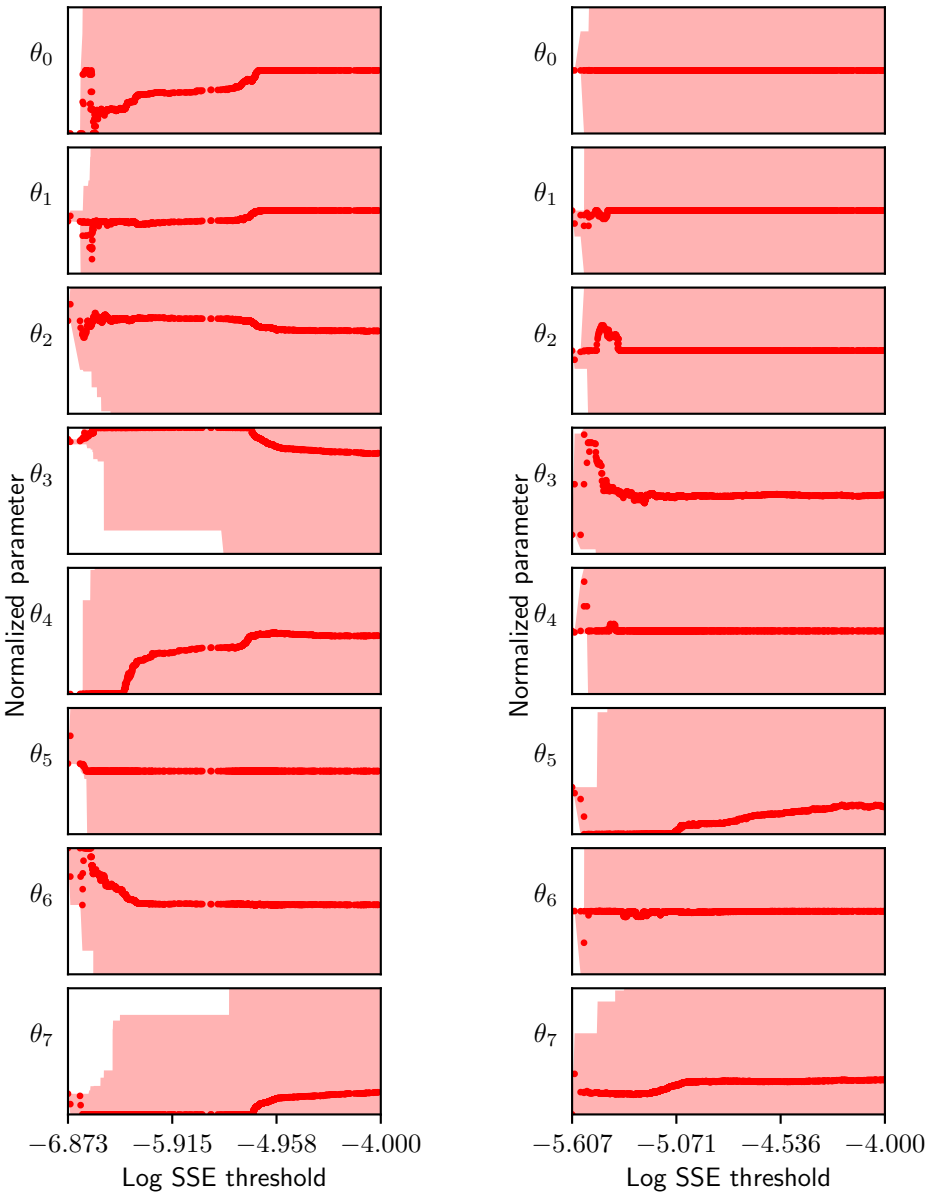
Figure 5.4: Best-seen performances on IMR tuning with uniform and BO approaches (continued).



(a) DPM-VO nshhb SSE

(b) ICP-LO nshhb SSE

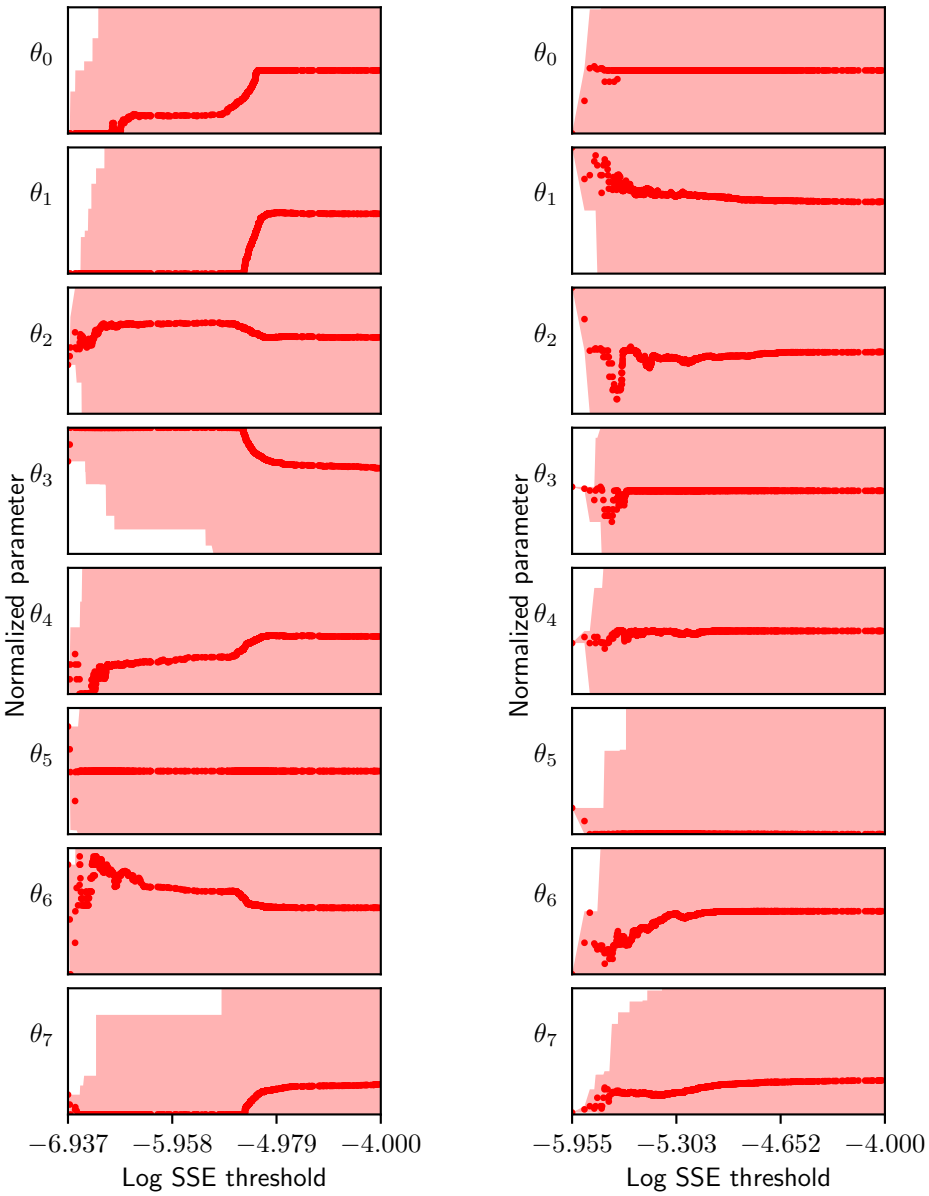
Figure 5.5: Ranges of parameters versus performance thresholds as seen across all optimization trials on IMR. Dots represent the median of the set of parameters performing at least as well as the x-coordinate value, and the shaded area shows the extreme ranges of the parameters. Note that the upper (rightmost) threshold range is reduced from the full dataset to show detail.



(c) DPM-VO tunnel SSE

(d) ICP-LO tunnel SSE

Figure 5.5: Ranges of parameters versus performance thresholds as seen across all optimization trials on KITTI (continued).



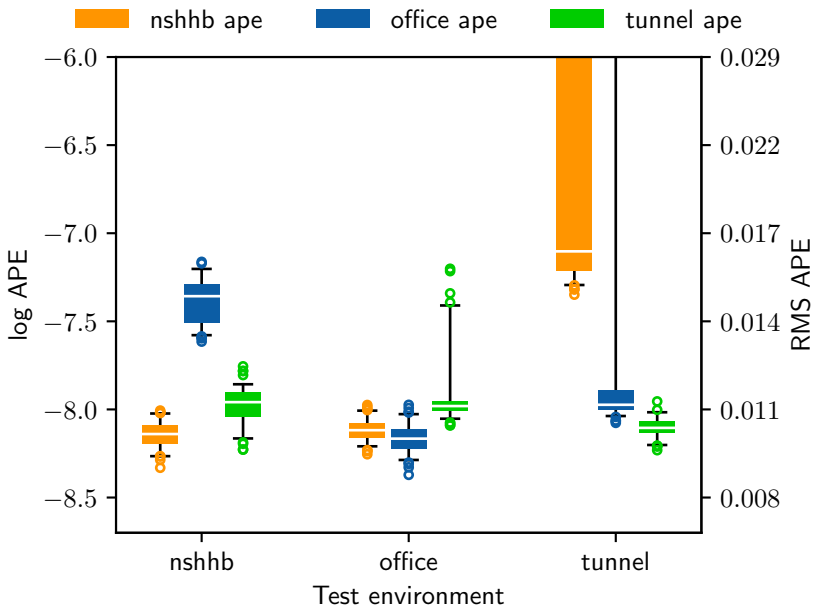
(e) DPM-VO office SSE

(f) ICP-LO office SSE

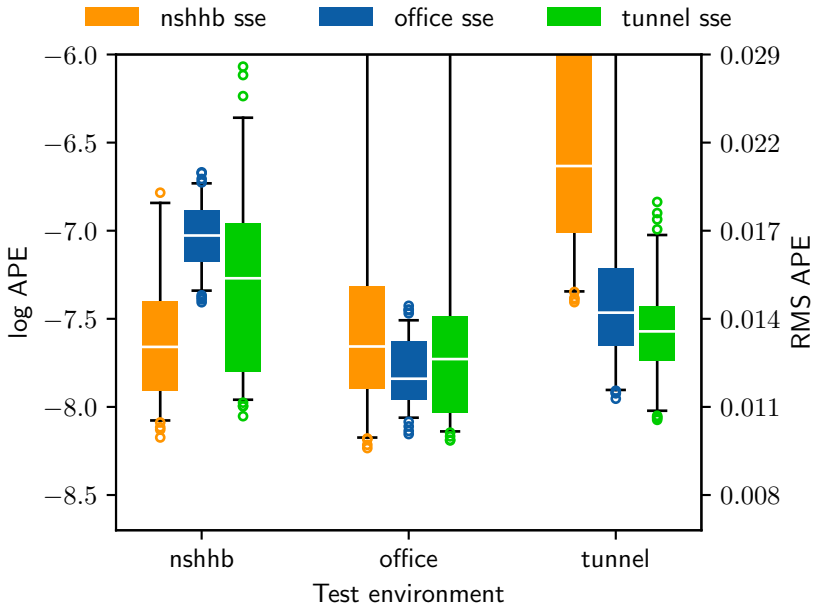
Figure 5.5: Ranges of parameters versus performance thresholds as seen across all optimization trials on KITTI (continued).

Cross-Environment Comparison

To test the benefit of tuning on-site, we evaluated the 3 top-performing configurations from each tuning trial in each environment 5 times each. This means each top configuration is evaluated 15 times each, which comes to a total of $2,700$ comparison evaluations. We report the corresponding RMS for the APE and SSE evaluations across these comparative evaluations in Figures 5.6, 5.7, 5.8, and 5.8.

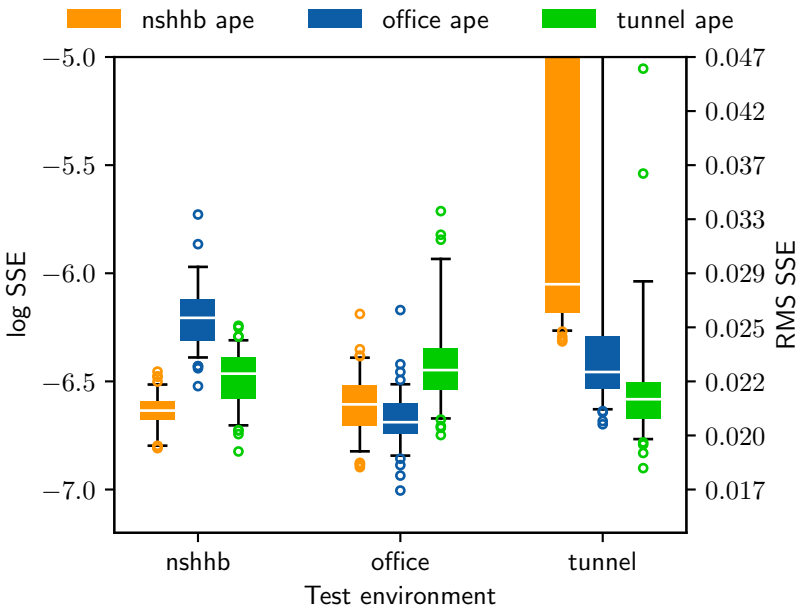


(a) DPM-VO RMS-APE tuned using APE

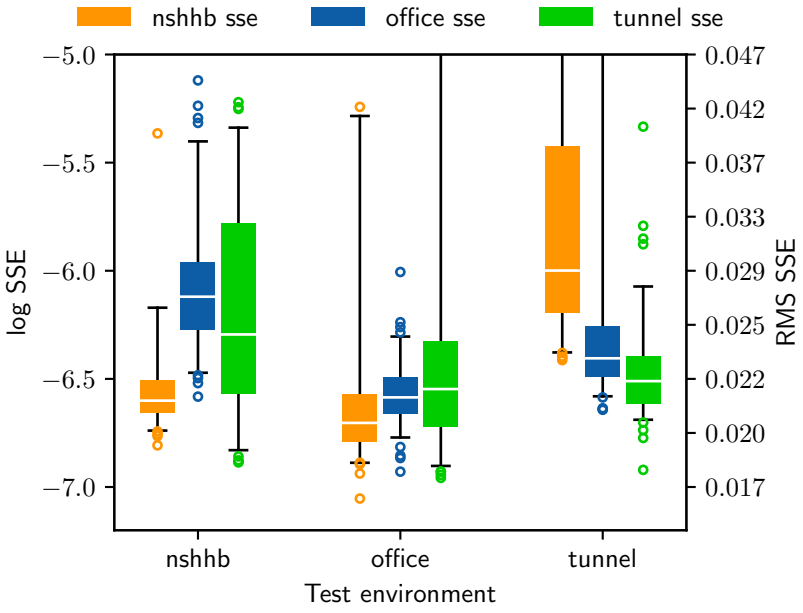


(b) DPM-VO RMS-APE tuned using SSE

Figure 5.6: Box plot of top DPM-VO configurations' RMS-APE in test environments when tuned on training environments indicated by legend. Median is shown by white line, box denotes upper and lower quartiles, and whiskers denote 5th and 95th percentiles with outliers shown as circles.

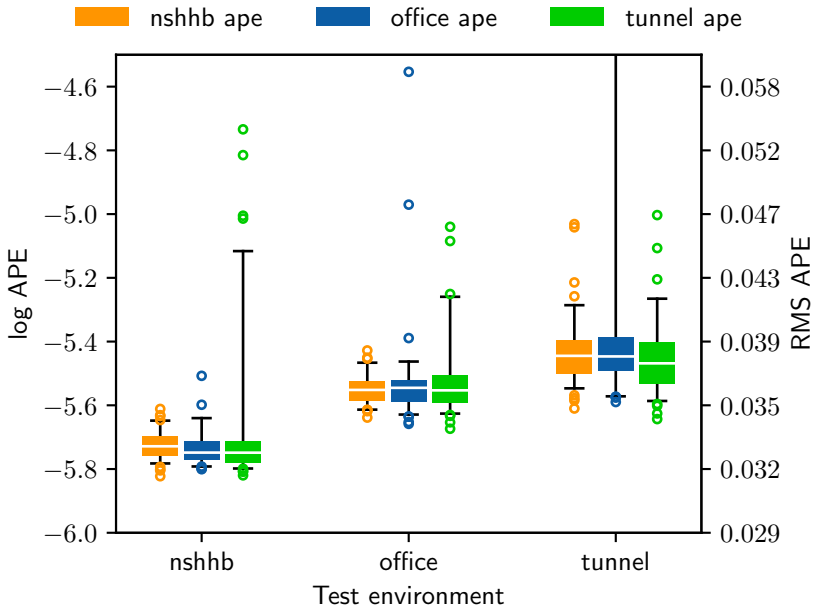


(a) DPM-VO RMS-SSE tuned using APE

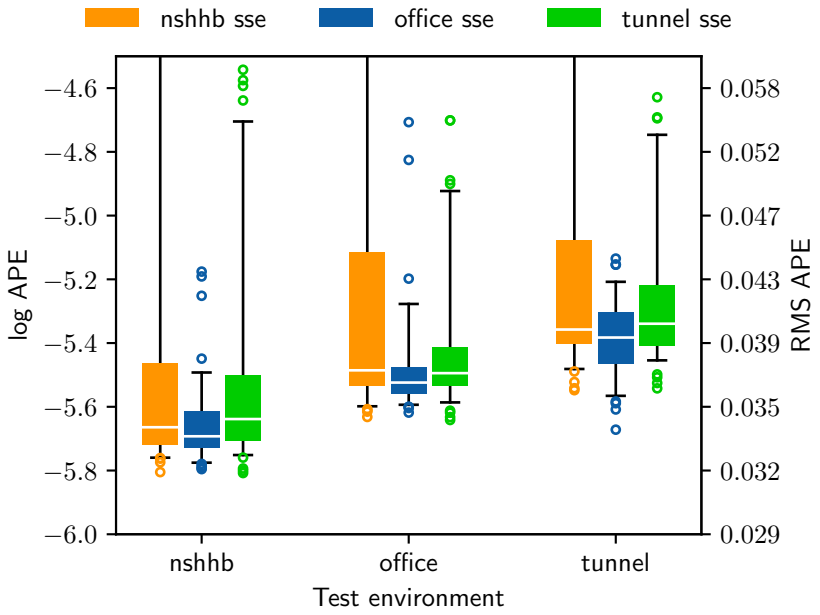


(b) DPM-VO RMS-SSE tuned using SSE

Figure 5.7: Box plot of top DPM-VO configurations' RMS-SSE in test environments when tuned on training environments indicated by legend. Median is shown by white line, box denotes upper and lower quartiles, and whiskers denote 5th and 95th percentiles with outliers shown as circles.

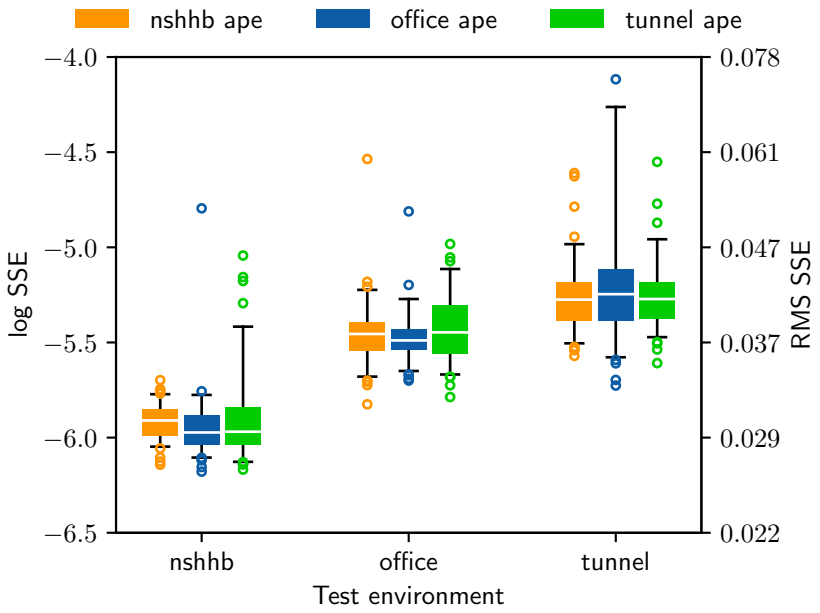


(a) ICP-LO RMS-APE tuned using APE

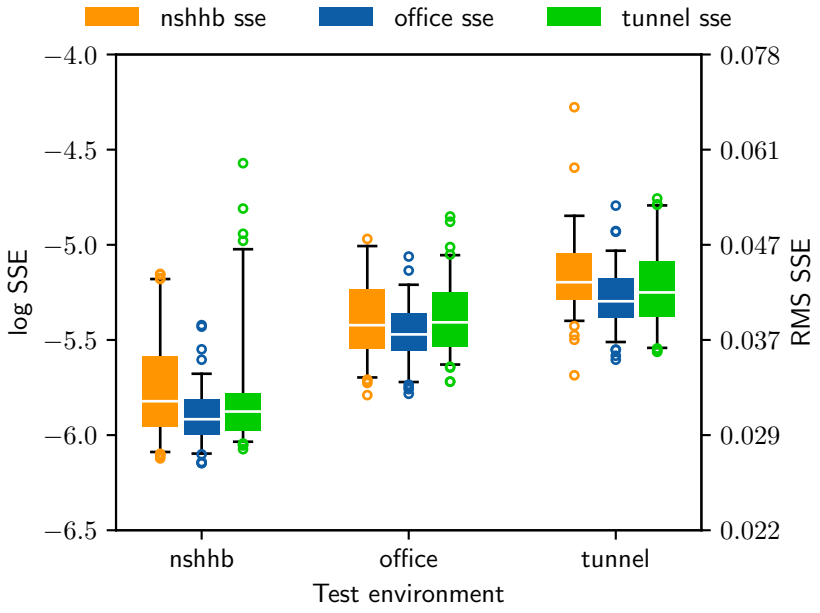


(b) ICP-LO RMS-APE tuned using SSE

Figure 5.8: Box plot of top ICP-LO configurations' RMS-APE in test environments when tuned on training environments indicated by legend. Median is shown by white line, box denotes upper and lower quartiles, and whiskers denote 5th and 95th percentiles with outliers shown as circles.



(a) ICP-LO RMS-SSE tuned using APE



(b) ICP-LO RMS-SSE tuned using SSE

Figure 5.9: Box plot of top ICP-LO configurations' RMS-SSE in test environments when tuned on training environments indicated by legend. Median is shown by white line, box denotes upper and lower quartiles, and whiskers denote 5th and 95th percentiles with outliers shown as circles.

5.3.5 Discussion

Comparison of Optimization Approaches

We discuss first the effectiveness of the various tested optimizers in our experiments. Starting with the KITTI ICP-LO experiments, we see in Figures 5.1a and 5.1c that BO consistently obtains the lowest APE by the end of each trial. CMA performs reasonably well on the ICP-LO city condition, but has very high variance on the more challenging ICP-LO road condition. In both cases, UR performs the worst, but is surprisingly competitive against the focused optimization approaches. We will return to this observation in a later section. BO's superior APE performance translates to SSE as well, across all trials in Figure 5.1b and on average in 5.1d. The better lower-bound SSE performance of CMA on the ICP-LO road condition does not originate from a lower APE, and as such can be attributed to either SSE variance or a disparity between the APE and SSE.

Performance is mixed for the SS-VO conditions. Looking at the APE for the SS-VO city condition in Figure 5.1e, we see that CMA performs best, followed by UR, and finally BO. The corresponding SSE in Figure 5.1f show UR performing best, however, with BO exhibiting large variance between UR and CMA. However, the magnitude of this variance (± 0.2 SSE and ± 0.1 APE) are fairly small already. Improving this result likely requires improving our performance evaluation approach. On the SS-VO road condition, seen in Figures 5.1g and 5.1h, BO outperforms the other two methods on average, but exhibits high variance, while CMA shows very low variance. This may suggest that the objective function had large local minima basins that trapped CMA, whereas the more aggressive BO was able to wander into better minima on 2 out of the 3 trials.

Since these initial results suggest that BO is more effective than CMA in some cases, and not much worse in others, we tested only UR and BO on the IMR system, but consider using both APE and SSE evaluations to guide the search. The resulting SSE performances on the DPM-VO conditions, seen in Figures 5.4a, 5.4c, and 5.4e, show that running BO with APE evaluations is competitive against running BO with SSE. We note also that UR with APE performs poorly in these tests, while UR with SSE performs surprisingly well, but defer discussion of this point to a later section.

BO with APE performs less favorably on the IMR ICP-LO conditions, with high variance in the nshhb and office conditions shown in Figures 5.4b and 5.4f, and a poor median in the tunnel condition shown in Figure 5.4d. UR with APE performs particularly poorly in the ICP-LO nshhb condition, suggesting that these issues may be due to poor APE-SSE parity. However, on the ICP-LO office condition, UR with APE is relatively monotonic compared to BO with APE, while UR with SSE outperforms BO with SSE. This suggests that the BO search tends to get trapped in local minima for this test condition, regardless of the evaluation approach, resulting in high variance. This may also explain the very large variance of BO with APE on the ICP-LO tunnel condition, which overall exhibits similar traits as the ICP-LO office results.

Overall we see that in many cases using BO with APE evaluations can reliably find high-performing configurations. In some other cases we note that there are issues with the APE-SSE parity, and in yet others we see that BO gets trapped in poor local minima. We believe that both of these issues can be addressed in future work.

Effective Difficulty of Parameter Tuning

We revisit our earlier observation of UR performing competitively against both CMA and BO in many of our experiments, despite the dimensionality being large (> 6) and the number of evaluations being low ($100 - 200$). One possible explanation is that shape of the objective functions does not sufficiently fulfill the assumptions used by CMA and BO, resulting in suboptimal performance. However, this seems unlikely, as we would expect this to result in high variance, but CMA and BO are mostly consistent across our experiments.

Instead, we believe this suggests that the effective dimensionality of the tuning optimization task is small. In other words, the number of important parameters for tuning is much less than the total number, such that UR is able to relatively densely sample the entire configuration space. Looking at the spread of ICP-LO parameters from the KITTI experiments in Figures 5.1a and 5.1c, we see that θ_2 and θ_3 exhibit a nearly full spread even at the lowest APEs, while θ_1, θ_5, and θ_6 are spread over half the normalized range. Only θ_0 and θ_4 show aggressively decreasing spreads resulting in effectively a point at an extreme value. As such, UR is actually sampling in an effectively 2 or 3 dimensional space.

On the SS-VO conditions in Figure 5.2e and 5.2g, most parameter ranges narrow slowly with decreasing APE. The final dramatic narrowing in Figure 5.2g occurs due to a few exceptional configurations found by BO in one trial. Since there are so few samples at this performance level in this condition, we cannot draw any strong conclusions about the parameter fineness here. Nonetheless, the results from the SS-VO city condition supports the hypothesis that the effective dimensionality for achieving near-optimal performance is small.

We note that it may be possible to discover and take advantage of this reduced dimensionality for faster and more robust optimization. If we believe that the low effective dimensionality arises from individual parameters in an independent way, we could apply the work of Kandasamy et al. [2015] on additive GP kernels for finding these subspaces. Alternatively the work of Djolonga et al. [2013] proposes to more generally discover the objective subspace.

Importance of Refinement to APE-based Tuning

We noted earlier the particularly poor performance of UR with APE on the IMR ICP-LO nshhb condition, and to a lesser degree on the DPM-VO nshhb condition as well. In these test conditions, the SSE of the best-seen configuration increases with more samples, suggesting that the APE-SSE parity is particularly poor in a few instances. This does not seem to affect the BO with APE

approaches as much, however, which still manage to perform competitively in these conditions, though with increased variance. One hypothesis is that parity may be negatively correlated with APE performance, such that lower APEs generally have better parity. As such, actively optimizing the APE, for instance with BO, leads us to better APES with better corresponding SSE bounds.

As an aside, we note that the process of discovering low parity configurations in the course of optimization proivdes invaluable feedback for discovering sources of overconfidence and refining the perception system itself. For instance, while tuning the ICP-LO system in hallways, we discovered that certain parallel degenerate geometries were resulting in the system becoming highly confident that the robot was not moving. Similarly, we discovered a variety of boundary cases when dealing with empty disparity images on our DS-VO system.

Benefits of Context Adaptation

We now turn our attention to the results from the cross-environment IMR experiments, starting by comparing tuning using APE versus SSE. In Figures 5.6a and 5.6b, we see that tuning the DPM-VO system using APE results in considerably lower APE losses across all conditions, and also with lower variance across trials. This same trend is true when looking at the SSE losses in Figures 5.7a and 5.7b for the `nshhb` and `office` environments, but not significantly so for the `tunnel` environment. This suggests that optimizing the APE gives more consistently performant configurations, possibly due to the issue of serendipity discussed in Chapter 4.4.6. The same trends can also be seen for the results on the ICP-LO system, where tuning with APE gives better APE performance, as seen in Figures 5.8a and 5.8b, as well as SSE performance, as seen in Figures 5.9a and 5.9b.

Looking across conditions, starting with the DPM-VO system tuned using APE seen in Figure 5.6a, we see that tuning and testing on the same environment gives the best APE performance. In some cases, such as tuning on `nshhb` and testing on `office`, the benefit is not significant, but the worst cases, such as tuning on `nshhb` and testing on `tunnel`, are dramatic. The same is true for the SSE performances reported in Figure 5.7a. This shows that adapting to context on-site can allow systems to avoid potentially catastrophic worst-case scenarios.

For the results on the ICP-LO system, however, we see in Figures 5.8a and 5.9a that there is relatively little difference in APE and SSE performance when tuning and testing across environments, aside from slightly reduced variance. This may be because the environments we tested in did not exhibit significant contextual variation for the ICP-LO system, whereas the surface texture variations across environments were important for the DPM-VO system. As such, the ICP-LO system can be considered relatively context-free in this deployment, but nonetheless benefits slightly from adapting on-site.

It is interesting to note that the benefits of tuning and testing on the same environment are not as strong when tuning with SSE, and in some cases, worse than tuning on a different environment. This can be seen in the SSEs for the DPM-VO system shown in Figure 5.7b, where tuning on `nshhb` and testing on

`office` gives a lower median SSE than tuning on `office`. Similarly for the ICP-LO system, in Figure 5.9b tuning on `office` and testing on `tunnel` is better than tuning on `tunnel`. This may be due to the difficulty of optimizing the SSE, as seen in our tuning experiments, as well as serendipity resulting in higher variance configurations being returned.

(a) lab

(b) tunnel

(c) office

Figure 5.3: The three test environments used in the IMR experiments.

Chapter 6

Predicting Perceptual Failures

Previously we presented parameter tuning as a mechanism for adapting to contextual variations after deployment. However, even perfect parameter tuning does not completely eliminate the possibility of perception failure, as some situations may simply be outside the operating capabilities of a system. For instance, no amount of tuning will allow a visual odometry system to estimate motion when there is no scene appearance texture. The best recourse in these situations may be to simply avoid the situation in the first place, or as the old adage goes, "an ounce of prevention is worth a pound (16 ounces) of cure."

To predict failure, we must first understand the complex interactions between perception systems and the context. However, modeling this relation is complicated by the wide variety of environment and system-dependent factors, as well as stochasticity in the system behavior itself. As such, elegant analytical approaches are only possible when deep prior knowledge is available. For instance, an engineer may understand that glare strongly affects a particular visual odometry system, and can thus design an algorithm to predict its effects. Unfortunately, the most critical contextual effects are often, by nature, the ones not anticipated by the system designers.

Instead of relying on hand-engineered rules and heuristics, we propose that systems should learn performance models from their own experiences. This does not wholly eliminate failures, but instead allows systems to avoid repeating past failures. In particular, we are interested in predicting performance directly from sensor data to make our overall approach general and easy to apply.

To be practical in a robotics setting we need a learning approach that demands relatively little computational and data resources during deployment. One approach is to aggregate large amounts of data from different deployments into a single general model. However, it is difficult to adapt such complex models on-site in the event that the system is deployed to an environment not in the training data. Instead, we opt for a local learning approach which focuses on

using data collected on-site to learn a less general, but more specialized model.

In this chapter, we first discuss relevant prior work on this topic before presenting our formulation. We then discuss three possible modeling approches in depth and present comparisons through experiments on the IMR system. Our experiments demonstrate the viability of our modeling approach in general for predicting perceptual failure.

6.1 Related Works

Our approach to performance prediction draws upon prior work in introspection, local learning, and density estimation. We cover relevant works from these fields below.

6.1.1 Introspection

Recently the term *introspection* has come to refer to predicting when a perception system will fail. The majority of approaches in this vein utilize machine learning techniques to generalize past experiences of failure. Zhang et al. [2014] demonstrated predicting failure in segmentation, classification, and other algorithms directly from image content. More recently this was extended by Saxena et al. [2017] to predicting whether a monocular depth estimation algorithm will fail using a deep neural network. Both of these approaches relied on large amounts of labeled data and offline training, though the work in Saxena et al. [2017] uses a self-supervised approach. Another highly relevant work is that of Gurau et al. [2017] on learning the local performance of a pedestrian detector in different outdoor locations. The authors propose a system that aggregates experiences within locations, and show that separating experiences based on location appearance changes improves prediction quality. Other works explored location Churchill et al. [2015] and appearance Dequaire et al. [2016] as context for predicting the heuristic performance of a visual odometry system.

Distinct from these learning-based approaches is the work of Grimmett et al. [2016], which proposes analytical methods for determining when queries to a classifiers are well-constrained by the training data. This allows a system to reason about potential failures without needing to experience them.

6.1.2 Local and Online Learning

Some of the earliest work on local and online learning for robots was that of Wellington and Stentz [2006], who proposed near-far learning for long-range terrain classification. In this framework, features of distant terrain are later associated with accurate near predictions to gather examples without supervision. A similar approach has been used for learning online to detect obstacles at long range from appearance, with an online non-parametric distribution used by Happold et al. [2006] and a neural network by Hadsell et al. [2009]. More

recently, Hawke et al. [2016] used an unsupervised learning approach to improve the performance of a pedestrian detector over multiple deployments.

6.1.3 Density Estimation

There is a broad body of literature on the topic of density estimation, or predicting the value of a probability density function. In general these approaches can be classified as non-parametric or parametric in nature.

We begin with the non-parametric approaches, of which the most well known is the Parzen-Rosenblatt Window approach, developed independently by Rosenblatt [1956] and Parzen [1962]. This approach is now more commonly referred to as kernel density estimation (KDE), referring broadly to techniques that represent a probability density function with a set of kernel basis functions. KDE has been applied to a number of tasks, for instance, Yeung and Chow [2002] use an estimate of normal network behavior density to classify unlikely behaviors as intrusions. Gerber [2014] use KDE to aggregate Twitter data for crime prediction. Elgammal et al. [2002] use KDE to model the distribution of image pixel intensities for segmenting foreground and background.

Aside from KDE are the closely related nearest neighbors approaches, which have been studied alongside KDE for many decades Mack and Rosenblatt [1979]. Nearest neighbors approaches model probability density with the empirical density of training data. These approaches have been explored for image classification by Boiman et al. [2008] and for KL-divergence computation using sampling by Wang et al. [2009b]. Related is work by Wu et al. [2014] using random-forests for density estimation to rapidly classify streams of data.

Finally there are the histogram-based representations of density, such as that used by Happold et al. [2006] for modeling the conditional relation between image features and traversability. Histogram representations can have issues with discretization errors, but can still find use due to their computational efficiency.

The other class of density estimation approaches are the parametric approaches. These methods fit parametric density functions to data, and are related to system identification approaches, such as that of Ghahramani and Roweis [1998]. For instance, Stauffer and Grimson [1999] model the distribution of image pixel intensities for background segmentation like Elgammal et al. [2002], but with a Gaussian mixture model (GMM) instead. Singh et al. [2010] use GMMs to model the multi-model distribution of loads in a power distribution network, and Laxhammar et al. [2009] compare GMMs to KDE for detecting anomalous marine vessel movements and find that the two are comparable for their setting.

6.2 Failure Prediction as Density Estimation

Before we can discuss predictin failure, we must define what it means for a perception system to fail. Our statistical formulation of perception already suggests

a straightforward and intuitive definition of failure as performance exceeding an application-specific threshold λ.

In what situation, though, will we be predicting if failure is likely? Recall that our setting is a robot operating autonomously in a deployment, and that our goal is to avoid situations which may result in failures. Thus, the latent x is not available, though the estimate $\hat{x}$ is. If we assume that this failure predictor will be learned after tuning is done, then we know that the configuration θ will be fixed. Finally, recall that in Chapter 3 we abstained from choosing a representation for context due to its complexity. Thus, it seems that the best we can do is attempt to predict whether failure will occur over all latents x and contexts ϕ given the estimate $\hat{x}$ and observation z. We argue, this is an ideal setup, as it involves marginalizing out factors beyond the perception system's control while using all tangible observed quantities.

Having established a setup, we must decide on an overall approach. It is tempting to treat this task as classification, where we collect many examples of whether $\rho_i > \lambda$ along with corresponding estimate $\hat{x}_i$ and observation z_i to learn a classifier. However, this approach has two issues.

First, the stochasticity in mapping from latent x and context ϕ to observation z, estimate $\hat{x}$, and finally performance ρ may not be homoscedastic, but rather, heteroscedastic. In other words, the variation in observed performances ρ may vary across the latent and context spaces. For instance, a visual odometry system may perform reliably in a well textured environment, but erratically when lighting conditions are poor. Most standard classification and regression approaches assume homoscedasticity and can perform poorly when learning on heteroscedastic data Kersting et al. [2007].

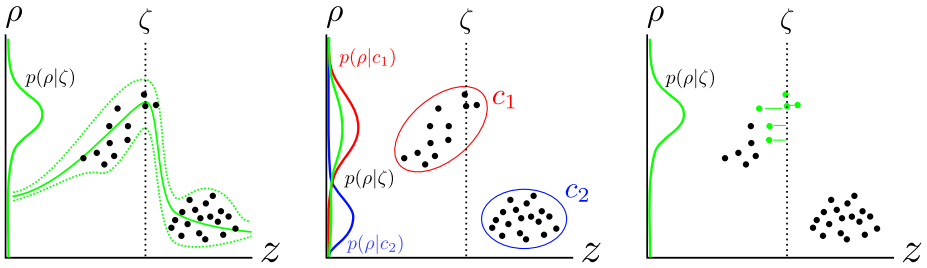
Second, observing only whether the system fails or not discards useful information in the scale of the losses incurred, and can also make the modeling task difficult when losses are near the threshold. With these considerations about stochasticity and information in mind, we propose to predict failure by modeling the distribution of losses given the observation and estimate, or $p(\rho|z, \hat{x})$.

6.2.1 Learning a Density Estimate

Formally, we denote the distribution model as $\hat{p} : \mathcal{Z} \mapsto \mathcal{S}$, where $\mathcal{S}$ is the space of probability distributions over real-valued scalars $\mathcal{R}$. Given examples of actual observations z_i and estimates $\hat{x}_i$ and their corresponding measured performances ρ_i, we can quantify the quality of a distribution model by its log-likelihood for the observed data:

$$\mathcal{L}(\hat{p}) = \sum_{i=1}^{N} \log\left(\hat{p}(\rho|z_i, \hat{x}_i)\right) \tag{6.1}$$

When this criteria is used to fit or select models, the procedure is commonly referred to as *maximum likelihood estimation*, or MLE. If prior information about the model is integrated by modifying the log-likelihood function, then it is a *maximum a posteriori* or MAP approach. In this work we consider only a MLE formulation and leave MAP extensions as future work.



(a) Parametric model (b) Mixture model (c) Non-parametric model

Figure 6.1: The three modeling approaches tested in this work. Training data are shown as black dots, a query observation ζ is indicated with a dotted line, and the predicted performance distribution $p(\rho|\zeta)$ is shown along the vertical axis in green. (a) Predicting the performance mean and variance as a function of observation ζ (b) Mixture representation with components c_1 and c_2 predicting performance by conditioning on ζ (c) Computing performance statistics from k-closest training instances.

6.2.2 Distribution Modeling Approaches

We describe here three approaches for modeling distributions using data. It is important to note that since we do not have ground truth for the conditional, we cannot simply formulate this as a distribution regression task. Rather, as previously mentioned, our data is pairs of received observations and resulting performances which a model must "explain".

Parametric Conditional Distribution

Parametric approaches use a function f_υ defined by parameters $\upsilon \in \Upsilon$ to predict a representation of $\hat{p}(\rho|z)$ given the observation z as input. The advantage of a parametric approach is that we can efficiently optimize the function parameters by using the gradient of the predictor log-likelihood with respect to the parameters:

$$\frac{\partial}{\partial \upsilon}\mathcal{L}(\hat{p}_\upsilon) = \sum_{i=1}^{N} \frac{\partial}{\partial \upsilon} \log\left(\hat{p}_\upsilon(\rho_i|z_i)\right) \tag{6.2}$$

$$= \sum_{i=1}^{N} \frac{\hat{p}'_\upsilon(\rho_i|z_i)}{\hat{p}_\upsilon(\rho_i|z_i)} \tag{6.3}$$

where $\hat{p}_\upsilon(\rho, z)$ denotes the probability of ρ under the distribution predicted by f_υ with input z, and $\hat{p}'_\upsilon(\rho, z)$ denotes the gradient of the probability with respect to the model parameters υ.

For practicality, we will restrict $\hat{p}$ in this approach to univariate Gaussian distributions. Since Gaussians can be fully parameterized by their mean and

standard deviation, our parametric function can be any two-dimensional multivariate function $f_v : \mathcal{Z} \mapsto \mathcal{R}^2$ that outputs the performance mean and log-standard deviation. An example of this is shown in Fig. 6.1a

The parametric function class or form can greatly affect the performance of the approach. It is also common to introduce regularization in the form of a penalty on the parameters themselves to aide generalization. Typically, the more powerful the function class, the more parameters it has and the more regularization is required. This has implications for the optimization complexity and overall learning efficiency and reliability. In our experiments, we use a relatively small artificial neural network (ANN) which is described in Sec. 6.3.2.

Parametric Joint Distribution

Mixture modeling approaches model the joint distribution $p(z, \rho)$ with a finite combination of independent distributions, referred to as the "mixture components". Data (z, ρ) is assumed to be generated by sampling one of the components, and then sampling from that component distribution.

A commonly used mixture model is the Gaussian mixture model (GMMs), shown in Fig. 6.1b, where each component is a multivariate Gaussian. Let c_i be a boolean variable that indicates if observation z and performance ρ was generated by component i. The joint distribution for k components is given by:

$$\hat{p}(z, \rho) = \sum_{i=1}^{k} \hat{p}(c_i) \mathcal{N}\left(\begin{bmatrix} z \\ \rho \end{bmatrix} \middle| \mu_i, \Sigma_i \right) \tag{6.4}$$

$$\mu_i = \begin{bmatrix} \mu_i^z \\ \mu_i^\rho \end{bmatrix} \tag{6.5}$$

$$\Sigma_i = \left[\begin{array}{c|c} \Sigma_i^{zz} & \Sigma_i^{z\rho} \\ \hline \Sigma_i^{\rho z} & \sigma_i^{\rho\rho} \end{array} \right] \tag{6.6}$$

where $[z, \rho]$ is the observation z and performance ρ concatenated into a single vector. The mixture weight term $\hat{p}(c_i)$ describes the prevalence of component i, and μ_i and Σ_i are the mean and covariance for component i, respectively.

To use a GMM for predicting the performance given an observation z we condition on the observation z for each mixture component. This results in another GMM:

$$\hat{p}(\rho|z) = \sum_{i=1}^{N} \hat{p}(c_i|z) \hat{p}(\rho|z, c_i) \tag{6.7}$$

Inferring the conditional probability of each component $\hat{p}(c_i|z)$ is a straightforward application of Bayes' rule:

$$\hat{p}(c_i|z) = \frac{\hat{p}(c_i)\hat{p}(z|c_i)}{\hat{p}(z)} = \frac{\hat{p}(c_i)\hat{p}(z|c_i)}{\sum_{j=1}^{N} \hat{p}(c_j)\hat{p}(z|c_j)} \tag{6.8}$$

We know from properties of Gaussian distributions that the conditional distribution $\hat{p}(\rho|z, c_i)$ is also a Gaussian and can be computed in closed form:

$$\hat{p}(\rho|z, c_i) = \mathcal{N}(\rho|\tilde{\mu}_i^\rho, \tilde{\sigma}_i^\rho) \tag{6.9}$$

$$\tilde{\mu}_i^\rho = \mu_i^\rho + \Sigma_i^{\rho z} \left(\Sigma_i^{zz}\right)^{-1} (z - \mu_i^z) \tag{6.10}$$

$$\tilde{\sigma}_i^\rho = \sigma_i^\rho - \Sigma_i^{\rho z} \left(\Sigma_i^{zz}\right)^{-1} \Sigma_i^{z\rho} \tag{6.11}$$

Mixture models are typically learned using the expectation maximization (EM) algorithm, interleaving optimizing mixture component parameters and membership probabilities. Like gradient-based techniques, EM approaches are only guaranteed to converge to a local minima while being relatively computationally efficient.

Model selection for GMMs consists of choosing an appropriate number of components k. In general, fewer components corresponds to greater regularization. Fixed offsets can also be added to the diagonals of the component covariances Σ_i to prevent degenerate components. Fitting can also be continued from a "warm starts" as data is received in an online setting.

Nonparametric Lookup

Nonparametric approaches model functional relationships by utilizing relevant data examples. By relying on data directly, non-parametric approaches can represent arbitrarily complex distributions. Examples of common non-parametric approaches include k nearest neighbors (kNN), Parzen windows, and Gaussian processes. In this work we consider a kNN approach, illustrated in Fig. 6.1c, due to its flexibility, simplicity, and computational efficiency.

Non-parametric approaches such as kNNs or histograms are often used to model distributions empirically. A common approach is to use kNN to retrieve the k closest seen observations z_i to a query z, and then retrieve their corresponding performances ρ_i. Since the training data are drawn from the same distribution that generated the query, with a sufficiently dense training dataset such that $z_i \approx z$, we can assume that $\rho_i \sim p(\rho|z)$. Accordingly, we can use the corresponding performances ρ_i to estimate the conditional distribution $p(\rho|z)$. In this work we approximate the conditional as a univariate Gaussian and thus compute the empirical mean $\bar{\mu}$ and standard deviation $\bar{\sigma}$ over the ρ_i:

$$\hat{p}(\rho|z) = \mathcal{N}(\rho|\bar{\mu}, \bar{\sigma}) \tag{6.12}$$

$$\bar{\mu} = \frac{1}{k} \sum_{i=1}^{k} \rho_i \tag{6.13}$$

$$\bar{\sigma}^2 = \frac{1}{k-1} \sum_{i=1}^{k} (\rho_i - \bar{\mu})^2 \tag{6.14}$$

If ρ is bounded, another approach is to bin the ρ_i into a histogram and use it as a discrete distribution.

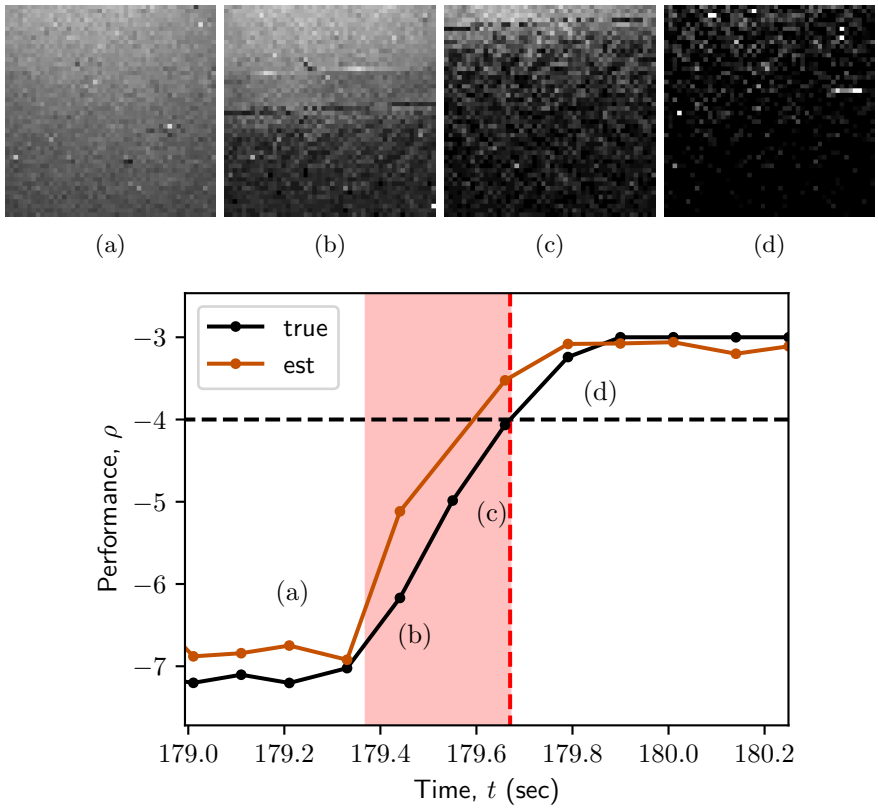


Figure 6.2: Example showing GMM prediction on DPM-VO dataset. The shaded region denotes one standard deviation of predicted performance, the black line shows the observed instantaneous performance, the red dotted line indicates the failure event, and the shaded red area shows 0.3 secs before the event.

The quality of a nonparametric model depends on the training data and distance metric used to define data similarity. Without sufficiently dense training data, the retrieved ρ_i may not be representative of the conditional at the query. A similar issue arises when the distance metric does not describe locality well.

Nonparametric methods vary in their computational and memory requirements. kNN is typically implemented with an efficient tree structure over the training data that allows lookups to be performed in logarithmic time with respect to the quantity of data. However, this implementation of kNN requires all of the training data to be stored, which can be an intensive memory requirement.

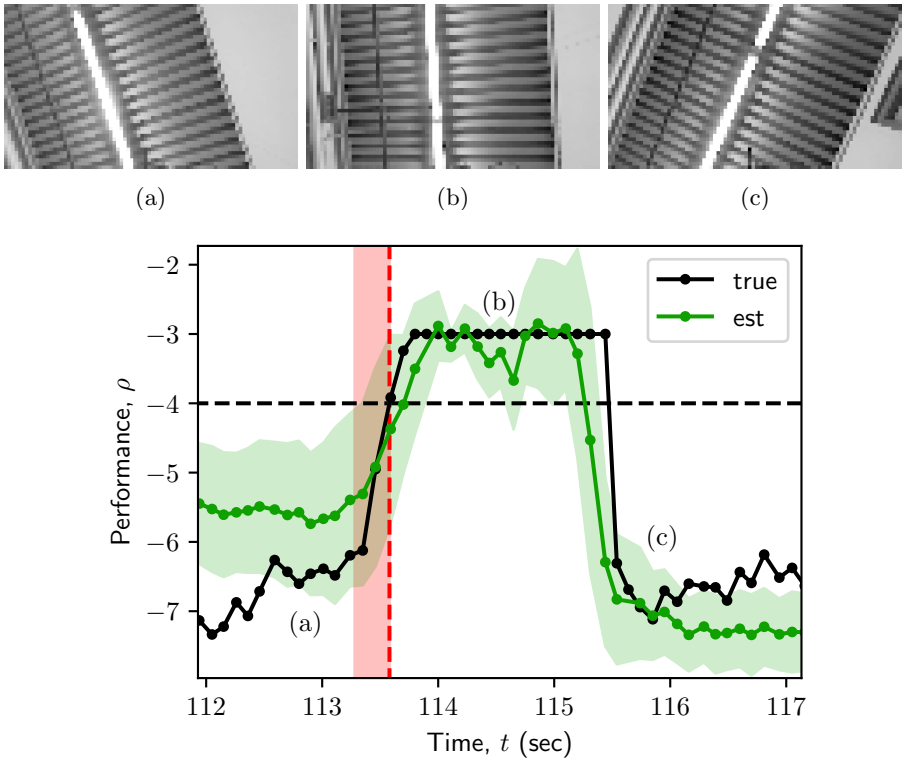


Figure 6.3: Example showing ANN prediction on DS-VO dataset. The shaded region denotes one standard deviation of predicted performance, the black line shows the observed instantaneous performance, the red dotted line indicates the failure event, and the shaded red area shows 0.3 secs before the event.

6.3 Experimental Validation

We present results using our proposed technique to model the contextual performance of two vision-based odometry systems for a ground robot.

Our goals in these experiments are as follows: First, we want to test how well each modeling approach scales to more complex and diverse contexts. Second, we want to test how sensitive each approach is to different training and test data, or in other words, how much variance the approach has. Third, we want to test what information (i.e. raw data, preprocessed, spatial, etc.) is important for performance modeling. We describe here our experimental process and present results from our IMR experiments. We then conclude with a discussion and analysis of these results.

6.3.1 Test Metrics

We quantify the quality of the learned performance models by applying them to the task of predicting perception failure, defined as falling below a minimum instantaneous performance $r_{min} = 4.0$. We predict the probability of failure occurring by computing the CDF of the predicted conditional $\hat{p}(\rho|z)$ and thresholding the probability. After classifying all test datapoints, we report the area under the receiver operating characteristic curve (AUC), as is common practice for quantifying classification performance. An AUC of 1.0 corresponds to perfect classification performance.

An issue with reporting performance across trials is that different environments have different complexities, resulting in changing modeling difficulty. As such, only part of the variance shown in Figs. 6.4a and 6.4b can be attributed to learner variance. To normalize for this changing difficulty, we also report the relative AUC (rAUC), which we define as the difference between the trial AUC for a test fold and the best AUC for that fold across all other learners and trials. The best AUC can be thought of as a crude estimate of the maximum attainable AUC for each fold. The rAUCs are shown also as box-whisker plots in Figs. 6.4c and 6.4d.

6.3.2 Modeling Approaches Tested

Preprocessing

To aid generalization in all of the tested methods, we follow the example of Simonyan and Zisserman [2014] and Simon et al. [2017] and first generate dense image features using the bottom layers of the pre-trained image classifier VGGNet Simonyan and Zisserman [2014]. Razavian et al. [2014] also used pre-trained features with linear classifiers and reported competitive classification performance.

This spatially compresses the image by a factor of 4 and produces 128 different response channels, resulting in $12 \times 12 \times 128$ DPM-VO images and $12 \times 21 \times 128$ DS-VO images. We further spatially downsample these features

by averaging over non-overlapping blocks, with different block sizes for DPM-VO and DS-VO, to produce $3 \times 3 \times 128$ feature images. Intuitively we expect that this should preserve high-level spatial relations in the image with relatively few dimensions.

We use the ubiquitous Principal Components Analysis (PCA) technique for dimensionality reduction due to its low computational complexity and ability to be run online in minibatches. This makes it well suited for our local learning setting. However, PCA is limited to extracting linear subspaces of high dimensional data, and as such is often combined with a non-linear feature transform to obtain any desired data invariances, ie. image translation invariance.

Parametric Function

We test a small ANN architecture with one convolution layer followed by two fully-connected layers. Specifically, the convolution layer consists of sixteen $1 \times 1 \times 128$ filters, which serve to further reduce the number of response channels while preserving spatial relations. Both fully connected layers were 32 units wide with ReLU activations between layers and *tanh* activations at the final output layer. The tanh outputs are scaled and shifted to allow mean outputs within $[-6, 10]$ and log standard deviations within $[-6, 3]$, corresponding to standard deviations in $[0.0025, 20]$. The performances in our dataset were clipped to lay within $[3, 8]$.

We implemented and trained our ANNs in Tensorflow[1] using batch normalization and the Adam optimizer with batch sizes of 100 datapoints. Each model was trained for 300 seconds or until convergence.

Mixture Model

Before fitting the GMM, we first dimensionally reduce and whiten the feature images using Principal Components Analysis (PCA). For efficiency we used a minibatch-based PCA algorithm. PCA was performed separately for each trial. We then fit a GMM to the dimensionally reduced images concatenated with the performance scalars to model the joint distribution. We regularized the fitting process by adding 0.1 to the diagonals of the component covariances.

We used 64 GMM components in all of our trials, as adding more components did not improve validation performance. However, we observed that the number of PCA components d strongly affected performance, so we repeated fitting for $d \in [2, 4, 8, 12, 16]$ and used the model with the best log-likelihood computed on 10% of the training data held out for model selection. We used scikit-learn[2] for both the PCA and GMM routines.

[1] https://www.tensorflow.org/

[2] http://scikit-learn.org

Non-parametric Lookup

We used the same PCA procedure to dimensionally reduce and whiten the feature images prior to kNN modeling. We then build a kd-tree-based kNN model on the dimensionally reduced images. To avoid degenerate predictions, we enforce a minimum empirical standard deviation of 0.1.

In addition to the PCA components d, we observed that the number of neighbors retrieved k also affected performance. Like with the GMMs, we tested $d \in [2, 4, 8]$ and $k \in [8, 12, 16, 20, 24]$ and used the model with the best log-likelihood computed on 10% of held-out training data. We used the scikit-learn implementation of kd-trees and kNN lookup.

6.3.3 Experimental Procedure

We test increasing contextual diversity by training and testing on multiple environments together. In our trials we tested conditions with one, three, and five environments, corresponding to 20, 60, and 100 minutes of training data, respectively.

We also test learning variance by running trials with different training and test fold assignments within each environment. Testing all possible combinations of environments and train/test assignments requires an impractically large combinatorial number of trials. As a compromise, we randomly sample 30 trials for the three and five environment conditions, and used all 15 combinations for the one environment condition.

Finally, to test the importance of spatial information, we run trials where the $3 \times 3 \times 128$ feature images are spatially averaged to size $1 \times 1 \times 128$ before learning. Note that this results in a smaller ANN, but only affects the complexity of PCA for the GMM and kNN approaches. We denote these "no spatial" test conditions with a "-ns" postfix on the method name, *e.g.* GMM-NS.

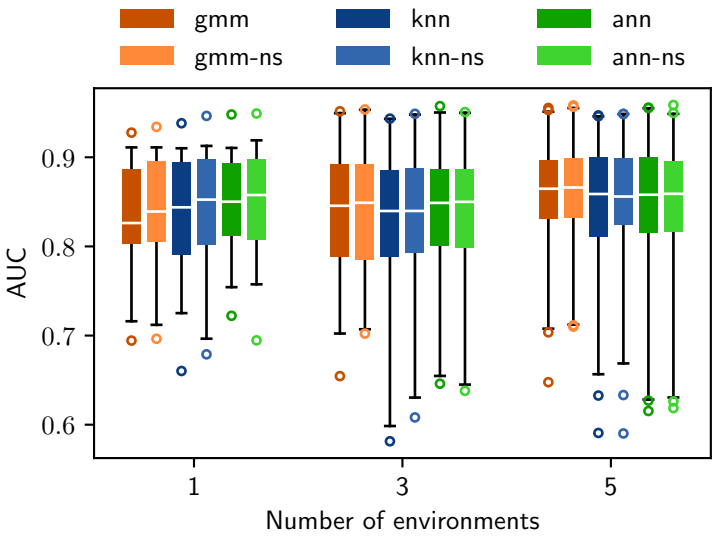
6.3.4 IMR Experiments

We teleoperated the IMR in a variety of environments and collected trajectories of paired sensor images and performances for the IMR's floor-facing DPM-VO and ceiling-facing DS-VO odometry systems. The DPM-VO images were spatially downsampled during collection to 48×48 pixels, and the DS-VO left camera's images were downsampled to 47×84 pixels. Both image streams were temporally downsampled to 10 Hz, and performance was calculated at each image time using discount parameters $T = 4$ and $\gamma = 0.6$.

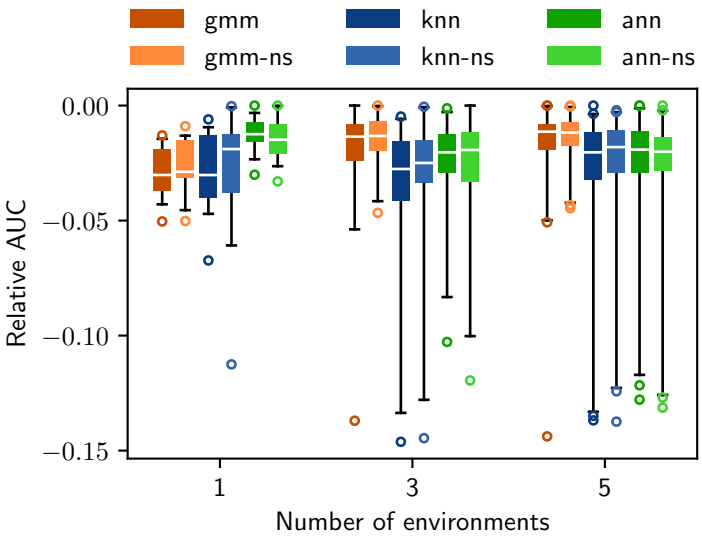
We collected 30 minutes of data in 5 environments for each system, giving us a total of $180,000$ datapoints. The test environments were specifically chosen for contextual variety that results in perception failures, ie. dark black carpet and textureless ceilings. In our experiments we divide each trajectory into three 10-minute folds. Through all of our trials we use a $2:1$ ratio of training to test data, corresponding to training on two folds and holding out the remainder.

The same sampled trial combinations were used in both the spatial and non-spatial conditions and on both the DPM-VO and DS-VO datasets, giving 900 total models trained. All experiments were performed on a desktop computer with a 6-core CPU and a GeForce 1080 GTX GPU which was used for training the ANN models.

Example traces showing failure prediction for both IMR systems in shown in Figures 6.2 and 6.3. The AUCs across all trials are shown in Figures 6.4a and 6.4b as box-whisker plots.

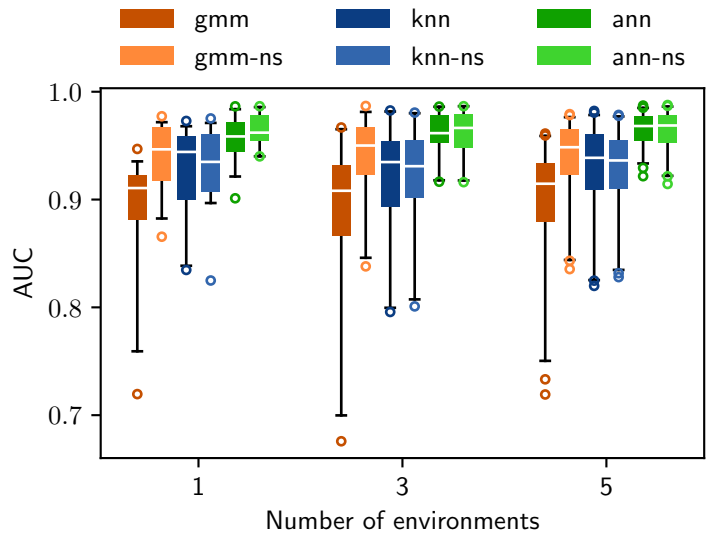


(a) DPM-VO AUC scores

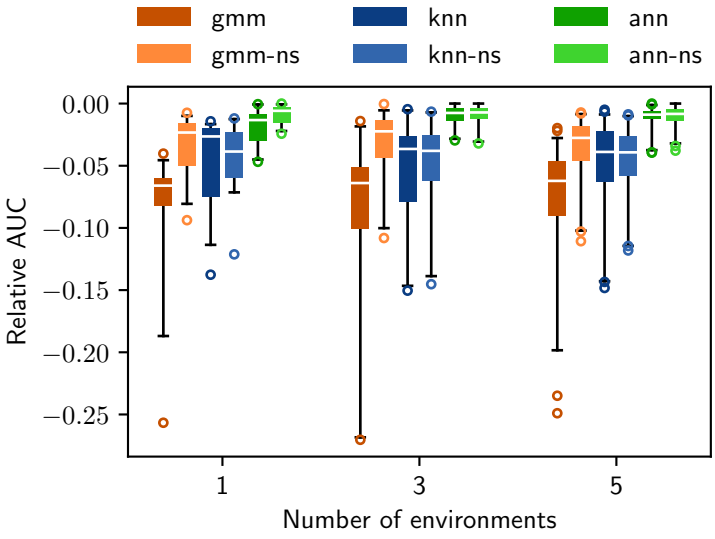


(b) DPM-VO relative AUC scores

Figure 6.4: Box-whisker representations of AUC scores across trials and test folds versus amount of environmental variation. The boxes show the second and third quartiles with a white line at the median, the whiskers extend to the 1st and 99th percentiles, and remaining outliers are shown as circles.



(c) DS-VO AUC scores



(d) DS-VO relative AUC scores

Figure 6.4: Box-whisker representations of AUC scores across trials and test folds versus amount of environmental variation (continued).

6.3.5 Discussion

Comparing Datasets

Overall we achieve reasonable performance on the DPM-VO dataset, as seen in Figure 6.4a, with a median AUC around 0.85, and quite good performance on the DS-VO dataset, shown in Figure 6.4b with the ANN models reporting around 0.96 AUC.

It is surprising that performance is better on the DS-VO dataset, as we expect that the DS-VO images of the ceiling exhibit greater diversity than the DPM-VO images of the floor. However, the DS-VO system has a weaker dependence on the robot velocity, as the ceiling is further away than the floor from the cameras. Thus, the DPM-VO dataset has a significant amount of stochasticity resulting from the robot velocity, making it more challenging to predict failures reliably. This could be addressed by augmenting the observations to make the velocity more observable.

Comparing Learners

Looking at Figure 6.4a, we see that all methods report similar AUC ranges on the DPM-VO dataset. In Figure 6.4c, however, the ANN methods have the best rAUC for one environment while the GMM methods are better for three and five environments. In contrast, the ANN methods significantly outperform all the other methods over the whole DS-VO dataset as reported in Figure 6.4b.

One possible cause for the ANN rAUC decreasing is that more DPM-VO environments increases the stochasticity in the DPM-VO data. This suggests that the ANN methods cannot handle highly stochastic behavior, possibly due to overfitting. Alternatively, it may be that the ANN architecture is too simple to capture the DPM-VO stochasticity, but this seems unlikely since the GMM methods perform well despite being linear. Correspondingly, these results suggest that GMMs handle stochasticity well.

The KNN methods perform worst out of all of the tested methods across both datasets, aside from GMM on the DS-VO dataset. The good performance of GMM-NS on DS-VO suggests that the conditional distribution is not extremely complex and nonlinear, leading us to conclude that the training data was simply not dense enough. This hypothesis could be tested by collecting more data and running more experiments with KNN.

Effect of Environmental Variation

We observe in all of Figure 6.4 very little effect from increasing environmental variation on both the AUC and rAUCs for all methods. This suggests that our regularization and model selection procedures are working correctly. The one exception is on the DPM-VO dataset, where increasing the number of environments dramatically lowers the 1st percentile for the KNN and ANN methods. As previously discussed, this is likely due to stochasticity in the data.

Learning Procedure Variance

All methods exhibit a wide range of AUCs ($\approx \pm 0.15$)on the DPM-VO dataset, as seen in Figure 6.4a, but have a relatively small range of rAUCs ($\approx \pm 0.03$) for one environment, as seen in Figure 6.4c. This suggests that the learning variance emerges primarily due to DPM-VO environmental diversity, possibly due to some environments having more stochasticity.

In contrast, the ranges of AUCs on the DS-VO dataset in Figure 6.4b correspond to roughly double the rAUC ranges in Figure 6.4d, suggesting an even balance between learning stochasticity and variations in environmental difficulty. The ANN methods both exhibit the smallest rAUC ranges (≈ 0.02) for all DS-VO trials, suggesting that it is the least sensitive to training data.

Spatial Information Tradeoff

Looking across Figure 6.4, we see that removing spatial information has relatively little effect on all the methods and datasets. The exception is GMM being outperformed by GMM-NS on the DS-VO dataset in Figures 6.4b and 6.4d. One possibility is that the variation resulting from spatial information takes up the PCA output dimensions instead of more salient texture information. However, KNN and KNN-NS perform comparably to each other and better than GMM, despite using fewer PCA dimensions. This suggests that somehow the spatial information is causing the GMM model to overfit, possibly by learning a large number of correlations between spatial dimensions.

Chapter 7

Conclusion

If a house is only as strong as its foundation, then perhaps we can say that a robot is only as reliable as its perception system. And just as foundations must conform to the land to be effective, so must perception systems adapt to their context.

At the beginning of this work we presented context as a absract concept to explain perceptual unreliability. With this understanding, we then developed mechanisms for adapting to context, relying on the ability to interact with and observe context's effects while never truly understanding its form. By formalizing perceptual performance in a modern probabilistic framework, we naturally derived an approach for evaluating performance without ground truth, the Approximate Posterior Estimate. This technique is attractively simple, and as seen in our thorough experiments on a simulated robot, two offline systems, and two hardware systems, is effective in practice.

By being able to obtain feedback on perceptual quality without requiring any external feedback, the Approximate Posterior Estimate enables us to develop two powerful mechanisms for context adaptation. The first is the ubiquitous practice of on-site parameter tuning, in effect, changing the behavior of a perception system to better suit the deployment. To automate this task, we characterized tuning as black-box optimization, which we deeply explored through a bevy of experiments. Our results on tuning the parameters of four KITTI and IMR systems show that widely-used optimization algorithms, especially Bayesian optimization, can find performant configurations quickly and reliably. Further, we showed that tuning a system on-site can avoid catastrophic degredation of performance resulting from context variation.

Still, as they say, "discretion is the better part of valor," for no amount of tuning can circumvent some failure modes. For these, we proposed to predict when failures are likely to occur by learning from past failures. With the ability to learn from mistakes, we argue, perception systems can close the loop on the act of perceiving. In our experiments on the two IMR vision systems, we showed that common modeling techniques, and artificial neural networks in particular, can learn to accurately predict perception failures in diverse environments from

only data collected on-site and without external supervision.

We believe that this thesis represents a significant step towards reliable, general robotic perception. The concepts in this work deal fundamentally with the limitations of human understanding as applied to engineering, and as such, will remain relevant even as the state of the art advances relentlessly.

Future Work

Nonetheless, there remain a number of interesting avenues deserving of future investigation. We detail a few of them here.

Improving the APE

The Approximate Posterior Estimate (APE) is the heart of the mechanisms described in this work; when it fails, so does the adaptation. In some sense, the APE is only as good as the probabilistic models it uses, and as such, poorly modeled phenomena can degrade APE effectiveness. One particularly problematic issue is that of bias, or systematic errors, which are difficult to detect with a relatively naive modeling approach like the adaptive Kalman filter (AKF). The other issue is noise modeling lag in the AKF, which is mostly a concern for low-bandwidth systems. A potential approach for both of these issues is to rely on learning approaches for higher fidelity noise models, such as our previous work in Hu and Kantor [2015] or that of Haarnoja et al. [2016].

Selecting Benchmarks for Evaluation

Our demonstrations have used heuristically-designed fixed trajectories and selected data logs as samples for evaluating system performance in a deployment. Strictly speaking, Monte Carlo methods do not require samples to be independent between expectations, only within computations for an expectation Ji and Li [2012]. Nonetheless, we expect that care should be given to the process of generating or selecting these benchmark samples so as to minimize the error in the Monte Carlo estimate.

From Black to Gray Boxes

Formulating tuning as black box optimization allows us to take advantage of powerful and general optimization algorithms, but at the cost of being completely naive about the optimization task itself. Human engineers use much more information than just the objective value when tuning, whether from diagnostics embedded in the system, temporal behavior, or just "a feeling." One possible way to formalize this extra information is the concept of instrument variables from statistics Imbens and Angrist [1994]. Another interesting avenue is exploring different evaluation fidelities in a framework such as that of Kandasamy et al. [2016].

Tuning with Configuration Libraries

Bayesian optimization provides a relatively sample-efficient way to tune parameters on-site, as seen in our experiments. However, our intuition is that there exist configurations that perform near-optimally in relatively similar environments. Thus, if we could discover a relatively small library of configurations that "span" a large variety of environments, we can quickly tune by running trials over the library. Techniques for selecting from the library using trials are well developed, in particular in multi arm bandits Jamieson et al. [2014], and concepts in submodular subset selection may be useful for efficiently building libraries Krause and Guestrin [2007].

End to End Performance Learning

Our demonstrated work on performance prediction relies on the ineffable power of pre-trained convolutional neural network features. However, these features are designed to be interfaced into a large neural network, so it comes as no surprise that the neural network model we tested generally performs better than the simpler approaches. Yet, the Gaussian mixture model (GMM) and nearest neighbor (KNN) offer promising computational advantages. Were it possible to incorporate these approaches into a larger learning framework, it may be possible to pre-learn features that are amenable to GMM or KNN learning online. Recent work from Finn et al. [2017] may be applicable in this regard.

Closing the Loop with Responsive Reconfiguration

We have proposed parameter tuning as a adaptation process to be performed upon deployment, akin to calibration. However, many parameters can be changed on-the-fly as the robot operates, allowing much faster, reactive adaptation to changing contexts. Learning a reconfiguration policy to do this involves reasoning about the dynamics of the perception system at a fine granularity. Our attempts at adapting model-free reinforcement learning techniques to this task, such as those described by Peters and Schaal [2006], have been hampered by sample complexity and difficulty dealing with variance. Model-based approaches, such as that of Deisenroth and Rasmussen [2011], may help with this aspect.

Bibliography

Pieter Abbeel, Adam Coates, Michael Montemerlo, Andrew Y. Ng, and Sebastian Thrun. Discriminative Training of Kalman Filters. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

S. Achar, B. Sankaran, S. Nuske, S. Scherer, and S. Singh. Self-supervised segmentation of river scenes. In *2011 IEEE International Conference on Robotics and Automation*, pages 6227–6232, May 2011.

Hatem Alismail, Browning Browning, and Simon Lucey. Direct Visual Odometry using Bit-Planes. *ArXiv e-prints arXiv:1064.00990*, 2016.

John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356. ISSN 1573-1405.

T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM Algorithm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568, Oct 2006. doi: 10.1109/IROS.2006.281644.

Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, Aug 1988. ISSN 0018-9219.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *JMLR*, page 305, 2012.

Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with Gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496, May 2016.

O. Boiman, E. Shechtman, and M. Irani. In defense of Nearest-Neighbor based image classification. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

Amine Bourki, Matthieu Coulm, Philippe Rolet, Olivier Teytaud, and Paul Vayssière. Parameter Tuning by Simple Regret Algorithms and Multiple Simultaneous Hypothesis Testing. In *ICINCO2010*, page 10, funchal madeira, Portugal, 2010.

Christopher Brunner and Thierry Peynot. *Perception Quality Evaluation with Visual and Infrared Cameras in Challenging Environmental Conditions*, pages 711–725. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-28572-1.

Christopher Brunner, Thierry Peynot, and Teresa Vidal-Calleja. Visual metrics for the evaluation of sensor data quality in outdoor perception. *International Journal of Intelligent Control and Systems*, 16(2):142–159, June 2011. Special Edition: Quantifying the Performance of Intelligent Systems.

W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization. volume 2, pages 1346–1352, 1997.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. An experimental comparison of Bayesian optimization for bipedal locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1951–1958, May 2014.

Sonia Chernova and Manuela Veloso. An evolutionary approach to gait learning for four-legged robots. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2562–2567 vol.3, Sept 2004.

W. Churchill, Chi Hay Tong, C. Guru, I. Posner, and P. Newman. Know your limits: Embedding localiser performance models in teach and repeat maps. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4238–4244, May 2015.

Javier Civera, Oscar G. Grasa, Andrew J. Davison, and J. M. M. Montiel. 1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5): 609–631, 2010. ISSN 1556-4967.

B. Clipp, Jongwoo Lim, J. M. Frahm, and M. Pollefeys. Parallel, real-time visual SLAM. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3961–3968, Oct 2010. doi: 10.1109/IROS.2010.5653696.

S. Daftry, S. Zeng, J. A. Bagnell, and M. Hebert. Introspective perception: Learning to predict failures in vision systems. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1743–1750, Oct 2016.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *In Proceedings of the International Conference on Machine Learning*, 2011.

J. Dequaire, C. H. Tong, W. Churchill, and I. Posner. Off the beaten track: Predicting localisation performance in visual teach and repeat. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 795–800, May 2016.

Josip Djolonga, Andreas Krause, and Volkan Cevher. High-Dimensional Gaussian Process Bandits. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1025–1033. 2013.

Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised Particle Filtering for Dynamic Bayesian Networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, pages 176–183, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-709-9.

J. Dunik, M. Simandl, and O. Straka. Unscented Kalman Filter: Aspects and Adaptive Setting of Scaling Parameter. *IEEE Transactions on Automatic Control*, 57(9):2411–2416, Sept 2012. ISSN 0018-9286.

A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163, Jul 2002. ISSN 0018-9219.

Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning CPG-based Biped Locomotion with a Policy Gradient Method: Application to a Humanoid Robot. *The International Journal of Robotics Research*, 27(2):213–228, 2008.

Clemens Eppner, Sebastian Höfer, Rico Jonschkowski, Roberto Martín-Martín, Arne Sieverling, Vincent Wall, and Oliver Brock. Lessons from the Amazon Picking Challenge: Four Aspects of Building Robotic Systems. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

N. Fairfield and D. Wettergreen. Active localization on the ocean floor with multibeam sonar. In *OCEANS 2008*, pages 1–10, Sept 2008.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

Matthew S. Gerber. Predicting crime using Twitter and kernel density estimation. *Decision Support Systems*, 61:115 – 125, 2014. ISSN 0167-9236.

Zoubin Ghahramani and Sam T. Roweis. Learning Nonlinear Dynamical Systems Using an EM Algorithm. In *Proceedings of the 11th International Conference on Neural Information Processing Systems*, NIPS'98, pages 431–437, Cambridge, MA, USA, 1998. MIT Press.

Andrew R. Gilpin. Table for Conversion of Kendall'S Tau to Spearman'S Rho Within the Context of Measures of Magnitude of Effect for Meta-Analysis. *Educational and Psychological Measurement*, 53(1):87–92, 1993. doi: 10.1177/0013164493053001007.

H. Grimmett, R. Triebel, R. Paul, and I. Posner. Introspective classification for robot perception. *The International Journal of Robotics Research*, 35(7): 743–762, 2016.

G. Grisetti, R. Kmmerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *2010 IEEE International Conference on Robotics and Automation*, pages 273–278, May 2010.

G. Grisetti, R. Kmmerle, and K. Ni. Robust optimization of factor graphs by using condensed measurements. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 581–588, Oct 2012.

C. Gurau, D.Rao, C.H. Tong, and I.Posner. Learn from experience: probabilistic prediction of perception performance to avoid failure. page 027836491773060, 10 2017.

Corina Gurău, Chi Hay Tong, and Ingmar Posner. Fit for Purpose? Predicting Perception Performance Based on Past Experience. In Dana Kulić, Yoshihiko Nakamura, Oussama Khatib, and Gentiane Venture, editors, *2016 International Symposium on Experimental Robotics*, pages 454–464, Cham, 2017. Springer International Publishing. ISBN 978-3-319-50115-4.

Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. Backprop KF: Learning Discriminative Deterministic State Estimators. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4376–4384. Curran Associates, Inc., 2016.

R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009. ISSN 1556-4967.

Lixing Han and Michael Neumann. Effect of dimensionality on the NelderMead simplex method. *Optimization Methods and Software*, 21(1):1–16, 2006.

Ankur Handa, Richard A. Newcombe, Adrien Angeli, and Andrew J. Davison. *Real-Time Camera Tracking: When is High Frame-Rate Best?*, pages 222–235. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-33786-4.

Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

M. Happold, M. Ollis, and N. Johnson. Enhancing Supervised Terrain Classification with Predictive Unsupervised Learning. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

K. Hausman, S. Weiss, R. Brockers, L. Matthies, and G. S. Sukhatme. Self-calibrating multi-sensor fusion with probabilistic measurement validation for seamless sensor switching on a UAV. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4289–4296, May 2016.

Jeffrey Hawke, Corina Gurău, Chi Hay Tong, and Ingmar Posner. *Wrong Today, Right Tomorrow: Experience-Based Classification for Robot Perception*, pages 173–186. Springer International Publishing, Cham, 2016. ISBN 978-3-319-27702-8.

H. Hu and G. Kantor. Parametric covariance prediction for heteroscedastic noise. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3057, Sept 2015.

Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. *Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera*, pages 235–252. Springer International Publishing, Cham, 2017. ISBN 978-3-319-29363-9.

Frank Hutter, Holger H. Hoos, and Thomas Stützle. Automatic Algorithm Configuration Based on Local Search. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pages 1152–1157. AAAI Press, 2007. ISBN 978-1-57735-323-2.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. *Sequential Model-Based Optimization for General Algorithm Configuration*, pages 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-25566-3.

Guido W. Imbens and Joshua D. Angrist. Identification and Estimation of Local Average Treatment Effects. *Econometrica*, 62(2):467–475, 1994. ISSN 00129682, 14680262.

Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sbastien Bubeck. lil' UCB : An Optimal Exploration Algorithm for Multi-Armed Bandits. In Maria Florina Balcan, Vitaly Feldman, and Csaba Szepesvri, editors, *Proceedings of The 27th Conference on Learning Theory*, volume 35 of *Proceedings of Machine Learning Research*, pages 423–439, Barcelona, Spain, 13–15 Jun 2014. PMLR.

Nataraj Jammalamadaka, Andrew Zisserman, Marcin Eichner, Vittorio Ferrari, and C. V. Jawahar. *Has My Algorithm Succeeded? An Evaluator for Human Pose Estimators*, pages 114–128. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-33712-3.

Hao Ji and Yaohang Li. Reusing Random Walks in Monte Carlo Methods for Linear Systems. *Procedia Computer Science*, 9:383 – 392, 2012. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2012.04.041. Proceedings of the International Conference on Computational Science, ICCS 2012.

K. Jo, Y. Jo, J. K. Suhr, H. G. Jung, and M. Sunwoo. Precise Localization of an Autonomous Car Based on Probabilistic Noise Models of Road Surface Marker Features Using Multiple Cameras. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3377–3392, Dec 2015. ISSN 1524-9050.

Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High Dimensional Bayesian Optimisation and Bandits via Additive Models. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 295–304, 2015.

Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabas Poczos. Gaussian Process Bandit Optimisation with Multi-fidelity Evaluations. In *Advances in Neural Information Processing Systems 29*, pages 992–1000. 2016.

Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most Likely Heteroscedastic Gaussian Process Regression. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 393–400, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3.

Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 3, pages 2619–2624 Vol.3, April 2004.

Andreas Krause and Carlos Guestrin. Near-optimal Observation Selection Using Submodular Functions. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pages 1650–1654. AAAI Press, 2007. ISBN 978-1-57735-323-2.

Philipp Krsi, Bastian Bcheler, Franois Pomerleau, Ulrich Schwesinger, Roland Siegwart, and Paul Furgale. Lighting-invariant Adaptive Route Following Using Iterative Closest Point Matching. *Journal of Field Robotics*, 32(4): 534–564, 2015. ISSN 1556-4967.

R. Laxhammar, G. Falkman, and E. Sviestins. Anomaly detection in sea traffic - A comparison of the Gaussian Mixture Model and the Kernel Density Estimator. In *2009 12th International Conference on Information Fusion*, pages 756–763, July 2009.

Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *CoRR*, abs/1603.02199, 2016.

Ilya Loshchilov and Frank Hutter. CMA-ES for Hyperparameter Optimization of Deep Neural Networks. *CoRR*, abs/1604.07269, 2016.

Y.P Mack and M Rosenblatt. Multivariate k-nearest neighbor density estimates. *Journal of Multivariate Analysis*, 9(1):1 – 15, 1979. ISSN 0047-259X.

H. Mania, A. Guy, and B. Recht. Simple random search provides a competitive approach to reinforcement learning. *ArXiv e-prints*, March 2018.

Alonso Marco, Philipp Hennig, Jeanette Bohg, Stefan Schaal, and Sebastian Trimpe. Automatic LQR tuning based on Gaussian process global optimization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 270–277, May 2016.

A. H. Mohamed and K. P. Schwarz. Adaptive Kalman Filtering for INS/GPS. *Journal of Geodesy*, 73(4):193–203, 1999. ISSN 0949-7714. doi: 10.1007/s001900050236.

L. Montesano, J. Minguez, and L. Montano. Probabilistic scan matching for motion estimation in unstructured environments. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3499–3504, Aug 2005.

Venkatraman Narayanan and Maxim Likhachev. Discriminatively-guided Deliberative Perception for Pose Estimation of Multiple 3D Object Instances. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

R. Newson. Parameters behind "nonparametric" statistics: Kendall's tau, Somers' D and median differences. *Stata Journal*, 2(1):45–64(20), 2002.

E. Olson. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407, May 2011.

Peter Ondruska, Corina Gurau, Letizia Marchegiani, Chi Hay Tong, and Ingmar Posner. Scheduled Perception for Energy-Efficient Path Following. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA, May 2015.

Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *Ann. Math. Statist.*, 33(3):1065–1076, 09 1962.

J. Peters and S. Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225, Oct 2006. doi: 10.1109/IROS.2006.282564.

L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3406–3413, May 2016.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 512–519, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-4308-1.

Thomas Röfer. *Evolutionary Gait-Optimization Using a Fitness Function Based on Proprioception*, pages 310–322. 2005.

John G. Rogers, Alexander J. B. Trevor, Carlos Nieto-Granda, Alex Cunningham, Manohar Paluri, Nathan Michael, Frank Dellaert, Henrik I. Christensen, and Vijay Kumar. *Effects of Sensory Precision on Mobile Robot Localization and Mapping*, pages 433–446. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-28572-1.

Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *Ann. Math. Statist.*, 27(3):832–837, 09 1956.

J. Rwekmper, C. Sprunk, G. D. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard. On the position accuracy of mobile robot localization based on particle filters combined with scan matching. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3158–3164, Oct 2012.

Dhruv Mauria Saxena, Vince Kurtz, and Martial Hebert. Learning robust failure response for autonomous vision based flight. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5824–5829. IEEE, June 2017.

Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. Lower Bounds on Regret for Noisy Gaussian Process Bandit Optimization. In *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1723–1742, Amsterdam, Netherlands, 07–10 Jul 2017.

Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. *Initialization-Free Monocular Visual-Inertial State Estimation with Application to Autonomous MAVs*, pages 211–227. Springer International Publishing, Cham, 2016. ISBN 978-3-319-23778-7.

T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand Keypoint Detection in Single Images Using Multiview Bootstrapping. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556, 2014.

R. Singh, B. C. Pal, and R. A. Jabr. Statistical Representation of Distribution System Loads Using Gaussian Mixture Model. *IEEE Transactions on Power Systems*, 25(1):29–37, Feb 2010. ISSN 0885-8950.

S. K. Smit and A. E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *2009 IEEE Congress on Evolutionary Computation*, pages 399–406, May 2009.

N. Snderhauf and P. Protzel. Towards a robust back-end for pose graph SLAM. In *2012 IEEE International Conference on Robotics and Automation*, pages 1254–1261, May 2012.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2951–2959. 2012.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 1015–1022, USA, 2010.

C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, page 252 Vol. 2, 1999. doi: 10.1109/CVPR.1999.784637.

Matthew Tesch, Jeff Schneider, and Howie Choset. Using response surfaces and expected improvement to optimize snake robot gait parameters. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1069–1074, Sept 2011.

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Optimizing Walking Controllers. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 168:1–168:8, New York, NY, USA, 2009a. ACM. ISBN 978-1-60558-858-2.

Q. Wang, S. R. Kulkarni, and S. Verdu. Divergence Estimation for Multidimensional Densities Via k-Nearest-Neighbor Distances. *IEEE Transactions on Information Theory*, 55(5):2392–2405, May 2009b. ISSN 0018-9448.

C. Wellington and A. Stentz. *Learning Predictions of the Load-Bearing Surface for Autonomous Rough-Terrain Navigation in Vegetation*, pages 83–92. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-32854-4.

K. Wu, K. Zhang, W. Fan, A. Edwards, and P. S. Yu. RS-Forest: A Rapid Density Estimator for Streaming Anomaly Detection. In *2014 IEEE International Conference on Data Mining*, pages 600–609, Dec 2014.

Dit-Yan Yeung and C. Chow. Parzen-window network intrusion detectors. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 4, pages 385–388 vol.4, 2002.

P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh. Predicting Failures of Vision Systems. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3573, June 2014.

X. Zhu and P. Milanfar. Automatic Parameter Selection for Denoising Algorithms Using a No-Reference Measure of Image Content. *IEEE Transactions on Image Processing*, 19(12):3116–3132, Dec 2010. ISSN 1057-7149.