Planning and Localization Using Contacts

Bradley L. Saund

May 2017

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

*Submitted in partial fulfillment of the requirements
for the degree of Masters of Computer Science.*

Keywords:

Abstract

A short summary.

Acknowledgments

My advisor is cool.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contacting the world can provide stability, support, and sensory information. Humans benefit from touching the world in situations from blindly detecting an object in the back of the refrigerator, to leaning on a stair railing. In the recent DARPA robotics challenge top robotics research teams from around the world submitted robots to attempt an obstacle course of challenges. Many of these robots fell on the stairs. Not a single robot used the railing [**?**]. Even drunk people know to use railings, so there is plenty of room for improvement in robotics.

However, in robotics there are good reasons to avoid contact with the environment. Hitting the world can provide large forces that destabilize or break the robot. Relying on contacts can be dangerous as slight alterations in the world or errors in robot position may drastically alter the contact forces. Instead of firmly grasping a handle, a few centimeters of error have caused robots to embarrassingly grasp air and fall over [**?**]. Even when working entirely in simulation contacts between rigid bodies cause problems for physics engines.

Contacts introduce non-smooth discontinuities, which presents a challenge for many tools in the roboticists arsenal. The benefits of contacts occur on a measure-zero manifold in the robot's configuration space. For example, hand rests comfortably on a table at a specific configuration of shoulder, elbow, and wrist positions. A slight extension of the elbow while fixing all other joints leads to the hand puncturing the table, while a slight contractions pulls the hand into the air and the table might as well not exist. Random sampling and gradient decent are two

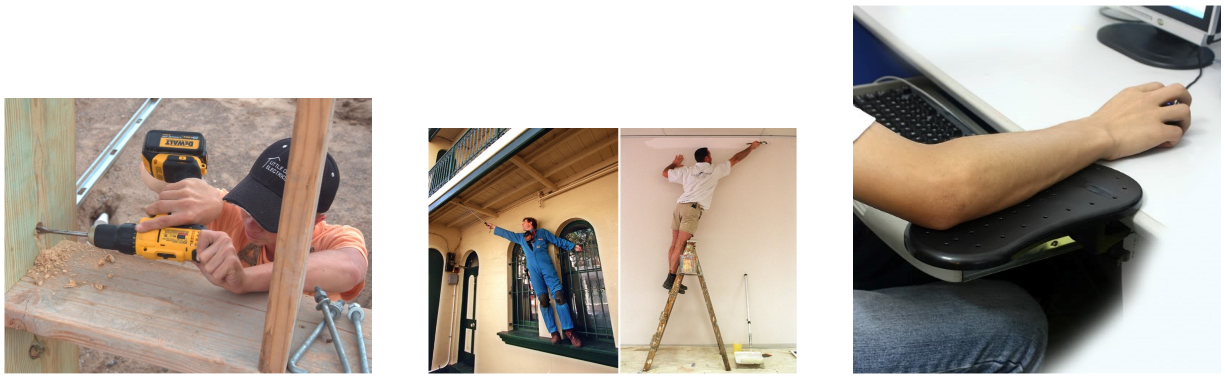


Figure 1.1: Humans using contact to improve reach, accuracy, and stability

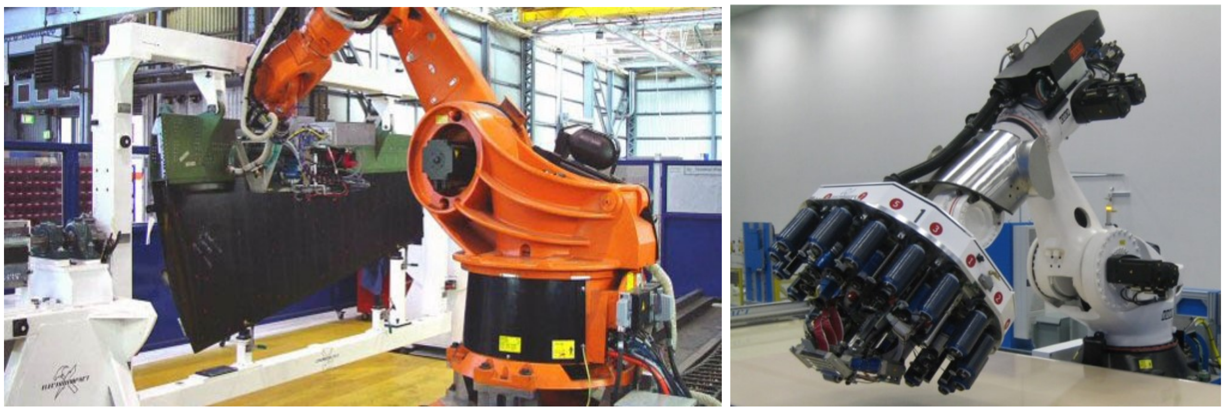


Figure 1.2: Robots working on the outside of parts held firmly in jigs

techniques that appear in the solution to many robotics challenges, but both have difficulty with these measure-zero contact manifolds. This thesis explores and extends techniques of sampling based and gradient methods to the problems of localization and planning, with the addition of contact forces and measurements.

1.1 Motivation

My personal motivation for the topics covered in this thesis come, in part, from my past experience as a robotics engineering building robots that build airplanes. Currently, large robotic arms equipped with expensive, specialized end effectors perform a variety of tasks for aerospace manufacturing, including drilling holes, inserting fasteners, milling, carbon fiber placement and layup, and sealant application. While there are many potential directions of research to improve these already impressive machines, two striking issues are addressed by my work.

1.1.1 Jigs are More Expensive Than Robots

Robots that perform one type of task on one section of an airplane can cost millions of dollars. However the jig that holds the airplane section while the robot works can cost more than the robot. These jigs may have more actuators and tighter tolarances than the robotic system. When the robot begins work on a new section, before ever making a measurement the jig has already located the part to within a few inches. A few scripted measurement with a probe, programmed by an engineer, are sufficient to fully localize the part to within the needed tolerances. When amortized over many airplanes the per-part cost drops. However, this process is inflexible and poorly suited to low-rate manufacturing.

If the robot could localize the part from a wide range of part configurations, and if the robot could reason about interal assembly tolerances then jigs could be make more cheaply, robots would become more versitile, less part-specific programming would be required, and machining accuracy could improve.

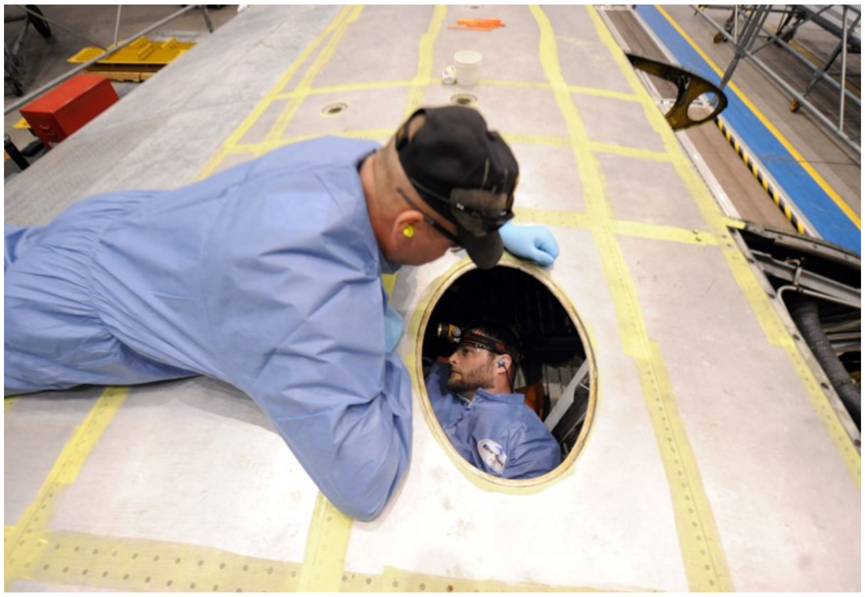


Figure 1.3: Workers crawling in the confined spaces of an airplane wing

1.1.2 Robots Stay on the Outside

Large robotic arms cannot fit in the convined spaces of an aircraft. While a team of people may include workers on both the inside and outside, large robotcs are limited to just the outside.

1.2 Contribution

1.3 Thesis Outline

Chapter 2

Localization using Contacts

Many robotic tasks require precisely localizing an object, for which tactile sensing is an appealing sensing modality. As motivation, consider touch localization of a partially manufactured part that requires additional machining operations. In order to handle objects with complex shape, prior information of the object shape is used, and most previous work assumes that the geometry (CAD) model will match the real object exactly.

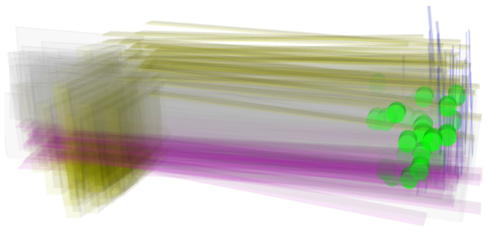
However, during the manufacturing and assembly processes there are tolerances between different sections of the assembly. A *datum* is defined as a geometric constraint within the object that is used as the reference to define the location of one section of the part with respect to another section. The tolerance is the allowed deviation of the actual manufactured dimensions from the nominal designed dimensions. It is assumed a part can be divided into precisely manufactured sections, and the introduced method focuses on handling errors due to imprecise machining over large distances and non-critical components, as well as assembly tolerances.

The introduction of tolerance increases the degrees of freedom (DOFs) of the system, as prior to measurement the true dimensions of the full part are unknown. These internal DOFs can be modeled as transformations with uncertainty between sections of the object. For objects with internal tolerances, perfectly localizing a single datum will not necessarily reduce the uncertainty of the full system sufficiently to perform the desired task. On the other hand, it is usually only necessary to localize a subset of the sections of an object.

In this localization problem, the task is to estimate the pose of a goal feature given multiple measurements obtained through probing. These probing measurements are modeled as a Markov process, where each measurement corresponds to a single action/observation pair. This model eliminates the need to store all of the past measurements. A particle filter numerically stores and updates the belief [**?**].

Figure 2.1 shows a visualization of the initial (2.1a) and final (2.1b) beliefs of the poses of the sections of the object. The task is to drill a hole, shown as a green cylinder, at a specific location defined by offsets from other sections. Internal tolerances prevent simply treating the entire system as a rigid body.

This chapter first considers rigid-body particle filtering for high-precision localization (section 2.2), which overcomes the particle starvation problem and serves as the basis for this paper [**?**]. The datum-based particle filter is then introduced to generalize this rigid-body approach for objects with coupled rigid sections (section 2.3). Two related but different approaches are


(a) Initial belief

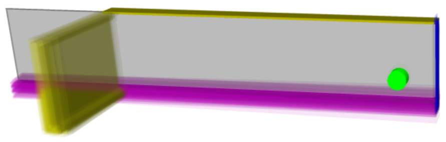

(b) Belief after localization of goal feature (green cylinder)

Figure 2.1: Visualization of the belief of the pose of all sections of the part
(a): The prior belief of the poses before localization. The uncertainty of the goal feature is too high to perform the task.
(b): The belief of the poses after performing measurements to localize the goal feature. The pose of the goal feature is now known well enough to perform the task. Although the bottom edge (purple) and perpendicular section still have noticeable error, precise localization of these features is not needed.

proposed for this problem. The first approach maintains a single particle filter system that stores the full joint distribution of the coupled datums, while the second approach simplifies the relationships by assuming independence between the distribution of the internal transformations and the pose of the sections, and models the system using separate coupled particle filters.

This chapter then generalizes the technique of choosing informative measurement actions to accommodate objects with coupled sections. To achieve this, many potential measurement actions are sampled and the action with the highest expected information gain is selected. Fully predicting the information gain over the continuous belief is computationally expensive, so similar to other approaches [**?**] this work involves a fast approximation for information gain that takes advantage of the discretized belief from the particle filter (section 2.4). This chapter concludes with demonstrations of the method on a simulated part (section 2.5).

2.1 Current Methods for Localizing Objects Using Contact Sensors

Recently, there have been a variety of approaches that allow robots to localize objects solely with contact sensors. Different contact sensors have been explored and developed, including basic binary sensors, 6-axis force and torque sensors [**?**], soft tactile sensor arrays [**?**], and bio-inspired fingertips [**?**]. Localization with laser sensors has also been used in the high-precision CNC localization, where a 3D point cloud is acquired in order to estimate the transformation between the actual and planned pose [**?**]. The localization approach presented here can be generalized to these sensors which can distinguish between contact and free-of-contact states.

Particle filters have been widely used and developed since their introduction. Unlike some other Bayesian estimation approaches such as Kalman filters [**?**], extended Kalman filters [**?**] and unscented Kalman filters [**?**], particle filters can easily model non-Gaussian and multi-modal probability distribution. For touch localization, contact sensors yield a highly non-linear measurement model, and the belief can frequently become multi-modal when multiple configurations are all consistent with the measurement. These properties make particle filter a popular approach for touch localization tasks.

However, particle filters will experience *particle starvation* for measurements with very low uncertainty and objects with a high dimensional configuration space [**?**]. Koval introduced the Manifold Particle Filter to address this issue by implementing different sampling and weighting strategies compared to the traditional particle filters [**?**][**?**]. Instead of sampling particles from the process model and weighting them based on the observation, samples are directly drawn from the contact manifold, given the observation. This provides a fast and robust solution for objects with simple shapes. For complex object geometries, as in this case, direct, efficient sampling from the contact manifold becomes difficult.

Petrovskaya tackles particle starvation during touch localization by combined Monte Carlo approaches with annealing as a smoothing technique [**?**]. She introduced Scaling Series algorithms for 6-DOF global tactile localization in both full-constrained and under-constrained scenarios to overcome particle starvation by adjusting particle density depending on the complexity of the posterior. Multiple iterations through the measurement data are used and the precision of the belief is scaled from low to high in order to avoid unnecessarily precise estimates in unlikely regions of belief space.

All of the work mentioned so far assumes the prior information of the object geometry matches the real piece perfectly. Hebert et al. [**?**] solve a touch localization problem using a Bayes filter where an object may have additional unknown parameters describing the shape, such as a screw driver with an unknown length handle. Measurement actions are selected using joint information gain over the object's pose and these internal parameters.

2.2 Rigid-Body Object Localization

The datum-based particle filter described later relies heavily on the framework and methods developed for rigid-body localization developed in [**?**], and this section provides an overview of the relevant details. The task is to determine the pose of an object by choosing and performing touch measurements, given the geometry of the object and prior belief over the distribution of poses.

The geometry of the object to be localized is stored in a STL file using a triangular mesh, defined in the *part frame*. The pose of the object can then be defined as the transformation between this *part frame* and the *world frame* of the workspace. The object is assumed to be fixed in the workspace during measurement and localization, thus the configuration will not change during the localization process.

In order to estimate the true distribution of the pose, each particle in the particle filter represents a single potential pose ${}^{i}x \in SE(3)$ of the object. For a rigid body, the pose includes both translational dimensions (x, y, z) and rotational Euler angles (α, β, γ). The state is a 6D vector $(x, y, z, \alpha, \beta, \gamma)$ in the configuration space. The particle filter updates based on a set of measurements $M_t = \{m_1, ..., m_t\}$ made by the robot directly on the object.

2.2.1 Measurement Model

A measurement action, $\mathcal{M}$ is defined by a start point $\mathcal{A}_p$ for the probe and a linear trajectory vector $\mathcal{A}_v$ both in $\mathbb{R}^3$. The measurement value m is the distance the probe travels in the direction of $\mathcal{A}_v$ until contact is made. The point of contact can then be recovered by $A_p + m\ \frac{\mathcal{A}_v}{||\mathcal{A}_v||}$. The entire information obtained from the measurement t is $z_t = \{\mathcal{M}_t, m_t\}$. Measurement error exists due to sensor error and robot uncertainty.

2.2.2 Problems with the standard particle filter

A common method of updating particles based on a measurement is importance sampling [**?**]. In importance sampling each particle is weighted by the probability of the measurement conditioned on the state that particle represents. This is usually followed by resampling, where particles are redrawn from the set of weighted particles with probability proportional to their weights. The effectiveness of importance sampling relies on the existence of multiple particles consistent with the measurement, such that inconsistent particles will have low weights and be unlikely to be resampled, but a sufficient number of particles will be resampled to model the true belief of the state.

Importance sampling tends to break down in situations with accurate measurements and low densities of particles. This is because when a measurement is consistent with the true prior belief yet no particles are consistent with the measurement, a situation called *particle starvation*, resampling will yield a set of particles that does not model the true posterior belief. A more accurate sensor measurement is consistent with a smaller volume of state space, thus a higher density of particles is required. For higher dimensional state spaces and more accurate sensors the number of particles required becomes prohibitively computationally expensive. This leads to the counter-intuitive result that particle filters tend to perform worse as measurement accuracy increases [**?**].

2.2.3 Rejection Sampling Method

An alternate approach is to use rejection sampling. Rejection sampling does not require a high density of particles to avoid particle starvation and failure of rejection sampling is far easier to notice and resolve. Most importantly, we can increase the limits of state dimensionality and measurement accuracy that can be handled efficiently.

Rejection sampling generates independent samples from a density f by sampling from a different distribution g. A constant M is determined such that $f(x) \leq Mg(x)\ \forall\ x$. A sample x^* drawn from $g(x)$ is accepted with probability $f(x^*)/Mg(x^*)$ and rejected otherwise. The process is repeated until the desired number of samples has been accepted. We wish to sample particles from the posterior $bel(x_{t+1})$ but cannot do so directly. Instead we sample from our continuous prior belief $bel(x)$ and possibly reject based on the measurement.

Broad Particles: We first reconstruct the continuous prior belief by broadening each of the particles. We apply a *Gaussian kernel* to the particles, with the kernel covariance proportional to the covariance of the particle states. This reduces particle starvation, as even if no prior particle is consistent with the measurement, the continuous belief generated fills in the gaps between particles.

2.2.4 Fast Evaluation of Sampled States

States sampled from this continuous prior are then rejected if they are inconsistent with the measurement. While the formulation of rejection sampling allows us to model complicated measurement error, we implement a binary measurement model. We reject all sampled particles where the measured point is sufficiently far from all faces of the object. We define "sufficiently far" as more than 3 standard deviations of the sensor measurement noise. As a low uncertainty measurement will accept only a thin manifold in state space, the probability of sampling a particle consistent with the measurement may be low, and a lot of sampling may be required, therefore we desire the rejection process to be fast.

To reduce the computational cost per sampled state we use discretized space, known as a *distance field* [?], to precompute and cache the minimum unsigned distance $D_f(p)$ from point p in voxelized space to the object surface $\partial S \subseteq \mathbb{R}^3$:

$$D_f(p) = \min_{q \in \mathbb{R}^3} (||p - q|| + f(q)) \tag{2.1}$$

$$f(q) = \begin{cases} 0, & \text{if } q \in \partial S \\ \infty & \text{otherwise.} \end{cases} \tag{2.2}$$

As the object is fixed during the localization process, voxelization can be done for the entire piece based on the given CAD mesh model in the precomputation step.

Voxelization: Voxelization is the key part to transform the mesh model to axis-aligned discretized space, which can be stored and accessed easily as a standard array. The array form of the CAD model can greatly facilitate the computation of the distance field, as described below. Each voxel is assumed to be a cube in 3D space. A fast 3D Triangle-Box Overlap method [?] is used to label the voxels that overlap the mesh triangles of the object surface. The voxel map is then mapped to a binary-valued 3D array $f(q)$, where each value is either 0 or ∞ depending on whether the corresponding voxel overlaps the object surface.

Voxelized Distance Field: The computation of distance field $D_f(p)$ takes the input of the computed binary array $f(q)$ (Eq. 2.2), and a linear-time algorithm for 3D distance field construction [?] is then used. The resulting distance field is also stored in an array for constant time access during the evaluation of sampled states.

Fast Evaluation: Different configurations result in different poses of the object in the workspace, which makes it difficult to compute the distance field directly in the world frame. Instead, the computation of the voxel map and distance field is relative to the object frame, where the object is assumed fixed during the entire localization. Each measurement M_t in the workspace is then transformed into the part frame, where the transform $T(x_{t+1})$ comes from the pose of the sampled state x_{t+1}. Therefore, by transforming back to the object frame, all measurements on this same object can share the same distance field, where the minimal unsigned distance $dist_u(M_t, S)$ between each measurement M_t and the object $S(x_{t+1})$ can be obtained directly:

$$dist_u(M_t, S(x_{t+1})) = D_f(T(x_{t+1})^{-1} M_t) \tag{2.3}$$

The signed distance $dist(M_t, S(x_t))$ between the probe and the object can be obtained from the unsigned distance, as shown in Eq. 2.4, by checking whether the voxel is inside or outside of the object. For the manifold shape object, ray-casting is applied from the corresponding voxel in a certain direction: the voxel is inside of the object only if the number of intersections between the ray and mesh is odd

$$dist(M_t, S(x_t) = \begin{cases} dist_u(M_t, S(x_t) - r_p, & \text{if } M_t \notin S \\ -dist_u(M_t, S(x_t) - r_p, & \text{otherwise.} \end{cases} \tag{2.4}$$

The unsigned distance $dist(M_t, S)$ is always 0 in the ideal case, however, when evaluating a sampled state, only those that satisfy $|dist(M_t, S)| > T_d$ will be rejected, where T_d is the tolerance selected according to the measurement uncertainty of the touch probe and the robot. If the distance is within the tolerance, ray-casting is then applied to check intersections from the start point $\mathcal{A}_p$ along the path vector $\mathcal{A}_v$ in order to determine whether the path is free of collision with other parts of the object.

When the measurement is very accurate, in order to sample enough particles from the prior belief, a large number of states will get rejected, which makes the ray-casting for all sampled states computationally expensive. Instead, early rejection is applied using a greater distance $dist_u(M_t, S) + r_p$ before the computation of signed distance.

2.2.5 Improvements on the Particle Filter

Adaptive Voxelization: When the workspace is large enough, it will not be feasible to compute a distance transform for the entire workspace while maintaining a sufficiently small voxel size due to the huge memory requirements. Instead, the distance field for each measurement is generated prior to the update. As the belief of the object's pose is represented as a distribution, the range of the distance field DF for each measurement M_t is selected so that:

$$\xi < \int_{T(x_t)^{-1} M_t \in DF} bel(x_t) dx \tag{2.5}$$

The voxel size is adjusted accordingly while keeping the number of voxels fixed. When there is large uncertainty the voxels will become larger to improve the sample speed. When the uncertainty is low the precision is increased.

Adaptive Bandwidth: The selection of the Gaussian kernel bandwidth h is important during the sampling process. A larger bandwidth is needed when the variance of $bel(x_t)$ is large for a fixed number of particles; otherwise a smaller bandwidth is preferred to avoid over-smoothing. Silvermans rule of thumb estimator [**?**] is used to dynamically adjust the bandwidth: $h(t) = (\frac{4}{(d+2)n})^{1/(d+4)} \hat{\sigma}_t$, where $\hat{\sigma}_t$ is the standard deviation of the sampled states and number of dimensions $d = 6$.

Adaptive Sample Size: The number of particles is determined by *Kullback-Leibler divergence* (KL-divergence) which measures the difference between the sample-based maximum likelihood estimate $\hat{p}$ and the true distribution p:

$$D_{KL}(\hat{p}, p) = \sum_x \hat{p}(x) log \frac{\hat{p}(x)}{p(x)}. \tag{2.6}$$

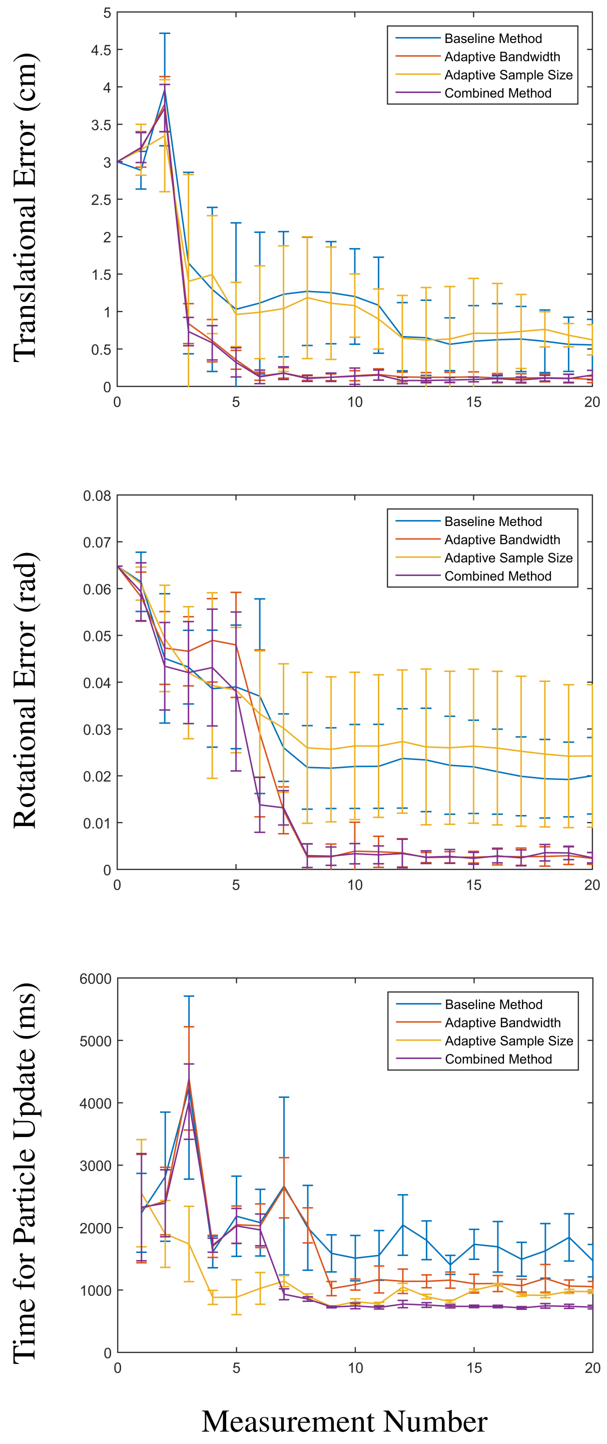


Figure 2.2: Comparison of the time and accuracy of the particle filter update step when using Adaptive Bandwidth and Adaptive Sample Size

Suppose that the true distribution is given by a discrete multinomial distribution with k different bins, it can be shown that with probability $1-\delta$, the KL-divergence is less than or equal to ϵ when the sample size n is given by [**?**]:

$$n = \frac{1}{2\epsilon}\chi^2_{k-1,1-\delta} \tag{2.7}$$

$$\approx \frac{k-1}{2\epsilon}\left(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right)^3 \tag{2.8}$$

where $\chi^2_{k-1,1-\delta}$ is the upper $1-\delta$ quantile of χ^2-distribution with $k-1$ degrees of freedom, and $z_{1-\delta}$ is the upper $1-\delta$ quantile of the normal distribution.

Therefore, the number of particles n can be adjusted according to the number k of bins with support, as shown in Eq. 2.3. The bins are implemented as a multidimensional grid with fixed size in the configuration space. During sampling, the number k of occupied bins is counted whenever the newly sampled state falls into an empty bin. The current sample size is counted and each increment of k will result in an update of desired sample size n. When the actual number of particles reaches the desired value or a predefined maximal limit, whichever is smaller, the sampling process finishes.

Figure 2.2 shows the comparison between the particle filters with different improvements that are mentioned above. The simulation uses a mesh model with 39444 triangles. The set of measurements used for each method is fixed and predefined. The baseline method represents the particle filter that only implements adaptive voxelization while using fixed kernel bandwidth $h = 0.0035$ and fixed number of particles ($n = 500$). Adaptive Bandwidth improves the convergence accuracy significantly compared to the baseline method, while Adaptive Sample Size generally leads to poorer convergence with faster speed. The combined method applies all of the improvements above. It can achieve both faster and more accurate convergence compared to the other methods.

2.3 Datum Based Particle Filter

The particle filter localization method presented above assumes that the object matches its CAD model exactly. However, this is usually not the case, due to tolerances in the manufacturing and assembly processes. To handle manufacturing deviations, features on parts are not located with respect to the part frame, but with respect to datums, (edges, surfaces, and holes) on the actual "as built" part. Incorporating the notion of datums, and their relationships, adds complexity because the relationship between the datum and the CAD model contains uncertainty. Thus, measuring one section of the assembly provides only uncertain updates to other sections, dependent on the specified tolerances. The following formulation treats these as semi-rigid parts, where each complete part is composed of rigid *sections*, coupled through a probabilistic distribution of transformations connecting the section frames. The datum based particle filter is introduced to allow updates on the belief of all sections of a part using the prior distribution of coupling transformations, and a measurement on a single section.

2.3.1 Datum Representation

The formulation introduced here treats the overall part as composed of separate, known sections. The problem is to precisely localize some feature which cannot be measured *directly* (e.g. a location to drill a hole) with respect to given datums (other sections). To localize the goal feature, certain datums must be localized in certain dimensions. For instance, Figure 2.3a shows a hole feature referenced to the top and right edges datums of the part. The true part configuration is shown in gray in 2.3b and 2.3c. In this example, it is necessary to localize the top edge's vertical position and orientation, but not its in-page or horizontal position. Similarly, the right edge only must be localized horizontally.

Two approaches are now introduced; the first explicitly represents the joint probability distribution between the sections, and the second stores separate, independent probability distributions for each section. Figures 2.1 and 2.3 visualize the independent-state particle filter. The full-state particle filter produces similar images.

The rest of this chapter uses the following notation. X_t^k is the set of N particles representing the belief of section k at time step t. Frequently t is omitted when implicit. Each particle is a configuration for a single section $X^k = \{{}^j x^k\}_{j=1}^N$. The omission of k indicates all necessary particles to represent the belief of the part: $X = \{X^k\}$ and $x = \{x^k\}$.

2.3.2 Geometric Relationships

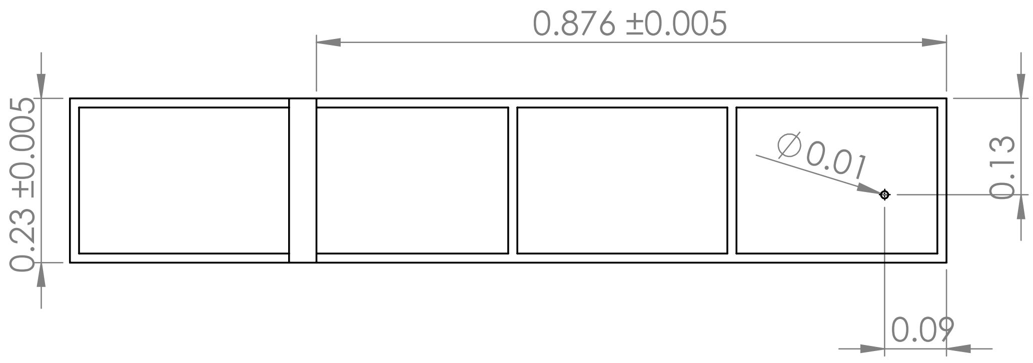
Geometric relationships are defined between two or more part sections. The existence of the tolerance introduces uncertainty to these relationships, which are modeled as a distribution of transformations in the configuration space between the pose of each section. More generally, the conditional probability $p(x_t^k|\mathbf{x_t})$ represents the belief of the pose of section k given the poses of the other sections.

A measurement is made on a single section at each step by a touch probe. Let the measurement on the section k at step t be $\mathcal{M}_t^k$, then the posterior of section k is $p(x_t^k|\mathbf{x_t}, M_t^k)$. In the following algorithms, the section that a measurement contacts is known. This assumption is reasonable if the uncertainty of the prior belief is small compared to the physical size of each section, and measurement are not chosen on the boundary between sections. If this assumption does not hold, localization can be performed for the whole object using the methods of section 2.2 to get better estimate, before considering the object as a combination of coupled sections.

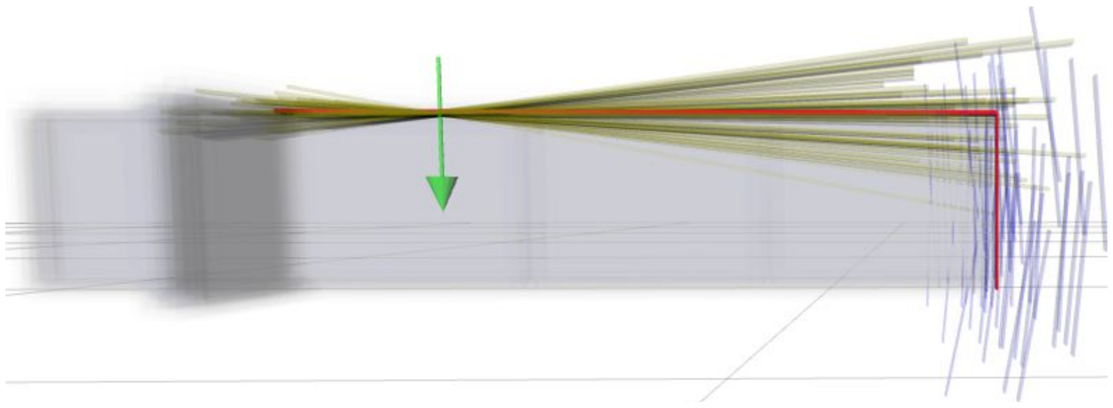
2.3.3 Full-State Representation

The first method presented maintains the full joint probability distribution between sections using a single set of full-state particles: $X_t = \{{}^j x_t\}$. Thus, each particle represents a combination of the poses of all of the sections (see Fig. 2.4), which is drawn from the prior joint distribution $bel(x_0)$.

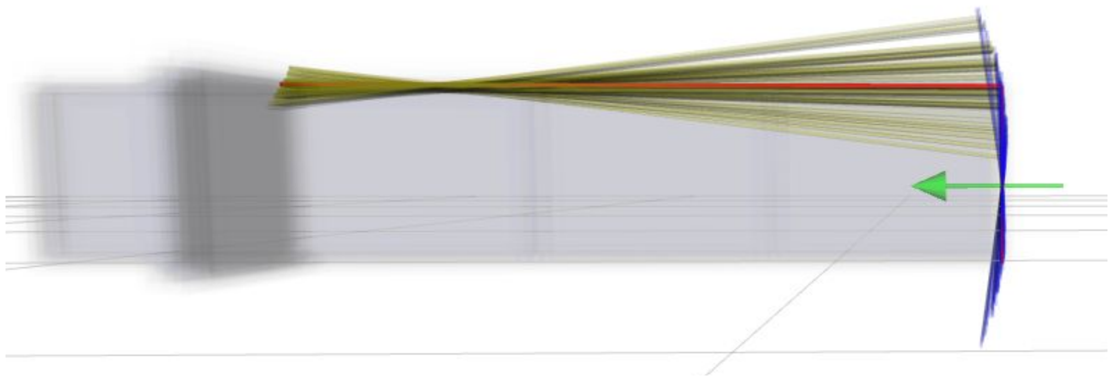
Instead of applying a Gaussian kernel in 6D configuration space as done in section 2.2, the continuous prior is estimated by Gaussian Mixtures in the full $6n$-dimensional configuration space, with the kernel covariance proportional to the covariance of the sampled states. This is



(a) CAD



(b) Update from a measurement on the top datum



(c) Update from another measurement on the right datum

Figure 2.3: Visualization of independent-state particle filter
(a): Side view of the CAD drawing with dimensions (simplified for clarity). This drawing indicates the nominal distance between the top and bottom edge is 0.23m, with a symmetric tolerance of 5mm. This drawing also defines a hole with a 1cm diameter, and the top edge as its vertical datum and side edge as its horizontal datum.
(b): The beliefs of the top (green) and right (blue) edges of the part are shown. The true part location is shown in gray. The measurement (arrow) on the top section partially localized the top edge. For clarity in the image, the belief of the other sections are not shown, and only 50 of the 500 particles are shown.
(c): A following measurement (arrow) on the right edge further localizes the part. This measurement provides information on the right edge directly, and the top edge indirectly.

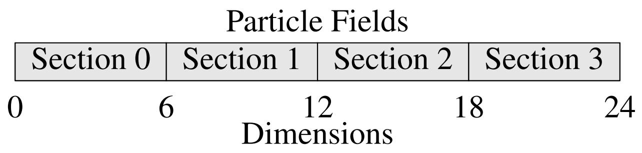


Figure 2.4: A 24-dimensional particle for object with 4 sections

Algorithm 1 Full-State Particle Filter

```
Input: number of particles $N$ and number of sections $n$
Input: particles $\mathbf{X_t}$
Input: observation $m_t^i$
Input: meshes $S = \{S_k\}_{k=1}^n$
Output: particles $\mathbf{X_{t+1}} = \{\mathbf{{}^jx_{t+1}}\}_{j=1}^N$
 1: build distance field $D_f(p)$
 2: $j \leftarrow \mathbf{1}$
 3: while $j \leq N$ do
 4:     $\mathbf{x} \sim p(x|\mathbf{X_t})$                    ▷ $x = \{x^k\}_{k=1}^n$
 5:     $dist \leftarrow D_f(T(x^i)^{-1} m_t^i)$
 6:     if $dist \leq \xi$ then
 7:         ${}^j x_{t+1} \leftarrow x$
 8:         $j \leftarrow j + 1$
 9:     end if
10: end while
```

achieved by applying kernel density estimation techniques. One popular method for the bandwidth selection is the Silverman's rule-of-thumb estimator[**?**]. Other techniques are discussed in more detail in [**?**].

Although it seems at first that the high dimensional state space will require a prohibitively huge number of particles to approximate the true distribution, in practice, this has been found to not the case. Largely, this is due to the small internal tolerances compared to the uncertainty of the pose of the object as a whole, so particles tend to cluster in a small subset of the full state space. Note that if there were no uncertainty in the transformation between sections, then the system reduces to a single 6 dimensional state space. In addition, the Gaussian kernel applied to the particles is capable of creating broad beliefs in specific dimensions[**?**], thus this particle filter is able to model the mixture of precisely localized, and poorly localized dimensions. Finally, the sample acceptance probability, $p(z_t|x_t)$, depends only on the 6 dimensional subspace corresponding to the pose of the section being touched, and although only a thin manifold is accepted with high probability, the particle starvation challenge is no worse than previously addressed in 2.2.

At any time step t the belief of the state $bel(\mathbf{x_t}) = p(\mathbf{x_t}|Z_t, x_0)$, and x_t^k is the portion of x_t representing the pose of section k. During rejection sampling, new full-state samples x_{t+1} are drawn from the estimated $\hat{bel}(\mathbf{x_t})$ directly. The probability of accepting a sample is based on the measurement M_t^k on section k, and is computed by extracting the 6D pose x_{t+1}^k from each full-

Algorithm 2 Independent-State Particle Filter

```
Input: number of particles $N$ and number of sections $n$
Input: sets of particles $\mathbf{X_t} = \{X_t^k\}_{k=1}^n$
Input: observation $m_t^i$
Input: meshes $S = \{S_k\}_{k=1}^n$,
Input: transformations $\{p(T_i^k)\}_{k=1}^n$
Output: particles $\mathbf{X_{t+1}} = \{X_{t+1}^k\}_{k=1}^n = \{{}^j x_{t+1}^k\}$
 1: build distance field $D_f(p)$ for section $S_i$
 2: for $k = 1, ..., n$ do
 3:     $j \leftarrow 1$
 4:     while $j \le N$ do
 5:         $x \sim p(x^k | X_t^k)$
 6:         $T_i^k \sim p(T_i^k)$
 7:         $\tilde{x} \leftarrow T_i^k \times x$
 8:         $dist \leftarrow D_f(T(\tilde{x})^{-1} M_t^i)$
 9:         if $dist \le \xi$ then
10:             ${}^j x_t^k \leftarrow x$
11:             $j \leftarrow j + 1$
12:         end if
13:     end while
14: end for
```

state sample x_{t+1}. Equation **??** and **??** are then used for the rejection sampling, except that the transform from the extracted x_{t+1}^k is used to transform the measurement. If $dist(M_t^k, S_k(x_{t+1}^k))$ is sufficiently large, this full-state sample $\mathbf{x_{t+1}}$ is rejected, otherwise accepted. Note that the full-state sample $\mathbf{x_{t+1}}$ is accepted based on $p(M_t^k | x_{t+1}^k)$, and the CAD model used for distance field k is the mesh for that particular section, S_k (shown in Algorithm 1). For simplicity, the pseudo-code does not include the adaptively adjusted sample size based on KL-divergence[**?**][**?**].

2.3.4 Independent-State Representation

The first approach described above tracks the updates of both the pose of each datum and the their transformations by maintaining a full-state representation of the distribution. An alternative is to maintain the probability distribution for each section separately. Instead of using a full high-dimensional particle filter for the full object, individual 6-dimensional particle filters are used for each individual section under the approximation that the belief over transformations between sections are fixed and independent. While this loses information compared with the full joint belief, in practice this loss is acceptable.

As in the rigid body particle filter described in 2.2, a sample in the particle filter for section k represents a $SE(3)$ pose of the geometry of section k. The transformation information between different sections are defined explicitly. The prior belief on the transformation from section k to section j is $bel(T_k^j)$, which is a distribution over $SE(3)$ transformations. A measurement on a single section updates all individual particle filters related through a defined transformation distribution. Given a measurement M^k on the section k, the updated belief becomes $p(x^j_{t+1}|T^j_k, M^k)$ for a related section j.

The update to the belief $bel(x^k_t)$ given a measurement performed on section k itself is identical to the particle filter update for the rigid object. Since T^k_k is the identity with probability 1, the updated belief can be written as:

$$bel(x^k) = p(x^k|T^k_k, M^k) = p(x^k|M^k) \tag{2.9}$$

For a section j that references section k ($k \neq j$), each new sample x^j_{t+1} is drawn from the prior of its corresponding particle filter j. x^j_{t+1} is then transformed from the frame of section j to the frame of section k:

$$\tilde{x}^k = {}^iT^k_j \times x^j \tag{2.10}$$

where ${}^iT^k_j \sim bel(T^k_j)$ is a sampled transformation from the distribution $bel(T^k_j)$. As the measurement was performed on section k, the geometry of section k is used rather than j when computing the consistency with the measurement. The sampled particle is accepted with probability $p(M_t|\tilde{x}^k)$. The above process is repeated until the desired number of particles have been accepted (shown in Algorithm 2).

2.4 Predicting Effective Measurement Actions

Performing measurements is expensive, so it is important to choose the measurement action that provides the most *information gain* on the goal feature. Each action is treated as a probabilistic decision over a set of particles approximating the belief of the goal feature. This is an approximation for the information gain for the underlying, continuous belief distribution. The best measurement may not be on the goal feature, and it may be impossible to even measure the goal feature directly. This formulation predicts the information gain on the goal feature for both a measurement directly on the goal feature, or indirectly for a measurement on datums or other sections of the part.

2.4.1 Information Gain

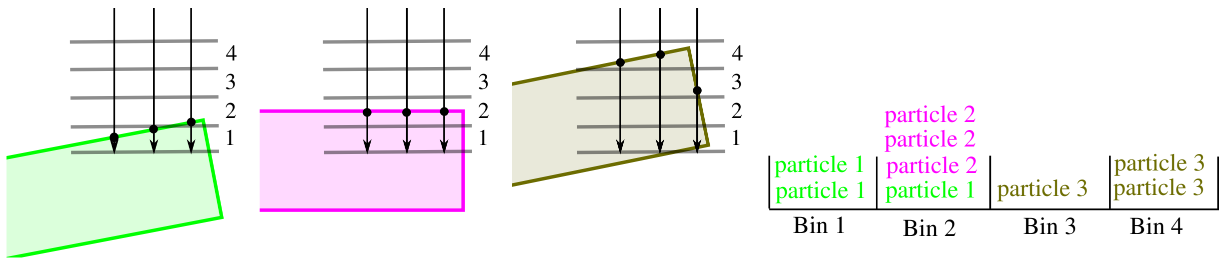
Given X^G, a set of particles representing the belief of the goal feature, the *information gain* from a measurement action $\mathcal{M}$ is defined as the expected reduction of entropy.

$$IG(X^G|\mathcal{M}) = H(X^G) - H(X^G|\mathcal{M}) \tag{2.11}$$

$H(X^G)$ is the entropy of the particles and $H(X^G|\mathcal{M})$ is the entropy of the particles conditioned on the measurement action.

The entropy of a discrete distribution of states depends only on the probabilities of each state occurring.

$$H(X^G) = -\sum_i w_i \log w_i \tag{2.12}$$



(a) Part in configuration: particle 1 (b) Part in configuration: particle 2 (c) Part in configuration: particle 3 (d) Binned Measurements

Figure 2.5: Binning of potential measurement on particles for use in calculating information gain. The three arrows represent the nominal measurement action $\mathcal{M}$ with simulated deviation δ_j. The horizontal lines divide the measurement values into the numbered bins.

where w_i is the weight of particle i.

To calculate the conditional entropy, $H(X^G|\mathcal{M})$, the measurement action $\mathcal{M}$ is simulated on the part distribution. Performing a measurement action yields a continuous distribution of measurement value. $\mathcal{W}$ samples are drawn from this distribution for each particle:

$$m_{i,j} = Simulate(\{\mathcal{M} + \delta_j\},{}^i X^G) + \eta_j \tag{2.13}$$

$$j \in \{1, 2, \ldots, \mathcal{W}\} \tag{2.14}$$

$$i \in \{1, 2, \ldots, N\} \tag{2.15}$$

where δ_j is the deviation from the nominal measurement action, $Simulate$ computes the value for a measurement action applied to the part in a specific configuration, and with η_j as measurement noise.

Measurement Simulation

To predict the measurement value obtained from a measurement action: $p(m_j)$, the measurement uncertainty is again approximated by a discrete sampling of the continuous distribution. For a single measurement action, and for each particle a sampling of measurement values is drawn from the distribution of measurement values that would be obtained if that particle was the true state.

The sensor measurement will indicate a distance z traveled along the measurement action $\mathcal{A}$ until reaching the part. We start by examining the distance from the start point along the vector until the first intersection with the part. Though a crude approximation of the true measurement value, the benefit of this model is that given a measurement action and part pose, the measurement value can be calculated as the intersection of a ray and a triangular mesh. Due to their heavy use in computer graphics, ray-mesh intersection algorithms have been heavily optimized and can be computed in parallel.

Measurement Width: While a ray is infinitely thin, the touch probe's spherical tip has a non-zero diameter, and thus will cast a cylinder rather than a ray. The true value returned by our sensor is the smallest distance until any contact with the part. Ray casting approximates the measurement cylinder by discrete uniformly spaced rays on the cylinder exterior, with the measurement value as the lowest ray-mesh intersection distance from this set.

Measurement Error: Error is caused both by inaccurate start positions and orientations due to robot positioning error, as well as inaccuracies in the sensor. In the most extreme cases error may cause a measurement to move from barely hitting an edge to completely missing the part. Thus it is clear neither adding a constant error term, nor a dependent Gaussian error will accurately model the error.

Instead discrete general method models this error. For each measurement action we make many simulated measurements where we perturb the initial conditions according to an error model for the robot and perturb the measured value according to a model of the sensor. Because our ray-mesh intersection method is cheap, the additional cost these extra simulations add is acceptable.

Bins

$H(X^G|\mathcal{M})$ is calculated by dividing the continuous values $m_{i,j}$ into discrete bins, b_k. The conditional entropy of this measurement action is then:

$$H(X^G|\mathcal{M}) = \sum_k p(b_k)\ H(X^G|b_k) \tag{2.16}$$

where $p(b_k)$ is the prior probability that this measurement will fall into bin b_k and $H(X|b_k)$ is the entropy of the particles within bin b_k. The likelihood of a bin is computed by summing the weights of the measurements in that bin. Defining the weight of the bin, W_k as:

$$W_k = \sum_{i,j} \mathbb{1}(m_{i,j} \in b_k) \cdot w_i \tag{2.17}$$

then:

$$p(b_k) = \frac{W_k}{\sum_{i,j} w_i} \tag{2.18}$$

$$= \frac{W_k}{\mathcal{W}} \tag{2.19}$$

Given a bin, the probability of a specific particle is:

$$p({}^iX|b_k) = \frac{\sum_j \mathbb{1}(m_{i,j} \in b_k) \cdot w_i}{W_k} \tag{2.20}$$

Then the entropy of the bin can be calculated:

$$H(X|b_k) = -\sum_i p({}^iX|b_k) \log(p({}^iX|b_k)) \tag{2.21}$$

Figure 2.5 visualizes this binning process. Three measurement actions (arrows) $\mathcal{M} + \delta_j$ are simulated on three configurations of the part (${}^ix$ = particle i). The intersection between

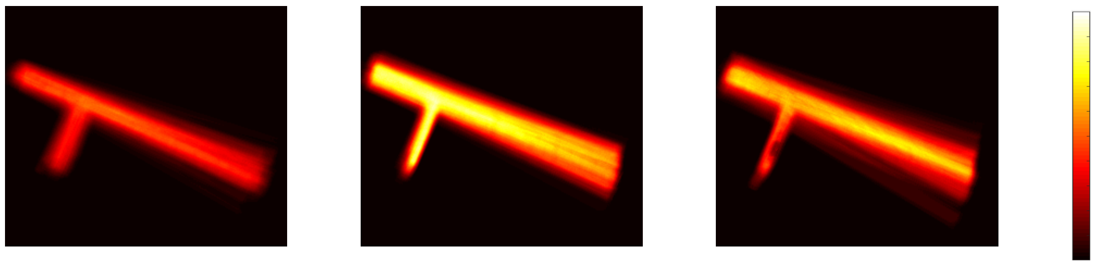

(a) Initial Belief (b) 1 Measurement (c) 2 Measurements

Figure 2.6: Heat maps showing information gain

the simulated measurement action and the part determines the measurement value $m_{i,j}$. These measurement values are sorted into bins.

Figure 2.6 visualizes the densely computed information gain for a grid of parallel rays as the "temperature" in a heat map. Initially **(a)** probing into the page is likely to miss the part, so information gain is low for all these measurements and we instead probe sideways. After this first measurement **(b)** the uncertainty is reduced, and probing into the page is more likely to hit the part and provide information. Once this second measurement is performed **(c)**, another measurement in the same place will not provide more information.

Adaptations are made to this process to accommodate the two representations as described next:

2.4.2 Information for Full-State Representation

The full-state representation does not maintain a set of particles over just the goal feature, but rather each particle, X, represents the full $6 \times n$ state. Using these full-state particles for X^G above provides a good metric for localizing every section of the part, but a poor metric for localizing the goal feature. This metric would often suggest to perform measurements on non-datum features that are irrelevant to the location of the goal feature. The error in this metric is due to

$$H(X^G|\mathcal{M}) \neq H(X|\mathcal{M}) \tag{2.22}$$

One approach is to incorporate domain knowledge when designing the full-state representation, by including only the relevant datums necessary for a particular task. Then, any information on this limited full state will be reduction of uncertainty of at least one datum required for the task.

An approach that does not require pruning irrelevant sections from the full state involves combining particles that produce similar configuration for the goal feature. To calculate $H(X^G|\mathcal{M})$ using particles X, measurement actions are used to sort the particles into bins as in described in section 2.4.1. Then, particles that produce sufficiently similar goal feature configurations are treated as identical particles when computing entropy, by combining these into groups L.

$$p({}^{L}X|b_k) = \frac{\sum_{i,j} \mathbb{1}(m_{i,j} \in b_k)\mathbb{1}(i \in L) \cdot w_i}{W_k} \tag{2.23}$$

$$H(X|b_k) = -\sum_{L} p({}^{L}X|b_k) \log(p({}^{L}X|b_k)) \tag{2.24}$$

These group can be constructed by discretizing the space of possible configurations for the goal feature. An issue which this paper does not address is the balance of a requiring a reasonably small number of particles while maintaining sufficient density for this descretization of goal feature configuration to produce meaningful group sizes.

2.4.3 Information for Independent-State Representation

The independent-state representation does maintain the set of goal feature particles, X^G, however additional steps are needed when computing Eq. 2.13. When simulating $\mathcal{M}$, the robot will measure some section, $\mathcal{S}$, of the part. Simulating the measurement using $X^{\mathcal{S}}$ is straightforward, but leads to computing $IG(X^{\mathcal{S}}|\mathcal{M})$, which is not the desired metric $IG(X^G|\mathcal{M})$.

The independent-state representation makes the approximation that the distribution of transformations between sections are fixed and independent, and this approximation is used to achieve the desired metric. A temporary set of particles $\tilde{X}^{\mathcal{S}}$ is created by sampling transforms $T_G^{\mathcal{S}}$ and applying these transforms to X^G. $\tilde{X}^{\mathcal{S}}$ is used in Eq. 2.13 to generate sample measurement values $\tilde{m}_{i,j}$, which are used in the calculation for bin entropy $H(X|b_k)$. While the independence approximation could be used again in the calculation of bin probabilities $p(b_k)$, this approximation is not needed. Measurements $m_{i,j}$ calculated using $X^{\mathcal{S}}$ are used to calculated $p(b_k)$.

2.5 Experiments

The following experiments validate both the full-state representation approach and independent-state representation approach in simulation. The software was implemented in ROS using C++. These experiment simulate a specific task which is common in manufacturing: localizing a target location, defined by datums, to drill a hole on an object. The datum-based particle filter was simulated on a structural component used in aircraft. This is the same object as used in the original rigid-body particle filter paper[**?**], with adaptations made to allow internal degrees of freedom. The object is composed of 5 precisely manufactured sections, and tolerance between sections was determined by engineering drawings (for precisely defined features), and educated guesses (for loosely defined relationships).

2.5.1 Measurement Selection

The target hole is localized by measuring its referenced datums. Specifically, the pose of the hole feature (green cylinder) in figure 2.1 is defined by an offset distance from the top and right sections shown in figure 2.3a, and the axis of the hole is orthogonal to the front plane. The hole does not exist yet, and thus cannot be measured directly. In order to localize the hole location precisely without direct measurement on the hole, it is assumed that the transformations between the target location and its defining datum sections have very small uncertainty along some dimensions, e.g. the vertical distance between the center of the hole and the top plane.

At each step, the measurement is simulated using ray-mesh intersection algorithms[**?**]. Potential measurement actions are sampled in the workspace. The information gain for each measurement is calculated based on the current estimated pose of each section. For each measurement performed, candidate actions are evaluated until 500 actions with non-zero information gain have been modeled. Only the measurement action with the largest expected information gain is "performed" in simulation and used to update the belief.

2.5.2 Simulation Results

Both proposed approaches are compared in similar settings. A total of 20 measurements are simulated during each trial. For the full-state representation, a maximum of 800 particles are used. For the independent-state representation, a maximum of 500 particles are used for each section. The simulation uses 5 mesh models for different sections. After each update, the average estimated pose of the hole is computed by averaging the hole poses produced from all particles.

Figure 2.7 shows the comparison between the full-state particle filter and independent-state particle filter. Translational error and rotational error are defined between the estimated pose of the hole and its true state. From the simulation, the errors of the estimated pose decreases rapidly for both approaches after each new measurement is applied on the system.

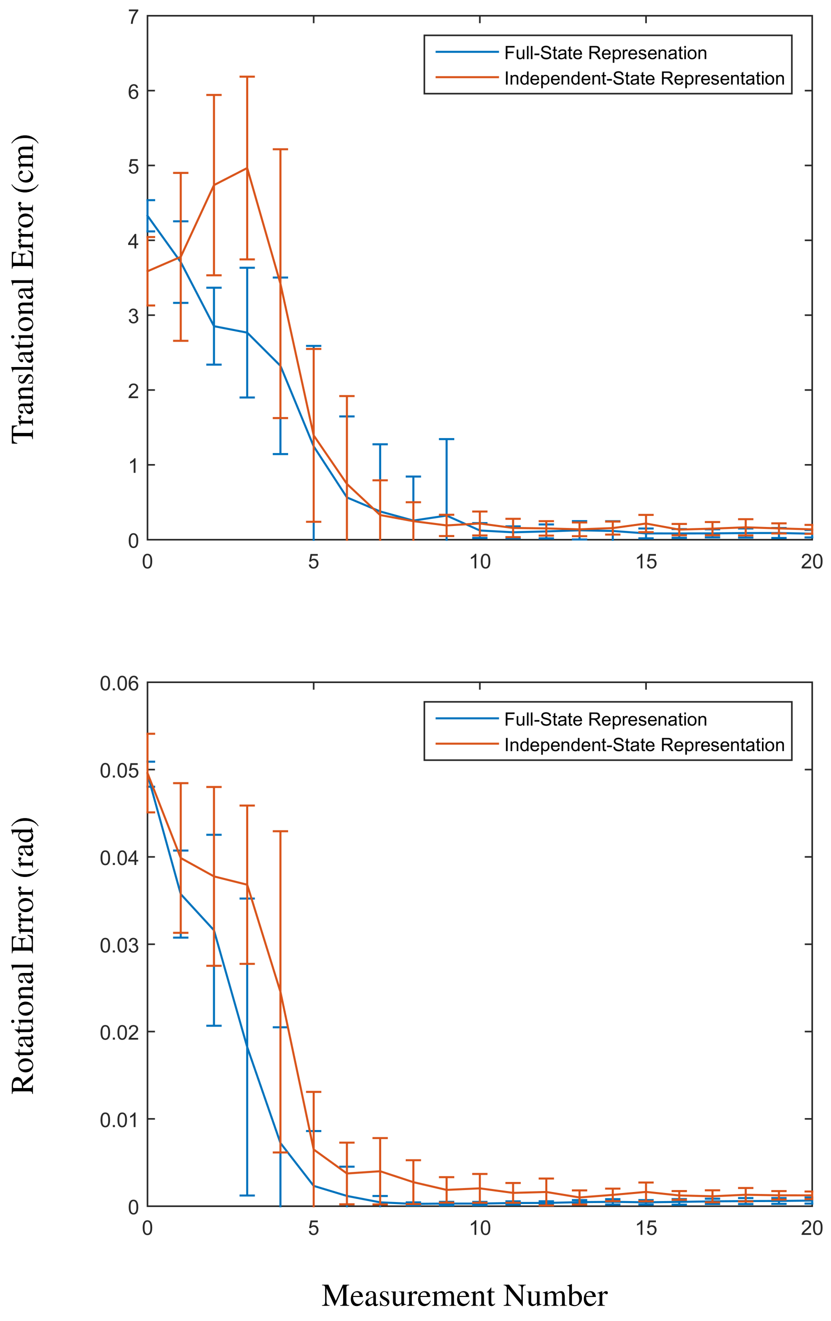


Figure 2.7: Comparison of the accuracy of the update step when using full-state particle filter and independent-state particle filter

Chapter 3

Planning with Contacts for Support

3.1 Contact Forces

Modeling contacts

3.1.1 Spring Model

3.1.2

3.2 Bracing

In path planning for robotic arms any contact with the environment is often considered a collision and thus valid paths have no interaction with the environment. However environmental contacts can reduce joint torques, damp vibrations, and stabilize an arm. Despite these benefits contacts are usually avoided because they bring algorithmic complexity during the planning stage and physical danger to the robot and environment if executed improperly. The increased planning complexity comes from the contact configurations being a measure-zero manifold in the robot configuration space, rendering naive planning in configuration space ineffective.

The benefits of bracing have been studied since the 1990s. Lew and Book proposed bracing a micro/macro manipulator with small precise arm mounted on the end of a long course arm and demonstrated bracing of the macro arm against multiple locations of the environment can damp vibration caused by the micro arm [**?**] [**?**]. Hollis and Hammer explored a similar micro/macro robot design and demonstrated $1\mu m$ accuracy, well over an order of improvement compared to their unbraced manipulator [**?**]. Both of these works did not address the planning problem as both contact locations and robot trajectories were manually specified.

Sampling based planners and Trajectory Optimization are two common approaches to path planning, however the measure-zero manifold of contact configurations poses problems to both planning techniques. Sampling based planners will never sample directly on this manifold, and extending a contact configuration to a new sampled point will immediately leave the contact manifold. Berenson developed a variation of an RRT planner capable of handling measure-zero manifolds by projecting invalid path extensions onto the valid configuration space [**?**]. The

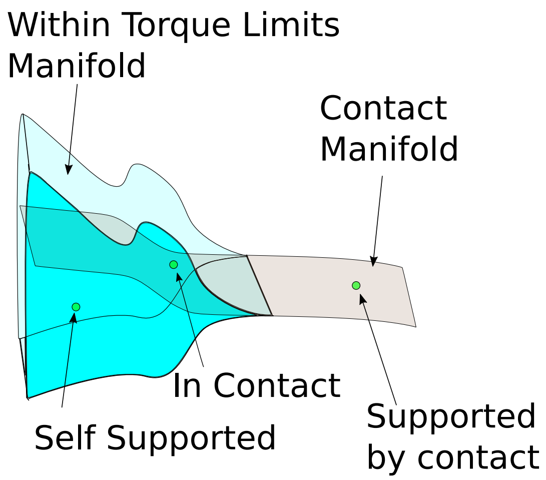


Figure 3.1: Illustration of valid configuration space for an arm potentially supported by contacts

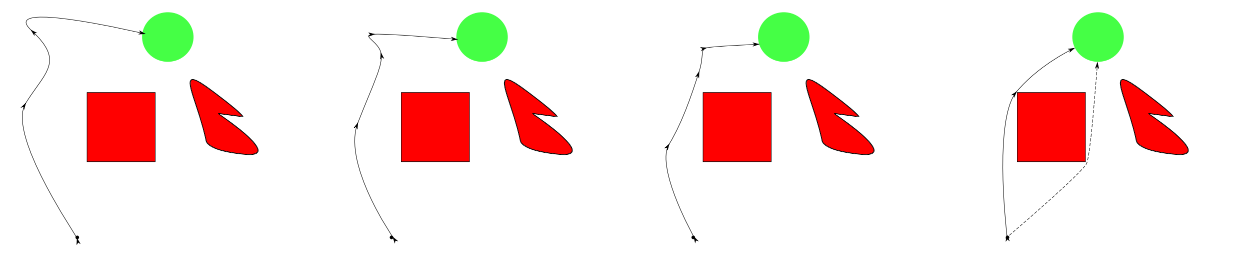


Figure 3.2: Optimizing the shortest path trajectory from an initial path converges to a local best, but possibly misses the globally best solution.

method of projection must be provided by the user.

Trajectory Optimization needs a smooth gradient of a cost function and in a naive implementation the benefits of contact will only be realized in a measure-zero domain surrounded by obstacle penetration and no contact regions. Softening contacts is commonly used in trajectory optimization to produce a gradient for obstacle avoidance, but produces local minima when used for attraction. Mordatch designed a cost function that continuously models both the benefit and cost of adding contacts and is able to produce trajectories which add and break contacts [**?**].

Rather than solving for a trajectory and contact locations simultaneously some approaches have separated these two problems. Given a sequence of contact modes for each link, Greenfield computed joint torques to produce desired dynamic behavior and applied this to a climbing snake robot [**?**]. Bretl et al. [**?**] developed algorithms for climbing robots that first select contact locations then create collision free trajectories for the robot's limbs between these contact locations. Tonneau et al [**?**] approaches from the opposite direction and first generates a trajectory then considers a discrete set of contacts close to that trajectory.

3.3 Discrete Selection of Contact Locations

3.3.1 Walking

3.3.2 Grasping

3.4 Trajectory Optimization

3.4.1 Methods for Finding the Minimum Cost Trajectory

3.4.2 Smoothing the Contacts to Discover Contacts

Trajectories for a robotic arm are created through trajectory optimization. The key approach used here is softening of contacts through the introduction of auxiliary variables which drastically smooth the cost function used at the expense of increasing the dimensionality of the state space. The formulation used in the paper follows from Mordatch's work [**?**].

A trajectory s is a series of robot states s_t at increasing points in time. Each state s_t contains the joint configuration θ_t as well as an axillary continuous variable c_t for each section of the robotic arm allowed to make contact with the environment. The variables c_t regulate the magnitude of the normal and frictional contact forces and will be describe in detail later in this document. At first these c variables seem unnecessary as the forward kinematics specified by the joint angles θ determine the locations of the robot, and therefore determine the contacts between the robot and the environment. However slight changes in these joint configurations result in slight motion of the robot which may result in huge changes in contact forces. Additionally without artificial terms in the cost function to model contact, the cost function is blind to potential contacts that may be close and useful.

Cost Function

The optimal solution s^* is computed by minimizing the cost function of the form

$$\begin{aligned} L(s) = {} & L_{ContactViolation} + L_u \\ & + L_{ObjectPenetration} + L_{Goal} \end{aligned}$$

$L_{ObjectPenetration}$ and L_{Goal} are straightforward, while the interplay between $L_{ContactViolation}$ and L_u provide the interesting structure allowing the optimization to find paths which use supporting contacts.

Cost $L_{ContactViolation}$

Each section of the robot able to make contact has an associated variable $c_t > 0$ at each time step. c intuitively represents the strength of the contact forces. Since if the robot is not physically in contact with the world there will not be a contact force the cost $L_{ContactViolation}$ is introduced to penalize non-zero values of c when the robot is not in contact.

$$L_{ContactViolation} = \sum_t \sum_i c_{t,i} d_{t,i}$$

Non-zero values for c are intentionally allowed when the robot is not in contact with the environment even though this is not physically realistic as this makes the cost function smooth. However when optimizing $c_{t,i}$ will tend towards 0 unless the robot is in contact.

Cost L_u

The cost L_u penalizes the input torque necessary to follow the trajectory specified. With no contacts these inputs could be calculated directly from the arm inverse dynamics. As discussed the physical contact forces are extremely sensitive to robot configuration, so to soften the forces we instead compute the contact forces that minimize the joint torque. When the trajectory is executed on the robot the joint controller will take responsibility for finding the slight adjustments in configuration to reach the desired contact forces. The path planner does not have to worry about the minute adjustments for a model that will not match reality to that detail anyway. Instead this path planner just needs to estimate the best contact forces possibly, according to the the following quadratic programming problem:

$$\begin{aligned} f, u = argmin_{\tilde{f},\tilde{u}} &||J^T\tilde{f} + \tilde{u} - \tau_{Free}|| \\ &+ \tilde{f}^T W\tilde{f} + \tilde{u}^T R\tilde{u} \end{aligned}$$

The input control regularization R is chosen based on the desired penalization of joint inputs. The contact force regularization W is dependant on the values of c, with

$$W_{j,j} = \frac{1}{c_{i,t}^2 + 1}$$

If c is large then the robot is in contact, the force regularization is small, so the contact force can be large. If c is small the robot is not near any contact location, the force regularization is large, thus the contact force is heavily penalized and will be small.

3.4.3 Other Cost Terms

$L_{Obstacle}$ penalizes penetration of the robot into the environment which is calculated using the robot forward kinematics and then collision detection.

L_{Goal} is a penalty on the last robot state s_T on the distance of the robot end effector from the goal location.

Auxillary Variables

3.5 Sample Based Planning

3.5.1 RRTs and their variants

3.5.2 Adaptation to Encourage Contacts

Given:

$X \in \mathbb{R}^d$:	d-dimensional configuration space
$X_{obs} \subset X$:	obstacles in the configuration space
$X_{free} = X \setminus X_{obs}$:	free space
$x_{start} \in X_{free}$:	starting configuration
$X_{goal} \subset X_{free}$:	set of goal configurations
$policy : X \times X \to TX$	policy to goal, maps to tangent space (velocity). Previous formulation was wrong, as r

The goal is to find a path in X_{free} from x_{start} to $x_{goal} \in X_{goal}$.

Algorithm 3 $T = (V, E) \leftarrow$ policyRRT(x_{start})

```
1: $T \leftarrow$ InitTree($x_{start}$)
2: **while** GoalNotReached($T, X_{goal}$) **do**
3:     $x_{rand} \in X \leftarrow$ Sample()
4:     $x_{nearest} \leftarrow$ Nearest($T, x_{rand}$)
5:     $T \leftarrow$ followPolicy($x_{nearest}, x_{rand}, T$, extensionLimit)
6:     **if** ExtensionSuccessful **then**
7:         $T \leftarrow$ followPolicy($x_{new}, x_{goal} \in X_{goal}, T, \infty$)
8:     **end if**
9: **end while**
```

Algorithm 4 $T = (V, E) \leftarrow$ followPolicy(x_{begin}, x_{end}, T, iterLimit)

```
1: $T_{new} \leftarrow$ InitTree($x_{begin}$)
2: $x_{prev} \leftarrow x_{begin}$
3: $x_{new} \leftarrow$ policy($x_{prev}, x_{end}$)
4: $i \leftarrow 0$ MovingTowardsGoal($x_{new}, T_{new}$) **and** [2] Nearest($T, x_{new}$) == $x_{begin}$ **and** [2] i $<$
   iterLimit **do**
5: $T_{new} \leftarrow$ AddNode($x_{new}, x_{prev}, T_{new}$)
6: $x_{prev} \leftarrow x_{new}$
7: $x_{new} \leftarrow$ policy($x_{prev}, x_{end}$)
8: $i \leftarrow i + 1$
9:
10: $T \leftarrow$ AddTreeToTree($T_{new}, T, x_{begin}$)
```

In many environments, it is relatively easy to compute a reactive policy that can make forwards progress towards a goal while avoiding collision with obstacles, such as a potential field approach, but such policies are prone to getting stuck in local minima/dead ends. PolicyRRT is a variant of RRT that seeks to combine the efficient, goal directed trajectory generation of reactive policies with the ability of RRT to find paths around local minima/dead ends. PolicyRRT fundamentally behaves like a standard RRT by growing a tree by iteratively extending towards a randomly sampled configuration from the nearest point in a search tree T. PolicyRRT uses the reactive policy to move towards the random sample, potentially enabling the planner to avoid collisions with obstacles during the extension. When PolicyRRT is successful in extending towards the sample (does not collide with an obstacle and does not get stuck in a local minima) it then follows the policy to extend towards the goal. To avoid oversampling regions associated with local minima, when PolicyRRT extends towards a point from some vertex $v \in T$ PolicyRRT halts extension when the trajectory leaves the Voronoi cell of v. Extension is also halted once the policy ceases to make progress (reaches a local minima) or exceeds a maximum number of iterations (only when extending towards x_{rand}?)

PolicyRRT is beneficial when a policy that would yield a path all the way from x_{start} and x_{goal} is difficult to find but finding a policy that can make some progress while avoiding obstacles, but may get stuck in local minima is relatively easy. An example of such a policy is a **potential field**: a cost function is constructed by placing attractors at the goals and repulsions at obstacles and the policy is to the gradient of this cost. Potential fields are generally much simpler to compute than the minima-free navigation functions, but cannot be guaranteed to produce a path. Augmenting the gradient descent policy with an RRT can help find paths out of any local minima.

Similarly augmenting an RRT with a reactive policy can assist in navigating narrow passages. Because the policy will naturally flow through corridors PolicyRRT no longer requires drawing samples inside the narrow corridor itself, as drawing a sample from the far side may suffice to draw the policy through the corridor.

3.6 Experiments

3.6.1 Simulation and Robot Model

3.6.2 Robot Snakes on a Plane

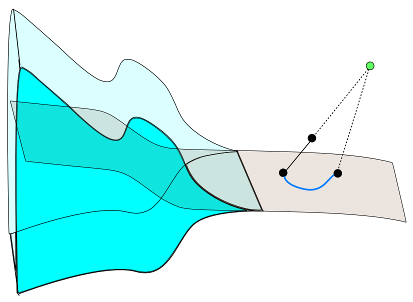


Figure 3.3: Illustration of Gradient RRT extension on the contact manifold

Chapter 4

Conclusion and Future Work

This paper introduced the *datum-based particle filter*, which provides a method to localize a task location defined by datums on an object with internal degrees of freedom. This method stores the belief as the belief over the poses of multiple rigid sections comprising the object using a particle filter, and selects measurement actions using the metric of information gain. Two implementations are described: a high dimensional particle filter capturing the full state, and multiple particle filters coupled through the tolerances between sections. The techniques to avoid particle starvation during rigid body localization have been extended to both implementations of the datum-based particle filter. Information gain of a potential measurement action is approximated as a discrete probabilistic decision process over the particles comprising the belief. The formulation presented distinguishes between useful information which updates the belief of the target feature, and information which only improves the belief of non-datum sections.