Articulated Pose Estimation via Over-parametrization and Noise Projection

by

Jonathan David Brookshire

B.S., University of Virginia (2002)
M.S., Carnegie Mellon University (2004)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Aug 2013

Author .....
Department of Electrical Engineering and Computer Science
September 1, 2013

Certified by .....
Seth Teller
Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Articulated Pose Estimation via Over-parametrization and Noise Projection

by

Jonathan David Brookshire

Submitted to the Department of Electrical Engineering and Computer Science
on September 1, 2013, in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

Outside the factory, robots will often encounter mechanical systems with which they need to interact. The robot may need to open and unload a kitchen dishwasher or move around heavy construction equipment. Many of the mechanical systems encountered can be described as a series of rigid segments connected by joints. The pose of a segment places constraints on adjacent segments because they are mechanically connected. When modeling or perceiving the motion of such an articulated system, it is beneficial to make use of these constraints to reduce uncertainty. In this thesis, we examine two aspects of perception related to articulated structures. First, we examine the special case of a single segment and recover the rigid body transformation between two sensors mounted on it. Second, we consider the task of tracking the configuration of a multi-segment structure, given some knowledge of its kinematics.

First, we develop an algorithm to recover the rigid body transformation, or extrinsic calibration, between two sensors on a link of a mobile robot. The single link, a degenerate articulated object, is often encountered in practice. The algorithm requires only a set of sensor observations made as the robot moves along a suitable path. Over-parametrization of poses avoids degeneracies and the corresponding Lie algebra enables noise projection to and from the over-parametrized space. We demonstrate and validate the end-to-end calibration procedure, achieving Cramer-Rao Lower Bounds. The parameters are accurate to millimeters and milliradians in the case of planar LIDARs data and about 1 cm and 1 degree for 6-DOF RGB-D cameras.

Second, we develop a particle filter to track an articulated object. Unlike most previous work, the algorithm accepts a kinematic description as input and is not specific to a particular object. A potentially incomplete series of observations of the object's links are used to form an on-line estimate of the object's configuration (i.e., the pose of one link and the joint positions). The particle filter does not require a reliable state transition model, since observations are incorporated during particle proposal. Noise is modeled in the observation space, an over-parametrization of the state space, reducing the dependency on the kinematic description. We compare our method to several alternative implementations and demonstrate lower tracking error for fixed observation noise.

Thesis Supervisor: Seth Teller
Title: Professor

Acknowledgments

This work would not have been possible without the support of many people. I would first thank my advisor, Professor Seth Teller. Seth repeatedly astounded me with his breath of knowledge and ability to quickly reach conclusions I gleaned only after a period of long reflection. He provided that delicate balance of guidance: enough to be helpful, but not so much as to be obtrusive. For their sake, I hope he continues to mentor students for many years to come.

Thanks also to my committee members, Professors Tomás Lozano-Pérez and Berthold Horn. They have contributed greatly to this work. I have appreciated their insightful comments and feedback.

My lab mates in the RVSN group were also crucial to this thesis. Much thanks to Albert Huang and Luke Fletcher for helping me get rolling with the forklift in the early days. Matt Walter and Matt Antone were both tremendously helpful and kind individuals: their intelligence was matched only by their humility. Steve Proulx was an invaluable teammate and helped keep the equipment operational and provide a sanity check on ideas. Thanks also to Bryt Bradley for having everything organized making all of our work possible. It has been a great pleasure to work with Mike Fleder, Sudeep Pillai, and Sachi Hemachandra over the past few years.

To give the proper thanks to my family would take more than the remainder of this document. My mom and dad have provided constant support throughout my life. My wife has been a constant source of strength and patience and reminded me about life outside these pages. My new daughter has provided motivation and focus.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many simple mechanical systems can be described as a set of rigid, discrete segments, connected by joints. Such articulated structures can be found in the kitchen (e.g., cabinet drawers, appliance doors, faucet handles, toaster levers), the factory (e.g., pliers, screwdrivers, clamps), and the office (e.g., box lids, folding tables, swivel chairs). At a construction site, for example, a backhoe is a tractor chassis with an attached, articulated arm. Even the relative poses of sensors on an autonomous forklift can be considered as a single segment, joint-less linkage.

Figure 1-1: The sensors on a robotic platform (left) form a simple, rigid mechanical chain. The arm of a backhoe (right) forms a more complex mechanical chain with several joints.

Robots in many domains must model and interact with articulated structures. A robot attempting to safely navigate a construction site would need to model the configurations of construction equipment. In an office environment, the robot might need to manipulate hinged drawers or open boxes. At the same time, the robot may need to model its internal articulated systems. Robot arms are articulated chains, and forward kinematics may have errors as a result of poor physical models and mechanical

deformations due to wear-and-tear. Merging observations from joint telemetry and image sensors, for example, can help reduce these errors.

Another special case of interest is that of a rigid chain with no joints. For example, precise models of sensor reference frames are required for data fusion. These reference frames form simple, rigid kinematic chains. (For convenience of notation, we consider articulated structures as rigid links connected by any number of joints, including the special case where there are no joints.)

In this thesis, we focus on two specific aspects of such "articulated" structures. First, we consider the case where two reference frames are connected via a single, rigid segment. The goal is to recover the 3D rigid body transform between two reference frames. This problem is especially common in robotics, where it presents itself as the extrinsic sensor calibration problem. As a result, we focus on the case where we desire the rigid body pose between pairs of sensors. Our method automatically recovers the calibration parameters, regardless of the sensor type, given incremental egomotion observations from each sensor. Neither knowledge of absolute sensor pose nor synchronous sensors is required.

Second, we consider the case where reference frames are linked by a series of segments, connected by multiple joints. Here, our task will be to estimate the most likely configuration of the structure, given some observations and a description of the mechanical chain of segments and joints. This ability is useful when a robot must interact with a moving, articulated object, or for tracking the robot's own manipulators. Our technique is not specific to any particular structure and we focus on the common situation where the observations are more accurate than the predictive model. The method is shown to achieve higher tracking accuracy than competing techniques.

When working with articulated structures, both in the context of extrinsic sensor calibration and tracking, we find that building a successful system depends on effective system parametrization. Throughout this work, we refer to "parametrization" to mean the choice of the quantities used to express the relevant specification of either the extrinsic calibration or the configuration of the articulated object. We often find that over-parametrization, or the use of more quantities than strictly required to express the relevant specification, is useful to avoid degeneracies and express noise. In the extrinsic sensor calibration algorithm, we use dual quaternions, an 8-element expression of a 6-DOF value, to avoid degeneracies. In the articulated object tracking algorithm, we diffuse particles in the observation space. In many situations, the observations contain more than enough information to recover the state and, thus, form an over-parametrization of the state space.

Over-parametrizations are often avoided in practice because the redundant quantities require additional constraints which complicate many algorithms. We employ several tools, building on Lie algebra and constrained information theory, to address these complications. We specifically address the representation of uncertainty when using over-parametrizations. Uncertainty is expressed as noise in a local tangent plane which can be projected to and from the over-parametrized space.

In Chapter 2, we address the topic of the rigid, single-segment mechanical chain in the context of extrinsic sensor calibration. The sensor calibration procedure accepts as input incremental pose observations from two sensors, along with uncertainties, and outputs an estimate of the calibration parameters and covariance matrix. In the planar case, we recover calibration parameters to within millimeters and milliradians of truth. For the 6-DOF case, the recovered parameters are accurate to within 1 cm and 1 degree of truth. Additionally, the 6-DOF parameters are consistent (when measured as a closed kinematic chain) to within millimeters and milliradians. We also demonstrate that the calibration uncertainty based on the CRLB closely matches the experimental uncertainty.

Then, in Chapter 3, we present a method to track a multi-segment linkage. The tracking method accepts a fixed model of the chain and observations as input and produces an estimate for the configuration of the chain (base pose and joint angles). We demonstrate the algorithm on several simulated and real examples. We demonstrate an order of magnitude reduction in the number of particles/hypotheses, while maintaining the same error performance.

Chapter 2

Extrinsic Calibration

The autonomous vehicle and the forklift shown in Figure 2-1 are robotic vehicles recently developed by the RVSN (Robotics, Vision, Sensor Networks) group at MIT. A common requirement for each of these vehicles is a map of nearby obstacles to enable safe navigation. Since different sensors have different perception capabilities and limited fields of view (FOV), each platform has an array of sensors. The Urban Grand Challenge [47] vehicle (Figure 2-1a) has five cameras, 12 planar LIDARs, a 3D Velodyne, and 16 radar sensors. The Agile Forklift [69] (Figure 2-1b) has 4 cameras and 15 planar LIDARs. When combined, the heterogeneous sensors provide a sensing network encompassing the vehicle and work to ameliorate the consequences of any one sensor's failure mode.

The sensors can be considered as a simple kinematic chain, with rigid segments connecting pairs of sensors. In order to combine the information from the various sensors into a single, fused obstacle map, the relative poses of the sensors (i.e., the parameters of each segment) must be known. Thus, we consider the problem of recovering the calibration parameters describing the rigid body transform between sensors.

The 6-DOF calibration task is a common problem (see Section 2.3). Previous approaches typically require manual measurements, establishing a global reference frame, or augmenting the environment with known structure. Instead, we develop a sensor-agnostic calibration process which accepts incremental pose observations and uncertainties as input and produces an estimate of the calibration parameters, along with an uncertainty, as output. Essentially, the estimate is produced by finding the calibration parameters most consistent with the rigid body motion of all the sensors. The technique does not require a special calibration rig, an augmented environment, sensors with overlapping field of views, or synchronous incremental pose observations. Simultaneous Localization and Mapping (SLAM) is not used and no global reference frame must be established.

(a) (b)

Figure 2-1: Both these autonomous vehicles require extrinsic sensor calibration to fuse data from multiple sensors.

2.1 Overview

A natural question is whether incremental pose observations alone contain sufficient information to recover relative sensor calibration. Intuition suggests that such observations are indeed sufficient (Figure 2-2) because the pose difference between two sensors causes them to experience different incremental motion (Figure 2-2b and Figure 2-2c). The calibration that best transforms the motion of the first sensor to align with that of the second sensor will be the desired rigid-body transformation.

We formalize this intuitive argument and develop a procedure to estimate the calibration. We show that the calibration is observable (i.e., that there exists sufficient information in the input observations to estimate the calibration), so long as certain degenerate motion paths are avoided. We explain the Cramer-Rao Lower Bound (CRLB) [71] and show how to compute it for the calibration estimate. As we will see, the CRLB provides a best-case minimum error (covariance matrix) for the estimate, enabling the accuracy of the calibration to be assessed. The CRLB covariance can also be incorporated, as a source of uncertainty, into subsequent processing.

Our method is applicable to any sensor, or sensor combination, that permits observation of its own incremental motion. The process can also be applied to find the pose of sensors relative to a robot frame, if the relative motion of the robot body can be observed (e.g., using inertial and odometry data). The calibration procedure can be summarized as follows:

1. Move the robot along a non-degenerate path.
2. Recover per-sensor incremental poses and covariances.
3. Resample asynchronous sensor data to correct for different sample times.
4. Use least squares to recover relative sensor calibration.
5. Compute the CRLB to estimate uncertainty of the calibration parameters.

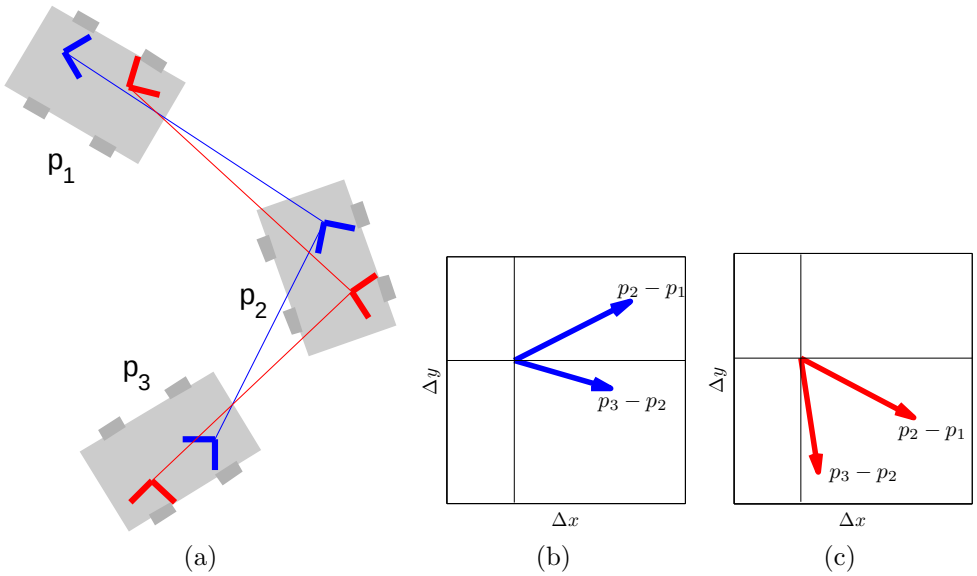


Figure 2-2: As the robot moves from p_1 to p_3, two sensors (positioned as in (a)) will experience different translational ((b) and (c)) and rotational (not shown) incremental motion. The calibration relating the sensors is the transformation that best brings the disparate observed motions into agreement.

2.1.1 Structure

As suggested in Table 2.1, we separate the discussion between 3-DOF (x, y, θ) calibration and 6-DOF $(x, y, z, \rho, \vartheta, \psi)$ calibration. Although 3-DOF calibrations can be handled by 6-DOF approaches, the 3-DOF case is easier to understand. In the sections describing the 3-DOF calibration, we develop the problem, analyze its observability, and demonstrate our calibration techniques. Subsequent sections discussing the 6-DOF calibration rely on the same underlying theory for parameter recovery, but require a significantly different parametrization. This parametrization, the unit dual quaternion (DQ), is developed in Sections 2.5.1, 2.5.2, and 2.5.3.

Pseudo-code for the algorithm is initially presented in Section 2.4.6 and developed further thereafter. Additionally, our algorithm can make use of observation uncertainties, but these can be difficult to obtain in practice. Therefore, we provide a method to estimate uncertainties in Section 2.4.7. Section 2.4.4 discusses bias in the calibration process.

In the 3-DOF and 6-DOF discussions, we rely on an understanding of estimation and information theory, especially as related to the maximum *a posteriori* estimator and the CRLB. We review these preliminary concepts in Section 2.2. Previous work on the calibration problem is reviewed in Section 2.3.

Table 2.1: Calibration outline

	3-DOF Calibration	6-DOF Calibration
Problem statement	§2.4.1	§2.5.4, §2.5.5
Observability & CRLB	§2.4.2	§2.5.6
Estimation	§2.4.3	§2.5.7
Interpolation	§2.4.5	§2.5.8
Results	§2.4.8	§2.5.9

2.1.2 Contributions

1. **Sensor-agnostic calibration.** Our research is distinct from its predecessors in that we consider the recovery of extrinsic calibration parameters, independent of the type of sensor. We require only that each sensor produce time-stamped data from which its 6-DOF (or planar 3-DOF) incremental motion can be estimated. This incremental motion could be recovered, for example, from image, laser, or IMU+odometry.

   If an uncertainty covariance for these motions is available, it can be incorporated into our estimation procedure. The resulting ability to recover the rigid body transforms between different types of sensors is important because many robotic systems employ a variety of sensors.

2. **Automatic parameter recovery.** No specific augmentation of the environment or manual measurement is required or assumed. Many calibration techniques rely on target objects with known characteristics in the sensor's field of view (e.g., fiducials or fixtures). Our calibration procedure, on the other hand, uses only per-sensor ego-motion observations which can often be estimated without targets. For example, scan-matching can be used to provide incremental poses without relying on a prepared environment. The result is a greatly simplified and user-independent calibration process.

3. **Global reference frame not required.** The most common form of calibration proceeds by establishing a global reference frame for each sensor, typically through some form of SLAM or the use of an external network (e.g., GPS). With each sensor localized in a global frame, the rigid body transform best aligning the two frames is the extrinsic calibration. Unfortunately, this method requires either solving the complex SLAM problem before calibration or relying on an external network (which is not always available). Our solution uses only incremental motion and does not require SLAM or an external network.

4. **Asynchronous sensor data supported.** Sensors are rarely synchronized (i.e., they do not have the same sampling frequency or phase). Our estimator, however, requires incremental motions observed over a common period. Although we can trivially interpolate between observations, we want to do so

according to the uncertainties of the observations. As a result, we use the Scaled Unscented Transform (SUT) [40] to interpolate sensor data and produce corresponding, resampled noise estimates.

5. **Principled noise handling.** Most calibration techniques assume all data is equally accurate, and produce a single set of calibration parameters. A distinct aspect of our calibration algorithm is that it not only accepts both observations and their covariances, but also generates a best-case uncertainty estimate for the calibration parameters. This is an important feature, as degeneracies in the observations have the potential to generate inaccurate calibration parameters if ignored. This inaccuracy can be unbounded due to the degeneracies. By examining the output covariance matrix for singularities, such situations can be avoided.

   Furthermore, the calibration is an often ignored source of uncertainty in the system. Small errors in the calibration rotations, for example, can produce significant errors for re-projected sensor data (especially at large distances). By providing an accurate uncertainty estimate, these ambiguities can be accounted for during sensor fusion in a principled way.

2.2 Estimation and Information Theory

In an estimation process, we desire to recover some notion of a true value having made some observations. Estimation theory provides many frameworks enabling estimates to be recovered from observations. Parameter estimation theory, for example, deals with the recovery of values which do not change over time. State estimation theory, on the other hand, enables time-varying values to be recovered. Our sensor calibration task is to estimate a time-invariant, rigid body transform. Accordingly, we focus here only on parameter estimation. (By contrast, the articulated object tracking work in the next chapter performs state estimation.) As we design an algorithm to produce an estimate, we can rely on techniques from information theory to assess the quality of our estimator. Information theory provides tools to answer questions such as:

1. Is it even possible to construct an estimator? We might wish to recover calibration parameters, but if, for example, our observations are of unrelated quantities (e.g., the weather conditions), we have little hope of success.

2. When will our estimator fail? Even if observations have the potential to enable estimation, there may be special alignments that do not allow it.

3. How good is the output estimate? Assuming it is possible to recover the parameters, we desire some measure of their quality (e.g., variance).

2.2.1 Maximum Likelihood & Maximum *a posteriori*

Suppose we wish to recover some estimate of parameters, x, given some set of observations, z. For the calibration task, x and z are vectors encoding a series of rigid body transformations. Classically, this problem has been approached from two vantage points: Bayesian and Non-Bayesian. In the Bayesian approach, the parameters are considered to be random values drawn from some probability distribution, $P(x)$. In the Non-Bayesian approach, an unknown true value of the parameters, x_0, is considered to exist.

In the Bayesian approach, we wish to produce an estimate, $\hat{x}$, of the parameters which is most likely given the observations. This is known as maximum *a posteriori* estimation (MAP):

$$\hat{x}^{MAP}(z) = \underset{x}{\operatorname{argmax}} P(x \mid z) \tag{2.1}$$

The probability of the parameters given the data, $P(x \mid z)$, is often difficult to compute because it requires solving an inverse problem (computing parameters from observations). Fortunately, Bayes' rule can be employed:

$$\hat{x}^{MAP}(z) = \underset{x}{\operatorname{argmax}} \frac{P(z \mid x) P(x)}{P(z)} \tag{2.2}$$

$$= \underset{x}{\operatorname{argmax}} P(z \mid x) P(x) \tag{2.3}$$

The forward model, $P(z \mid x)$, represents the probability of some observations given the parameters. Notice that $P(z)$, the probability of a particular set of observations, does not affect the maximization. The probability of some parameters, $P(x)$, on the other hand, requires more careful consideration. From a non-Bayesian approach, $P(x)$ can be considered uniform and removed from the optimization — essentially, every parameter x is considered equally likely. This is known as the maximum likelihood estimator:

$$\hat{x}^{ML}(z) = \underset{x}{\operatorname{argmax}} P(z|x) \tag{2.4}$$

In the MAP, $P(x)$ is modeled explicitly. For example, a Gaussian distribution might be used to calculate the probability of a particular set of calibration parameters. Thus, the MAP and ML estimators differ only in the assumption they make regarding the uniformity of $P(x)$.

2.2.2 The Cramer-Rao Lower Bound

As we might expect, noise at the inputs of the estimator will cause noise at the output. The CRLB describes a lower limit on the noise at the output. That is, if the covariance on the output of an estimator reaches the CRLB, the estimator has

extracted all available information from the input data. The remaining noise is a function only of the noise on the inputs and cannot be reduced further.

The derivation of the CRLB depends on an assumption that the estimate is unbiased (for our calibration problem, we address the unbiased nature of our estimator in Section 2.4.4). If the estimator is unbiased, then

$$E\left[\hat{x}\left(z\right)-x\right]=\int\left(\hat{x}\left(z\right)-x\right)P\left(z|x\right)dz=0 \tag{2.5}$$

Building on this fact, we follow the derivation of the CRLB in [71] for scalar states and observations (later, we extend to the vector case). Taking the derivative of both sides and assuming the derivative of the integrand is integrable, yields:

$$\frac{\partial}{\partial x}\int\left(\hat{x}\left(z\right)-x\right)P\left(z|x\right)dz=\frac{\partial}{\partial x}0 \tag{2.6}$$

$$\int\frac{\partial}{\partial x}\left(\left(\hat{x}\left(z\right)-x\right)P\left(z|x\right)\right)dz=0 \tag{2.7}$$

$$\int-P\left(z|x\right)+\left(\hat{x}\left(z\right)-x\right)\frac{\partial}{\partial x}P\left(z|x\right)dz=0 \tag{2.8}$$

Since $P\left(z|x\right)$ is a distribution which must integrate to unity, $\int P\left(z|x\right)dz=1$, and

$$\int\left(\hat{x}\left(z\right)-x\right)\frac{\partial}{\partial x}P\left(z|x\right)dz=1 \tag{2.9}$$

Noting the definition for the derivative involving a natural logarithm,

$$\frac{\partial\ln y}{\partial x}=\frac{1}{y}\frac{\partial y}{\partial x}\Rightarrow\frac{\partial y}{\partial x}=y\frac{\partial\ln y}{\partial x}\Rightarrow\frac{\partial P\left(z|x\right)}{\partial x}=P\left(z|x\right)\frac{\partial\ln P\left(z|x\right)}{\partial x} \tag{2.10}$$

Then, plugging in to Equation 2.9:

$$\int\left(\hat{x}\left(z\right)-x\right)P\left(z|x\right)\frac{\partial\ln P\left(z|x\right)}{\partial x}dz=1 \tag{2.11}$$

The Schwarz inequality, $\left(\int f\left(z\right)g(z)\,dz\right)^{2}\leq\int f(z)^{2}\,dz\cdot\int g(z)^{2}\,dz$ can be applied if we factor $P\left(z|x\right)$:

$$\int\underbrace{\left(\left(\hat{x}\left(z\right)-x\right)\sqrt{P\left(z|x\right)}\right)}_{=f(z)}\underbrace{\left(\sqrt{P\left(z|x\right)}\frac{\partial\ln P\left(z|x\right)}{\partial x}\right)}_{=g(z)}dz=1 \tag{2.12}$$

$$\left(\int f\left(z\right)g(z)\,dz\right)^{2}=1 \tag{2.13}$$

$$1\leq\int\left(\left(\hat{x}\left(z\right)-x\right)\sqrt{P\left(z|x\right)}\right)^{2}dz\cdot\int\left(\sqrt{P\left(z|x\right)}\frac{\partial\ln P\left(z|x\right)}{\partial x}\right)^{2}dz \tag{2.14}$$

The first term to the right of the inequality is the variance of the estimate:

$$\int \left(\left(\hat{x}\left(z \right) - x \right) \sqrt{P\left(z|x \right)} \right)^2 dz = \int \left(\hat{x}\left(z \right) - x \right)^2 P\left(z|x \right)\, dz = E\left[\left(\hat{x}\left(z \right) - x \right)^2 \right] \quad (2.15)$$

And, similarly, for the second term to the right of the inequality:

$$\int \left(\sqrt{P\left(z|x \right)} \frac{\partial \ln P\left(z|x \right)}{\partial x} \right)^2 dz = \int \left(\frac{\partial \ln P\left(z|x \right)}{\partial x} \right)^2 P\left(z|x \right)\, dz \quad (2.16)$$

$$= E\left[\left(\frac{\partial \ln P\left(z|x \right)}{\partial x} \right)^2 \right] \quad (2.17)$$

Substituting,

$$1 \leq E\left[\left(\hat{x}\left(z \right) - x \right)^2 \right] \cdot E\left[\left(\frac{\partial \ln P\left(z|x \right)}{\partial x} \right)^2 \right] \quad (2.18)$$

$$\mathrm{var}\left(\hat{x}\left(z \right) \right) \; = E\left[\left(\hat{x}\left(z \right) - x \right)^2 \right] \geq E\left[\left(\frac{\partial \ln P\left(z|x \right)}{\partial x} \right)^2 \right]^{-1} \quad (2.19)$$

We see then that the uncertainty of the estimate, $\mathrm{var}\left(\hat{x}\left(z \right) \right)$, can be no less than: $J = E\left[\left(\frac{\partial \ln P(z|x)}{\partial x} \right)^2 \right]^{-1}$.

When x and z are vector quantities, the Fisher Information Matrix (FIM), J, is written as:

$$J = \mathrm{E}\left[\left(\nabla_x \ln P\left(z|x \right) \right) \left(\nabla_x \ln P\left(z|x \right) \right)^T \right] \quad (2.20)$$

and $\mathrm{var}\left(\hat{x}\left(z \right) \right) \; \geq J$. Here, as in [2], we abuse notation slightly: $A \geq B$ is interpreted as $A - B \geq 0$, meaning that the matrix $A - B$ is positive semi-definite. The ∇_x operator calculates the gradient of the operand along the direction of x.

We see immediately that if the FIM is non-invertible, then the variance on one or more of our parameters will be unbounded. As a somewhat oversimplified analogy, consider J as a scalar; if $J = 2$, then the variance must be greater than $1/2$. If $J = 0$ (corresponding to a rank deficient matrix J), then the variance is infinite. An infinite variance means that nothing is known about the parameter. Thus — at least intuitively — we can see that if J is rank deficient, we will not be able to estimate x.

Throughout the following sections, we will use several terms. First, a system is said to be *observable* if the corresponding FIM is full rank. This means that a system is observable only if all the output parameters can be determined from the inputs. Later, we will show that, in our setting, the calibration parameters are observable from incremental pose inputs.

An estimator is *efficient* if it achieves the CRLB. It is not generally possible to show that non-linear transforms (e.g., such as our rigid body transformations) result in estimators which are efficient [71, p71]. However, as demonstrated in Section 2.4.2, our estimator comes close to the CRLB.

2.3 Background

Ideally, it would be easy to determine the transformation relating any pair of sensors. One could use a ruler to measure the translational offset, and a protractor to measure the orientation offset. In practice, it is rarely so easy. A sensor's coordinate origin typically lies inside the body of the sensor, inaccessible to direct measurement. Curved and textured sensor housings, and obstructions due to the robot itself, usually make direct measurements of sensor offsets difficult or impossible.

Alternatively, one could design and fabricate precision sensor mounts and eliminate the calibration problem. While this approach may be practical for small, closely-spaced sensors such as stereo rigs, it would be cumbersome and costly for larger systems. Moreover, the need for machined mounts would hinder field adjustment of sensor configurations.

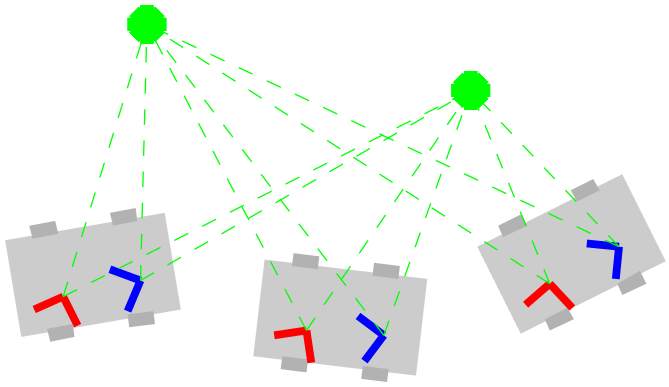


Figure 2-3: Calibration solutions requiring SLAM attempt to establish a common reference frame for the sensors (blue and red) using landmarks (green).

In light of these practical considerations, we desire a software-only algorithm to estimate calibration using only data easy to collect in the field. One approach would be to establish a common global reference frame, then estimate the absolute pose of each sensor at every sample time relative to that frame. The calibration would then be the transformation that best aligns these poses. For example, in the setting of Figure 2-3, the task would be to find the calibration that aligns the paths of the blue and red sensors, having been estimated from some set of landmarks (green). In practice, however, recovering accurate absolute sensor pose is difficult, either because of the effort required to establish a common frame (e.g., through SLAM), or the error that accumulates (e.g., from odometry or inertial drift) when expressing all robot motions in a common frame.

Any calibration method employing global, metrical reconstruction will necessarily invoke SLAM as a subroutine, thus introducing all the well-known challenges of solving SLAM robustly (including data association, covariance estimation, filter consistency, and loop closure).

Researchers have, nevertheless, attempted to recover absolute pose through various localization methods. Jones [39] and Kelly [45] added calibration parameters to the state of an Extended Kalman Filter and an Unscented Kalman Filter (UKF), respectively. Although the calibration parameters are static, they are expressed as part of a dynamic state. Recognizing this, the authors fix the values after some initial period and prevent further updates. Jones [39] also shows observability; a comparison with our work is provided in Section 2.4.2.4.

Gao [28] proposed a Kalman filter incorporating GPS, IMU, and odometry to estimate a robot's pose. They place pairs of reflective tape, separated by known distances, throughout the environment. These reflectors are easily identified in the LIDAR returns and they optimize over the 6-DOF sensor pose to align the reflector returns. To account for GPS drift, they add DOFs to the optimization allowing the robot to translate at several points during its travel.

Levinson [48] recovered calibration through alignment of sensed 3D surfaces generated from a multi-beam (e.g., Velodyne) LIDAR. Here, the robot's pose is required (e.g., via SLAM) to accumulate sufficiently large surfaces. They extended their work in [49] to calibrate 3D LIDARs and cameras. They do so by correlating depth discontinuities in the 3D LIDAR data to pixel edges in the images.

Maddern [51] recovers the calibration parameters by minimizing the entropy of point cloud distributions. The authors compute a Gaussian Mixture Model for 3D LIDAR and 2D LIDAR points, then attempt to minimize the Rényi entropy measure for the point cloud. The technique requires a global estimate of the robot's position (e.g., via GPS), but they minimize long-term drift by performing the optimization on subsets of the data.

When it is possible to augment or prepare the environment, absolute pose can also be established with known, calibrated landmarks. For example, Blanco [4] provides sensor datasets with known calibrations to the community. The calibrations for the GPS, LIDARs, and cameras are recovered automatically. For the GPS and LIDAR data, they use SLAM for each sensor and then optimize for the calibrations. For the camera, they rely on known, hand-selected 3D landmarks in the environment. Similarly, Ceriani [15] used GPS and calibrated rigs to establish a global sensor reference frame.

As in our work, others have also used incremental poses for calibration. Censi [14] optimizes for several intrinsic parameters (e.g., robot wheel base) and a 3-DOF calibration. The authors use scan-matching to determine an incremental pose, then optimize for the robot's wheel base and a 3-DOF 2D LIDAR calibration. Our method, however, is distinct in that it accounts for observation noise in a principled way.

Boumal [6] investigates recovering incremental rotations. The authors use two Langevin distributions, a rotationally symmetric distribution designed for spherically constrained quantities: one for inliers and one for outliers. Using an initial guess based on the SVD, they show good robustness against outliers.

Incremental motions have also been used to recover "hand-eye calibration" parameters. The authors in [34, 21, 17, 72] recover the calibration between an end-effector

and an adjacent camera by commanding the end-effector to move with some known velocity and estimating the camera motion. The degenerate conditions in [17, 72] are established through geometric arguments. We confirm their results via information theory and the CRLB. Further, the use of the CRLB allows our algorithm to identify a set of observations as degenerate, or nearly degenerate (resulting in a large variance), in practice. Dual quaternions have been used elsewhere [21]. We extend this notion and explicitly model the system noise via a projected Gaussian.

2.4 3-DOF Calibration

This section considers the problem of recovering 2D (3-DOF) calibration parameters. After formally stating the calibration problem (Section 2.4.1), we show that calibration is observable (Section 2.4.2). We describe the estimation process and bias computation in Section 2.4.3 and Section 2.4.4, respectively. In practice, one typically must (1) interpolate incremental poses (Section 2.4.5) and their associated covariances for asynchronous sensor data and (2) reliably estimate the covariances for relative pose observations for common sensors such as LIDARs (Section 2.4.7). Using simulated and real data from commodity LIDAR and inertial sensors, we demonstrate calibrations accurate to millimeters and milliradians (Section 2.4.8).

Later, in Section 2.5, we consider the full 3D (6-DOF) calibration process. We present the 2D process first, however, because the mathematical analysis, optimization techniques, and noise model are significantly easier to understand in the 2D case. This section ignores the non-Euclidean nature of SE(2) for simplicity. Later, when we address such issues for SE(3), we realize that similar adjustments are required for SE(2).

2.4.1 Problem Statement

Our task is to estimate the static calibration parameter vector $k = [x_s, y_s, \theta_s]$ representing the translational and rotational offsets from one sensor, r, to a second sensor, s. Figure 2-4 shows the graphical model [46] relating the variables of interest, with v_{ri} the latent, true incremental poses of sensor r at time i and z_{ri} and z_{si} the corresponding observed incremental poses of the sensors. As indicated in the graphical model, the observed incremental poses of both sensors depend only on the calibration and the true motion of one of the sensors. As a result, the true motion of the other sensor, v_{si}, need not be estimated.

Figure 2-5 illustrates the system input. In this example, the r and s sensors travel with the planar robot (left), producing $N = 3$ incremental pose observations (right). We define $g(\cdot)$ to relate the true motion of sensor s, v_{si}, to v_{ri} and the rigid body transform k:

$$v_{si} = g\left(v_{ri}, k\right) \tag{2.21}$$

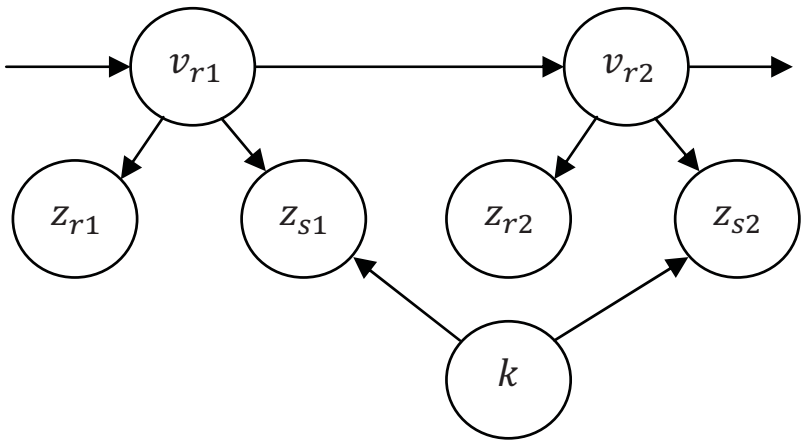


Figure 2-4: Graphical model of calibration and incremental poses is shown.

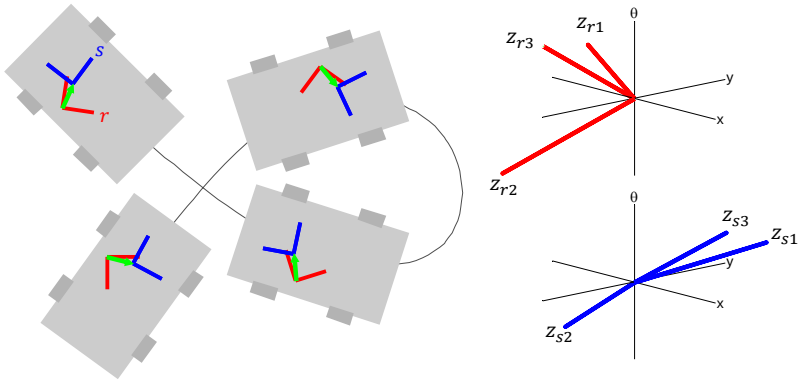


Figure 2-5: Incremental poses for the r (red) and s (blue) sensors are observed as the robot travels. The observations are plotted on the right in x-y-θ coordinates. The true calibration parameters will transform the red and blue observations into alignment.

Again, the true motion of sensor s need not be estimated because it can be calculated from the other estimated quantities. Further, let $z_r = [z_{r1}, z_{r2}, \cdots, z_{rN}]$, $z_s = [z_{s1}, z_{s2}, \cdots, z_{sN}]$, and $z = [z_r, z_s]$.

Each measurement for the r sensor, z_{ri}, is a direct observation of the latent variable v_{ri}. Each measurement for the s sensor, z_{si}, is an observation based on its true motion, v_{si}. If the $\circ$ operator is used to compound rigid body transformations, then:

$$v_{si} = g\left(v_{ri}, k\right) = k^{-1} \circ v_{ri} \circ k \tag{2.22}$$

Essentially, the $g(\cdot)$ function transforms the incremental motion from the r frame into the s via rotations.

Although we are interested only in the calibration parameters k, the true incremental poses $v_r = [v_{r1}, v_{r2}, \cdots, v_{rN}]$ at every sample time are also unknown variables. The parameters to estimate are $x = [v_r, k] = [v_{r1}, v_{r2}, \cdots, v_{rN}, x_s, y_s, \theta_s]$. This state vector can grow rapidly, adding three terms (two translations and one rotation) for

each observation. However, as we demonstrate with real data in Section 2.4.8, the estimates converge quickly with manageable N; our experiments used $N = 88$ and $N = 400$.

Finally, we define the equations relating the expected observations and true values as:

$$G\left(v_r, k\right) = \begin{bmatrix} v_{r1} & \cdots & v_{rN} & g(v_{r1}, k) & \cdots & g(v_{rN}, k) \end{bmatrix}^T \tag{2.23}$$

Note that G has dimension $6N \times 1$.

2.4.2 Observability & the Cramer-Rao Lower Bound

Before attempting to recover the calibration from the observed incremental pose data, we must first show that there exists sufficient information in the observed data to determine the calibration parameters. As described in Section 2.2, such observability is guaranteed when the the CRLB is full rank.

2.4.2.1 The FIM and the Jacobian

If we assume an additive Gaussian noise model, then:

$$z = G(x) + \delta \tag{2.24}$$

where $\delta \sim N(0, \Sigma_z)$. Alternatively, we can write that $P(z|x) \sim N\left(G(x), \Sigma_z\right)$ and apply Equation 2.20 in order to determine the FIM for the calibration parameters. We assume that the observation covariance, Σ_z, is full rank. That is, we assume that all dimensions of the observations are actually measured.

Let $y = z - G\left(x\right)$, then $\nabla_{\mathrm{x}} y = -J_G$, where J_G is the Jacobian of G. Note that J_G is a function of x, but we eliminate the parameter for simplicity. Then

$$\begin{aligned} \nabla_{\mathrm{x}} \ln P\left(z \mid x\right) &= \nabla_{\mathrm{x}} \ln \left[c \exp\left(\frac{-1}{2} \left(y^T \Sigma_z^{-1} y \right) \right) \right] && (2.25) \\ &= \nabla_{\mathrm{x}} \ln\left(c\right) + \ln \exp\left(\frac{-1}{2} \left(y^T \Sigma_z^{-1} y \right) \right) && (2.26) \\ &= \nabla_{\mathrm{x}} \frac{-1}{2} \left(y^T \Sigma_z^{-1} y \right) && (2.27) \\ &= J_G^T \Sigma_z^{-1} y && (2.28) \end{aligned}$$

Substituting into Equation 2.20 and proceeding as in [75]:

$$J = E\left[\left(\nabla_{\mathrm{x}} \ln P\left(z \mid x\right)\right)\left(\nabla_{\mathrm{x}} \ln P\left(z \mid x\right)\right)^{T}\right] \tag{2.29}$$

$$= E\left[\left(J_G^T\ \Sigma_z^{-1} y\ \right)\left(J_G^T\ \Sigma_z^{-1} y\ \right)^{T}\right] \tag{2.30}$$

$$= E\left[J_G^T\ \Sigma_z^{-1} y y^T \left(\Sigma_z^{-1}\right)^T J_{G}\right] \tag{2.31}$$

$$= J_G^T\ \Sigma_z^{-1} E\left[y y^T\right]\left(\Sigma_z^{-1}\right)^T J_{G} \tag{2.32}$$

$$= J_G^T\ \Sigma_z^{-1} \Sigma_z \left(\Sigma_z^{-1}\right)^T J_{G} \tag{2.33}$$

$$= J_G^T\ \left(\Sigma_z^{-1}\right)^T J_{G} \tag{2.34}$$

$$= J_G^T\ \Sigma_z^{-1}\ J_{G} \tag{2.35}$$

In Equation 2.32, the expectation is over the observations z, so all the terms except yy^T factor. In Equation 2.33, we make use of the fact that $E\left[yy^T\right] = \Sigma_z$ by definition.

As we examine the degenerate conditions, we will be interested in the rank(J). If we factor Σ_z using, for example, the Cholesky factorization, such that $\Sigma_z^{-1} = LL^T$, then:

$$\mathrm{rank}(J) = \mathrm{rank}\left(J_G^T\ \Sigma_z^{-1}\ J_{G}\right) \tag{2.36}$$

$$= \mathrm{rank}\left(J_G^T L L^T J_G\right) \tag{2.37}$$

$$= \mathrm{rank}\left((L^T J_G)^T (L^T J_G)\right) \tag{2.38}$$

$$= \mathrm{rank}\left(L^T J_G\right) \tag{2.39}$$

$$= \mathrm{rank}(J_G) \tag{2.40}$$

We can restate Equation 2.35 as follows: if the observation noise has a Gaussian distribution, then the FIM can be calculated as a function of the observation Jacobian and observation covariance. This has two important implications:

1. Given the covariance and having computed the Jacobian, we have the quantities needed to calculate a lower bound on the noise of our estimate (Equation 2.35).

2. As shown in Equation 2.40, J has full rank if and only if J_G has full column rank. Thus, by analyzing the column rank of J_G, we can determine whether the calibration is observable.

2.4.2.2 The FIM Rank

First, consider the case where there is one incremental pose observation for each sensor, i.e., $z = [z_{r1}, z_{s1}] = [z_{r1x}, z_{r1y}, z_{r1\theta}, z_{s1x}, z_{s1y}, z_{s1\theta}]$. The parameter vector is then $x = [v_{r1}, k] = [v_{r1x}, v_{r1y}, v_{r1\theta}, x_s, y_s, \theta_s]$. The resulting 6×6 Jacobian matrix of Equation 2.23 has the form below. This matrix is clearly rank-deficient because rows

3 and 6 are the same (the elided values denoted by $\cdots$ are not relevant).

$$J_{G1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.41}$$

Second, consider the case where there are two incremental pose observations. The observation vector is $z = [z_{r1}, z_{r2}, z_{s1}, z_{s2}]$. The parameter vector is then $x = [v_{r1}, v_{r2}, k]$. The Jacobian is now 12×9 and has the form:

$$J_{G2} = \begin{bmatrix} I_{6\times6} & 0_{6\times3} \\ \cdots & M_{6\times3} \end{bmatrix} \tag{2.42}$$

Due to the 6×6 identity matrix, the first 6 columns are linearly independent. In order to show that J_{G2} has full rank (i.e., rank 9), we need only understand — and avoid — the conditions under which the columns of M become linearly dependent.

2.4.2.3 Conditions for FIM Rank-Deficiency

When a and b can be found such that $aM_1 + bM_2 + M_3 = 0$, where M_i is the i-th column of M, J_{G2} will be rank deficient. With $c_s = cos(\theta_s)$, $c_{1s} = cos(v_{r1\theta} - \theta_s)$, $x_1 = v_{r1x}$, etc.,

$$M = \begin{bmatrix} c_{1s} - c_s & -s_{1s} - s_s & y_s c_{1s} + (y_1 - y_s)c_s + x_s s_{1s} - (x_1 - x_s)s_s \\ s_{1s} + s_s & c_{1s} - c_s & -x_s c_{1s} - (x_1 - x_s)c_s + y_s s_{1s} - (y_1 - y_s)s_s \\ 0 & 0 & 0 \\ c_{2s} - c_s & -s_{2s} - s_s & y_s c_{2s} + (y_2 - y_s)c_s + x_s s_{2s} - (x_2 - x_s)s_s \\ s_{2s} + s_s & c_{2s} - c_s & -x_s c_{2s} - (x_2 - x_s)c_s + y_s s_{2s} - (y_2 - y_s)s_s \\ 0 & 0 & 0 \end{bmatrix} \tag{2.43}$$

Since the columns of M cannot be eliminated in general, the system is observable. However, it is useful to further consider which specific kinds of observations will cause M to reduce. By knowing the characteristics of paths that do not admit calibration, we can avoid traversing them in practice. There are several types of path degeneracies:

1. Paths for which each sensor observes constant incremental poses. In this case, $\text{rank}(J_{GN}) = \text{rank}(J_{G1})$.

2. Paths for which the robot remains at a fixed orientation with respect to some arbitrary world frame.

3. Paths for which both sensors move on concentric circles.

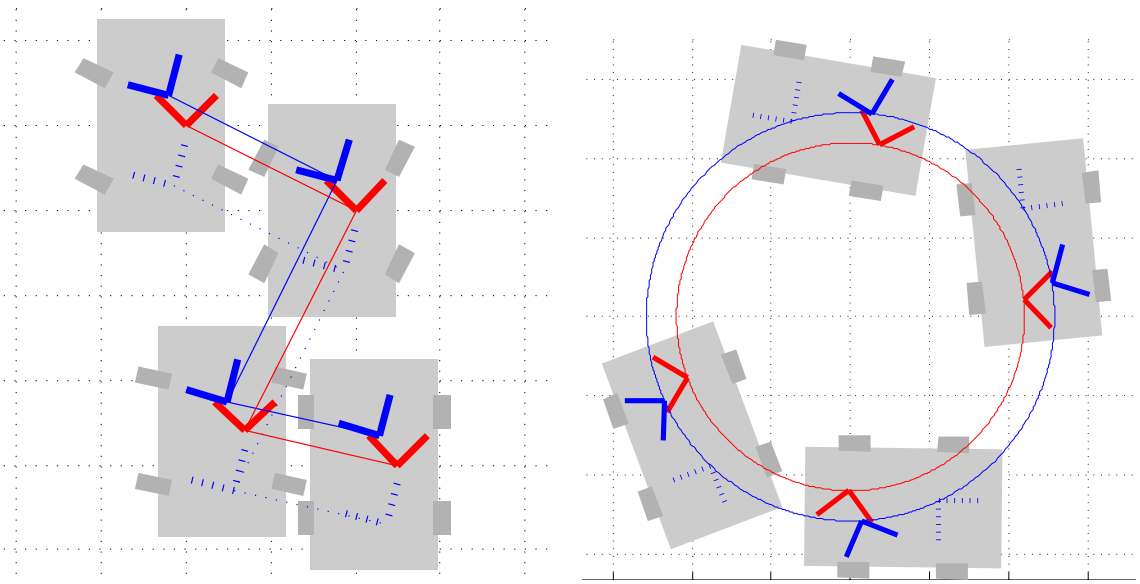


Figure 2-6: Sensor movements without rotation, as for a translating holonomic robot (which can move in any dimension regardless of state) (a), can prevent calibration. (The dotted blue lines show blue sensor poses corresponding to alternative calibration parameters.) The calibration of a non-holonomic robot (e.g., an automobile) cannot be recovered if the robot is driven in a straight line. Concentric, circular sensor motion (b) can also prevent calibration.

The first case is relatively uninteresting and can be easily avoided in practice by varying the robot's velocity. The second case arises if the robot travels such that the sensors experience only translation (Figure 2-6a). In the third case, infinitely many recovered poses for sensor s (e.g., lying along the blue circle in Figure 2-6b) will be consistent with the observed incremental poses. (To see this, compare the relative offsets between pairs of blue sensor frames and pairs of dotted blue sensor frames.) In practice, speed variation alone will not prevent degeneracy; it is the undesirable path geometry itself that must be avoided.

2.4.2.4 Other Methods of Showing Observability

Martinelli [52] and Kelly [45] added the static calibration parameters to the dynamic state vector of a Kalman filter. In order to address the question of observability, they used non-linear observability analysis (instead of the FIM) based on Lie derivatives. In their formulations, the robot pose (relative to some arbitrary world frame) is part of the state vector. However, because most sensors make only relative observations, the state is not observable, i.e., there exists an unobservable transform between the robot poses and the world frame. Martinelli overcame this limitation by removing the unobservable portions of the robot pose from the state vector; Kelly overcame this limitation by assuming that the sensors can provide world-relative observations.

In our formulation, the parameter vector includes only relative motions, so we need not make any simplifications or assumptions to show observability.

2.4.3 Estimation

Using the MAP estimator, Equation 2.3, and applying the independence assumptions implied by the graphical model in Figure 2-4 yields:

$$\hat{x}^{MAP}(z) = \underset{x=[v_r,k]}{\operatorname{argmax}} P(v_{r0})P(k)\prod_{i=1}^{N} P(z_{ri}|v_{ri})P(z_{si}|v_{ri},k)P(v_{ri}|v_{r,i-1}) \tag{2.44}$$

In many cases, it may be reasonable to assume that nothing is known about the distribution of the first incremental pose of the sensor, $P(v_{r0})$. If acceleration data or a motion model is available, then the way in which the incremental poses relate, expressed in $P(v_{ri}|v_{r,i-1})$, can be estimated. Similarly, if there is a good initial guess about the calibration itself, then it might be reasonable to assume a Gaussian distribution centered about the guess, reflected in $P(k)$. If a reasonable value for one or more of these distributions cannot be justified, however, then a uniform distribution may be most appropriate. If $P(v_{r0})$, $P(v_{ri}|v_{r,i-1})$, and $P(k)$ are distributed uniformly, they can be removed from the maximization, and Equation 2.44 becomes the ML estimator (c.f., Equation 2.4):

$$\hat{x}^{ML}(z) = \underset{x=[v_r,k]}{\operatorname{argmax}} \prod_{i=1}^{N} P(z_{ri}|v_{ri})P(z_{si}|v_{ri},k) \tag{2.45}$$

In our experiments (Section 2.4.8), $P(v_{r0})$ and $P(k)$ are assumed to be uniformly distributed because we do not wish to assume any domain-specific prior information. For example, we do not wish to assume an initial guess of k is available to serve as the mean for a Gaussian distribution for $P(k)$. We also found that assuming $P(v_{ri}|v_{r,i-1})$ to be uniformly distributed was reasonable. This is because we sampled our time steps to be sufficiently large that the incremental motion during time $i-1$ was independent of the incremental motion during time i. (If a good acceleration model were available, or the robot moved more slowly, this assumption could be altered to produce better estimates.)

When the incremental pose observations for sensor r and s, as reflected in $P(z_{ri}|v_{ri})$ and $P(z_{si}|v_{ri},k)$, are normally distributed, the optimization becomes a non-linear Least Squares Estimation (LSE). To see this, let:

$$\Sigma = \text{blkdiag}\left([\Sigma_{r1},\cdots,\Sigma_{rN},\Sigma_{s1},\cdots,\Sigma_{sN}]\right) \tag{2.46}$$

where $P(z_{ri}|v_{ri}) = N(z_{ri};v_{ri},\Sigma_{ri})$ and $P(z_{si}|v_{ri},k) = N(z_{si};g(v_{ri},k),\Sigma_{si})$. The blkdiag$(\cdot)$ function arranges the 3×3 covariance matrices along the diagonal of the resulting

$6N \times 6N$ covariance Σ. The LSE can then be found by minimizing:

$$\begin{aligned} \Delta &= z - G(v_r, k) \\ c(x, z) &= \Delta^T \Sigma^{-1} \Delta \\ \hat{x}^{LSE}(z) &= \underset{x=[v_r,k]}{\operatorname{argmin}} \left(c(x, z)\right) \end{aligned} \tag{2.47}$$

The LSE cost function $c(x, z)$ returns a single number that can be minimized by many solvers (e.g., Matlab's `fminunc` function). In practice, we found that solvers which minimize the individual errors directly (e.g., Matlab's `lsqnonlin`), rather than their summed squares, converge faster. If $\Sigma^{-1} = \Psi\Psi^T$, where Ψ is found e.g. via the lower Cholesky factorization, then

$$c(x, z) = \Delta^T \Psi \Psi^T \Delta = (\Psi^T \Delta)^T (\Psi^T \Delta) \tag{2.48}$$

The individual errors are now represented in the $6N \times 1$ vector $\Psi^T \Delta$ and solvers such as `lsqnonlin` can be used.

2.4.4 Evaluating Bias

The nonlinear LSE is not necessarily unbiased. That is, the estimate $\hat{x}$ may exhibit an offset from the true value x_0. Using a technique developed by Box [7], the bias can be approximated and, if significant, subtracted from the estimate. We (approximately) evaluate the bias, rather than assume that it is small. In our simulations and experiments, we indeed found it to be negligible compared to the CRLB.

In order to calculate an expected value for the bias, Box approximates it with a second-order Taylor series:

$$\begin{aligned} \mathrm{E}\left[\hat{x} - x_0\right] &= \left(\tfrac{-1}{2}\right)\ J^{-1}\ J_G^T\ \Sigma^{-1}\ m \\ m &= \begin{bmatrix} \operatorname{trace}(H_1 J^{-1}) & \cdots & \operatorname{trace}(H_{6N} J^{-1}) \end{bmatrix}^T \end{aligned} \tag{2.49}$$

where J is the FIM, J_G is the Jacobian, and H_i is the Hessian (second derivative) of the i-th observation (i.e., the Hessian of the i-th row in the G matrix). Note that both the bias and the CRLB are driven smaller as $\det(J)$ increases, where J again represents the amount of information in the observations. Thus, as the CRLB of the calibration is reduced, so to is the bias.

2.4.5 Interpolation

One requirement of our algorithm is that the relative motions z_r and z_s be observed at identical times. In practice, however, sensors are rarely synchronized. We must, therefore, interpolate observations and their associated covariance matrices to a common set of sample times. As shown in Figure 2-7, a sensor has traveled along the blue path with several relative motions (blue arrows). In order to synchronize with a second sensor, for example, we wish to calculate the relative motions (black arrows)

and covariances at new times. Although other researchers have developed techniques for interpolating between two covariances [25], we could find no previous work related to interpolating both incremental poses and their covariances. Our key contribution is that the interpolation can be formulated as a function; then, using a sigma point method similar to that used in the UKF, we can estimate resulting motions and covariances.

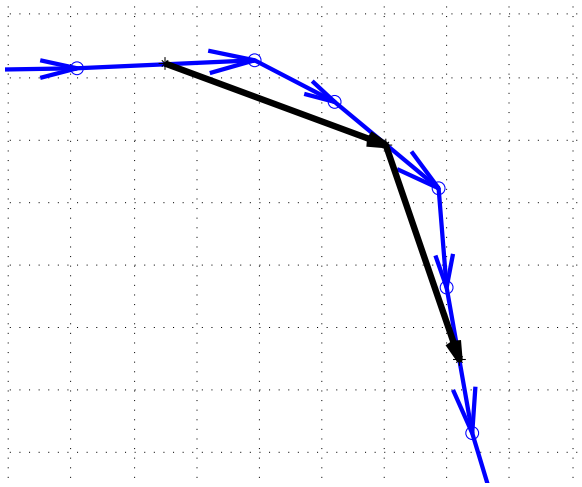


Figure 2-7: Resampling incremental poses (blue) at new time steps may make interpolated observations (black) dependent.

Our goal is to take the relative motions, $\overline{v_A}$, and the associated covariance Σ_A, sampled at corresponding times in set A, and interpolate (or resample) them to produce the mean relative motion $\overline{v_B}$ and covariance Σ_B at new times in set B. Note that Σ_A will often be band-diagonal; in other words, the incremental poses observed will be assumed to be independent. On the other hand, Σ_B will most likely not be band-diagonal, because new, adjacent interpolations in $\overline{v_B}$ will likely depend upon the same observation in $\overline{v_A}$ (c.f., Figure 2-7), and thus be correlated. Such new dependencies will occur unless $B \subseteq A$. That is, the incremental poses in $\overline{v_B}$ will depend on each other unless every sample time in B exists exactly in A.

When the observations of both sensors are resampled at common times, Σ in Equation 2.47 will take the form $\Sigma = \text{blkdiag}\left([\Sigma'_r, \Sigma'_s]\right)$, where Σ'_r and Σ'_s are the resampled covariances of r and s, respectively.

2.4.5.1 The 3-DOF Interpolation Function

As shown in Function 2.4.1, we construct a nonlinear interpolation function, f, of the form: $\mathtt{v}_B = f(A, \mathtt{v}_A, B)$, where $\mathtt{v}_A$ and $\mathtt{v}_B$ are $3N \times 1$ vectors of incremental pose observations. Notice that this function does not, by itself, estimate the new covariances; instead, it only interpolates the motion. There are many possible ways to design this function. In our case, we simply assume a constant-velocity model: `ConstantVelocityInterpolation` in Function 2.4.3 performs a linear inter-

polation/extrapolation between robot poses. Our implementation successively accumulates the relative motions into a common reference frame (Function 2.4.2), resamples these absolute poses using a naïve weighted average approach (Function 2.4.3), and then calculates the new relative motions (Function 2.4.4).

Function 2.4.1: $f(A,\mathtt{v}_A,B)$

inputs : A the N sample times of incremental motions
$\mathtt{v}_A$ the N incremental motions at times A
B the M sample times at which to resample
outputs: $\mathtt{v}_B$ the M incremental motions at times B

$x_A \leftarrow$ `AccumulateStates(`$\mathtt{v}_A$`)` `// accumulate incremental motions`
`// into a common reference frame`
$x_B \leftarrow$ `ResampleStates(`A, x_A, B`)` `// resample poses at new times`
$\mathtt{v}_B \leftarrow$ `MakeIncremental` (x_B) `// produce incremental motions`
return $\mathtt{v}_B$

Function 2.4.2: AccumulateStates($\mathtt{v}_A$)

inputs : $\mathtt{v}_A$ the N incremental motions at times A
outputs: x_A the $N+1$ accumulated states at times A

$x_A(0) \leftarrow I$ `// initialize global frame at origin`
for $i = 1$ **to** N **do**
  $d_i \leftarrow \mathtt{v}_A(i-1)$ `// i-th incremental motion`
  $x_A(i) \leftarrow x_A(i-1) \circ d_i$ `// accumulate`
return x_A

2.4.5.2 Means and Covariances

The next task is to determine how the mean and covariance are affected when propagated through the nonlinear function f. Fortunately, such a problem has already been addressed successfully by the Unscented Transform (UT). We do not use a UKF, but we employ the UT to estimate how the mean and covariance propagate through a nonlinear function. Whereas the UKF must propagate the state forward in time and predict measurements, we estimate only the mean and covariance of our relative measurements after interpolation.

To avoid sampling non-local effects due to a possibly large state space, we use the Scaled UT (SUT) [40]. Similar to the UT, the SUT estimates the mean and covariance by applying f to $2n+1$ "sigma points" (where n is the dimension of the space). These deterministic sigma points $\mathcal{X}_i$ are centered about the mean of, and

Function 2.4.3: ResampleStates(A, x_A, B)

inputs : A the N sample times of incremental motions
x_A the $N+1$ accumulated states at times A
B the M sample times at which to resample
outputs: x_B the $M+1$ accumulated states at times B

for $i = 1$ **to** M **do**
  `// interpolate the poses at each new time`
  $x_B(i) \leftarrow$ `ConstantVelocityInterpolation` (A, x_A, B_{i-1})
`// establish an initial sample based on the average sample period`
$x_B(0) \leftarrow$ `ConstantVelocityInterpolation` $(B, x_B, B_0 - \overline{\Delta B})$
return x_B

Function 2.4.4: MakeIncremental(x_B)

inputs : x_B the $M+1$ accumulated states at times B
outputs: v_B the M incremental motions at times B

for $i = 1$ **to** M **do**
  `// find the incremental motion between successive poses`
  $\mathsf{v}_B(i-1) \leftarrow (x_B(i-1))^{-1} \circ x_B(i)$
return v_B

discretely approximate, the original distribution:

$$\mathcal{X}_i = \begin{cases} \overline{v_A} & i = 0 \\ \overline{v_A} + \langle \sigma \rangle_i & 1 \le i \le n \\ \overline{v_A} - \langle \sigma \rangle_{i-n} & n+1 \le i \le 2n \end{cases} \tag{2.50}$$

where $\sigma = \sqrt{n\alpha^2 \Sigma_A}$ is calculated with the Cholesky factorization and we set $\alpha = 10^{-2}$ so as to mitigate extreme effects away from the mean [40]. The $\langle \cdot \rangle_i$ operator extracts the i-th column from its argument.

The sigma points are passed through the function, $\mathcal{Y}_i = f(A, \mathcal{X}_i, B)$, and the new mean and covariance are calculated:

$$\begin{aligned} \overline{v_B} &= \textstyle\sum_{i=0}^{2n} W_{m,i} \mathcal{Y}_i \\ \Sigma_B &= \textstyle\sum_{i=0}^{2n} W_{c,i} \left(\mathcal{Y}_i - \overline{v_B}\right) \left(\mathcal{Y}_i - \overline{v_B}\right)^T \end{aligned} \tag{2.51}$$

As in [40], we use the sigma point weights with $\beta = 2$:

$$\begin{aligned} W_{m,0} &= 1 - \tfrac{1}{\alpha^2} & W_{m,i} &= \tfrac{1}{2n\alpha^2} \\ W_{c,0} &= W_{m,0} + (\beta + 1 - \alpha^2) & W_{c,i} &= W_{m,i} \end{aligned} \tag{2.52}$$

Using this method, we can interpolate the mean and covariance of our incremental pose estimates at new times. It is important to remember, however, that the resulting covariances are only approximations. The interpolation process is nonlinear and, as a result, the transformed distribution (approximated by $\mathcal{Y}_i$) is not necessarily Gaussian, yet is forced to be so.

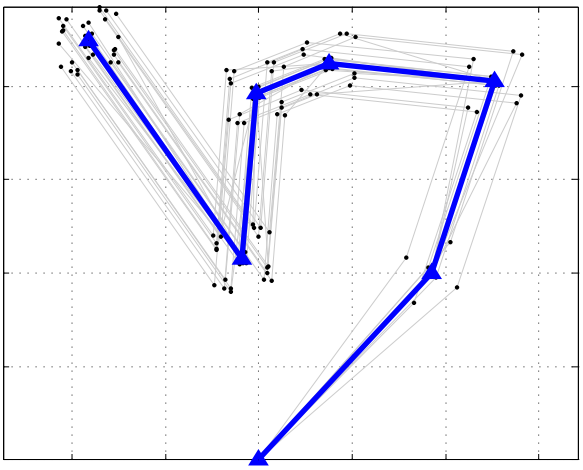


Figure 2-8: A robot travels along a mean path (blue) starting from the bottom and ending in the upper left. Gray lines show the sigma points, $\mathcal{X}_i$, each representing a path. Points associated with each sample time are shown as block dots.

2.4.5.3 Example

To illustrate the resampling process, consider a robot which has traveled with a mean path shown in blue in Figure 2-8. Each of the $\mathcal{X}_i$ sigma points (each corresponding to a robot path) is produced by simultaneously adding correlated noise to all incremental pose observations. In this case, the observed incremental poses are independent. As a result, the noise "accumulates" as the robot progresses (i.e., the paths mostly overlap at the start and diverge increasingly toward the end of the path).

These sigma points (the gray paths in Figure 2-8) are then resampled at the new times to produce the transformed sigma points (the gray paths in Figure 2-9). Incremental poses are determined between pairs of black dots at successive sample times and used to produce the mean and covariances for the incremental poses. Notice that the envelope of resampled sigma point paths follows the general envelope of the original sigma points. That is, the resampled uncertainty is an interpolation of the nearby original uncertainties. Finally, the two mean paths are compared in Figure 2-10. In this particular example, interpolation is shown between only pairs of points; however, the procedure also handles situations where more than two poses must be interpolated (i.e., the case were both the sampling frequency and phase are different).

2.4.6 The Algorithm

Algorithm 2.4.5 shows pseudo-code for the planar calibration, with `Optimize` some non-linear least-squares optimizer of the matrix-valued `Cost` function given below, and `JacobianOfG` the Jacobian of G (see Equation 2.23) evaluated at the estimate $\hat{x}$.

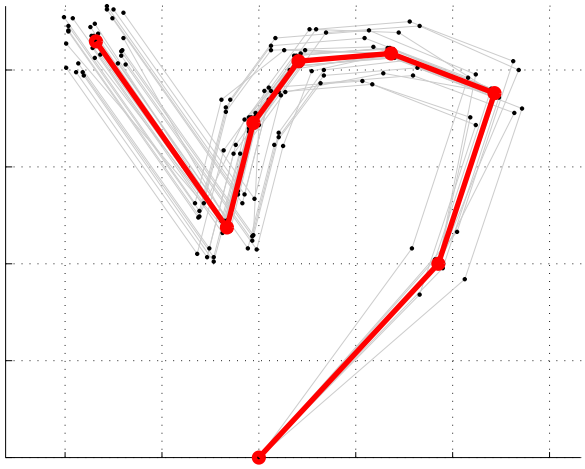


Figure 2-9: The sigma points are resampled to produce the $\mathcal{Y}_i$ paths (gray lines) shown here. The mean path is shown in red.

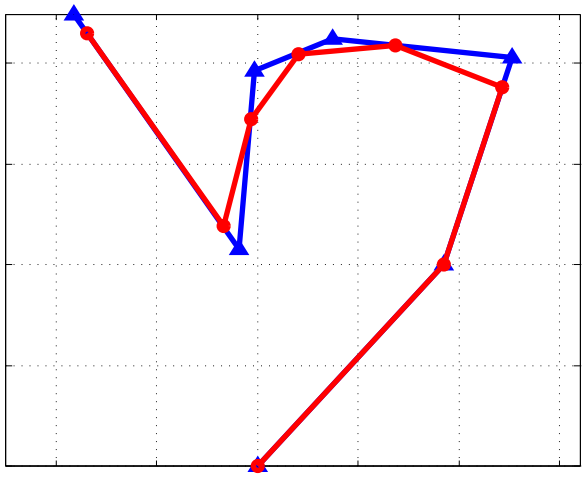


Figure 2-10: The original mean path (blue) at times A and resampled mean path (red) at times B.

Algorithm 2.4.5: Calibrate $(z_r,\Sigma_r,t_r,z_s,\Sigma_s,t_s)$

inputs : z_r, z_s observed incremental motions of sensors r, s (resp.) as $3N \times 1$ matrices
Σ_r, Σ_s $3N \times 3N$ covariances of z_r, z_s
t_r, t_s sample times of z_r, z_s

outputs: $\hat{k}$ Calibration estimate
$CRLB$ Cramer-Rao Lower Bound

$z'_s, \Sigma'_s \leftarrow$ `Interpolate`(t_s,z_s,Σ_s,t_r)
$\Sigma \leftarrow$ `blkdiag`(Σ_r,Σ'_s)
$\Psi \leftarrow$ `chol`(Σ^{-1})
$x_0 \leftarrow [v_{r0}, k_0] = [z_r, 0, 0, 0]$ `// initial guess`
$\hat{x} \leftarrow$ `Optimize`$(x_0,\ z_r,\ z'_s,\ \Psi,$ `Cost`$)$
$J_G \leftarrow$ `JacobianOfG`$(\hat{x})$
$J \leftarrow J_G^T \Sigma^{-1} J_G$
$\hat{k} \leftarrow \hat{x}_{3N-2:3N}$ `// extract 3-vector and` 3×3 `covariance matrix`
$CRLB \leftarrow J^{-1}_{(3N-2:3N,\ 3N-2:3N)}$ `// corresponding to the calibration`

Function 2.4.6: Interpolate$(A,\overline{v_A},\Sigma_A,B)$

Create χ according to Equation 2.50
for $\chi_i \in \chi$ **do**
  $\mathcal{Y}_i \leftarrow f\,(A, \mathcal{X}_i, B)$ `// see Section 2.4.5`
Calculate $\overline{v_B}$ and Σ_B according to Equation 2.51
return $\overline{v_B}, \Sigma_B$

Function 2.4.7: Cost$(x,\ z_r,\ z'_s,\ \Psi)$

$v_r, k \leftarrow x$
$\Delta \leftarrow \begin{bmatrix} z_r \\ z'_s \end{bmatrix} - \begin{bmatrix} v_r \\ G(v_r,k) \end{bmatrix}$
return $(\Psi^T \Delta)$ `// of dimension` $6N \times 1$

2.4.7 Practical Covariance Measurements

In order to estimate the calibration, our technique requires the covariances of the incremental poses; the inverse covariances serve as observation weights during LSE optimization (Equation 2.47). Observations with low uncertainty should have heavier weights than those with higher uncertainty. Thus, we must accurately characterize sensor noise in order to recover valid calibration parameters.

In some situations, it may be possible to confidently calculate the covariance matrix for each observation. For example, many GPS systems report uncertainty based on time skews and other well-understood physical quantities. On the other hand, estimating the accuracy of the output of a scan-matching algorithm can be difficult. Even with good sensor models, the variance may depend on the environment. For example, an empty room with many right angles may support more accurate localization than a room with lots of clutter.

Ideally, we would like to move the robot repeatedly along exactly the same path, collect observations at the exact same elapsed times during each run, and compute the variance of the incremental pose observations. However, it is impractical to repeat exactly the same path and timing in an unprepared environment. Instead, we outline a technique that does not require specialized equipment and that can be easily employed in the field. The idea is to move the robot along any path, but pause periodically. During each pause, we collect many observations from each sensor. By randomly drawing an observation made during each pause, we can effectively generate multiple experiments along a single path.

As an example, consider two planar LIDARs attached to a mobile robot. Figure 2-11 shows sample pose estimates from the two sensors. Notice that the density of estimates increases at intervals where we paused the robot. We refer to the observations within each pause as a "cluster."

Off-line, we approximated the covariance of the incremental poses between clusters. For each pair of adjacent clusters, we selected a random LIDAR scan from each cluster, and used scan-matching to find the relative motion between the two scans. By repeating this drawing and scan-matching, we assembled a set of relative motions, the covariance of which approximates the true incremental pose covariance.

This technique is appropriate only when inter-cluster incremental poses can be recovered solely from observations drawn from clusters. For example, scan-matching from LIDAR data or egomotion estimation from machine vision could provide suitable relative motion measurements, but inertial or odometry data alone would not suffice.

2.4.8 Results

2.4.8.1 Simulation

To assess the quality of our calibration estimate and validate the CRLB, we simulated a robot traveling such that sensor r moved along the path shown in Figure 2-12.

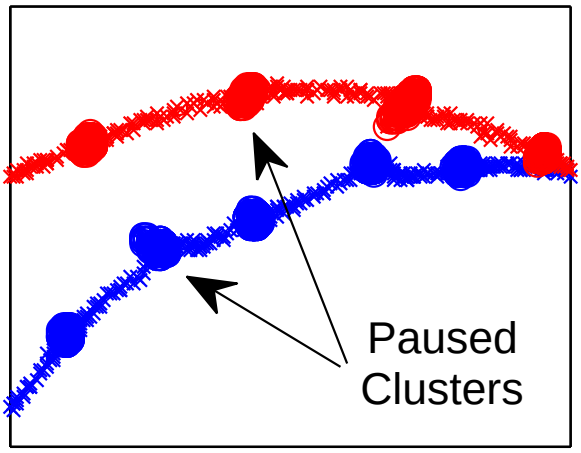


Figure 2-11: Observations drawn from each paused interval can be compared to estimate incremental pose covariances off-line.

The path had 400 incremental pose observations, and the path for sensor s was offset by various calibrations during the simulations. The 3×3 covariance for each observation was fixed, but varied across observations with a magnitude of 1%-6% of the incremental pose vector.

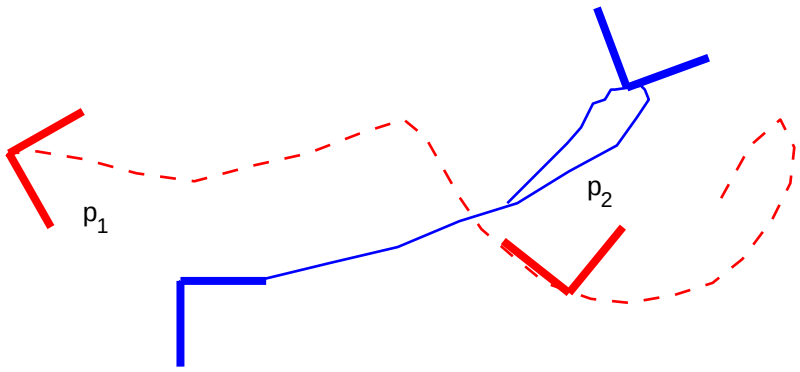


Figure 2-12: A simulated travel path for sensor r (blue), with calibration $k = [-0.3\,\text{m}, -0.4\,\text{m}, 30°]$ applied to sensor s (red). Two example sensor frames are shown at p_1 and p_2.

Thirty different calibrations were simulated, with k drawn randomly and uniformly from $[\pm 3\,\text{m}, \pm 3\,\text{m}, \pm\pi\,\text{rad}]$. In order to validate the CRLB, we executed 200 Monte Carlo simulations, sampling each incremental pose vector from its Gaussian distribution. Figure 2-13 shows the results of one such trial with $k = [-0.41\,\text{m}, 1.17\,\text{m}, -162°]$. The mean and standard deviations from the simulations match well with the true value and the CRLB, respectively.

As the number of simulations increases, we expect that the variance of the calibration will be the lowest attainable variance (i.e., the CRLB). With only 200 simulations, we hope that the standard deviation of our calibration estimates will lie near

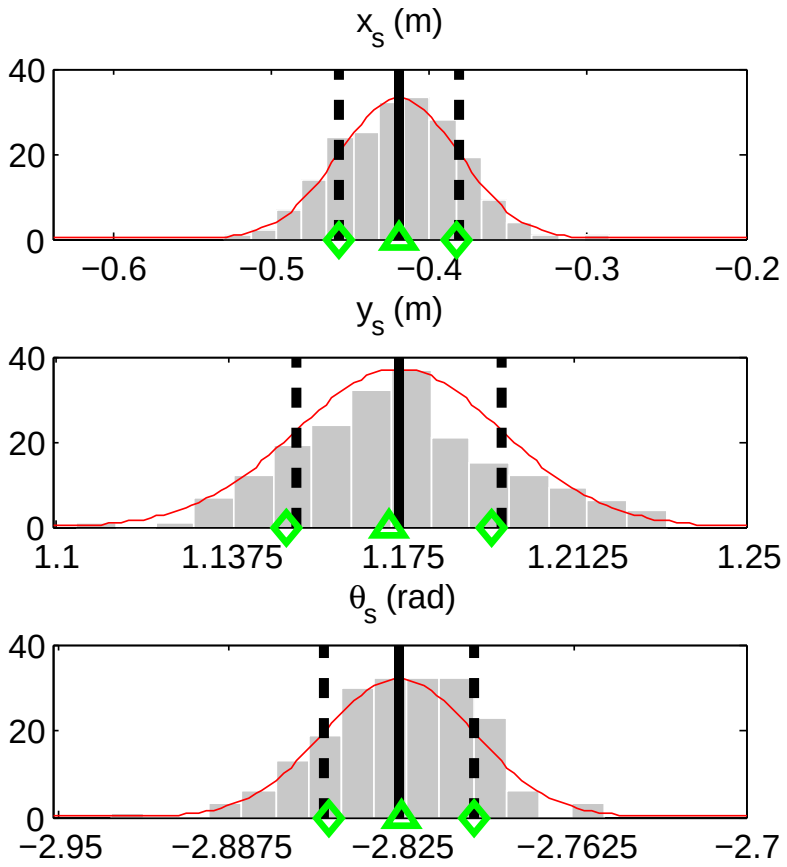


Figure 2-13: Histograms (gray) of calibration estimates from 200 simulations of the path in Figure 2-12 match well with truth (triangles) and the CRLB (diamonds). Vertical lines indicate mean (solid) and one standard deviation (dashed).

the CRLB. Indeed, the left column of Figure 2-14 shows that the standard deviation of x_s, y_s, and θ_s versus the calculated CRLB for x_s, y_s, and θ_s, respectively, generally agree.

The close agreement of the predicted CRLB and simulated standard deviations also supports the use of the CRLB (the CRLB is only valid for unbiased estimates, as described in Section 2.2.2). This is not surprising, since the bias (approximated via the Box technique, c.f. Section 2.4.4) is relatively small compared to the CRLB, as shown in the right column of Figure 2-14.

The scale of the standard deviations (i.e., the scale of the y-axes in Figure 2-14 and the x-axes in Figure 2-13) are arbitrary since we arbitrarily chose the covariance matrices for the observations. Figure 2-15 shows the relationship between the variance of the observations and the CRLB. As expected, as observation noise increases, the CRLB also increases. This indicates that as less information is present in the data, less can be deduced about the calibration. Note that this particular numerical relationship holds only for the path shown in Figure 2-12. Other paths produce different CRLBs (e.g., a degenerate path as in Section 2.4.2.4 would produce an "infinite" CRLB).

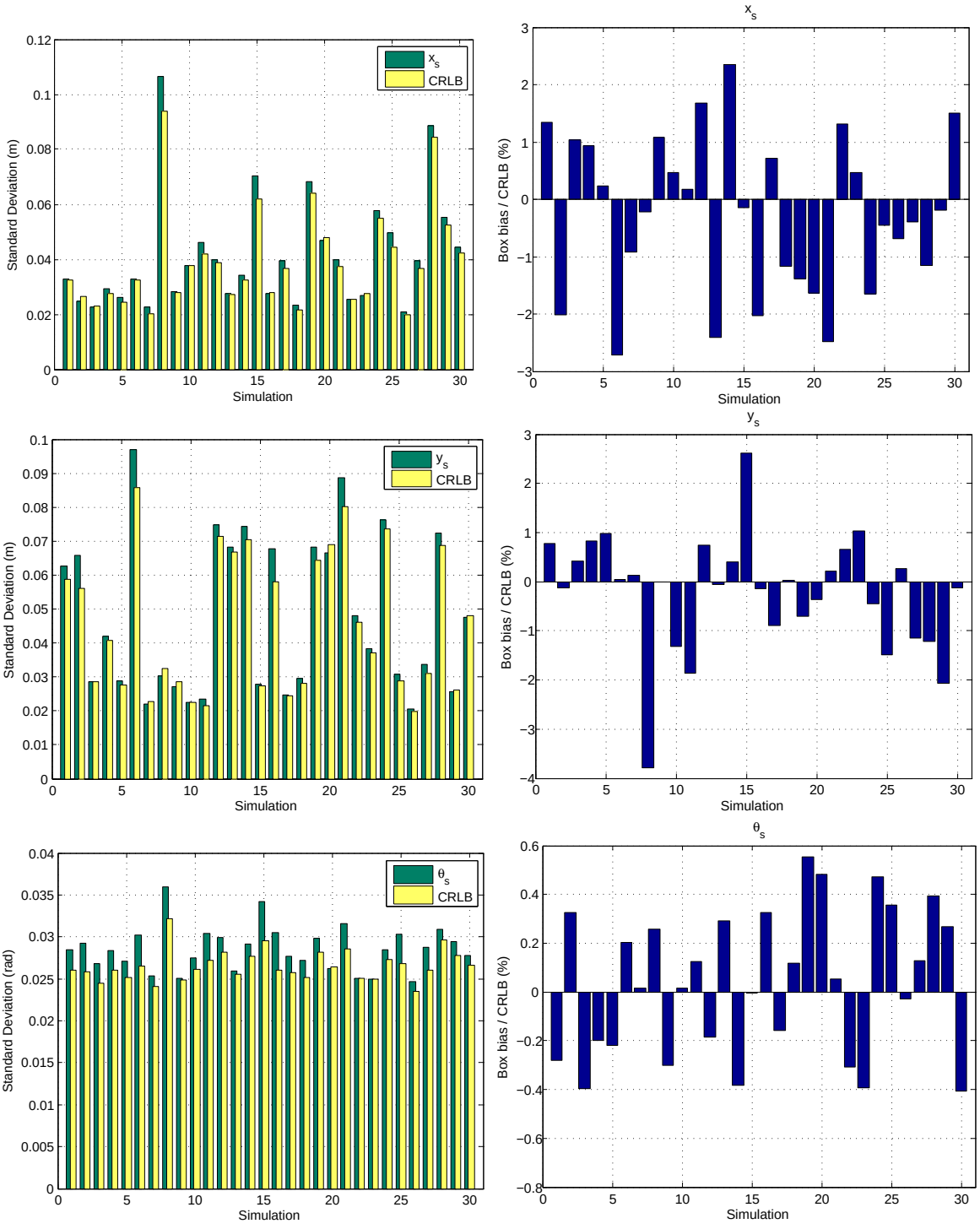


Figure 2-14: For 30 different simulated calibration runs, the parameter standard deviation (left, green) and CRLB (left, yellow) match well. Additionally, the Box bias is relatively small compared to the CRLB.

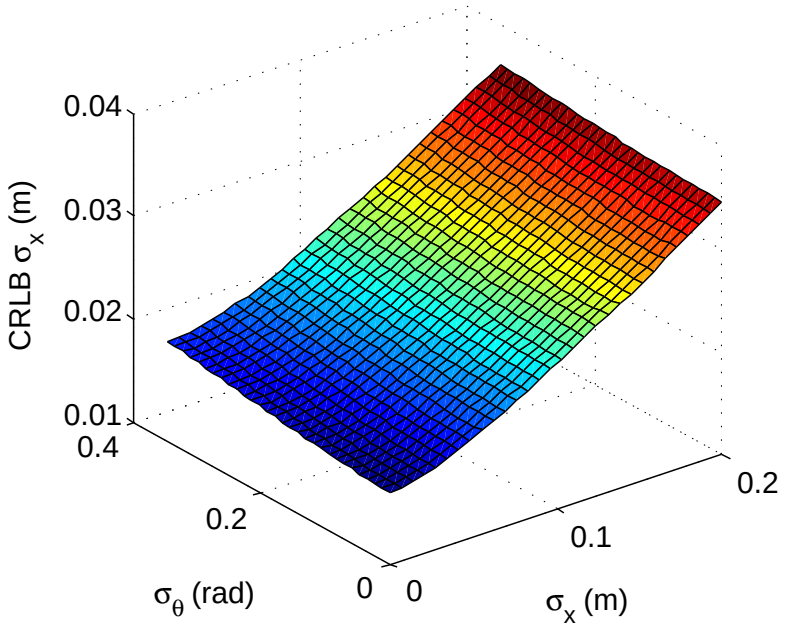


Figure 2-15: The CRLB increases with observation noise.

2.4.8.2 Real Data

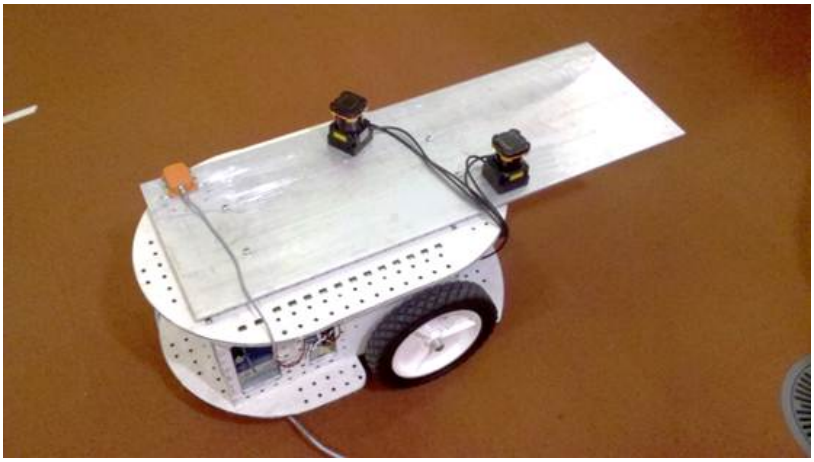


Figure 2-16: The robot test platform with configurable hardware, provides ground truth calibrations.

We attached a 30 cm × 120 cm machined plate to a small, differentially-steered mobile robot (see Figure 2-16). The plate included a grid of holes, allowing two Hokoyu UTM-30LX LIDARs to be mounted with various "ground truth" calibrations.

We drove the robot manually along a non-degenerate path (Figure 2-17), paused every few centimeters, calculated the mean and covariances of 88 incremental pose observations (Section 2.4.7), and estimated the calibration. To validate the CRLB for real data (a step which would not be necessary in general), we sampled the paused clusters to create multiple samplings along the same physical robot path. Then we estimated incremental poses by scan-matching line features between successive scans. No historical data was maintained (e.g., SLAM was not used).

Figure 2-18 shows the distribution of errors from the $k = [-0.2\,\text{m}, -0.5\,\text{m}, -120°]$ calibration; Table 2.2 shows results from real data for four different sensor calibrations.

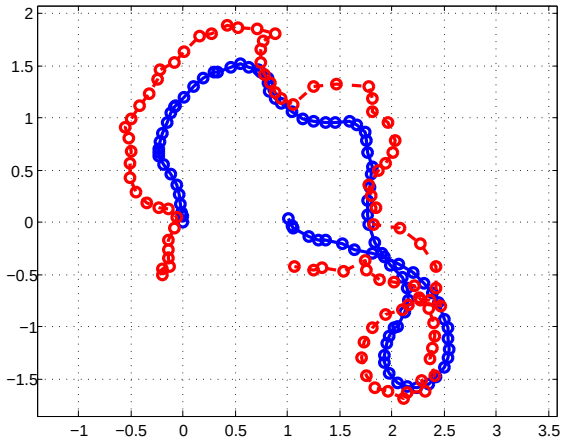


Figure 2-17: Paths of r and s when $k = [-0.2\,\text{m}, -0.5\,\text{m}, -120°]$ are shown.

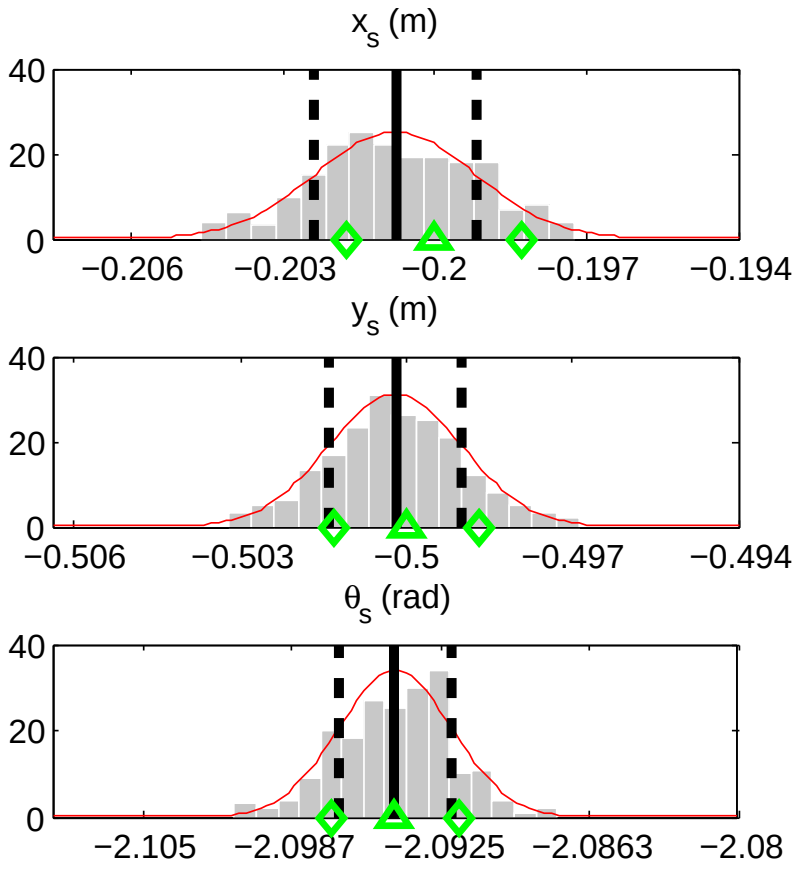


Figure 2-18: Estimates from 200 trials using real path of Figure 2-17 are shown.

The average error was at most 1.19 mm (and often much less than 1 mm) and less than approximately 1 milliradian. The standard deviation of the estimates was within 0.2 mm and 0.4 milliradians of the CRLB. Thus, both the calibration estimates and the CRLB agree well with the experimental data.

We also validated the algorithm by finding the calibration between the Hokoyu LIDARs r and s and the robot frame u. We used odometry from wheel encoders and rotation rate data from an Xsens MTi IMU to estimate the robot frame's translational and rotational incremental motions, respectively. We manually measured the robot frame's calibration parameters to be $k = [-0.08\,\text{m}, -0.06\,\text{m}, 90°]$ at the center of the axle, with a measurement accuracy of about 1 cm and 5°. We used the procedure in Section 2.4.7 to estimate observation covariances for the LIDARs and adopted reasonable values for the covariances of the robot frame incremental poses. Our method estimated the transformation from r to u as $[-0.081\,\text{m}, -0.063\,\text{m}, 90.3°]$ and from r to s to u as $[-0.080\,\text{m}, -0.061\,\text{m}, 90.3°]$. Both estimates are in close agreement with manual measurements and the orientation enforced by our mounting plate.

Table 2.2: Calibrations recovered from real 2D data

True Calibration			Estimation Error[a]			CRLB Error[b]		
x (m)	y (m)	θ (deg)	x (mm)	y (mm)	θ (mrad)	x (mm)	y (mm)	θ (mrad)
-0.2	-0.5	0	-0.19	0.03	0.37	-0.14	-0.13	-0.24
-0.2	-0.5	-90	-1.19	0.64	-1.06	-0.03	-0.12	-0.18
-0.2	-0.5	-120	-0.78	-0.20	0.00	-0.16	-0.09	-0.34
-0.2	-0.2	-60	-0.05	0.49	-0.73	-0.05	-0.21	-0.20

[a] Difference between mean of the estimates and the true calibration.
[b] Difference between standard deviation of the estimates and the CRLB.

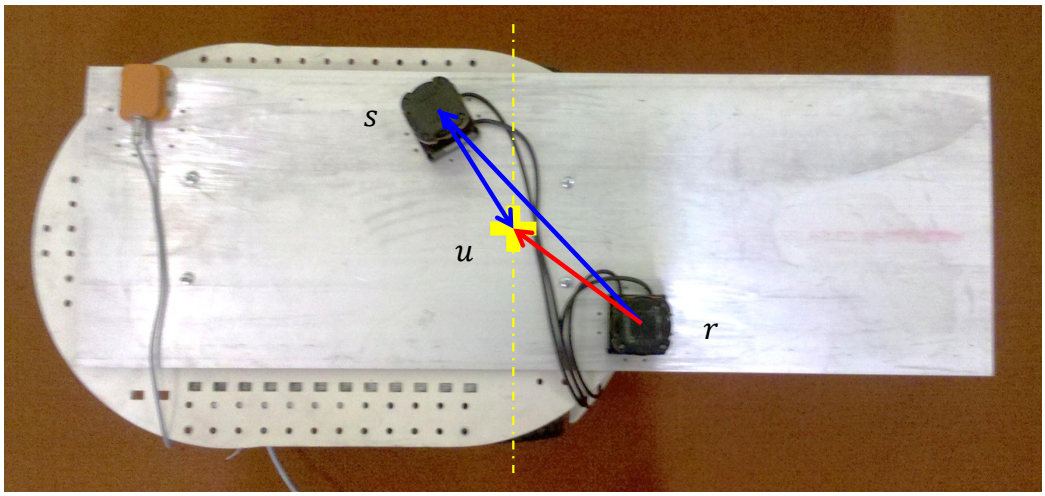


Figure 2-19: We recovered the closed calibration chain between the two LIDARs $\{r, s\}$ and the robot frame (u, combined IMU and odometry).

2.5 6-DOF Calibration

In Section 2.4 we examined the recovery of calibration parameters for a 2D (3-DOF) planar system. In this section, we will extend this notion to the 3D (6-DOF) case. We will again show that inter-sensor calibration and an uncertainty estimate can be accurately and efficiently recovered from incremental poses (and uncertainties) as observed by each sensor. Using notation similar to that of the previous section, Figure 2-20 shows sensors r and s, each able to observe its own incremental motion v_{ri} and v_{si}, respectively, such that the calibration k again satisfies:

$$v_{si} = g\left(v_{ri}, k\right) \tag{2.53}$$

As before, our algorithm will find the k that best aligns two series of observed incremental motions. The algorithm takes as input two sets of 6-DOF incremental pose observations and a 6×6 covariance matrix associated with each incremental pose. It produces as output an estimate of the 6-DOF calibration and a CRLB on

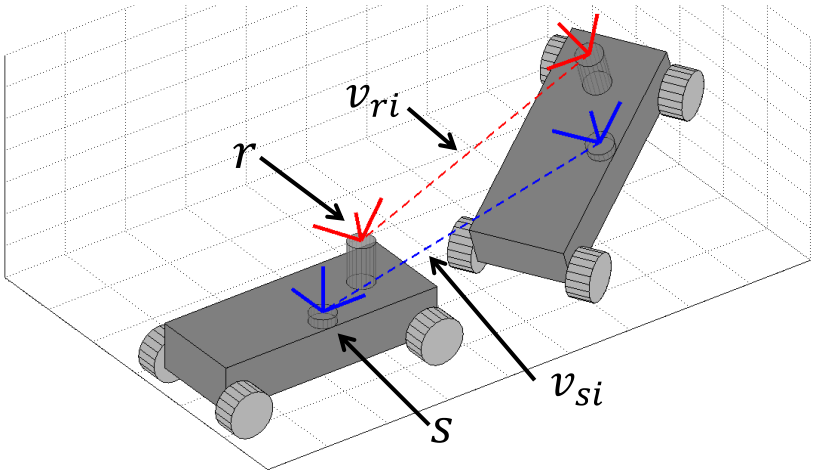


Figure 2-20: The incremental motions of the r (red) and s (blue) sensors are used to recover the calibration between the sensors as the robot moves. The dotted lines suggest the incremental motions, v_{ri} and v_{si}, for sensors r and s, respectively.

the uncertainty of that estimate (see Equation 2.20).

We begin again by confirming that the calibration is, in general, observable. This singularity analysis will reveal that 6-DOF calibration cannot be recovered from planar-only motion or when the sensors rotate only around a single axis. This confirms previous findings [17, 72] and provides a variance estimator useful in practice.

A key aspect of this section is the choice of representation for elements of the Special Euclidean group, $SE(3)$, which combines a translation in $\mathbb{R}^3$ with a rotation in $SO(3)$. Ideally, we desire a representation that:

1. Supports vector addition and scaling in order to formulate a principled noise model, and

2. Yields a simple form for g in Equation 2.53 in order to readily identify singularities in the FIM.

We considered pairing translations with a number of rotation representations — Euler angles, Rodrigues parameters, and quaternions — but each lacks some of the criteria above. Instead, we compromised by representing each element of $SE(3)$ as a unit dual quaternion (DQ) in the space $\mathbb{H}$. Each DQ $q \in \mathbb{H}$ has eight parameters and can be expressed as:

$$q = q_r + \varepsilon q_\varepsilon \tag{2.54}$$

where q_r is a "real" unit quaternion representing the rotation, q_ε is the "dual part" representing the translation, and $\varepsilon^2 = 0$. An 8-element DQ is over-parametrized (thus subject to two constraints) when representing a 6-DOF rigid body transform.

Although DQ's are not minimal, they are convenient for this problem, combining in a way analogous to quaternion composition and yielding a simple form for

g — about 20 lines of Matlab containing only polynomials and no trigonometric functions (see Section A.3). An Euler-angle representation, by contrast, is minimal but produces much more complex expressions involving hundreds of lines of trigonometric terms. Homogeneous transformations yield a simple form of g, but require maintenance of many constraints. The DQ representation offers a good balance of compactness and convenience.

Ordinary additive Gaussian noise cannot be employed with DQ's, since doing so would produce points not in $SE(3)$. Instead, we define a noise model using a projected Gaussian in the Lie algebra of DQ's [33] which is appropriate for this over-parametrized form.

To identify singularities of the FIM, we adapt communication theory's "blind channel estimation" methods to determine the calibration observability. Originally developed to determine the CRLB on constrained imaginary numbers [67], these methods extend naturally to DQ's.

An introduction to DQ's, Lie groups, and the DQ SLERP interpolation algorithm are provided in Section 2.5.1, Section 2.5.2, and Section 2.5.3, respectively. The 6-DOF calibration problem is formally stated in Section 2.5.4, along with a noise model appropriate for the DQ representation in Section 2.5.5. System observability is proven and degenerate cases are discussed in Section 2.5.6. The optimization process for constrained parameters is described in Section 2.5.7, along with techniques for resampling asynchronous data and converting between representations provided in Section 2.5.8. Experimental results from simulated and real data are given in Section 2.5.9. Additional proofs can be found in Appendix A.

2.5.1 Unit Dual Quaternions (DQ's)

Our calibration algorithm requires optimization over elements in $SE(3)$. Optimization over rigid body transformations is not new (e.g., [31]) and is a key component of many SLAM solutions. In our setting, DQ's prove to be a convenient representation both because they yield a simple form for g (Equation 2.53) and because they can be implemented efficiently [44]. Optimization with DQ's was also examined in [26], but their cost function included only translations; our optimization must simultaneously minimize rotation error.

A DQ, q, can be written in several ways:

- As an eight-element vector $[q_0, \cdots, q_7]$
- As two four-element vectors $[q_r, q_\varepsilon]$ (c.f. Equation 2.54). Note that q_r is a unit quaternion representing rotation, and q_ε is a quaternion (not necessarily of unit length) encoding the translation.
- Or, as a sum of imaginary and dual components:

$$q = (q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) + \varepsilon(q_4 + q_5\mathbf{i} + q_6\mathbf{j} + q_7\mathbf{k}) \tag{2.55}$$

In this form, two DQ's multiply according to the standard rules for the imaginary numbers $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$.

We write DQ multiplication as $a_1 \circ a_2$, where $\{a_1, a_2\} \in \mathbb{H}$. When we have vectors of DQ's, e.g., $a = [a_1, a_2]$ and $b = [b_1, b_2]$, where $\{b_1, b_2\} \in \mathbb{H}$, we write $a \circ b$ to mean $[a_1 \circ b_1, a_2 \circ b_2]$.

A pure rotation defined by unit quaternion q_r is represented by the DQ $q = [q_r, 0, 0, 0, 0]$. A pure translation, defined by $t = [t_0, t_1, t_2]$, can be represented by the DQ:

$$q = \left[1, 0, 0, 0, 0, \frac{t_0}{2}, \frac{t_1}{2}, \frac{t_2}{2}\right] \tag{2.56}$$

Given rigid body transform q, the inverse transform q^{-1} is:

$$q^{-1} = [q_0, -q_1, -q_2, -q_3, q_4, -q_5, -q_6, -q_7] \tag{2.57}$$

such that $q \circ q^{-1} = q^{-1} \circ q = \mathbf{I} = [1, 0, 0, 0, 0, 0, 0, 0]$. A vector $v = [v_0, v_1, v_2]$ can be represented as a DQ by:

$$q_v = [1, 0, 0, 0, 0, v_0, v_1, v_2] \tag{2.58}$$

The DQ form q_v of vector v transforms according to q as:

$$q'_v = q \circ q_v \circ q^* \tag{2.59}$$

where q^* is the dual-conjugate [44] to q:

$$q^* = [q_0, -q_1, -q_2, -q_3, -q_4, q_5, q_6, q_7] \tag{2.60}$$

DQ transforms can be composed as with unit quaternions, such that applying transform A, then transform B to point v yields:

$$q'_v = q_B \circ (q_A \circ q_v \circ q_A^*) \circ q_B^* = q_{BA} \circ q_v \circ q_{BA}^* \tag{2.61}$$

where $q_{BA} = q_B \circ q_A$. If the incremental motion v_{ri} and calibration k are expressed as DQ's, then Equation 2.53 becomes:

$$g(v_{ri}, k) := k^{-1} \circ v_{ri} \circ k \tag{2.62}$$

A DQ has eight parameters but represents only six DOFs. Accordingly, each DQ is subject to two constraints [53, p. 53-62]:

1. $q_r^T q_r = 1$. This constraint ensures that the quaternion, representing rotation, always has unit length.

2. $q_r^T q_\varepsilon = 0$. This constraint enforces the unit length on the DQ:

$$\begin{aligned} |q| &= (q_r + \varepsilon q_\varepsilon)^T (q_r + \varepsilon q_\varepsilon) && (2.63)\\ &= q_r^T q_r + 2 q_r^T q_\varepsilon + \varepsilon^2 q_\varepsilon^T q_\varepsilon && (2.64)\\ &= q_r^T q_r + 0 + 0 && (2.65)\\ &= 1 && (2.66) \end{aligned}$$

2.5.2 DQ's as a Lie Group

Ideally, we would like to perform vector operations (e.g., addition and subtraction) on the components of DQ's directly. Unfortunately, doing so would likely violate the DQ constraints. However, we can use a Lie algebraic formulation which allows us to perform local vector operations while implicitly maintaining the constraints.

Lie groups are smooth manifolds for which associativity of multiplication holds, an identity exists, and the inverse is defined [29, 42]; examples include $\mathbb{R}^n$, $SO(3)$, and $SE(3)$. However, $\mathbb{R}^n$ is also a vector space (allowing addition, scaling, and commutativity), but $SO(3)$ and $SE(3)$ are not. For this reason, points in $SO(3)$ and $SE(3)$ cannot simply be interpolated or averaged. We use the Lie algebra to enable an optimization method that requires these operations.

Lie algebra describes a local neighborhood (i.e., a tangent space) of a Lie group. So the Lie algebra of DQ's, $\mathfrak{h}$, can be used to express a vector space tangent to some point in $\mathbb{H}$. Within this vector space, DQ's can be arithmetically manipulated. Once the operation is complete, points in the Lie algebra can be projected back into the Lie group.

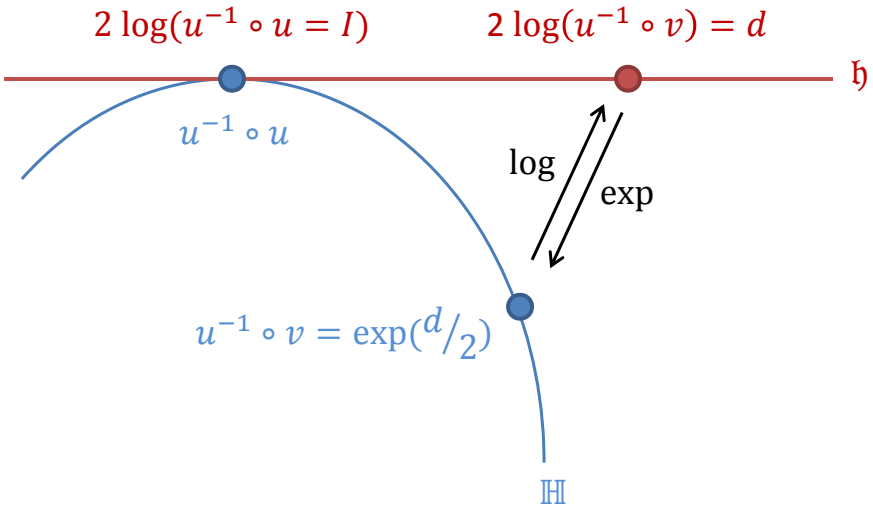


Figure 2-21: The mapping between the Lie group, $\mathbb{H}$, and the Lie algebra, $\mathfrak{h}$, is performed at the identity, i.e., $u^{-1} \circ u$.

The logarithm maps from the Lie group to the Lie algebra, and the exponent maps from the Lie algebra to the Lie group. Both mappings are done at the identity. For example, if we have two elements of a Lie group $\{u, v\} \in \mathbb{H}$, the "box" notation of [33] expresses the difference between v and u as:

$$d = v \boxminus u \tag{2.67}$$

Here, $\{u, v\}$ are each eight-element vectors and d is a six-element vector. That is, the box-minus operator connotes:

$$d = 2\log\left(u^{-1} \circ v\right) \tag{2.68}$$

where $d \in \mathfrak{h}$. In the Lie group, $u^{-1} \circ v$ is a small transformation relative to the identity, i.e., relative to $u^{-1} \circ u$ (see Figure 2-21). The factor of two before the log is a result of the fact that DQ's are multiplied on the left and right of the vector during transformation (c.f. Equation 2.59).

Similarly, the box-plus addition operator [33] involves exponentiation. If $d \in \mathfrak{h}$, then $\exp d \in \mathbb{H}$. If $d = 2\log(u^{-1} \circ v)$, then:

$$u \circ \exp \frac{d}{2} = u \circ \exp(\log(u^{-1} \circ v)) \tag{2.69}$$

$$= u \circ u^{-1} \circ v \tag{2.70}$$

$$= v \tag{2.71}$$

Since d applies a small transform to u, we use the box-plus operator to write:

$$v = u \boxplus d = u \circ \exp \frac{d}{2} \tag{2.72}$$

This definition of the difference between DQ's yields a Gaussian distribution as follows: imagine a Gaussian drawn on the line $\mathfrak{h}$ in Figure 2-21. Exponentiating points on this Gaussian "projects" the distribution onto $\mathbb{H}$. This projected Gaussian serves as our noise model (Section 2.5.4).

Summarizing, the Lie group/algebra enables several key operations: (1) addition of two DQ's, (2) subtraction of two DQ's, and (3) addition of noise to a DQ.

2.5.2.1 Logarithm of Dual Quaternions

The logarithm of some $q \in \mathbb{H}$ can be calculated as [65]:

$$\begin{aligned}\log q = \big(\frac{1}{4(\sin\theta)^3}[&(2\theta - \sin(2\theta))q^3 \\ &+ (-6\theta\cos\theta + 2\sin(3\theta))q^2 \\ &+ (6\theta - \sin(2\theta) - \sin(4\theta))q \\ &+ (-3\theta\cos\theta + \theta\cos(3\theta) \\ &\quad - \sin\theta + \sin(3\theta))\mathbf{I}]\big)_{1:3,5:7}\end{aligned} \tag{2.73}$$

where θ is the rotation angle associated with the DQ, and exponentiation of a DQ is implemented through repeated multiplication ($\circ$). (This expression incorporates a correction, provided by Selig, to that given in [65].) The $(\cdot)_{1:3,5:7}$ removes the identically zero-valued first and fifth elements from the 8-vector. To avoid the singularity at $\theta = 0$, the limit of $\log q$ can be evaluated as:

$$\lim_{\theta \to 0} \log q = [0, 0, 0, q_5, q_6, q_7] \tag{2.74}$$

For compactness, we write $\log a$ to mean $[\log a_1, \log a_2]$.

2.5.2.2 Exponential of Dual Quaternions

If $d \in \mathfrak{h}$, $w = [0, d_0, d_1, d_2]$ is a quaternion, and $q = [w, 0, d_3, d_4, d_5]$, we can exponentiate d as [65]:

$$\begin{aligned}\exp d = \frac{1}{2}(2\cos|w| + |w|\sin|w|)\mathbf{I} \\ - \frac{1}{2}(\cos|w| - 3\text{sinc}|w|)q + \frac{1}{2}(\text{sinc}|w|)q^2 \\ - \frac{1}{2|w|^2}(\cos|w| - \text{sinc}|w|)q^3 \end{aligned} \tag{2.75}$$

Notice that q and $\mathbf{I}$ are 8 element DQ's, and $\exp d$ also produces an 8 element DQ from the 6 element d. Further, the singularity at $w = 0$ can be avoided by evaluating:

$$\lim_{|w|\to 0} \exp d = \mathbf{I} + q \tag{2.76}$$

2.5.3 DQ SLERP

In $SO(3)$, unit quaternions can be interpolated via spherical linear interpolation (SLERP). SLERP operates by traveling along the shortest arc between two points on the 4-D unit sphere. The DQ SLERP [44] performs a similar traversal, but does so for $SE(3)$. Thus, DQ SLERP provides a way to interpolate between two rigid body transforms. However, unlike interpolation with translations and Euler angles or translations and quaternion SLERP, DQ SLERP interpolates with constant linear and angular velocity.

Consider DQ's $\{p, q, r\} \in \mathbb{H}$. We wish to interpolate between p and q according to some scale, t, to a new transform, r. Using the box notation, we wish to find:

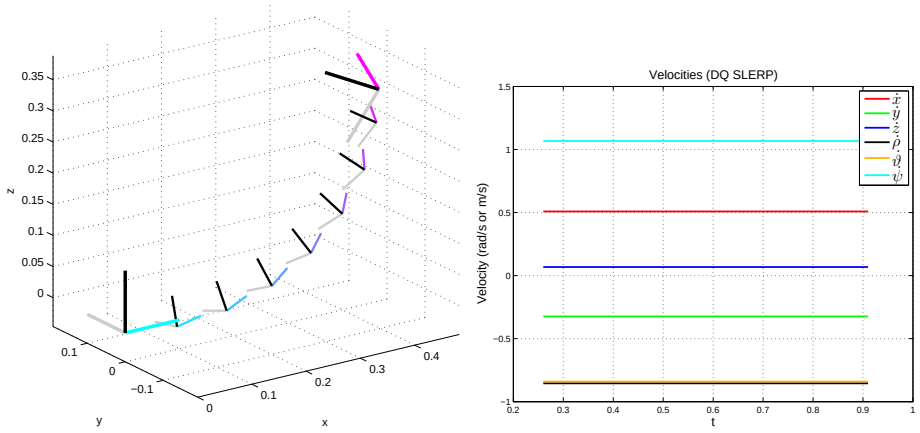
$$r = p \boxplus t\,(q \boxminus p) \tag{2.77}$$
$$= p \boxplus 2\,t \log\left(p^{-1} \circ q\right) \tag{2.78}$$
$$= p \circ \exp\left[t \log\left(p^{-1} \circ q\right)\right] \tag{2.79}$$
$$= p \circ \exp\left[\log\left(p^{-1} \circ q\right)^t\right] \tag{2.80}$$
$$= p \circ \left(p^{-1} \circ q\right)^t \tag{2.81}$$

Notice that if $t \in [0, 1]$, then interpolation is performed. However, there is nothing that prevents t from being outside this range and, thus, the same procedure can be used to perform extrapolation. Also, although Equation 2.81 produces the simplest expression, we actually use Equation 2.79 to implement the t-power operation.

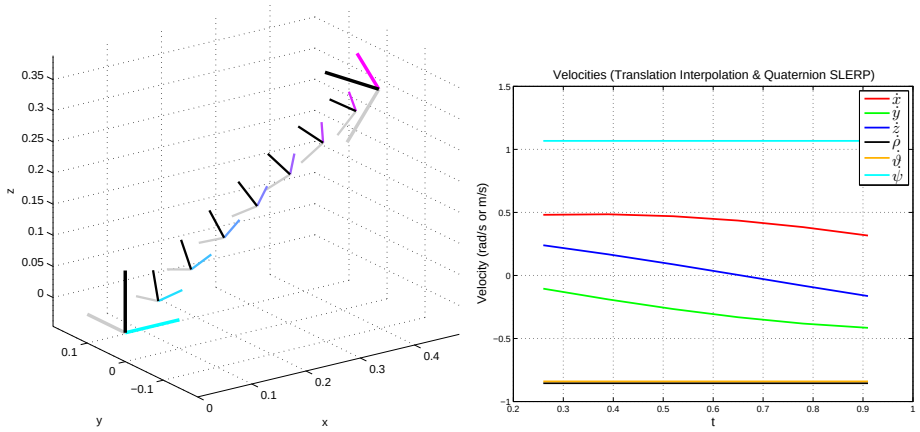
Figure 2-22 shows a comparison of three different interpolation methods in $SE(3)$. In the left column, two poses (cyan and magenta) are interpolated to produce a series of intermediate poses (smaller axes). The right column shows the discrete linear and

angular velocities experienced at each time step (using translations and Euler angles). In Figure 2-22a, the DQ SLERP procedure is used; note that the linear and angular velocities are constant. In Figure 2-22b, the translation is linearly interpolated and the rotation is interpolated via the quaternion SLERP. Here, the angular velocities remain constant, but the linear velocities vary during the interpolation. The linear velocities, as measured in the incremental frames, vary because the incremental frame rotates. Finally, in Figure 2-22c, we interpolate linearly in the space of translations and Euler angles. Here, both the linear and angular velocities vary throughout the interpolation.

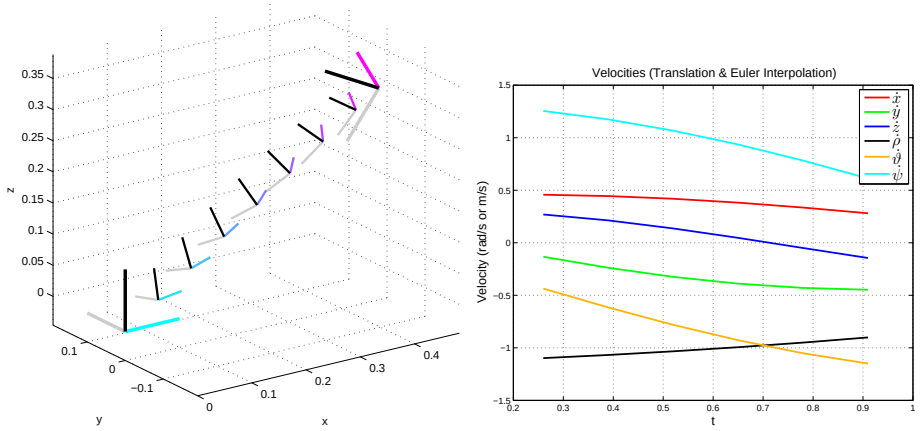
Determining which interpolation method is appropriate will depend on the application. Later, in Section 2.5.8, we will assume a constant velocity model and use the DQ SLERP. There are also other interpolation methods for $SE(3)$ not discussed here [44, 79, 80].



(a)



(b)



(c)

Figure 2-22: Three different methods for interpolating between the cyan and magenta poses are depicted. The DQ SLERP is used in (a), the quaternion SLERP is used in (b), and a linear interpolation in Euler angles is used in (c). The right column shows the linear and angular velocities.

2.5.4 Problem Statement

Given the observed incremental motions and their covariances, the problem is to estimate the most likely calibration. We formulate this task as a non-linear least squares optimization with a state representation of DQ's. DQ's were chosen to verify the CRLB in Section 2.5.9.1. In Section 2.5.7, we show how to perform the calculation in the over-parametrized state space. (In order to avoid singularities, we chose an over-parametrization rather than a minimal representation.)

The calibration parameters to estimate are $k = [k_0, \cdots, k_7]$, where $k \in \mathbb{H}$. The 6-DOF incremental poses from each sensor form a series of DQ observations (our experiments use FOVIS [35] and KinectFusion [55]). Again, let $z_r = [z_{r1}, z_{r2}, \cdots, z_{rN}]$ and $z_s = [z_{s1}, z_{s2}, \cdots, z_{sN}]$ be the observations from sensor r and s, respectively. Note that $\{z_{ri}, z_{si}\} \in \mathbb{H}$. Finally, let $z = [z_r, z_s]$ be the $(2N)$ observations. As in Section 2.4.1 for the planar case, both the incremental poses of sensor r and the calibration must be estimated [9]. Therefore, the state to estimate is $x = [v_r, k]$ consisting of $(N + 1)$ DQ's, where $v_r = [v_{r1}, v_{r2}, \cdots, v_{rN}]$ is the latent incremental motion of sensor r.

We then formulate the task as a maximum likelihood optimization:

$$\hat{x}^{ML}(z) = \underset{x=[v_r,k]}{\operatorname{argmax}} \prod_{i=1}^{N} P(z_{ri}|v_{ri})P(z_{si}|v_{ri}, k) \tag{2.82}$$

under the constraint that $\{v_{ri}, k\} \in \mathbb{H}$.

The probability functions might be assumed to be Gaussian. However, it is clear that adding Gaussian noise to each term of a DQ will not generally produce a DQ. Instead, we use the projected Gaussian.

By comparison, other approaches [21, 17, 72] simply ignore noise and assume that the observations and the dimensions should be equally weighted. However, when observation uncertainty varies (e.g., when the number of features varies in a vision algorithm) or when the uncertainty among dimensions varies (e.g., translations are less accurate than rotations), explicit representation of noise minimizes the effects of inaccurate observations. Further, a principled noise model allows recovery of the CRLB and, thus, the calibration's uncertainty.

2.5.5 Process Model

The process model can be written as:

$$z = G(x) \circ \exp \frac{\delta}{2} \equiv G(x) \boxplus \delta \tag{2.83}$$

$$G(x) = [v_{r1}, \cdots, v_{rN}, g(v_{r1}, k), \cdots, g(v_{rN}, k)] \tag{2.84}$$

where $\delta \in \mathfrak{h}$ acts as a projected Gaussian: $\delta \sim N(0, \Sigma_z)$. Here, the expected observations $G(x)$ have been corrupted by a noisy transformation with δ. Notice that,

in contrast to the noise model used in the 3-DOF case (Equation 2.24), the process model is not additive Gaussian noise, $z = G(x) + \delta$, which would result in $z \notin \mathbb{H}$.

The difference between the observations, z, and the expected values, $G(x)$, is $\lambda = -2\log\left(z^{-1} \circ G(x)\right)$, where λ includes $12N$ parameters — six for each of the $2N$ observations. The posteriors in Equation 2.82 can be written:

$$\begin{aligned} P(z|x) &\sim f\left(z^{-1} \circ G(x)\right) \\ f\left(z^{-1} \circ G(x)\right) &= \frac{1}{\sqrt{(2\pi)^{12N}|\Sigma_z|}} e^{-\frac{1}{2}\lambda^T \Sigma_z^{-1} \lambda} \end{aligned} \tag{2.85}$$

2.5.6 Observability

We will proceed as in the 3-DOF case and show that the 6-DOF calibration parameters are observable. The FIM (Equation 2.20) will be shown to be observable in general, then examined for particular singular situations. The FIM also provides for the calculation of the CRLB.

2.5.6.1 Fisher Information Matrix

In order to find the FIM, let $H(y) = -2\log y$ apply the box-minus ($\boxminus$) operator. H accepts a $16N \times 1$ parameter vector of DQ's in the Lie group, and returns a $12N \times 1$ vector of differences in the Lie algebra. We find the gradient of λ by applying the chain rule, and use the fact that the derivative of the logarithm is rotationally invariant for $SE(3)$ (see Section A.1 and [19]):

$$\nabla_x \lambda = \nabla_x H\left(z^{-1} \circ G(x)\right) \tag{2.86}$$

$$= \nabla_x H\left(G(x)\right) \tag{2.87}$$

$$= \left[\nabla_p H(p)|_{p=G(x)}\right]^T \left[\nabla_x G(x)\right]^T \tag{2.88}$$

$$= \underbrace{J_H}_{12N \times 16N} \underbrace{J_G}_{16N \times 8(N+1)} \tag{2.89}$$

Then, we calculate:

$$\nabla_x \ln P\left(z|x\right) = \nabla_x \ln c e^{-\frac{1}{2}\lambda^T \Sigma_z^{-1} \lambda} \tag{2.90}$$

$$= \nabla_x \left(-\frac{1}{2}\lambda^T \Sigma_z^{-1} \lambda\right) \tag{2.91}$$

$$= \left(J_H J_G\right)^T \Sigma_z^{-1} \lambda \tag{2.92}$$

$$= J_G^T J_H^T \underbrace{\Sigma_z^{-1}}_{12N \times 12N} \underbrace{\lambda}_{12N \times 1} \tag{2.93}$$

Substituting into the FIM Equation 2.20:

$$J = \mathrm{E}\left[\left(\nabla_x \ln P\left(z|x\right)\right)\left(\nabla_x \ln P\left(z|x\right)\right)^T\right] \tag{2.94}$$

$$= E\left[\left(J_G^T J_H^T \Sigma_z^{-1}\lambda\right)\left(J_G^T J_H^T \Sigma_z^{-1}\lambda\right)^T\right] \tag{2.95}$$

$$= E\left[J_G^T J_H^T \Sigma_z^{-1}\lambda\lambda^T\left(\Sigma_z^{-1}\right)^T J_H J_G\right] \tag{2.96}$$

$$= J_G^T J_H^T \Sigma_z^{-1} E\left[\lambda\lambda^T\right]\left(\Sigma_z^{-1}\right)^T J_H J_G \tag{2.97}$$

$$= J_G^T J_H^T \Sigma_z^{-1}\Sigma_z\left(\Sigma_z^{-1}\right)^T J_H J_G \tag{2.98}$$

$$= J_G^T J_H^T \Sigma_z^{-1} J_H J_G \tag{2.99}$$

Since each of the $(N+1)$ DQ's in x is subject to two constraints, J_G is always rank-deficient by at least $2(N+1)$. Of course, our interest is not in rank deficiencies caused by over-parametrization but in singularities due to the observations. Thus, we must distinguish singularities caused by over-parametrization from those caused by insufficient or degenerate data.

2.5.6.2 Cramer-Rao Lower Bound

In the communications field, a related problem is to estimate complex parameters of the wireless transmission path. These complex "channel" parameters are usually unknown but obey some constraints (e.g., they have unit magnitude). Since the CRLB is often useful for system tuning, Stoica [67] developed techniques to determine the CRLB with constrained parameters.

Following [67], suppose the m constraints are expressed as $f^c = [f_1^c, \cdots, f_m^c]$ such that $f^c(x) = 0$. Let F^c be the Jacobian of f^c and U be the null space of F^c such that $F^c(x)U = 0$. When $K = U^T JU$ is non-singular, the CRLB exists. In our case,

$$K = U^T JU = U^T J_G^T J_H^T \Sigma_z^{-1} J_H J_G U \tag{2.100}$$

$$= U^T J_G^T J_H^T L L^T J_H J_G U \tag{2.101}$$

$$= (L^T J_H J_G U)^T (L^T J_H J_G U) \tag{2.102}$$

where $\Sigma_z^{-1} = LL^T$ (e.g., by using the Cholesky factorization). In order to find the cases where J is singular, we examine the rank of K:

$$\mathrm{rank}(K) = \mathrm{rank}\big(\left(L^T J_H J_G U\right)^T (L^T J_H J_G U)\big) \tag{2.103}$$

Since $\mathrm{rank}(A^T A) = \mathrm{rank}(A)$,

$$\mathrm{rank}(K) = \mathrm{rank}(L^T J_H J_G U) \tag{2.104}$$

Further, since each observation is full rank, Σ_z is full rank; L^T is a $12N \times 12N$ matrix with rank $12N$ and $\mathrm{rank}(L^T A) = \mathrm{rank}(A)$. Thus (see Section A.2):

$$\mathrm{rank}(K) = \mathrm{rank}(J_H J_G U) = \mathrm{rank}(J_G U). \tag{2.105}$$

For the calibration estimation, there are $m = 2(N+1)$ constraints (two for each DQ in $x = [v_r, k]$). Separating the DQ real and dual components as $v_{ri} = q_{ri} + \varepsilon q_{\varepsilon i}$ and $k = q_{rk} + \varepsilon q_{\varepsilon k}$, then $f^c(x)$ is shown in Equation 2.106.

$$f^c(x) = \begin{bmatrix} q_{r1}^T\, q_{r1} - 1 \\ q_{\varepsilon 1}^T\, q_{\varepsilon 1} \\ \cdots \\ \cdots \\ q_{rN}^T\, q_{rN} - 1 \\ q_{\varepsilon N}^T\, q_{\varepsilon N} \\ q_{rk}^T\, q_{rk} - 1 \\ q_{\varepsilon k}^T\, q_{\varepsilon k} \end{bmatrix} = 0 \tag{2.106}$$

Since each term in $f^c(x)$ is at most quadratic, each element of the Jacobian, F, is simply a linear function of the DQ parameters. The null space, U, can then be determined using standard analytic or numeric techniques.

2.5.6.3 Degeneracies

As shown in Section 2.4.2.2, one observation is insufficient to recover the calibration. For two observations and a particular null space matrix U, $J_G U$ has the form shown in Figure 2-23. Notice that columns 7-18, which in this example correspond to v_{r1} and v_{r2}, are always linearly independent due to the unity entries. This is not surprising since v_{ri}, the estimated motion of sensor r, is directly observed by z_{ri}. Only the first six columns, corresponding to the six DOF's of the calibration, can possibly be reduced. By examining linear combinations of these columns, we can identify a singularity which, if avoided in practice, will ensure that the calibration is observable.

Suppose the two incremental motions experienced by sensor r are $\{a, b\} \in \mathbb{H}$ and let a_i be the i-th term of the 8-element DQ, a. When $a_1 b_3 = a_3 b_1$ and $a_2 b_3 = a_3 b_2$, $J_G U$ is singular and the calibration is unobservable. Since the 2nd-4th elements of the DQ correspond to the 2nd-4th elements of a unit quaternion representing the rotation, it follows that these relations hold only when there is no rotation or when the rotation axes of a and b are parallel. Thus, the *calibration is unobservable when the sensors rotate only about parallel axes.*

In principle, this analysis could have been completed using any representation for $SE(3)$. However, attempting the analysis using Euler angles and Mathematica 8.0 exceeded 24GB of memory without completing; manual analysis was equally difficult. By contrast, DQ over-parametrization made both manual and automated analyses tractable to perform and simple to interpret, making readily apparent the degeneracy arising from a fixed axis of rotation.

The condition to avoid degeneracy has several common special cases:

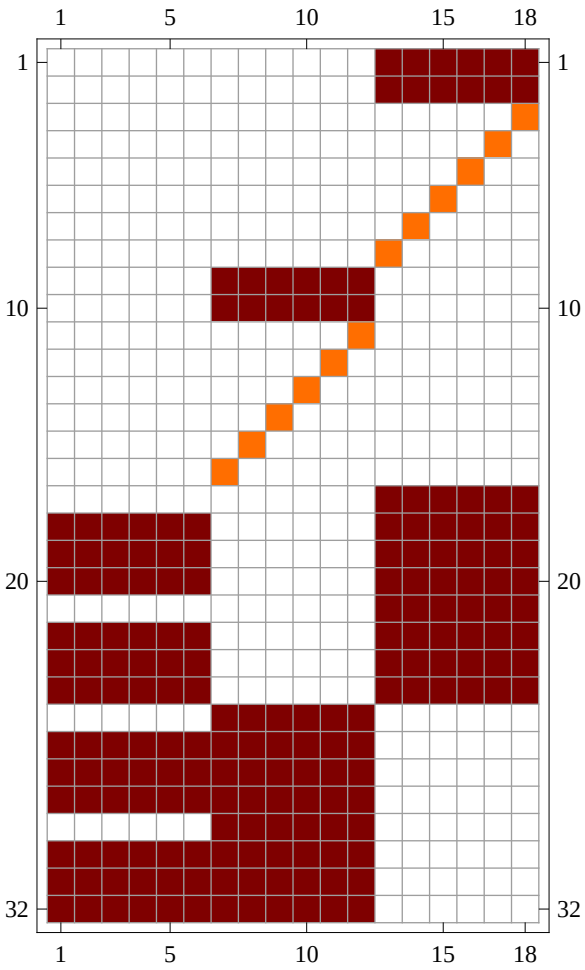


Figure 2-23: Visualization of the matrix $J_G U$ shows that only the first six columns can be reduced. Blank entries are zero, orange are unity, and red are more complex quantities.

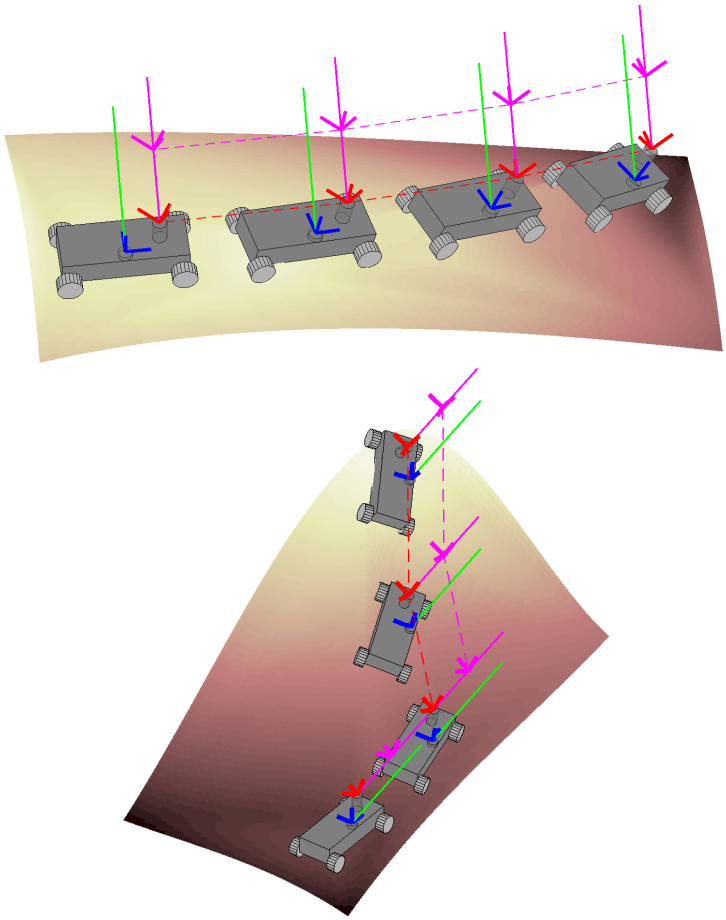


Figure 2-24: Two robots driven along the suggested paths experience rotation about only one axis (green). As a result, the true calibration relating the two true sensor frames (red/blue) cannot be determined. The magenta lines and frames show ambiguous locations for the red sensor frame.

1. Constant velocity. When the sensors move with constant velocity, the axes of rotation are constant.

2. Translation only. When the sensors only translate, no rotation is experienced.

3. Planar motion. When the robot travels only in a plane, the rotation axis is fixed (i.e., perpendicular to the plane).

These special cases, however, do not fully characterize the degeneracy. So long as the axis of rotation of the incremental poses remains fixed, any translations and any magnitude of rotation will not avoid singularity.

In Figure 2-24, for example, a robot is traveling along the suggested 3D terrain. Although the robot translates and rotates some varying amount between poses, it always rotates about the same axis (green lines). In such situations, the calibration is at least ambiguous along a line parallel to the axis of rotation (magenta line). That is, if r is held fixed and s is translated along such a line, the observations from sensor

s remain fixed. Because multiple calibrations can produce the same observations, the calibration is unobservable.

2.5.7 Optimization

In order to estimate the calibration from Equation 2.82, we perform non-linear least squares optimization, using a modified Gauss-Newton (GN) algorithm [33]. GN optimization attempts to minimize a non-linear least squares cost function (such as ours Equation 2.82). By linearizing the cost function at each update step and then solving for the linear least squares solution, the following update rule is obtained (see Section 3.2.7 for a more detailed discussion of Gauss-Newton optimization):

$$x_{t+1} = x_t - \left(J_t^T J_t\right)^{-1} J_t^T \left(f\left(x_t\right) - z\right) \tag{2.107}$$

where J_t is the Jacobian at the current estimate of the function f to minimize.

This rule can be adapted to the calibration task, which operates in the Lie algebra, with:

$$x_{t+1} = x_t \boxplus \left(-\left(\mathcal{J}_t^T \Sigma^{-1} \mathcal{J}_t\right)^{-1} \mathcal{J}_t^T \Sigma^{-1} \left(G\left(x_t\right) \boxminus z\right)\right) \tag{2.108}$$

The term $G(x_t) \boxminus z$ represents error elements in $\mathfrak{h}$, which are scaled by the gradient and added (via $\boxplus$) to the current parameter estimate to produce a new set of DQ's. The method computes error, and applies corrections to the current estimates, in the tangent space via the Lie algebra. After each update, the parameters lie in $\mathbb{H}$ and there is no normalization required.

$\mathcal{J}_t$ is the analytic Jacobian at the current estimate [73], calculated by:

$$\mathcal{J}_t = \left(\nabla_h H\left(z^{-1} \circ G\left(x_t \circ \exp\left(\frac{h}{2}\right)\right)\right)\right)\Bigg|_{h=0} \tag{2.109}$$

Essentially, the method shifts the parameters x_t via $h \in \mathfrak{h}$, then evaluates that shift at $h = 0$.

2.5.8 Interpolation

Although the DQ representation facilitates the FIM analysis and there are methods to develop a noise model, data and covariance matrices will typically be available in more common formats, such as Euler angles. Furthermore, sensors are rarely synchronized, so incremental motions may be observed over different sample periods. Following the process in Section 2.4.5 and [9], we use the Scaled Unscented Transform (SUT) [40] to (1) convert incremental motion and covariance data to the DQ representation and (2) resample observations and covariances from different sensors at common times.

The SUT creates a set of sigma points, $\mathcal{X}$, centered about the mean and spaced according to the covariance matrix (Equation 2.50). Each point is passed through the

interpolation function f^i to produce a set of transformed points $\mathcal{Y}$ (see Equation 2.51). A new distribution is then created to approximate the weighted $\mathcal{Y}$.

We use a similar approach to that of [33] to incorporate the Lie algebra into the SUT. This process uses the standard SUT equations (Equation 2.50 and Equation 2.51), but replaces the addition with $\boxplus$ and subtraction with $\boxminus$.

The interpolation function, f^i in Function 2.5.1, converts the Euler states to DQ's. It then accumulates the incremental DQ motions (using Function 2.4.2) into a common reference frame and resamples them at the desired times, typically the sample times of the reference sensor. Resampling is done via the DQ SLERP operator (see Section 2.5.3) which interpolates between poses with constant speed and with shortest path [44]. The function then calculates the incremental motions using Function 2.4.4.

Function 2.5.1: $f^i(A,\mathtt{v}_A,B)$

inputs : A the N sample times of incremental motions
$\mathtt{v}_A$ the N incremental motions at times A expressed as translations and Euler angles
B the M sample times at which to resample

outputs: $\mathtt{v}_B$ the M incremental motions at times B expressed as DQ's

```
v_ADQ ← TransAndEulerToDQ(v_A)          // convert to DQ's
x_A ← AccumulateStates(v_ADQ)      // accumulate incremental motions
                                   // into a global reference frame
x_B ← DQSLERP(A, x_A, B)              // resample poses at new times
v_B ← MakeIncremental (x_B)           // produce incremental motions
return v_B
```

2.5.8.1 Averaging DQ's

One challenge with applying the SUT in the Lie algebra is that Equation 2.51 requires an average of DQ's $\mathcal{Y}$ weighted by W. Because averaging each individual element of a DQ does not in general produce a DQ, we require an alternate means to estimate the average. This averaging method must be able to incorporate negative weights, such as those generated by the SUT.

We initially attempted to adapt the procedures in [33] and [44], originally designed for uniform and positive weights, respectively. The method in [33] operates by performing steepest gradient descent with an error estimated in the Lie algebra. This idea is augmented in [44] with an initial estimate based on a weighted sum of DQ elements and a subsequent normalization.

However, we found that both methods often failed to converge when the $\mathcal{Y}_i$'s were similar (i.e., when the covariance was small). This is may be because [33, 44] use a fixed gain during the gradient descent, causing the optimizer to quickly diverge from nearby minimums. The use of a fixed gain is a known limitation [3] and, instead, we use the GN optimization routine already developed in Equation 2.108. We minimize

the sum of the errors between the mean and each DQ, E, appropriately weighted, and recover the estimate of the mean, $b \in \mathbb{H}$.

$$E(W, \mathcal{Y}) = \underset{b}{\operatorname{argmin}} \sum_{i=1}^{N} W_i \left(\mathcal{Y}_i \boxminus b\right) \tag{2.110}$$

$$= \underset{b}{\operatorname{argmin}} \sum_{i=1}^{N} W_i \log\left(b^{-1} \circ \mathcal{Y}_i\right) \tag{2.111}$$

To use this procedure, we need a reasonable initial guess for the mean. In the case of the SUT, the first propagated sigma point $\mathcal{Y}_0$ is often a good estimate for the mean. In some circumstances, however, a good initial guess may not be available. For example, we might need to average the hypotheses of a particle filter to produce a mean estimate (e.g., the `WeightedStateMean` in Algorithm 3.2.1).

A possible solution can be seen by considering a weighted average of three real numbers:

$$\overline{x} = \frac{x_1}{3} + \frac{x_2}{3} + \frac{x_3}{3} \tag{2.112}$$

It would be convenient if the analogous computation could be done for DQ's. However, scalar division has no meaning for DQ's (i.e., there is no way to "divide by three"). The only scaling operation we can perform with DQ's is interpolation between two elements. With that in mind, suppose we define the mean recursively as follows:

$$\overline{x} = \overbrace{\frac{x_1}{3} + \frac{2}{3}\underbrace{\left(\frac{x_2}{2} + \frac{x_3}{2}\right)}_{\text{first interpolation}}}^{\text{second interpolation}} \tag{2.113}$$

With this formulation, the mean can be considered as a series of pair-wise interpolations and suggests Function 2.5.2 for DQ's. In practice, we found this procedure to give good initial guesses. It is important to note, however, that the procedure is order dependent. That is, different orderings of the elements in x will give different results. Consequently, it is used only to provide an initial guess for optimization.

2.5.9 Results

2.5.9.1 Simulation

We validated the calibration estimates and constrained CRLB by simulating robot motion along the path shown in Figure 2-25. The $N = 62$ observations and covariances per sensor were simulated using a translation/Euler angle parametrization (instead of DQ's) to model data obtained from sensors in practice. Additive Gaussian noise was applied to this minimal representation with a magnitude 10-40% of the true value.

Function 2.5.2: DQMean(x,w)

inputs : x the N DQ's to average
w the N weights of each corresponding DQ

outputs: y an approximation of the DQ mean

if $N = 1$ **then**
  $y \leftarrow x$
else
  $z \leftarrow$`DQMean(`$x_{2:N}$, $w_{2:N}$`)`
  $s \leftarrow \Sigma w$
  $y \leftarrow$`DQSLERP(`x_1, z, $1 - \frac{w_1}{s}$`)`

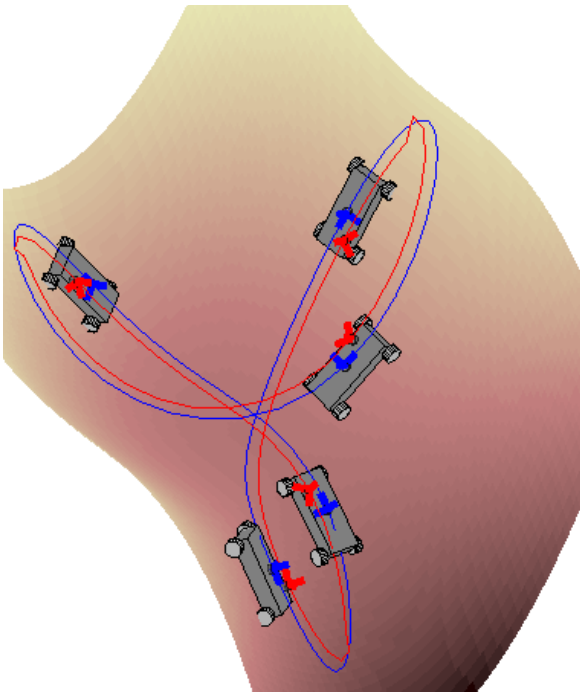


Figure 2-25: Motion is simulated such that the red and blue sensors traveled the paths as shown. (The path is always non-degenerate.) In this image $k = \left[0.1, 0.05, 0.01, 0, 0, \frac{\pi}{3}\right]$.

Thirty different calibrations were simulated, each with rigid body parameters k uniformly drawn from $[\pm 3\,\text{m}, \pm 3\,\text{m}, \pm 3\,\text{m}, \pm\pi\,\text{rad}, \pm\pi\,\text{rad}, \pm\pi\,\text{rad}]$. The observations and covariances were then converted to DQ's using the interpolation method of Section 2.5.8. We validated the CRLB by performing 400 Monte Carlo simulations [2] for each calibration, sampling each velocity vector from its Gaussian distribution.

Figure 2-26 shows results for the sample calibration:

$$k_{Euler} = [2.79\,\text{m}, -2.79\,\text{m}, -1.45\,\text{m}, -0.51\,\text{rad}, 0.94\,\text{rad}, -1.22\,\text{rad}]$$

in meters (m) and radians (rad) or, in DQ form:

$$k_{DQ} = [0.77, 0.07, 0.49, -0.40, 0.30, 1.99, -0.57, 0.21]$$

As shown, the mean and standard deviations from the simulations are well matched with the true value and the CRLB, respectively. It is important to note that the covariance matrix corresponding to Figure 2-26 is calculated on an over-parametrization — there are only six DOF's in the 8-element DQ representation. Due to these dependent (i.e., constrained) parameters, the covariance matrix is singular. However, during optimization we use a covariance matrix expressed in the Lie algebra; thus, we avoid problems with such singular matrices.

The left columns of Figure 2-27 and Figure 2-28 show the error between the thirty true calibrations and the mean of the estimated values for each DQ parameter. The parameters were recovered to within about 0.01 of truth. The right columns compare the standard deviation of the parameters resulting from the Monte Carlo experiments and the predicted CRLB. In general, the method came close (within about 0.01) to the best-case CRLB.

2.5.9.2 Real data

We further validated the estimator with two different types of depth sensors and motion estimation algorithms. First, we collected data with two Microsoft Kinect RGB-D cameras, mounted on three different machined rigs with known calibrations. The RGB-D data from each camera was processed using the Fast Odometry from VISion (FOVIS) [35] library, which uses image features and depth data to produce incremental motion estimates. Second, we collected data with two rigidly mounted Xtion RGB-D cameras and used the KinectFusion algorithm [55] for motion estimation. For all four calibrations, we moved each rig by hand along a path in 3D. The interpolation algorithm (Section 2.5.8) was used to synchronize the translations/Euler angles and convert to DQ's.

We characterized the noise in both systems using data from stationary sensors. We projected the noise into $\mathfrak{h}$ and, using a chi-squared goodness-of-fit test, we found the projected Gaussian to be a good approximation (at 5% significance) for both FOVIS and KinectFusion.

In practice, we found that feature-rich environments yielded higher-quality calibration estimates. Our datasets using FOVIS, for example, averaged ~100 feature correspondences between frames.

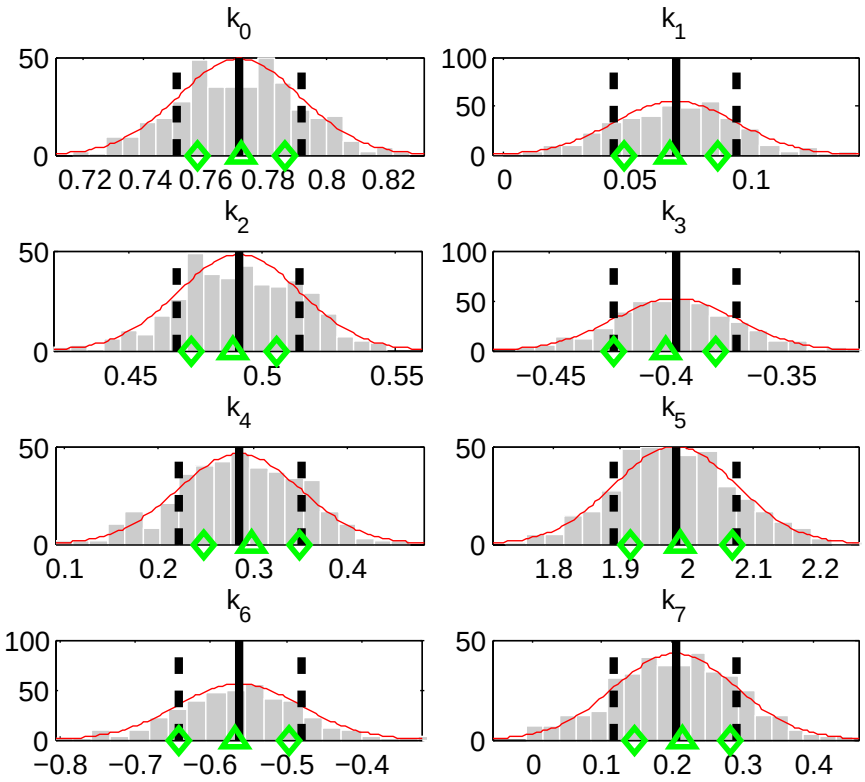


Figure 2-26: Histograms (gray) of calibration estimates from 400 simulations of the path in Figure 2-25 match well with the true calibration (green triangles) and constrained CRLB (green diamonds). Black lines indicate the sample mean (solid) and one standard deviation (dashed); the red lines show a fitted Gaussian.

Table 2.3: Ground truth calibrations recovered from real 3D data

Calibration	Sensor Type	x (m)	y (m)	z (m)	ρ (rad)	ϑ (rad)	ψ (rad)
#1	Kinect/FOVIS	-0.045	-0.305	-0.572	-1.316	0.906	-1.703
#2	Kinect/FOVIS	-0.423	-0.004	0.006	-0.000	0.000	3.141
#3	Kinect/FOVIS	-0.165	0.204	-0.244	1.316	-0.906	3.009
#4	Xtion/KinectFusion	-0.040	0.025	0.000	-0.052	0.000	3.141

Table 2.4: Difference between mean of the estimates and the true calibrations in Table 2.3

Calibration	x (mm)	y (mm)	z (mm)	ρ (mrad)	ϑ (mrad)	ψ (mrad)
#1	0.75	11.00	11.83	22.32	2.20	9.92
#2	-7.17	-14.34	-3.20	-19.95	3.53	-7.28
#3	-7.77	3.92	-13.65	3.31	0.61	-5.83
#4	-6.19	8.22	6.06	-4.03	-17.39	5.34

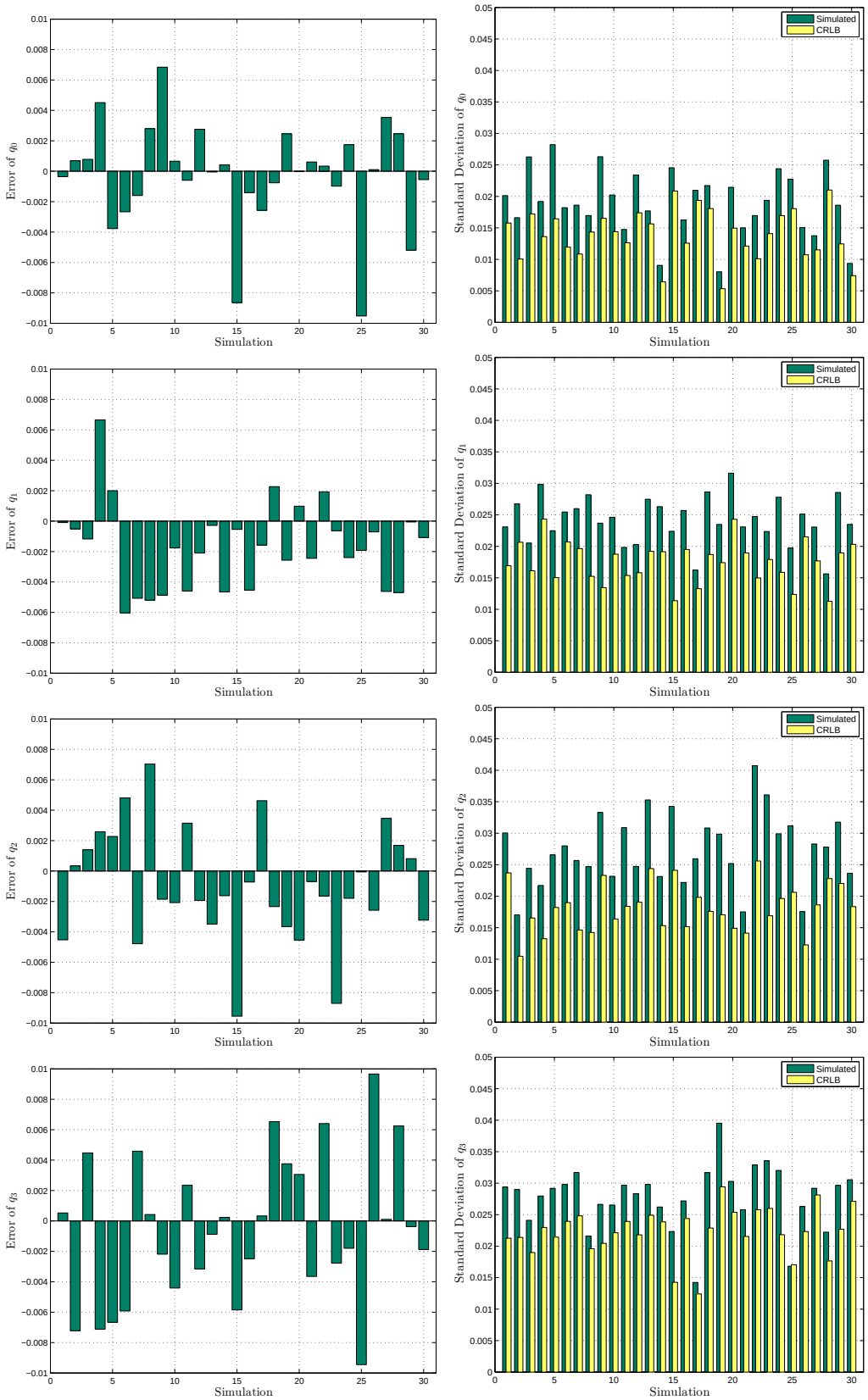


Figure 2-27: The error between the known calibration and the mean estimate was less than ± 0.01 for each DQ parameter. Parameter q_0-q_4 are shown here.

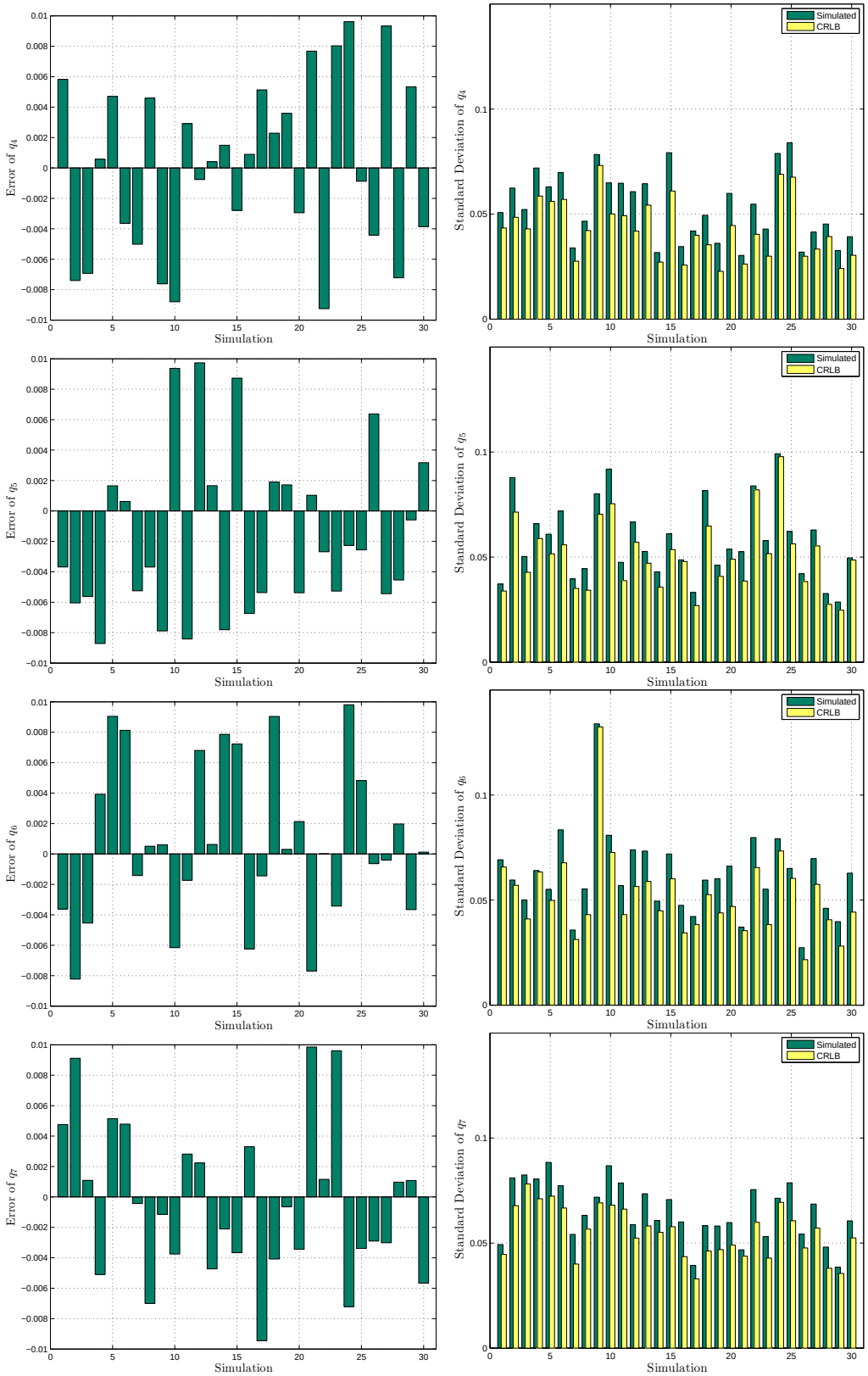


Figure 2-28: The error between the known calibration and the mean estimate was less than ± 0.01 for each DQ parameter. Parameter q_5-q_7 are shown here.

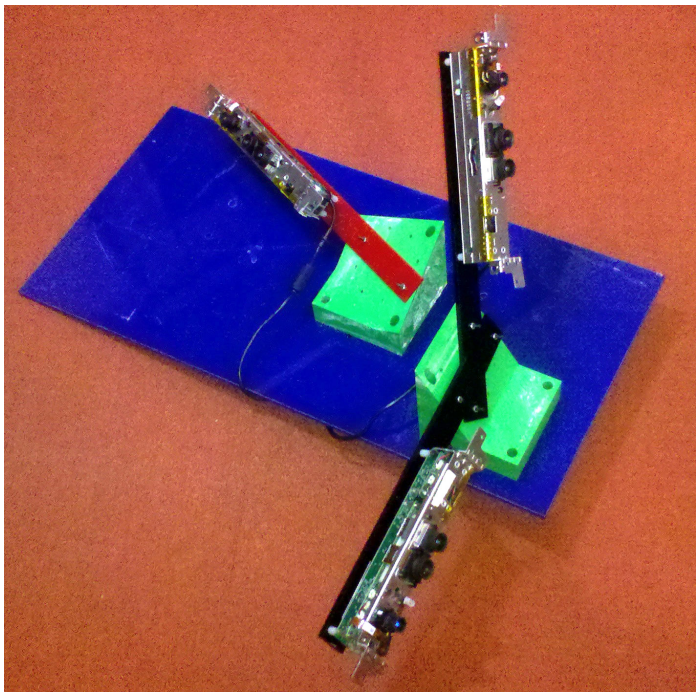


Figure 2-29: We assess the method's consistency by recovering the loop of calibrations relating three RGB-D sensors.

Table 2.3 lists the four true calibrations tested, and Table 2.4 summarizes the results. For clarity, the transforms are shown as translations and Euler angles, but all processing was done with DQ's. We assumed a constant variance for each DOF. The first three calibrations used Microsoft Kinects [50] and FOVIS [36] with $2N \simeq 2000$ observations, and the last calibration used Asus Xtions [1] and KinectFusion [55] with $2N \simeq 400$. In each case, the method recovered the inter-sensor translation to within about 1.4 cm and the rotation to within about 1.27 degrees.

We assessed the estimator's consistency with three rigidly mounted depth cameras r, s, and t (Figure 2-29). We estimated the pairwise calibrations $k_{s,r}$, $k_{t,s}$, and $k_{r,t}$, where, e.g., $k_{s,r}$ is the calibration between r and s. The closed loop of estimated transformations should return to the starting sensor frame:

$$k_{s,r} \circ k_{t,s} \circ k_{r,t} = \mathbf{I} \tag{2.114}$$

The accumulated error was small — the translation error was $[-4.27, -2.66, 7.13]$ mm and rotation error (again in Euler angles) was $[7.03, -5.20, -1.49]$ milliradians.

Chapter 3

Articulated Object Tracking

3.1 Overview

Imagine an autonomous robot, working at a construction site, such as the one shown in Figure 3-1. The robot might be responsible for some subset of human activities (e.g., transporting materials, digging, etc.) and be expected to work alongside humans, both on foot and operating construction equipment. In addition to performing its tasks, the robot must perceive, interact with, and avoid the construction equipment — this requires knowledge of the space occupied by the equipment. Whereas in the last chapter we estimated the parameters of a single-segment chain as part of calibration, in this chapter, we discuss techniques to track objects with more complex articulation. We demonstrate the techniques on examples including excavators, robot end-effectors, and household objects.

To understand some of the requirements for the articulated object tracker, we can closely examine the motivating construction site scenario. The construction equipment might include excavators, cranes, backhoes, concrete pumping trucks, scissor lifts, cherry pickers, etc. Each of these objects has several moving segments, but the motion of these segments is constrained with respect to one another. If we consider a kinematic model equivalent to an excavator, each link can move, but its movement relative to other links is constrained via the joints. Thus, information about the pose of the links, along with a kinematic model, provides some information about the pose of the entire system. Our goal is to take as input a kinematic model and noisy observations of the links, use the model to combine the observations, and output a valid estimate of the articulated object's configuration.

The construction site scenario presents several challenges and requirements for our tracking system:

1. **Safely avoid equipment.** The robot can be damaged by any part of the construction equipment, not just the end-effector/bucket. Parts of the operator cab and arm links must be avoided as well. As a result, our method must estimate the pose of the entire piece of equipment: the base pose and the joint

Figure 3-1: In this motivating construction site example, the two excavators must be tracked and avoided by an autonomous system. Yellow lines shows a notional kinematic chain provided as input to our tracking system.

values.

2. **Large equipment in a crowded environment.** Excavators, e.g., are enormous machinery and job sites are often crowded with equipment and materials. As a result, it is likely that part of the equipment will lie outside the field of view (FOV) of the robot's sensors. Thus, we must be able to leverage partial observations, when only a few links are visible.

3. **Few manufacturers and types of equipment.** There are relatively few equipment manufacturers (e.g., Caterpillar, John Deere, Komatsu) and types of equipment. As a result, it is reasonable to think that kinematic models might be available. When they are unavailable, the models might be generated by other means (see Section 3.2.1).

4. **Fielded equipment subject to wear-and-tear.** Although kinematic models may be available, we do not expect that the models will perfectly match the actual equipment. Our method should be reasonably robust to small deviations between the model and the actual system.

5. **Multiple types of equipment at site.** At any single job site, the robot may encounter several different kinds of equipment. Since it is not practical to design a tracking solution for each type of equipment, we desire a flexible framework, applicable to any articulated object.

6. **Equipment is not instrumented; limited communication (if any) with operators.** In the near term, it is unlikely that the robot will have any direct communication with the equipment itself or the human operators. Nor do we expect to have any estimate of the actual joint values from the equipment. As a result, all observations must be made remotely (e.g., via vision, LIDAR, etc.).

7. **Equipment is human-controlled.** Although the robot is autonomous, the construction equipment is manually operated. The human operator may not be predictable and the task may not be cyclic; it is unlikely we can learn meaningful predictive models. As a result, we expect to have relatively poor state transition/dynamic models.

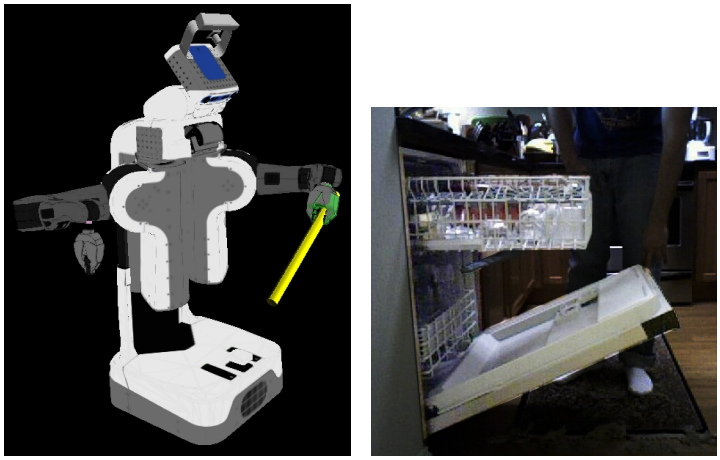
8. **Vehicles move on non-planar terrain.** Many construction vehicles have planar chassis, but often the ground will not be level (indeed, the ground's slope may change as the construction progresses). Thus, it is imperative to track both the N-DOF linkage and a 6-DOF base pose.

9. **Vehicles have redundant DOFs.** Given some subset of link observations, there may be more than one possible equipment configuration. While a historical estimate can be used to provide some temporal coherence, our tracker should also be able to maintain multiple hypotheses.

In addition to the construction site example, we also consider two additional domains. First, a robot arm is itself an articulated object. Despite being part of the robot, often there is still ambiguity in the end-effector's position. This uncertainty may be the result of errors in the sensor calibration, errors in the robot model, or imperfections in the mechanical construction. Additionally, the robot may be holding an object with non-rigid grip. For example, we consider a PR2 robot [77] holding a pipe which slips in the gripper (Figure 3-2a); since we can model the direction of slip, we can better estimate the pipe and the gripper. Second, articulated objects are also common around the house. A dishwasher, for example, has two drawers and a door, forming a 3-joint system (Figure 3-2b).

Motivated by the construction site, PR2, and dishwasher scenarios above, the goal is to create an articulated object tracker. As shown in Figure 3-3, the tracker accepts as input observations of the system. It also assumes access to a kinematic model, including the ability to calculate forward kinematics, calculate a kinematic Jacobian, and verify joint limit satisfaction. The system outputs an estimate of the articulated object's configuration, including the 6-DOF pose of a link (base) and the joint values. Given this configuration information, the entire pose of the object can be reconstructed.

We use a particle filter to track the articulated object's configuration. The particle filter maintains a discrete set of hypotheses. When new observations arrive, the particles are propagated primarily using the observation model. If the observation model is degenerate, the state transition model is used secondarily. We maintain a discrete approximation to the Optimal Importance Function (OIF) (see Section 3.2.4.2), which allows us to easily verify joint limits and calculate particle weights.



(a) (b)

Figure 3-2: A robot non-rigidly gripping a pipe and a dishwasher with a door and shelves are examples of articulated systems we consider.

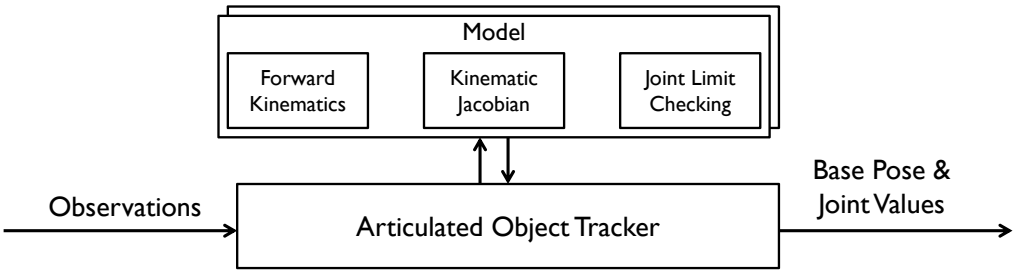


Figure 3-3: A stream of observations and a static kinematic model are inputs to the Articulated Object Tracker. The tracker estimates the object's base pose and joint values (i.e., the object's configuration).

The particle filter formulation also requires a diffusion step, where random perturbations are added to the particle's hypothesis. Adding this random noise in the state space, as called for in the typical particle filter implementation, results in a filter whose performance depends on the state parametrization. We wish to reduce this dependence because the parametrization is provided as input to our system (and, thus, not under our control). Instead, we add perturbation noise in the observation space to avoid this dependency. This observation space is an over-parametrization of the state space, so we employ related ideas from the calibration work (Section 2.5).

Figure 3-4 shows an example of the input and output for an excavator. The observations used in this example are image detections, such as those from an image tracker. In this work, we assume that the observations are associated (i.e., it is known which observation corresponds to which point on the kinematic chain) since this is often the case for high-level, image-based object detectors.

We demonstrate this technique on simple 2D kinematic chains (Section 3.5.2), a planar cross-section of a 3-DOF dishwasher (Section 3.5.3), a 9-DOF PR2 manipulation example (Section 3.5.4), and a 10-DOF construction equipment example

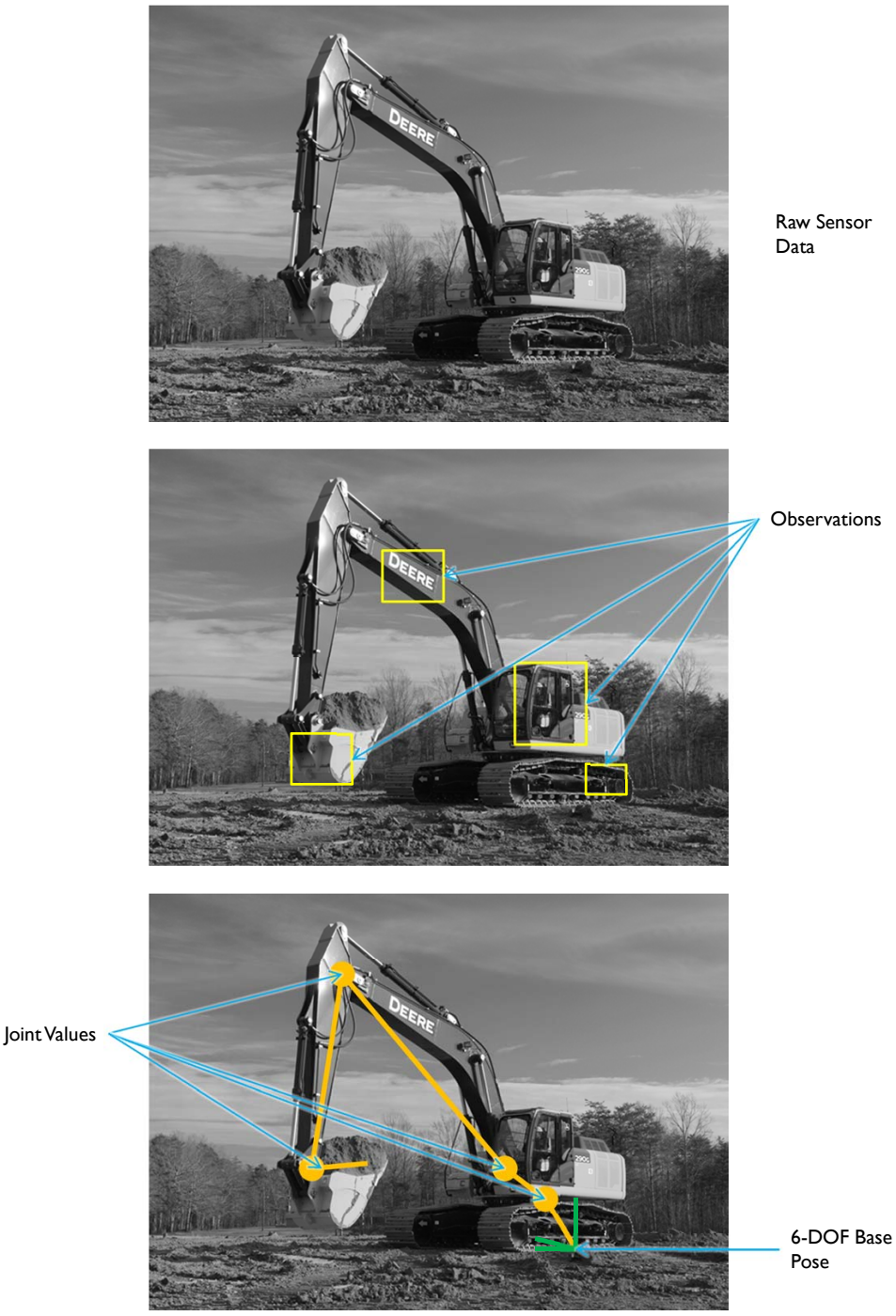


Figure 3-4: The goal of the articulated object tracker is to estimate the joint values and base pose of an articulated object, such as the excavator shown here. Observations might be image detections.

(Section 3.5.5). We compare our particle filtering method to a baseline particle filter and to an Unscented Kalman Filter (UKF) implementation.

3.1.1 Structure

Articulated object tracking has been most commonly studied in the context of human movement. We review previous work and possible sources for the kinematic description in Section 3.2. There, we also outline the particle and Kalman filters and discuss closely related ideas of optimization and velocity end-effector control. Our method is an extension of a particle filter, and we compare it to several other strategies. The alternative strategies of optimization, baseline particle filtering, and the UKF are described in Section 3.3.

Section 3.4 describes our method. We begin by describing a 2D implementation, then generalize to the 3D case. Pseudo-code is provided in Function 3.4.1 and Function 3.4.2. Finally, Section 3.5 describes our experiments on simulated and real data. Simulated work includes 2D kinematic chains, including multi-modal examples. Real experiments include a household dishwasher appliance, a PR2 robot manipulation example, and an excavator construction site example. Throughout, we compare our method to a baseline particle filter. For the excavator, we also compare against a UKF implementation and constant velocity state transition models.

3.1.2 Contributions

1. **Pose estimation for a generic, articulated object.** Our work is unique in that we focus on estimating the pose of a generic articulated object. We treat the kinematic model as a black box: we assume only the ability to calculate forward kinematics, a kinematic Jacobian, and to verify joint limits. This is distinct from most previous work, which has been designed to either estimate the kinematic structure itself (and not track its configuration) or has been designed to estimate the configuration of one specific articulated object (e.g., a model of a human).

2. **Incorporation of observations during particle proposal.** Although we assume access to a kinematic model, we do not assume access to a model of the system's dynamics. In other words, we do not have any specific information about how the object will move. As a result, we can assume only the simplest of motion models with generality. (We use zero-velocity and constant-velocity models.)

   As a result, our ability to predict a future position of an articulated structure will often be poor. That is, our state transition model will have high uncertainty. A particle filter which proposes from the state transition model (see Section 3.2.4.1) will perform poorly because particles will often not be in high probability regions.

Instead, we incorporate the current observations into particle proposal by creating a local linear approximation to the system kinematics. This avoids the need for complex, possibly ambiguous inverse kinematics.

3. **Proposal using observations, despite omissions or singular configurations.** Additionally, we do not require that all observations be made at each time step. The inputs to the algorithm are detections, e.g., from image processing techniques, which are prone to occasional failure. These failures are unavoidable in practice, and a system which required all observations at every time step would be brittle.

   It is important to note that the baseline particle filtering technique already gracefully handles missing observations. However, this is done by proposing from the state transition model and then updating against whatever observations are available. Thus, its ability to handle missing observations depends on proposing from the state transition model. Our method, on the other hand, gracefully handles missing observations and proposes using whatever observations are available.

   Further, we carefully handle situations where the observations and/or the configuration of the articulated object are degenerate. In these cases, our method relies only on the state transition model. To our knowledge, no previous work has addressed these issues.

4. **Perturbation of particles in the observation space.** Baseline particle filters typically have a diffusion step, where random noise is added to the hypotheses; this noise is often added in the state space. This additive Gaussian noise, however, causes the filter's performance to become dependent on the specific state parametrization used.

   Although not an issue in previous work, where the kinematic model (and, thus, the state parametrization) is known in advance or does not change, the performance of a filter in our setting can become dependent on the expression of the kinematic model. In other words, the same kinematic model, expressed in different ways, can significantly affect the tracking accuracy.

   In order to minimize these effects, we perturb hypotheses in the observation space. With no noise, the observation space can be considered a (possibly) over-parametrized representation of the state space. As a result, we can build on ideas of projected noise from the calibration chapter. The observation space remains fixed in our system and, thus, error performance is more nearly independent of the state parametrization.

5. **Characterization of effects of model noise.** As in some previous work, a kinematic model is assumed and used throughout our method. However, it is not reasonable to expect the model itself to match the physical object perfectly. We expect that the particle filter will handle this mismatch; however, we perform an analysis comparing the sensitivity of our technique to errors in the kinematic

model. To our knowledge, this type of analysis has not previously appeared in the literature.

3.2 Background

3.2.1 The Kinematic Model

The kinematic model is provided as an input to our system. In our implementation, this input is provided as a Unified Robot Description Format (URDF) file [78]. The URDF is parsed into a kinematic tree, consisting of joints and links, which we use to calculate forward kinematics, partial derivatives (Jacobians), and joint limits. This URDF can come from a variety of sources, including an object model database, user input, or some prior estimation process.

The object model database might contain URDF representations for a variety of articulated objects. These models can be generated from the original Computer Aided Design (CAD) files. In the case of construction equipment, for example, manufacturers might pre-load kinematic descriptions for tracking. The model for the simple kinematic chain (see Section 3.5.2), for example, was calculated from a CAD representation.

When a model is not available from the database, a human user might annotate sensor data to provide an *ad hoc* model. A user could employ a 3D drafting program to adjust model parameters to match 3D point cloud and image data, for example. In this way, the user can customize the model to match a particular object.

Finally, related work has shown success automatically estimating a kinematic model from a moving, articulated object. These strategies generally proceed by tracking the movement of rigid bodies (links), then estimating the kinematic relationships between these bodies (joints). In contrast, we observe the links and attempt to estimate the most likely configuration of the object (joint values).

Sturm [68], for example, gives a method to compute the probability of particular joint types, finding a kinematic tree that maximizes the joint probability over all joints, offset with a penalty for model complexity. The authors construct a graph where vertices correspond to links and edges correspond to joint type (prismatic or rotational). The authors then use a combination of consensus sampling and Expectation Maximization (EM) [22] to determine the most likely kinematic parameters.

Katz [43] also gives a method to recover the DOFs of an articulated system. The authors actively manipulate an object and track image features. Using spatial, temporal, and appearance criteria, the features are segmented into clusters of rigidly co-moving parts. The resulting rigid bodies are then considered pairwise and classified as independent or as related by a fixed, rotational, or prismatic joint.

In an effort to enable door and drawer manipulation by robots, other researchers have also shown how to estimate the state of such objects. These methods are often specific to a single revolute or prismatic joint (e.g. [63, 38, 54]) and focus on estimating

the geometric parameters (e.g., door radius or drawer traversal axis).

3.2.2 Generic Articulated Object Tracking

Much of the research in articulated object tracking is focused on tracking parts of the human body (see Section 3.2.3). The only articulated tracking work validated on several different kinematic objects, of which we are aware, is by Comport [20]. Comport tracks an articulated object by optimizing a single hypothesis at each time step. The hypothesis is projected into an image and errors in edge correspondences are correlated to velocities of the kinematic chain. The technique is demonstrated on a rotational joint, prismatic joint, helical joint, and a 2-DOF robotic arm. Their primary contribution is an efficient computation of the Jacobian between edge errors (i.e., errors between image edge detections and projected edges) and joint velocities. Their technique is specific to image detections and, as it maintains a single greedy hypothesis, will be subject to local minima. Additionally, their method assumes all links are observed at all times.

3.2.3 Articulated Human Tracking

Deformable part models [27] have been widely used for object detection, especially for person tracking. These methods attempt to learn a model of each articulated object and to track it. Learning typically occurs in image space, hindering applicability to general mechanisms and motions.

In related work, Deutscher [23] used what they referred to as an "annealing particle filter" to estimate the configuration of a 29-DOF model of the human body. Similar to our work, they wished to track a multi-modal articulated object. In order to reduce the number of particles, they also suggested proposing particles based on observations. However, instead of relying on a local tangent space for proposal, they employed a series of weighting functions designed to iteratively guide particles toward a global maximum in an annealing process. Although effective, the weighting functions must be customized to the application (in their case, the human body) and the method is not easily applicable to a generic kinematic tree.

Whereas we focus on situations where the process model is poor or unavailable, other researchers have focused on developing more accurate motion models. Urtasun [74] developed a Gaussian process technique which learns motion models. The algorithm accepts image detections as training data, along with a 20-DOF model of the human body, and learns motion models for cyclic motion (e.g., walking or golf swings). The learned model allows the pose of the human to be estimated during periods of occlusions, but is not appropriate for less predictable motions.

Ziegler [81] tracked the upper torso using a 14-DOF model. Using a state vector consisting of the torso pose and joint values, the authors generated random noise for the system model of a UKF. The measurement model was obtained by projecting observations onto a 3D rendering. Although effective, their method would not handle

multi-modal distributions (due to the use of the UKF) and is specialized to 3D point cloud data.

Also recognizing the complexity of tracking articulated object with particle filters (due to the high number of DOFs and resulting large number of particles), the authors of [59, 58] decompose the task of tracking a single, large kinematic chain into tracking smaller chains. Each link is tracked in image space and optimized in a pairwise fashion with links connected via an immediate joint. With this simplification, they demonstrated real-time performance. However, the algorithm will generally fail to produce a global, joint optimization.

Torso tracking has also been adapted to GPU execution. Cabido [13] uses a particle filter to represent multiple hypotheses for an 8-DOF configuration of the upper torso. Each hypothesis provides a segmentation to the input image, which is then reorganized so that every link (e.g., arms, torso, head) is in a fixed position in a template window. Each template window is convolved over the input image on the GPU and the most likely candidates are propagated to the next time step. The work is primarily limited to planar articulated object tracking.

Research interests have also focused on tracking the human hand. Rehg [60] tracked a 9-DOF model of two fingers on a hand. By reasoning about occlusions for the current state, they reduced the search space for template matching in the image. The authors extended the work in [16, 61], developing a 2D kinematic chain which approximated a 3D kinematic chain, and then tracked with these 2D kinematic constraints. The authors showed that the motion of the 2D chain is well defined even along the unobserved depth axis of the camera; this enabled elimination of some of the unobservable dimensions with the 2D chain. By contrast, our method relies on the state transition model when unobserved dimensions arise.

3.2.4 Particle Filter (PF)

In a filtering application, the goal is to recover the most likely current state of the system from a series of observations. In this work, the state of the system will be the base pose and the joint values. The observations will be measurements ranging from 6-DOF poses (e.g., the pose of a PR2 robot manipulator Section 3.5.4) or 2-DOF image features (e.g., the pixel coordinates for features on a dishwasher Section 3.5.3).

A particle filter [37] maintains a discrete approximation to a probability distribution using a collection of state hypotheses, or particles. Each particle is propagated across filter steps according to a state transition model, an observation model, and actual observations. With an infinite number of particles, a particle filter can be implemented trivially. In practice, a finite number of particles will eventually result in degeneracy (only a single particle is likely), and periodic resampling is required [24].

The goal of a particle filter is to find state estimates, $x_{0:n}$, which maximize $P(x_{0:n} | z_{0:n})$. In other words, we wish to find the series of most probable states, given the observations. In general, it will not be possible to sample directly from this "target distribution" $P(x_{0:n} | z_{0:n})$. However, we can employ importance sampling

and sample from an alternative "proposal distribution" $\pi\left(x_{0:n}\left|z_{0:n}\right.\right)$. Then, we can weight each particle according to the ratio of the target and proposal distributions; these weighted samples will represent/approximate the target distribution. Thus, the weight can be defined as:

$$w = \frac{P\left(x_{0:n}\left|z_{0:n}\right.\right)}{\pi\left(x_{0:n}\left|z_{0:n}\right.\right)} \tag{3.1}$$

We can then apply Bayes rule to the numerator:

$$w = \frac{P\left(z_{0:n}\left|x_{0:n}\right.\right) \cdot P\left(x_{0:n}\right)}{P(z_{0:n}) \cdot \pi\left(x_{0:n}\left|z_{0:n}\right.\right)} \tag{3.2}$$

Applying the Markov independence assumptions yields:

$$w = \frac{P\left(z_0\left|x_0\right.\right)\prod_k P\left(z_k\left|x_k\right.\right) \cdot P\left(x_0\right)\prod_k P\left(x_k\left|x_{k-1}\right.\right)}{P(z_{0:n}) \cdot \pi\left(x_{0:n}\left|z_{0:n}\right.\right)} \tag{3.3}$$

$$w = \frac{P\left(z_0\left|x_0\right.\right) \cdot P\left(x_0\right)}{P(z_{0:n}) \cdot \pi\left(x_{0:n}\left|z_{0:n}\right.\right)} \prod_k P\left(z_k\left|x_k\right.\right) \cdot P\left(x_k\left|x_{k-1}\right.\right) \tag{3.4}$$

Noting that $P(z_{0:n})$ is a constant,

$$w \propto \frac{P\left(z_0\left|x_0\right.\right) \cdot P\left(x_0\right)}{\pi\left(x_{0:n}\left|z_{0:n}\right.\right)} \prod_k P\left(z_k\left|x_k\right.\right) \cdot P\left(x_k\left|x_{k-1}\right.\right) \tag{3.5}$$

The proposal distribution in the denominator can be further simplified with Bayes rule:

$$\begin{aligned}
\pi\left(x_{0:n}\left|z_{0:n}\right.\right) &= \pi\left(x_n\left|x_{0:n-1}, z_{0:k}\right.\right) \cdot \pi\left(x_{0:n-1}, z_{0:k}\right) && (3.6)\\
&= \pi\left(x_n\left|x_{0:n-1}, z_{0:k}\right.\right) \cdot \pi\left(x_{n-1}\left|x_{0:n-2}, z_{0:k}\right.\right) \cdot \pi\left(x_{0:n-2}, z_{0:k}\right) && (3.7)\\
&= \pi\left(x_0\left|z_{0:n}\right.\right) \prod_k \pi\left(x_k\left|x_{0:k-1}, z_{0:n}\right.\right) && (3.8)
\end{aligned}$$

In many cases, Equation 3.5 and Equation 3.8 can be further simplified using a causal assumption (i.e., the past does not depend on the future):

$$\pi\left(x_{0:n}\left|z_{0:n}\right.\right) = \pi\left(x_0\left|z_0\right.\right) \prod_k \pi\left(x_k\left|x_{0:k-1}, z_{0:k}\right.\right) \tag{3.9}$$

The particle weights can now be written recursively, and depend only on the current and past observations. The recursive formula for the weights is:

$$w_0^{(i)} \propto \frac{P\left(z_0\left|x_0^{(i)}\right.\right) \cdot P\left(x_0^{(i)}\right)}{\pi\left(x_0^{(i)}\left|z_0\right.\right)} \tag{3.10}$$

$$w_k^{(i)} \propto \frac{P\left(z_k\left|x_k^{(i)}\right.\right) \cdot P\left(x_k^{(i)}\left|x_{k-1}^{(i)}\right.\right)}{\pi\left(x_k^{(i)}\left|x_{0:k-1}^{(i)}, z_{0:k}\right.\right)} w_{k-1}^{(i)} \tag{3.11}$$

Algorithm 3.2.1 shows the complete filtering technique. A new sample state is drawn from the proposal distribution, given the previous state and the observations. (Here, our proposal distribution is assumed to model a Markovian system, where we depend only on the most recent state and observation.) Again, since the proposal and target distributions differ, the correcting importance weight is calculated. These weights are then normalized to account for the missing normalization constant (i.e., the proportionality constraint in Equation 3.11). The best current estimate is often obtained by averaging (e.g., computing the weighted mean or mode) the current set of particles.

Algorithm 3.2.1: PF(x_{k-1}, w_{k-1}, R_k, z_k, Σ_{z_k})

inputs : x_{k-1}, w_{k-1} previous N_s particles and weights
R_k state transition noise covariance
z_k, Σ_{z_k} current observations and covariance

outputs: x_k, w_k new particles and weights
$\overline{x_k}$ current estimate

$\{x_k, \alpha_k\} \leftarrow$ `Proposal(`x_{k-1}, R_k, z_k, Σ_{z_k}`)`

for $i = 1$ **to** N_s **do**
  $w_k^{(i)} \leftarrow \alpha_k^{(i)} w_{k-1}^{(i)}$ `// update weights`

for $i = 1$ **to** N_s **do**
  $w_k^{(i)} \leftarrow \frac{w_k^{(i)}}{\sum_{j=0}^{N_s} w_k^{(j)}}$ `// Normalize weights`

$N_{\text{eff}} \leftarrow \left(\sum_{j=0}^{N_s} \left(w_k^{(j)} \right)^2 \right)^{-1}$

if $N_{\text{eff}} < N_t$ **then** `// resample if` N_{eff} `below threshold` N_t
  $x_k \leftarrow$ `Resample(`x_k, w_k`)`
  $w_k \leftarrow N_s^{-1}$

$\overline{x_k} \leftarrow$ `WeightedMean(`x_k, w_k`)`

Function 3.2.2: Proposal(x_{k-1}, R_k, z_k, Σ_{z_k})

outputs: x_k, α_k new particles and weight scale

`// Sample next generation of particles and calculate weight scale`
for $i = 1$ **to** N_s **do**
  $x_k^{(i)} \backsim \pi\left(x_k^{(i)} \middle| x_{k-1}^{(i)}, z_k; R_k, \Sigma_{z_k} \right)$ `// sample from the proposal`
  $\alpha_k^{(i)} \leftarrow \frac{P\left(z_k \middle| x_k^{(i)}; \Sigma_{z_k}\right) \cdot P\left(x_k^{(i)} \middle| x_{k-1}^{(i)}; R_k\right)}{\pi\left(x_k^{(i)} \middle| x_{k-1}^{(i)}, z_k\right)}$ `// update per Equation 3.11`

A common performance metric for particle filters is the number of effective parti-

cles:

$$N_{\text{eff}} = \left(\sum_{i=1}^{N_s} \left(w_k^{(i)} \right)^2 \right)^{-1} \tag{3.12}$$

where, again, $w_k^{(i)}$ is the weight of the i-th particle at time k. When all particles have roughly the same weights and represent the distribution well, N_{eff} will approach the number of particles, N_s. When the filter becomes degenerate, N_{eff} will approach unity. If it falls below some threshold, we use $N_t = N_s/5$, resampling is performed.

The purpose of resampling is to eliminate particles which do not have significant weight and focus computation on highly probable states (i.e., particles with significant weight). During resampling, a new set of particles is drawn, with replacement, from the previous set, according to their weights. In this way, low probability particles are mostly ignored and heavily weighted particles are repeated.

In this work, we use Latin Hypercube Sampling (LHS) [64]. In this resampling method, the cumulative probability distribution (CDF) is generated for the samples using the weights. The CDF is divided into N_s equally spaced ranges, a sample is drawn from each range bin, and the corresponding particle is propagated to the next generation. Unlike a simple random sampling strategy, LHS will guarantee that low probability samples (near the tails of the CDF) are always generated.

The primary advantage of the particle filter is its ability to represent an arbitrary distribution. As described in Section 3.3.1, this ability was important in our application to represent the multi-modal nature of the configuration of the kinematic chains. However, particle filters can exhibit degeneracy when a single particle has most of the weight. In our case, we use a resampling step to avoid this. Particle filters can also suffer from a sample impoverishment problem, when resampling causes just a few samples to be duplicated; as a result, sample diversity is lost. As seen in Section 3.4, our method rarely exhibits this problem since it maintains a higher N_{eff} and reduces the need for resampling.

3.2.4.1 State Transition Proposal

When using a particle filter, considerable flexibility remains when choosing the proposal distribution. A common choice is to use the state transition model:

$$\pi \left(x_k^{(i)} \middle| x_{0:k-1}^{(i)}, z_{0:k} \right) = P \left(x_k^{(i)} \middle| x_{k-1}^{(i)} \right) \tag{3.13}$$

Then, Equation 3.11 reduces to:

$$w_k^{(i)} \propto \frac{P \left(z_k \middle| x_k^{(i)} \right) \cdot P \left(x_k^{(i)} \middle| x_{k-1}^{(i)} \right)}{P \left(x_k^{(i)} \middle| x_{k-1}^{(i)} \right)} w_{k-1}^{(i)} \tag{3.14}$$

$$w_k^{(i)} \propto P \left(z_k \middle| x_k^{(i)} \right) w_{k-1}^{(i)} \tag{3.15}$$

and only requires the observation model. Thus, for this proposal distribution, only the readily available state transition and observation models are required. A flow diagram is shown in Figure 3-5.

Using the state transition model for proposal is a good choice when it results in particles which are in the high probability regions of $P\left(z_k \left| x_k^{(i)} \right.\right)$. In other words, it is a good choice when the state transition model is relatively accurate. However, when the state transition model is poor (as in our setting), the proposed particles may result it relatively low weights, eventually causing particle degeneracy.

This proposal distribution has been used successfully with many Simultaneous Localization and Mapping (SLAM) tasks [70]. Given the robot's commanded motion and previous particle, a new particle is proposed using the state transition model. The particle weights are then calculated, expressing how well the observations (e.g., LIDAR readings) agree with the proposed particles.

3.2.4.2 Optimal Importance Function (OIF) proposal

There are many other possible choices for the target distribution, $\pi\left(x_k^{(i)} \left| x_{0:k-1}^{(i)}, z_{0:k} \right.\right)$. The optimal choice is the distribution which minimizes the variance in the weights of the particles [24]. The optimal importance function is well known to be:

$$\pi\left(x_k^{(i)} \left| x_{0:k-1}^{(i)}, z_{0:k} \right.\right) = \; P\left(x_k^{(i)} \left| x_{k-1}^{(i)}, z_k \right.\right) \tag{3.16}$$

It is often not possible to sample directly from this OIF and, instead, we make an approximation. When available, however, this choice for the target proposal results in a zero variance on the particle weights, resulting in each particle representing an equal portion of the target distribution. To see this, first we calculate the expected value of the weights using the definition of expected value:

$$E_\pi\left[w_k^{(i)}\right] = \int \frac{P\left(z_k \left| x_k \right.\right) \cdot P\left(x_k \left| x_{k-1}^{(i)} \right.\right)}{\pi\left(x_k \left| x_{0:k-1}^{(i)}, z_{0:k} \right.\right)} \cdot w_{k-1}^{(i)} \cdot \pi\left(x_k \left| x_{0:k-1}^{(i)}, z_{0:k} \right.\right) dx_k \tag{3.17}$$

$$= w_{k-1}^{(i)} \int P\left(z_k \left| x_k \right.\right) \cdot P\left(x_k \left| x_{k-1}^{(i)} \right.\right) dx_k \tag{3.18}$$

$$= w_{k-1}^{(i)} \int \frac{P\left(z_k \left| x_k \right.\right) \cdot P\left(x_k \left| x_{k-1}^{(i)} \right.\right) P\left(x_{k-1}^{(i)}\right)}{P\left(x_{k-1}^{(i)}\right)} dx_k \tag{3.19}$$

$$= w_{k-1}^{(i)} \int \frac{P\left(z_k, x_k, x_{k-1}^{(i)}\right)}{P\left(x_{k-1}^{(i)}\right)} dx_k \tag{3.20}$$

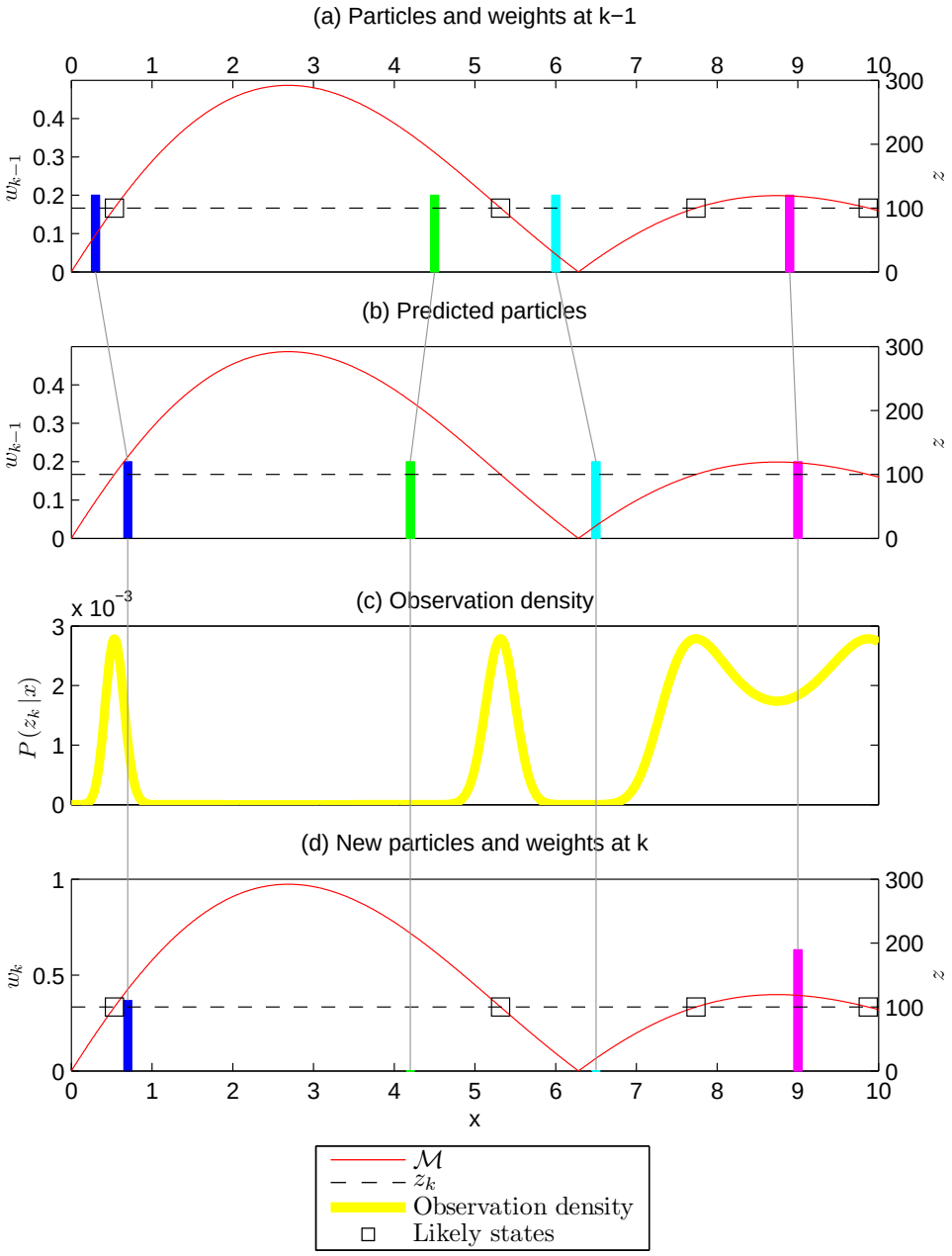


Figure 3-5: The current observation, z_k (black dotted line), intersects the possible configurations (red line) at four places which indicates four possible configurations that explain the observation. For each cycle of the particle filter algorithm, the previous generation of particles (a) is used to sample from a proposal distribution ($\mathcal{M}$ is $f(x)$, the observation manifold). In (a), particles are represented by their position along the x-axis; their weight is their height. Particles are colored consistently from (a)-(d). Here, the state transition model $P(x|x_{k-1})$ is used to predict the particle evolution (b). The weights are updated via the observation density, $P(z|x)$, in (c) and the next generation of particles result (d). Resampling (not shown) may then be necessary.

x_k is effectively marginalized out, and

$$E_\pi\left[w_k^{(i)}\right] = w_{k-1}^{(i)} \frac{P\left(z_k, x_{k-1}^{(i)}\right)}{P\left(x_{k-1}^{(i)}\right)} \tag{3.21}$$

$$= w_{k-1}^{(i)} \cdot P\left(z_k \left| x_{k-1}^{(i)}\right.\right) \tag{3.22}$$

With the expectation, the variance of the weights can be evaluated:

$$E_\pi\left[\left(w_k^{(i)} - E_\pi\left[w_k^{(i)}\right]\right)^2\right] \tag{3.23}$$

$$= E_\pi\left[\left(w_k^{(i)}\right)^2\right] - \left(E_\pi\left[w_k^{(i)}\right]\right)^2 \tag{3.24}$$

$$= \int \left[\frac{P\left(z_k \left| x_k\right.\right) \cdot P\left(x_k \left| x_{k-1}^{(i)}\right.\right)}{\pi\left(x_k \left| x_{0:k-1}^{(i)}, z_{0:k}\right.\right)} \cdot w_{k-1}^{(i)}\right]^2 \cdot \pi\left(x_k \left| x_{0:k-1}^{(i)}, z_{0:k}\right.\right) dx_k - \left(w_{k-1}^{(i)} \cdot P\left(z_k \left| x_{k-1}^{(i)}\right.\right)\right)^2 \tag{3.25}$$

$$= \left(w_{k-1}^{(i)}\right)^2 \left[\int \frac{\left[P\left(z_k \left| x_k\right.\right) \cdot P\left(x_k \left| x_{k-1}^{(i)}\right.\right)\right]^2}{\pi\left(x_k \left| x_{0:k-1}^{(i)}, z_{0:k}\right.\right)} \cdot dx_k - P\left(z_k \left| x_{k-1}^{(i)}\right.\right)^2\right] \tag{3.26}$$

$$= \left(w_{k-1}^{(i)}\right)^2 \left[\int \frac{P\left(z_k, x_k, x_{k-1}^{(i)}\right)^2}{P\left(x_{k-1}^{(i)}\right)^2 \cdot \pi\left(x_k \left| x_{0:k-1}^{(i)}, z_{0:k}\right.\right)} \cdot dx_k - P\left(z_k \left| x_{k-1}^{(i)}\right.\right)^2\right] \tag{3.27}$$

$$\tag{3.28}$$

$$= \left(w_{k-1}^{(i)}\right)^2 \left[\int \frac{P\left(x_k^{(i)} \left| x_{k-1}^{(i)}, z_k\right.\right) \cdot P\left(z_k \left| x_{k-1}^{(i)}\right.\right) \cdot P\left(x_{k-1}^{(i)}\right) \cdot P\left(z_k, x_k, x_{k-1}^{(i)}\right)}{P\left(x_{k-1}^{(i)}\right)^2 \cdot \pi\left(x_k \left| x_{0:k-1}^{(i)}, z_{0:k}\right.\right)} \cdot dx_k \right.$$

$$\left. - P\left(z_k \left| x_{k-1}^{(i)}\right.\right)^2\right] \tag{3.29}$$

Substituting in the OIF from Equation 3.16, we obtain:

$$= \left(w_{k-1}^{(i)}\right)^2 \left[\int \frac{P\left(x_k \left| x_{k-1}^{(i)}, z_k\right.\right) \cdot P\left(z_k \left| x_{k-1}^{(i)}\right.\right) \cdot P\left(z_k, x_k, x_{k-1}^{(i)}\right)}{P\left(x_{k-1}^{(i)}\right) \cdot P\left(x_k \left| x_{k-1}^{(i)}, z_k\right.\right)} \cdot dx_k - P\left(z_k \left| x_{k-1}^{(i)}\right.\right)^2\right] \tag{3.30}$$

$$= \left(w_{k-1}^{(i)}\right)^2 \left[P\left(z_k \left| x_{k-1}^{(i)}\right.\right) \int \frac{P\left(z_k, x_k, x_{k-1}^{(i)}\right)}{P\left(x_{k-1}^{(i)}\right)} \cdot dx_k - P\left(z_k \left| x_{k-1}^{(i)}\right.\right)^2\right] \tag{3.31}$$

$$= \left(w_{k-1}^{(i)}\right)^2 \left[P\left(z_k \left| x_{k-1}^{(i)}\right.\right) P\left(z_k \left| x_{k-1}^{(i)}\right.\right) - P\left(z_k \left| x_{k-1}^{(i)}\right.\right)^2\right] \tag{3.32}$$

$$E_\pi\left[\left(w_k^{(i)} - E_\pi\left[w_k^{(i)}\right]\right)^2\right] = 0 \tag{3.33}$$

Thus, when the OIF can be used, the variance of the weights is zero. (In Section 3.4, our method uses an approximation to the OIF.)

Substituting into Equation 3.11, we can solve for the weight update recursion for the OIF:

$$w_k^{(i)} \propto \frac{P\left(z_k \left| x_k^{(i)}\right.\right) \cdot P\left(x_k^{(i)} \left| x_{k-1}^{(i)}\right.\right)}{P\left(x_k^{(i)} \left| x_{k-1}^{(i)}, z_k\right.\right)} w_{k-1}^{(i)} \tag{3.34}$$

$$w_k^{(i)} \propto \frac{P\left(z_k \left| x_k^{(i)}\right.\right) \cdot P\left(x_k^{(i)} \left| x_{k-1}^{(i)}\right.\right)}{\left[\frac{P\left(z_k|x_k^{(i)}\right) \cdot P\left(x_k^{(i)}|x_{k-1}^{(i)}\right)}{P\left(z_k|x_{k-1}^{(i)}\right)}\right]} w_{k-1}^{(i)} \tag{3.35}$$

$$w_k^{(i)} \propto P\left(z_k|x_{k-1}^{(i)}\right) w_{k-1}^{(i)} \tag{3.36}$$

Further, using Bayes rule on the OIF, it can be shown that:

$$P\left(x_k \left| x_{k-1}^{(i)}, z_k\right.\right) = \frac{P\left(z_k|x_k\right) \cdot P\left(x_k|x_{k-1}^{(i)}\right)}{P\left(z_k|x_{k-1}^{(i)}\right)} \tag{3.37}$$

The denominator, $P(z_k|x_{k-1}^{(i)})$, is the probability of the observation, given the previous state. In general we will not know this quantity; however, we can marginalize over the current state:

$$P\left(z_k|x_{k-1}^{(i)}\right) = \int P\left(z_k|x\right) \cdot P\left(x|x_{k-1}^{(i)}\right) dx \tag{3.38}$$

Note that the two terms in the integral are analogous to the two terms in the numerator of Equation 3.16 and can be readily calculated.

3.2.5 Unscented Kalman Filter (UKF)

Due to the multiple local optima often encountered when attempting to track a 3D articulated object, we make use of a PF to capture the multi-modal nature. To highlight the importance, we will compare several examples against an Unscented Kalman filter (UKF), which is capable of representing only a uni-modal distribution (see Section 3.5.1.5 and Section 3.5.5 for examples). Somewhat similar to the PF, the UKF uses discrete samples; however, the UKF carefully selects these samples (i.e., sigma points) such that a resulting Gaussian approximation can be easily computed.

The Kalman filter (KF) is the minimum-variance state estimator for a linear system with Gaussian noise. With the possibility of rotational joints, however, our kinematic chain is non-linear and a KF is inappropriate. The Extended Kalman filter

(EKF) uses a Taylor approximation to produce a locally linear system, to which a KF can then be applied. Alternatively, the UKF utilizes a discrete approximation to propagate hypotheses through a non-linear function. This sigma point hypotheses technique was used in the calibration work (Section 2.4.5.2) to estimate the covariance after resampling.

Algorithm 3.2.3 shows a typical implementation of the UKF [40]. Here, the state at frame k, x_k, will represent a base pose and the values of the joints. The observations z_t will be a series of 6-DOF poses with uncertainty covariance matrix Σ_z. Because of the $SE(3)$ poses, Algorithm 3.2.3 uses the $\boxplus$ and $\boxminus$ operators, indicating that the addition and subtraction are done on the manifold of valid configurations. That is, the base pose is added and subtracted using the Lie algebra, and joint values are added in Euclidean space. The individual observations, which are elements in $SE(3)$, are composed in the Lie algebra. Similarly, the `WeightedStateMean` and `WeightedObservationMean` approximate the mean on the corresponding manifolds (see Section 2.5.8.1).

The UKF in Algorithm 3.2.3 uses sigma points and propagates them through the non-linear `StateTransition` and `Observation` functions. A detailed example of the Scaled Unscented Transform for time resampling is provided in Section 2.4.5.3 for extrinsic sensor calibration.

3.2.5.1 UKF Joint Limits

We also wish to handle joint limit checking, which Algorithm 3.2.3 does not provide. In its default form, the UKF returns a mean and covariance which lie in the state space (i.e., base pose and joint values). As a result, neither the mean, nor any other sample from the distribution, will necessarily obey joint limits.

One solution described in [66] is to find the most likely member, x_k, of the distribution subject to the joint limits. For example:

$$\hat{x_k} = \underset{x}{\operatorname{argmin}} \left((x - x_k)^T \Sigma_x^{-1} (x - x_k)) \right) \tag{3.39}$$

subject to Verify$(x) = 1$, where the `Verify` function returns unity only if no joint limits are violated, and 0 otherwise. Here, we desire $\hat{x_k}$ which satisfies the `Verify` and is nearest to the mean, x_k, in the Mahalanobis sense. In general, the optimization implied by Equation 3.39 is difficult, since the `Verify` function provides no opportunity to calculate a gradient.

However, in our domain, the `Verify` function will usually check only that individual joints, q are within some bounds: e.g., $\frac{-\pi}{2} < q < \frac{\pi}{2}$. (By contrast, the joint limits on a dishwasher are not as simple, see Equation 3.98.) Consequently, the joint limit inequalities are typically linear and can be expressed in the form $Dx \leq d$. Thus, the problem can often be restated as:

$$\hat{x_k} = \underset{x}{\operatorname{argmin}} \left((x - x_k)^T \Sigma_x^{-1} (x - x_k)) \right) \tag{3.40}$$

Algorithm 3.2.3: UKF(x_{k-1},$\Sigma_{x_{k-1}}$,R_k,z_k,Σ_{z_k})

inputs : x_{k-1}, $\Sigma_{x_{k-1}}$ previous $M \times 1$ state and $M \times M$ covariance
R_k state transition noise covariance
z_k, Σ_{z_k} current observations and covariance

outputs: x_k, Σ_x new state estimate and covariance

`// Predict the state estimate`

$$\mathcal{X}_{k-1} = \begin{cases} x_{k-1} & i = 0 \\ x_{k-1} \boxplus \langle \Sigma_{x_{k-1}} \rangle_i & 1 \leq i \leq M \\ x_{k-1} \boxplus -\langle \Sigma_{x_{k-1}} \rangle_{i-M} & M+1 \leq i \leq 2M \end{cases}$$ `// per Equation 2.50`

$\mathcal{X}'_{k-1} \leftarrow$ `StateTransition(`$\mathcal{X}_{k-1}$`)`

$\overline{x_k} \leftarrow$ `WeightedStateMean(`$\mathcal{X}'_{k-1}$, W_m`)` `// per Equation 2.52`

$\overline{\Sigma_{x_k}} \leftarrow \sum_{i=0}^{2M} W_{c,i} \left(\mathcal{X}'_{k-1,i} \boxminus \overline{x_k}\right) \left(\mathcal{X}'_{k-1,i} \boxminus \overline{x_k}\right)^T + R_k$

`//` $\overline{x_k}$ `and` $\overline{\Sigma_{x_k}}$ `are state and uncertainty after prediction`

`// Update the state estimate with observations`

$$\mathcal{Y}_k = \begin{cases} \overline{x_k} & i = 0 \\ \overline{x_k} \boxplus \langle \overline{\Sigma_{x_k}} \rangle_i & 1 \leq i \leq M \\ \overline{x_k} \boxplus -\langle \overline{\Sigma_{x_k}} \rangle_{i-M} & M+1 \leq i \leq 2M \end{cases}$$

$Z_k \leftarrow$ `Observation(`$\mathcal{Y}_k$`)`

$\hat{z_k} \leftarrow$ `WeightedObservationMean(`Z_k, W_m`)`

$P_{zz} \leftarrow \sum_{i=0}^{2M} W_{c,i} \left(Z_{k,i} \boxminus \hat{z_k}\right) \left(Z_{k,i} \boxminus \hat{z_k}\right)^T + \Sigma_{z_k}$

$P_{xz} \leftarrow \sum_{i=0}^{2M} W_{c,i} \left(\mathcal{Y}_{k,i} \boxminus \overline{x_k}\right) \left(Z_{k,i} \boxminus \hat{z_k}\right)^T$

$K_k \leftarrow P_{xz} P_{zz}^{-1}$ `// compute Kalman gain`

$x_k \leftarrow \overline{x_k} \boxplus K_k \left(z_k \boxminus \hat{z_k}\right)$

$\Sigma_x \leftarrow \overline{\Sigma_{x_k}} - K_k P_{zz} K_k^T$

subject to $Dx \leq d$, where D and d are provided as inputs instead of `Verify`.

A solution to this optimization is provided by active set [8] quadratic programming. If D and d are modified to only included the violated (i.e., active) constraints, then the solution is:

$$\hat{x_k} = x_k - \Sigma_x^{-1}\hat{D}^T\left(\hat{D}\Sigma_x^{-1}\hat{D}^T\right)^{-1}\left(\hat{D}x_k - \hat{d}\right) \tag{3.41}$$

Notice this projection is the pseudo-inverse, similar to the one used in our particle propagation method (see Section 3.4). In this way, the output of the UKF can be guaranteed to satisfy linear joint limits. In situations where the joint limits are non-linear, such as the dishwasher example in Section 3.5.3, an optimization process can be used to find a satisfying solution. Note, however, that this may require the derivative of the joint limit equations.

3.2.6 Manipulator Velocity Control

In Section 3.4, we will relate perturbations of physical points on an articulated object to perturbations of the state of the articulated object. If we consider these perturbations as velocities, then a close analog exists with robotic end-effector velocity control [32, 12, 76, 18]. Velocity control of a robotic end-effector often requires a way to relate desired velocities of the manipulator to desired velocities of the joints. Analogously, in our case, we wish to relate velocities (perturbations) of observations (points on the kinematic chain) to state velocities (velocities of the base pose and joint values). As a result, we briefly review Jacobian velocity control here.

The goal of Jacobian velocity control is to control the joint velocities so as to produce some desired 6-DOF end-effector velocity. The end-effector pose, x, can be related via the forward kinematics, f, to the joint positions, q, via:

$$x = f(q) \tag{3.42}$$

In this section, x is a 6×1 vector containing, for example, translation and Euler angles, and q is an $M \times 1$ vector containing the M joint positions. For a redundant manipulator, note that $M > 6$.

The Jacobian of f, J, is $6 \times M$, such that:

$$\dot{x} = J\dot{q} \tag{3.43}$$

relating the joint velocities, $\dot{q}$, to the end-effector velocities, $\dot{x}$. If J were invertible (e.g., $M = 6$ and $\text{rank}(J) = 6$), then we could simply solve for the joint velocities via:

$$\dot{q} = J^{-1}\dot{x} \tag{3.44}$$

For a redundant manipulator, however, J is not invertible. The issue is that the redundant joints cannot be solved for uniquely. One strategy, used in the pseudo-inverse, is to establish a cost function and then optimize over the joints to minimize this cost function.

The pseudo-inverse finds the solution to the least squares problem:

$$\dot{q}_{PI} = \underset{\dot{q}}{\operatorname{argmin}} \left| J\dot{q} - \dot{x} \right|^2 \tag{3.45}$$

When the exact end-effector velocity cannot be achieved, the pseudo-inverse will find the joint velocities which achieve the desired end-effector velocity most closely. On the other hand, when the exact end-effector velocity can be achieved, the pseudo-inverse also has the property that it will minimize $\dot{q}^T\dot{q}$ (i.e., the sum of the squares of the joint velocities). In this way, the pseudo-inverse will find the "smallest" joint motion to achieve the end-effector velocity.

The pseudo-inverse can be defined as $J^\dagger = \left(J^T J\right)^{-1} J^T$ (see Section 3.4.3 for alternative formations). Then,

$$\dot{q} = J^\dagger \dot{x} \tag{3.46}$$

In the case of a redundant manipulator, the extra degrees of freedom can be used to achieve additional goals. In other words, the desired end-effector velocity does not fully define the joint velocities, and extra DOFs remain. More formally, we can define the null space projector of J as:

$$\mathcal{N}(J) = I - J^\dagger J \tag{3.47}$$

We can then request a set of velocities, $\dot{q}_N$, which, when projected into the null space, achieve the same end-effector motion:

$$\dot{q} = J^\dagger \dot{x} + \mathcal{N}(J)\, \dot{q}_N \tag{3.48}$$

These $\dot{q}_N$ velocities are referred to as secondary priorities, in that they are satisfied only when they do not conflict with the primary joint velocities, $J^\dagger \dot{x}$. In Section 3.4, these null space dimensions (where observations do not define the kinematic chain) will be proposed using the state transition model.

3.2.7 Gauss-Newton Method

In addition to an analogy with manipulator velocity control, our particle proposal method also has a similar relationship with non-linear optimization techniques. As we will see in Section 3.4, we update each particle by performing a local linear approximation and adjusting the particle's state to be nearer (in the Mahalanobis sense) to the observations. This linear approximation and minimization step is similar to the Gauss-Newton optimization technique [3].

The Gauss-Newton method is an iterative technique which accepts an initial guess and finds a local minimum of the sum of squares of target functions. Suppose we wish to minimize a function, e, where:

$$e(x) = (z - f(x))^T (z - f(x)) \tag{3.49}$$

where f is a vector valued function and z is the observation vector. In our case, f might be the distance from the current state to the observations.

When given some initial guess, x_0, the minimum can be found by:

$$x_i = x_{i-1} + \left(J^T J\right)^{-1} J^T \left(z - f\left(x_{i-1}\right)\right) \tag{3.50}$$

$$x_i = x_{i-1} + J^\dagger \left(z - f\left(x_{i-1}\right)\right) \tag{3.51}$$

where J is the Jacobian of f.

When operating in a non-Euclidean space, we can again modify this standard form using the $\boxplus$ and $\boxminus$ operators:

$$x_i = x_{i-1} \boxplus J^\dagger \left(z \boxminus f\left(x_{i-1}\right)\right) \tag{3.52}$$

Note that in the case of $SE(3)$, for example, the Jacobian would now consider the relationships between partials in the Lie algebra.

3.3 Alternate Solutions

3.3.1 Optimization Only

To motivate our method, we first consider several alternative solutions and discuss their shortcomings. Given that we have relatively accurate observations and a kinematic model, one possible solution is to simply optimize over configurations at each time step. (It is also possible to accumulate data over several time steps and then optimize. However, the system would then be a filter; we explore these ideas in later sections.)

Although this approach is simple, the first problem is that the system may be redundant (i.e., have multiple solutions). An optimization technique, which fails to account for the history of previous observations, may have difficulty converging to the correct solution. The redundancies may result from the kinematic structure — the articulated object may have several configurations that will satisfy the observations. Or, the system may not be observable; that is, the observations do not uniquely define the object's position. This effect may be the result of degeneracies or missing observations. The second problem with optimization is that no estimate of uncertainty is provided and that knowledge is often useful for later processing. Third, joint limits are not inherently supported, and may require additional constraints. Fourth, the optimization is often performed in the state space (and will experience parametrization dependence). Finally, an optimization solution cannot model multi-modal distributions.

For example, consider the task of tracking an excavator at a construction site. Suppose the observations are detections made on an image (Figure 3-16a) and correspond to rays in 3D space (Figure 3-16b). An optimization-based solution would attempt to find a configuration of the excavator which most closely adheres to these observations.

Figure 3-6 shows three example configurations for an excavator. For each configuration, several thousand initial conditions were sampled and a LevenbergMarquardt [3] non-linear optimizer was used to find the nearest configuration. The rays corresponding to the observed pixels are depicted as lines emanating from the camera origin (not shown). For example, Figure 3-6a shows the ground truth position of the excavator with observations; (b) and (c) show the modes of the optimized configurations. (Modes were obtained by performing EM on a Gaussian mixture.)

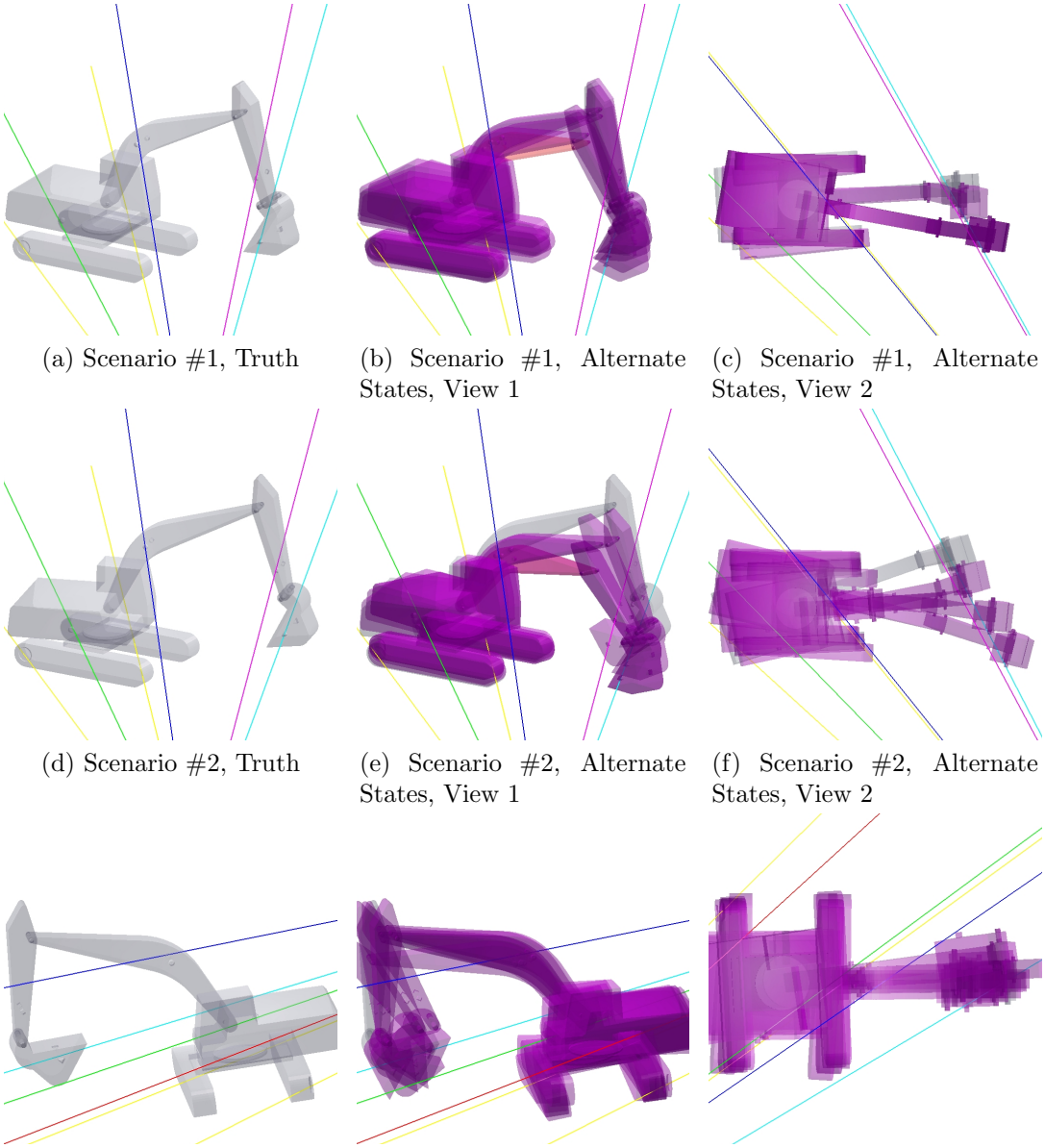
In Figure 3-6(a)-(c), two discrete solutions are recovered. As the excavator's cab rotates clockwise, another configuration also explains the observations. Multiple solutions are found in (d)-(f). Here, the true position of the excavator is nearly parallel to the camera (to see this, note that the gray configuration in (f) is nearly perpendicular to the detection rays). The DOFs of the articulated object provide enough freedom to "slide" the excavator toward and away from the camera while still explaining the observations. In (g)-(i), the excavator's stick (link adjacent to the bucket) is not observed. Consequently, there are two possible configurations for the stick-bucket joint.

Clearly, we wish to incorporate previous observations and attempt to track the excavator. This led us to investigate filtering approaches. Even relying on previous states may be insufficient to avoid all local minima, however. As a result, we eventually focus on particle filtering techniques which can maintain hypotheses at multiple modes and easily converge to a good solution when the configuration can be disambiguated.

3.3.2 Baseline Particle Filter

A filtering technique, which incorporates previous data, can alleviate some of the difficulties with optimization. A filter can use previous states to help estimate the current state. Here, we describe a "baseline" particle filter, which uses the simplest state transition proposal method, as described in Section 3.2.4.1. Unlike the optimization only strategy, the baseline PF can handle joint limits easily via rejection sampling, can handle partial observations, and provides a representation of uncertainty. It can also model multi-modal distributions. Although this solution is better than optimization, it exhibits several limitations discussed here. Later, this baseline will serve as a benchmark for our results.

The first problem with the baseline implementation is that it assumes a reasonably good state transition model. Although our system is provided with a kinematic model, we are not provided with a motion model. Instead, we are relegated to chose from generic motion models, such as zero-velocity or constant-velocity models, which may not predict state transitions well. As noted by [31], this is particularly problematic when the observations are more accurate than the state transition model. As shown in Figure 3-7, when $P(x_k|x_{k-1})$ is more uncertain than $P(z_k|x_k)$, many particles must be sampled from $P(x_k|x_{k-1})$ to adequately sample the narrow $P(z_k|x_k)$. (A better solution, and part of our method, is to incorporate observations during proposal.)



(a) Scenario #1, Truth

(b) Scenario #1, Alternate States, View 1

(c) Scenario #1, Alternate States, View 2

(d) Scenario #2, Truth

(e) Scenario #2, Alternate States, View 1

(f) Scenario #2, Alternate States, View 2

(g) Scenario #3, Truth

(h) Scenario #3, Alternate States, View 1

(i) Scenario #3, Alternate States, View 2

Figure 3-6: Each row shows a sample scenario from different views (columns). The gray rendering is the true pose, shown alone in the first column and in the background in other columns. The magenta renderings show alternative configurations which also explain the observations (rays). In all these examples, multiple modes explain the observations.

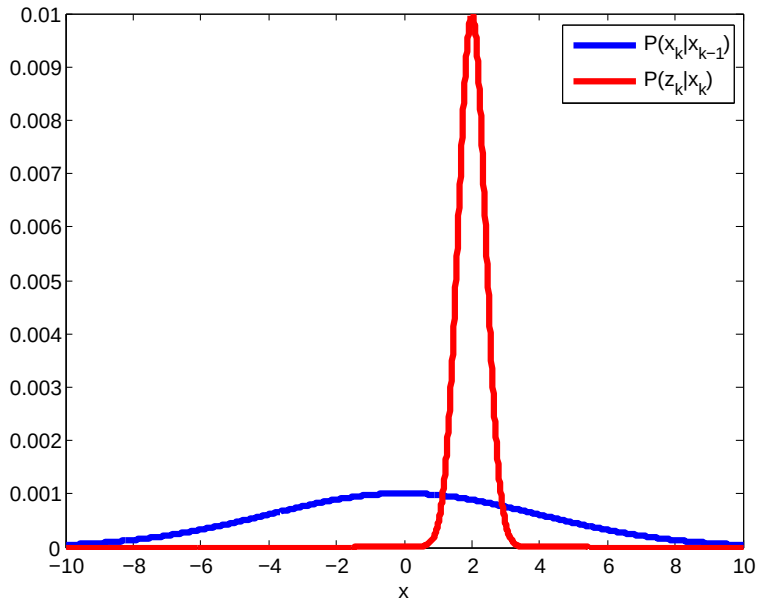


Figure 3-7: In situations where the observation model is more accurate than the state transition model, many samples must be drawn from $P(x_k|x_{k-1})$ to capture the peak of $P(z_k|x_k)$.

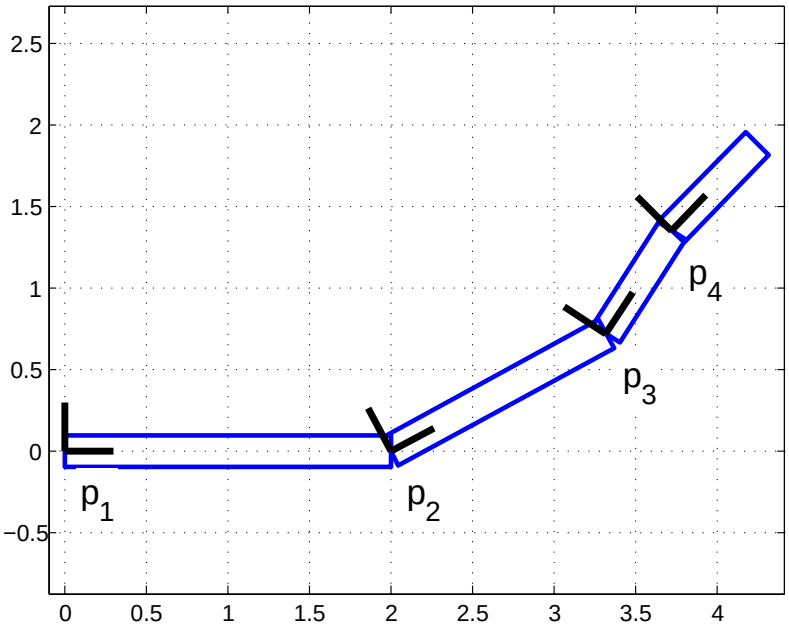


Figure 3-8: An example planar articulated system with four rigid links and three revolute joints.

A second problem with generic motion models is that they often suffer from parametrization dependence. That is, the performance of a filter which uses these generic models will depend on the mathematical expression of the state. To see this, again suppose we wish to track the simple 2D kinematic linkage shown in Figure 3-8 which is moving independently. A natural parametrization for the state space might be:

$$x = \begin{bmatrix} p_1 & \alpha_{12} & \alpha_{23} & \alpha_{34} \end{bmatrix}^T \tag{3.53}$$

where $p_1 \in SE(2)$ is the pose of the base link and $\alpha_{ij} \in \mathbb{R}$ are the joint angles between links i and j. Further, the observations might be the $p_i \in SE(2)$ poses of each link:

$$z = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \end{bmatrix}^T \tag{3.54}$$

Again following Section 3.2.4.1, we propose from $P(x_k|x_{k-1}^{(i)})$. If we assume a zero-velocity model, we add Gaussian noise (AGN) to propagate the state, i.e., $x \sim N(\mu_x, \Sigma_x)$. An example of proposed particles is shown in Figure 3-9. We might also assume that our observations are corrupted by AGN, i.e., $z \sim N(f(\mu_x), \Sigma_z)$, where f is the forward kinematics. The isocontours of such a distribution are also shown in Figure 3-9. It is clear that they are dissimilar — in the case of the system model, the distributions become increasingly "banana-like" as we proceed further out from the base link.

In the context of a particle filter, this dissimilarity means that many samples proposed from $P(x_k|x_{k-1}^{(i)})$ would have low weights due to the mismatch with $P(z_k|x_k)$ at the distal links. Wasted computation and degraded accuracy result.

Observe further that a slight change to the parametrization also affects particle proposal. To see this, consider an alternative formulation:

$$\hat{x} = \begin{bmatrix} \alpha_{21} & p_2 & \alpha_{23} & \alpha_{34} \end{bmatrix}^T$$

Here, the base link is taken as p_2, and AGN on this state produces the plot shown in Figure 3-10. Comparing Figure 3-9 and Figure 3-10 highlights that particle proposal in the baseline method depends on the state parametrization. To mitigate this problem, we will later see that we propose noise in the observation space, rather than the state space, because it is fixed in our application.

3.3.3 Unscented Kalman Filter

The UKF is another possible technique to track an articulated object. As in Section 3.3.2, the state might consist of a base pose and the joint angles. The UKF calculates sigma points around the current state, predicts motion according to the state transition model, and updates the estimate given the new observations (see Algorithm 3.2.3). The UKF models the uncertainty as a Gaussian, so the current state estimate is always unimodal and associated with a covariance matrix. The UKF also handles partial observations and, as noted in Section 3.2.5, an additional optimization step can be used to handle joint limit verification.

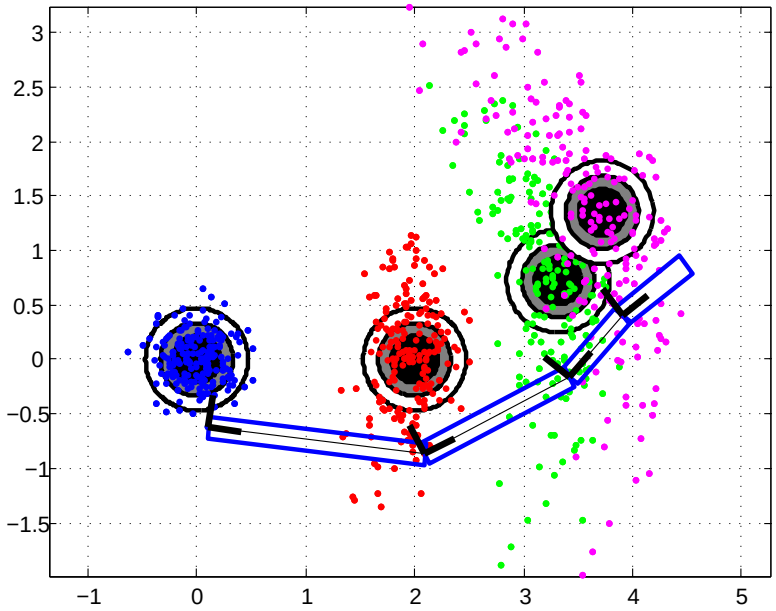


Figure 3-9: The dots show the link positions of the particles proposed from $P(x_k|x_{k-1}^{(i)})$ for links 1-4 (respectively blue, red, green, and magenta). The underlying contours (black) illustrate the Gaussian $P(z_k|x_k)$. Notice that for link 4, for example, many particles are unlikely according to the observation model. An example configuration for the linkage is also shown.

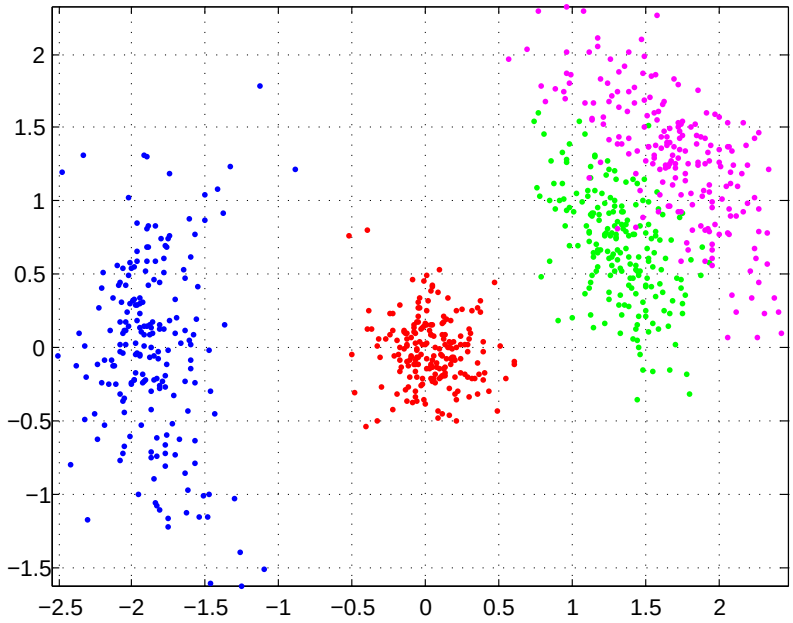


Figure 3-10: In the baseline method, altering the state parametrization also affects the proposed particles (c.f., Figure 3-9).

Although the UKF may be a viable filtering technique for some objects, in general, the distribution of configurations for an articulated object will be multi-modal and cannot be captured by a single Gaussian. As discussed in Section 3.3.1, the excavator may have many possible configurations which explain the observations. As shown in Section 3.5.1.4, the ambiguity due to multi-modal distributions is difficult to resolve when the object starts in a degenerate configuration. In other words, prior information may not be sufficient. Since the UKF cannot handle such multi-modal distributions, we use a particle filter.

Like the baseline particle filter, the UKF also exhibits a state parametrization dependence. This occurs because the sigma point calculation (c.f., Algorithm 3.2.3) adds small perturbations in the state space.

$$\mathcal{Y}_k = \begin{cases} \overline{x_k} & i = 0 \\ \overline{x_k} \boxplus \langle \overline{\Sigma_{x_k}} \rangle_i & 1 \leq i \leq n \\ \overline{x_k} \boxplus -\langle \overline{\Sigma_{x_k}} \rangle_{i-n} & n+1 \leq i \leq 2n \end{cases} \tag{3.55}$$

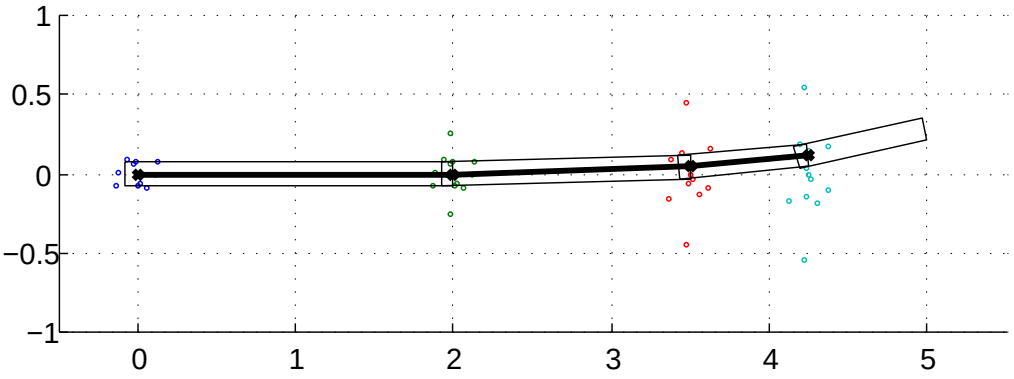
To see this, again consider the four-link kinematic chain in Figure 3-8. A state parametrization of:

$$x = \begin{bmatrix} p_1 & \alpha_{12} & \alpha_{23} & \alpha_{34} \end{bmatrix}^T \tag{3.56}$$

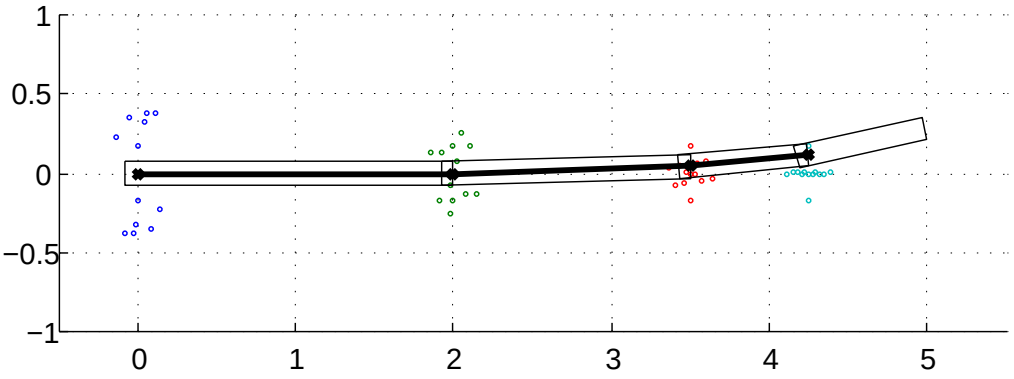
yields sigma points as shown in Figure 3-11a. On the other hand, a state parametrization of:

$$x = \begin{bmatrix} p_4 & \alpha_{12} & \alpha_{23} & \alpha_{34} \end{bmatrix}^T \tag{3.57}$$

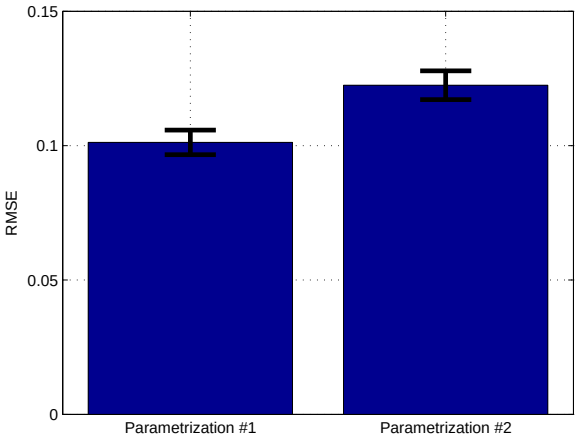
yields the different sigma points shown in Figure 3-11b. One hundred iterations of a 50-frame simulation reveal that this difference affects the RMSE performance of the two parametrizations, as shown in Figure 3-11c. As mentioned for the baseline particle filter, we propose noise in the observation space to address this dependence. It is important to note that this effect can also be minimized by using a reliable state transition model. Again, however, we do not assume a reliable state transition model is available.



(a) Sigma points for parametrization #1 with base pose as p_1, the left-most link (blue).



(b) Sigma points for parametrization #2 with base pose as p_4, the right-most link (cyan).



(c)

Figure 3-11: Different state parametrizations result in different sigma points for the UKF (a), (b). The sigma points for links 1-4 are shown here in blue, green, red, and cyan, respectively. In this situation, different tracking error (c) results. Error bars indicate one standard deviation.

3.4 Our Method

The previous section highlighted some of the shortcomings of alternative methods to track an articulated object. In this section, we describe an algorithm which addresses these issues. Because we need to track a state with a multi-modal distribution, we use a particle filter. Because our state transition model is inaccurate, relative to our observations, we need to incorporate the observations during particle proposal. Finally, because we wish to minimize the effects of state parametrization, we perform operations in the fixed observation space where possible, using the state space only when unavoidable.

To simplify the discussion, we will first develop our ideas for planar articulated objects. We will then expand the technique to 3D articulated objects, building on techniques from the calibration chapter. The algorithm uses a particle filter (Algorithm 3.2.1), and the following sections focus on the particle proposal function, `Proposal`. Our particle proposal approximates the OIF (c.f., Section 3.2.4.2), incorporating both the observations and the previous state.

3.4.1 Planar Articulated Object Tracking

During particle proposal, the algorithm considers the valid configurations of the articulated object as a manifold $\mathcal{M}$ in a higher-dimensional space. We chose this high dimensional space to be the observation space so that proposed particles correspond to observation noise and because this space is fixed in our application. However, other choices are possible (e.g., link poses could be used for noise proposal regardless of the observation or state representation). Regardless, the manifold is defined by the forward kinematics and observation function, $f(x)$, which converts from an $M \times 1$ state vector, x, to a point in the higher dimensional space. A complete observation z is an N dimensional point in the observation space. (We assume our system is observable, i.e., $N \geq M$.)

The complete algorithm is presented in Function 3.4.1. Generally, it proceeds by:

1. Using available observations, z_k, to find the nearest valid configuration/state (i.e., the closest point on $\mathcal{M}$).
2. Approximating the OIF (Equation 3.16) using a set $\mathcal{X}^{(i)}$ of discrete samples, generated by:
   (a) Adding noise in the observation (not state) space, then using a first-order Taylor approximation and projection to map the noise to the state space,
   (b) Adding noise in the state space only when the Jacobian is singular, and
   (c) Rejection sampling to satisfy joint limits.

As a result of (1), we can exploit observations during the proposal stage without requiring an inverse kinematic/observation function. As we will see, we can extract

information from observations which are either redundant ($\dim(z_k) > M$) or incomplete ($\dim(z_k) < M$). As a result of (2a), the particle proposals are less dependent on the state parametrization, because particle perturbations are created in the fixed observation space. Attribute (2b) handles degenerate observations by relying on the state transition model. In (2c), we ensure that joint limits are satisfied. Because rejection sampling can be expensive, we sample directly from the discrete approximation $\mathcal{X}^{(i)}$ when choosing $x_k^{(i)}$ (rather than re-approximating with a Gaussian, as in [30]).

As mentioned, the algorithm is not provided with a state transition model, we select a model which will generalize to many different kinematic chains. In all our examples, we found a zero-velocity model sufficient. That is, we use the state transition model $P(x_k|x_{k-1}) \sim N(x_{k-1}, \Sigma_x)$. If another transition model were more appropriate, e.g., a constant-velocity model, the following techniques could be easily modified by adding the constant velocity, in addition to diffusion noise, in the null space.

3.4.1.1 Incorporating Observations During Particle Proposal

The algorithm begins by considering each particle $x_{k-1}^{(i)}$ from the previous time step. For notational simplicity, let $x = x_{k-1}^{(i)}$. We wish to update this particle using whatever observations are available. Thus, we find a new point m_k which is both near the previous particle and near the observations:

$$m_k = x + \widehat{dx} \tag{3.58}$$

where $\widehat{dx}$ is a small change in the state space. We wish to find m_k by minimizing the Mahalanobis distance between the observation and the configuration manifold. Further, the observations, z_k, may be incomplete. Let Q be an $N \times N$ diagonal matrix whose diagonal entries are one if the observation is made at time k and zero otherwise. Thus, Q selects the valid observations.

$$\widehat{dx} = \underset{dx}{\operatorname{argmin}} \left[Qz_k - Qf(x + dx)\right]^T \Sigma_{obs}^{-1} \left[Qz_k - Qf(x + dx)\right] \tag{3.59}$$

where Σ_{obs} is the covariance of the observations. Let $L^T L = \Sigma_{obs}^{-1}$ be the Cholesky factorization of Σ_{obs}^{-1}. Then

$$\begin{aligned} \widehat{dx} &= \underset{dx}{\operatorname{argmin}} \left[Qz_k - Qf(x + dx)\right]^T L^T L \left[Qz_k - Qf(x + dx)\right] \\ &= \underset{dx}{\operatorname{argmin}} \left|LQz_k - LQf(x + dx)\right|^2 \end{aligned} \tag{3.60}$$

The algorithm then finds a nearer point on the manifold by projecting onto a tangent plane (Figure 3-12). We assume that $\mathcal{M}$ is well-approximated by a first-order Taylor series in a neighborhood corresponding to the observation noise. That is, we assume that:

$$f(x + dx) = f(x) + J_x \cdot dx \tag{3.61}$$

where J_x is the Jacobian of f at x. We also note that $z_k = f(x) + dz$, where dz is the vector between the previous particle and the current observation.

$$\begin{aligned}\widehat{dx} &= \underset{dx}{\operatorname{argmin}} \left| LQf(x) + LQdz - LQf(x) - LQJ_x\ dx \right|^2 \\ &= \underset{dx}{\operatorname{argmin}} \left| LQdz - LQ\ J_x\ dx \right|^2 \end{aligned} \tag{3.62}$$

The pseudo-inverse, $(\cdot)^\dagger$, solves this problem:

$$\widehat{dx} = (LQJ_x)^\dagger\ LQ\ dz \tag{3.63}$$

Substituting back into Equation 3.58:

$$m_k = x + (LQJ_x)^\dagger\ LQ\ dz \tag{3.64}$$

$$= x_{k-1}^{(i)} + (LQJ_x)^\dagger\ LQ\ \left(z_k - f\left(x_{k-1}^{(i)} \right) \right) \tag{3.65}$$

Thus, m_k is a configuration of the articulated object which is near the most recent observations (at least on the local tangent plane). The corresponding line 5 in Function 3.4.1 takes a single Gauss-Newton optimization step (see Section 3.2.7) toward the observations. This process is illustrated in Figure 3-15; in (a), an observation (dotted line) is made and there are four possible explanations for this observation (intersection of $\mathcal{M}$ and z_k). Each particle is optimized (with a single Gauss-Newton step) towards a local minimum with the new observations, (b). In this way, the particles are shifted toward higher probability regions.

It is important to note that the matrix product (LQJ_x) may be singular for several reasons: (1) it is always singular; (2) it is singular due to a specific configuration; or (3) it is singular due to missing observations. We assume that (1) is not the case, as the system would be unobservable and unable to be reliably tracked. If (2) or (3) is the case, then there is a non-trivial null space. In that situation, there are dimensions of $x_{k-1}^{(i)}$ which are unaltered in m_k. On the other hand, the dimensions that lie in the range are updated and "pulled" as close as possible to the observation. This has the pleasant result that all information is extracted from the observations, while the unobserved dimensions remain centered at $x_{k-1}^{(i)}$. This performance is consistent with our zero-velocity state transition model; alternative models could update $x_{k-1}^{(i)}$ by manipulating the null space. When z_k has redundant observations and over-constrains $x_{k-1}^{(i)}$, Equation 3.65 computes $\widehat{dx}$ by projecting onto J_x.

3.4.1.2 Diffusion During Particle Proposal

We discretely approximate the OIF by randomly selecting points about m_k and then weighting by Equation 3.37. The algorithm creates a set $\mathcal{X}^{(i)} = \left\{\ \mathcal{X}_1\ \ \cdots\ \ \mathcal{X}_P\ \right\}$ of points centered around m_k (lines 6-14). These P points discretely approximate the OIF and are sampled in line 21 to select $x_k^{(i)}$, the next generation particle.

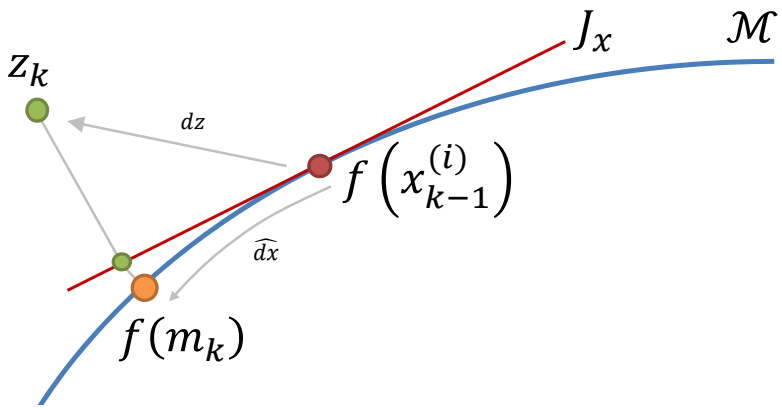


Figure 3-12: The method updates the previous particle, $x_{k-1}^{(i)}$, with the observations, z_k, using the Taylor approximation and subsequent projection.

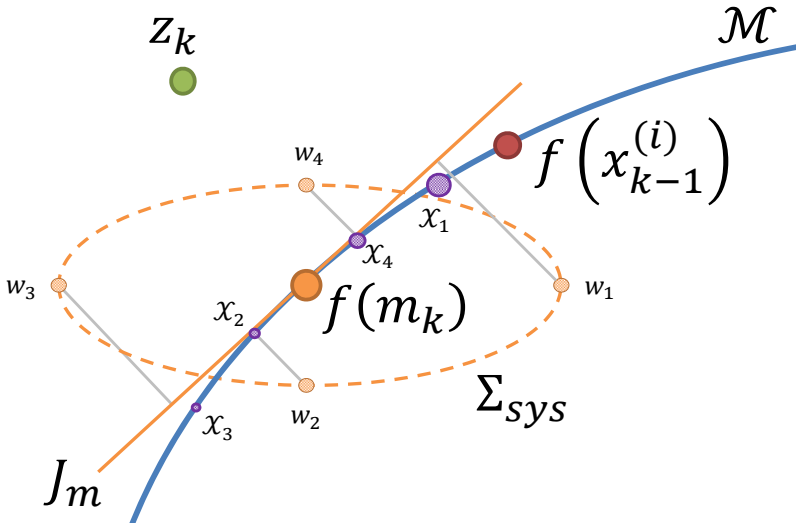


Figure 3-13: A Taylor approximation and a projection relate noise in observation space to noise in state space.

Our algorithm proposes noise in the observation space, rather than the state space, because it remains fixed in our application. Along some dimensions, however, it may not be possible to convert uncertainty in the observation space to uncertainty in the state space. This can occur when the Jacobian is singular due to a specific joint configuration or missing observations. When this occurs, the algorithm proposes using state uncertainty in this null space.

It is interesting to note that the observation space can be thought of as an over-parametrization of the state space. To see this, consider a noise-free set of observations. Assuming $N > M$, then the observations use more parameters to describe the state than are strictly necessary. An analogous situation was faced in Section 2.5, where an 8-element DQ was used to represent a 6-DOF pose. In that situation, noise was expressed in the Lie algebra, a locally tangent space. Here, we project the noise into a local tangent approximation.

As shown in Figure 3-13, we sample AGN perturbations from $w_j \sim N(0, \Sigma_{sys})$. Σ_{sys} is an $M \times M$ matrix which expresses the uncertainty of the state in the observation space. In our experiments (Section 3.5) we estimate Σ_{sys} using *ad hoc* methods, but it could also be determined by passing sigma points through the forward kinematics [10]. Again, using a Taylor approximation:

$$\underbrace{\mathcal{X}_j}_{M\times 1} = \underbrace{m_k}_{M\times 1} + \underbrace{J_m^\dagger}_{M\times N}\underbrace{w_j}_{N\times 1} \tag{3.66}$$

The equation uses the pseudo-inverse of the Jacobian, J_m, evaluated at m_k, to convert uncertainty in the observation space to uncertainty in the state space. Then w_j is simply projected onto $\mathcal{M}$ to produce the points $\mathcal{X}_j$.

The conditions under which J_m is singular must again be considered. Either (1) J_m is always singular or (2) J_m is singular due to a specific configuration. Since the system is observable, (1) is not the case. If (2) is the case, then there exists a non-trivial null space and $\mathcal{X}_j$ will equal m_k along the null dimensions of J_m. This is a problem because no noise has been added along the null dimensions. In other words, w_j does not define a unique change in m_k and the pseudo-inverse extinguishes perturbations along those dimensions.

To see this, consider again the four-link chain shown in Figure 3-14a with all links aligned along the horizontal axis. If a bearing-only sensor is positioned at $(0, -1)$ looking along the horizontal axis, the Jacobian is degenerate; the entire linkage can slide along the horizontal axis without changing the bearing observations. Using Equation 3.66, the particles shown in Figure 3-14b are generated. Note that along the (horizontal) null dimension, there is no perturbation in the particles for the left-most link. Increasingly to the right, there is some displacement in the horizontal dimension due only to motion in the constraint manifold. Thus, Equation 3.66 does not meaningfully distribute the particles, which is undesirable.

The solution is to use the state space to propose points in the null space of J_m.

Revising Equation 3.66:

$$\mathcal{X}_j = m_k + J_m^\dagger \; w_j + \underbrace{\mathcal{N}\left(J_m\right)}_{M \times M} \underbrace{v_j}_{M \times 1} \tag{3.67}$$

where $v_j \sim N(0, \Sigma_x)$ and Σ_x is the $M \times M$ covariance matrix for the state (as used by the baseline method). The $\mathcal{N}\left(J_m\right) = I - J_m^\dagger \; J_m$ matrix is the null space projector matrix. When using Equation 3.67, particles are well distributed even when singularities exist in the Jacobian (see Figure 3-14c). Notice that because we make use of the state space in these situations, our method is not entirely independent of the state parametrization. However, it is used sparingly, relative to other methods.

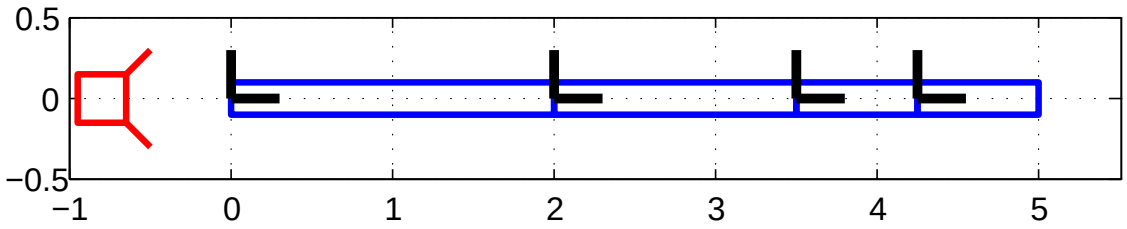
Our method draws samples from the high-dimensional observation space and projects them onto a (typically, lower-dimensional) plane tangent to the configuration manifold. The samples are then transferred onto the manifold itself according to the Taylor approximation. An alternative could be to sample directly on the tangent plane, for example from a distribution suitably constructed from the sigma points [40] of Σ_{sys} (Figure 3-13).

3.4.1.3 Rejection Sampling

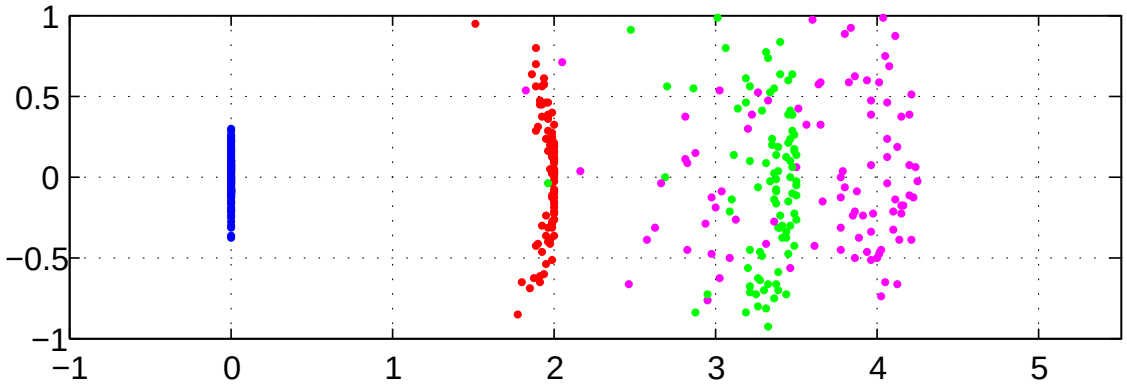
Although the samples produced by Equation 3.67 are guaranteed to satisfy the kinematic equations (i.e., lie in the manifold), they may not satisfy the joint limits. As a result, Function 3.4.1 iterates Equation 3.67 in line 8-line 14, generating a sample and evaluating it against the joint limits. If the particle satisfies the joint limits, it is added to $\mathcal{X}^{(i)}$; otherwise, it is discarded. This rejection sampling [46] is appropriate only for joint limits which define a volume in the state space. Joint limits defining equality constraints can be incorporated into the kinematic constraints.

3.4.1.4 OIF Sampling

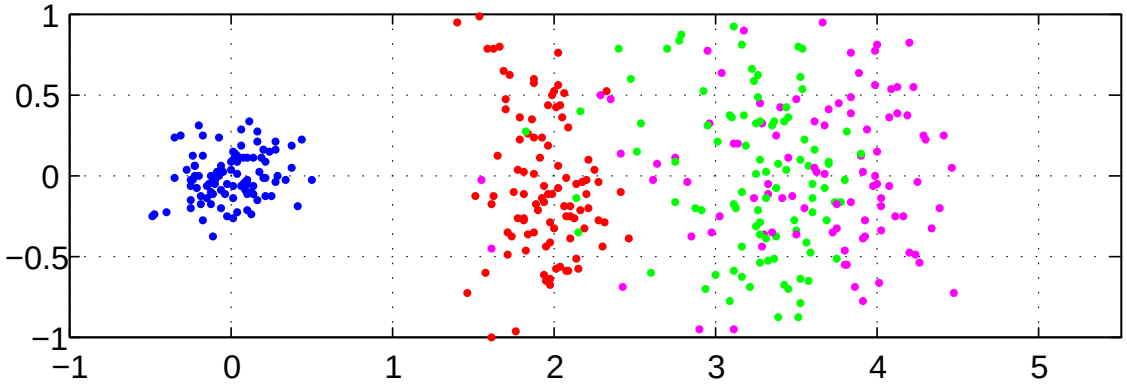
Finally, the OIF is evaluated using Equation 3.16 and a discrete version of Equation 3.38. Lines 15-20 calculate likelihood of each $\mathcal{X}_j$. For each particle, $\mathcal{X}^{(i)}$ forms a discrete approximation for the OIF distribution $P\left(x_k \left| x_{k-1}^{(i)}, z_k \right.\right)$ and their likelihoods can be summed to approximate $P\left(z_k | x_{k-1}^{(i)}\right)$ (c.f., Equation 3.38). Figure 3-15d illustrates this — each particle has a discrete approximation to the OIF. Since all $\mathcal{X}_j$'s satisfy the joint limits and kinematic constraints, we can select $x_k^{(i)}$ by drawing from $\mathcal{X}^{(i)}$ according to the OIF probabilities.



(a) A bearing-only sensor (red) observes the four-link kinematic chain. In this configuration, the system is singular because the chain can be moved along the horizontal axis without affecting the observations.



(b) Particles proposed with Equation 3.66 have little motion along the singular dimension.



(c) Particles proposed with Equation 3.67 use the state space to propose only along the singular dimension, reducing effects of the state parametrization.

Figure 3-14: Proposing particles at singularities

Function 3.4.1: Proposal2D(x_{k-1}, w_{k-1}, R_k, z_k, Σ_{z_k}, Q)

inputs : x_{k-1}, w_{k-1} previous N_s particles and weights
R_k state covariance $M \times M$ matrix in state space
z_k the $N \times 1$ observations at time k
Σ_{z_k} covariance matrix for the z_k observations
Q observation selection $N \times N$ matrix specifying valid observations

params : f forward kinematic/observation function
P number of discrete elements used to approximate OIF
Σ_{sys} state covariance $N \times N$ matrix in observation space

outputs: $\{x_k, \alpha\}$ updated N_s particles and weight scales

$L \leftarrow$ `Cholesky` $(\Sigma_{z_k}^{-1})$ // $L^T L = \Sigma_{z_k}^{-1}$
$x_k \leftarrow \emptyset$; $\alpha \leftarrow \emptyset$
for $x_{k-1}^{(i)} \in x_{k-1}$ **do**
  $J_x \leftarrow$ `Jacobian` $(f, x_{k-1}^{(i)})$
  $m_k \leftarrow x_{k-1}^{(i)} + (LQJ_x)^\dagger \; LQ \; \left(z_k - f\left(x_{k-1}^{(i)}\right)\right)$
  `// create` N_s `samples via rejection sampling where`
  `//` $\mathcal{X}^{(i)}$ `is discretely distributed according to` $P(x|x_{k-1}^{(i)}, z_k)$
  $J_m \leftarrow$ `Jacobian` (f, m_k)
  $\mathcal{N}(J_m) \leftarrow I - J_m^\dagger \; J_m$ `// null space projector matrix`
  $p \leftarrow P$; $\mathcal{X}^{(i)} \leftarrow \emptyset$
  while $p > 0$ **do**
    $w_j \sim N(0, \Sigma_{sys})$ `// propose state noise in observation space`
    $v_j \sim N(0, R_k)$ `// propose state noise in state space`
    $\mathcal{X}_j \leftarrow m_k + J_m^\dagger \; w_j + \mathcal{N}(J_m) \, v_j$
    if `JointLimitsSatisified(`$\mathcal{X}_j$`)` **then**
      $\mathcal{X}^{(i)} \leftarrow \{\mathcal{X}^{(i)}, \mathcal{X}_j\}$; $p \leftarrow p - 1$
  `// determine OIF probability of each candidate particle in` $\mathcal{X}^{(i)}$
  for $\mathcal{X}_j \in \mathcal{X}^{(i)}$ **do**
    $P(z_k|\mathcal{X}_j) \leftarrow N(z_k; f(\mathcal{X}_j), \Sigma_{z_k})$
    $P\left(\mathcal{X}_j|x_{k-1}^{(i)}\right) \leftarrow N\left(\mathcal{X}_j; x_{k-1}^{(i)}, \Sigma_{sys}\right)$
  $P\left(z_k|x_{k-1}^{(i)}\right) \leftarrow \sum_{\mathcal{X}_j \in \mathcal{X}^{(i)}} P(z_k|\mathcal{X}_j) \cdot P\left(\mathcal{X}_j|x_{k-1}^{(i)}\right)$
  for $\mathcal{X}_j \in \mathcal{X}^{(i)}$ **do**
    $P\left(\mathcal{X}_j \left| x_{k-1}^{(i)}, z_k\right.\right) \leftarrow \frac{P(z_k|\mathcal{X}_j) \cdot P\left(\mathcal{X}_j|x_{k-1}^{(i)}\right)}{P\left(z_k|x_{k-1}^{(i)}\right)}$
  `// sample from` $\mathcal{X}^{(i)}$ `according to OIF probabilities`
  $x_k \leftarrow \{x_k,$ `RandomSample` $(\mathcal{X}^{(i)}, P(\mathcal{X}^{(i)}|x_{k-1}^{(i)}, z_k))\}$
  $\alpha \leftarrow \{\alpha, P\left(z_k|x_{k-1}^{(i)}\right)\}$ `// append weight to` α
return $\{x_k, \alpha\}$

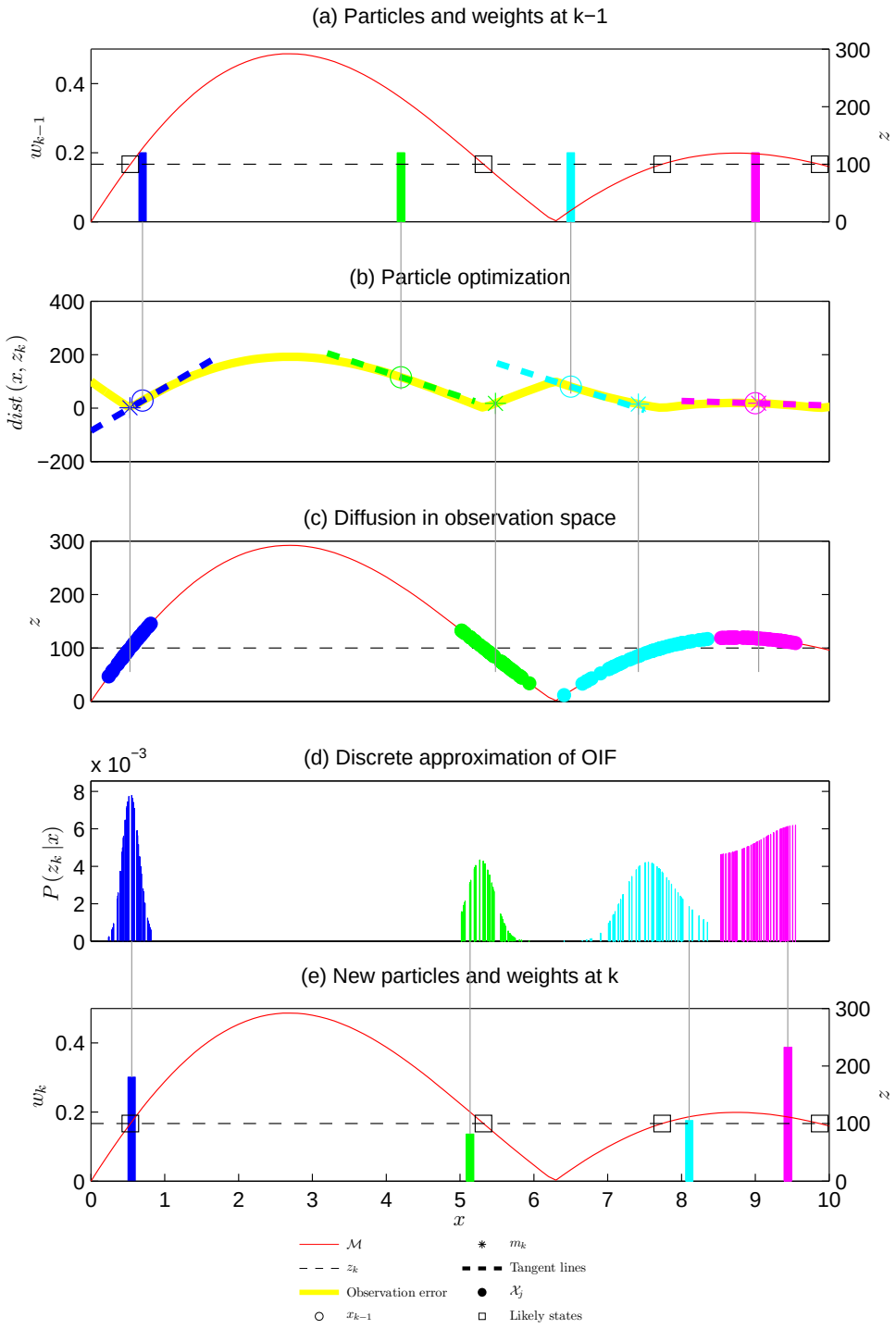


Figure 3-15: An observation (dotted line) is obtained in (a); intersections with $\mathcal{M}$ (red) are likely configurations (black squares). The particles (x_{k-1}) are then optimized (b) toward the likely configurations (m_k, color asterisks). Random perturbations are added in the observation space (c). For each particle, $\mathcal{X}^{(i)}$ in (d) approximates the OIF. $\mathcal{X}^{(i)}$ is then sampled to select the next generation of particles (e).

3.4.2 3D Articulated Object Tracking

The planar articulated object tracking extends naturally to the 3D case, where the base pose is in $SE(3)$ instead of $SE(2)$. We again use a particle filter, optimize toward the observations, and propose noise using the same process. The 3D case does, however, require several modifications. First, for convenience, we standardize our observations to be 6-DOF poses with a possibly singular precision matrix. Second, we re-derive our tangent approximation using the Lie algebra, and show that our conclusions for the Euclidean case are still applicable. Third, we allow the particle optimization to execute more than one cycle, allowing for more significant movement of the object. This proved advantageous for articulated objects which moved significantly between frames.

As in the planar case, our algorithm was also developed with a few assumptions. First, we assume that the observation model is more accurate than the state transition model. Although we found this to be true in all of our examples, if it were not the case, the baseline particle filter might be a better choice. Second, we assume that the observations/kinematics form a redundant system and can be best represented by a multi-modal distribution; if not, the UKF might be a better solution. Third, we assume that the system is observable over time, although not necessarily at each time step. Finally, we assume that the observations are associated with points on the object (i.e., data association is provided).

3.4.2.1 6-DOF Observations

For planar articulated object tracking (see Section 3.4.1), the algorithm required only that the observations be sufficient to make the system observable. Although this requirement is sufficient, in practice we found that additional requirements made the system more flexible. Specifically, we desired an interface such that many different kinds of detectors could provide input without requiring code modification to the algorithm.

For example, an image-based detector would provide observations in the form of $\{x, y\}$ pixel locations. An ICP-based fitting algorithm operating on point cloud data might provide a complete 6-DOF pose. Or, an optimization routine based on a planar LIDAR might provide a 3-DOF pose. In principle, the algorithm can be configured to accept any such type of observation, but might require re-coding or re-compiling the code each time.

Instead, we chose to standardize our observation input to always be 6-DOF poses. To allow for detectors which cannot provide complete information, we associate each pose with a precision matrix. Importantly, we allow this precision matrix to be singular to represent unobserved DOFs. For example, an image-based detector would observe only two DOFs. The remaining four DOFs are unobserved and would be singular in the associated precision matrix. Such an example is shown in Figure 3-16. Here, a TLD-based detector [41] tracks six features in video of an excavator (Figure 3-16a). Each of these detections defines a ray in 3D space (Figure 3-16b).

The precision matrix is then defined to be singular along that ray. Figure 3-16c shows example poses drawn using the resulting precision matrix. Any pose along the ray, regardless of orientation, is considered to explain the pixel detection.

Internally, we use a precision matrix to represent uncertainty rather than its inverse, the more common covariance matrix. This is because a covariance matrix cannot easily encode an infinite covariance, but a precision matrix can encode a zero (singular) precision. In the following discussion, we may write Σ^{-1} to mean the precision matrix, with the understanding that we are not performing the inversion explicitly.

Although this encoding for the observations has advantages, it also imposes limitations which are not, in general, required. First, this encoding may require additional computation. Rather than computing two DOFs for an image detection, we are now processing a full six DOFs (four of which are zero). Additionally, we now maintain a 6×6 precision matrix for each observation, rather than a 2×2. Second, the technique is limited to observations which can be expressed with Gaussian uncertainty on a 6-DOF pose. This encoding would be insufficient to represent a multi-modal uncertainty. For example, the position of a wheel with spokes, symmetric every 30 degrees, could not be represented with this encoding.

3.4.2.2 Tangent Approximation in the Manifold

For planar articulated objects, we performed an optimization step followed by the addition of noise in the observation space. The same idea will be applied here to the 3D case. The exception is that we derive the Taylor approximation while accounting for the fact that the observation space is no longer $\mathbb{R}^N$ — instead it is a compounded $SE(3)$ space. Although any representation of $SE(3)$ is suitable for the derivation, we will again make use of dual quaternions, $\mathbb{H}$, and their Lie algebra, $\mathfrak{h}$, using the notation from the calibration work in Section 2.5.2. Where the observations are $N \times 1$, $\mathcal{M} \subseteq \mathbb{H}^{N/8}$ (a DQ has 8 elements).

Additionally, the state is itself not $\mathbb{R}^M$; rather, we treat the state as $\mathbb{H} \times \mathbb{R}^{M-8}$ encoding a 6-DOF base pose and the joint angles. For example, if $b \in \mathbb{H}$ is the base pose and $q_i \in \mathbb{R}$ are the joint angles, then the state is:

$$x = \begin{bmatrix} b & q_0 & \cdots & q_{M-8} \end{bmatrix} \tag{3.68}$$

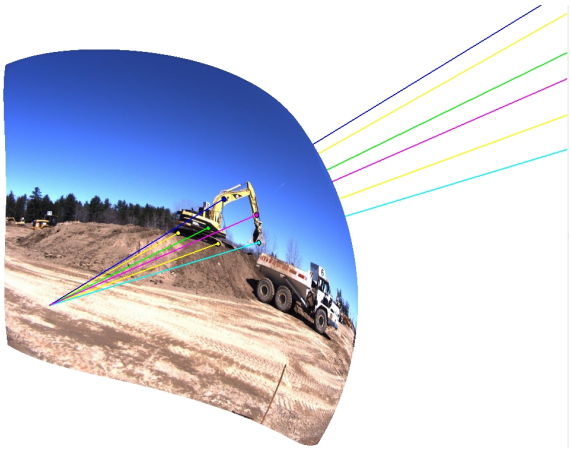
A small perturbation in the state can be represented by a Lie algebraic velocity, $c \in \mathfrak{h}$, and joint velocities, $r_i \in \mathbb{R}$:

$$\delta = \begin{bmatrix} c & r_0 & \cdots & r_{M-8} \end{bmatrix} \tag{3.69}$$

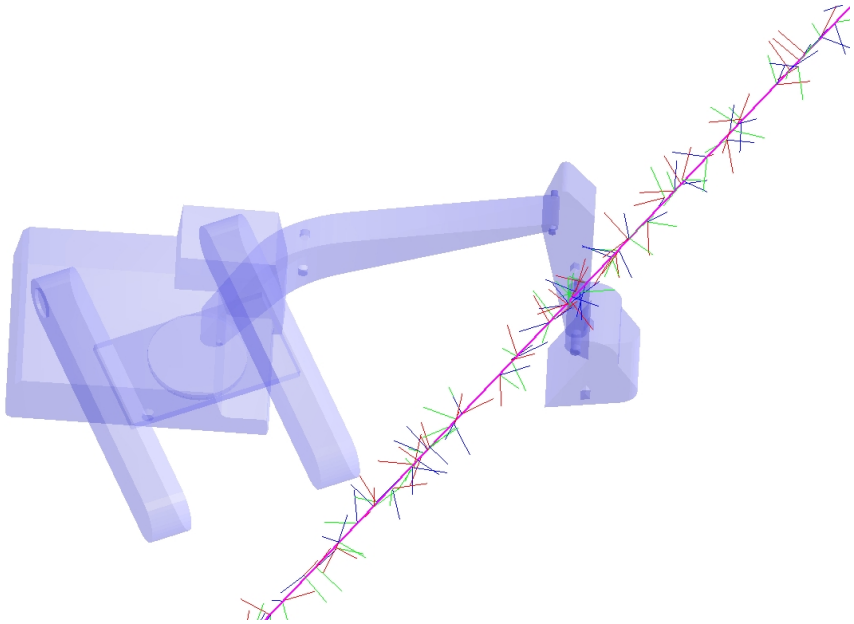
where δ is $(\mathfrak{M} = 6 + M - 8) \times 1$. We then define an additional operator $\oplus$ such that:

$$x \oplus \delta = \begin{bmatrix} b \boxplus c & q_0 + r_0 & \cdots & q_{M-8} + r_{M-8} \end{bmatrix} \tag{3.70}$$

(a) Observations are 2D pixel locations on the articulated excavator.



(b) Each pixel location defines a ray observation in 3D space, emanating from the camera origin.



(c) A precision matrix is defined such that samples drawn from the associated Gaussian distribution lie along the ray. Here, we show samples for the detection on the excavator's stick (the magenta rays). The "barbed" lines along the ray are axes corresponding to sampled poses.

Figure 3-16: Examples of singular observations along rays associated with image observations are shown.

In this way, the pose of the articulated object is updated by compounding transforms and the joint angles are added in a Euclidean fashion.

Proceeding as in the planar case, Section 3.4.1.1, we wish to find a new point m_k which is near the new observations:

$$m_k = x \oplus \widehat{dx} \tag{3.71}$$

As before, we desire to find dx:

$$\widehat{dx} = \underset{dx}{\operatorname{argmin}} \left[Qz_k \boxminus Qf\left(x \oplus dx\right)\right]^T \Sigma_{obs}^{-1} \left[Qz_k \boxminus Qf\left(x \oplus dx\right)\right] \tag{3.72}$$

where Σ_{obs}^{-1} is a possibly singular precision matrix associated with the observations. Note that this precision matrix expresses a Gaussian uncertainty in the Lie algebra. Simplifying again,

$$\widehat{dx} = \underset{dx}{\operatorname{argmin}} |LQz_k \boxminus LQf\left(x \oplus dx\right)|^2 \tag{3.73}$$

We can again make the same Taylor approximation, $f\left(x \oplus dx\right) = f\left(x\right) \boxplus J_x \cdot dx$, and note that $z_k = f\left(x\right) \boxplus dz$. Until now, there has been no change in the derivation, except for notation. However, when we substitute:

$$\widehat{dx} = \underset{dx}{\operatorname{argmin}} \left| \left(\underbrace{LQf\left(x\right)}_{N\times 1} \boxplus \underbrace{LQdz}_{\mathfrak{N}\times 1} \right) \boxminus \left(LQf\left(x\right) \boxplus \underbrace{LQJ_x\ dx}_{\mathfrak{N}\times 1} \right) \right|^2 \tag{3.74}$$

we can no longer simplify, since operations in $SE(3)$ do not commute. ($\mathfrak{N} = 6N/8$, because elements in $\mathbb{H}$ and $\mathfrak{h}$ are 8 and 6 elements, respectively.) That is $(b \boxplus c) \boxminus (b \boxplus d) \neq c - d$. However, as shown in Section B.1, it is still the case that:

$$\widehat{dx} = \underset{dx}{\operatorname{argmin}} \left|LQdz - LQJ_x\ dx\right|^2 \tag{3.75}$$

Therefore, we can proceed analogously to the planar case, except with the manifold operators:

$$m_k = x_{k-1}^{(i)} \oplus \left(LQJ_x\right)^{\dagger}\ LQ\ \left(z_k \boxminus f\left(x_{k-1}^{(i)}\right)\right) \tag{3.76}$$

3.4.2.3 Revisiting Observation Optimization

In the PR2 "pipe swing" example (see Section 3.5.4), we found that the motion of the pipe was so significant between frames that our linear Taylor approximation occasionally failed. As the pipe reached its most vertical point, it started to fall. Within 2-3 frames, it had rotated more than 90 degrees. The planar algorithm, suitably adapted to 3D, failed because the Taylor approximation was poor. In other

words, the single Gauss-Newton optimization step was insufficient to bring particles into a high probability region.

Our solution is to perform additional, successive Taylor approximations. That is, the method performs additional steps of Gauss-Newton optimization. Unlike many optimization techniques, however, the method does not minimize some explicit closeness measure. Instead, we optimize until some high probability particles are generated. In practice this often occurs in one step. Occasionally, such as during the high-acceleration frames of the PR2 examples, as many as three steps are required. The algorithm is detailed in Function 3.4.2.

3.4.2.4 Algorithm

The final algorithm is shown in `Proposal3D`, Function 3.4.2. To calculate the `Jacobian3D` and the forward kinematics, f, we relied on the Orocos Kinematics Dynamics Library (KDL) [11]. The mechanical description of the object is supplied in a URDF file, parsed, and loaded as a KDL kinematic tree. Using that tree, the forward kinematics and Jacobian can be easily calculated. Since KDL internally uses 6-element twists to represent rotational and translational velocities, which are also members of the Lie algebra of $SE(3)$, we also use them here. As a result, `Jacobian3D` returns a matrix with rows corresponding to twists of observations and columns corresponding to a twist in the base pose and joint velocities. All covariance and precision matrices also use twists to express incremental changes.

Figure 3-17 qualitatively compares our method to the optimization, UKF, and baseline PF discussed as alternatives in Section 3.3. The optimization method suffers from many shortcomings, and the UKF lacks the ability to represent multi-modal distributions. We primarily compare our method against the baseline and, as we will see in the next section, we achieve lower RMSE with fewer particles.

3.4.3 The Pseudo-inverse

We know that small changes in the state space, dx, can be related to small changes in the observation space, dz, via the Jacobian:

$$\underbrace{dz}_{N\times 1} = \underbrace{J}_{N\times M} \quad \underbrace{dx}_{M\times 1} \tag{3.77}$$

And, analogously,

$$\underbrace{dx}_{M\times 1} = \underbrace{J^{\dagger}}_{M\times N} \quad \underbrace{dz}_{N\times 1} \tag{3.78}$$

Of practical importance is how to compute the pseudo-inverse, $J^{\dagger}$, when J is singular. The pseudo-inverse can be defined as:

$$J^{\dagger} = \left(V^{T}\right)^{-1} S^{-1} U^{-1} \tag{3.79}$$

Function 3.4.2: Proposal3D(x_{k-1}, w_{k-1}, R_k, z_k, Σ_{z_k}, Q)

inputs : x_{k-1}, w_{k-1} previous N_s particles and weights
R_k state covariance $\mathfrak{M} \times \mathfrak{M}$ matrix in state space
z_k the $N \times 1$ observations at time k, $z_k \in \mathbb{H}^{N/8}$
$\Sigma_{z_k}^{-1}$ precision matrix for the z_k observations
Q observation selection $\mathfrak{N} \times \mathfrak{N}$ matrix specifying valid observations
δ minimum particle weight threshold

params: f forward kinematic/observation function
P number of discrete elements used to approximate OIF
Σ_{sys} state covariance $\mathfrak{N} \times \mathfrak{N}$ matrix in observation space

outputs: $\{x_k, \alpha\}$ updated N_s particles and weight scales

$[U, S, V] =$ `svd` $(\Sigma_{z_k}^{-1})$ `// NB:` $U = V^T$

$L \leftarrow \sqrt{S}V^T$ `//` $L^T L = \Sigma_{z_k}^{-1}$`; cannot use Cholesky because may be singular`

$x_k \leftarrow \emptyset$; $\alpha \leftarrow \emptyset$

for $x_{k-1}^{(i)} \in x_{k-1}$ **do**

  $s \leftarrow 0$

  repeat

    $J_x \leftarrow$ `Jacobian` 3D(f, $x_{k-1}^{(i)}$)

    $m_k \leftarrow x_{k-1}^{(i)} \oplus (LQJ_x)^\dagger \; LQ \; \left(z_k \boxminus f\left(x_{k-1}^{(i)}\right)\right)$

    `//` create N_s `samples via rejection sampling where`

    `//` $\mathcal{X}^{(i)}$ `is discretely distributed according to` $P(x|x_{k-1}^{(i)}, z_k)$

    ... see Function 3.4.1 ...

    `//` `determine OIF probability of each candidate particle in` $\mathcal{X}^{(i)}$

    ... see Function 3.4.1 ...

    $s \leftarrow s + 1$

  until $P\left(z_k | x_{k-1}^{(i)}\right) > \delta$ **or** $s \geq 3$ `// repeat until sufficient weight`

  for $\mathcal{X}_j \in \mathcal{X}^{(i)}$ **do**

    $P\left(\mathcal{X}_j \left| x_{k-1}^{(i)}, z_k \right.\right) \leftarrow \frac{P(z_k|\mathcal{X}_j) \cdot P\left(\mathcal{X}_j | x_{k-1}^{(i)}\right)}{P\left(z_k | x_{k-1}^{(i)}\right)}$

  `//` `sample from` $\mathcal{X}^{(i)}$ `according to OIF probabilities`

  $x_k \leftarrow \{x_k,$ `RandomSample` $(\mathcal{X}^{(i)}, P(\mathcal{X}^{(i)}|x_{k-1}^{(i)}, z_k))\}$

  $\alpha \leftarrow \{\alpha, P\left(z_k | x_{k-1}^{(i)}\right)\}$ `// append weight to` α

return $\{x_k, \alpha\}$

Method	Description	Provides temporal coherence	Handles joint limits	Reduces state param. dependence	Handles partial observations	Incorporates (partial) observations during proposal	Provides estimate of uncertainty	Does not require accurate state transition model	Handles multi-modal distributions
Optimization	Optimize the state of the articulated object until it most nearly matches the observations.		Requires constraint-based optimization			N/A			
UKF	Track state of articulated object via a UKF.		Requires additional optimization			N/A		An accurate state transition model reduces param. dependence	
Baseline PF	Track via PF and propose using state transition model.							Particles are proposed via the state transition model	
Our Method (PF)	Track via PF and propose using observation model.								

Figure 3-17: Qualitative comparison of optimization, UKF, baseline PF, and our method. Red indicates not supported; yellow indicates supported under additional conditions; and green, indicates fully supported.

where

$$[U, S, V] = \text{svd}\,(J) \tag{3.80}$$

such that $USV^T = J$.

Since V and U are orthogonal matrices representing rotations, $U^{-1} = U^T$ and $V^{-1} = V^T$. Further, S is a diagonal matrix so its inverse can be obtained by inverting each element along the diagonal. Therefore,

$$J^\dagger = VS^{-1}U^T \tag{3.81}$$

Suppose we define $\overline{dz} = U^T dz$ and $\overline{dx} = V^T dx$. In this way, $\overline{dz}$ and $\overline{dx}$ are rotated versions of their counterparts, expressed in the U and V bases. Then:

$$dz = Jdx \tag{3.82}$$

$$U\overline{dz} = JV\overline{dx} \tag{3.83}$$

$$U\overline{dz} = US\underbrace{V^TV}_{=I}\overline{dx} \tag{3.84}$$

$$U^TU\overline{dz} = \underbrace{U^TU}_{=I}S\overline{dx} \tag{3.85}$$

$$\overline{dz} = S\overline{dx} \tag{3.86}$$

It becomes evident that the diagonal values in S scale each dimension of the state space to a corresponding change in the observation space. A similar analysis can be performed to show that:

$$S^{-1}\overline{dz} = \overline{dx} \tag{3.87}$$

This form of the equation demonstrates the well-known issue with matrix inversion — when the diagonal elements of S are small, the diagonal elements of S^{-1} are large. This means that a small change in the observation space can result in a large change in the state space. At the extreme, dividing by a zero eigenvalue will result in an infinite scaling. Most implementations prevent this and instead of using inversion, rely on an approximation.

For example, suppose:

$$S = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \tag{3.88}$$

$$S^\dagger = \begin{bmatrix} f(\sigma_1) & 0 & 0 \\ 0 & f(\sigma_2) & 0 \\ 0 & 0 & f(\sigma_3) \end{bmatrix} \tag{3.89}$$

In the most typical implementation:

$$f(\sigma) = \begin{cases} 1/\sigma & \sigma > \tau \\ 0 & else \end{cases} \tag{3.90}$$

where τ is chosen based on the CPU precision. In our case, however, choosing $\tau = 10^{-6}$, for example, would allow small changes in the observations to be scaled by a factor of 10^6. Recalling that dz is Gaussian noise (Equation 3.67), scaling dz by such a large value would produce state estimates which are unlikely in terms of $P\left(x \left| x_{k-1}^{(i)}, z_k \right.\right)$. In other words, small noise in the observations along degenerate state dimensions will cause the state to move significantly. The result is that both the likelihood of observations will be small and the likelihood of the state, relative to the previous states, will be small. In that event, the particle filter may fail with a collection of unlikely hypotheses.

Several possibilities are listed here and depicted in Figure 3-18:

1. Naïve. This solution simply implements the inverse and ignores the divide-by-zero problem.
$$f(\sigma) = \frac{1}{\sigma} \tag{3.91}$$

2. Shift. This solution is common in velocity controllers. The addition of a small term in the denominator guarantees that large reciprocals are avoided.
$$f(\sigma) = \frac{1}{\sigma + \tau} \tag{3.92}$$

3. Truncate. This method avoids large reciprocal eigenvalues by simply setting them to zero.
$$f(\sigma) = \begin{cases} 1/\sigma & \sigma > \tau \\ 0 & else \end{cases} \tag{3.93}$$

4. Clip. Here, reciprocals are simply limited to a maximum value.
$$f(\sigma) = \begin{cases} 1/\sigma & \sigma > \tau \\ 1/\tau & else \end{cases} \tag{3.94}$$

5. Taper. With this method, developed in [56], reciprocal eigenvalues larger than the threshold are linearly tapered to zero.
$$f(\sigma) = \begin{cases} 1/\sigma & \sigma > \tau \\ \sigma/\tau^2 & else \end{cases} \tag{3.95}$$

The problem with the naïve method is that small eigenvalues cause large eigenvalues in the pseudo-inverse (as discussed previously). The Shift and Clip methods are problematic because when $\sigma = \infty$, then, truly, $f(\sigma) = 0$. This means that, when motion along a state dimension is not possible, observation noise should not be able to artificially create it. The Shift and Clip methods, however, still permit motion (i.e., the null space always $\emptyset$). The Truncate method, perhaps the most common, suffers from none of the previous issues. However, it does have the undesirable discontinuity at $\sigma = \tau$ — any eigenvalue below this threshold is summarily ignored and

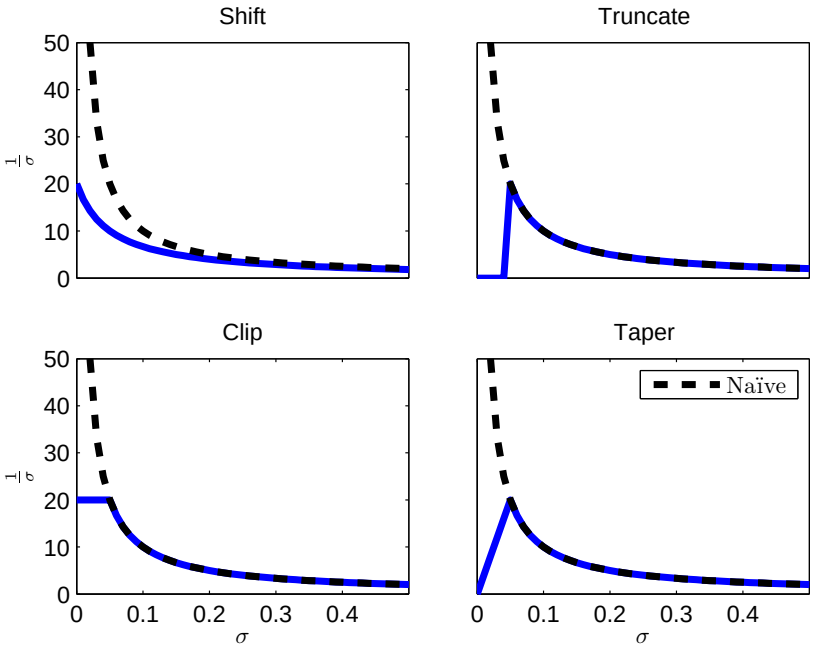


Figure 3-18: The Naïve method of calculating the pseudo-inverse (dashed line) suffers from a singularity when an eigenvalue is zero, i.e., $\sigma = 0$. Several heuristics exist for handling this situation in practice.

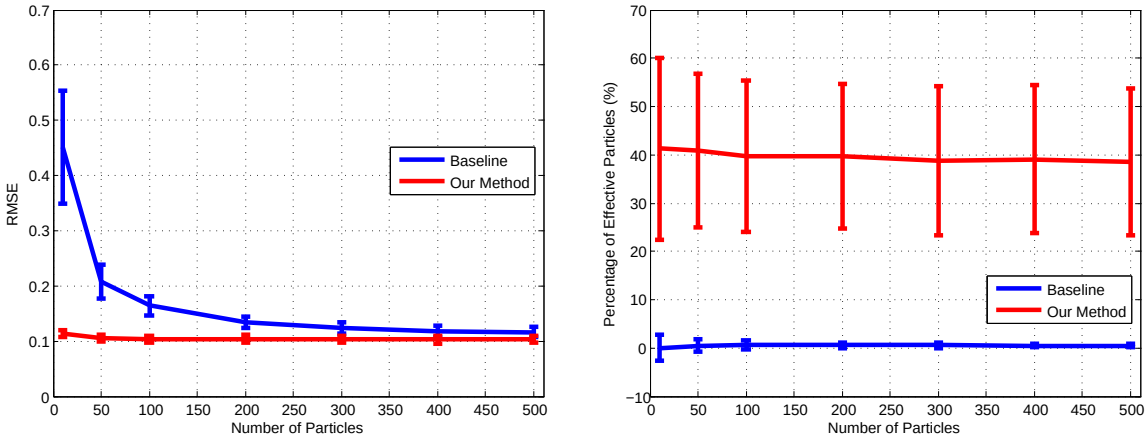
shifted into the null space. This is a problem in our algorithm, because we will be choosing a relatively high $\tau = 1$ (to avoid large movement in the state space), and, as a result, many dimensions of the state may receive no noise perturbations. Note that, although we chose a high τ, we enter this region (and thus use the state transition model) only occasionally.

The Taper method has the advantage that, as an eigenvalue becomes nearly zero (the matrix becomes increasingly singular), the eigenvalues are increasingly moved into the null space. Further, despite even a large value of τ, noise can always be produced in any dimension.

3.5 Experiments

3.5.1 Planar Simulation

We compared the baseline approach and our method on two different examples in simulation: a four-link kinematic chain with pose observations and a dishwasher with a door and two drawers. For each system, we (1) varied the number of particles, (2) performed 100 Monte Carlo simulations, and (3) evaluated the root mean squared error (RMSE) of the tracker.



(a) RMSE

(b) Number of effective particles

Figure 3-19: Results for the kinematic chain. Error bars show one standard deviation about the mean.

3.5.1.1 Four-link Kinematic Chain with Pose Observations

The kinematic chain in Figure 3-8 was simulated with a state vector consisting of the $SE(2)$ pose of the leftmost link and each of the joint values (c.f. Equation 3.53). The observation vector consisted of an $SE(2)$ pose for each of the links (c.f. Equation 3.54).

As shown in Figure 3-19a, our method achieves the same tracking accuracy (RMSE) with 10 particles as does the baseline method with ~500 particles. As seen in Figure 3-19b, our method also demonstrated a higher average N_{eff} indicating that the particles cover the state space more effectively.

3.5.1.2 Parametrizations and the Four-link Kinematic Chain

In Section 3.3.3, we examined two different parametrizations of the four-link kinematic chain and demonstrated that the choice of parametrization affects the RMSE for the UKF (see Figure 3-11c). We performed the same experiment, using the parametrizations in Equation 3.56 and Equation 3.57 for our method. As shown in Figure 3-20, the RMSE is nearly the same for both state parametrizations using our method. Our method will generally exhibit reduced dependence on the state parametrization. However, it is not strictly independent of the state parametrization. Dependencies may arise when degeneracies exist in the observation space and the method begins to propose from the state transition model.

3.5.1.3 Dishwasher

A dishwasher is a simple articulated object which a household robot is likely to encounter (see Figure 3-21). The state for the dishwasher consists of a pose $p \in SE(2)$

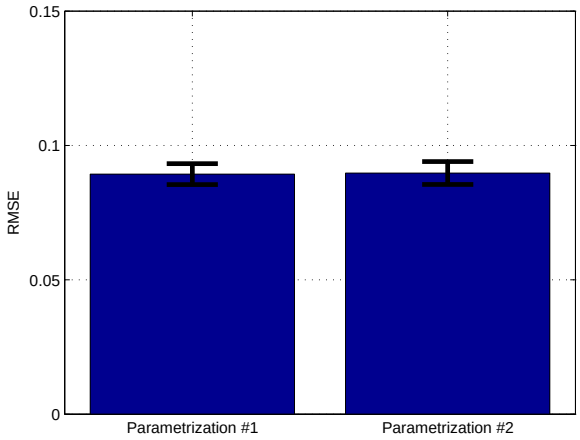


Figure 3-20: Different state parametrizations do not significantly affect the RMSE for the four-link kinematic chain. The error bars indicate one standard deviation. This was not the case for the UKF (see Figure 3-11).

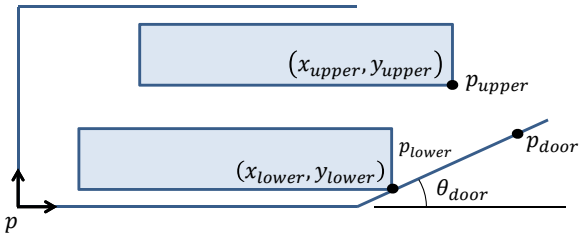


Figure 3-21: The dishwasher consists of two articulated drawers and one door. Joint limits prevent configurations in which the door and drawers would overlap or extend beyond physical limits. y_{upper} and y_{lower} are static parts of the kinematic model.

for the dishwasher's basin and joint values for its doors and drawers:

$$x = \begin{bmatrix} p & \theta_{door} & x_{upper} & x_{lower} \end{bmatrix}^T \tag{3.96}$$

The observations consist of poses for the dishwasher door and each of the two drawers, $z \in SE(2)^3$:

$$z = \begin{bmatrix} p_{door} & p_{upper} & p_{lower} \end{bmatrix}^T \tag{3.97}$$

Unlike the previous example, the dishwasher joints are subject to joint limits. The drawers have limited travel in the horizontal direction and the door cannot close into the drawers. These constraints are satisfied via rejection sampling.

$$0 \leq \theta_{door} \leq \min\left(\text{atan}\left(\frac{y_{upper}}{x_{upper}}\right), \text{atan}\left(\frac{y_{lower}}{x_{lower}}\right)\right) \tag{3.98}$$

$$0 \leq x_{upper} \leq x_{max} \tag{3.99}$$

$$0 \leq x_{lower} \leq x_{max} \tag{3.100}$$

The RMSE for the dishwasher, using our method (Figure 3-22), was relatively constant at ~0.02 for 10-500 particles. The baseline method did not achieve that performance with fewer than ~500 particles. Additionally, our method averaged 40-60% effective particles, compared to less than 5% with the baseline method.

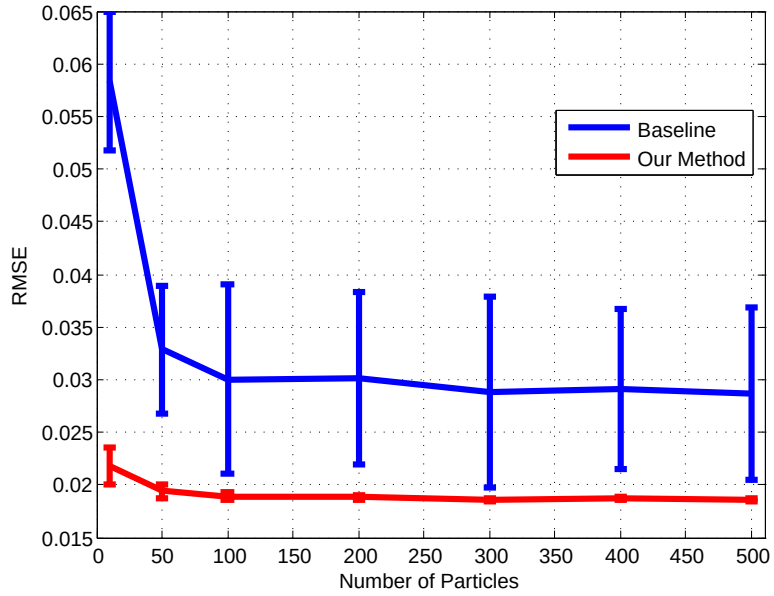


Figure 3-22: Results for the dishwasher simulation are shown.

3.5.1.4 Multi-modal Kinematic Chains

Since the algorithm uses a particle filter, it can successfully model a multi-modal system. To demonstrate this, we simulate a 4-link kinematic chain over 27 frames. This chain moves like a spring, "contracting" along the horizontal axis. The middle column of each row in Figure 3-23 shows a sample frame from the simulation. In this case, the observations are at the center of each link, such that they do not disambiguate between two possible configurations, as shown by the rendered particles in the right column of Figure 3-23. The two clusters of particles are mirrored about the horizontal axis and explain the observations equally well. It is not until frame 24 in the fourth row that the right-most link moves such that the particles collapse to a single, correct hypothesis. Figure 3-24 shows the weights associated with the the two "positive" and "negative" configuration modes.

The ability to track a multi-modal distribution, as shown here, is a common motivation for using the particle filter. In our case, multiple modes often arise from an object with redundant configurations, missing observations, or ambiguous observations. A UKF, able to maintain only a single hypothesis, will converge to one of the modes and may experience large tracking error when the wrong one is tracked.

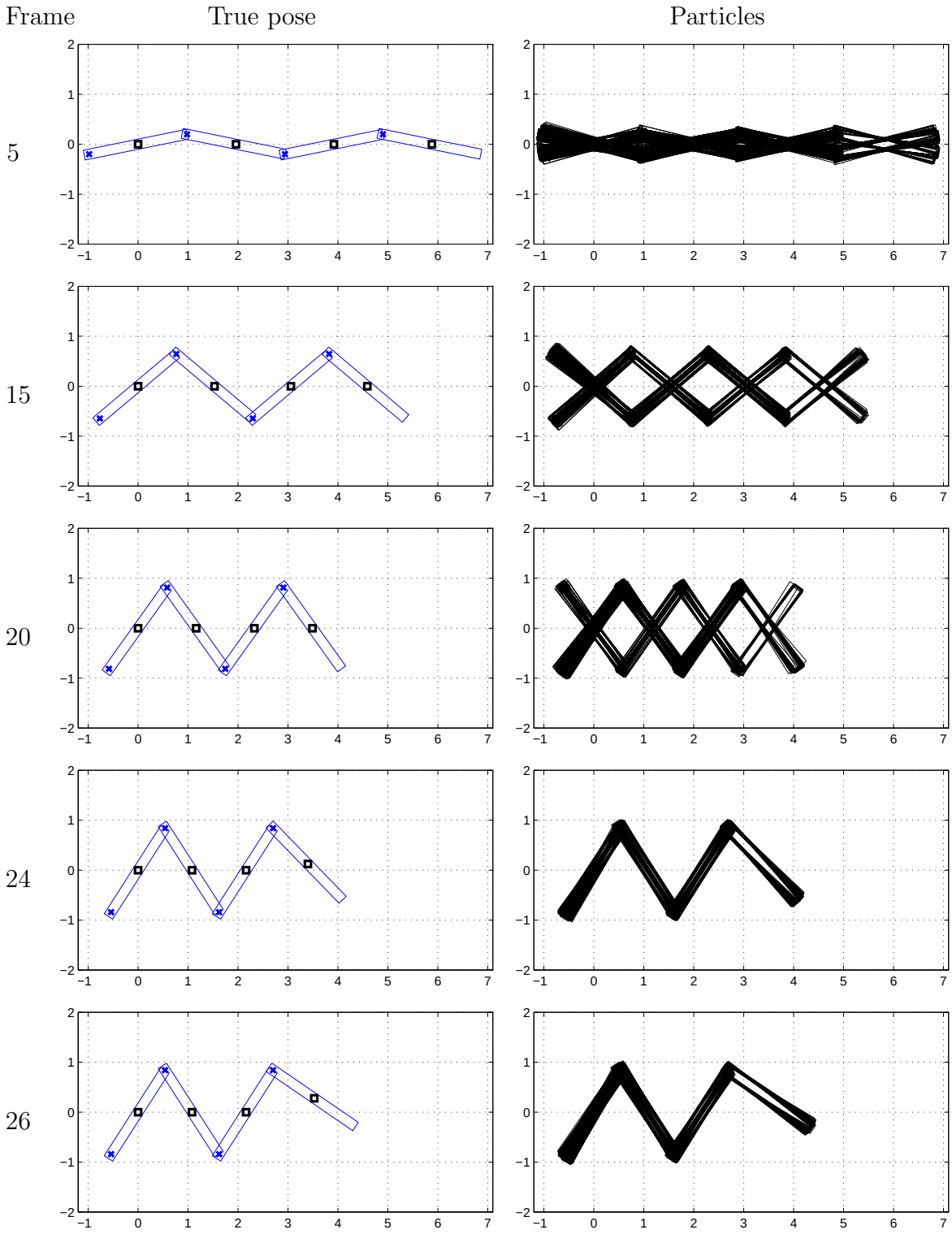


Figure 3-23: The true motion of a 4-link chain (blue) is shown in the middle column for several sample frames. Observations are shown as black squares. For each frame (row), the configurations associated with the particles are shown on the right (black). Note that until frame 24, when the motion of the chain is unambiguous, two clusters are maintained by the particles.

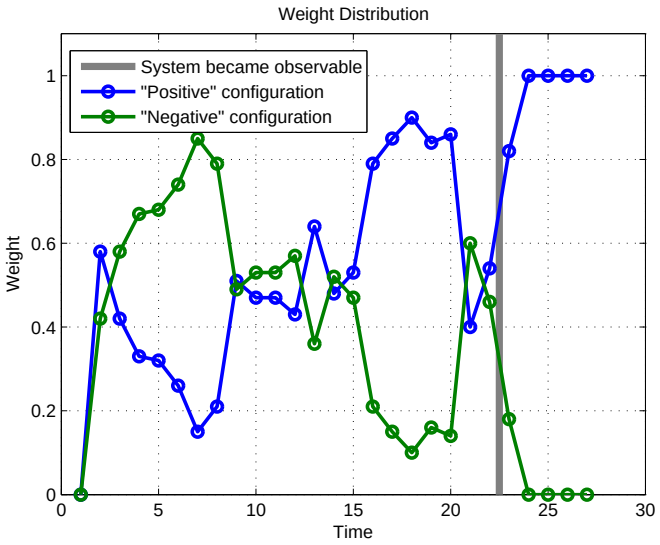


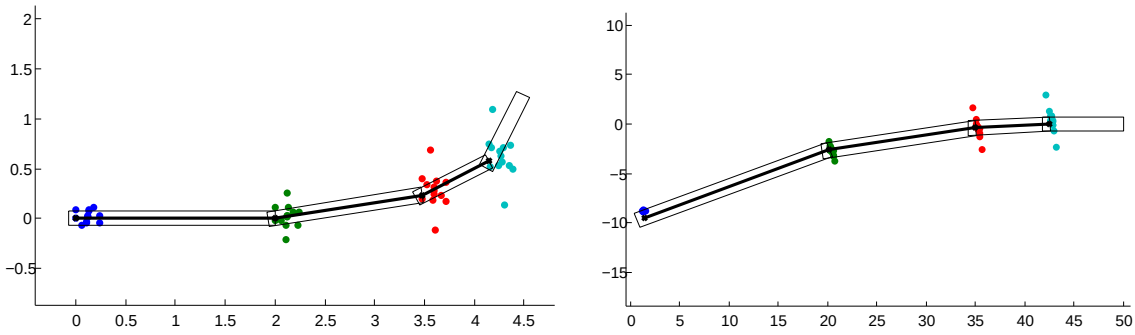
Figure 3-24: Particle weights are divided between two clusters explaining the observations until the system moves to disambiguate the configuration. With noise-free observations, the weights would be nearly equal until frame 24; the results from noisy observations shown here cause the particle weights to deviate from exactly 50%.

3.5.1.5 Comparison with UKF

As mentioned in Section 3.3.3, the UKF exhibits a state parametrization dependence. In this example, we demonstrate that, when the articulated object's motion is favorable to the parametrization, both our method and the UKF perform similarly. However, when the encoding does not correspond to the motion well (i.e., such that the UKF's sigma points do not cover likely configurations well), our method has lower RMSE than the UKF.

The columns of Figure 3-26 show sample frames from two simulations for two different kinematic chains. Both simulations use the parametrization of Equation 3.56, including the base pose of the left-most link and the three joint angles. Because both simulations use the same parametrization, both propose similar sigma points (see Figure 3-25). In particular, note that sigma points near the base pose (blue) are tightly grouped, while sigma points near the right-most link (cyan) are spaced further apart. Again, this difference in spacing is due to the non-linear nature of the parametrization. In other words, this difference results because the UKF uses Euclidean addition in the state space to add to the mean state (c.f., Equation 3.55).

If a state transition model is particularly good, then the small differences in sigma points may be negligible. However, our work does not assume that a good state transition model is available. For example, in Simulation #1, the left-most link (base pose) remains fixed and the right-most link moves significantly; the sigma points cover the motion well. On the other hand, for Simulation #2, the left-most link moves significantly and the right-most link remains fixed; the sigma points do not cover the motion well.



(a) Sample sigma points for Simulation #1

(b) Sample sigma points for Simulation #2

Figure 3-25: Sigma points for UKF simulations.

As seen in Figure 3-27a for Simulation #1, the UKF and our method perform similarly well. However, in 3-27b for Simulation #2, our method exhibits lower RMSE than the UKF. This is because our method is less dependent on the state parametrization. In situations where a good state transition model was available or the "best" parametrization could be guaranteed, the UKF might be a reasonable alternative.

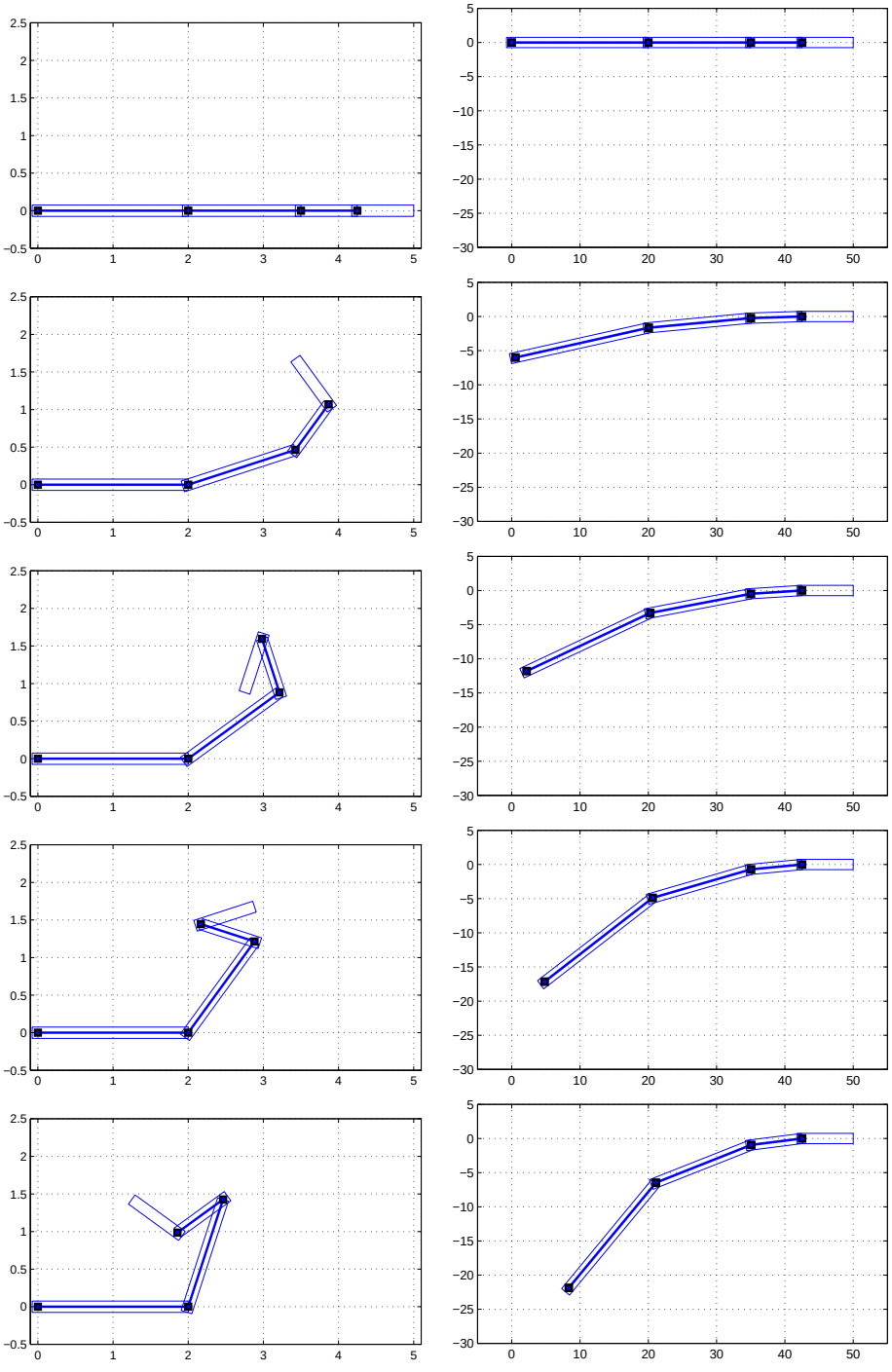
3.5.2 Planar Kinematic Chain

We constructed the 5-DOF planar kinematic chain shown in Figure 3-28 with a known kinematic model. For this "toy" example, a 6cm AprilTag [57] was affixed to each 30cm link, allowing the $SE(2)$ pose of each link to be observed. We then manually moved the chain through a series of continuous, known configurations. At each configuration, we obtained several seconds of data from different camera positions, creating many different samplings of sensor noise and emulating the Monte Carlo technique used in simulation. Each rotational joint was stepped in 5 degree (0.09 rad) increments through a series of 19 positions. By comparing the known ground truth to the observations, we found the observation noise to be about 0.05 rad/frame and 0.05 cm/frame.

As shown in Figure 3-29, our method outperforms the baseline method, as it is able to take advantage of relatively accurate observations and noise proposal in the observation space. The number of effective particles, 10-30%, was lower than in simulation, but still exceeded the baseline method's performance, which averaged about one effective particle. The baseline method is hampered because it depends on a state transition model for proposal and only a poor one is available. Our method proposes particles specifically chosen to produce the noise expected in the observations.

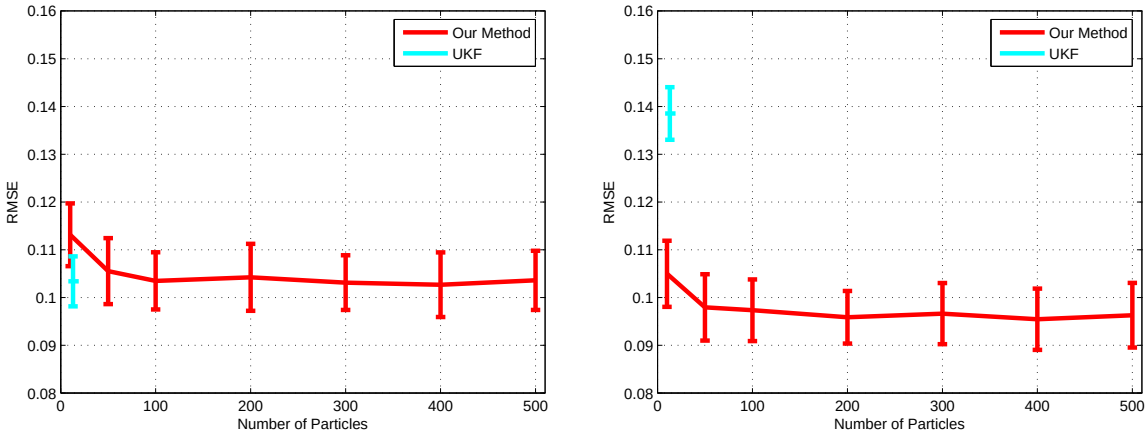
3.5.3 Dishwasher

Approximately 1300 frames of RGB-D data were collected of a dishwasher (see Figure 3-30) being opened and closed. Ground truth and the kinematic model were



(i) Simulation #1 (j) Simulation #2

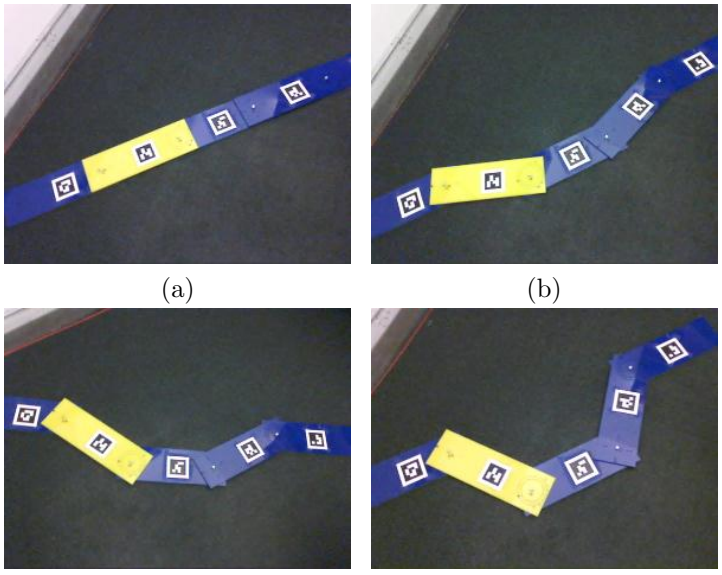
Figure 3-26: The same parametrization is used for two different simulations, resulting in different UKF performance, relative to our method (see Figure 3-27).



(a) RMSE for Simulation #1

(b) RMSE for Simulation #2

Figure 3-27: The RMSE performance of our method and the UKF is similar for Simulation #1, where the parametrization produces favorable sigma points. This is not always the case, however, as illustrated by Simulation #2. The x-axis location of the UKF results corresponds to the number of sigma points.



(a) (b)

(c) (d)

Figure 3-28: Sample configurations for this 5-DOF toy example were constructed with "stop-frame" style animation.

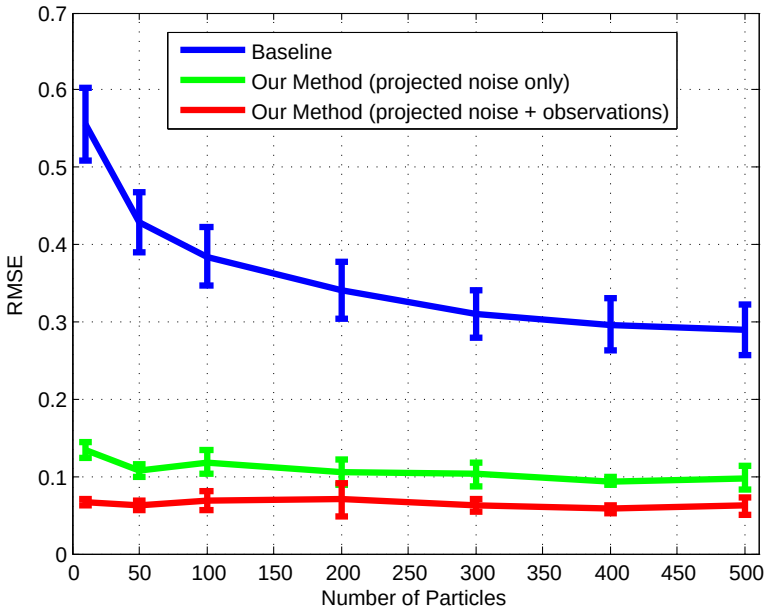


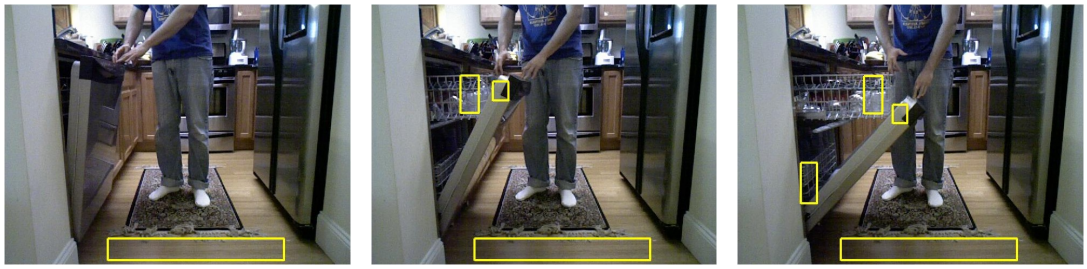
Figure 3-29: For the kinematic chain, proposing noise in the observation space (green) yields RMSE improvement over the baseline approach (blue). RMSE is further reduced (red) by centering proposals around the observations.

established via manual annotation. Three independent TLD [41] trackers were manually initialized on the door and drawers on the first frame they came into view. As a result, there are many frames with missing observations. For example, in the first frame, the drawers are not tracked because they are obscured by the door. Some observations are also missing as the TLD tracker occasionally lost track. Nonetheless, our algorithm is still able to use these partial observations during particle proposal, c.f. Equation 3.65.

Unlike the simulation (which uses poses), these observations consist of the positions of the door and drawers and a vertical estimated from a static patch on the floor. Although the vertical is not necessary for observability, we found both the baseline and our method benefited from the additional information.

Since only the positions of the dishwasher door and drawer are available, a singularity exists when the door is closed (vertical). In this configuration, observation movement in the horizontal direction can be explained both by movement of the base pose or by a slight opening of the door and movement of the drawers. The null space term in Equation 3.67 is crucial so that meaningful particles are proposed during the initial frames.

Figure 3-31 shows results for the dishwasher. In addition to higher accuracy, our method also exhibits substantially less variation. This is primarily due to periods when the observations were insufficient to make the system observable (which did not occur in the other examples). Both methods would "wander" during these singularities, each proposing unconstrained particles in the null space. However, when observations became available again, our method was able to quickly recover and



(a) (b) (c)

Figure 3-30: A TLD tracker provided positions of the dishwasher's articulated links as input. A vertical was also extracted.

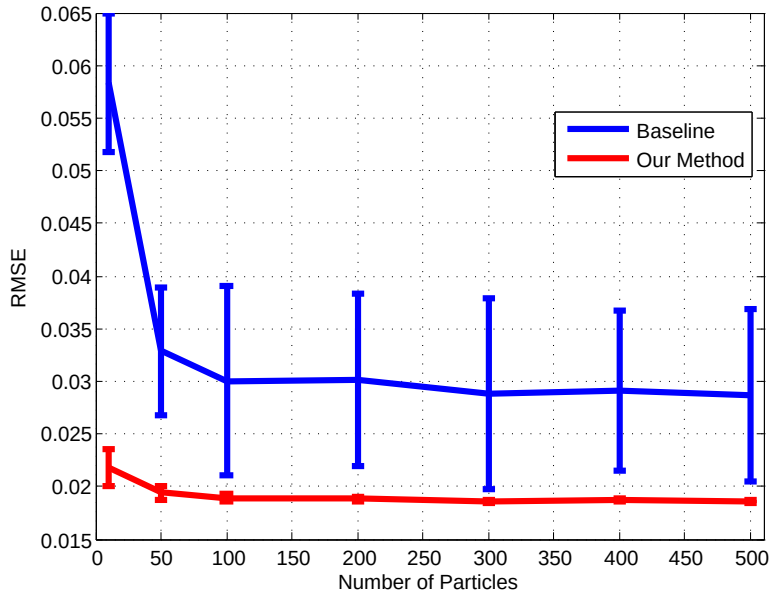


Figure 3-31: In addition to lower RMSE, our method demonstrated less variation in accuracy while tracking the dishwasher, because it quickly recovered when missing observations resumed.

begin proposing particles near those observations. Our Matlab implementation executed at ~23 FPS, exceeding the 10 FPS rate of the RGB-D data. Finally, our method maintained ~50% effective particles, while the baseline had only about 5%.

We also consider the effect of model noise on performance. As mentioned, we anticipate that most models will be imperfect. We simulated Gaussian noise with a standard deviation of 0.05m on the dishwasher housing width, height, and the vertical locations of the two drawers. A comparison of the model error to RMSE tracking performance is shown in Figure 3-32 with best fit lines. The lines have similar slopes, indicating that, for the dishwasher example, both the baseline approach and our method behave comparably under increasing model noise.

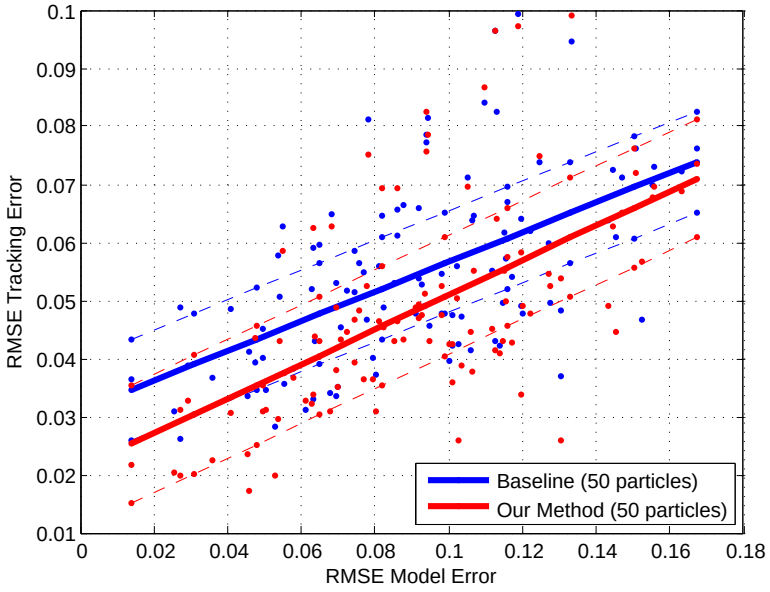


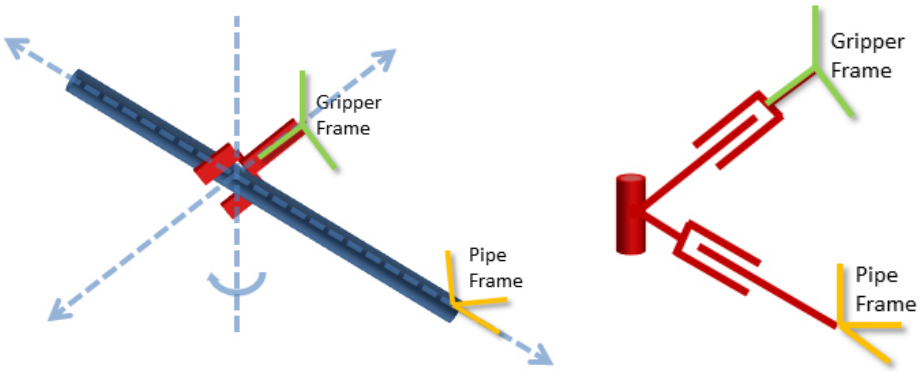
Figure 3-32: The baseline and our method are affected similarly by model noise. Dotted lines show one standard deviation.

3.5.4 PR2

In this experiment, we consider a PR2 robot holding a 60cm PVC pipe, and the goal is to estimate the pose of the pipe. This task is interesting because PR2 cannot grip the pipe rigidly due to its weight and length. As a result, the pipe tends to slip in the gripper; however, this slip occurs only along certain directions. As suggested in Figure 3-33, the pipe may translate in the gripper along the pipe's long axis or along the grip direction. The pipe can also rotate about the suggested axis.

We might wish to know the pose of the pipe so that it can be accurately manipulated — when, for example, inserting into a mating coupling. An obvious solution might be to track the pipe in RGB-D data using a RANSAC cylinder fit. A complication with this approach, however, is that the absolute pose of the pipe is not observable. Rotation about the pipe's long axis cannot be observed (by any of the PR2's sensors) due to the rotational symmetry. (In practice, because the end of the pipe is small, we also found that the position estimate along the long axis was also inaccurate.) Thus, even recovering a relative pose of the pipe is not possible by observing only the pipe itself.

Fortunately, more information is available. Although, as mentioned, the gripper does not hold the pipe rigidly along all six DOFs, it does provide rigid support along three DOFs. Thus, 6-DOF information from the pose of the gripper (as provided by the PR2's telemetry and forward kinematics) and 4-DOF information about the partial pose of the pipe (as provided by a RANSAC cylinder fit on RGB-D data) can be combined to track the pipe in 6-DOF. (Note that we do not track the orientation about the pipe's long axis absolutely — this rotation is tracked relative to the initial orientation.) It is clear that the RANSAC estimate of the pipe's location will have



(a) (b)

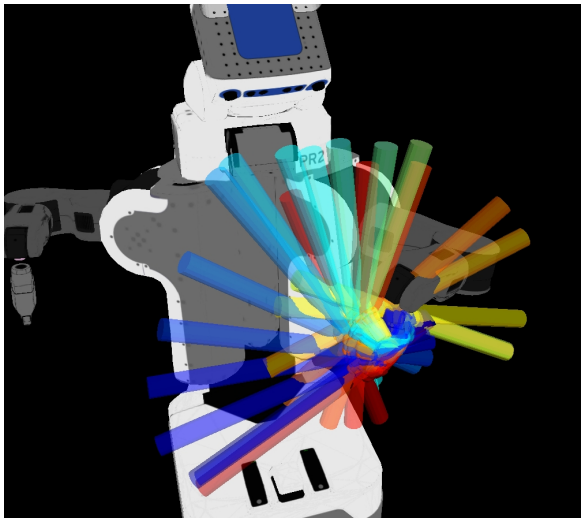
Figure 3-33: The PR2's grip on the pipe is not rigid, but still constrains some movement. The system can be modeled as a 3-DOF kinematic chain.

errors; but the pose of the gripper is also subject to errors. Mechanical slop, imperfect PR2 models, and errors in the Kinect calibration all lead to errors in the Kinect-to-gripper transform.

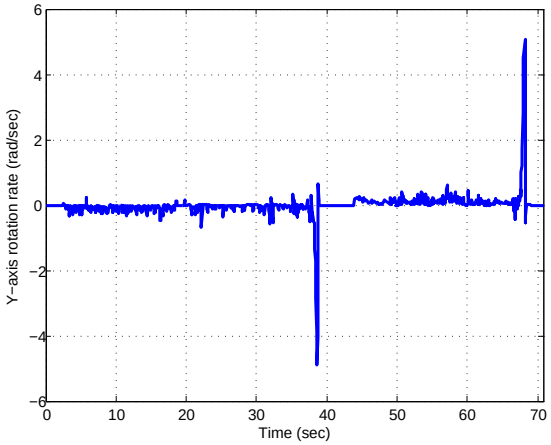
We achieve the pipe tracking by explicitly modeling the DOFs between the gripper and the pipe. As shown in Figure 3-33b, the system can be modeled as a 6-DOF pose of the gripper and a kinematic chain with three joints (two prismatic and one rotational). Two sequences were collected with the PR2. The "pipe-swing" scenario in Figure 3-34 demonstrates significant pipe rotation in the gripper as the gripper rotates. Figure 3-34b shows the rotational velocity; notice two significant spikes in velocity prior to 40 and 70 seconds. These spikes correspond to events when the pipe passed a vertical orientation and "fell" (or, more accurately, slipped) in the gripper. During these two events, the pipe moved over 90 degrees in just 2-3 frames, and the state transition model was particularly poor especially during these frames.

In the second "axis-rotate" scenario, the PR2 moved to rotate the pipe primarily along its long axis (see Figure 3-35). Figure 3-35b shows the rotational velocity as the gripper makes several forward and backward rotations with the pipe. Here, we track the pipe's pose (relative to the start pose). This would not be possible by observing only the pipe (the pipe's orientation cannot be determined due to the rotational symmetry). Information from the gripper's pose is combined (via the kinematic chain constraints) to estimate a full 6-DOF pose of the gripper. It is important to note that this ability is a result of the model in Figure 3-33b and not our particular tracking technique.

Figure 3-36 and Figure 3-37 show the RMSE and number of effective particles for the pipe-swing and the axis-rotate experiments, respectively. In both cases, our method achieves error levels at around 20 particles lower than those achieved by the baseline method with fewer than 500 particles. The N_{eff} is also higher, indicating that our particles are located in more high-probability regions.

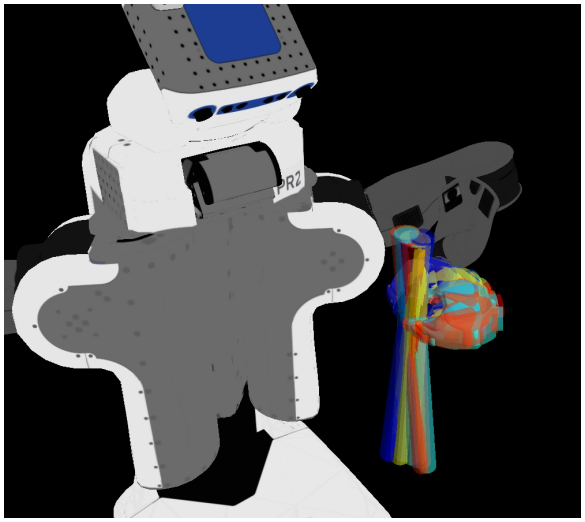


(a)

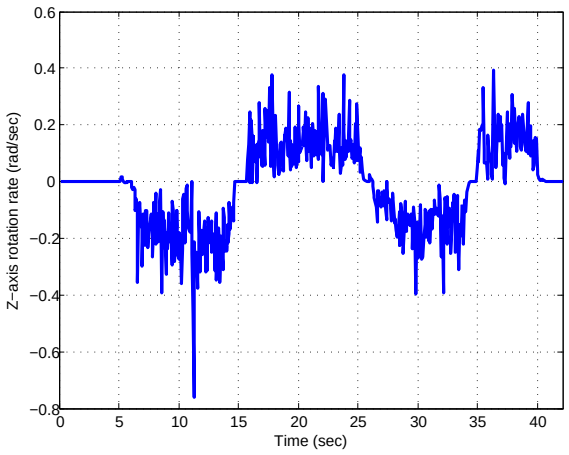


(b)

Figure 3-34: In this sequence, the PR2 rotates the pipe. (a) shows the pipe poses color coded by time; the pipe proceeds through red-yellow-blue poses. The two large velocity spikes in (b) correspond to times when the pipe underwent significant slip in the gripper.

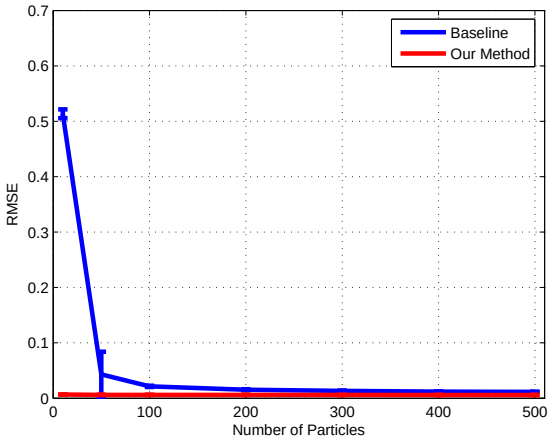


(a)

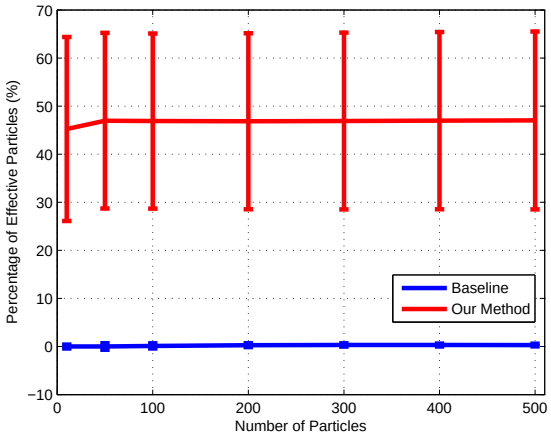


(b)

Figure 3-35: In this sequence, the PR2 moved its gripper so as to rotate the pipe approximately along its long axis (here, nearly vertical).

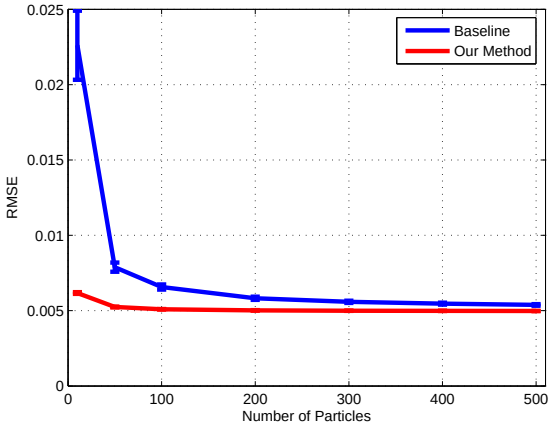


(a)

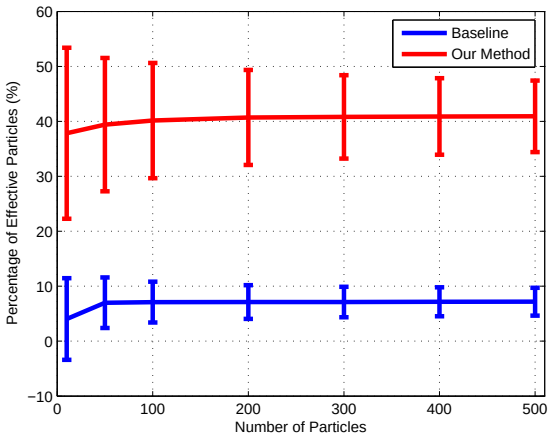


(b)

Figure 3-36: RMSE and number of effective particle performance for the pipe-swing sequence in Figure 3-34 are shown.



(a)



(b)

Figure 3-37: RMSE and number of effective particle performance for the axis-rotate sequence in Figure 3-35 are shown.

Figure 3-38: At HCA in Brentwood, NH, operators train on several different types of equipment in parallel.

3.5.5 Excavator

Section 3.1 presented a motivating construction site example. Here, an autonomous robot is working alongside human-operated heavy construction equipment. As described, the robot is responsible for safely navigating the busy, crowded environment. The robot has no communication with the operators or interface with the equipment, and relatively poor predictive models for how it will move. Because the equipment is standard, the robot does have kinematic models and can leverage them to estimate the pose of the construction equipment even when it is only partially visible.

In this experiment, we visited the Heavy Construction Academy (HCA) in Brentwood, NH (see Figure 3-38). Here, operators train on various types of equipment, including excavators, backhoes, front-end loaders, bulldozers, dump trucks, graters, and skidsteer loaders. We observed an excavator operating and collected data with a PointGrey CMLN-13S2C camera and Velodyne HDL-32E 3D LIDAR. (The Velodyne was not used in the following experiments, but did provide a check during ground truth processing.) We also collected video using a Sony HF10 camcorder. Both the PointGrey camera and Sony camcorder were calibrated using a standard checkerboard technique [5].

The excavator is a CATERPILLAR 322BL (CAT322BL), and includes a tracked base and rotating operator cab/engine. The heavy lift arm consists of three links, referred to as the boom, the stick, and the bucket (proceeding outward from the cab). We constructed a model of the CAT322BL excavator using third-party datasheets [62]. The datasheets were not complete, but provided adequate measurements to make reasonable extrapolations. The model was constructed in SolidWorks (see Figure 3-40) and a custom SolidWorks macro was used to create a URDF file and STL mesh files. The URDF file was parsed to produce a KDL object from which the forward kinematics could be calculated. Combined with the STL mesh files, the URDF could

Figure 3-39: We captured operation of an excavator with a PointGrey camera (mounted left) and 3D LIDAR (mounted right).

also be used to render the excavator in different configurations. We do not expect that our model of the CAT322BL is perfect but, as we will see, it demonstrates that even approximate models are sufficient to achieve quality results.

The image-based observations were made using the seven independent TLD [41] trackers. The trackers were manually initialized during the first frame on the points shown in Figure 3-16a. The observations included two points on the tracks, two points on the cab (the point on the back of the cab is not visible in this frame), a point on the boom, a point on the stick, and a point on the bucket.

The excavator has 10-DOFs: six DOFs for the base pose and four joints (track-

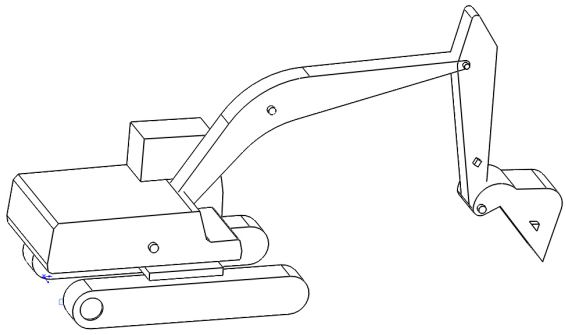


Figure 3-40: The CATERPILLAR 322BL excavator was modeled in SolidWorks using specifications available online [62].

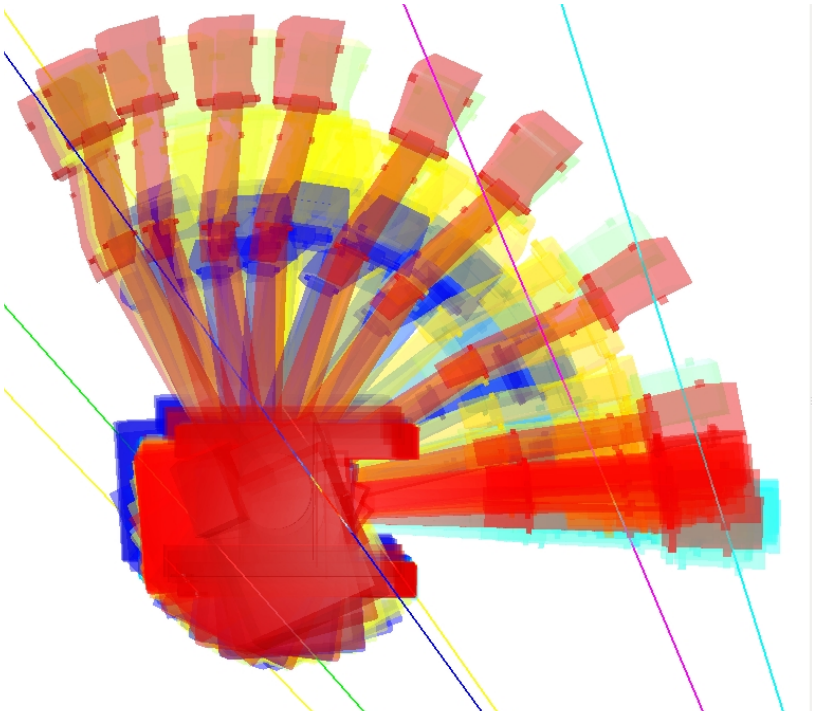


Figure 3-41: The excavator loads a dump truck three times. Viewed from above, this graphic shows the entire 1645-frame sequence, color coded by time. The three loads correspond to blue, yellow, and red, in that order.

cab joint, cab-boom joint, boom-stick joint, and stick-bucket joint). Each of the pixel observations defines a ray in 3D space (see Figure 3-16c), corresponding to two constraints. As a result, the seven observations create 14 constraints, and it can be shown that these constraints are sufficient to observe the 10-DOF excavator configuration. In principle, as few as five observations are required. However, the observation on the back of the cab is often obscured and we found the tracking was improved (for all trackers) with an additional point on the tracks. As described in Section 3.4.2.1, the pixel locations for the observations are encoded as 6-DOF poses with singular precision matrices.

3.5.5.1 Results for Dump Truck Loading

The excavator loads a dump truck (see Figure 3-16a) with three bucket fulls, repeatedly moving from roughly parallel to the camera's image plane to facing away from the camera. Ground truth was established by manually entering key-frames. Between key-frames, DQ SLERP (Section 2.5.3) was used to interpolate the base pose and linear interpolation was used for the joint values. The location of key-frames was established by aligning against both the camera images and the 3D LIDAR data. The ground truth is shown in Figure 3-41.

The plots in Figure 3-42 show the RMSE and the effective number of particles of our method versus several alternatives. We evaluate our method against a baseline

PF and UKF. We also experimented with a different state transition model: our method uses a zero-velocity model, as do the baseline and UKF methods. However, to give the baseline and UKF methods a possible benefit, we also evaluate them with a constant-velocity state transition model.

1. Baseline. This is the standard particle filter, as used in previous comparisons, which proposes from the state transition model.

2. Baseline with constant velocity. This method combines the baseline particle filter with a constant velocity state transition model. The excavator's state is expanded to include velocity components for all DOFs, creating a 20 DOF state vector.

3. UKF. The UKF (see Section 3.2.5) maintains a single hypothesis for the 10 DOF configuration of the excavator. For a 10-DOF system, the UKF creates 21 sigma points (see Algorithm 3.2.3). Since sigma points are somewhat analogous to particles in a PF, we plot UKF results with a "Number of Particles" (x-axis) equal to 21.

4. UKF with constant velocity. Similar to the baseline with constant velocity, this method uses a UKF with a state expanded to include velocity terms. The results are plotted with a "Number of Effective" particles of 41, again corresponding to the number of sigma points.

For the particle filters (baseline and our method), the filter is considered to have failed if all particles have zero probability. The baseline methods were simulated with 400-2000 particles (using fewer than 400 particles resulted in failure every time). Figure 3-42a and 3-42b show the RMSE results, at different scales. In the following, we discuss a few interesting aspects to these graphs.

First, our method performs better than the UKF and UKF with constant velocity methods. As noted in Section 3.5.1.5 where we considered the UKF in simulation, the UKF and our method can achieve the same RMSE depending on the circumstance. However, for the excavator, the multi-modal nature of the distribution of configurations (see Figure 3-6 for examples) impairs the UKF. As an example, consider Figure 3-43a. Here, the excavator was turned away from the camera, and the stick and bucket were obscured. As the excavator came back into view, the bucket was visible before the stick. As a result, two kinematic configurations explained the observations and this was captured by the particles of our method. The UKF, on the other hand, was attracted toward the wrong local minimum (Figure 3-43b). The UKF required about 100 additional frames in order to converge to the correct solution. Although the UKF will occasionally chose the correct mode, this kind of event happened often enough in the sequence to impair the RMSE performance.

Second, examining only the RMSE plots, it appears that at, 400 particles, our method exhibits similar RMSE to the baseline method. Simultaneously, the RMSE error of the baseline method seems to increase with the number of particles. These

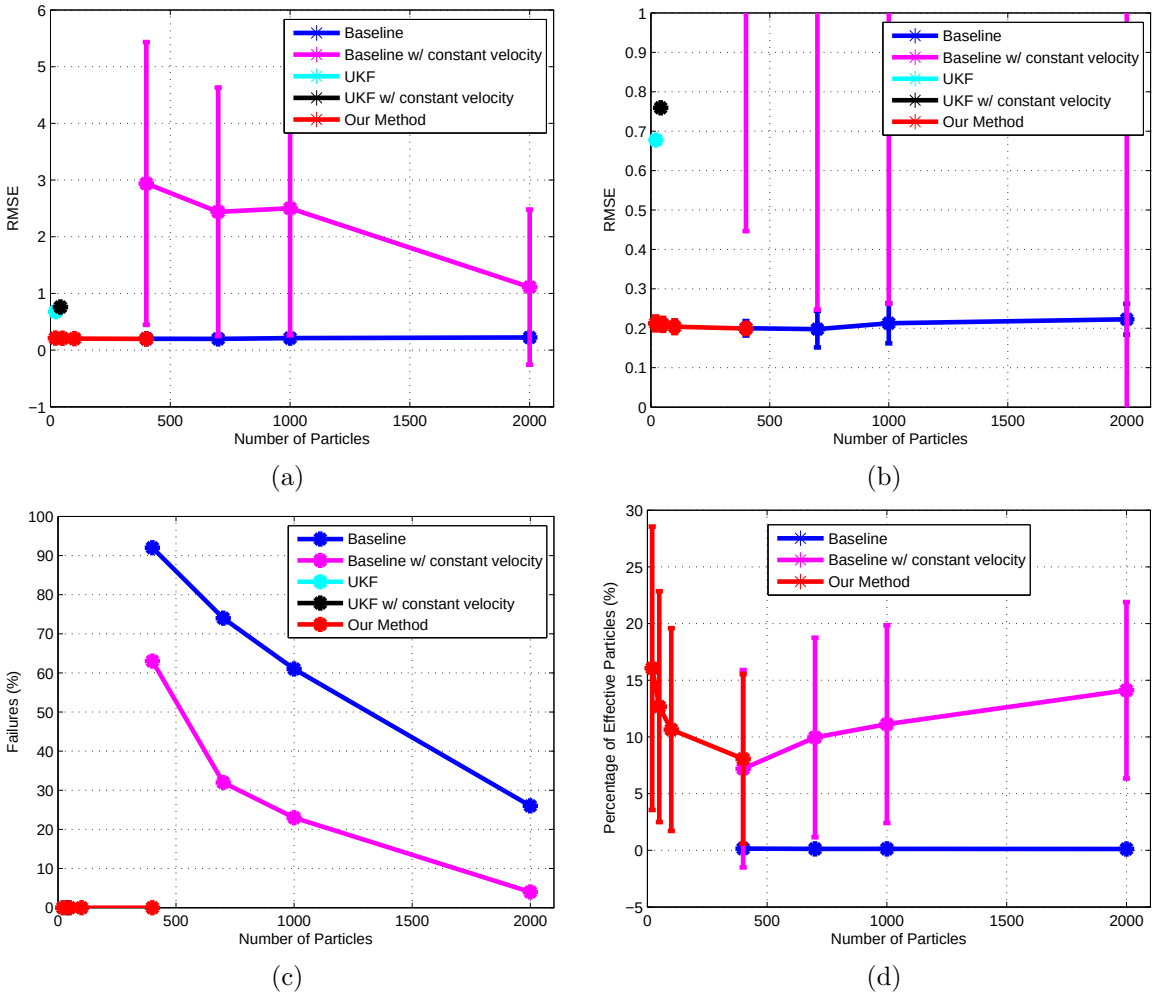


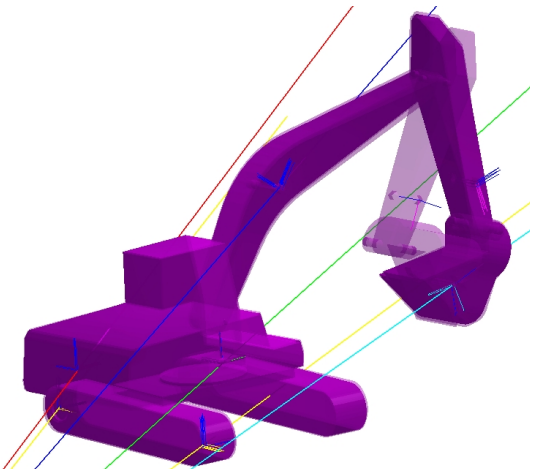
Figure 3-42: These plots show errors, failures, and number of effective particles for the excavator example. The top row shows the sample RMSE plot at different zoom scales.

two surprising results can be explained by examining the failures in Figure 3-42c. The baseline method exhibits over 90% failures at 400 particles. The handful of successes indicate the filter randomly selects lucky particles and happens to find a good solution. As more particles are added, the probability of finding a good solution increases and the number of failures decreases. Despite the occasional lucky performance for the baseline method, our method achieves a 100% success rate for all number of particles while still maintaining low RMSE.

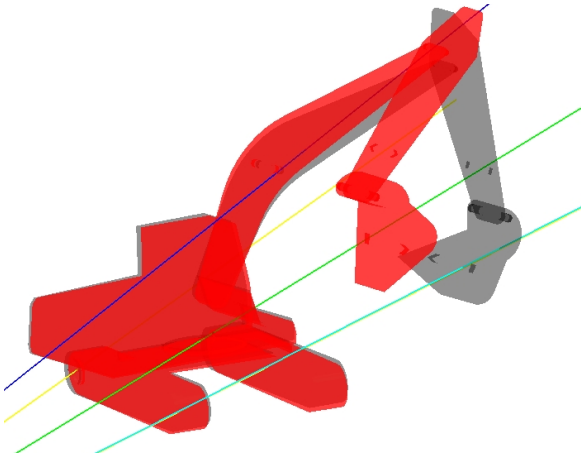
Third, the constant velocity state transition model did not help reduce RMSE for the baseline and UKF filters. Occasionally a constant velocity model can provide a good first order approximation to motion, but this depends on the smoothness of the motion relative to the frame rate. In this scenario, the motion was sufficiently non-linear (or, alternatively, the frame rate was sufficiently low), that a constant velocity assumption caused increased RMSE. This occurred because the excavator would make sudden stops or starts and the constant velocity model would tend to overshoot the true position. We do not evaluate a constant velocity assumption with our method because it requires the Jacobian of the state. This would have required calculating accelerations for the kinematic tree and is beyond the scope of this work. Regardless, the state model is used sparingly by our method, only along singular dimensions, and would not have significantly affected the results.

Generally, we do not expect the kinematic model to be perfect. (Indeed, the model of the excavator used here is approximated from online datasheets.) Imperfections in the model will cause errors in the forward kinematics and Jacobian calculations. In order to characterize the effects of model noise, we distorted our model of the excavator by adding Gaussian noise to the kinematic parameters. In particular, we added zero-mean translational and rotational noise, with standard deviations of 0.05m and 0.05rad, respectively, to all kinematic parameters. We then compared the tracking error of the baseline PF to our method, as shown in Figure 3-44, for a short segment of the dump truck loading. For the baseline and our method, we used a number of particles where RMSE tracking error was similar, 2000 and 100 particles, respectively.

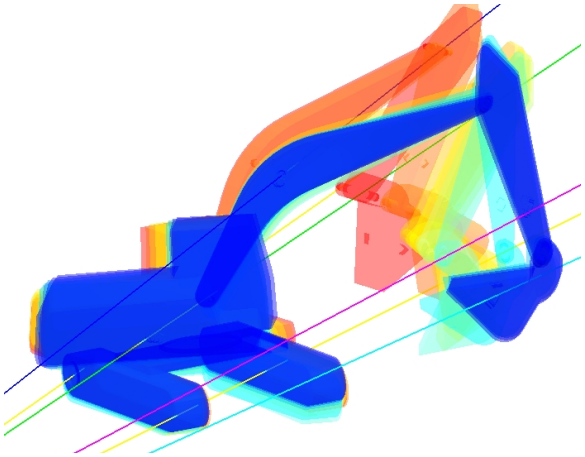
Imperfections in the excavator model cause the tracking error for our method to increase at a greater rate than for the baseline. This is because our method uses the model for particle proposal and weighting, whereas the baseline method uses it for only weighting. The errors in the model cause non-linear effects: small rotational errors are magnified by the large size (several meters) of the cab, boom, and stick. Since our method depends on the model more, it is more susceptible to these imperfections. By contrast, this effect was less noticeable in the model error analysis for the dishwasher (see Figure 3-32). There, only translational errors were added and the structure was relatively small. Although our method is more sensitive to model errors, it still out performed the baseline method with a reasonable (yet imperfect) model (see Figure 3-42) generated from datasheets.



(a) In this frame, the stick is not visible and two possible configurations explain the observations. The purple excavators show all 20 particles at this time step. Notice that the two modes are represented.



(b) The UKF's track (red) is not the correct local minimum and is far from the true configuration (grey).



(c) Having chosen the wrong local minimum (red), the UKF takes over 100 frames to converge to the correct one (blue). Intermediate configurations are shown by color from red to blue.

Figure 3-43: Comparison of the UKF and our method for a frame of the excavator experiment

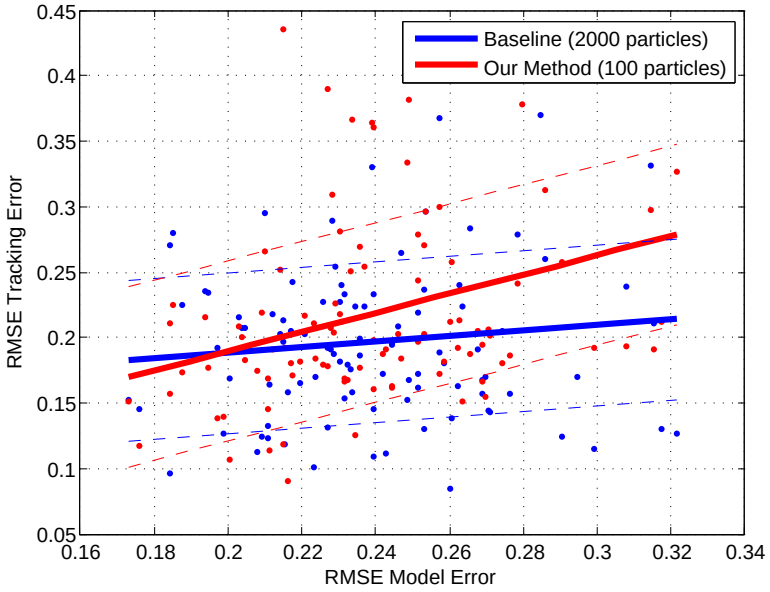


Figure 3-44: We simulated errors in the excavator model by adding random noise to all kinematic parameters. Each point represents a different simulation. The solid lines show a best fit; the dotted lines show one standard deviation.

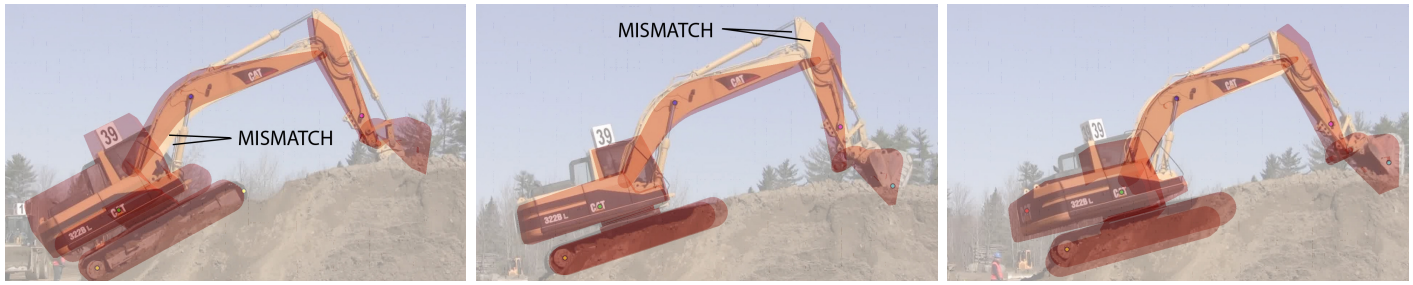
3.5.5.2 Results for Climbing

We also collected video of an excavator climbing a hill, as shown in Figure 3-45, using a hand-held camcorder. This video series differed from the previous dump truck loading scenario because there was camera movement and significant movement of the tracks. In the previous video, the tracks remained stationary. In this sequence, the operator also maneuvered the equipment so that the arm was used to "pull" the excavator up the hill. In Figure 3-45, notice that the bucket (relative to the scene) remains fixed as the excavator climbs the hill. This maneuver highlights one of the difficulties with selecting a state transition model: most models would have assumed relatively stationary tracks and a more mobile bucket.

Ground truth was not reliable for this sequence, because the absence of 3D LIDAR data made it difficult to manually resolve ambiguities. Nevertheless, our method is able to track the excavator well. Figure 3-46 shows three frames (rows) from the sequence for the UKF, baseline, and our method. In each frame, the estimated excavator's pose is projected into the camera frame. When tracked correctly, the excavator should be darkened by the estimated pose. For the first frame of all methods (first column), the bucket is not detected and its position is unobservable.

The UKF again fell into local minimums for the first and second frames. In the first frame, the cab is significantly tilted forward; in the second, the stick/bucket joint is in the wrong position. The baseline method did better at 400 particles, but tended to misalign the tracks. Our method demonstrates good alignment throughout the sequence.

Figure 3-45: The excavator climbs the hill by sinking the bucket into the ground and pulling itself up.



(a) UKF

(b) Baseline

(c) Our Method

Figure 3-46: The excavator's estimated position is projected into the camera frame (red) for three methods. Each column shows one frame from the series. Ideally, the red/darkened virtual excavator should just cover the excavator in the image. For example, the middle frame for the UKF shows a mismatch for the stick and bucket.

3.5.6 Frame rate

In this section, we have demonstrated at least an order-of-magnitude reduction in the number of required particles to track at similar levels of RMS error. This has a corresponding increase in our method's frame rate. Figure 3-47 compares the frame rate for the Dishwasher, PR2, and Excavator examples. In each case, our method exhibited a 4×-8× speed up over the baseline method. Since the frame rate of the particle filter is linearly proportional to the number of particles, we might have expected a 10× improvement. This did not happen, however, because of the discrete approximation required to calculate the particle weights in our method.

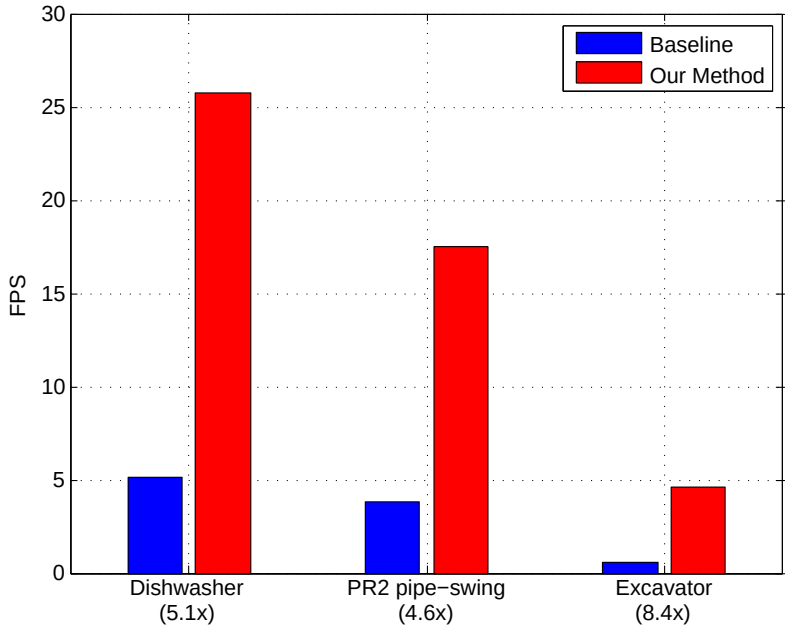


Figure 3-47: The improved particle generation in our method resulted in a 4×-8× speed up over the baseline method.

Chapter 4

Conclusion

This work has focused on the task of articulated pose estimation. We first examined a relatively simple example: an articulated chain with a single segment and no joints. When two sensors are attached, recovering the rigid body transform associated with the segment becomes a particularly common problem in robotics: extrinsic sensor calibration. Second, we focused on tracking articulated structures with many segments and joints. In some cases, these structures were part of the robot (e.g., in the PR2) task; in others, the structures were not under the robot's control (e.g., the excavator). This tracking ability is useful for the robot to better estimate its own state, or to estimate the state of its environment (e.g., to help navigate a construction site safely).

4.1 Contributions

The calibration work presents a practical method that recovers the 6-DOF rigid body transform between two sensors, from each sensor's observations of its 6-DOF incremental motion. Our contributions include a sensor agnostic, automatic calibration process. We provide a novel method to estimate the calibration without relying on accurate sensor localization or any specific preparation of the environment.

We show observability of the calibrated system, including an analysis of degenerate conditions. Throughout the process we treat observation noise in a principled manner, allowing calculation of a lower bound on the uncertainty of the estimated calibration. The CRLB provides a confidence limit on the estimated parameters which can be used to design robot paths and incorporated into subsequent algorithms. From a practical perspective, we present a means to interpolate observations and covariances from asynchronous sensors. These techniques enable accurate estimation of calibration parameters in both simulated and real experiments.

Additionally, we illustrate the use of a constrained DQ parametrization which greatly simplified the algebraic machinery of degeneracy analysis. Such over-parametrizations are typically avoided in practice, however, because they make it difficult

to perform vector operations (adding, scaling, averaging, etc.), develop noise models, and identify system singularities. We assemble the tools for each required operation, employing the Lie algebra to define local vector operations and a suitable projected Gaussian noise model. We also employ a constrained formulation of the CRLB.

In the articulated object tracking chapter, we describe an algorithm that estimates the base pose and joint positions of an articulated object. The method accepts some observations of the structure's segments and a kinematic description of the object (including the ability to perform forward kinematics, calculate the Jacobian, and check joint limits). It then outputs the articulated structure's configuration, including a 6-DOF base pose and values for the joints.

Unlike previous work which has primarily focused on modeling some specific articulated object, our contribution works on a generic articulated object. We use a particle filter, and demonstrate the advantages of incorporating observations during proposal, while appropriately using the state transition model during singular situations. We perform particle diffusion in the observation space, resulting in a robustness to state parametrization dependency. Whereas the UKF might be appropriate when an articulated object has a uni-modal distribution and the baseline implementation when a good predictive model is available, our method is preferred when tracking a multi-modal distribution when the predictive model is poor. We demonstrate more than an order-of-magnitude reduction in the number of particles over a baseline implementation and compare against a UKF. As a result, we see as much as a factor of 8 increase in the frame rate.

4.2 Future Work

4.2.1 Calibration

The calibration technique uses non-linear least squares to determine the most likely calibration explaining differences in incremental poses. Least squares techniques can suffer when there are large numbers of outliers. The errors from these outliers, when squared, tend to dominate the error term and move the optimization away from the true solution. In our experiments, we found our incremental motion observations reasonably free of outliers. However, in circumstances where this was not the case, additional steps might need to be taken. For example, Random Sample Consensus (RANSAC) might be used to select a subset of the incremental poses, optimize, and then reselect. This technique has the potential to ignore large numbers of outliers.

The work presented also focuses on pairs of sensors. However, the equation describing the optimization between two sensors Equation 2.44 can be naturally expanded to include multiple sensors. This would be useful in situations such as those shown in Figure 2-1 which contain many sensors whose data is to be fused together. In that case, the algorithm would recover the joint calibrations which best explain the incremental motions.

The CRLB gives a lower bound on the uncertainty of an estimate. Interestingly, in the signals community, the CRLB is often used as an error signal to be minimized. In the context of calibration, the CRLB might be indirectly used to navigate the mobile robot. For example, we might path plan using the CRLB as a part of a cost function. This would result in vehicle movements which minimized the uncertainty of the calibration, providing an "automated calibration driving." One challenge, however, is that the CRLB would need to be computed online, and possibly as part of forward simulations during path planning. This would require the computation of the observation Jacobian and the inverse of the observation covariance. For this to be fast and reliable, some trade-off between the number of observations (length of time history) and calibration accuracy would be required.

Another interesting aspect for future work would be to consider degenerate observations. In the articulated object tracking, we represented observations as 6-DOF poses with singular precision matrices. This idea could also be applied to the calibration work. This would allow information from 3-DOF LIDARs to be calibrated against 6-DOF Kinects naturally. Unobservable aspects of the calibration (e.g., the vertical positioning of a LIDAR) would then be identifiable in the CRLB.

4.2.2 Articulated Object Tracking

One assumption in our system is that the association between observations and location on the kinematic structure is known. In other words, we assume the data association problem between observations and the model is solved. We feel this is a reasonable assumption because the kinematic model is known. Therefore, it is reasonable to assume that image detectors, for example, could be trained on that same model. Furthermore, we envision our input observations as the output of relatively high-level detectors (i.e., the output of a TLD tracker) which are more discriminating than low level features (e.g., SIFT features).

On the other hand, some systems may be limited to low-level features (e.g., SIFT) as the input. In these cases, it is unlikely that the association between the dynamically generated features and the model will be known. The problem then would become to not only estimate the pose of the articulated structure but also to estimate the associations between observations and the kinematic model. Data association is a challenging, but well studied, problem. Data clustering and data association filters are commonly used solutions. Multiple hypothesis techniques, similar to our particle filtering approach, would benefit from our technique here, as our method reduces the number of hypotheses which must be maintained for the pose. This reduces the search space which will still contain the DOFs for the data associations.

We have also assumed that the kinematic model of the articulated structure is provided as an input. As discussed in the background section Section 3.2, previous work has focused on automatically recovering the parameters (joint type and link parameters). This previous work would supply the input kinematic model to ours. Future work might attempt to track the pose and recover the model parameters

simultaneously. In some situations, this may be possible. However, the freedom to change the segment parameters, number of joints, joint types, and configuration can easily degenerate into an unobservable system. Without any priors, an infinitely jointed (prismatic and revolute) object could be made to fit any series of observations. As a result, it would seem that strong priors or additional constraints would be required to make this goal attainable.

Our method favors particle proposal using the observation model rather than the state transition model. In our examples, we found this a successful technique since our observations are more reliable than our state transition model. Although the observations may be more reliable in general, sporadic outlying observations still occur. For example, a detector for the excavator's bucket may spuriously fire somewhere else in the image. This is not a problem if the measurement's uncertainty matrix is correct; however, often the covariance is artificially low in these situations. As a result, highly distorted configurations (particles) are proposed. Our method is somewhat robust to this kind of noise, as the state transition model is still used in the OIF weighting. These distorted configurations receive low weight and are likely eliminated in subsequent resampling. Failure can occur, however, if all the particles receive a low weight due to the spurious observation. Future work might consider the possibility of proposing particles both from the state transition model and the observation model. Since our OIF approximation is discrete, the addition of more particles will not affect the algorithm. However, processing time per particle would increase.

As suggested in Figure 3-3, the articulated object tracker forms a sequential processing chain with up-stream detectors. However, knowledge of the object's configuration can be used to guide object detection. Future work might consider "closing the loop" between our method and the observation system. For example, knowledge of the excavator's pose could provide a strong prior for a feature detector attempting to locate the cab, boom, stick, and bucket in an image.

Finally, our method approximates the OIF by sampling in the observation space and then using a Taylor approximation to project onto the state manifold. This works by sampling P random particles from a Gaussian to form $\mathcal{X}$, a discrete approximation to the OIF distribution. An alternative would be to project the sigma points from the Gaussian onto the manifold and then sample from the new "projected" Gaussian. This could prove to require fewer than P particles, because the manifold will have fewer dimensions. Nonetheless, a trade-off study would be required, as it would also distort the projection (by enforcing an unnecessary Gaussian assumption).

Appendix A

Additional Calibration Proofs

A.1 Lie Derivative

We wish to show that the derivative of the logarithm is rotationally invariant for $SE(3)$. In other words, $\nabla_a \left(f(a) \boxminus b\right) = \nabla_a \left(f(a)\right)$ when $\{a, b\} \in \mathbb{H}$. This is the Lie group equivalent of the Euclidean $\frac{\partial}{\partial x}\left(f(x) - y\right) = \frac{\partial}{\partial x} f(x)$ where $\{x, y\} \in \mathbb{R}^n$. Intuitively, this rotational invariance occurs because the gradient is taken in the locally tangent plane of the Lie algebra, which rotates with the group element.

$$\begin{aligned}
\nabla_a \left(f(a) \boxminus b\right) &= \nabla_a \left[-2 \log\left(b^{-1} \circ f(a)\right)\right] && \text{(A.1)}\\
&= \nabla_a \left[-2 \log\left(\overline{B} f(a)\right)\right] && \text{(A.2)}\\
&= \nabla_a \left[-2 \log\left(\overline{B}\right) - 2 \log\left(\mathbf{I} f(a)\right)\right] && \text{(A.3)}\\
&= \nabla_a \left[-2 \log\left(\mathbf{I} f(a)\right)\right] && \text{(A.4)}\\
&= \nabla_a \left(f(a) \boxminus 0\right) && \text{(A.5)}\\
&= \nabla_a \left(f(a)\right) && \text{(A.6)}
\end{aligned}$$

For clarity in Equation A.2, we have used a matrix representation of the DQ such that $\overline{B} f(a) = b^{-1} \circ f(a)$ [53]. $\overline{B}$ is an 8×8 matrix which corresponds to the DQ multiplication by b^{-1}; $\mathbf{I}$ is the 8×8 identity matrix.

A.2 Jacobian Ranks

If A is $D \times E$, B is $E \times F$, $\mathcal{N}(A)$ is the null space of A, $\mathcal{R}(B)$ is the column space of B, and $\dim A$ is the number of vectors in the basis of A, then $\text{rank}(AB) = \text{rank}(B) - \dim\left[\mathcal{N}(A) \cap \mathcal{R}(B)\right]$. Substituting from Equation 2.105,

$$\text{rank}(J_H J_G U) = \text{rank}(J_G U) - \dim\left[\mathcal{N}(J_H) \cap \mathcal{R}(J_G U)\right]$$

Intuitively, this means that if a column of $J_G U$ lies in the null space of J_H, information is lost during the multiplication and the rank of the matrix product is reduced. In

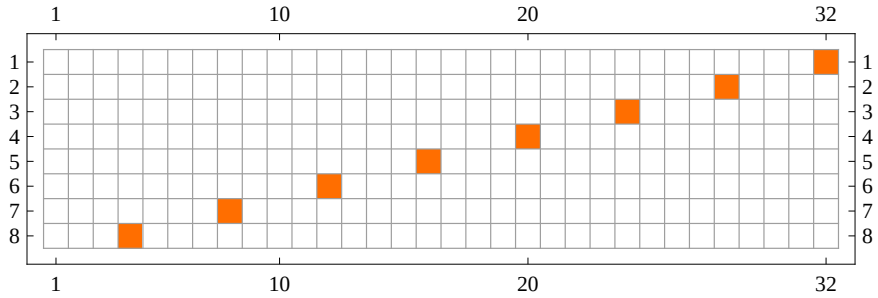


Figure A-1: The matrix $\mathcal{N}(J_H)^T$, depicted here for $N = 2$, reveals $4N$ DOF's corresponding to the constraints of the $2N$ DQ's in z. Blank entries are zero; orange are unity.

order to show that $\text{rank}(J_H J_G U) = \text{rank}(J_G U)$, there are two cases:

1. If $J_G U$ is singular, then $\text{rank}(J_G U) < 6(N+1)$, where N is the number of observations. This implies $\text{rank}(J_H J_G U) < 6(N+1)$. Thus, $J_G U$ is singular implies $J_H J_G U$ is singular.

2. If $J_H J_G U$ is singular, then either $J_G U$ is singular or $\dim[\mathcal{N}(J_H) \cap \mathcal{R}(J_G U)] > 0$.

   - If $J_G U$ is singular, then this is the case above.
   - If $J_G U$ is not singular, then $\text{rank}(J_G U) = 6(N+1)$. The task then becomes to determine $\dim[\mathcal{N}(J_H) \cap \mathcal{R}(J_G U)]$. Since $J_G U$ is full rank, $\mathcal{R}(J_G U)$ is the columns of $J_G U$. Furthermore, there are $4N$ columns in $\mathcal{N}(J_H)$, one for each of the two constraints of the $2N$ DQ's. (Figure A-1 shows $\mathcal{N}(J_H)$ for $N = 2$.) It can be shown that $\text{rank}([J_H, J_G U]) = \text{rank}(J_H) + \text{rank}(J_G U)$. In other words, none of the columns of $\mathcal{N}(J_H)$ will intersect with the columns of $J_G U$. Thus, $\mathcal{N}(J_H) \cap \mathcal{R}(J_G U) = \emptyset$ and $\text{rank}(J_H J_G U) = \text{rank}(J_G U)$. Since $J_G U$ is not singular, $J_H J_G U$ is not singular, which is a contradiction. Only the former possibility remains, and $J_H J_G U$ is singular implies $J_G U$ is singular.

   In conclusion, $J_H J_G U$ is singular if and only if $J_G U$ is singular. Intuitively, this is not a surprising result; the log function is designed to preserve information when mapping between the Lie group and the Lie algebra.

A.3 DQ Expression for g

When expressed using DQ's, g in Equation 2.21, can be expressed relatively simply for the 6-DOF case. Here, g_i is the i-th element of the DQ returned by $g(\cdot)$ and v_i is the i-th element of the DQ v_{ri}.

$$g_0 = v_0 \tag{A.7}$$

$$g_1 = k_0^2 v_1 + k_1^2 v_1 - \left(k_2^2 + k_3^2\right) v_1 + 2k_0 \left(k_3 v_2 - k_2 v_3\right) + 2k_1 \left(k_2 v_2 + k_3 v_3\right) \tag{A.8}$$

$$g_2 = -2k_0 k_3 v_1 + k_0^2 v_2 - k_1^2 v_2 + \left(k_2^2 - k_3^2\right) v_2 + 2k_2 k_3 v_3 + 2k_1 \left(k_2 v_1 + k_0 v_3\right) \tag{A.9}$$

$$g_3 = 2k_0 \left(k_2 v_1 - k_1 v_2\right) + 2k_3 \left(k_1 v_1 + k_2 v_2\right) + \left(k_0^2 - k_1^2 - k_2^2 + k_3^2\right) v_3 \tag{A.10}$$

$$g_4 = 2\left(k_0 k_4 + k_1 k_5 + k_2 k_6 + k_3 k_7\right) v_0 + v_4 \tag{A.11}$$

$$\begin{aligned} g_5 =& 2\left(-k_2 \left(k_6 v_1 - k_5 v_2 + k_4 v_3\right) + k_3 \left(-k_7 v_1 + k_4 v_2 + k_5 v_3\right)\right) + k_0^2 v_5 + k_1^2 v_5 \\ & - \left(k_2^2 + k_3^2\right) v_5 + 2k_0 \left(k_4 v_1 + k_7 v_2 - k_6 v_3 + k_3 v_6 - k_2 v_7\right) \\ & + 2k_1 \left(k_5 v_1 + k_6 v_2 + k_7 v_3 + k_2 v_6 + k_3 v_7\right) \end{aligned} \tag{A.12}$$

$$\begin{aligned} g_6 =& 2k_1 k_6 v_1 - 2k_0 k_7 v_1 + 2k_0 k_4 v_2 - 2k_1 k_5 v_2 + 2k_1 k_4 v_3 + 2k_0 k_5 v_3 \\ & + 2k_2 \left(k_5 v_1 + k_6 v_2 + k_7 v_3 + k_1 v_5\right) + k_0^2 v_6 - k_1^2 v_6 + k_2^2 v_6 - k_3^2 v_6 \\ & + 2k_0 k_1 v_7 - 2k_3 \left(k_4 v_1 + k_7 v_2 - k_6 v_3 + k_0 v_5 - k_2 v_7\right) \end{aligned} \tag{A.13}$$

$$\begin{aligned} g_7 =& 2k_2 \left(k_4 v_1 + k_7 v_2 - k_6 v_3 + k_0 v_5 + k_3 v_6\right) + 2\left(k_1 \left(k_7 v_1 - k_4 v_2 - k_5 v_3\right)\right. \\ & \left. + k_3 \left(k_5 v_1 + k_6 v_2 + k_7 v_3 + k_1 v_5\right) + k_0 \left(k_6 v_1 - k_5 v_2 + k_4 v_3 - k_1 v_6\right)\right) \\ & - k_2^2 v_7 + \left(k_0^2 - k_1^2 + k_3^2\right) v_7 \end{aligned} \tag{A.14}$$

Appendix B

Additional Articulated Object Tracking Proofs

B.1 Minimization on Manifolds

We desire to show that:

$$\underset{\delta_2}{\operatorname{argmin}} \left|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\right|^2 = \underset{\delta_2}{\operatorname{argmin}} \left|(\delta_1 - \delta_2)\right|^2 \tag{B.1}$$

where $x \in \mathbb{H}$ and $\{\delta_1, \delta_2\} \in \mathfrak{h}$. By considering $SE(3)$ operations on a sphere, it can be shown [33] that:

$$\left|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\right| = \operatorname{acos}\left(\cos(\alpha)\,\cos(\beta)\; + \sin(\alpha)\,\sin(\beta)\,\cos(\gamma)\;\right) \tag{B.2}$$

where γ is the angle between δ_1 and δ_2, $\alpha = |\delta_1|$, and $\beta = |\delta_2|$. By definition, $|\delta_1 - \delta_2| \to 0 \Rightarrow \gamma \to 0$. Thus as the right side of Equation B.1 is minimized, $\gamma \to 0$, and the effects on the left side are:

$$\lim_{\gamma \to 0} \left|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\right|^2 \tag{B.3}$$

$$= \lim_{\gamma \to 0} \left|\operatorname{acos}\left(\cos(\alpha)\,\cos(\beta)\; + \sin(\alpha)\,\sin(\beta)\,\cos(\gamma)\;\right)\right|^2 \tag{B.4}$$

$$= \lim_{\gamma \to 0} \left|\operatorname{acos}\left(\cos(\alpha)\,\cos(\beta)\; + \sin(\alpha)\,\sin(\beta)\;\right)\right|^2 \tag{B.5}$$

$$= \lim_{\gamma \to 0} \left|\operatorname{acos}\left(\cos(\alpha - \beta)\right)\right|^2 \tag{B.6}$$

$$= \lim_{\gamma \to 0} \left|\alpha - \beta\right|^2 \tag{B.7}$$

$$= \lim_{\gamma \to 0} \left||\delta_1| - |\delta_2|\right|^2 \tag{B.8}$$

$$= \lim_{\gamma \to 0} \delta_1^T \delta_1 + \delta_2^T \delta_2 - 2|\delta_1||\delta_2| \tag{B.9}$$

$$= \delta_1^T \delta_1 + \delta_2^T \delta_2 - 2\delta_1^T \delta_2 \tag{B.10}$$

$$= \left|\left(\delta_1 - \delta_2\right)^T \left(\delta_1 - \delta_2\right)\right|^2 \tag{B.11}$$

$$= \left|\delta_1 - \delta_2\right|^2 \tag{B.12}$$

This relies on the dot product $\delta_1^T \delta_2 = |\delta_1||\delta_2| \cos(\gamma)$, and $\gamma \to 0 \Rightarrow \delta_1^T \delta_2 = |\delta_1||\delta_2|$. Thus, the left and right sides of Equation B.1 reach the same value.

Bibliography

[1] Asus. Xtion PRO, Aug. 2013. URL `http://www.asus.com/Multimedia/Xtion_PRO/`.

[2] Y. Bar-Shalom, T. Kirubarajan, and X. Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[3] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[4] J. Blanco, F. Moreno, and J. Gonzalez. A collection of outdoor robotic datasets with centimeter-accuracy ground truth. *Autonomous Robots*, 27:327–351, 2009.

[5] J.-Y. Bouguet. Camera Calibration Toolbox for Matlab, Jul 2013. URL `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[6] N. Boumal, A. Singer, and P.-A. Absil. Robust estimation of rotations from relative measurements by maximum likelihood. Unpublished, 2013.

[7] J. Box. Bias in nonlinear estimation. *Journal of the Royal Statistical Society*, 33:171–201, 1971.

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Mar. 2004.

[9] J. Brookshire and S. Teller. Automatic calibration of multiple coplanar sensors. *Robotics Science and Systems*, 2011.

[10] J. Brookshire and S. Teller. Extrinsic calibration from per-sensor egomotion. In *Robotics Science and Systems*, 2012.

[11] H. Bruyninckx, P. Soetens, and B. Koninckx. The real-time motion control core of the Orocos project. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 2766 – 2771, 2003.

[12] S. R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Technical report, IEEE Journal of Robotics and Automation, 2004.

[13] R. Cabido, D. Concha, J. Pantrigo, and A. Montemayor. High speed articulated object tracking using GPUs: A particle filter approach. In *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, pages 757–762, 2009.

[14] A. Censi, L. Marchionni, and G. Oriolo. Simultaneous maximum-likelihood calibration of odometry and sensor parameters. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[15] S. Ceriani, G. Fontana, A. Giusti, et al. Rawseeds ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, 27:353–371, 2009.

[16] T.-J. Cham and J. Rehg. A multiple hypothesis approach to figure tracking. In *Computer Vision and Pattern Recognition*, volume 2, page 244, 1999.

[17] H. H. Chen. A screw motion approach to uniqueness analysis of head-eye geometry. In *Computer Vision and Pattern Recognition*, Jun 1991.

[18] S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *Robotics and Automation, IEEE Transactions on*, 13(3):398–410, 1997.

[19] G. Chirikjian. *Stochastic Models, Information Theory, and Lie Groups*, volume 2. Birkhuser Boston, 2010.

[20] A. Comport, E. Marchand, and F. Chaumette. Kinematic sets for real-time robust articulated object tracking. *Image and Vision Computing*, 25(3):374–391, 2007.

[21] K. Daniilidis. Hand-eye calibration using dual quaternions. *International Journal of Robotics Research*, 18, 1998.

[22] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[23] J. Deutscher, A. Blake, and I. D. Reid. Articulated body motion capture by annealed particle filtering. In *Computer Vision and Pattern Recognition*, pages 126–133, 2000.

[24] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[25] I. Dryden, A. Koloydenko, and D. Zhou. Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging. *The Annals of Applied Statistics*, 3(3):1102–1123, 2009.

[26] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Applied Machine Vision*, 9 (5-6), Mar. 1997.

[27] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition*, Jun 2008.

[28] C. Gao and J. Spletzer. On-line calibration of multiple LIDARs on a mobile vehicle platform. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 279–284, 2010.

[29] V. Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *Computer Vision and Pattern Recognition*, 2004.

[30] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[31] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3D. In *Intelligent Robots and Systems*, Nov 2007.

[32] B. Hamner, S. C. Koterba, J. Shi, R. Simmons, and S. Singh. An autonomous mobile manipulator for assembly tasks. *Autonomous Robots*, 28(1), January 2010.

[33] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 2011.

[34] B. Horn and K. Ikeuchi. The mechanical manipulation of randomly oriented parts. *Scientific American*, 251(2):100–111, 1984.

[35] A. Huang, A. Bachrach, P. Henry, et al. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *International Symposium of Robotics Research*, Aug 2011.

[36] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis. Analysis and improvement of the consistency of extended kalman filter based slam. In *ICRA*, May 2008.

[37] M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29, 1998.

[38] A. Jain and C. Kemp. Pulling open doors and drawers: Coordinating an omnidirectional base and a compliant arm with equilibrium point control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2010.

[39] E. Jones and S. Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *International Journal of Robotics Research*, Oct 2010.

[40] S. Julier. The scaled unscented transformation. In *IEEE American Control Conference*, volume 6, pages 4555–4559, 2002.

[41] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. *International Conference on Pattern Recognition*, 2010.

[42] K. Kanatani. *Group Theoretical Methods in Image Understanding*. Springer-Verlag New York, Inc., 1990.

[43] D. Katz, M. Kazemi, J. Bagnell, and A. Stentz. Interactive segmentation, tracking, and kinematic modeling of unknown articulated objects. Technical Report CMU-RI-TR-12-06, Robotics Institute, March 2012.

[44] L. Kavan, S. Collins, J. Zara, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. In *ACM Transactions on Graphics*, volume 27. ACM Press, 2008.

[45] J. Kelly and G. Sukhatme. Visual-inertial simultaneous localization, mapping and sensor-to-sensor self-calibration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 360–368, 2009.

[46] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[47] J. Leonard, J. P. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.

[48] J. Levinson and S. Thrun. Unsupervised calibration for multi-beam lasers. In *International Symposium on Experimental Robotics*, 2010.

[49] J. Levinson and S. Thrun. Automatic online calibration of cameras and lasers. In *Robotics Science and Systems*, 2013.

[50] S. Lutta, K. Tsunoda, K. Gehner, and R. Markovic. Gesture keyboarding. Patent US 0199228, Aug 2010.

[51] W. Maddern, A. Harrison, and P. Newman. Lost in translation (and rotation): Fast extrinsic calibration for 2D and 3D LIDARs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Minnesota, USA, May 2012.

[52] A. Martinelli, D. Scaramuzza, and R. Siegwart. Automatic self-calibration of a vision system during robot motion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2006.

[53] J. McCarthy. *An Introduction to Theoretical Kinematics*. MIT Press, 1990.

[54] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, et al.P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. Rusu, B. Marthi, G. Bradski, K. Konolige, B. Gerkey, and E. Berger. Autonomous door opening and plugging in with a personal robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[55] R. Newcombe, A. Davison, S. Izadi, et al. KinectFusion: real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality*, Oct 2011.

[56] G. Niemeyer. Lecture notes. Willow Garage Controls Lecture, Jul 2012.

[57] E. Olson. AprilTag: A robust and flexible multi-purpose fiducial system. Technical report, University of Michigan APRIL Laboratory, May 2010.

[58] P. Pan and D. Schonfeld. Adaptive resource allocation in particle filtering for articulated object tracking. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 729–732, 2008.

[59] W. Qu and D. Schonfeld. Real-time decentralized articulated motion analysis and object tracking from videos. *Image Processing, IEEE Transactions on*, 16 (8):2129–2138, 2007.

[60] J. M. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *Computer Vision and Pattern Recognition*, 1995.

[61] J. M. Rehg, D. D. Morris, and T. Kanade. Ambiguities in visual tracking of articulated objects using two- and three-dimensional models. In *International Journal of Robotics Research*, 2003.

[62] RITCHIESpecs. CATERPILLAR 322BL Hydraulic Excavator, July 2013. URL `http://www.ritchiespecs.com/specification?type=&category=Hydraulic+Excavator&make=Caterpillar&model=322B+L&modelid=104005`.

[63] T. Ruhr, J. Sturm, D. Pangercic, M. Beetz, and D. Cremers. A generalized framework for opening doors and drawers in kitchen environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2012.

[64] M. Sanjeev Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.

[65] J. Selig. Exponential and Cayley maps for dual quaternions. *Advances in Applied Clifford Algebras*, 20, 2010.

[66] D. Simon. *Optimal State Estimation*. John Wiley & Sons, 2006.

[67] P. Stoica and B. C. Ng. On the Cramer-Rao bound under parametric constraints. *Signal Processing Letters, IEEE*, 5(7):177–179, Jul 1998.

[68] J. Sturm, C. Stachniss, and W. Burgard. A probabilistic framework for learning kinematic models of articulated objects. *Journal of Artificial Intelligence Research*, 41, Aug 2011.

[69] S. Teller, A. Correa, R. Davis, L. Fletcher, E. Frazzoli, J. Glass, J. How, J. H. Jeon, S. Karaman, B. Luders, N. Roy, T. Sainath, and M. Walter. A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, 2010.

[70] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. MIT Press, 2005.

[71] H. V. Trees. *Detection, Estimation, and Modulation Theory, Part I*. John Wiley & Sons, New York, 1968.

[72] R. Y. Tsai and R. K. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Trans. Robot. Autom.*, 5(3), Jun 1989.

[73] A. Ude. Nonlinear least squares optimisation of unit quaternion functions for pose estimation from corresponding features. In *International Conference on Pattern Recognition*, volume 1, Aug 1998.

[74] R. Urtasun, D. J. Fleet, and P. Fua. 3D people tracking with Gaussian process dynamical models. In *Computer Vision and Pattern Recognition*, pages 238–245, 2006.

[75] Z. Wang and G. Dissanayake. Observability analysis of SLAM using Fisher information matrix. In *International Conference on Control, Automation, Robotics and Vision*, pages 1242–1247, Dec. 2008.

[76] D. Whitney. Resolved motion rate control of manipulators and human prostheses. *Man-Machine Systems, IEEE Transactions on*, 10(2):47–53, 1969.

[77] Willow Garage. Personal Robot PR2, June 2013. URL `https://www.willowgarage.com/pages/pr2/specs`.

[78] Willow Garage. Unified Robot Description Format (URDF), May 2013. URL `http://www.ros.org/wiki/urdf`.

[79] M. Zefran and V. Kumar. Interpolation schemes for rigid body motions. In *Computer-Aided Design*, volume 30, 1998.

[80] M. Zefran and V. Kumar. Two methods for interpolating rigid body motions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 2922–2927, may 1998.

[81] J. Ziegler, K. Nickel, and R. Stiefelhagen. Tracking of the articulated upper body on multi-view stereo image sequences. In *Computer Vision and Pattern Recognition*, pages 774–781, 2006.